This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2582562, IEEE Transactions on Information Forensics and Security

Scaling up Class-Specific Kernel Discriminant Analysis for large-scale Face Verification

Alexandros Iosifidis, *Senior Member, IEEE*, and Moncef Gabbouj, *Fellow, IEEE*

***Abstract*—In this paper, a novel approximate solution of the criterion used in non-linear class-specific discriminant subspace learning is proposed. We build on the Class-Specific Kernel Spectral Regression method which is a two-step process formed by an eigenanalysis step and a kernel regression step. Based on the structure of the intra-class and out-of-class scatter matrices, we provide a fast solution for the first step. For the second step, we propose the use of approximate kernel space definitions. We analytically show that the adoption of randomized and class-specific kernels have the effect of regularization and Nyström-based approximation, respectively. We evaluate the proposed approach in face verification problems and compare it with existing approaches. Experimental results show the effectiveness and efficiency of the proposed Approximate Class-Specific Kernel Spectral Regression method, since it can provide satisfactory performance and scale well with the size of the data.**

***Index Terms*—Class-Specific Kernel Discriminant Analysis, Approximation, Nonlinear Subspace Learning, Kernel Regression.**

I. Introduction

Face recognition and verification problems have attracted the attention of the research community for more than two decades due to their importance in many real-world applications, including security, human-computer interaction and human behaviour analysis for assisted living [1], [2], [3], [4]. The problems of face recognition and face verification are conceptually different. On the one hand, face recognition is a multi-class problem where the objective is to recognize the identity of a person (through the analysis of a facial image depicting him/her) from a pool of K known person identities. Face verification, on the other hand, is a binary problem where the objective is to verify whether a facial image depicts a person of interest (i.e. the client). In the latter case, the number of persons belonging to the impostor class is unknown.

One research track that has been shown to achieve excellent performance in these two problems exploits the power of *subspace learning techniques*. The most well-known and commonly employed subspace learning technique is Principal Component Analysis (PCA) [5]. PCA defines a subspace for data projection that preserves most of the available information, in the sense of minimal l_2 norm-based reconstruction error. However, being an unsupervised technique, PCA does not exploit discriminative information encoded in the class labels of the training data and, thus, its discrimination power is limited. Linear Discriminant Analysis (LDA) [5], [6], [7], [8], [9], [10] is perhaps the most well studied discriminant subspace learning method finding applications in many multi-class problems, including face, facial expression, person and activity recognition [11], [12], [13], [14], [15], [16], [17], [18] among others.

A property of LDA that limits its application in verification problems is the fact that the maximal dimensionality of the learnt subspace is restricted by the number of classes K forming the problem at hand, since the rank of the between-class scatter matrix is at most equal to $K-1$. A direct consequence of this is that in binary (two-class) problems, the subspace learned by LDA is formed by only one dimension, which might not be the optimal choice for class discrimination. In order to overcome this limitation of LDA, class-specific approaches have been proposed [19], [20], [21], [22], [23]. Class-specific subspace learning techniques determine 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). Moreover, by appropriately defining the intra-class and out-of-class scatter matrices, the maximal dimensionality of the learnt subspace is restricted by the number of client samples. This in turn leads to better class discrimination and better performance [20], [23].

In order to determine nonlinear data projections (which have been found to outperform linear ones with a large extend in face recognition and verification problems), both multi-class and class-specific subspace learning techniques can be extended to their nonlinear counterparts by exploiting the well-known *kernel trick* [24], [25], [26], [27]. Kernel versions of subspace learning techniques require the calculation of the kernel matrix $\mathbf{K} \in \mathbb{R}^{N \times N}$, where N is the training set cardinality, expressing pairwise dot-products of the training data representations in the kernel space, and the eigen-decomposition of the corresponding scatter matrices which are expressed as functions of $\mathbf{K}$. For large-scale problems the application of standard kernel versions of subspace learning techniques becomes computationally intractable. This is due to the fact that the time complexity of the required eigen-decomposition step is of the order of $O(N^3)$ [23]. Recently, several algorithms of reduced computational complexity for nonlinear versions of multi-class discriminant analysis techniques have been proposed, see e.g. [28], [29]. However, only a few recent studies focused on reducing the complexity of nonlinear class-specific discriminant learning, e.g. [30].

In this paper, we revisit the Class-Specific Kernel Spectral Regression (CS-KSR) method [22], [23], which is a class-specific extension of the Kernel Spectral Regression proposed for multi-class discriminant learning [28]. CS-KSR approaches

A. Iosifidis and M. Gabbouj are with the Department of Signal Processing, Tampere University of Technology, P. O. Box 553, FIN-33720 Tampere, Finland. e-mail: {alexandros.iosifidis,moncef.gabbouj}@tut.fi

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2582562, IEEE Transactions on Information Forensics and Security

the class-specific nonlinear subspace learning problem by applying a two-step process, i.e. i) a generalized eigenanalysis process applied on the in-class and out-of-class scatter matrices expressed in the kernel space, and ii) a kernel regression process. This approach has been shown to provide similar (or even better) performance when compared to the orignal Class-Specific Kernel Discriminant Analysis (CS-KDA) approach. We show that, based on the structure of the intra-class and out-of-class scatter matrices, the first computation step is trivial and can be solved by applying a much simpler and faster process involving only the class labels of the training data. Subsequently, we employ three approximate kernel space definitions, i.e. prototype-based kernels [31], randomized kernels [32] and class-specific kernels [30], in order to further speed up the overall process. The result of these two processing steps forms the proposed Approximate Class-Specific Kernel Spectral Regression (ACS-KSR) method. Moreover, we analytically show that the exploitation of a randomized kernel has the effect of regularization in the overall solution and that the exploitation of a class-specific kernel is directly connected with Nyström-based kernel approximation, where the basis of the approximation is determined by the samples forming the client class.

We evaluate the proposed method in medium- and large-scale face verification problems, where it is shown that it can scale well with respect to the cardinality of the training set, while achieving satisfactory performance. Comparisons with related methods show the effectiveness and efficiency of the proposed technique.

The main contributions of the paper are:

- The exploitation of intra-class and out-of-class scatter matrices structure in order to speed up the CS-KSR optimization process.
- The exploitation of approximate kernel space definitions in order to further speed up the CS-KSR optimization process.
- An analysis showing that the exploitation of randomized kernel has the effect of regularization in the CS-KSR solution.
- An analysis showing that the exploitation of class-specific kernel is directly connected with Nyström-based approximation for kernel-based class-specific nonlinear subspace learning.
- The proposal of a novel approximate solution of the criterion used for class-specific nonlinear subspace learning which is able to scale well with the size of the training set.

The remainder of the paper is structured as follows. In Section II we briefly describe the face verification problem and introduce notations that will be used throughout the rest of the paper. The criterion used for class-specific subspace learning and the solutions given by CS-KDA and CS-KSR are described in Section III. Our analysis is described in detail in Section IV. Experiments conducted in order to illustrate the efficiency and effectiveness of the proposed ACS-KSR method are provided in Section V. Finally, conclusions are drawn in Section VI.

II. Problem Statement

Let us denote by $\mathcal{U}$ a database formed by N facial images. Let us assume that C of the facial images belong to the client class, i.e. they depict the person of interest, while the remaining $I = N - C$ facial images belong to the impostor class. Let us also assume that all facial images in $\mathcal{U}$ have been pre-processed and represented by facial vectors $\mathbf{x}_i \in \mathbb{R}^D$, $i = 1, \ldots, N$. Facial vectors are followed by binary labels $l_i \in \{+1, -1\}$ denoting whether the facial vector $\mathbf{x}_i$ belongs to the client $(+1)$ or impostor (-1) class. Given the client and impostor facial vectors, a verification system learns a class-specific model for the client person.

In the test phase, a new facial image (not included in the training set) is introduced to the verification system and the system should decide whether this facial image depicts the client person, or not. This is achieved by introducing the facial image representation to the learnt model and verifying whether it fulfils some criteria, like a bounded Euclidean distance from the mean client facial image vector in the discriminant subspace.

III. Theoretical Background

In order to nonlinearly map $\mathbf{x}_i \in \mathbb{R}^D$ to its d-dimensional image $\mathbf{y}_i \in \mathbb{R}^d$ we follow the standard kernel approach. That is, we map the input space $\mathbb{R}^D$ to the kernel space $\mathcal{F}$ using a nonlinear function $\phi(\cdot)$, i.e. $\phi(\mathbf{x}_i) \in \mathcal{F}$. In $\mathcal{F}$, we determine a linear projection:

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

where $\mathbf{W} \in \mathbb{R}^{|\mathcal{F}| \times d}$. In practice, since data representations $\phi(\mathbf{x}_i)$ in $\mathcal{F}$ cannot be directly computed, the so-called kernel tick is exploited [24], [25], [27]. That is, the multiplication in (1) is inherently computed using dot products in $\mathcal{F}$. To do so, the so-called kernel function $\kappa(\cdot, \cdot)$ expressing dot products between training data in $\mathcal{F}$, i.e. $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$, is employed. Dot products between all training vectors are stored to the so-called kernel matrix $\mathbf{K} \in \mathbb{R}^{N \times N}$. Each column of $\mathbf{K}$ contains dot products between a training vector and the entire training set, i.e. $\mathbf{k}_i = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$, $j = 1, \ldots, N$. Let us denote by $\mathbf{\Phi} = [\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_N)] \in \mathbb{R}^{|\mathcal{F}| \times N}$ a matrix containing the data representations in $\mathcal{F}$. Then, the kernel matrix can be defined as $\mathbf{K} = \mathbf{\Phi}^T \mathbf{\Phi}$.

