Uni-orthogonal Nonnegative Tucker Decomposition for Supervised Image Classification

Rafal Zdunek

Institute of Telecommunications, Teleinformatics and Acoustics,
Wroclaw University of Technology, Wybrzeze Wyspianskiego 27,
50-370 Wroclaw, Poland
rafal.zdunek@pwr.wroc.pl

Abstract. The Tucker model with orthogonality constraints (often referred to as the HOSVD) assumes decomposition of a multi-way array into a core tensor and orthogonal factor matrices corresponding to each mode. Nonnegative Tucker Decomposition (NTD) model imposes nonnegativity constraints onto both core tensor and factor matrices. In this paper, we discuss a mixed version of the models, i.e. where one factor matrix is orthogonal and the remaining factor matrices are nonnegative. Moreover, the nonnegative factor matrices are updated with the modified Barzilai-Borwein gradient projection method that belongs to a class of quasi-Newton methods. The discussed model is efficiently applied to supervised classification of facial images, hand-written digits, and spectrograms of musical instrument sounds.

1 Introduction

The Tucker model [1] decomposes a multi-way array into a core tensor multiplied by a factor matrix along each mode. When orthogonality constraints are imposed onto all the factor matrices, the model is referred to as Higher-Order Singular Value Decomposition (HOSVD), and can be regarded as a multi-linear extension to SVD [2]. When factor matrices and a core tensor are nonnegatively constrained, the model is referred to as Nonnegative Tucker Decomposition (NTD), and it can be considered as a generalization to Nonnegative Tensor Factorization (NTF) or nonnegativity constrained PARAFAC model [3, 4]. In Semi-NTD (SNTD) models, nonnegativity constraints are relaxed for a core tensor or selected factor matrices [5].

In this paper, we assume a special case of SNTD, where one factor matrix is orthogonal, the others are nonnegative, and a core tensor is unsigned. This approach combines NTD with HOSVD, which is particularly useful when the model is applied to image classification. Assuming the images to be classified are arranged to form a three-way array, the orthogonality constraint should be imposed onto the factor matrix that corresponds to the mode along which the images are stacked. The orthogonal column vectors in that factor can be regarded as discriminant vectors, especially as there are as many vectors as classes. The

G. Maino and G.L. Foresti (Eds.): ICIAP 2011, Part I, LNCS 6978, pp. 88–97, 2011.

core tensor multiplied along all but that mode contains lateral slices that can be considered as feature images.

There are many applications of the Tucker-based models. A survey of the applications can be found, e.g. in [5, 6, 7]. Vasilescu and Terzopoulos [8] applied the Tucker model to extract TensorFaces in computer vision. Feature representations in TensorFaces are considerably more accurate than in EigenFaces that can be obtained from the standard PCA technique. The Tucker model has been also used for analyzing facial images by Wang and Ahuja [9], and Vlasic *et al* [10]. Savas and Elden [11] applied the HOSVD for identifying handwritten digits. NTD has been applied to image feature extraction [12], image clustering [5], and supervised image segmentation [13, 14].

The Tucker decomposition can be obtained with many numerical algorithms. The factor matrices in HOSVD are typically estimated by finding leading left singular vectors of a given data tensor unfolded along each mode. However, a number of algorithms for estimating NTD is considerably greater. Similarly as for Nonnegative Matrix Factorization (NMF) [15], NTD can be estimated with multiplicative updates, projected gradient descent, projected least squares, and active set methods [5, 7, 12, 13, 14, 16].

In this paper, we attempt to estimate the nonnegatively constrained factor matrices with the modified GPSR-BB method that was originally proposed by Figueiredo, Nowak, and Wright [17] for reconstruction of sparse signals. The GPSR-BB is based on a similar approximation to the inverse Hessian as in the Barzilai-Borwein gradient projection method [18, 19]. This method has been extended in [5, 20] to efficiently solve nonnegatively constrained systems of linear equations with multiple right-hand sides, and then applied for NMF problems.

The paper is organized as follows: the next section reviews the selected Tucker models and the related basic algorithms. The uni-orthogonal NTD is discussed in Section 3. Section 4 is concerned with the modified GPSR-BB method for estimating nonnegative factor matrices. The classification results are presented in Section 5. Finally, the conclusions are given in the last section.

