Enhancing Bilinear Subspace Learning by Element Rearrangement

Dong Xu, Shuicheng Yan, Stephen Lin, Thomas S. Huang, and Shih-Fu Chang

Abstract—The success of bilinear subspace learning heavily depends on reducing correlations among features along rows and columns of the data matrices. In this work, we study the problem of rearranging elements within a matrix in order to maximize these correlations so that information redundancy in matrix data can be more extensively removed by existing bilinear subspace learning algorithms. An efficient iterative algorithm is proposed to tackle this essentially integer programming problem. In each step, the matrix structure is refined with a constrained *Earth Mover's Distance* procedure that incrementally rearranges matrices to become more similar to their low-rank approximations, which have high correlation among features along rows and columns. In addition, we present two extensions of the algorithm for conducting supervised bilinear subspace learning. Experiments in both unsupervised and supervised bilinear subspace learning demonstrate the effectiveness of our proposed algorithms in improving data compression performance and classification accuracy.

Index Terms—Bilinear subspace learning, element rearrangement, earth mover's distance, dimensionality reduction.

1 Introduction

Image data naturally contain a significant amount of information redundancy, as evidenced by spatial coherence and structural commonalities found within images and image sets. For processing and analysis of images, it is often advantageous to pare away these redundancies so that the intrinsic features of the data are revealed. This is the goal of dimensionality reduction (or subspace analysis) techniques, which aim to decrease the size of a feature space by removing correlations among the features. Dimensionality reduction has proven to be useful for unsupervised learning tasks such as data compression and facilitates supervised learning by identifying fewer but effective features.

Frequently used dimensionality reduction techniques such as Principal Components Analysis (PCA) [1], Linear Discriminant Analysis (LDA) [2], and Tensorfaces [3] commonly unfold each image into a single column vector, which we refer to as an *image-as-vector* representation in this work. While correlations among different pixels can be reduced in these vector-based techniques, lengthy column vectors typically result in the curse of dimensionality and classification performance may be degraded because of the small-sample-size problem.

Recently, a large number of works (e.g., [4], [5], [6], [7], [8], [9], [10], [11], [12]) have sought to process image data in their original form (e.g., images as matrices instead of as vectors), which we refer to as an *image-as-matrix* representation in this work. By aiming to reduce correlations only within image rows and columns, rather than among all pixels in an image, the curse of dimensionality is avoided and the small-sample-size problem becomes greatly diminished. For image data represented in matrix form [11], [12], appreciably higher recognition performance has been experimentally reported, especially in cases with small training sets. Similar improvements have also been found for the more general case of tensor data, where correlations are removed along column vectors of mode-k flattened matrices [5], [6], [7], [8], [9].

Correlations within 2D image data, however, are not limited to the elements along certain matrix dimensions. Work on natural image statistics indicates that such correlations are frequently present among different regions both spatially within an image and temporally through an image sequence [13]. For enhancement of bilinear subspace learning, a natural question that arises is whether these correlations can be reduced while preserving the performance benefits of processing image data in their original matrix form.

In this paper, we propose to address this problem by rearranging the elements within matrices to increase the correlations among features along rows and columns, which we will refer to as intramatrix correlations. An illustration of element rearrangement is shown in Fig. 1. By aligning elements in a way that increases correlations along the rows and columns of a matrix, greater reductions in dimensionality can be achieved with existing bilinear subspace analysis methods.

We show in Section 2 that rearranging matrix elements to maximize intramatrix correlations is essentially an *integer programming* problem with a nonlinear objective function. Since this problem is NP-hard [14], we present in this work an approximate iterative solution. As described in Section 3, we first employ Generalized Low-Rank Approximation (GLRAM) [11] or Concurrent Subspace Analysis (CSA) [8][1] to compute projection matrices from the training data matrices, then the elements in these matrices are rearranged to become more similar to the reconstructions of the training matrices by the projection matrices. This procedure is then iterated using the rearranged matrices. Reconstructed matrices are used to guide the rearrangement process because they are similar to the training data matrices and have high correlation among features along certain rows or columns. For greater computational feasibility, the displacement of elements within a matrix is constrained in each iteration to a local neighborhood or a nearest feature-based neighborhood. Element rearrangement is formulated as a constrained *Earth Mover's Distance* [15] problem, where the flows from the elements of the original matrix to those of the target matrix naturally constitute a pure network flow model [16]. An integer solution can then be reached using a general linear programming technique, e.g., the Simplex method.

Since rearrangement is employed as a preprocessing step to increase intramatrix correlations, it can be used in conjunction with any bilinear subspace analysis technique. For supervised subspace learning, in Section 3, we extend the proposed algorithm using 2DLDA [12] and 2DMFA [17] as examples. In the extension, the weighted and centered class means (for 2DLDA) or the difference matrices of the nearest margin pairs between different classes (for 2DMFA) are used as training samples, and the element rearrangement algorithm is then performed on these new matrices. With this approach, the most discriminant information becomes encoded in the first few dimensions of the derived subspace.

- *D. Xu is with the School of Computer Engineering, Nanyang Technological University, 50 Nanyang Avenue, Blk N4, Singapore 639798, Singapore. E-mail: dongxu@ntu.edu.sg.*
- *S. Yan is with the Department of Electrical and Computer Engineering, National University of Singapore, 4 Engineering Drive 3, Singapore 117576, Singapore. E-mail: eleyans@nus.edu.sg.*
- *S. Lin is with Microsoft Research Asia, 5F, Beijing Sigma Center, No. 49, Zhichun Road, Haidian District, Beijing 100080, PR China. E-mail: stevelin@microsoft.com.*
- *T.S. Huang is with Beckman Institute, University of Illinois at Urbana-Champaign, 405 North Mathews Avenue, Urbana, IL 61801. E-mail: huang@ifp.uiuc.edu.*
- *S.-F. Chang is with the Department of Electrical Engineering, Columbia University, 500 W. 120th St. Rm 1312, New York, NY 10027. E-mail: sfchang@ee.columbia.edu.*

Manuscript received 13 June 2008; revised 27 Oct. 2008; accepted 20 Jan. 2009; published online 23 Feb. 2009.
Recommended for acceptance by S. Chaudhuri.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2008-06-0349.
Digital Object Identifier no. 10.1109/TPAMI.2009.51.

1. CSA and GLRAM were independently proposed. CSA is a generalization of GLRAM for general tensor data of arbitrary order.

0162-8828/09/$25.00 © 2009 IEEE Published by the IEEE Computer Society
Authorized licensed use limited to: Nanyang Technological University. Downloaded on August 24, 2009 at 08:33 from IEEE Xplore. Restrictions apply.

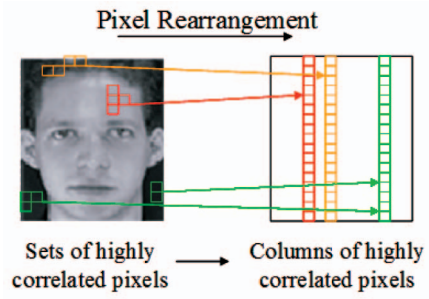

Fig. 1. An illustration of element rearrangement for a data matrix. Note that the correlations along both the rows and columns are enhanced after element arrangement.

