Learning Modewise Independent Components from Tensor Data Using Multilinear Mixing Model

Haiping Lu

Institute for Infocomm Research
Singapore
hplu@ieee.org

Abstract. Independent component analysis (ICA) is a popular unsupervised learning method. This paper extends it to multilinear modewise ICA (MMICA) for tensors and explores two architectures in learning and recognition. MMICA models tensor data as mixtures generated from modewise source matrices that encode statistically independent information. Its sources have more compact representations than the sources in ICA. We embed ICA into the multilinear principal component analysis framework to solve for each source matrix alternatively with a few iterations. Then we obtain mixing tensors through regularized inverses of the source matrices. Simulations on synthetic data show that MMICA can estimate hidden sources accurately from structured tensor data. Moreover, in face recognition experiments, it outperforms competing solutions with both architectures.

Keywords: independent component analysis, mixing model, tensor, multilinear subspace learning, unsupervised learning

1 Introduction

Independent component analysis (ICA) is an important *unsupervised learning* method for finding representational components of data with maximum statistical independence [1]. While principal component analysis (PCA) [2] gives *independent components* (ICs) only for *Gaussian* data, ICA finds ICs for the general case of *non-Gaussian* data [3]. ICA can be performed under two different architectures for image representation and recognition [4]. *Architecture I* treats images as random variables and pixels as random trials to find spatially local basis images that are statistically independent. *Architecture II* treats pixels as random variables and images as random trials to find factorial code that reflects global properties.

Real-world data are often specified in a high-dimensional space while they are highly constrained to a *subspace* [5]. Thus, *dimensionality reduction* is frequently employed to transform a high-dimensional data set into a low-dimensional subspace while retaining most of the underlying structure. As the number of ICs

found by ICA typically corresponds to the dimension of the input, dimensionality reduction in ICA is usually achieved through PCA [4].

Recently, there has been a surge of interest in learning subspace of multidimensional data, i.e., *tensor* data, from their natural multidimensional representations. Examples are 2-D/3-D images, videos, and multi-way social networks [6, 7]. *Multilinear subspace learning* of tensor data operates on natural tensor representations without reshaping into vectors. Thus, it can obtain simpler and more compact representations, and handle big data more efficiently [6]. There have been many multilinear extensions of PCA [8–12], linear discriminant analysis [13–17], and canonical correlation analysis [18–20]. In contrast, multilinear extensions of ICA have not been well addressed, though several works deal with the ICA problem using tensor-based approaches.

In [21], ICA mixing matrix is identified by decomposing the higher-order cumulant tensor for vector-valued observation data. In [22], a tensor probabilistic ICA algorithm was formulated for fMRI analysis, with selected voxels represented as very-high-dimensional vectors. The *multilinear ICA* (MICA) model in [23] analyzes multiple factors for image ensembles organized into a tensor according to different image formation factors such as people, views, and illuminations. It requires a large number of samples for training, e.g., 36 well-selected samples per class in [23]. Furthermore, MICA represents images as vectors and needs to know the forming factors. In this sense, MICA is a *supervised learning* method requiring data to be labeled with such information. In *unsupervised learning* without labels, it *degenerates* to classical ICA. Another work with the same name MICA in [24] uses a multilinear expansion of the probability density function of source statistics but represents data as vectors too. To the best of our knowledge, the only multilinear ICA formulation based on tensor input data is the *directional tensor ICA* (DTICA) in [25, 26], which estimates *two* mixing matrices for images. It forms row and column directional images by shifting the rows/columns and applies FastICA [27] to row/column vectors. As in [4], PCA is used for dimensionality reduction in DTICA. Neither MICA in [23] nor DTICA has demonstrated ***blind source separation*** capability, a classical application of ICA.

This paper aims to develop a multilinear extension of ICA that can do blind source separation for tensors with appropriate modeling. For example, Fig. 1(a) shows ten mixtures generated from two simple binary patterns in Fig. 1(b) with a *multilinear mixing model* similar to classical ICA. We propose a *multilinear modewise ICA* (MMICA) model for tensor data with *modewise ICs* that can model this generation process like ICA. We then develop an MMICA algorithm to estimate these modewise ICs, i.e., to estimate the sources in Fig. 1(b) from the observed mixtures in Fig. 1(a). As an ICA extension to tensors, MMICA can be applied to domains where ICA has been applied in the past.

This work is inspired by previous attempts in multilinear extensions of ICA and motivated by the compact representations of multilinear subspace learning methods. The MMICA model, to be presented in Section 2, assumes that tensor observation data have rich structures and they are mixtures generated from

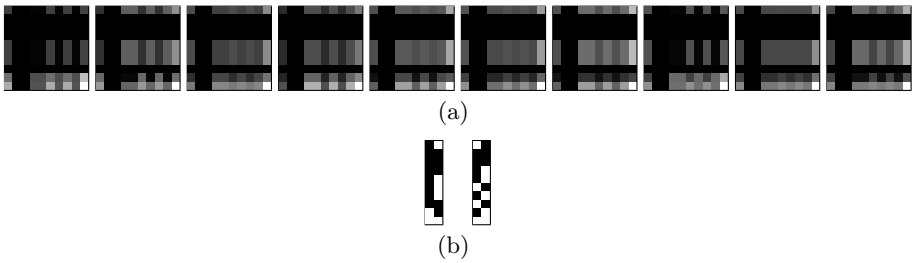

Fig. 1. The structured data in (a) are all mixtures generated from the source data in (b) with a *multilinear mixing model*. MMICA can recover the sources in (b) from observed mixtures in (a).

simple *modewise sources*, as illustrated in Fig. 1. We formulate the MMICA algorithm in Section 3 as an extension of *multilinear PCA* (MPCA) [10] to the non-Gaussian case by embedding ICA into MPCA to deal with non-Gaussianity of data in each mode. Also, we explore two architectures suggested in [4] for MMICA. Next, we discuss the differences with related works in Section 4. Finally, in Section 5, we show the *blind source separation* capability of MMICA through simulations and its recognition capability on real face data.

Note: for convenience of discussion, the acronym MICA below refers to the method in [23] rather than that in [24].