2 Tucker Models

Given a N-way tensor $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$, the Tucker model has the following form:

$$\begin{aligned} \mathcal{Y} &= \mathcal{G} \times_1 \boldsymbol{U}^{(1)} \times_2 \boldsymbol{U}^{(2)} \times_3 \ldots \times_N \boldsymbol{U}^{(N)} \\ &= \sum_{j_1=1}^{J_1} \sum_{j_2=1}^{J_2} \cdots \sum_{j_N=1}^{J_N} g_{j_1,j_2,\ldots,j_N} \boldsymbol{u}_{j_1}^{(1)} \circ \boldsymbol{u}_{j_2}^{(2)} \circ \ldots \circ \boldsymbol{u}_{j_N}^{(N)}, \end{aligned} \tag{1}$$

where $\mathcal{G} = [g_{j_1,j_2,\ldots,j_N}] \in \mathbb{R}^{J_1 \times J_2 \times \ldots \times J_N}$ is the core tensor of rank-$(J_1, J_2, \ldots, J_N)$ with $J_n \leq I_n$ for all $n = 1, \ldots, N$ and $1 \leq j_n \leq J_n$. The matrices $\boldsymbol{U}^{(1)} = [\boldsymbol{u}_1^{(1)}, \ldots, \boldsymbol{u}_{J_1}^{(1)}] = [u_{i_1,j_1}] \in \mathbb{R}^{I_1 \times J_1}$, $\boldsymbol{U}^{(2)} = [\boldsymbol{u}_1^{(2)}, \ldots, \boldsymbol{u}_{J_2}^{(2)}] = [u_{i_2,j_2}] \in \mathbb{R}^{I_2 \times J_2}$, $\boldsymbol{U}^{(N)} = [\boldsymbol{u}_1^{(N)}, \ldots, \boldsymbol{u}_{J_N}^{(N)}] = [u_{i_N,j_N}] \in \mathbb{R}^{I_N \times J_N}$ are factor matrices, where

$i_n = 1, \ldots, I_n$, $j_n = 1, \ldots, J_n$, and $n = 1, \ldots, N$. The symbol $\times_n$ denotes the n-mode tensor product, and $\boldsymbol{u}^{(k)} \circ \boldsymbol{u}^{(l)} = \boldsymbol{u}^{(k)}(\boldsymbol{u}^{(l)})^T \in \mathbb{R}^{M \times N}$ is the outer product of the vectors $\boldsymbol{u}^{(k)} \in \mathbb{R}^M$ and $\boldsymbol{u}^{(l)} \in \mathbb{R}^N$.

In the original Tucker model [1] and in HOSVD [2], the factor matrices are column-wise orthogonal, i.e. $(\boldsymbol{U}^{(1)})^T\boldsymbol{U}^{(1)} = \boldsymbol{I}_{J_1}$, $(\boldsymbol{U}^{(2)})^T\boldsymbol{U}^{(2)} = \boldsymbol{I}_{J_2}$, ..., $(\boldsymbol{U}^{(N)})^T\boldsymbol{U}^{(N)} = \boldsymbol{I}_{J_N}$, where $\boldsymbol{I}_{J_1} \in \mathbb{R}^{J_1 \times J_1}$, $\boldsymbol{I}_{J_2} \in \mathbb{R}^{J_2 \times J_2}$, $\boldsymbol{I}_{J_N} \in \mathbb{R}^{J_N \times J_N}$ are identity matrices. In contrary to SVD of a matrix, the core tensor $\mathcal{G}$ in HOSVD is not a super diagonal tensor but it is rather a dense tensor. For all $n = 1, \ldots, N$, the column-wise orthogonal factor $\boldsymbol{U}^{(n)}$ can be computed from the SVD of the n-mode unfolded tensor $\mathcal{Y}$. Let $\boldsymbol{Y}_{(n)} \in \mathbb{R}^{I_n \times \prod_{p \neq n} I_p}$ be a matrix that is obtained from the tensor $\mathcal{Y}$ by unfolding it along n-mode. Thus $\boldsymbol{U}^{(n)} = [\boldsymbol{u}_1^{(n)}, \ldots, \boldsymbol{u}_{J_n}^{(n)}] \in \mathbb{R}^{I_n \times J_n}$, where $\boldsymbol{u}_j^{(n)}$ is the j-th left singular vector of $\boldsymbol{Y}_{(n)}$ or the j-th leading eigenvector of the symmetric semi-positive defined matrix $\boldsymbol{Y}_{(n)}(\boldsymbol{Y}_{(n)})^T \in \mathbb{R}^{I_n \times I_n}$. Having the factor matrices $\{\boldsymbol{U}^{(n)}\}$, the core tensor can be readily updated with the formula $\mathcal{G} \leftarrow \mathcal{Y} \times_1 (\boldsymbol{U}^{(1)})^T \times_2 (\boldsymbol{U}^{(2)})^T \times_3 \ldots \times_N (\boldsymbol{U}^{(N)})^T$.