Finally, we demonstrate in Section 4 the utility of element rearrangement for data compression and supervised bilinear subspace learning. The experiments clearly demonstrate that element rearrangement can improve the performance of previous bilinear subspace analysis algorithms. While this paper primarily focuses on matrices (i.e., second order tensors), the algorithms and analysis presented here can be easily extended to handle higher order tensor input, such as Gabor-filtered images or video sequences, by tensorization in the graph embedding framework [17].

2 Problem Formulation and Iterative Solution

In this section, we first briefly review GLRAM [11] and CSA [8]. We then introduce our formulation and solution of the element rearrangement problem for maximizing intramatrix correlation.

2.1 Problem Formulation

We express each element of the training sample set in matrix form as $X_i \in \mathbb{R}^{m\times n}$, $i = 1, 2, \ldots, N$, where m and n are the height and width of the data matrix and N is the number of samples. We also define the norm of matrix X as

$$\|X\| = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} X_{ij}^2}$$

and denote $U \in \mathbb{R}^{m\times m'}$ and $V \in \mathbb{R}^{n\times n'}$ as two projection matrices, in which m' and n' are the height and the width in the low-dimensional space. For each input matrix X_i, the low-dimensional representation is $Y_i = U^T X_i V$ and the reconstructed image is $X_i^{REC} = UU^T X_i V V^T$. The objective function of GLRAM in [11], and also of the second order tensor version of CSA in [8], aims to minimize reconstruction error, i.e.,

$$(U, V)^* = arg\min_{(U,V)} \sum_{i=1}^{N} \|UU^T X_i V V^T - X_i\|^2. \quad (1)$$

There is no closed-form solution for the above objective function, so an iterative procedure is proposed in [8], [11]. As shown in [8], [11], 2DPCA [10] is a special case of [8], [11] by setting $U = I$ and assuming the mean of all the samples to be zero, i.e., $\sum_{i=1}^{N} X_i = 0$. The projection matrix V in 2DPCA can be directly solved by eigenvalue decomposition. Further details on GLRAM, CSA, and 2DPCA can be found in [8], [11], and [10].

In this work, we utilize GLRAM or CSA in formulating the problem of element rearrangement. For ease of understanding, we denote the position of a matrix element as (r^p, c^p) and its global index as p, where $p = (r^p - 1)n + c^p$. Similarly, the position (r^q, c^q) corresponds to the global index q. We denote $(X_i)_{r^p,c^p}$ or $(X_i)_p$ as the value of the entry in matrix X_i at position (r^p, c^p) or p. We also define the rearrangement operator (or permutation matrix) as $R \in \{0,1\}^{(m\times n)\times(m\times n)}$ and $(X_i(R))_q = (X_i)_p$ if $R_{pq} = 1$. For operator R, we have the properties $\sum_p R_{pq} = 1$ and $\sum_q R_{pq} = 1$, meaning that the elements in matrices X_i and $X_i(R)$ have a one-to-one correspondence. In element rearrangement, we wish to minimize the objective function $F(U, V, R)$ expressed as

$$F(U, V, R) = \sum_{i=1}^{N} \|X_i(R) - UU^T X_i(R) V V^T\|^2. \quad (2)$$

Essentially, this function seeks *rearranged* matrices that best approximate their reconstructed low-rank matrices, which have high correlation among features along rows and columns.

This objective function presents a complicated integer programming problem with nonlinear objectives. Since this problem is NP-hard [14], we present in the following section an approximate solution to iteratively compute the projection matrices U, V and the rearrangement operator R. Note that when R is fixed, U and V can be computed with GLRAM or CSA [11], [8]. In the following sections, we therefore focus on how to compute the rearrangement operator R.

2.2 Solution under Spatially Local Neighborhood Constraints

Given the projection matrices U and V, minimization of $F(U, V, R)$ with respect to element rearrangement operator R is an integer programming problem. For the tth iteration step, let $\{X_i^t, i = 1, 2, \ldots, N\}$ be the matrix data rearranged from X_i^{t-1}. GLRAM or CSA can be applied on X_i^t to compute the projection matrices U and V, then the reconstructed matrix at the tth iteration can be computed as

$$X_i^{tREC} = UU^T X_i^t V V^T. \quad (3)$$

Constraints should be imposed to facilitate optimization; otherwise, the total number of parameters would balloon to $4{,}096^2$ for a matrix of size 64×64, which is computationally prohibitive both in complexity and memory requirements. The first constraint is that, at each iteration, each element p of a matrix can be moved only within its spatially local neighborhood LN_p^γ, bounded by a distance of $2\sqrt{2}$ pixels in this work. Thus, $R_{pq} = 0$ if $q \notin LN_p^\gamma$ and only about $\gamma = 25$ pixel locations are under consideration for each p. In each step, we fix the term $UU^T X_i(R) V V^T$ in (2) to be X_i^{tREC} because, after stepwise pixel rearrangement, the value of the objective function will be further reduced if we substitute $UU^T X_i(R) V V^T$ for X_i^{tREC} according to the new operator R (See Section 2.5 for justification). Then, the objective function in (2) can be rewritten as

$$F(R) = \sum_{i=1}^{N} \|X_i^t(R) - X_i^{tREC}\|^2. \quad (4)$$

Let us define

$$f_{pq} = \sum_{i=1}^{N} \left\|(X_i^t)_{r^p,c^p} - (X_i^{tREC})_{r^q,c^q}\right\|^2. \quad (5)$$

Since $R_{pq} \in \{0, 1\}$, the objective function can then be expressed as

$$F(R) = \sum_{p}\sum_{q\in LN_p^\gamma} f_{pq} R_{pq}. \quad (6)$$

The correspondence between the elements of the original matrix and the rearranged matrix must be one-to-one, which imposes the following constraints:

$$\sum_{p;q\in LN_p^\gamma} R_{pq} = 1, \forall\ q, \quad \sum_{q;p\in LN_q^\gamma} R_{pq} = 1, \forall\ p. \quad (7)$$

Authorized licensed use limited to: Nanyang Technological University. Downloaded on August 24, 2009 at 08:33 from IEEE Xplore. Restrictions apply.

The problem is then simplified into an integer programming problem with a linear objective function and linear constraints. If we relax the integer constraints for the elements R_{pq}, then the problem becomes a constrained *Earth Mover's Distance* [15] problem (i.e., linear programming), outlined as follows:

$$\begin{aligned} &\arg\min_{R_{pq}} \sum_{p} \sum_{q \in LN_p^\gamma} f_{pq} R_{pq}, s.t. \\ &1.\ 0 \le R_{pq} \le 1; \\ &2.\ \sum_{p; q \in LN_p^\gamma} R_{pq} = 1, \forall\ q; \\ &3.\ \sum_{q; p \in LN_q^\gamma} R_{pq} = 1, \forall\ p. \end{aligned} \tag{8}$$

The linear programming problem described in (8) will have an integer solution, as guaranteed by the following theorem.

Theorem 1 [14], [16]. *An integer programming problem*

$$\begin{aligned} &\arg\min_{R_{pq}} \sum_{p,q} f_{pq} R_{pq},\ s.t. \\ &\sum_{q} R_{pq} = 1, \forall\ p, \quad \text{and} \quad \sum_{p} R_{pq} = 1, \forall\ q \end{aligned} \tag{9}$$

can be solved as a linear programming problem with the constraints $0 \le R_{pq} \le 1, \forall\ p, q$.