By exploiting the Representer Theorem, the data projection matrix $\mathbf{W}$ can be expressed as a linear combination of the training data in $\mathcal{F}$, i.e.:

$$\mathbf{W} = \sum_{i=1}^{N} \phi(\mathbf{x}_i) \mathbf{a}_i^T = \mathbf{\Phi} \mathbf{A}, \tag{2}$$

where $\mathbf{A} \in \mathbb{R}^{N \times d}$. By using (2), the data representations in $\mathbb{R}^d$ are given by:

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

In class-specific discriminant subspace learning [20], the objective is to determine data representations $\mathbf{y}_i \in \mathbb{R}^d$ in a feature space where the client class is as compact as possible, while the impostor class is spread far away as much as possible

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2582562, IEEE Transactions on Information Forensics and Security

from the client class. This can be mathematically expressed as a maximization problem optimizing the following criterion:

$$\mathcal{J}(\mathbf{W}) = \frac{D_I}{D_C}, \tag{4}$$

where D_I is the distance of the impostor vectors from the client class mean vector $\mathbf{m} = \frac{1}{C}\sum_{i,l_i=1}\mathbf{y}_i$ and D_C is the distance of the client vectors from $\mathbf{m}$. Using (1) and expressing the client mean vector as $\mathbf{m} = \mathbf{W}^T\mathbf{m}_\phi$, where $\mathbf{m}_\phi \in \mathbb{R}^{|\mathcal{F}|}$ is the client class mean expressed in $\mathcal{F}$, we have:

$$D_I = \sum_{i,l_i=-1} \|\mathbf{W}^T\phi(\mathbf{x}_i) - \mathbf{W}^T\mathbf{m}_\phi\|_2^2, \tag{5}$$

$$D_C = \sum_{i,l_i=1} \|\mathbf{W}^T\phi(\mathbf{x}_i) - \mathbf{W}^T\mathbf{m}_\phi\|_2^2. \tag{6}$$

For the calculation of $\mathbf{W}$, $\mathcal{J}(\mathbf{W})$ in (4) is expressed as:

$$\mathcal{J}(\mathbf{W}) = \frac{Tr\left(\mathbf{W}^T\mathbf{S}_I\mathbf{W}\right)}{Tr\left(\mathbf{W}^T\mathbf{S}_C\mathbf{W}\right)}, \tag{7}$$

where $Tr(\cdot)$ is the trace operator. $\mathbf{S}_I \in \mathbb{R}^{|\mathcal{F}|\times|\mathcal{F}|}$ and $\mathbf{S}_C \in \mathbb{R}^{|\mathcal{F}|\times|\mathcal{F}|}$ are the out-of-class and in-class scatter matrices in $\mathcal{F}$:

$$\mathbf{S}_I = \sum_{i,l_i=-1} \left(\phi(\mathbf{x}_i) - \mathbf{m}_\phi\right)\left(\phi(\mathbf{x}_i) - \mathbf{m}_\phi\right)^T \tag{8}$$

and

$$\mathbf{S}_C = \sum_{i,l_i=1} \left(\phi(\mathbf{x}_i) - \mathbf{m}_\phi\right)\left(\phi(\mathbf{x}_i) - \mathbf{m}_\phi\right)^T. \tag{9}$$

By substituting (2) in (8) and (9), (7) can be expressed as a function of $\mathbf{A}$ as follows:

$$\mathcal{J}(\mathbf{A}) = \frac{Tr\left(\mathbf{A}^T\mathbf{M}_I\mathbf{A}\right)}{Tr\left(\mathbf{A}^T\mathbf{M}_C\mathbf{A}\right)}, \tag{10}$$

where $\mathbf{M}_I \in \mathbb{R}^{N\times N}$ and $\mathbf{M}_C \in \mathbb{R}^{N\times N}$ are given by:

$$\begin{aligned}\mathbf{M}_I &= \mathbf{K}_I\mathbf{K}_I^T - \frac{1}{C}\mathbf{K}_I\mathbf{1}_I\mathbf{1}_I^T\mathbf{K}_C^T \\ &- \frac{1}{C}\mathbf{K}_C\mathbf{1}_C\mathbf{1}_I^T\mathbf{K}_I^T + \frac{1}{C^2}\mathbf{K}_C\mathbf{1}_C\mathbf{1}_C^T\mathbf{K}_C^T\end{aligned} \tag{11}$$

and

$$\mathbf{M}_C = \mathbf{K}_C\left(\mathbf{I} - \frac{1}{C}\mathbf{1}_C\mathbf{1}_C^T\right)\mathbf{K}_C^T, \tag{12}$$

where $\mathbf{1}_I \in \mathbb{R}^I$ and $\mathbf{1}_C \in \mathbb{R}^C$ are vectors of ones. $\mathbf{K}_I \in \mathbb{R}^{N\times I}$ and $\mathbf{K}_C \in \mathbb{R}^{N\times C}$ are matrices formed by the columns of $\mathbf{K}$ corresponding to the impostor and client data, respectively [23]. The solution of (10) is given by applying eigenanalysis to the matrix $\mathbf{M} = \mathbf{M}_C^{-1}\mathbf{M}_I$ [33]. Since the rank of $\mathbf{M}$ is at most $C-1$, the dimensionality of the learnt space d is restricted by the number of samples forming the positive class (or by the dimensionality of the input space D), i.e. $d \leq \min(C-1, D)$.

A Spectral Regression-based solution of (7) has been recently proposed in [22], [23]. Let us denote by $\mathbf{w}$ an eigenvector of the problem $\mathbf{S}_I\mathbf{w} = \lambda\mathbf{S}_C\mathbf{w}$ with eigenvalue λ. $\mathbf{w}$ can be expressed as a linear combination of the training data in $\mathcal{F}$, i.e., $\mathbf{w} = \mathbf{\Phi a}$. By setting $\mathbf{Ka} = \mathbf{t}$, this eigenanalysis problem can be transformed to the following equivalent problem $\mathbf{P}_I\mathbf{t} = \lambda\mathbf{P}_C\mathbf{t}$, where $\mathbf{P}_I = \mathbf{e}_I\mathbf{e}_I^T - \frac{1}{C}\mathbf{e}_I\mathbf{e}_C^T - \frac{1}{C}\mathbf{e}_C\mathbf{e}_I^T + \frac{1}{C^2}\mathbf{e}_C\mathbf{e}_C^T$ and $\mathbf{P}_C = (1 - \frac{2}{C} + \frac{1}{C^2})\mathbf{e}_C\mathbf{e}_C^T$. $\mathbf{e}_C \in \mathbb{R}^N$ is a vector having elements $e_{C,i} = 1$ if $l_i = 1$ and $e_{C,i} = 0$ if $l_i = -1$. $\mathbf{e}_I \in \mathbb{R}^N$ is a vector having elements $e_{I,i} = 1$ if $l_i = -1$ and $e_{I,i} = 0$ if $l_i = 1$.

The solution $\mathbf{P}_I\mathbf{t} = \lambda\mathbf{P}_C\mathbf{t}$ is obtained by applying eigenanalysis to the matrix $\mathbf{P} = \mathbf{P}_I^{-1}\mathbf{P}_C$. Since $\mathbf{P}_I$ is singular, the following regularized eigenanalysis problem is solved:

$$(\mathbf{P}_I + r\mathbf{I})\mathbf{t} = \lambda\mathbf{P}_C\mathbf{t} \tag{13}$$

and eigenanalysis is applied to the regularized matrix $\tilde{\mathbf{P}} = (\mathbf{P}_I + r\mathbf{I})^{-1}\mathbf{P}_C$, where r is a small positive value[1], is solved.

In CS-KSR, $\mathbf{A}$ is obtained by applying the following two-step process:

- Solution of the eigenproblem (13). This process leads to the determination of a matrix $\mathbf{T} = [\mathbf{t}_1, \ldots, \mathbf{t}_d]$, where $\mathbf{t}_k$ is the eigenvector corresponding to the k-th largest eigenvalue.
- Calculation of the matrix $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_d]$, where $\mathbf{Ka}_k = \mathbf{t}_k$. In the case where $\mathbf{K}$ is non-singular, the vectors $\mathbf{a}_k$ are given by $\mathbf{a}_k = \mathbf{K}^{-1}\mathbf{t}_k$. Otherwise, $\mathbf{a}_k$ can be calculated by solving the following set of linear equations:
$$(\mathbf{K} + \delta\mathbf{I})\mathbf{a}_k = \mathbf{t}_k, \tag{14}$$
where $\delta \geq 0$ is a regularization parameter. Thus, $\mathbf{a}_k$ is given by $\mathbf{a}_k = (\mathbf{K} + \delta\mathbf{I})^{-1}\mathbf{t}_k$.

IV. Proposed Approximate Class-Specific Kernel Spectral Regression method

In this Section, we provide our analysis and describe the proposed Approximate Class-Specific Kernel Spectral Regression (ACS-KSR) method. We start by providing an analysis of the eigenanalysis problem (13) in the following subsection. We show that an equivalent solution can be given by applying a much faster process. This has been shown in [22] for the case where the positive class is formed by only one sample, thus leading to a feature space of one dimension. We extend the analysis in [22] for the general case.

A. Eigenanalysis step

Here we show that by exploiting the structure of matrices $\mathbf{P}_I$ and $\mathbf{P}_C$, the solution of the eigenproblem (13) can be obtained by applying a much faster process. Based on the definition of matrices $\mathbf{P}_I$ and $\mathbf{P}_C$ we can observe that they are block matrices having the following form:

$$\mathbf{P}_C = \left[\begin{array}{c|c} (1 - \frac{2}{C} + \frac{1}{C^2})\mathbf{1}_C\mathbf{1}_C^T & \mathbf{0} \\ \hline \mathbf{0} & \mathbf{0} \end{array}\right] \tag{15}$$

and

$$\mathbf{P}_I + r\mathbf{I} = \left[\begin{array}{c|c} \frac{1}{C^2}\mathbf{1}_C\mathbf{1}_C^T + r\mathbf{I} & -\frac{1}{C}\mathbf{1}_C\mathbf{1}_I^T \\ \hline -\frac{1}{C}\mathbf{1}_I\mathbf{1}_C^T & \mathbf{1}_I\mathbf{1}_I^T + r\mathbf{I} \end{array}\right]. \tag{16}$$

[1] r is a regularization parameter usually taking values equal to 0.01 or 0.001.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2582562, IEEE Transactions on Information Forensics and Security

Let us define the following matrices:

$$\mathbf{P}_C^{11} = (1 - \frac{2}{C} + \frac{1}{C^2})\mathbf{1}_C\mathbf{1}_C^T \quad (17)$$

$$\mathbf{P}_I^{11} = \frac{1}{C^2}\mathbf{1}_C\mathbf{1}_C^T + r\mathbf{I} \quad (18)$$

$$\mathbf{P}_I^{12} = -\frac{1}{C}\mathbf{1}_C\mathbf{1}_I^T \quad (19)$$

$$\mathbf{P}_I^{21} = -\frac{1}{C}\mathbf{1}_I\mathbf{1}_C^T \quad (20)$$

$$\mathbf{P}_I^{22} = \mathbf{1}_I\mathbf{1}_I^T + r\mathbf{I}. \quad (21)$$

Using (18)-(21), the following matrices can be defined:

$$\mathbf{C}_1 = \mathbf{P}_I^{11} - \mathbf{P}_I^{12}(\mathbf{P}_I^{22})^{-1}\mathbf{P}_I^{21} \quad (22)$$

$$\mathbf{C}_2 = \mathbf{P}_I^{22} - \mathbf{P}_I^{21}(\mathbf{P}_I^{11})^{-1}\mathbf{P}_I^{12} \quad (23)$$