In NTD [3, 4], the core tensor and factor matrices are all nonnegative, i.e. $g_{j_1,j_2,\ldots,j_N} \geq 0$ and $u_{i_n,j_n} \geq 0$ for $i_n = 1, \ldots, I_n$, $j_n = 1, \ldots, J_n$, and $n = 1, \ldots, N$. The nonnegative factor matrices $\boldsymbol{U}^{(n)}$ are updated alternatingly – similarly as the factors in NMF [15]. To apply the alternating optimization procedure, note that the mode-n unfolding of the model (1) is as follows:

$$\begin{aligned} \boldsymbol{Y}_{(n)} &= \boldsymbol{U}^{(n)}\boldsymbol{G}_{(n)} \left(\boldsymbol{U}^{(N)} \otimes \ldots \otimes \boldsymbol{U}^{(n+1)} \otimes \boldsymbol{U}^{(n-1)} \otimes \ldots \otimes \boldsymbol{U}^{(1)}\right)^T \\ &= \boldsymbol{U}^{(n)}\boldsymbol{Z}^{(n)}, \end{aligned} \tag{2}$$

where $\boldsymbol{G}_{(n)} \in \mathbb{R}^{J_n \times \prod_{p \neq n} J_p}$ is the unfolded tensor $\mathcal{G}$ along the n-mode, and the symbol $\otimes$ denotes the Kronecker product. Applying the projected ALS algorithm to (2), we have:

$$\boldsymbol{U}^{(n)} = \left[\boldsymbol{Y}_{(n)}(\boldsymbol{Z}^{(n)})^T(\boldsymbol{Z}^{(n)}(\boldsymbol{Z}^{(n)})^T)^{-1}\right]_+, \quad n = 1, \ldots, N, \tag{3}$$

where $[\xi]_+ = \max\{0, \xi\}$ is the projection of ξ onto the nonnegative orthant of $\mathbb{R}$. The core tensor $\mathcal{G}$ can be updated with the formula:

$$\mathcal{G} \leftarrow \left[\mathcal{Y} \times_1 (\boldsymbol{U}^{(1)})^\dagger \times_2 (\boldsymbol{U}^{(2)})^\dagger \times_3 \ldots \times_N (\boldsymbol{U}^{(N)})^\dagger\right]_+, \tag{4}$$

where $(\boldsymbol{U}^{(n)})^\dagger = \left((\boldsymbol{U}^{(n)})^T(\boldsymbol{U}^{(n)})\right)^{-1}(\boldsymbol{U}^{(n)})^T \in \mathbb{R}^{J_n \times I_n}$ is the Moore-Penrose pseudoinverse of $\boldsymbol{U}^{(n)}$ for $n = 1, \ldots, N$. The columns of the nonnegative factor matrices $\boldsymbol{U}^{(n)}$ are often normalized to the unit l_p norm, i.e. $\boldsymbol{u}_l^{(n)} \leftarrow \frac{\boldsymbol{u}_l^{(n)}}{||\boldsymbol{u}_l^{(n)}||_p}$, where $p = 1$ or $p = 2$, and $l = 1, \ldots, J_n$, $n = 1, \ldots, N$.

3 Uni-orthogonal NTD

We assume that the training and testing images have the same resolution ($I_1 \times I_2$), and the training images arranged along the mode-3 form the 3-way tensor

$\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, where I_3 is the number of training images. Thus $N = 3$, and the N-way Tucker model (1) simplifies to the Tucker3 model [6].