The equivalence of integer programming and linear programming arises from the unimodular coefficient matrix of the integer programming problem in (9) and is based on the assumption that linear programming is solved with the simplex method. Further details on this theorem can be found in [14]. It is straightforward to prove that our proposed spatially constrained procedure in (8) is equivalent to (9) with $f_{pq} = +\infty$ for $q \notin LN_p^\gamma$. Hence, the linear programming solution in (8) is also an integer solution, meaning that for any given p (or q), there exists only one R_{pq} that is equal to 1 for all q (or p). Consequently, the element rearrangement operator R_{pq} constructs a one-to-one correspondence between the original and target matrices.

Based on the derived solution R, we rearrange the matrix data and update the training samples to obtain X_i^{t+1}. Then, GLRAM or CSA and the proposed procedure are repeated until convergence, which occurs when R contains no more element movement, i.e., $R_{pq} = 0$ if $p \neq q$. The overall solution method is outlined in Algorithm 1.

Algorithm 1. Algorithm for Element Rearrangement
Given sample matrices $\{X_1, X_2, \ldots, X_N\}$ and a size (m', n') for the low-dimensional matrices.

1: Initialize $X_i^1 = X_i$, for $i = 1, \ldots, N$;
2: for $t = 1, 2, \ldots$
  a: Based on the matrix data $X_i^t, i = 1, 2, \ldots, N$, compute U^t and V^t, using GLRAM [11] or CSA [8]. If $t > 1, U^t$ and V^t are initialized as U^{t-1} and V^{t-1} in GLRAM or CSA; otherwise, they are initialized as identity matrices according to [11], [8].
  b: Compute the rearrangement operator R by solving the linear programming problem in (8) or (10).
  c: Update the matrix X_i^t to X_i^{t+1} according to R.
  d: If $R_{pq} = 0$ for all $p \neq q$, then exit.
3: Output the rearranged data $\{X_i^t, i = 1, \ldots, N\}$.

2.3 Solution under Feature-Based Nearest Neighbor Constraints

To efficiently minimize the objective function in (4), another natural and more elegant approach is to constrain the rearrangement of an element p to only its nearest neighbors in the feature domain, namely, by f_{pq} in (5). For each element p, we define the set NN_p^γ (with $\gamma = 25$ in our implementation) as p plus the $\gamma - 1$ nearest neighbors of p measured in the feature domain. We refer to the new constraints as feature-based nearest neighbor constraints in this work. We include the position p in the set NN_p^γ to guarantee a one-to-one mapping. Without adding p, it is possible for p to be moved to another position without its original position being filled by another element.

Note that all the analysis and theorems presented for solution under spatial neighborhood constraints can be directly applied to this case based on feature neighborhood constraints. We can reformulate another constrained *Earth Mover's Distance* [15] problem under feature neighborhood constraints in (10). When compared with (8), we only change LN_p^γ to NN_p^γ. Also, Algorithm 1 is applicable for computing U, V, and R for this new solution:

$$\begin{aligned} &\arg\min_{R_{pq}} \sum_{p} \sum_{q \in NN_p^\gamma} f_{pq} R_{pq}, s.t. \\ &1.\ 0 \le R_{pq} \le 1; \\ &2.\ \sum_{p; q \in NN_p^\gamma} R_{pq} = 1, \forall\ q; \\ &3.\ \sum_{q; p \in NN_q^\gamma} R_{pq} = 1, \forall\ p. \end{aligned} \tag{10}$$

The above constraints in (8) and (10) are used mainly to facilitate the computation of the linear programming problem. Without these constraints, the number of parameters would be $4{,}096^2$ for a matrix of size 64×64, which is computationally prohibitive both in complexity and memory. Although movement is constrained in each iteration within a local spatial neighborhood or nearest neighbors in the feature domain, arbitrary movement among pixels is possible after a sequence of iterations.

2.4 Complexity Analysis

We discuss complexity as follows: Each step iterates between GLRAM (or CSA) and linear programming for element rearrangement. For a matrix of size 64×64 and $\gamma = 25$, there exists about $4{,}096 \times 25 = 102{,}400$ parameters[2] and $4{,}096 \times 2 = 8{,}192$ constraints. The constraint matrix is very sparse. Currently, linear programming with the simplex method is the most time-consuming part. According to [15], the complexity is smaller than $O(h^3 log(h))$, where h is the size of the image.

The linear programming problem in (8) can be rapidly processed. In our experiments on a 2.8 G CPU with 1.0 G memory, the training time for each iteration is about 30 seconds with unoptimized Matlab code, and algorithm convergence is reached in about 20 iterations. The linear programming problem in (10) is relatively slow. Under the same experimental configuration, the training time for each iteration is about 450 seconds, and algorithm convergence is reached after about 25 iterations. The main additional computation cost comes from searching for the most similar neighbors of each position.

2.5 Convergency Justification

To prove the convergence of Algorithm 1, we will make use of an auxiliary function. We define $X_i|_{i=1}^N$ as $X_1, X_2, \ldots, X_N$ and $X_i'|_{i=1}^N$ as $X_1', X_2', \ldots, X_N'$. Let

2. The actual number of parameters in Algorithm 1 is 98,596 because some neighboring pixels do not exist for border pixels.

Authorized licensed use limited to: Nanyang Technological University. Downloaded on August 24, 2009 at 08:33 from IEEE Xplore. Restrictions apply.

$$F(X_i|_{i=1}^N) = \sum_{i=1}^N \|X_i - UU^T X_i VV^T\|^2,$$
$$G(X_i|_{i=1}^N, X_i'|_{i=1}^N) = \sum_{i=1}^N \|X_i - U'U'^T X_i' V'V'^T\|^2, \quad (11)$$

where (U, V) are computed from $X_i|_{i=1}^N$ by minimizing $\sum_{i=1}^N \|X_i - UU^T X_i VV^T\|^2$ with the GLRAM or CSA algorithm and, similarly, (U', V') are computed from $X_i'|_{i=1}^N$ by minimizing $\sum_{i=1}^N \|X_i' - U'U'^T X_i' V'V'^T\|^2$ with the GLRAM or CSA algorithm.