Then, we have:

$$(\mathbf{P}_I + r\mathbf{I})^{-1} = \left[\begin{array}{c|c} \mathbf{C}_1^{-1} & -(\mathbf{P}_I^{11})^{-1}\mathbf{P}_I^{12}\mathbf{C}_2^{-1} \\ \hline -\mathbf{C}_2^{-1}\mathbf{P}_I^{21}(\mathbf{P}_I^{11})^{-1} & \mathbf{C}_2^{-1} \end{array}\right] \quad (24)$$

and

$$\tilde{\mathbf{P}} = \left[\begin{array}{c|c} \mathbf{C}_1^{-1}\mathbf{P}_C^{11} & \mathbf{0} \\ \hline -\mathbf{C}_2^{-1}\mathbf{P}_I^{21}(\mathbf{P}_I^{11})^{-1}\mathbf{P}_C^{11} & \mathbf{0} \end{array}\right] = \left[\begin{array}{c|c} \tilde{\mathbf{P}}_1 & \mathbf{0} \\ \hline \tilde{\mathbf{P}}_2 & \mathbf{0} \end{array}\right]. \quad (25)$$

$\tilde{\mathbf{P}}$ is a block matrix and each of its non-zero blocks is a constant matrix, i.e. $\tilde{\mathbf{P}}_1 = p_1\mathbf{1}_C\mathbf{1}_C^T$ and $\tilde{\mathbf{P}}_2 = p_2\mathbf{1}_I\mathbf{1}_C^T$. Such a matrix has a dominant eigenvector (corresponding to the largest eigenvalue) of the form $\mathbf{t}_1 = [t_{1,C}\mathbf{1}_C^T, t_{1,I}\mathbf{1}_I^T]^T$, where $t_{1,C}$ and $t_{1,I}$ are scalars satisfying $Ct_{1,C}^2 + It_{1,I}^2 = 1$ [34]. The remaining eigenvectors can be determined by applying an orthogonalization technique, e.g. the GramSchmidt process. Here we should note that the case where $t_{1,C} = t_{1,I} = 1/\sqrt{N}$ (i.e. the vector $\frac{1}{\sqrt{N}}\mathbf{1}_N$) also corresponds to an eigenvector of $\tilde{\mathbf{P}}$, but it is useless since it maps all the training data to the same point. Therefore, we pick the vector $\frac{1}{N}\mathbf{1}_N$ as our first eigenvector, we use an orthogonalization technique in order to define the remaining eigenvectors and then we remove it. This process is algorithmically illustrated in Pseudocode 1. As can be seen, this process employs only the training labels l_i, $i = 1, \ldots, N$ and, thus, it is independent of the training data.

Pseudocode 1: Calculation of $\mathbf{T}$

procedure $\mathbf{T}$ = TARGETS_CALCULATION($\mathbf{l}$, d)

$N = length(\mathbf{l})$;
$\mathbf{T} = rand(2, d+1)$; $\mathbf{Z} = zeros(N, d+1)$;
$\mathbf{f}_1 = find(\mathbf{l} == 1)$; $\mathbf{f}_2 = find(\mathbf{l} == -1)$;
$\mathbf{Z}(\mathbf{f}_1, :) = repmat(T(1, :), length(\mathbf{f}_1), 1)$;
$\mathbf{Z}(\mathbf{f}_2, :) = repmat(T(2, :), length(\mathbf{f}_2), 1)$;
$\mathbf{Z}(:, 1) = ones(N, 1)/\sqrt{N}$;
$\mathbf{Q} = qr(\mathbf{Z})$; $\mathbf{Q}(:, 1) = []$; $\mathbf{T} = \mathbf{Q}^T$

After the determination of the target vectors $\mathbf{t}_k, k = 1, \ldots, d$ stored in matrix $\mathbf{T}$, we proceed with the kernel regression step of CS-KSR. Here we should note that the above process can be directly used within the CS-KSR method in order to accelerate its operation. In the following, we employ three approximate kernel regression schemes in order to further speedup the learning process. The adoption of such approximate kernel regression schemes leads to an important reduction on memory requirements, which allows us to apply the proposed ACS-KSR method in large-scale verification problems.

B. Kernel Regression step

In order to determine the projection vectors $\mathbf{A} = [\mathbf{a}_1, \ldots, \mathbf{a}_d]$, CS-KSR solves the following kernel regression problem:

$$\mathbf{W}^* = argmin \ \|\mathbf{W}^T\mathbf{\Phi} - \mathbf{T}\|_2^2 \quad (26)$$

and the projection vectors in $\mathbf{A}$ are obtained by expressing $\mathbf{W}$ as a linear combination of the training data representations in the kernel space, i.e. $\mathbf{W} = \mathbf{\Phi}\mathbf{A}$. In order to accelerate the kernel regression step, we follow a different approach. That is, we solve the kernel regression problem in (26) and express matrix $\mathbf{W}$ as a linear combination of a set of R ($R < N$) reference vectors $\mathbf{\Psi} \in \mathbb{R}^{|\mathcal{F}| \times R}$, i.e. $\mathbf{W} = \mathbf{\Psi}\mathbf{A}$. Then, (26) can be expressed as a function of $\mathbf{A}$ as follows:

$$\begin{aligned} \mathbf{A}^* &= argmin \ \|\mathbf{A}^T\mathbf{\Psi}^T\mathbf{\Phi} - \mathbf{T}\|_2^2 \\ &= argmin \ \|\mathbf{A}^T\tilde{\mathbf{K}} - \mathbf{T}\|_2^2, \end{aligned} \quad (27)$$

where we replaced $\mathbf{\Psi}^T\mathbf{\Phi}$ with $\tilde{\mathbf{K}}$. $\tilde{\mathbf{K}} \in \mathbb{R}^{R \times N}$ is a *reduced kernel* expressing the training data representations in a kernel space defined on the reference data $\mathbf{\Psi}$. A similar approach has been previously used to approximate kernel SVM and ELM solutions [32], [35]. It is worth noting here that by using the training data representations in $\mathcal{F}$ as reference vectors (i.e. by setting $\mathbf{\Psi} = \mathbf{\Phi}$) the two problems (26) and (27) are equivalent.

By determining the saddle point of (27) with respect to $\mathbf{A}$, we obtain:

$$\mathbf{A} = \left(\tilde{\mathbf{K}}\tilde{\mathbf{K}}^T\right)^{-1}\tilde{\mathbf{K}}\mathbf{T}^T. \quad (28)$$

Here we can observe that matrix $\tilde{\mathbf{K}}\tilde{\mathbf{K}}^T$ is always non-singular and, thus, regularization is not required (as opposite to (14)).

To determine the reference vectors $\mathbf{\Psi}$ we employ the following three choices:

- *Random kernel*: the reduced kernel space can be determined by randomly selecting R training vectors $\mathbf{x}_i$. That is, the matrix $\tilde{\mathbf{K}}$ will be a sub-matrix of the original kernel matrix $\mathbf{K}$, where only those R rows corresponding to the selected training vectors are kept. This approach corresponds to the use of a randomized kernel space, which has been proposed for reducing the complexity of the kernel SVM classifier [32]. In Appendix A, we analytically show that this choice has the effect of regularization in the solution of the standard kernel regression model. In the following, we refer to this variant of the proposed method by using the suffix *(rtds)* standing for *random training data selection*.
- *Class-specific kernel*: the reduced kernel space can be determined by selecting the $R = C$ training vectors belonging to the client class. That is, the matrix $\tilde{\mathbf{K}}$ will be the sub-matrix of the original kernel matrix $\mathbf{K}$, where only those C rows corresponding to the client training vectors are kept. This approach corresponds to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2582562, IEEE Transactions on Information Forensics and Security

TABLE I: Time and memory complexities of CS-KSR and ACS-KSR methods.

Dataset	Time	Memory
CS-KSR	$O(\frac{40}{6}N^3 + (D+d)N^2)$	$4N^2 + dN + DN$
ACS-KSR	$O((C^2 + (d+D)R)N + R^3 + dR^2 - \frac{1}{3}C^3)$	$(d+R)N + DN$
ACS-KSR (pv)	$O((C^2 + (d+D+kD)R)N + R^3 + dR^2 - \frac{1}{3}C^3)$	$(d+R)N + (N+R)D$

the use of class-specific kernel space, which has been proposed in our previous work [30]. In Appendix B, we analytically show that this approach is closely connected to the Nyström-based kernel matrix approximation when the approximation reference data are the client vectors. In the following, we refer to this variant of the proposed method by using the suffix *(csks)* standing for *class-specific kernel space*.

- *Prototype-based kernel*: the reduced kernel space can be determined by using as reference vectors prototype vectors, obtained by e.g. applying K-Means clustering (with R centroids) on the training vectors $\mathbf{x}_i$. Let us define as $\mathbf{p}_i \in \mathbb{R}^D$, $i = 1, \ldots, R$ the prototype vectors. Then, matrix $\tilde{\mathbf{K}}$ will have values $[\tilde{K}]_{ij} = \kappa(\mathbf{p}_i, \mathbf{x}_j)$. This approach has also been used in SVM-based classification [31] where it was shown to achieve good performance. In the following, we refer to this variant of the proposed method by using the suffix *(pv)* standing for *prototype vectors*.

C. The ACS-KSR method

Combining the processing steps described in subsections IV-A and IV-B, the proposed method is obtained. ACS-KSR is algorithmically illustrated in Pseudocode 2.

Pseudocode 2: The proposed ACS-KSR

procedure $\mathbf{A}$ = ACSKSR($\tilde{\mathbf{K}}$, l, d)

% Calculate the target vectors
    $\mathbf{T}$ = targets_calculation (l, d);

% Calculate $\mathbf{A}$
    $[R, p] = chol(\tilde{\mathbf{K}}\tilde{\mathbf{K}}^T)$;
    $\mathbf{A} = R \setminus (R^T \setminus \tilde{\mathbf{K}}) * \mathbf{T}^T$;

D. Time and Memory complexity

Here we compare the time and memory complexities of the proposed ACS-KSR method with those of the standard CS-KSR technique [22], [23]. The time complexity of CS-KSR is as follows [36]:

- Kernel matrix calculation having a time complexity of $O(DN^2)$.
- Calculation of the matrix $\mathbf{P} = \mathbf{P}_I^{-1}\mathbf{P}_C$ having time complexity of $O(2N^3)$.
- Eigenanalysis of $\mathbf{P}$ having a time complexity of $O(\frac{9}{2}N^3)$.
- Calculation of $\mathbf{A}$ through (14), which has a time complexity of $O(\frac{1}{6}N^3 + dN^2)$.

Thus, the time complexity of CS-CSR is equal to $O(\frac{40}{6}N^3 + (D+d)N^2)$.

The time complexity of ACS-KSR is as follows:

- Calculation of $\mathbf{T}$ having a time complexity of $O(C^2N - \frac{1}{3}C^3)$.
- Calculation of $\tilde{\mathbf{K}}$ having a time complexity of $O(DRN)$.
- Calculation of $\mathbf{A}$ through (28) having a time complexity of $O(R^3 + dRN + dR^2)$.