2 Multilinear Mixing Model for Tensors

2.1 Notations and fundamentals

We briefly introduce some notations and operations needed. For more details, please refer to [6, 28–30].

Vectors are denoted by lowercase boldface letters, e.g., $\mathbf{x}$; matrices by uppercase boldface, e.g., $\mathbf{U}$; and tensors by calligraphic letters, e.g., $\mathcal{A}$. Their elements are denoted with indices in parentheses. *Indices* are denoted by lowercase letters and span the range from 1 to the uppercase letter of the index whenever appropriate, e.g., $n = 1, 2, ..., N$.

Multidimensional arrays are referred to as *tensors* in mathematics. The number of dimensions N defines the *order* of a tensor. Tensor is a generalization of vector and matrix. Vectors are first-order tensors, and matrices are second-order tensors. An Nth-order tensor is denoted as $\mathcal{A} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$. It is addressed by N indices i_n, $n = 1, ..., N$, and each i_n addresses the n-mode of $\mathcal{A}$.

The *n-mode product* of a tensor $\mathcal{A}$ by a matrix $\mathbf{U} \in \mathbb{R}^{J_n \times I_n}$, denoted by $\mathcal{A} \times_n \mathbf{U}$, is a tensor with entries [29]:

$$(\mathcal{A} \times_n \mathbf{U})(i_1, ..., i_{n-1}, j_n, i_{n+1}, ..., i_N) = \sum_{i_n} \mathcal{A}(i_1, ..., i_N) \cdot \mathbf{U}(j_n, i_n). \quad (1)$$

The *n-mode vectors* of $\mathcal{A}$ are the I_n-dimensional vectors obtained by varying i_n while keeping all the other indices fixed. The *n-mode unfolded matrix* of $\mathcal{A}$, denoted as $\mathbf{A}_{(n)} \in \mathbb{R}^{I_n \times (I_1 \times ... \times I_{n-1} \times I_{n+1} \times ... \times I_N)}$, is formed with the n-mode vectors of $\mathcal{A}$ as its column vectors. An n-mode matrix or vector is denoted as $\mathbf{A}^{(n)}$ or $\mathbf{a}^{(n)}$, respectively. A *rank-one tensor* $\mathcal{A}$ equals to the outer product (denoted by '$\circ$') of N vectors [29]:

$$\mathcal{A} = \mathbf{u}^{(1)} \circ \mathbf{u}^{(2)} \circ ... \circ \mathbf{u}^{(N)}, \tag{2}$$

which means that

$$\mathcal{A}(i_1, i_2, ..., i_N) = \mathbf{u}^{(1)}(i_1) \cdot \mathbf{u}^{(2)}(i_2) \cdot ... \cdot \mathbf{u}^{(N)}(i_N) \tag{3}$$

for all values of indices.

2.2 MMICA model for tensor mixtures

The simplified noise-free ICA model [3] assumes that we observe M linear mixtures $\{x_m\}$ $(m = 1, ..., M)$ of P sources $\{s_p\}$ (the latent variables):

$$x_m = a_{m_1} s_1 + a_{m_2} s_2 + ... + a_{m_P} s_P, \tag{4}$$

where each mixture x_m and each IC (source) s_p are random scalar variables. The P sources $\{s_p\}$ are assumed to be independent. In ICA for random vector variables $\{\mathbf{x}_m\}$, each $\mathbf{x}_m$ is a mixture of P independent vector sources $\{\mathbf{s}_p\}$:

$$\mathbf{x}_m = a_{m_1} \mathbf{s}_1 + a_{m_2} \mathbf{s}_2 + ... + a_{m_P} \mathbf{s}_P. \tag{5}$$

For random Nth-order tensor variables $\{\mathcal{X}_m\}$ of dimension $I_1 \times ... \times I_N$, we propose a mixing model similar to (4) and (5) assuming P tensor variables $\{\mathcal{S}_p\}$ as the sources:

$$\mathcal{X}_m = a_{m_1} \mathcal{S}_1 + a_{m_2} \mathcal{S}_2 + ... + a_{m_P} \mathcal{S}_P. \tag{6}$$

Real-world tensor data often have rich structures. Therefore, we assume that the source tensors have compact representation as rank-one tensors (see equation (2)). For a simpler model, we further assume that these simple rank-one tensors are formed by $P_1 \times P_2 \times ... \times P_N = P$ vectors with one set in each mode, where the n-mode set has P_n *independent column vectors*: $\{\mathbf{s}_{p_n}^{(n)}, p_n = 1, ..., P_n\}$, and each source tensor is the outer product of N vectors, one from each mode, i.e.,

$$\mathcal{S}_p = \mathbf{s}_{p_1}^{(1)} \circ \mathbf{s}_{p_2}^{(2)} \circ ... \circ \mathbf{s}_{p_N}^{(N)}. \tag{7}$$

Next, we form an Nth-order *mixing tensor* $\mathcal{A}_m \in \mathbb{R}^{P_1 \times P_2 \times ... \times P_N}$ by stacking all the P mixing parameters $\{a_{m_1}, a_{m_2}, ..., a_{m_P}\}$ in (6) into an Nth-order tensor so its size $P_1 \times P_2 \times ... \times P_N = P$. Correspondingly, we form the *n-mode source matrix* $\mathbf{S}^{(n)} \in \mathbb{R}^{I_n \times P_n}$ with *independent columns* $\{\mathbf{s}_{p_n}^{(n)}, p_n = 1, ..., P_n\}$. We can then write the multilinear mixing model (6) in a form of the tensor-to-tensor-projection [6], an adaption of the Tucker decomposition model [31] to subspace learning, as

$$\mathcal{X}_m = \mathcal{A}_m \times_1 \mathbf{S}^{(1)} \times_2 \mathbf{S}^{(2)} \times ... \times_N \mathbf{S}^{(N)}, \tag{8}$$

We name this model as the **MMICA** model.