Definition. *The function $G(X_i|_{i=1}^N, X_i'|_{i=1}^N)$ is an auxiliary function for $F(X_i|_{i=1}^N)$ if the following two conditions are satisfied: 1) $F(X_i|_{i=1}^N) \leq G(X_i|_{i=1}^N, X_i'|_{i=1}^N)$ and 2) $G(X_i|_{i=1}^N, X_i|_{i=1}^N) = F(X_i|_{i=1}^N)$.*

Theorem 2. *The function $G(X_i|_{i=1}^N, X_i'|_{i=1}^N)$ defined above is an auxiliary function for $F(X_i|_{i=1}^N)$.*

Proof. The second condition for an auxiliary function is clearly satisfied, so we prove here that the first condition also holds. For *fixed* $X_i|_{i=1}^N$, we also define another function:

$$Q(\tilde{Z}_i|_{i=1}^N, \tilde{U}, \tilde{V}) = \sum_{i=1}^N \|X_i - \tilde{U}\tilde{Z}_i\tilde{V}^T\|^2,$$

where $\tilde{Z}_i \in \mathbf{R}^{m' \times n'}$ is the lower-dimensional representation and $\tilde{U} \in \mathbf{R}^{m \times m'}$ and $\tilde{V} \in \mathbf{R}^{n \times n'}$ are two projection matrices. Note that with *fixed* $X_i|_{i=1}^N$ and a given $(\tilde{U}, \tilde{V})$, the optimum $\tilde{Z}_i|_{i=1}^N$ is given by $\tilde{Z}_i|_{i=1}^N = \tilde{U}^T X_i \tilde{V}|_{i=1}^N$. For more details, please refer to the prior works on GLRAM and CSA. So, we have

$$F(X_i|_{i=1}^N) \leq Q(U'^T X_i V'|_{i=1}^N, U', V') \leq Q(U'^T X_i' V'|_{i=1}^N, U', V') = G(X_i|_{i=1}^N, X_i'|_{i=1}^N).$$

Note that U and V are optimal by minimizing the objective function $F(X_i|_{i=1}^N) = \sum_{i=1}^N \|X_i - UU^T X_i VV^T\|^2$ in (11). From the first term $F(X_i|_{i=1}^N)$ to the second one $Q(U'^T X_i V'|_{i=1}^N, U', V')$, the optimal U and V in (11) are replaced by other projection matrices U' and V', so we can conclude that the first term will not be larger than the second one. Similarly, the second term is not larger than the third one, since, for given U' and V', the optimal $\tilde{Z}_i|_{i=1}^N = U'^T X_i V'|_{i=1}^N$ are replaced with $U'^T X_i' V'|_{i=1}^N$ in the third item.

$G(X_i|_{i=1}^N, X_i'|_{i=1}^N)$ is therefore an auxiliary function for $F(X_i|_{i=1}^N)$ □

If the auxiliary function takes the parameters $X_i|_{i=1}^N$ and $X_i'|_{i=1}^N$ as the rearranged matrices $X_i^{t+1}|_{i=1}^N$ and $X_i^t|_{i=1}^N$ from two successive steps in Algorithm 1, respectively, we have the following Theorem 3.

Theorem 3. *From Algorithm 1 and (8) (or (10)), the objective function $F(X_i^t|_{i=1}^N)$ will monotonically decrease until convergence.*

Proof. If we set X_i^{t+1} as X_i and X_i^t as X_i', respectively, from Theorem 2, we have $G(X_i^{t+1}|_{i=1}^N, X_i^{t+1}|_{i=1}^N) \leq G(X_i^{t+1}|_{i=1}^N, X_i^t|_{i=1}^N)$. Note in this case, $U'U'^T X_i' V'V'^T$ in (11) is $X_i^{t\,REC}$. Therefore, $G(X_i^{t+1}|_{i=1}^N, X_i^t|_{i=1}^N)$ is the objective function in (4), in which $X_i^{t+1} = X_i^t(R)$. The objective function in (8) or (10) aims to minimize the objective function in (4), so we have $G(X_i^{t+1}|_{i=1}^N, X_i^t|_{i=1}^N) \leq G(X_i^t|_{i=1}^N, X_i^t|_{i=1}^N)$. Then, we can conclude that $0 \leq F(X_i^{t+1}|_{i=1}^N) \leq F(X_i^t|_{i=1}^N)$, namely, $F(X_i^t|_{i=1}^N)$ will monotonically decrease until convergence. □

2.6 Connection with Previous Work

Jebara proposed a series of works [18], [19], [20], [21] to estimate an optimal permutation matrix for each individual image so as to achieve the permutation invariant property for PCA, convex learning, Kernel PCA (KPCA), and Support Vector Machine (SVM). Our work is intrinsically different with Jebara's work in the following aspects: 1) In Jebara's works, an images is represented as a collection of (X, Y, I) pixel vectors and, hence, these algorithms belong to the *image-as-vector* category. In contrast, our work deals with bilinear subspace analysis algorithms (e.g., 2DLDA, 2DMFA, 2DPCA, GLRAM), in which each image is represented in its intrinsic form (i.e., a matrix) and, hence, belongs to the *image-as-matrix* category. 2) Jebara's work aims to align multiple different images *in an image-dependent way* (i.e., *each individual image has its own permutation matrix*). Our work aims to reorganize the data matrix structure *in an image-independent way* by pursuing only a *single permutation matrix for all the images* to better utilize the intramatrix correlations for bilinear subspace analysis. Therefore, the motivations are intrinsically different. Potentially, our work could be further extended to enhance intramatrix correlations by aligning multiple images with individual permutation matrices, which will be investigated in our future work.

3 Extension for Supervised Subspace Learning

The purpose of the element rearrangement algorithm is to enhance intramatrix correlations, which is useful for matrix data compression. Another important task of subspace learning is to derive low-dimensional representations with strong discriminative power for different data classes. In this section, we examine how element rearrangement can enhance the discriminating power of bilinear subspaces.

For the task of classification, the class labels of training samples X_i are denoted by $c_i \in \{1, 2, \ldots, N_c\}$, where N_c is the total number of classes and n_c is the number of samples in the cth class. Here, we discuss two popular supervised subspace learning algorithms, namely, LDA and Marginal Fisher Analysis (MFA) [17]. As discussed in [12], [17], LDA and MFA can both be extended to handle matrix data, referred to as 2DLDA and 2DMFA here. In the following, we take 2DLDA and 2DMFA as examples of enhancing discriminative ability by element rearrangement for supervised bilinear subspace learning.

3.1 Extension for 2DLDA

LDA seeks a lower dimensional representation that minimizes intraclass scatter and at the same time, maximizes interclass scatter. Let the vector representation of the training data be denoted by $\{x_1, x_2, \ldots, x_N\}$. LDA and 2DLDA can then be, respectively, expressed as

$$\max_P \frac{\sum_{c=1}^{N_c} n_c \|P^T(\bar{x}_c - \bar{x})\|^2}{\sum_{i=1}^N \|P^T(x_i - \bar{x}_{c_i})\|^2}, \quad (12)$$

$$\max_{U,V} \frac{\sum_{c=1}^{N_c} n_c \|UU^T \bar{X}_c VV^T - UU^T \bar{X} VV^T\|^2}{\sum_{i=1}^N \|UU^T X_i VV^T - UU^T \bar{X}_{c_i} VV^T\|^2}, \quad (13)$$

where $\bar{X}_c$ and $\bar{x}_c$ represent the mean of samples in the cth class, while $\bar{X}$ and $\bar{x}$ denote the mean of all samples.

We develop a two-step procedure for improving algorithmic classification capability by element rearrangement. First, we compute the null space of the denominator of (12). And then, we reconstruct each data sample using P_N as

$$y_i = P_N P_N^T x_i. \quad (14)$$

We denote the corresponding matrix representation of the reconstructed data as $\{Y_i, i = 1, 2, \ldots, N\}$. For the reconstructed data, the intraclass scatter is zero; hence, the optimization of (13) is simplified to

Authorized licensed use limited to: Nanyang Technological University. Downloaded on August 24, 2009 at 08:33 from IEEE Xplore. Restrictions apply.