In our approach, we have the following model:

$$\mathcal{Y} = \mathcal{G} \times_1 \boldsymbol{U}^{(1)} \times_2 \boldsymbol{U}^{(2)} \times_3 \boldsymbol{U}^{(3)}, \tag{5}$$

where $\mathcal{G} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$ is the core tensor, the factor matrices $\boldsymbol{U}^{(1)} = [u^{(1)}_{i_1,j_1}] \in \mathbb{R}^{I_1 \times J_1}$ and $\boldsymbol{U}^{(2)} = [u^{(2)}_{i_2,j_2}] \in \mathbb{R}^{I_2 \times J_2}$ are nonnegative ($u^{(1)}_{i_1,j_1}, u^{(2)}_{i_2,j_2} \geq 0$), and the factor matrix $\boldsymbol{U}^{(3)} \in \mathbb{R}^{I_3 \times J_3}$ is column-wise orthogonal, i.e. $(\boldsymbol{U}^{(3)})^T \boldsymbol{U}^{(3)} = \boldsymbol{I}_{J_3}$. The number J_3 should be equal to the number of classes, and the numbers J_1 and J_2 should satisfy the conditions: $1 \leq J_1 << I_1$ and $1 \leq J_2 << I_2$. Thus, our Uni-Orthogonal NTD (UO-NTD) is given by Algorithm 1.

Algorithm 1. UO-NTD

Input : $\mathcal{Y} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, J_1, J_2, J_3 - lower ranks, k_{max} - number of inner iterations

Output: Factor matrices $\boldsymbol{U}^{(1)} \in \mathbb{R}^{I_1 \times J_1}$, $\boldsymbol{U}^{(2)} \in \mathbb{R}^{I_2 \times J_2}$ and $\boldsymbol{U}^{(3)} \in \mathbb{R}^{I_3 \times J_3}$, $\mathcal{G} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$ - core tensor

1 Initialize (randomly) $\boldsymbol{U}^{(1)}$ and $\boldsymbol{U}^{(2)}$ with positive numbers, and $\boldsymbol{U}^{(3)}$ and $\mathcal{G}$ with real numbers ;

2 **repeat**

3 $\quad \boldsymbol{Z}^{(1)} = \boldsymbol{G}_{(1)}(\boldsymbol{U}^{(3)} \otimes \boldsymbol{U}^{(2)})^T$;

4 $\quad \boldsymbol{U}^{(1)} \leftarrow \texttt{gpsrbb}(\boldsymbol{Y}_{(1)}, \boldsymbol{Z}^{(1)}, \boldsymbol{U}^{(1)}, k_{max})$; `// Update for` $\boldsymbol{U}^{(1)}$

5 $\quad \boldsymbol{u}^{(1)}_l \leftarrow \frac{\boldsymbol{u}^{(1)}_l}{||\boldsymbol{u}^{(1)}_l||_2}$, where $l = 1, \ldots, J_1$; `// Normalization of` $\boldsymbol{U}^{(1)}$

6 $\quad \boldsymbol{Z}^{(2)} = \boldsymbol{G}_{(2)}(\boldsymbol{U}^{(3)} \otimes \boldsymbol{U}^{(1)})^T$;

7 $\quad \boldsymbol{U}^{(2)} \leftarrow \texttt{gpsrbb}(\boldsymbol{Y}_{(2)}, \boldsymbol{Z}^{(2)}, \boldsymbol{U}^{(2)}, k_{max})$; `// Update for` $\boldsymbol{U}^{(2)}$

8 $\quad \boldsymbol{u}^{(2)}_l \leftarrow \frac{\boldsymbol{u}^{(2)}_l}{||\boldsymbol{u}^{(2)}_l||_2}$, where $l = 1, \ldots, J_2$; `// Normalization of` $\boldsymbol{U}^{(2)}$

9 $\quad \boldsymbol{Z}^{(3)} = \boldsymbol{G}_{(3)}(\boldsymbol{U}^{(2)} \otimes \boldsymbol{U}^{(1)})^T$;

10 $\quad \boldsymbol{U}^{(3)} = \boldsymbol{Y}_{(3)}(\boldsymbol{Z}^{(3)})^T(\boldsymbol{Z}^{(3)}(\boldsymbol{Z}^{(3)})^T)^{-1}$; `// Update for` $\boldsymbol{U}^{(3)}$