Thus, the time complexity of ACS-KSR is equal to $O((C^2 + (d+D)R)N + R^3 + dR^2 - \frac{1}{3}C^3)$. In the case where the reference vectors are determined by applying K-Means, an additional complexity of the order of $O(kRND)$ is required for clustering the training vectors. As has been shown in [37], [31] a small number of K-Means iterations, e.g. $k = 5$, is sufficient for the determination of a reasonably good set of prototype vectors and, thus, this term does not contribute significantly to the time complexity of this processing step. In all our experiments, we used the same value ($k = 5$) for prototype vector determination.

Comparing the time complexities of the two methods, we can see that the one of CS-KSR is a cubic function of the training data size N. On the other hand, the time complexity of ACS-KSR is a linear function of N, while it is a cubic function of the reference data size R and the number of client samples C. This means that, using a relatively small value for R, the time complexity of ACS-KSR can be much lower than the time complexity of CS-KSR. In the case where the number of client samples is high, an additional important computational complexity is added, however, in most verification problems we expect that the number of client samples is much lower than the number of training samples.

In terms of memory, CS-KSR requires the storage of four $N \times N$ matrices and one $N \times d$ matrix, while ACS-KSR requires the storage of one $R \times N$ matrix and one $N \times d$ matrix. In the case where the reference vectors are determined by applying K-Means a matrix of $R \times D$ dimensions is also required. All methods require the storage of the training data in a $D \times N$ matrix. The time and memory complexities of CS-KSR and ACS-KSR are summarized in Table I. Thus, the memory complexity of CS-KSR is a quadratic function of the training data size N, while the one of ACS-KSR scales linearly with respect to N.

V. Experiments

In this section, we present experiments conducted in order to evaluate the performance of the proposed ACS-KSR method in face verification problems. We have employed three facial image datasets to this end, namely the AR [38], a combination of the Public Figures Face and Labeled Face in the Wild (PubFig+LFW) [39], and the YouTube Faces [40] datasets.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2582562, IEEE Transactions on Information Forensics and Security

A description of the datasets is provided in subsection V-A. Information related to the cardinality of the datasets, training-test partitions and data dimensionality is provided in Table II. Experimental results are given in subsection V-B. In all our experiments, we have employed the RBF kernel function:

$$\mathbf{K}(\mathbf{x}_i, \mathbf{x}_j) = exp\Big(- \frac{\|\mathbf{x}_i - \mathbf{x}_j\|_2^2}{2\sigma^2}\Big), \qquad (29)$$

where the value of the Gaussian scale σ is set equal to the mean Euclidean distance between the training vectors $\mathbf{x}_i$, which corresponds to the natural scaling value of each dataset. All experiments have been conducted on a 24-core, 2.5GHz workstation with 96GB RAM, using single floating point precision and a MATLAB implementation.

A. Datasets description

The AR dataset [38] consists of over 4000 facial images depicting 70 male and 56 female faces. In our experiments we have used the preprocessed (cropped) facial images provided by the database, depicting 100 persons (50 males and 50 females) 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). Each person was recorded in two sessions, separated by two weeks. Sample images of the dataset are illustrated in Figure 1.

Fig. 1: *Facial images depicting a person from the AR dataset.*

The PubFig+LFW dataset [39] has been created by combining the Public Figures (PubFig) [41] and the Labeled Faces in the Wild (LFW) [42] datasets, in order to mimic a web-scale face recognition scenario of finding specific celebrities while ignoring all other faces. Five dataset partitions depicting 200 persons of PubFig with a random $75\%/25\%$ train/test split are provided. All faces of the LFW dataset (except from the facial images depicting the 138 overlapping person ID classes) were added as distractors, thus, converting the closed-universe face recognition problem of the PubFig dataset to an open-universe one. Sample facial images from PubFig+LFW dataset are illustrated in Figure 2.

The Youtube Faces dataset [40] consists of 621126 facial images depicting 1595 persons. All images have been downloaded from YouTube. In our experiments we have employed the facial images depicting persons in at least 500 images, resulting to a dataset of 370319 images and 340 classes. Sample images from these datasets are illustrated in Figure 3.

B. Results

In our first experiment, we have applied the proposed ACS-KSR method on the facial images of AR dataset. Since there is no commonly used experimental protocol of this dataset for face verification, we randomly split each ID class into two sets

Fig. 2: *Facial images depicting persons from the PubFig+LFW dataset.*

Fig. 3: *Facial images depicting persons from the YouTube dataset.*

(70% for training and 30% for testing) and we form a training set and a test set of 1820 and 780 samples, respectively. Using training samples, we determine 100 discriminant subspaces (each for a person in the dataset). Subsequently, we map the test samples in all discriminant subspaces and calculate the equal error rate (ERR) metric for each face verification problem. In order to calculate the ERR metric for each face verification problem, we map test samples to the corresponding discriminant subspace and we measure its similarity to the mean client vector determined using the client vectors as $s_i = \|\mathbf{y}_i - \mathbf{m}\|_2^{-1}$. The similarity values of all test samples are sorted in a descending order, and the ERR metric is calculated. We measure the performance of the method by calculating the mean ERR over all face verification problems. In order to eliminate randomness related to the training-test partition, we apply the above-described process five times and report the mean performance and the corresponding standard deviation. In order to obtain facial image representations, we re-scaled the original facial images to 40×30-pixel images, which are subsequently vectorized in order to obtain 1200-dimensional facial vectors.

In our first experiment, we have tested the performance of the proposed method for discriminant space dimensionalities equal to $d = \{1, 2, 3, 4, 5, 10\}$ and reference vector set cardinalities $R = \{50, 100, 250, 500, 1000, N\}$. In Table III we provide the performance obtained by using a reference vector cardinality equal to $R = N$. In this Table, we also provide the performance of Support Vector Machine (SVM),

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2582562, IEEE Transactions on Information Forensics and Security

TABLE II: Facial image datasets information.

Dataset	Dimensionality (D)	# of classes (C)	# of data	# of training data (N)	# of test data
AR	1200	100	2600	1820	780
PubFig+LFW	1536	200	47189	35469	11720
YTFaces	1770	340	370319	259223	111096

Least Mean Squares regression (LMS), Linear Discriminant Analysis (LDA), Class-Specific LDA (CS-LDA), Kernel Discriminant Analysis (KDA), Kernel Spectral Regression (KSR), Class-Specific KDA (CS-KDA) and Class-Specific KSR (CS-KSR). As can be seen, the proposed method is both efficient and effective. When compared to the CS-KSR solution, ACS-KSR requires lower training time. Since the regression step of both methods is the same, the difference in time lies in the eigenanalysis step (as detailed in Section IV-A). A comparison of all class-specific approaches with respect to the discriminant space dimensionality d is illustrated in Figure 4. As can be seen, all methods achieve their best performance for a discriminant space dimensionality $d > 1$. In addition, it can be seen that the proposed method provides similar performance for $d \geq 4$. The performance of the proposed method for various reference vector set cardinalities R is illustrated in Figures 5 and 6 for the ACS-KSR (rtds) and ACS-KSR (pv) variants, respectively. The performance of both variants converges for values $R \geq 500$.

TABLE III: Performance (mean ERR %) and training time per class (seconds) on AR dataset.

Method	Performance	Training Time
SVM	**0.76±0.01**	**0.2±0.01**
LMS	2.98±1.27	0.11±0.01
LDA	3.32±0.66 (d=1)	0.57±0.01
CS-LDA	8.47±0.76 (d=10)	1.52±0.04
KDA	1.86±0.75 (d=1)	1.04±0.01
KSR	2.71±0.65 (d=1)	0.69±0.01
CS-KDA	1.34±0.42 (d=2)	1.06±0.05
CS-KSR	0.79±0.12 (d=10)	0.97±0.02
ACS-KSR	**0.79±0.12 (d=10)**	**0.2±0.02**

Given that all three variants of the proposed ACS-KSR method exploit the same target vectors (obtained by applying the process described in subsection IV-A), we expect that differences that may occur by applying them on the same data will be due to the different approximate solutions of the kernel regression step, as described in subsection IV-B. The use of class-specific kernel space for approximate kernel regression, leads to an error rate equal to 27.26 ± 1.16, which is higher when compared to random data selection and prototype-based kernel choices. This can be explained by the fact that the number of reference vectors in this case is much smaller (approximately 1% of the size of the problem). Another reason might be related to our analysis in Appendix B, stating that the exploitation of class-specific kernels is directly connected to the Nyström-based kernel matrix approximation, where the reference vectors of the approximation are the client class vectors. This choice might lead to insufficient kernel matrix approximation, especially in face verification problems, where

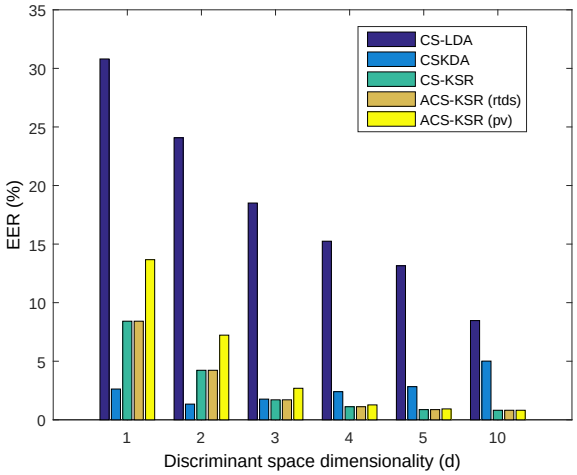

Fig. 4: *Performance of class-specific approaches for different values of d ($R = N$).*

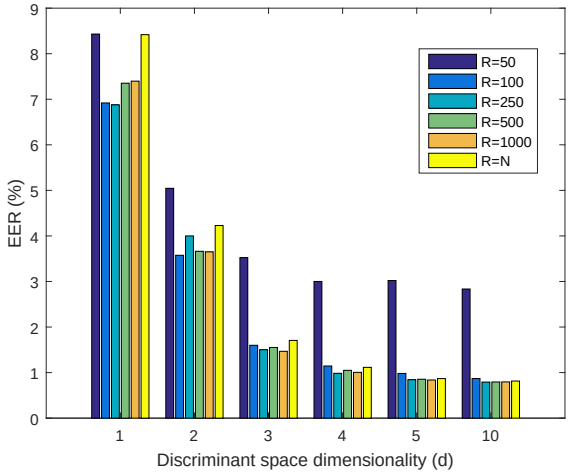

Fig. 5: *Performance of ACS-KSR (rtds) for different reference vector set cardinalities (R).*

the client class is expected to be unimodal. Random data selection and prototype vector-based approximate solutions lead to a better performance, since, on the one hand, random data selection has the effect of regularization (Appendix A) while, on the other hand, by using prototype vectors for kernel space definition, we expect that our input space will be sufficiently sampled [31]. Obviously, when an exact kernel regression solution is calculated, the latter two variants of the proposed method provide the same performance.

In our second experiment, we applied the proposed method on the PubFig+LFW dataset, where we used the standard

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2582562, IEEE Transactions on Information Forensics and Security

TABLE IV: Performance (mean ERR %) on PubFig+LFW dataset for approximate solutions of the proposed method using different number of reference vectors.