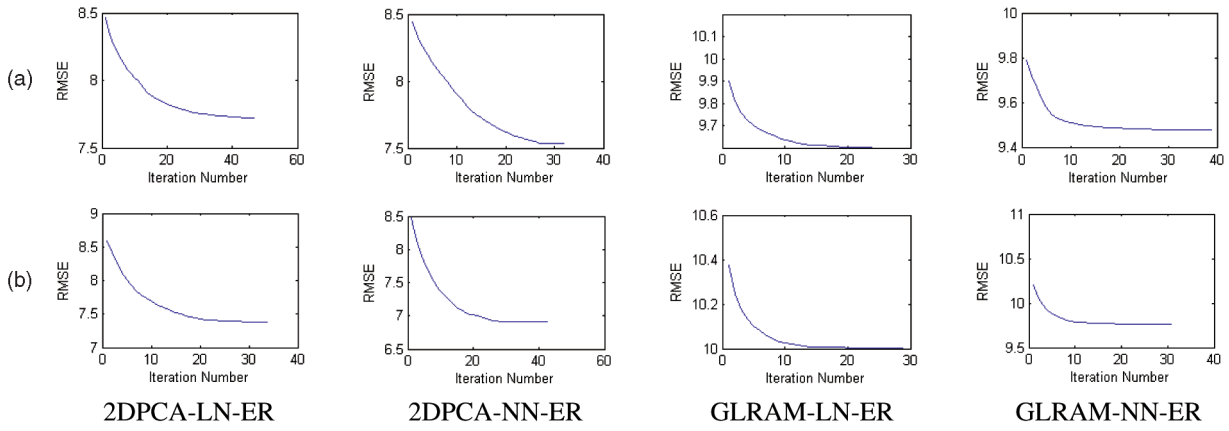

Fig. 2. Algorithm convergence with element rearrangement. RMSE versus number of iterations on (a) the CMU PIE and (b) FERET databases.

$$\max_{U,V} \sum_{c=1}^{N_c} n_c \|UU^T(\bar{Y}_c - \bar{Y})VV^T\|^2, \tag{15}$$

which is equivalent to minimizing

$$\sum_{c=1}^{N_c} n_c \|(\bar{Y}_c - \bar{Y}) - UU^T(\bar{Y}_c - \bar{Y})VV^T\|^2.$$

This objective function is the same as (1) except that the samples are the weighted difference matrices $\{\sqrt{n_c}(\bar{Y}_c - \bar{Y}), c = 1, 2, \ldots, N_c\}$. Algorithm 1 can, hence, be used here to minimize the objective function by element rearrangement.

3.2 Extension for 2DMFA

As shown in [17], LDA is motivated from the assumption that the data of each class follow a Gaussian distribution, which is not always satisfied in real-world problems. Moreover, interclass scatter does not well characterize the separability of different classes without the Gaussian distribution assumption. To avoid the need for the Gaussian distribution assumption, Yan et al. [17] proposed MFA and 2DMFA, which have the following objective function:

$$\max_{P} \frac{\sum_{c=1}^{N_c} \sum_{(i,j)\in N_{k_2}^-(c)} \left\|P^T x_i - P^T x_j\right\|^2}{\sum_{i=1}^{N} \sum_{j\in N_{k_1(i)}^+} \left\|P^T x_i - P^T x_j\right\|^2}, \tag{16}$$

$$\max_{U,V} \frac{\sum_{c=1}^{N_c} \sum_{(i,j)\in N_{k_2}^-(c)} \|U^T X_i V - U^T X_j V\|^2}{\sum_{i=1}^{N} \sum_{j\in N_{k_1(i)}^+} \|U^T X_i V - U^T X_j V\|^2}, \tag{17}$$

where $N_{k_1}^+(i)$ indicates the k_1 nearest neighbors of sample x_i within the same class and $N_{k_2}^-(c)$ denotes a set of margin pairs that are obtained as follows: For each class c, distances between its samples and samples of other classes are computed in the original feature space, then, for the k_2 smallest distances, the corresponding point pairs $\{(i,j), c_i = c, c_j \neq c\}$ are chosen.

Similarly, we also develop a two-step procedure for improving algorithmic classification capability by element rearrangement. First, we compute the null space of the denominator of (16), which we represent as P_N. In our experiments, we fix k_1 in (16) as $n_c - 1$ because in our experiments, the total number of samples in each class is the same, so P_N from LDA and MFA are the same. And then, we reconstruct each data sample using P_N as in (14). We denote the corresponding matrix representation of the reconstructed data as $\{Y_i, i = 1, 2, \ldots, N\}$. For the reconstructed data, the intraclass scatter is zero and we can optimize the following objective function:

$$\max_{U,V} \sum_{c=1}^{N_c} \sum_{(i,j)\in N_{k_2}^-(c)} \left\|UU^T(Y_i - Y_j)VV^T\right\|^2, \tag{18}$$

which is equivalent to minimizing

$$\sum_{c=1}^{N_c} \sum_{(i,j)\in N_{k_2}^-(c)} \left\|(Y_i - Y_j) - UU^T(Y_i - Y_j)VV^T\right\|^2.$$

So, the samples are the difference matrices of the k_2 (set to 25 in our experiments) nearest sample pairs between different classes.

3.3 Classification

After the pixel rearrangement process, 2DLDA and 2DMFA are then conducted on the rearranged training data to learn the projection matrices U and V. Data are projected to a lower dimension by the derived projection matrices, and then, classified with a proper classifier. In this work, we use the Nearest Neighbor classifier for simplicity.

4 Experiments

We present a set of experiments to verify the effectiveness of the matrix element rearrangement algorithms for both unsupervised and supervised tasks. For performance analysis on face image compression, we take as examples the two unsupervised learning algorithms GLRAM [11] (or CSA [8]) and 2DPCA [10], which is a special case of GLRAM (or CSA) that computes only one projection matrix. When element rearrangement is included with these algorithms, they are labeled as 2DPCA-LN-ER or GLRAM-LN-ER for the spatial neighborhood constraints and 2DPCA-NN-ER or GLRAM-NN-ER for the feature neighborhood constraints. For supervised learning, we take 2DLDA [12] and 2DMFA [17], which are the bilinear versions of LDA and MFA, respectively, as examples to examine performance on face recognition. We consider only spatial neighborhood constraints in these experiments. When element rearrangement is included, we refer to the algorithms as 2DLDA-LN-ER and 2DMFA-LN-ER.

We use the CMU PIE [22] and FERET [23] databases for experiments. For the CMU PIE database, we choose five near frontal poses ($C27$, $C05$, $C29$, $C09$, and $C07$) and illuminations indexed as 08, 11, 10, and 13. Due to incompleteness of data, only 63 peoples are used in this work, with each person having 20 images. We also test our algorithm on a subset of the FERET database. This subset includes 1,400 images of 200 individuals (each with seven images labeled as ba, bc, bd, be, bf, bg, and bh). All of the gray-level images are aligned by fixing the locations of the two eyes, normalizing in size to a resolution of 64×64 pixels, and

Authorized licensed use limited to: Nanyang Technological University. Downloaded on August 24, 2009 at 08:33 from IEEE Xplore. Restrictions apply.