2.3 Regularized estimation of mixing tensor

When applying MMICA to learning and recognition, we estimate the source matrices $\{\mathbf{S}^{(n)}\}$ from M observed mixtures $\{\mathcal{X}_m\}$ (to be described in Sec. 3). To get the mixing tensor $\mathcal{A}_m$ from an observation tensor $\mathcal{X}_m$ based on $\{\mathbf{S}^{(n)}\}$, we use (8) to get

$$\mathcal{A}_m = \mathcal{X}_m \times_1 \mathbf{S}^{(1)^+} \times_2 \mathbf{S}^{(2)^+} \times ... \times_N \mathbf{S}^{(N)^+}, \tag{9}$$

where $\mathbf{S}^{(n)^+} = (\mathbf{S}^{(n)^T}\mathbf{S}^{(n)})^{-1}\mathbf{S}^{(n)^T}$ is the left inverse of $\mathbf{S}^{(n)}$. The superscript 'T' denotes the transpose of a matrix[1]. As $\mathbf{S}^{(n)^T}\mathbf{S}^{(n)}$ can be poorly conditioned in practice, we introduce a *regularized* left inverse of $\mathbf{S}^{(n)}$ to reduce the estimation variance by adding some small bias as [32, 33]

$$\mathbf{S}_r^{(n)^+} = (\mathbf{S}^{(n)^T}\mathbf{S}^{(n)} + \eta\mathbf{I}_{P_n})^{-1}\mathbf{S}^{(n)^T}, \tag{10}$$

where η is a small *regularization* parameter and $\mathbf{I}_{P_n}$ is an identity matrix of size $P_n \times P_n$. Thus, the mixing tensor is approximated as

$$\hat{\mathcal{A}}_m = \mathcal{X}_m \times_1 \mathbf{S}_r^{(1)^+} \times_2 \mathbf{S}_r^{(2)^+} \times ... \times_N \mathbf{S}_r^{(N)^+}. \tag{11}$$

3 MMICA Algorithm

3.1 MMICA by embedding ICA into MPCA

We solve the MMICA problem by embedding ICA into the MPCA framework [10], following the PCA+ICA in [4]. The procedures are *centering, initialization of source matrices, partial multilinear projection, modewise PCA, and modewise ICA*. Modewise ICA can be carried out in two architectures as in [4], where Architecture I is commonly used for traditional blind source separation task of ICA and Architecture II is for estimation of ICs for images[2]. The MMICA algorithm is summarized in Algorithm 1, with details described below.

The input to MMICA is a set of M tensor data samples $\{\mathcal{X}_m \in \mathbb{R}^{I_1 \times ... \times I_N}, m = 1, ..., M\}$. We need to specify two parameters: one is Q, the percentage of energy to be kept in PCA, and the other is K, the maximum number of iterations. Input data are centered first as in ICA or MPCA by subtracting the sample mean

$$\bar{\mathcal{X}} = \frac{1}{M}\sum_{m=1}^{M}\mathcal{X}_m. \tag{12}$$

There is no other data manipulation involved, such as data *re-sampling* and *re-arrangement* in DTICA [26].

[1] Only real-valued data are considered in this paper.
[2] We refer to the code at: http://mplab.ucsd.edu/∼marni/icaFacesCode.tar

Algorithm 1 Multilinear Modewise ICA (MMICA)

Input: M tensor samples $\{\mathcal{X}_m \in \mathbb{R}^{I_1 \times ... \times I_N}, m = 1, ..., M\}$, the percentage of energy to be kept in PCA Q, the maximum number of iterations K.
$\diamond$ Center the input samples by subtracting the mean $\bar{\mathcal{X}}$.
$\diamond$ Initialize source matrices $\mathbf{S}^{(n)} = \mathbf{I}_{I_n}$ and $P_n = I_n$ for $n = 1, ..., N$.
for $k = 1$ **to** K **do**
  for $n = 1$ **to** N **do**
    • Calculate partial multilinear projection $\tilde{\mathcal{A}}_m$ according to (13) for $m = 1, ..., M$.
    • Form $\tilde{\mathbf{A}}$ with columns consisting of n-mode vectors from $\{\tilde{\mathcal{A}}_m, m = 1, ..., M\}$.
    • Perform PCA on $\tilde{\mathbf{A}}$ and keep Q% of the total energy. Obtain $\mathbf{U}$ with the first R eigenvectors as its columns. Update $P_n = R$.
    • ***Architecture I*:** Perform FastICA on $\mathbf{U}^T$ to get $\mathbf{A}$ and $\mathbf{W}$. Set $\mathbf{S}^{(n)} = \mathbf{UA}$.
    • ***Architecture II*:** Get $\mathbf{V} = \mathbf{U}^T\tilde{\mathbf{A}}$ and perform FastICA on $\mathbf{V}^T$ to get $\mathbf{A}$ and $\mathbf{W}$. Set $\mathbf{S}^{(n)} = \mathbf{UW}^T$.
  end for
end for
Output: $\{\mathbf{S}^{(n)}, n = 1, ..., N\}$

3.2 Iterative alternating estimation

In the MMICA model (8), data are generated from all N source matrices $\{\mathbf{S}^{(n)}, n = 1, ..., N\}$ rather than any one of them individually. Unfortunately, we can not determine these N matrices simultaneously, except when $N = 1$ where it is degenerated to the classical ICA. Estimating $\mathbf{S}^{(n)}$ in a particular mode n needs the knowledge of other source matrices $\{\mathbf{S}^{(j)}, j \neq n\}$. Therefore, to solve MMICA, we follow the *iterative alternating projection* method [6]. We estimate $\mathbf{S}^{(n)}$ conditioned on all the other source matrices $\{\mathbf{S}^{(j)}, j \neq n\}$, alternating between modes. This is *significantly different* from DTICA [26], which is non-iterative.

Initialization: Since all source matrices depend on each other in estimation, we need to initialize them before proceeding. We adopt a simple strategy to initialize the n-mode source matrix $\mathbf{S}^{(n)}$ to an identity matrix $\mathbf{I}_{I_n}$ of size $I_n \times I_n$. Thus, the n-mode source dimension P_n is initialized to I_n.