Method	R=500	R=1000	R=1500	R=2000	R=2500
ACS-KSR (rtds)	14.36±0.14 (d=5)	12.48±0.36 (d=5)	16.39±0.8 (d=5)	11.14±0.66 (d=5)	15.43±1.47 (d=5)
ACS-KSR (pv)	14.69±0.16 (d=5)	15.47±0.53 (d=5)	17.08±0.58 (d=5)	13.65±0.26 (d=5)	11.49±0.82 (d=5)

TABLE V: Training time per class (seconds) on PubFig+LFW dataset for approximate solutions of the proposed method using different number of reference vectors.

Method	R=500	R=1000	R=1500	R=2000	R=2500
ACS-KSR (rtds)	14.51±0.11	17.53±0.95	20.28±0.09	26.13±0.11	32.76±0.21
ACS-KSR (pv)	15.15±0.15	17.76±0.93	20.67±0.25	26.52±0.13	33.46±0.15

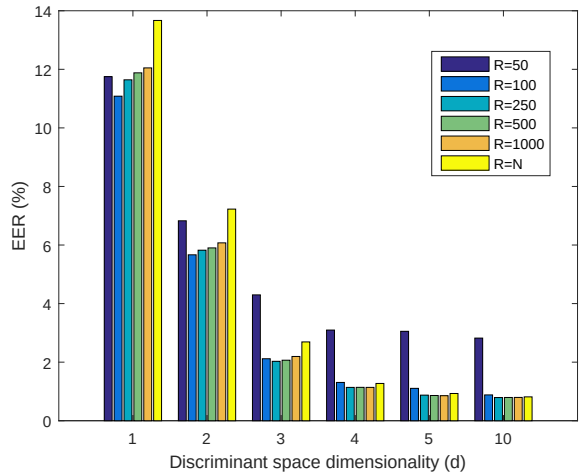

Fig. 6: *Performance of ACS-KSR (pv) for different reference vector set cardinalities (R).*

TABLE VI: Performance (mean ERR %) and training time per class (seconds) on PubFig+LFW dataset for various nonlinear classification methods.

Method	Performance	Training time
SVM	**7.86±0.43**	26.31±0.52
LMS	17.48±0.15	**9.24±0.11**
LDA	15.61±0.27	24.21±0.79
CS-LDA	14.69±0.34	37.58±0.85
ELM	25.81±0.91 (R=2500)	33.71±0.64
RKSVM	18.95±0.46 (R=2500)	28.96±0.52
AKELM	17.21±0.98 (R=2000)	18.84±0.94
RFR	20.62±0.84 (R=2500)	13.08±0.72
ACS-KSR (rtds)	**11.14±0.66 (R=2000)**	26.13±0.11
ACS-KSR (pv)	**11.49±0.82 (R=2500)**	33.46±0.15

training/test partitions provided by the database. Facial images are represented using the 1536-dimensional vectors $\mathbf{x}_i$ obtained by applying the representation scheme suggested in [39]. Since the application of the standard kernel approach for this dataset is very slow, we provide results of the approximate versions of the proposed method. We have tested the variants of the proposed ACS-KSR method that employ random training vectors and prototype vectors as reference vectors using different reference vector size cardinalities $R = \{500, 1000, 1500, 2000, 2500\}$ and target dimensions $d = \{1, 5, 10\}$. We omit reporting the performance of the ACS-KSR variant exploiting a class-specific kernel definition, since it's performance was inferior to that of the other two variants. The results are provided in Table IV, while the mean training time required to train each class-specific model is provided in Table V.

Here we should note that, in order to speedup the training process, we have employed the same reference vector set for all the discriminant models calculated for the various client classes. That is, on each experiment, we have randomly selected a set of reference vectors for the variant of the proposed ACS-KSR method using the random training data selection approach and we used this set for calculating the kernel matrix $\tilde{\mathbf{K}}$ for the calculation of all the discriminant projections through (28). In a similar manner, we applied K-Means only once for the variant of the proposed ACS-KSR method using prototype vectors. In this way, the training process can be accelerated, since the reference vectors and matrix $\left(\tilde{\mathbf{K}}\tilde{\mathbf{K}}^T\right)^{-1}\tilde{\mathbf{K}}$ used for the calculation of matrix $\mathbf{A}$ in (28) are calculated only once. Thus, after determining the discriminant space of the first face verification problem, for the determination of the discriminant subspace for all other verification problems, we need to apply the process described in subsection IV-A for the calculation of matrix $\mathbf{T}$ and a matrix multiplication.

In order to evaluate the performance of the proposed methods, we have applied the following four approximate classification schemes on the PubFig+LFW dataset using the same dataset partitions: Extreme Learning Machine (ELM) [43], Reduced Kernel Support Vector Machine (RKSVM) [32], Approximate Kernel Extreme Learning Machine (AKELM) [35], Random Feature Regression (RFR) [44] using the same feature space dimensionality or reference vector set cardinality, i.e. $R = \{500, 1000, 1500, 2000, 2500\}$. In addition, we have tested the performance of SVM, LMS, LDA and CS-LDA. In Table VI, we provide the best performance achieved by each method, along with the corresponding training time. For all algorithms we have applied the same process described above in order to speedup the training of multiple verification problems. As can be seen, the proposed ACS-KSR method

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2582562, IEEE Transactions on Information Forensics and Security

TABLE VII: Performance (mean ERR %) and training time per class (seconds) on PubFig+LFW dataset for various non-linear discriminant analysis approaches. Methods trained on a random subset of the training data are followed by an asterisk.

Method	Performance	Training time
KDA*	27.15±0.45 (d=1)	54.18±3.88
KSR*	23.5±0.58 (d=1)	**4.81±0.33**
CS-KDA*	23.48±0.42 (d=1)	95.71±1.91
CS-KSR*	23.36±0.47 (d=5)	28.13±0.47
ACS-KSR (rtds)	**11.14±0.66 (d=10)**	26.13±0.11
ACS-KSR (pv)	**11.49±0.82 (d=10)**	33.46±0.15

TABLE X: Performance (mean ERR %) and training time per class (seconds) on Youtube Faces dataset for various nonlinear classification methods.

Method	Performance	Training time
SVM	**1.68±0.09**	23.99±0.67
LMS	21.9±1.34	**8.65±0.27**
LDA	28.27±0.66	75.71±2.07
CS-LDA	23.62±0.77	73.49±1.38
ELM	17.08±0.57 (R=1500)	95.79±2.91
RKSVM	15.44±0.55 (R=1500)	97.59±3.13
AKELM	12.02±0.57 (R=500)	13.87±2.19
RFR	27.67±0.48 (R=2000)	114.41±0.61
ACS-KSR (rtds)	**2.8±0.15 (R=2500)**	47.02±7.83
ACS-KSR (pv)	**2.26±0.11 (R=2500)**	47.48±7.32

outperforms all the discriminant analysis-based methods, while requiring the same training time. Here we should note that, since approximate solutions are adopted for the kernel methods, linear methods become competitive, as they are able to efficiently make use of the entire training set. SVM provides the best performance, while the performance of discriminant analysis-based methods is still inferior to the performance provided by the proposed approximate kernel solutions.

Moreover, we compare the proposed approximate discriminant learning approach with that of applying the original methods on a subset of the training data. To this end, we have trained the original KDA, KSR, CS-KDA and CS-KSR methods on reduced training sets formed by 5000 (randomly selected) training vectors and measured their performance on the same test sets used to evaluate the performance of the proposed method. Performance of each method and the corresponding training times are provided in Table VII. The proposed approximate class-specific discriminant learning approach compares favorably to the other approaches, while it requires similar training time.

Finally, we have applied the proposed method on the Youtube Faces dataset. We have employed the facial image representation suggested in [40]. Specifically, the bounding box of the face detector output has been expanded by 2.2 of its original size and cropped from the video frame. The expanded bounding box is resized to an image of 200×200 pixels. The centered 100×100 pixels are subsequently used in order to obtain the facial image, which is converted to grayscale. Facial images are then aligned by fixing the coordinates of automatically detected facial feature points [45] and the Local Binary Patter (LBP) description [46] is employed, leading to a facial image representation $\mathbf{x}_i$ of $D = 1770$ dimensions.

Here we should note that, in YouTube Faces dataset, recent works have employed a pair-wise verification experimental protocol [40], [47], [48]. Such methods build a verification model by using image pairs followed by a label denoting whether the to images depict the same person. Our method tackles the verification problem described in section II and builds a model for a client using a set of images (with negative and positive training images). Since there is no widely adopted experimental protocol on the YouTube Faces dataset for the face verification problem tackled in this paper, we randomly split each ID class in two sets (70% for training and 30% for testing) and we form a training set and a test set of $259,223$ and $111,096$ samples, respectively. Using training samples, we determine 340 discriminant subspaces (each for a person in the dataset). Subsequently, we map the test samples in all discriminant subspaces and calculate the equal error rate (ERR) metric for each face verification problem. We measure the performance of the method by calculating the mean ERR over all face verification problems. In order to eliminate randomness related to the training-test partition, we apply the above-described process five times and report the mean performance and the corresponding standard deviation.

We have tested the variants of the proposed ACS-KSR method that employ random training vectors and prototype vectors as reference vectors using different reference vector size cardinalities $R = \{500, 1000, 1500, 2000, 2500\}$ and target dimensions $d = \{1, 5, 10, 25\}$. In order to speedup the overall training process in the experiments on the YouTube Faces data set, the prototype vectors used in the ACS-KSR (pv) variant of the proposed method have been determined by clustering a set of $5 \cdot 10^4$ randomly sampled training vectors. In addition, we have applied the same process described above in order to speedup the training of multiple verification problems. The performance achieved by the proposed method is provided in Table VIII. The corresponding (per class) training times are provided in Table IX. In addition, we compare the performance of the proposed method with that of SVM, LMS, LDA, CS-LDA, ELM, RKSVM, AKELM and RFR. other linear and nonlinear methods in Table X. Similar to the PubFig+LFW case, the proposed method provides good performance, while requiring similar training time with that of the competing classifiers.

VI. Conclusions

In this paper we described a novel approximate method for class-specific nonlinear discriminant analysis. We built on the kernel Spectral Regression approach, where by using class-specific intra-class and out-of-class scatters definition, we showed that the eigenanalysis problem solved for finding class-specific target vectors for regression can be obtained by applying a fast vector orthogonalization process. In order to speedup the kernel regression processing step, we have employed approximate kernel space definitions. We showed that the use of randomized and class-specific kernels has

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2582562, IEEE Transactions on Information Forensics and Security

TABLE VIII: Performance (mean ERR %) on Youtube Faces dataset for approximate solutions of the proposed method using different number of reference vectors.