11 $\quad \boldsymbol{u}^{(3)}_l \leftarrow \frac{\boldsymbol{u}^{(3)}_l}{||\boldsymbol{u}^{(3)}_l||_2}$, where $l = 1, \ldots, J_3$; `// Normalization of` $\boldsymbol{U}^{(3)}$

12 $\quad \boldsymbol{U}^{(3)} \leftarrow \boldsymbol{U}^{(3)} \left((\boldsymbol{U}^{(3)})^T \boldsymbol{U}^{(3)} \right)^{-1/2}$; `// Column-wise orthogonalization`

13 $\quad \mathcal{G} \leftarrow \mathcal{Y} \times_1 (\boldsymbol{U}^{(1)})^\dagger \times_2 (\boldsymbol{U}^{(2)})^\dagger \times_3 (\boldsymbol{U}^{(3)})^T$; `// Update for` $\mathcal{G}$

14 **until** `Stop criterion` is satisfied ;

The GPSR-BB algorithm given in Steps 4 and 7 in Algorithm 1 is described in Section 4. The stop criterion in Algorithm 1 can be determined with many rules. It might be a fixed number of iterations (usually less than 50) or the truncation of iterations when the normalized residual error drops below a certain threshold.

Each lateral slice (along mode-3) of the tensor $\mathcal{G} \times_1 \boldsymbol{U}^{(1)} \times_2 \boldsymbol{U}^{(2)} \in \mathbb{R}^{I_1 \times I_2 \times J_3}$ can be considered as a basis image that has rather holistic nature (similarly as in

PCA) than a part-based representation. For classification of handwritten digits, each base image is expected to represent each digit. Each lateral image in the tensor $\mathcal{Y}$ is therefore a linear combination of the basis images. The coefficients of that linear combination are given by row vectors of the factor matrix $\boldsymbol{U}^{(3)}$ that can be regarded as encoding vectors. Classification can be performed in the low-dimensional space of encoding vectors.

Unsupervised classification can be obtained by clustering the encoding vectors. For supervised classification the encoding vectors for testing images should be computed using the basis images that have been already estimated for training images. Then, each testing image can be classified according to the highest similarity in the space of encoding vectors.

The algorithm of supervised classification is given by Algorithm 2. The testing images are collected in the tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times R}$ in the similar way as training images in $\mathcal{Y}$, where R is the number of testing images. The unfolded tensor $\mathcal{T}$ with respect to the mode-3 is given by the matrix $\boldsymbol{T}_{(3)}$. Indices of the classes to which training images belong are given in the vector $\boldsymbol{c}^{(train)} \in \mathbb{R}^{I_3}$. Algorithm 2 returns the vector $\boldsymbol{c}^{(test)} \in \mathbb{R}^{R}$ that contains the classes of testing images.

Algorithm 2. Supervised classification

Input : $\mathcal{G} \in \mathbb{R}^{J_1 \times J_2 \times J_3}$, $\boldsymbol{U}^{(1)} \in \mathbb{R}^{I_1 \times J_1}$, $\boldsymbol{U}^{(2)} \in \mathbb{R}^{I_2 \times J_2}$ and $\boldsymbol{U}^{(3)} \in \mathbb{R}^{I_3 \times J_3}$, $\boldsymbol{c}^{(train)} \in \mathbb{R}^{I_3}$ - classes of training images, $\mathcal{T} \in \mathbb{R}^{I_1 \times I_2 \times R}$ - testing images

Output: $\boldsymbol{c}^{(test)} \in \mathbb{R}^{R}$ - classes of testing images

1 Initialize (randomly) $\boldsymbol{U}^{(1)}$ and $\boldsymbol{U}^{(2)}$ with positive numbers, and $\boldsymbol{U}^{(3)}$ and $\mathcal{G}$ with real numbers ;

2 $\boldsymbol{Z}^{(3)} = \boldsymbol{G}_{(3)}(\boldsymbol{U}^{(2)} \otimes \boldsymbol{U}^{(1)})^T$;

3 $\boldsymbol{U}^{(test)} = \boldsymbol{T}_{(3)}(\boldsymbol{Z}^{(3)})^T(\boldsymbol{Z}^{(3)}(\boldsymbol{Z}^{(3)})^T)^{-1}$; `// Encoding vectors`