Modewise processing: In each iteration, we process modewise from 1-mode to N-mode, a simple mode ordering used by many other algorithms [6]. For a particular mode n, we have all other source matrices $\{\mathbf{S}^{(j)}, j \neq n\}$ fixed and estimate $\mathbf{S}^{(n)}$ by first calculating the *partial multilinear projection* based on (11) as

$$\tilde{\mathcal{A}}_m = \mathcal{X}_m \times_1 \mathbf{S}_r^{(1)^+} ... \times_{n-1} \mathbf{S}_r^{(n-1)^+} \times_{n+1} \mathbf{S}_r^{(n+1)^+} ... \times_N \mathbf{S}_r^{(N)^+}. \tag{13}$$

Next, we form a matrix $\tilde{\mathbf{A}} \in \mathbb{R}^{I_n \times (M \times \prod_{j=1, j\neq n}^{N} P_j)}$ by concatenating $\{\tilde{\mathbf{A}}_{m(n)}, m = 1, ..., M\}$, the n-mode unfolded matrix of $\{\tilde{\mathcal{A}}_m\}$, so that the columns of $\tilde{\mathbf{A}}$ consist of n-mode vectors from $\{\tilde{\mathcal{A}}_m\}$. We then perform PCA on $\tilde{\mathbf{A}}$ and keep Q percent of the total energy/variations, resulting a PCA basis matrix $\mathbf{U} \in \mathbb{R}^{I_n \times R}$ with R leading eigenvectors. Subsequently, we update the n-mode source dimension as $P_n = R$.

3.3 Two architectures

ICA can be performed under two architectures [4] and so can MMICA. We use the popular FastICA [27] to maximize the modewise non-Gaussianity for modewise IC estimation. FastICA takes a data matrix in and returns a mixing matrix $\mathbf{A}$ and a separating matrix $\mathbf{W}$. They can be used under two architectures in MMICA in the following ways:

- **Architecture I:** FastICA on $\mathbf{U}^T$ gives mixing matrix $\mathbf{A}$ and separating matrix $\mathbf{W}$. Thus, we set the n-mode source matrix as

$$\mathbf{S}^{(n)} = \mathbf{U}\mathbf{A}. \tag{14}$$

- **Architecture II:** We first obtain the PCA projection as $\mathbf{V} = \mathbf{U}^T\tilde{\mathbf{A}}$, and FastICA on $\mathbf{V}^T$ gives $\mathbf{A}$ and $\mathbf{W}$. Hence, we set the n-mode source matrix as

$$\mathbf{S}^{(n)} = \mathbf{U}\mathbf{W}^T. \tag{15}$$

3.4 Discussion on MMICA

Identifiability and number of ICs: Following ICA [1], the independent column vectors of modewise source matrices in MMICA are identifiable up to permutation and scaling if they (except one at most) have non-Gaussian distributions and the number of mixtures is no smaller than the number of ICs to be estimated. However, MMICA can not estimate the number of modewise ICs, as in the general case of ICA. When this number is unknown, we determine it by specifying Q in PCA, as described above.

Convergence and termination: The convergence problem is difficult in ICA. To the best of our knowledge, for FastICA, local convergence analysis is only available for the so-called one-unit case, which considers only one row of the separating matrix [34]. Here, we provide empirical results on the convergence properties of MMICA in Sec. 5, where it converges in one iteration in studies on synthetic data while its classification accuracy stabilizes in just a few iterations in face recognition experiments. Thus, we terminate the iteration by setting K, the maximum number of iterations, to a small number for efficiency.

3.5 Feature selection for classification

After obtaining the separated source matrices $\{\mathbf{S}^{(n)}\}$, we have the MMICA representation (coordinates in the mixing tensor) $\hat{\mathcal{A}}$ of a sample $\mathcal{X}$ from (11). Though we can use $\hat{\mathcal{A}}$ directly for classification tasks, we can select a subspace for the convenience of comparison with linear learning algorithms and also for better classification accuracy, as pointed out in [4]. Thus, we further select and sort MMICA features through the same class discriminability as in [4, 25, 26] to study its classification performance.

We can view the MMICA representation $\hat{\mathcal{A}} \in \mathbb{R}^{P_1 \times \ldots \times P_N}$ as being projected through $\prod_{n=1}^{N} P_n$ *elementary multilinear projections* (EMPs) $\{\mathbf{s}_{p_n}^{(n)}, n = 1, ..., N\}$

[12], where $\mathbf{s}_{p_n}^{(n)}$ is the p_nth column of $\mathbf{S}^{(n)}$. For each component $\hat{\mathcal{A}}(p_1, ..., p_N)$, extracted by EMP $\{\mathbf{s}_{p_n}^{(n)}, n = 1, ..., N\}$, we define a class discriminability $\gamma_{p_1 p_2 ... p_N}$ as the ratio of between-class variability to within-class variability, measured by scatters calculated from the training samples:

$$\gamma_{p_1 ... p_N} = \frac{\sum_{c=1}^{C} N_c \cdot \left[\bar{\hat{\mathcal{A}}}_c(p_1, ..., p_N) - \bar{\hat{\mathcal{A}}}(p_1, ..., p_N)\right]^2}{\sum_{m=1}^{M} \left[\hat{\mathcal{A}}_m(p_1, ..., p_N) - \bar{\hat{\mathcal{A}}}_{c_m}(p_1, ..., p_N)\right]^2}, \tag{16}$$

where C is the number of classes, M is the number of training samples, N_c is the number of samples for class c and c_m is the class label for the mth training sample $\mathcal{X}_m$. $\hat{\mathcal{A}}_m$ is the mixing tensor for $\mathcal{X}_m$. The mean feature tensor

$$\bar{\hat{\mathcal{A}}} = \frac{1}{M} \sum_{m} \hat{\mathcal{A}}_m \tag{17}$$

and the class mean feature tensor

$$\bar{\hat{\mathcal{A}}}_c = \frac{1}{N_c} \sum_{m, c_m = c} \hat{\mathcal{A}}_m. \tag{18}$$

We arrange the entries in $\hat{\mathcal{A}}$ into a feature vector $\hat{\mathbf{a}}$ according to the magnitude of $\gamma_{p_1 ... p_N}$ in descending order. The first P entries of $\hat{\mathbf{a}}$, i.e., the P most discriminable components, are selected for classification tasks.