Method	R=500	R=1000	R=1500	R=2000	R=2500
ACS-KSR (rtds)	15.99±0.01 (d=5)	14.88±0.84 (d=5)	10.35±0.69 (d=5)	6.88±0.55 (d=10)	2.8±0.15 (d=10)
ACS-KSR (pv)	9.69±1.13 (d=5)	15.60±1.06 (d=5)	10.58±1.32 (d=5)	5.96±0.66 (d=10)	2.26±0.11 (d=10)

TABLE IX: Training time per class (seconds) on Youtube Faces dataset for approximate solutions of the proposed method using different number of reference vectors.

Method	R=500	R=1000	R=1500	R=2000	R=2500
ACS-KSR (rtds)	14.23±2.22	19.71±3.12	24.27±4.12	34.61±2.14	47.02±7.83
ACS-KSR (pv)	13.97±2.89	19.28±2.04	24.25±4.13	36.31±2.46	47.48±7.32

the effect of regularization and the Nyström-based approximation, respectively. We evaluated the proposed approach in face verification problems and we compared its performance with related approaches. Extensive experiments indicate that the proposed approach can achieve satisfactory performance, while scaling well with the size of the data.

Interesting extensions of the proposed method include: a) the exploitation of multiple kernel types in order to determine kernel spaces which further increase class discrimination [49], [50], b) gradient-based iterative optimization which can be exploited for scaling the method to even bigger datasets, which may not fit in memory. We will investigate these possibilities in our future research.

APPENDIX A
EFFECT OF RANDOM KERNEL IN KERNEL REGRESSION

Here we give an analysis of the effect of using a (randomly selected) subset of the training vectors in order to determine the reduced kernel matrix $\tilde{\mathbf{K}}$ for kernel regression (28). Let us denote by $\mathbf{b} \in \mathbb{R}^N$ an indicator vector, having elements equal to $b_i = 1$, if the $\mathbf{x}_i$ is selected as a reference sample and $b_i = 0$ if $\mathbf{x}_i$ is not selected. That is, $\|\mathbf{b}\| = R$. Let us also denote by $\mathbf{t}_k$ the target values corresponding to the k-th eigenvector (i.e. the k-th column of $\mathbf{T}$).

The mean training error of the reduced kernel regression model can be expressed as follows:

$$\begin{aligned} L(\mathbf{W},\mathbf{b}) &= \min_{\mathbf{w}_k}\frac{1}{N}\sum_{k=1}^{C}\|\mathbf{\Phi}\mathbf{w}_k - \mathbf{t}_k\|_2^2 \\ &= \min_{\mathbf{a}_k}\frac{1}{N}\sum_{k=1}^{C}\|\mathbf{K}(\mathbf{a}_k\circ\mathbf{b}) - \mathbf{t}_k\|_2^2 \\ &= \min_{\mathbf{a}_k}\frac{1}{N}\sum_{k=1}^{C}\Big[(\mathbf{a}_k\circ\mathbf{b})^T\mathbf{K}\mathbf{K}(\mathbf{a}_k\circ\mathbf{b}) \\ &- 2\mathbf{t}_k\mathbf{K}\mathbf{K}(\mathbf{a}_k\circ\mathbf{b}) + \mathbf{t}_k^T\mathbf{t}_k\Big] \end{aligned} \quad (30)$$

The expected value of the training error can be expressed as follows:

$$\begin{aligned} E[L(\mathbf{W},\mathbf{b})] &= E\Bigg[\min_{\mathbf{a}_k}\frac{1}{N}\sum_{k=1}^{C}\Big[(\mathbf{a}_k\circ\mathbf{b})^T\mathbf{K}\mathbf{K}(\mathbf{a}_k\circ\mathbf{b}) \\ &- 2\mathbf{t}_k\mathbf{K}\mathbf{K}(\mathbf{a}_k\circ\mathbf{b}) + \mathbf{t}_k^T\mathbf{t}_k\Big]\Bigg] \\ &\leq \min_{\mathbf{a}_k}\frac{1}{N}E\Bigg[\sum_{k=1}^{C}\Big[(\mathbf{a}_k\circ\mathbf{b})^T\mathbf{K}\mathbf{K}(\mathbf{a}_k\circ\mathbf{b}) \\ &- 2\mathbf{t}_k\mathbf{K}\mathbf{K}(\mathbf{a}_k\circ\mathbf{b}) + \mathbf{t}_k^T\mathbf{t}_k\Big]\Bigg] \\ &= \min_{\mathbf{a}_k}\frac{1}{N}\Bigg[\frac{R}{N}\mathbf{a}_k^T\left(\frac{R}{N}\mathbf{K}\mathbf{K} + \beta\mathbf{D}\right)\mathbf{a}_k \\ &- \frac{2R}{N}\mathbf{t}_k^T\mathbf{K}\mathbf{a}_k + \mathbf{t}_k^T\mathbf{t}_k\Bigg] = \mathcal{J}_k^{\mathbf{b}}, \end{aligned} \quad (31)$$

where $\beta = 1 - \frac{R}{N}$ and $\mathbf{D} = diag(\mathbf{K}\mathbf{K})$. Solving for $\nabla_{\mathbf{a}_k}\mathcal{J}_k^{\mathbf{b}} = 0$, we obtain:

$$\mathbf{a}_k = \left(\frac{R}{N}\mathbf{K}\mathbf{K} + \beta\mathbf{D}\right)^{-1}\mathbf{K}\mathbf{t}_k. \quad (32)$$

Substituting (32) in (31), we obtain:

$$\mathcal{J}_k^{\mathbf{b}} = \frac{1}{N}\left[\mathbf{t}_k^T\mathbf{t}_k - \frac{R}{N}\mathbf{t}_k^T\mathbf{K}\left(\frac{R}{N}\mathbf{K}\mathbf{K} - \frac{N-R}{N}\mathbf{D}\right)^{-1}\mathbf{K}\mathbf{t}_k\right]. \quad (33)$$

Thus, by setting $\mathbf{b} = \mathbf{1}_N$ the training error of kernel regression is equal to:

$$\mathcal{J}_k^{\mathbf{1}} = \frac{1}{N}\mathbf{t}_k^T\mathbf{t}_k - \frac{1}{N}\mathbf{t}_k^T\mathbf{K}(\mathbf{K}\mathbf{K})^{-1}\mathbf{K}\mathbf{t}_k. \quad (34)$$

From (34) it can be seen that, in the case where $\mathbf{K}$ is invertible, kernel regression achieves zero training error. For the training

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2582562, IEEE Transactions on Information Forensics and Security

error bound of reduced kernel regression we have:

$$\begin{aligned}
&E[L(\mathbf{W},\mathbf{b})] - L(\mathbf{W}_{out},\mathbf{1}) = \frac{1}{N}\mathbf{t}_k^T\mathbf{t}_k \\
&-\frac{R}{N^2}\mathbf{t}_k^T\mathbf{K}\left(\frac{R}{N}\mathbf{K}\mathbf{K}+\frac{N-R}{N}\mathbf{D}\right)^{-1}\mathbf{K}\mathbf{t}_k \\
&= \frac{1}{N}\mathbf{t}_k^T\mathbf{t}_k - \frac{1}{N}\mathbf{t}_k^T\mathbf{K}\left(\mathbf{K}\mathbf{K}+\frac{N-R}{R}\mathbf{D}\right)^{-1}\mathbf{K}\mathbf{t}_k \\
&= 1 - \frac{1}{N}\mathbf{t}_k^T\mathbf{K}\mathbf{K}\left(\mathbf{K}\mathbf{K}+\frac{1-p}{p}\mathbf{D}\right)^{-1}\mathbf{t}_k \\
&\leq 1 - \frac{1}{N}\mathbf{t}_k^T\mathbf{K}\mathbf{K}\left(\mathbf{K}\mathbf{K}+\frac{1-p}{p}\mathbf{I}\right)^{-1}\mathbf{t}_k \\
&= 1 - \frac{1}{N}\mathbf{t}_k^T\mathbf{U}\mathbf{\Lambda}\left(\mathbf{\Lambda}+\frac{1-p}{p}\mathbf{I}\right)^{-1}\mathbf{U}^T\mathbf{t}_k, \qquad (35)
\end{aligned}$$

where we have set $R = pN$, $\mathbf{\Lambda} = diag(\lambda_1,\ldots,\lambda_N)$ is a diagonal matrix containing the eigenvalues of $\mathbf{KK}$ (and thus $\mathbf{\Lambda} = diag(l_1^2,\ldots,l_N^2)$, where l_i is the i-th eigenvalue of $\mathbf{K}$) and $\mathbf{U} \in \mathbb{R}^{N\times N}$ is an orthogonal matrix containing the corresponding eigenvectors. As can be seen in (35), the training error bound of reduced kernel regression has a regularization form. In the case where $p = 1$, the training error bound is equal to zero. For any other value of $0 < p < 1$, the training error bound is a function of p and of the eigenvalues λ_i, $i = 1,\ldots,N$. This fact can be exploited in order to determine appropriate numbers of reference vectors R, since for a given problem, the eigenvalues of the eigenvalues l_i of $\mathbf{K}$ are fixed.

To illustrate the result of (35), consider the special case where $\mathbf{KK}$ has its eigenvalues equal to $\alpha R = \alpha pN$. In this case, the error bound of reduced kernel regression is equal to:

$$E[L(\mathbf{W},\mathbf{b})] - L(\mathbf{W},\mathbf{1}) \leq 1 - \frac{\alpha p^2 N}{\alpha p^2 N - p + 1}. \qquad (36)$$

Thus, for any value $\epsilon > 0$, the minimum value of p satisfying $E[L(\mathbf{W},\mathbf{b})] - L(\mathbf{W},\mathbf{1}) \leq \epsilon$ is given by:

$$p = \frac{(\epsilon-1) + \sqrt{(1-\epsilon)^2 + 4\alpha N(1-\epsilon)}}{2\alpha N}. \qquad (37)$$

For example, for a training error bound $\epsilon = 0.001$, when $\alpha = 10$ and $N = 100000$, a value of $p = 0.001$ should be used.

APPENDIX B
RELATION BETWEEN CLASS-SPECIFIC KERNEL AND NYSTRÖM-BASED KERNEL APPROXIMATION

Here we provide an analysis indicating the relation between class-specific kernel and Nyström-based kernel approximation for the case of class-specific nonlinear projections. In order to do this, let us assume that the training data representations in $\mathcal{F}$ are centered to the mean of the client class $\mathbf{m}_\phi$. Then, the (centered) out-of-class and in-class scatter matrices are given by:

$$\tilde{\mathbf{S}}_I = \sum_{i,l_i=-1}\phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^T = \mathbf{\Phi}_I\mathbf{\Phi}_I^T \qquad (38)$$

and

$$\tilde{\mathbf{S}}_C = \sum_{i,l_i=1}\phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^T = \mathbf{\Phi}_C\mathbf{\Phi}_C^T, \qquad (39)$$

where $\mathbf{\Phi}_I \in \mathbb{R}^{|\mathcal{F}|\times I}$ and $\mathbf{\Phi}_C \in \mathbb{R}^{|\mathcal{F}|\times C}$ are matrices containing the centered data representations of the impostor and client class in $\mathcal{F}$, respectively.