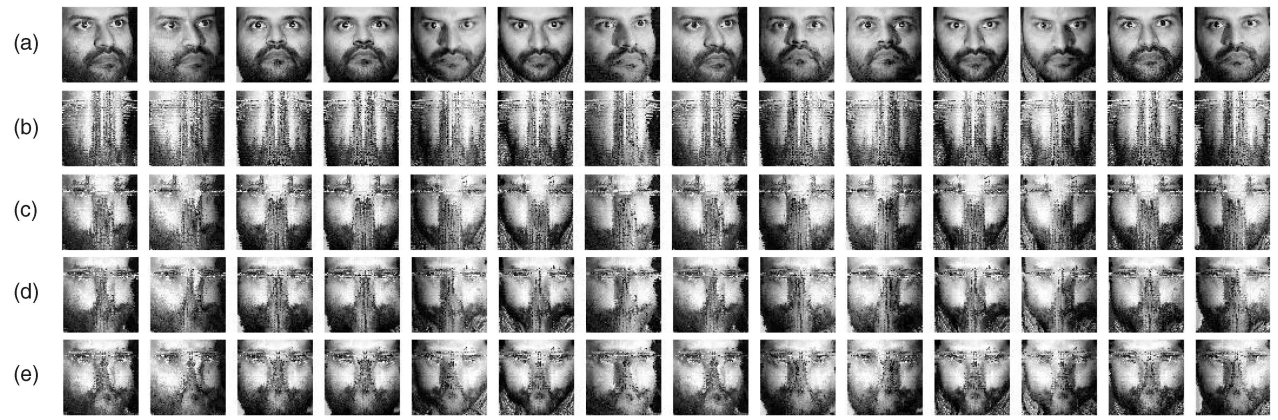

Fig. 3. Comparison of 14 element-rearranged images from the CMU PIE database with the GLRAM-LN-ER algorithm. (a) The original images. (b), (c), (d), and (e) The rearranged images for d = 2, 3, 4, and 5, respectively.

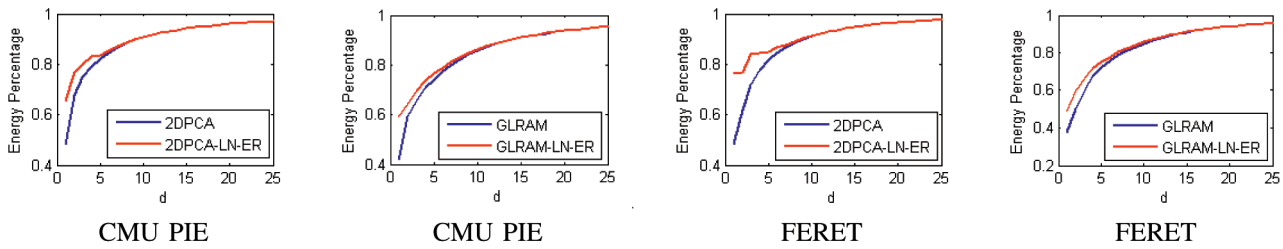

Fig. 4. Comparison of Energy Percentage of Eigenvalues for different d on the CMU PIE and FERET databases.

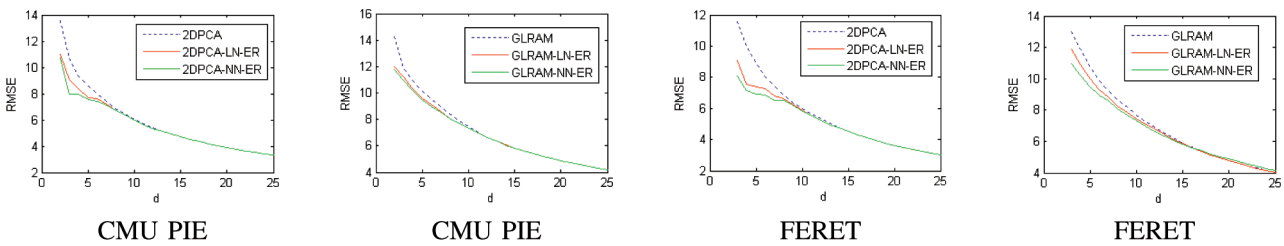

Fig. 5. RMSE for different values of d on the CMU PIE and FERET databases.

preprocessing with histogram equalization. We also normalize the gray-scale values by dividing by 255 and then subtracting 0.5. Typical examples from the CMU PIE database before gray-scale value normalization are given in Fig. 3a.

4.1 Data Compression

First, we will demonstrate the convergence of 2DPCA-LN-ER, 2DPCA-NN-ER, GLRAM-LN-ER, and GLRAM-NN-ER, and present the images after element rearrangement. Then, we use two criteria, as suggested in [11], to examine data compression performance: 1) *Root Mean Squared Error* (RMSE) versus different values of d, the dimension of the lower dimensional space and 2) RMSE versus different values of Compression Ratios (CRs). For greater clarity, we set the reduced dimension m' to d for 2DPCA, 2DPCA-LN-ER, and 2DPCA-NN-ER. Also, for GLRAM, GLRAM-LN-ER, and GLRAM-NN-ER, we assume that the height and width after dimensionality reduction are the same, such that $m' = n' = d$.

4.1.1 Convergence Analysis and Element-Rearranged Images

We first take 2DPCA-LN-ER, 2DPCA-NN-ER, GLRAM-LN-ER, and GLRAM-NN-ER as examples to illustrate algorithmic convergence. In Fig. 2, we plot RMSEs, defined as

$$\sqrt{\sum_{i=1}^{N} \|X_i^t - X_i^{tREC}\|^2 / N},$$

for different numbers of iterations and $d = 5$. Convergence to a local minimum is evident for all of the algorithms.

In Fig. 3, we take GLRAM-LN-ER as an example to plot the element-rearranged image data computed on the CMU PIE database for different values of d. From the results, we can see that smaller values of d lead to greater correlation among the elements of each row/column vector. This is observed because most of the image information has been concentrated into the first d principal components along the row/column dimensions.

4.1.2 Energy Percentage of Eigenvalues

We also plot in Fig. 4 the Energy Percentage of Eigenvalues for different values of d on the CMU PIE and FERET databases. Energy Percentage of Eigenvalues is defined as $\sum_{i=1}^{d} \lambda^i / \sum_{i=1}^{n} \lambda^i$ for 2DPCA or 2DPCA-LN-ER, where λ^i is the ith eigenvalue when computing the projection matrix V. For GLRAM or GLRAM-LN-ER, Energy Percentage of Eigenvalues is defined as $\sum_{i=1}^{d} \lambda_1^i \sum_{i=1}^{d} \lambda_2^i / (\sum_{i=1}^{m} \lambda_1^i \sum_{i=1}^{n} \lambda_2^i)$, where λ_1^i and λ_2^i are the ith eigenvalue when computing the projection matrices U and V, respectively. When comparing GLRAM-LN-ER and 2DPCA-LN-ER with GLRAM and 2DPCA, respectively, we observe that eigenvalue's energy is concentrated more in the foremost few

Authorized licensed use limited to: Nanyang Technological University. Downloaded on August 24, 2009 at 08:33 from IEEE Xplore. Restrictions apply.