4 Differences with Related Works

MMICA is different from MICA, DTICA and ICA in several aspects, as discussed in the following.

4.1 MMICA vs. MICA

The origin of MMICA in (8) traces back to the higher-order singular value decomposition (HOSVD) [29] and Tucker decomposition [31]. Therefore, it shares mathematical similarity with MICA [23] and DTICA [26, 25], which are both based on HOSVD. However, the MMICA model represents multidimensional data as *tensors* while the MICA model represents them as *vectors* (e.g., 2D faces are represented as 8560×1 vectors in [23]). Thus, 'N' in MMICA represents the order (*number of dimensions*) of a single tensor sample, while 'N' in MICA represents the *number of forming factors* for an ensemble of many samples, with each sample represented as a vector. As mentioned in Sec. 1, MICA is designed as a *supervised learning* method for data labeled with forming factors such as *people* (subject ID), *views* and *illuminations* so the tensor in [23] is formed with four modes as *pixels* $\times$ *people* $\times$ *views* $\times$ *illuminations*. Thus, MICA degenerates to ICA when these factors are unknown, i.e., in unsupervised

learning. In contrast, MMICA is an *unsupervised learning* method that does not require such labels. Furthermore, *hidden sources* are not defined in [23]. Hence, the MICA model can not interpret tensor data as in (6) and it cannot perform blind source separation for tensors while MMICA can do so.

4.2 MMICA vs. DTICA:

MMICA and DTICA both model a number of tensors with a generative model. DTICA models tensor mixtures with N *mixing matrices and one single source tensor*, built from a *factor-analysis* point of view. In contrast, MMICA models tensor mixtures with *one single mixing tensor and* N *source matrices*, built from an *independent-component-analysis* point of view. Thus, MMICA can be interpreted in a similar manner as the classical ICA model [3] and perform blind source separation while DTICA cannot be similarly interpreted as mixing several sources and separate sources since its model only involves one *single source tensor*. Furthermore, the MMICA algorithm is iterative while the DTICA algorithm is noniterative, and DTICA requires resampling while MMICA does not require. In addition, DTICA is only formulated for one architecture while we have formulated both architectures for MMICA. Lastly, MMICA makes use of *regularization* to get better results than DTICA.

4.3 MMICA vs. ICA

From (4) to (8), while the classical ICA model assumes that the sources are *mutually independent*, the MMICA model assumes that the sources are *structured tensors formed from modewise matrices with independent columns* instead, which has a simpler and more compact representation when $N > 1$. For the same number of mixing parameters $P = \prod_{n=1}^{N} P_n$, the sources $\{\mathbf{S}^{(n)}\}$ to estimate in MMICA have a size of $\sum_{n=1}^{N} (I_n \times P_n)$ while those in ICA have a size of $\left(\prod_{n=1}^{N} I_n\right) \times \left(\prod_{n=1}^{N} P_n\right)$. E.g., for $N = 3$, $P = 8$, $\{P_n = 2\}$ and $\{I_n = 10\}$, MMICA sources have a size of 60, while ICA sources have a size of 8000, which is about 132 times larger. For $N = 3$, $P = 125$, $\{P_n = 5\}$ and $\{I_n = 100\}$, MMICA sources have a size of 1.5×10^3, while ICA sources have a size of 1.25×10^8, which is about 8.3×10^4 times larger.

5 Experiments

MMICA is applicable to tensors of any order, such as videos, 3-D images, and multi-way social networks [7, 6]. In particular, MMICA can be applied to domains where ICA has been applied in the past, such as biometrics [4], bioinformatics [35], and neuroimaging [36]. For easy visual illustration, this paper studies 2-D images only, which are *matrix data*, i.e., *second-order tensor data* ($N = 2$). We evaluate MMICA on both synthetic and real data. For synthetic data, we study its capability in estimating hidden sources given their mixtures. For real data, we

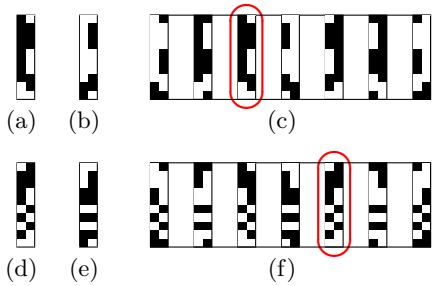

Fig. 2. Blind source separation on synthetic data: (a) true 1-mode source, (b) MMICA estimate of 1-mode source, (c) equivalent patterns of 1-mode MMICA estimate, (d) true 2-mode source, (e) MMICA estimate of 2-mode source, (f) equivalent patterns of 2-mode MMICA estimate. (The pattern matched with the true source is enclosed with an oval.)

test it on face recognition, which is widely-used for learning algorithm evaluation [26, 8, 14, 25] with practical importance in security-related applications such as biometric authentication and surveillance.

5.1 Blind source separation on synthetic data

Data generation: This experiment studies whether MMICA can estimate source matrices from synthetic mixture data generated according to (8). The source matrices used are as shown in Fig. 1(b), which are reproduced in Figs. 2(a) and 2(d). Each source matrix is a randomly generated simple *binary pattern* of size 10×2 ($I_n = 10, P_n = 2$). We generated 100 mixtures ($M = 100$) according to (8) by drawing the elements of mixing tensors randomly from a uniform distribution on the unit interval. Figure 1 (a) shows ten such mixtures as 8-bit gray images.

Hidden source recovery: We applied MMICA with $Q = 100$ using Architecture I for this blind source separation task, followed by binarization to obtain binary source patterns in Figs. 2(b) and 2(e). Since ICA estimation is only unique up to sign and permutation [3], the estimated MMICA sources in Figs. 2(b) and 2(e) are equivalent to the patterns in Figs. 2(c) and 2(f), respectively. One pattern in Figs. 2(c) and 2(f) matches Figs. 2(a) and 2(d) exactly, respectively. Thus, independent modewise source patterns are estimated correctly. To the best of our knowledge, this is the first multilinear extension of ICA for tensor data with demonstrated capability of blind source separation.