Having defined these matrices, the following (equivalent to $\mathcal{J}$ in (7)) criterion can be defined:

$$\begin{aligned}
\tilde{\mathcal{J}} &= \frac{Tr\left(\mathbf{W}^T\tilde{\mathbf{S}}_I\mathbf{W}\right)}{Tr\left(\mathbf{W}^T\tilde{\mathbf{S}}_C\mathbf{W}\right)} + 1 \\
&= \frac{Tr\left(\mathbf{W}^T\tilde{\mathbf{S}}_I\mathbf{W}\right)}{Tr\left(\mathbf{W}^T\tilde{\mathbf{S}}_C\mathbf{W}\right)} + \frac{Tr\left(\mathbf{W}^T\tilde{\mathbf{S}}_C\mathbf{W}\right)}{Tr\left(\mathbf{W}^T\tilde{\mathbf{S}}_C\mathbf{W}\right)} \\
&= \frac{Tr\left(\mathbf{W}^T\tilde{\mathbf{S}}_T\mathbf{W}\right)}{Tr\left(\mathbf{W}^T\tilde{\mathbf{S}}_C\mathbf{W}\right)}, \qquad (40)
\end{aligned}$$

where $\tilde{\mathbf{S}}_T$ expresses the scatter of all samples from the mean vector of client class in the discriminant space and is given by:

$$\tilde{\mathbf{S}}_T = \sum_{i=1}^{N}\phi(\mathbf{x}_i)\phi(\mathbf{x}_i)^T = \mathbf{\Phi}_T\mathbf{\Phi}_T^T. \qquad (41)$$

The solution of (40) is given by solving the generalized eigenanalysis problem $\tilde{\mathbf{S}}_C\mathbf{v} = \lambda\tilde{\mathbf{S}}_T\mathbf{v}$, corresponding to eigenanalysis to the matrix $\tilde{\mathbf{S}} = \tilde{\mathbf{S}}_T^{-1}\tilde{\mathbf{S}}_C$. In order to handle the singularity of $\tilde{\mathbf{S}}_T$, a regularized version is adopted, i.e. $\tilde{\mathbf{S}} = (\tilde{\mathbf{S}}_T + r\mathbf{I})^{-1}\tilde{\mathbf{S}}_C$.

When a class-specific kernel is used in order to approximate the solution of (40), we express the projection matrix $\mathbf{W}$ as a linear combination of the client samples representation the kernel space, i.e. $\mathbf{W} = \mathbf{\Phi}_C\mathbf{A}$. Thus, the solution of $\tilde{\mathcal{J}}$ is given by solving the ratio trace problem:

$$\begin{aligned}
\tilde{\mathcal{J}}_2 &= \left(\mathbf{A}^T\mathbf{\Phi}_c(\mathbf{\Phi}\mathbf{\Phi}^T + r\mathbf{I})\mathbf{\Phi}_C\mathbf{A}\right)^{-1}\left(\mathbf{A}^T\mathbf{\Phi}_C^T\mathbf{\Phi}_C\mathbf{\Phi}_C^T\mathbf{\Phi}_C\mathbf{A}\right) \\
&= \left(\mathbf{A}^T(\tilde{\mathbf{K}}\tilde{\mathbf{K}}^T + r\mathbf{K}_C)\mathbf{A}\right)^{-1}\left(\mathbf{A}^T\mathbf{K}_C\mathbf{K}_C\mathbf{A}\right), \qquad (42)
\end{aligned}$$

where $\mathbf{K}_C \in \mathbb{R}^{C\times C}$ is the kernel matrix formed by the centered client vectors.

The solution of (42) is given by solving the generalized eigenanalysis problem:

$$\mathbf{K}_C\mathbf{K}_C\mathbf{a} = \lambda(\tilde{\mathbf{K}}\tilde{\mathbf{K}}^T + r\mathbf{K}_C)\mathbf{a} \qquad (43)$$

and the columns of $\mathbf{A}$ are the eigenvectors corresponding to the minimal eigenvalues of the matrix $\mathbf{Z} = (\tilde{\mathbf{K}}\tilde{\mathbf{K}}^T + r\mathbf{K}_C)^{-1}(\mathbf{K}_C\mathbf{K}_C)$. Analysing $\mathbf{Z}$, its first term can be expressed as follows:

$$(\tilde{\mathbf{K}}\tilde{\mathbf{K}}^T + r\mathbf{K}_C)^{-1} = \frac{1}{r}\mathbf{K}_C^{-1} - \frac{1}{r^2}\mathbf{K}_C^{-1}\tilde{\mathbf{K}}(\mathbf{I} + \frac{1}{r}\tilde{\mathbf{K}}^T\mathbf{K}_C^{-1}\tilde{\mathbf{K}})^{-1}\tilde{\mathbf{K}}_T\mathbf{K}_C^{-1}. \qquad (44)$$

By substituting (44) in (43) we obtain:

$$\begin{aligned}
\lambda\mathbf{a} &= \left(\frac{1}{r}\mathbf{K}_C^{-1} - \frac{1}{r^2}\mathbf{K}_C^{-1}\tilde{\mathbf{K}}(\mathbf{I} + \frac{1}{r}\tilde{\mathbf{K}}^T\mathbf{K}_C^{-1}\tilde{\mathbf{K}})^{-1}\tilde{\mathbf{K}}^T\mathbf{K}_C^{-1}\right)\mathbf{K}_C\mathbf{K}_C\mathbf{a} \Rightarrow \\
\lambda\mathbf{a} &= \left(\frac{1}{r}\mathbf{I} - \frac{1}{r^2}\mathbf{K}_C^{-1}\tilde{\mathbf{K}}\left(\mathbf{I} + \frac{1}{r}\tilde{\mathbf{K}}^T\mathbf{K}_C^{-1}\tilde{\mathbf{K}}\right)^{-1}\tilde{\mathbf{K}}^T\right)\mathbf{K}_C\mathbf{a}. \qquad (45)
\end{aligned}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIFS.2016.2582562, IEEE Transactions on Information Forensics and Security

Multiplying both sides of (45) with $\mathbf{K}_C$ we get:

$$\lambda \mathbf{K}_C \mathbf{a} = \Big(\frac{1}{r}\mathbf{K}_C - \frac{1}{r^2}\tilde{\mathbf{K}}(\mathbf{I} + \frac{1}{r}\tilde{\mathbf{K}}^T\mathbf{K}_C^{-1}\tilde{\mathbf{K}})^{-1}\tilde{\mathbf{K}}^T\Big)\mathbf{K}_C\mathbf{a}. \tag{46}$$

Substituting $\mathbf{K}_C\mathbf{a} = \mathbf{q}$ in (46) we obtain:

$$\begin{aligned}
\lambda\mathbf{q} &= \Big(\frac{1}{r}\mathbf{K}_C - \frac{1}{r^2}\tilde{\mathbf{K}}(\mathbf{I} + \frac{1}{r}\tilde{\mathbf{K}}^T\mathbf{K}_C^{-1}\tilde{\mathbf{K}})^{-1}\tilde{\mathbf{K}}^T\Big)\mathbf{q} \Rightarrow \\
\mathbf{K}_C\mathbf{q} &= \frac{\lambda}{r}\tilde{\mathbf{K}}(\mathbf{I} + \frac{1}{r}\tilde{\mathbf{K}}^T\mathbf{K}_C\tilde{\mathbf{K}}^T)^{-1}\tilde{\mathbf{K}}^T\mathbf{q} \Rightarrow \\
\frac{\lambda}{r}\mathbf{q} &= \Big(\tilde{\mathbf{K}}(\mathbf{I} + \frac{1}{r}\tilde{\mathbf{K}}^T\mathbf{K}_C^{-1}\tilde{\mathbf{K}})\tilde{\mathbf{K}}^T\Big)^{-1}\mathbf{K}_c\mathbf{q} \Rightarrow \\
\mathbf{B}\mathbf{q} &= \mu\mathbf{q},
\end{aligned} \tag{47}$$

where $\mu = \frac{\lambda}{r}$ and $\mathbf{B} = \Big(\tilde{\mathbf{K}}(\mathbf{I} + \frac{1}{r}\tilde{\mathbf{K}}^T\mathbf{K}_C^{-1}\tilde{\mathbf{K}})\tilde{\mathbf{K}}^T\Big)^{-1}\mathbf{K}_c$.

The matrix $\mathbf{B}$ has the same eigenvalues with the matrix:

$$\tilde{\mathbf{B}} = \Big(\tilde{\mathbf{K}}^T\mathbf{K}_C^{-1}\tilde{\mathbf{K}}\Big)^{-1}\Big(\mathbf{I} + \frac{1}{r}\tilde{\mathbf{K}}^T\mathbf{K}_C^{-1}\tilde{\mathbf{K}}\Big) = \mathcal{K}^{-1}\Big(\mathbf{I} + \frac{1}{r}\mathcal{K}\Big), \tag{48}$$

where $\mathcal{K} \in \mathbb{R}^{N\times N}$ is the Nyström-based approximation of the original kernel matrix $\mathbf{K}$ using the client vectors as reference data. In order to show this, let us define the matrices $\mathbf{C} = \tilde{\mathbf{K}}$, $\mathbf{F} = \mathbf{K}_C^{-1}$ and $\mathbf{G} = (\mathbf{I} + \frac{1}{r}\tilde{\mathbf{K}}^T\mathbf{K}_C^{-1}\tilde{\mathbf{K}})^{-1}$. Then $\mathbf{B} = (\mathbf{P}\mathbf{M}^T)^{-1}$, where $\mathbf{M} = \mathbf{F}\mathbf{C}$ and $\mathbf{P} = \mathbf{C}\mathbf{G}$. The matrices $\mathbf{P}$ and $\mathbf{M}$ are symmetrizable [51]. It hods:

$$(\mathbf{P}\mathbf{M}^T)^{-1} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^T \tag{49}$$

$$(\mathbf{M}^T\mathbf{P})^{-1} = \mathbf{V}\mathbf{\Sigma}\mathbf{V}^T, \tag{50}$$

where $\tilde{\mathbf{K}} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T$. Thus, the relationship between eigenvectors of $\mathbf{B}$ and $\tilde{\mathbf{B}}$ can be given by:

$$\tilde{\mathbf{K}} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T \Rightarrow \tilde{\mathbf{K}}\mathbf{V} = \mathbf{U}\mathbf{\Lambda} \Rightarrow \mathbf{U} = \tilde{\mathbf{K}}\mathbf{V}\mathbf{\Lambda}^{-1}. \tag{51}$$

and

$$\tilde{\mathbf{K}} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T \Rightarrow \mathbf{U}^T\tilde{\mathbf{K}} = \mathbf{\Lambda}\mathbf{V}^T \Rightarrow \mathbf{U} = \tilde{\mathbf{K}}^T\mathbf{U}\mathbf{\Lambda}^{-1}. \tag{52}$$

After solving (47) we obtain the eigenvectors $\mathbf{q}_k$, $k = 1, \ldots, d$. Then, we solve for $\mathbf{a}_k = \mathbf{K}_C^{-1}\mathbf{q}_k$, $k = 1, \ldots, d$.