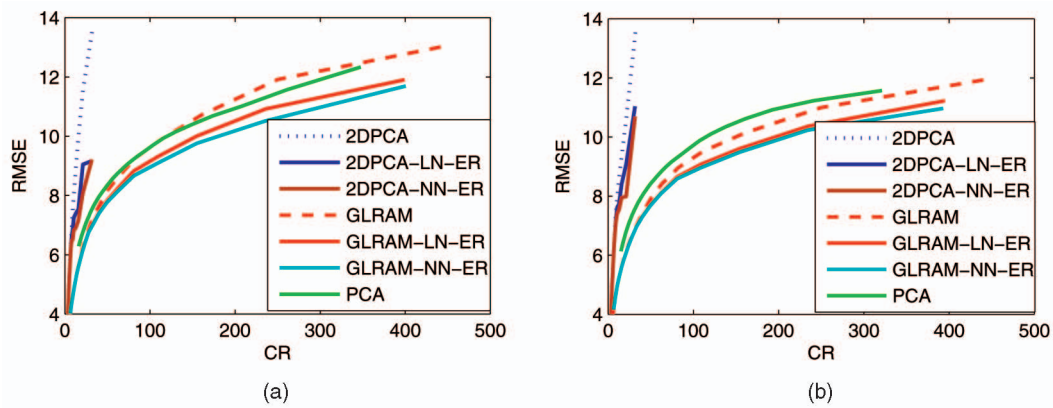

Fig. 6. The RMSEs of different algorithms for different compression ratios on (a) the FERET database and (b) the CMU PIE database.

eigenvalues, and the correlations among the features of rows and columns are enhanced.

4.1.3 RMSE versus Different Values of d

In the initial conference version of this work [24], we reported that reconstructed images from 2DPCA-LN-ER and GLRAM-LN-ER have higher visual quality and greater similarity to the original images than those from 2DPCA and GLRAM, respectively. Here, we support these qualitative judgments with a quantitative evaluation. In Fig. 5, we use the FERET and CMU PIE databases to compare the RMSEs of 2DPCA-LN-ER and 2DPCA-NN-ER to 2DPCA, as well as GLRAM-LN-ER and GLRAM-NN-ER to GLRAM, for different values of d, where d is the reduced dimension in the lower dimensional space. The proposed element rearrangement algorithm is shown to effectively remove more redundancy, especially when d is small. When d is sufficiently large, the RMSEs from GLRAM-LN-ER/GLRAM-NN-ER and 2DPCA-LN-ER/2DPCA-NN-ER are very close to those of GLRAM and 2DPCA since, in these cases, the RMSEs without element rearrangement are already very low. We also observe that, when d is small, GLRAM-NN-ER and 2DPCA-NN-ER outperform GLRAM-LN-ER and 2DPCA-LN-ER, respectively, which demonstrates that the nearest neighbor constraint in the feature domain can bring better compression performance.

4.1.4 RMSE versus Different Values of CR

We compare the RMSEs of different algorithms at various CRs. CR is defined as Nmn/s with s as the number of scales required to represent the data. According to the work on GLRAM [11], for a given lower dimension d, we have $s = d(N + m*n)$ for PCA, $s = d(N*m + n)$ for 2DPCA, and $s = (N*d + m + n)*d$ for GLRAM. For 2DPCA-LN-ER, 2DPCA-NN-ER, GLRAM-LN-ER, and GLRAM-NN-ER, the additional space s_{ad} for storing the element rearrangement index matrix must also be considered.

In our implementation, we employ a simple technique for compression of the index matrix, but a more sophisticated method could be used in its place. For a single $64 * 64$ image, index values range from 1 to 4,096, requiring 12 bits for encoding. We observe that for most matrix elements (more than 80 percent when d is small), the change in index from element rearrangement is relatively small, within 2^8 index values. Based on this observation, we use a 2-bit header to indicate four possible cases: *00* means that the element does not change in position, *01* and *10* mean that the global index changes within 2^8 index values and with positive or negative offset sign, and *11* means that the global index change is beyond 2^8. For cases 01 and 10, the global index differences are encoded in eight bits, and for case 11, the full 12 bits are used to directly encode the new index after element rearrangement. Considering that 32-bit values are needed for floating-point projection matrices and the lower dimensional representation, we set $s_{ad} = (f_1 * 8 + f_2 * 12 + 2) * 4{,}096/32$, where f_1 and f_2 are the percentages of pixels of case 01 and 11, respectively.

Fig. 6 plots the RMSE for different compression ratios. These results indicate better performance of GLRAM-LN-ER and GLRAM-NN-ER than GLRAM, and also of 2DPCA-LN-ER and 2DPCA-NN-ER in comparison to 2DPCA. Also, GLRAM-NN-ER outperforms GLRAM-LN-ER, and 2DPCA-NN-ER achieves better compression performance than 2DPCA-LN-ER. The performance of 2DPCA is the lowest, likely because 2DPCA only removes redundancies among different rows and does not remove redundancies among columns or along the person dimension [8], [11]. As observed in [11], GLRAM is not always better than the original PCA, possibly because GLRAM does not remove redundancies along the person dimension. As also noted in [10], [11], it is possible to apply PCA as a second-stage dimensionality reduction to remove more redundancy for 2DPCA, 2DPCA-LN-ER, 2DPCA-NN-ER, GLRAM, GLRAM-LN-ER, and GLRAM-NN-ER.

TABLE 1
The Top-One Recognition Rates (Percent) on the FERET and CMU PIE Database

	FERET	PIE(G4/P16)	PIE(G5/P15)	PIE(G6/P14)
PCA [1]	45.1 (440)	43.3 (191)	49.2 (264)	49.2 (341)
1DLDA [2]	91.8 (33)	76.2 (44)	83.2 (52)	88.7 (51)
2DLDA [12]	94.8 (14 × 8)	81.9 (30 × 8)	85.8 (16 × 6)	88.9 (22 × 6)
2DLDA-LN-ER	***95.6*** (22 × 4)	***83.7*** (12 × 8)	***86.1*** (26 × 10)	***91.2*** (18 × 10)
1DMFA [17]	92.1 (32)	77.3 (46)	84.3 (52)	89.5 (54)
2DMFA [17]	94.0 (14 × 6)	82.8 (20 × 4)	86.1 (18 × 4)	89.2 (20 × 16)
2DMFA-LN-ER	***96.5*** (18 × 4)	***86.4*** (26 × 4)	***88.6*** (20 × 8)	***91.2*** (14 × 8)

Note that the numbers in parentheses correspond to the feature dimensions with the best results after dimensionality reduction.

Authorized licensed use limited to: Nanyang Technological University. Downloaded on August 24, 2009 at 08:33 from IEEE Xplore. Restrictions apply.

4.2 Classification

To examine the effects of element rearrangement on classification performance, we compare 2DLDA-LN-ER and 2DMFA-LN-ER with 2DLDA and 2DMFA on the FERET and CMU PIE databases. We also report the results from PCA, LDA, and MFA, which are referred to as 1DLDA and 1DMFA here. After dimensionality reduction, we use the nearest neighbor classifier based on euclidean distance.