Effects of iteration and regularization: For this binary source estimation problem, MMICA has recovered the true hidden patterns with only one iteration, showing good convergence. If there is no regularization (using (9)), the mixing tensors can be recovered exactly. Using (11) with $\eta = 10^{-3}$, the estimation has a small average error of $0.005(\pm 0.001)$.

5.2 Face recognition studies

Data: The Pose, Illumination, and Expression (PIE) database [37] is widely used for testing face recognition performance. It contains 68 individuals with face images captured under varying pose, illumination and expression. As using the full set leads to low recognition rates for all compared ICA algorithms, here we report the results from a subset of medium difficulty, with five frontal or near frontal poses (C05, C07, C09, C27, C29) under 14 illumination conditions (05 to 14 and 18 to 21, excluding the poorest 7 illumination conditions). Thus, there are about 70 (5×14) samples per subject and a total number of 4,754 face images (with six faces missing). All face images were manually cropped, aligned (with manually annotated eye coordinate) and normalized to 32×32 pixels, with 256 gray levels per pixel. We test face recognition performance under varying number of training samples per subject, denoted by L.

Algorithms and settings: Since this paper focuses on examining ICA and its extensions under both Architectures I and II, we evaluate MMICA against the classical ICA, MICA in [23] and DTICA in [26]. For fair comparison, we consider *unsupervised learning* only. Hence, training data are not labeled (with image forming factors: poses, illuminations and expressions) and MICA *degenerates* to the classical ICA in this case. Effectively, we have six algorithms to compare: ICA1/MICA1, DTICA1 and MMICA1 for Architecture I, and ICA2/MICA2, DTICA2 and MMICA2 for Architecture II. For DTICA, we form the directional images with the amount of shift $l = 2$, as suggested in [26]. We fix the regularization parameter in (10) as $\eta = 10^{-3}$ for MMICA. All algorithms tested employ FastICA version 2.5[3] [27] with identical (default) settings for fair comparison. We test four values of Q (85, 90, 95, 98), the energy kept in PCA. For all six algorithms, we sort extracted features in descending class discriminability γ in (16) and take the first P features for recognition. To classify extracted features, we use the nearest neighbor classifier with Euclidean distance measure.

Gray-level face images are naturally second-order tensors (matrices), i.e., $N = 2$. Therefore, they are input directly as 32×32 tensors to DTICA and MMICA. For ICA/MICA, they are vectorized to 1024×1 vectors as input. For each subject in a face recognition experiment, $L(= 4, 6, 8, 10)$ samples were randomly selected for training and the rest were used for testing. We report the best results over Q and P, averaged from ten such random splits (repetitions).

Impact of iterations: We first study the effect of the number of iterations K on the recognition performance of MMICA. Typical results are shown in Fig. 3 for up to 20 iterations with $P = 60$ and $Q = 98$. The figure shows that all accuracy curves are stable with respect to K, while the first a few iterations are more effective for MMICA2 than for MMICA1 in general. Based on this study, we set $K = 3$ in MMICA to reduce the computational cost.

Recognition results: Figures 4(a) and 4(b) show the best average recognition rates for each algorithm with up to 300 features tested ($P = 1, ..., 300$) for Architecture I and II, respectively. The error bars indicate the standard de-

[3] Code at http://www.cis.hut.fi/projects/ica/fastica/code/FastICA_2.5.zip

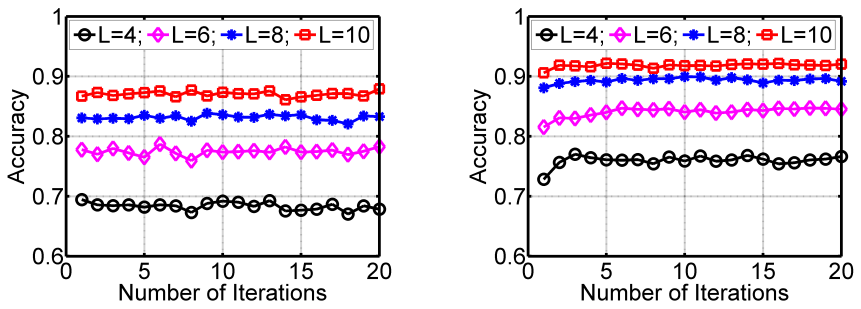

Fig. 3. The impact of iteration numbers on the face recognition accuracy of MMICA with (a) Architecture I (MMICA1) and (b) Architecture II (MMICA2), for $L = 4, 6, 8, 10$.

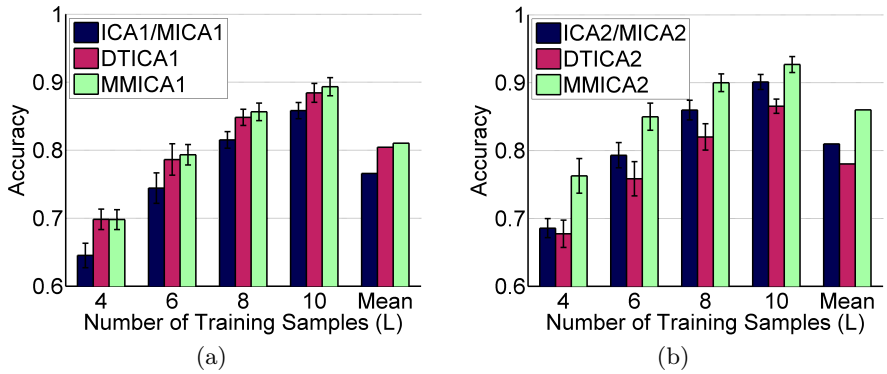

Fig. 4. The average face recognition accuracy comparison of ICA/MICA, DTICA and MMICA for $L = 4, 6, 8, 10$ from ten repetitions with (a) Architecture I and (b) Architecture II. The error bars in (a) and (b) indicate the standard deviations. The results are the best ones for each method from testing four values of Q (85, 90, 95, 98) and 300 values of P (1,...,300).

viations. Different performance variation is observed for two architectures. Using Architecture I, both DTICA1 and MMICA1 outperform ICA1/MICA1 by around 4% on average. MMICA1 outperforms ICA1/MICA1 by 5.3%, 4.9%, 4.1% and 3.5% for $L = 4$, 6, 8 and 10, respectively, with more advantage for a smaller L. However, the performance difference between DTICA1 and MMICA1 is small ($< 1\%$ on average). Using Architecture II, DTICA2 is inferior to ICA2/MICA2 in all cases. MMICA2 outperforms ICA2/MICA2 by 7.7%, 5.7%, 4.0% and 2.6% (mean=5%) for $L = 4$, 6, 8 and 10, respectively, again showing superior performance for a smaller L.