REFERENCES

[1] P. Barr, J. Noble, and R. Biddle, "Video game values: Human-computer interaction and games," *Interacting with Computers*, vol. 19, no. 2, pp. 180–195, 2007.

[2] Z. Li, U. Park, and A. Jain, "A discriminative model for age invariant face recognition," *IEEE Tran. Inf. Forens. and Sec.*, vol. 6, no. 3, pp. 1028–1037, 2011.

[3] A. Tefas and I. Pitas, "Human centered interfaces for assisted living," *Int. Conf. Man-Mach. Inter.*, 2011.

[4] Z. Li, S. Liao, A. Jain, and S. Li, "Coupled discriminant analysis for heterogeneous face recognition," *IEEE Tran. Inf. Forens. and Sec.*, vol. 7, no. 6, pp. 1707–1716, 2012.

[5] R. Duda, P. Hart, and D. Stork, *Pattern Classification, 2nd ed.* Wiley-Interscience, 2000.

[6] J. Ye, "Least squares linear discriminant analysis," *Int. Conf. Mach. Learn.*, pp. 1087–1093, 2007.

[7] S. Ji and J. Ye, "Generalized linear discriminant analysis: A unified framework and efficient model selection," *IEEE Tran. Neur. Netw.*, vol. 19, no. 10, pp. 1768–1782, 2008.

[8] J. Shuiwang and J. Ye, "Kernel uncorrelated and regularized discriminant analysis: A theoretical and computational study," *IEEE Tran. Knowl. Data Eng.*, vol. 20, no. 10, pp. 1311–1321, 2008.

[9] W. Yang, D. Dai, and H. Yan, "Feature extraction and uncorrelated discriminant analysis for high-dimensional data," *IEEE Tran. Knowl. Data Eng.*, vol. 20, no. 5, pp. 601–614, 2008.

[10] A. Iosifidis, A. Tefas, and I. Pitas, "On the optimal class representation in Linear Discriminant Analysis," *IEEE Tran. Neur. Netw. Learn. Syst.*, vol. 24, no. 9, pp. 1491–1497, 2013.

[11] Q. Liu, X. Tang, H. Lu, and S. Ma, "Face recognition using kernel scatter-difference-based discriminant analysis," *IEEE Tran. Neur. Net.*, vol. 17, no. 4, pp. 1081–1085, 2006.

[12] S. Zafeiriou, G. Tzimiropoulos, M. Petrou, and T. Stathaki, "Regularized kernel discriminant analysis with a robust kernel for face recognition and verification," *IEEE Tran. Neur. Net. Learn. Syst.*, vol. 23, no. 3, pp. 526–534, 2012.

[13] M. Siddiqi, R. Ali, A. Khan, Y. Park, and S. Lee, "Human facial expression recognition using stepwise linear discriminant analysis and hidden conditional random fields," *IEEE Tran. Im. Proc.*, vol. 24, no. 4, pp. 1386–1398, 2015.

[14] A. Iosifidis, A. Tefas, and I. Pitas, "Multidimensional sequence classification based on fuzzy distances and discriminant analysis," *IEEE Tran. Knowl. Data Eng.*, vol. 25, no. 11, pp. 2564–2575, 2012.

[15] ——, "Activity based person identification using fuzzy representation and discriminant learning," *IEEE Tran. Inf. Forens. Sec.*, vol. 7, no. 2, pp. 530–542, 2012.

[16] A. Iosifidis, A. Tefas, N. Nikolaidis, and I. Pitas, "Multi-view human movement recognition based on fuzzy distances and linear discriminant analysis," *Comp. Vis. Image Und.*, vol. 116, pp. 347–360, 2012.

[17] A. Iosifidis, A. Tefas, and I. Pitas, "Kernel reference discriminant analysis," *Patt. Rec. Lett.*, vol. 49, pp. 85–91, 2014.

[18] ——, "Discriminant bag of words based representation for human action recognition," *Patt. Rec. Lett.*, vol. 49, pp. 185–192, 2014.

[19] Y. Kittler, J. Li, and J. Matas, "Face verification using client specific Fisher faces," *Stat. Direct. Shapes Im.*, pp. 63–66, 2000.

[20] G. Goudelis, S. Zafeiriou, A. Tefas, and I. Pitas, "Class-specific kernel discriminant analysis for face verification," *IEEE Tran. Inf. Forens. Sec.*, vol. 2, no. 3, pp. 570–587, 2007.

[21] S. Zafeiriou, G. Tzimiropoulos, M. Petrou, and T. Stathaki, "Regularized kernel discriminant analysis with a robust kernel for face recognition and verification," *IEEE Tran. Neur. Netw.*, vol. 23, no. 3, pp. 526–534, 2012.

[22] S. Arashloo and J. Kittler, "Class-specific kernel fusion of multiple descriptors for face verification using multiscale binarised statistical image features," *IEEE Tran. Inf. Forens. and Sec.*, vol. 9, no. 12, pp. 2100–2109, 2014.

[23] A. Iosifidis, A. Tefas, and I. Pitas, "Class-specific reference discriminant analysis with application in human behavior analysis," *IEEE Tran. Hum. Mach. Syst.*, vol. 45, no. 3, pp. 315–326, 2015.

[24] K. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf, "An introduction to kernel-based learning algorithms," *IEEE Tran. Neur. Netw.*, vol. 12, no. 2, pp. 181–201, 2001.

[25] J. Lu, K. Plataniotis, and A. Venetsanopoulos, "Face recognition using kernel direct discriminant analysis algorithms," *IEEE Tran. Neur. Netw.*, vol. 14, no. 1, pp. 117–126, 2003.

[26] G. Baudat and F. Anouar, "Generalized discriminant analysis using a kernel approach," *Neural Computation*, vol. 12, pp. 2385–2404, 2000.

[27] W. Zheng, Z. Lin, and H. Wang, "L1-norm kernel discriminant analysis via bayes error bound optimization for robust feature extraction," *IEEE Tran. Neur. Netw. Learn. Syst.*, vol. 25, no. 4, pp. 793–805, 2013.

[28] D. Cai, X. He, and J. Han, "Srda: An efficient algorithm for large-scale discriminant analysis," *IEEE Tran. Knowl. Data Eng.*, vol. 20, no. 1, pp. 1–12, 2008.

[29] S. Ghorai, A. Mukherjee, and P. Dutta, "Discriminant analysis for fast multiclass data classification through regularized kernel function approximation," *IEEE Tran. Neur. Net.*, vol. 21, no. 6, pp. 1020–1029, 2010.

[30] A. Iosifidis, M. Gabbouj, and P. Pekki, "Class-specific nonlinear projections using class-specific kernel spaces," *IEEE Int. Conf. Big Data Science and Eng.*, pp. 1–8, 2015.

[31] K. Zhang, L. Lan, J. Kwok, S. Vucetic, and B. Parvin, "Scaling up graph-based semisupervised learning via Prototype Vector Machines," *IEEE Tran. Neur. Netw. Learn. Syst.*, vol. 26, no. 3, pp. 444–457, 2015.

[32] Y. Lee and S. Huang, "Reduced support vector machines: A statistical theory," *IEEE Tran. Neur. Net.*, vol. 18, no. 1, pp. 1–13, 2007.

[33] Y. Jia, F. Nie, and C. Zhang, "Trace ratio problem revisited," *IEEE Tran. Neur. Netw.*, vol. 20, no. 4, pp. 729–735, 2009.

[34] G. Stewart, *Matrix Algorithms Volume II: Eigensystems*. SIAM, 2001.

[35] A. Iosifidis, A. Tefas, and I. Pitas, "Large-scale nonlinear facial image classification based on approximate kernel extreme learning machine," *IEEE Int. Conf. Im. Proc.*, 2015.

[36] G. Golub and C. Loan, *Matrix Computations*. Johns Hopkins University Press, 3rd edition, 1996.

[37] K. Zhang and J. Kwok, "Clustered Nyström method for large scale manifold learning and dimensionality reduction," *IEEE Tran. Neur. Netw.*, vol. 21, no. 10, pp. 1576–1587, 2010.
[38] A. Martinez and A. Kak, "PCA versus LDA," *IEEE Tran. Pat. Anal. Mach. Intel.*, vol. 23, no. 2, pp. 228–233, 2001.
[39] E. Ortiza and B. Beckerb, "Face recognition for web-scale datasets," *Comp. Vis. Im. Underst.*, vol. 118, pp. 153–170, 2014.
[40] L. Wolf, T. Hassner, and I. Maoz, "Face recognition in unconstrained videos with matched background similarity," *Comp. Vision and Patt. Recogn.*, 2011.
[41] N. Kumar, A. Berg, P. Belhumeur, and S. Nayar, "Acquiriing linear subspaces for face recognition under variable lighting," *IEEE Tran. Pat. Anal. Mach. Intel.*, vol. 33, no. 10, pp. 1962–1977, 2011.
[42] G. Huang, M. Ramesh, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," *Tech. Report, University of Massachusetts*, 2007.
[43] G. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multi-class classification," *IEEE Tran. Syst. Man Cyb.:P Part B*, vol. 42, no. 2, pp. 513–529, 2012.
[44] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," *Adv. Neural Inf. Proc. Syst.*, pp. 1177–1184, 2007.
[45] M. Everingham, J. Sivic, and A. Zisserman, ""Hello! My name is...Buffy" - Automatic naming of characters in TV video," *Brit. Mach. Vision Conf.*, 2006.
[46] T. Ahonen, A. Hadid, and P. M., "Face description with local binary patterns: Application to face recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 12, pp. 2037–2041, 2006.
[47] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to humanlevel performance in face verification," *Comp. Vision and Patt. Recogn.*, 2014.
[48] M. Günther, L. El Shafey, and S. Marcel, "Face recognition in challenging environments: An experimental and reproducible research survey," in *Face Recognition Across the Imaging Spectrum*, 1st ed., T. Bourlai, Ed. Springer, 2016.
[49] S. Sonnenburg, G. Ratsch, C. Schafer, and B. Scholkopf, "Large scale multiple kernel learning," *J. Mach. Learn. Res.*, vol. 7, pp. 1531–1565, 2006.
[50] M. Gonen and E. Alpaydin, "Multiple kernel learning algorithms," *J. Mach. Learn. Res.*, vol. 12, pp. 2211–2268, 2011.
[51] N. Wermuth and H. Riissmann, "Eigenanalysis of symmetrizable matrix products: a result with statistical applications," *Scan. J. Statist*, vol. 20, pp. 361–367, 1993.