Several parameters need to be set beforehand for the different algorithms and should be chosen to give a fair comparison. For 1DLDA and 1DMFA, we choose the dimension in the PCA step according to a 95 percent energy criterion, similarly to [25], and report the best result over each possible dimension in the LDA and MFA step. For 1DMFA, 2DMFA, and 2DMFA-LN-ER, we fix k_1 as $n_c - 1$ because in our experiments, the total number of samples in each class is the same, and we empirically sample k_2 from 25 to 600 at an interval of 25 and report the best results. For 2DLDA, 2DLDA-LN-ER, 2DMFA, and 2DMFA-LN-ER, we fix the iteration number to 10 when iteratively computing the left and right projection matrices U and V [12], [17], and report the best result over values of m' and n' from 2 to 40 at an interval of 2. Note that the above parameters are taken from their original algorithms; we do not introduce any new parameters during the element rearrangement for 2DLDA-LN-ER. For 2DMFA-LN-ER, the only parameter k_2 in (18) is fixed to 25 in our experiments.

For the FERET database, three images ba, bc, and bh are used for training, and the other four images bd, be, bf, and bg are used for testing. For the CMU PIE database, the image set is partitioned into different gallery and probe sets, where the label Gm/Pn indicates that m images per person are randomly selected for training and the remaining n images are used for testing. The experimental results are reported in Table 1. Several observations can be made: 1) The bilinear subspace learning algorithms 2DLDA and 2DMFA are generally better than 1DLDA and 1DMFA for multiview face recognition, which is consistent with findings in [9], [12]; 2) 1DMFA and 2DMFA mostly outperform 1DLDA and 2DLDA with the exception of 2DMFA on the FERET database, possibly because of the inhomogeneous distributions of the training and testing sets in this case; and 3) 2DLDA-LN-ER and 2DMFA-LN-ER demonstrate higher accuracy than 2DLDA and 2DMFA, respectively, which supports the use of element rearrangement.

5 Conclusions

In this paper, we have studied the problem of how to rearrange elements of a data matrix for better unsupervised and supervised bilinear subspace learning. For unsupervised learning, this problem was formulated to find a matrix element rearrangement operator that maximizes intramatrix correlation. An approximate iterative solution based on a computationally feasible linear programming problem was proposed. In addition, the iterative algorithm was extended to supervised bilinear subspace learning problems for improvement of classification ability. The proposed algorithms have achieved encouraging results for both the unsupervised and supervised tasks.

Currently, the linear programming procedure is relatively slow for feature neighborhood constraints in (10) and the algorithm for compressing the index is basic. In future work, we plan to investigate efficient algorithms, such as Minimum-Weight Bipartite Matching [26], to replace the simplex method, and to investigate more effective solutions for index compression. In addition, we also plan to extend our work to enhance intramatrix correlations by aligning multiple images with individual permutation matrices to further improve the performance.

Acknowledgments

This material is based upon work funded by the Singapore National Research Foundation Interactive Digital Media R&D Program under research grant NRF2008IDM-IDM004-018 and NRF-2008IDM-IDM004-029 as well as the US Government. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the US Government.

References

[1] M. Turk and A. Pentland, "Face Recognition Using Eigenfaces," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 586-591, 1991.
[2] P. Belhumeur, J. Hespanha, and D. Kriegman, "Eigenfaces versus Fisherfaces: Recognition Using Class Specific Linear Projection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 711-720, July 1997.
[3] M. Vasilescu and D. Terzopoulos, "Multilinear Subspace Analysis for Image Ensembles," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 93-99, 2003.
[4] H. Chen, H. Chang, and T. Liu, "Local Discriminant Embedding and Its Variants," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 846-852, 2005.
[5] G. Dai and D. Yeung, "Tensor Embedding Methods," *Proc. Assoc. for the Advancement of Artificial Intelligence*, pp. 330-335, 2006.
[6] D. Tao, X. Li, W. Hu, S.J. Maybank, and X. Wu, "Supervised Tensor Learning," *Proc. IEEE Conf. Data Mining*, pp. 450-457, 2005.
[7] D. Tao, X. Li, X. Wu, and S.J. Maybank, "General Tensor Discriminant Analysis and Gabor Features for Gait Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 29, no. 10, pp. 1700-1715, Oct. 2007.
[8] D. Xu, S. Yan, L. Zhang, H. Zhang, Z. Liu, and H. Shum, "Concurrent Subspace Analysis," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 203-208, 2005.
[9] S. Yan, D. Xu, Q. Yang, L. Zhang, X. Tang, and H. Zhang, "Discriminant Analysis with Tensor Representation," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 526-532, 2005.
[10] J. Yang, D. Zhang, A. Frangi, and J. Yang, "Two-Dimensional pca: A New Approach to Appearance-Based Face Representation and Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 26, no. 1, pp. 131-137, Jan. 2004.
[11] J. Ye, "Generalized Low Rank Approximations of Matrices," *Machine Learning*, vol. 61, pp. 167-191, 2005.
[12] J. Ye, R. Janardan, and Q. Li, "Two-Dimensional Linear Discriminant Analysis," *Advances in Neural Information Processing Systems*, pp. 1569-1576, MIT Press, 2004.
[13] E. Simoncelli and B. Olshausen, "Natural Image Statistics and Neural Representation," *Ann. Rev. Neuroscience*, vol. 24, pp. 1193-1216, 2001.
[14] G. Nemhauser and L. Wolsey, *Integer and Combinatorial Optimization*. Wiley, 1988.
[15] Y. Rubner, C. Tomasi, and L. Guibas, "The Earth Mover's Distance as a Metric for Image Retrieval," *Int'l J. Computer Vision*, vol. 40, no. 2, pp. 99-121, Nov. 2000.
[16] P. Jensen and J. Bard, *Operations Research Models and Methods*. John Wiley and Sons, 2003.
[17] S. Yan, D. Xu, B. Zhang, H. Zhang, Q. Yang, and S. Lin, "Graph Embedding and Extensions: A General Framework for Dimensionality Reduction," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 29, no. 1, pp. 40-51, Jan. 2007.
[18] T. Jebara, "Images as Bags of Pixels," *Proc. IEEE Conf. Computer Vision*, pp. 265-272, 2003.
[19] T. Jebara, "Convex Invariance Learning," *Proc. Int'l Conf. Artificial Intelligence and Statistics*, 2003.
[20] T. Jebara, "Kernelizing Sorting, Permutation and Alignment for Minimum Volume PCA," *Proc. Conf. Learning Theory*, 2004.
[21] P. Shivaswamy and T. Jebara, "Permutation Invariant svms," *Proc. Int'l Conf. Machine Learning*, 2006.
[22] T. Sim, S. Baker, and M. Bsat, "The CMU Pose, Illumination, and Expression Database," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 25, no. 12, pp. 1615-1618, Dec. 2003.
[23] P. Phillips, H. Moon, S. Rizvi, and P. Rauss, "The Feret Evaluation Methodology for Face-Recognition Algorithms," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, pp. 1090-1104, Oct. 2000.
[24] S. Yan, D. Xu, S. Lin, T. Huang, and S.-F. Chang, "Element Rearrangement for Tensor-Based Subspace Learning," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2007.
[25] X. He, S. Yan, Y. Hu, P. Niyogi, and H. Zhang, "Face Recognition Using Laplacianfaces," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 27, no. 3, pp. 328-340, Mar. 2005.
[26] J. Munkres, "Algorithms for the Assignment and Transportation Problems," *J. SIAM*, vol. 5, no. 1, pp. 32-38, Mar. 1957.

Authorized licensed use limited to: Nanyang Technological University. Downloaded on August 24, 2009 at 08:33 from IEEE Xplore. Restrictions apply.