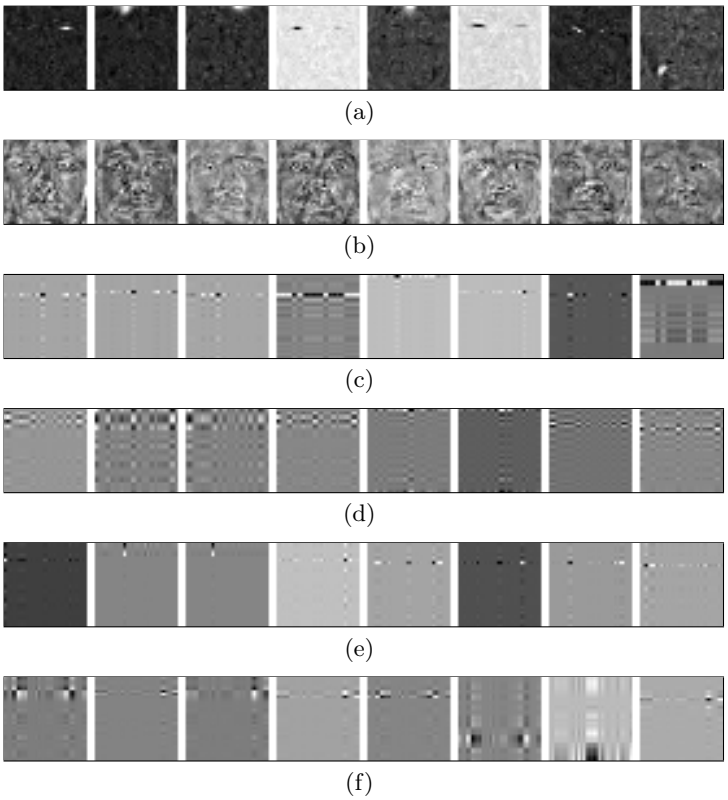

Fig. 5. Eight most discriminable bases obtained from the PIE database with $L = 10$ for (a) ICA1/MICA1, (b) ICA2/MICA2, (c) DTICA1, (d) DTICA2, (e) MMICA1, and (f) MMICA2.

5.3 Feature characteristics

Next, we examine the characteristics of learned features to gain some insight. Figure 5 depicts the eight most discriminable projection bases as 8-bit gray-level images for ICA/MICA, DTICA and MMICA obtained from the PIE database with $L = 10$ using Architectures I and II.

In Figs. 5(a), 5(c) and 5(e), similar to the observations in [4], each projection basis reflects the closeness of each pixel to a cluster of pixels having similar behavior across images. Therefore, these bases are sparse for all three algorithms. In particular, several DTICA1 bases share similar characteristics with MMICA1 bases, showing more structured information than ICA1/MICA1 bases. This may partly be the reason for their closer recognition performance.

With Architecture II, more global properties are encoded. Each ICA-based projection basis attempts to capture a cluster of similar images or image patches, as in Figs. 5(b), 5(d) and 5(f). While this architecture generates more face-like bases for ICA2/MICA2, we found that DTICA2 and MMICA2 bases are

quite different, where each basis captures a particular local pattern of the face image instead. DTICA2 and MMICA2 bases have strong structures due to their multilinear nature, with MMICA2 sparser than DTICA2 on the whole. Although each MMICA basis has a size of $32 + 32 = 64$ while each ICA/MICA basis has a size of $32 \times 32 = 1024$, which means 15 times larger, the simpler MMICA bases have achieved much better recognition performance than the more complex ICA/MICA bases. A possible explanation is that the recognition task here has the *small sample size* problem, where the number of samples is small relative to the size of variables to be estimated. Sparser bases have more compact size, leading to *less overfitting* and *better generalization*.

6 Conclusions

We have introduced the multilinear modewise ICA for tensor data using a multilinear mixing model. MMICA extracts modewise independent sources directly from tensor representations through an iterative alternating projection method. We solved this problem by embedding ICA into the MPCA framework and examined two ICA architectures. Studies on synthetic data indicate that MMICA can recover hidden sources from their mixtures accurately. Moreover, experiments on face recognition show different behaviors under different architectures. Using Architecture I, MMICA has similar performance as DTICA while they both outperform ICA/MICA. For Architecture II, MMICA gives the best performance and it is particularly effective when there are only a small number of samples for training. We further examined the extracted features in order to understand the implications and found that MMICA features are sparser and more structured even with Architecture II.

Acknowledgments

The author would like to thank the anonymous reviewers of this work for their insightful and constructive comments.

References

1. Hyvärinen, A., Karhunen, J., Oja, E.: Independent Component Analysis. John Wiley & Sons (2001)
2. Jolliffe, I.T.: Principal Component Analysis. Second edn. Springer Series in Statistics (2002)
3. Hyvärinen, A., Oja, E.: Independent component analysis: Algorithms and applications. Neural Networks **13**(4-5), 411–430 (2000)
4. Bartlett, M.S., Movellan, J.R., Sejnowski, T.J.: Face recognition by independent component analysis. IEEE Transactions on Neural Networks **13**(6), 1450–1464 (2002)
5. Shakhnarovich, G., Moghaddam, B.: Face recognition in subspaces. In Li, S.Z., Jain, A.K., eds.: Handbook of Face Recognition, pp. 141–168. (2004)

6. Lu, H., Plataniotis, K.N., Venetsanopoulos, A.N.: A survey of multilinear subspace learning for tensor data. Pattern Recognition **44**(7), 1540–1551 (2011)
7. Faloutsos, C., Kolda, T.G., Sun, J.: Mining large time-evolving data using matrix and tensor tools. International Conference on Machine Learning 2007 Tutorial (2007) http://www.cs.cmu.edu/~christos/TALKS/ICML-07-tutorial/ICMLtutorial.pdf
8. Ye, J.: Generalized low rank approximations of matrices. Machine Learning **61**(1-3), 167–191 (2005)
9. Ye, J., Janardan, R., Li, Q.: GPCA: An efficient dimension reduction scheme for image compression and retrieval. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 354–363 (2004)
10. Lu, H., Plataniotis, K.N., Venetsanopoulos, A.N.: MPCA: Multilinear principal component analysis of tensor objects. IEEE Transactions on Neural Networks **19**(1), 18–39 (2008)
11. Lu, H., Plataniotis, K.N., Venetsanopoulos, A.N.: Uncorrelated multilinear principal component analysis through successive variance maximization. In: International Conference on Machine Learning, pp. 616–623 (2008)
12. Lu, H., Plataniotis, K.N., Venetsanopoulos, A.N.: Uncorrelated multilinear principal component analysis for unsupervised multilinear subspace learning. IEEE Transactions on Neural Networks **20**(11), 1820–1836 (2009)
13. Ye, J., Janardan, R., Li, Q.: Two-dimensional linear discriminant analysis. In: Advances in Neural Information Processing Systems (NIPS), pp. 1569–1576 (2004)
14. Yan, S., Xu, D., Yang, Q., Zhang, L., Tang, X., Zhang, H.J.: Discriminant analysis with tensor representation. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 526–532 (2005)
15. Tao, D., Li, X., Wu, X., Maybank, S.J.: General tensor discriminant analysis and gabor features for gait recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **29**(10), 1700–1715 (2007)
16. Tao, D., Li, X., Wu, X., Maybank, S.J.: Tensor rank one discriminant analysis-a convergent method for discriminative multilinear subspace selection. Neurocomputing **71**(10-12), 1866–1882 (2008)
17. Lu, H., Plataniotis, K.N., Venetsanopoulos, A.N.: Uncorrelated multilinear discriminant analysis with regularization and aggregation for tensor object recognition. IEEE Transactions on Neural Networks **20**(1), 103–123 (2009)
18. Lee, S.H., Choi, S.: Two-dimensional canonical correlation analysis. IEEE Signal Processing Letters **14**(10), 735–738 (2007)
19. Kim, T.K., Cipolla, R.: Canonical correlation analysis of video volume tensors for action categorization and detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **31**(8), 1415–1428 (2009)
20. Lu, H.: Learning canonical correlations of paired tensor sets via tensor-to-vector projection. In: the 23rd International Joint Conference on Artificial Intelligence (2013)
21. Lathauwer, L.D., Vandewalle, J.: Dimensionality reduction in higher-order signal processing and rank-$(R_1, R_2, ..., R_N)$ reduction in multilinear algebra. Linear Algebra and its Applications **391**, 31–55 (2004)
22. Beckmann, C.F., Smith, S.M.: Tensorial extensions of independent component analysis for multisubject fMRI analysis. NeuroImage **25**(1), 294–311 (2005)
23. Vasilescu, M.A.O., Terzopoulos, D.: Multilinear independent components analysis. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 547–553 (2005)

24. Raj, R.G., Bovik, A.C.: MICA: A multilinear ICA decomposition for natural scene modeling. IEEE Transactions on Image Processing **17**(3), 259–271 (2009)
25. Zhang, L., Gao, Q., Zhang, D.: Directional independent component analysis with tensor representation. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–7 (2008)
26. Gao, Q., Zhang, L., Zhang, D., Xu, H.: Independent components extraction from image matrix. Pattern Recognition Letters **31**(3), 171–178 (2010)
27. Hyvärinen, A.: Fast and robust fixed-point algorithms for independent component analysis. IEEE Transactions on Neural Networks **10**(3), 626–634 (1999)
28. Lathauwer, L.D., Moor, B.D., Vandewalle, J.: On the best rank-1 and rank-$(R_1, R_2, ..., R_N)$ approximation of higher-order tensors. SIAM Journal of Matrix Analysis and Applications **21**(4), 1324–1342 (2000)
29. Lathauwer, L.D., Moor, B.D., Vandewalle, J.: A multilinear singualr value decomposition. SIAM Journal of Matrix Analysis and Applications **21**(4), 1253–1278 (2000)
30. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Review **51**(3), 455–500 (2009)
31. Tucker, L.R.: Some mathematical notes on three-mode factor analysis. Psychometrika **31**, 279–311 (1966)
32. Friedman, J.H.: Regularized discriminant analysis. Journal of the American Statistical Association **84**(405), 165–175 (1989)
33. Lu, H., Plataniotis, K.N., Venetsanopoulos, A.N.: Boosting discriminant learners for gait recognition using MPCA features. EURASIP Journal on Image and Video Processing **2009**, Article ID 713183, 11 pages (2009) doi:10.1155/2009/713183.
34. Oja, E., Yuan, Z.: The FastICA algorithm revisited: convergence analysis. IEEE Transactions on Neural Networks **17**(6), 1370–1381 (2006)
35. Scholz, M., Gatzek, S., Sterling, A., Fiehn, O., Selbig, J.: Metabolite fingerprinting: detecting biological features by independent component analysis. Bioinformatics **20**(15), 2447–2454 (2004)
36. Zuo, X.N., Kelly, C., Adelstein, J.S., Klein, D.F., Castellanos, F.X., Milham, M.P.: Reliable intrinsic connectivity networks: test-retest evaluation using ICA and dual regression approach. Neuroimage **49**(3), 2163–2177 (2010)
37. Sim, T., Baker, S., Bsat, M.: The CMU pose, illumination, and expression database. IEEE Transactions on Pattern Analysis and Machine Intelligence **25**(12), 1615–1618 (2003)