4 $\boldsymbol{u}_l^{(test)} \leftarrow \frac{\boldsymbol{u}_l^{(test)}}{||\boldsymbol{u}_l^{(test)}||_2}$, where $l = 1, \ldots, J_3$; `// Normalization of` $\boldsymbol{U}^{(test)}$

5 $\boldsymbol{c}^{(test)} \leftarrow$ `knnclassify`$(\boldsymbol{U}^{(test)}, \boldsymbol{U}^{(3)}, \boldsymbol{c}^{(train)}, 1, 'cosine')$; `// Matlab function`

The `knnclassify` function in Step 5 of Algorithm 2 comes from the *Bioinformatics Toolbox* in Matlab 2008. It uses the nearest-neighbor method for classification of the rows in the matrix $\boldsymbol{U}^{(test)}$ into one of the classes of the the matrix $\boldsymbol{U}^{(3)}$. We used only one nearest neighbor, and the cosine measure to determine the similarity between samples (row vectors).

4 Modified GPSR-BB Algorithm

To solve the system (2) with respect to the nonnegativity constrained $\boldsymbol{U}^{(n)}$, we formulate the Nonnegative Least Squares (NNLS) problem:

$$\min_{\mathbf{U}^{(n)} \geq \mathbf{0}} \Psi(\boldsymbol{U}^{(n)}), \quad \text{where} \quad \Psi(\boldsymbol{U}^{(n)}) = \frac{1}{2}||\boldsymbol{Y}_{(n)} - \boldsymbol{U}^{(n)}\boldsymbol{Z}^{(n)}||_F^2 \tag{6}$$

which can be solved with the modified GPSR-BB method [17] that is based on the Spectral Projected Gradient (SPG) method [21]. For (6), the SPG takes the form of the following updates:

$$\boldsymbol{U}^{(n)} \leftarrow \boldsymbol{U}^{(n)} - \operatorname{diag}\{\boldsymbol{\lambda}^{(n)}\}\boldsymbol{D}^{(n)}, \tag{7}$$

with the search direction defined by

$$\boldsymbol{D}^{(n)} = \left[\boldsymbol{U}^{(n)} - \operatorname{diag}\{\boldsymbol{\alpha}^{(n)}\}\nabla_{\mathbf{U}^{(n)}}\Psi(\boldsymbol{U}^{(n)})\right]_{+} - \boldsymbol{U}^{(n)}, \tag{8}$$

where the step length $\boldsymbol{\lambda}^{(n)} \in [0,1]^{I_n}$ minimizes $\Psi(\boldsymbol{U}^{(n)} - \operatorname{diag}\{\boldsymbol{\lambda}^{(n)}\}\boldsymbol{D}^{(n)})$ and $\boldsymbol{\alpha}^{(n)} \in \mathbb{R}^{I_n}$ should be selected such that the matrix $\boldsymbol{H}^{(n)} = \operatorname{diag}\{\boldsymbol{\alpha}^{(n)}\}$ approximates the inverse to the Hessian of $\Psi(\boldsymbol{U}^{(n)})$. This approach comes from the Barzilai-Borwein gradient projection method [18, 19], thus the updates for the search direction (8) have the quasi-Newton nature.

The factors $\boldsymbol{\alpha}^{(n)}$ can be computed from the secant equation which for the quasi-Newton update in (8) takes the form: $\boldsymbol{H}_{k+1}^{(n)}\boldsymbol{S}_k^{(n)} = \boldsymbol{W}_k^{(n)}$, where $\boldsymbol{S}_k^{(n)} = \boldsymbol{U}_{k+1}^{(n)} - \boldsymbol{U}_k^{(n)}$ and $\boldsymbol{W}_k^{(n)} = \nabla_{\mathbf{U}^{(n)}}\Psi(\boldsymbol{U}_{k+1}^{(n)}) - \nabla_{\mathbf{U}^{(n)}}\Psi(\boldsymbol{U}_k^{(n)}) = -\operatorname{diag}\{\boldsymbol{\lambda}^{(n)}\}\boldsymbol{D}^{(n)}$, and k is the number of an iterative step. From the secant equation, we have:

$$\begin{aligned}
\bar{\boldsymbol{\alpha}}_{k+1}^{(n)} &= \frac{\operatorname{diag}\left\{\boldsymbol{W}_k^{(n)}(\boldsymbol{S}_k^{(n)})^T\right\}}{\operatorname{diag}\left\{\boldsymbol{S}_k^{(n)}(\boldsymbol{S}_k^{(n)})^T\right\}} = \frac{\operatorname{diag}\left\{\boldsymbol{S}_k^{(n)}\boldsymbol{Z}^{(n)}(\boldsymbol{Z}^{(n)})^T(\boldsymbol{S}_k^{(n)})^T\right\}}{\operatorname{diag}\left\{\boldsymbol{S}_k^{(n)}(\boldsymbol{S}_k^{(n)})^T\right\}} \\
&= \frac{\operatorname{diag}\left\{\boldsymbol{D}^{(n)}\boldsymbol{Z}^{(n)}(\boldsymbol{Z}^{(n)})^T(\boldsymbol{D}^{(n)})^T\right\}}{\operatorname{diag}\left\{\boldsymbol{D}^{(n)}(\boldsymbol{D}^{(n)})^T\right\}} \\
&= \frac{\left[\boldsymbol{D}^{(n)} \circledast (\boldsymbol{D}^{(n)}\boldsymbol{Z}^{(n)}(\boldsymbol{Z}^{(n)})^T)\right]\mathbf{1}_{J_n}}{\left[\boldsymbol{D}^{(n)} \circledast \boldsymbol{D}^{(n)}\right]\mathbf{1}_{J_n}},
\end{aligned} \tag{9}$$

where the symbol $\circledast$ denotes the Hadamard multiplication, $\operatorname{diag}\{\boldsymbol{X}\}$ is a vector created from main diagonal entries of the matrix $\boldsymbol{X}$, and $\mathbf{1}_{J_n} = [1,\ldots,1]^T \in \mathbb{R}^{J_n}$.

Inserting (7) to (6), and from $\frac{\partial}{\partial \boldsymbol{\lambda}^{(n)}}\Psi(\boldsymbol{U}^{(n)}) \triangleq 0$, we get the update for $\boldsymbol{\lambda}^{(n)}$ in the closed-form:

$$\bar{\boldsymbol{\lambda}}^{(n)} \leftarrow \frac{\left[\boldsymbol{D}^{(n)} \circledast \nabla_{\mathbf{U}^{(n)}}\Psi(\boldsymbol{U}^{(n)})\right]\mathbf{1}_{J_n}}{\left[\boldsymbol{D}^{(n)} \circledast (\boldsymbol{D}^{(n)}\boldsymbol{Z}^{(n)}(\boldsymbol{Z}^{(n)})^T)\right]\mathbf{1}_{J_n}}. \tag{10}$$

The final form of the modified GPSR-BB algorithm is given by Algorithm 3.

5 Classification Results

The proposed UO-NTD algorithm has been tested for various supervised image classification problems. For the reference, the ALS-NTD algorithm that updates

Algorithm 3. GPSR-BB Algorithm

Input : $\boldsymbol{Y}_{(n)}$, $\boldsymbol{Z}^{(n)}$, $\boldsymbol{U}^{(n)}$ - initial guess, k_{max}, α_{min}, α_{max}
Output: $\boldsymbol{U}^{(n)}$ - mode-n factor matrix,

for $k = 1, 2, \ldots, k_{max}$ **do**
- $\boldsymbol{F}^{(n)} = \nabla_{\mathbf{U}^{(n)}} \Psi(\boldsymbol{U}^{(n)}) = (\boldsymbol{U}^{(n)} \boldsymbol{Z}^{(n)} - \boldsymbol{Y}_{(n)})(\boldsymbol{Z}^{(n)})^T$; `// Gradient`
- $\boldsymbol{D}^{(n)} = \left[\boldsymbol{U}^{(n)} - \mathrm{diag}\{\boldsymbol{\alpha}^{(n)}\}\boldsymbol{F}^{(n)}\right]_+ - \boldsymbol{U}^{(n)}$; `// Search direction`
- $\boldsymbol{\lambda}^{(n)} \leftarrow \max\{0, \min\{1, \bar{\boldsymbol{\lambda}}^{(n)}\}\}$; `// where` $\bar{\boldsymbol{\lambda}}^{(n)}$ `is given by (10)`
- $\boldsymbol{U}^{(n)} \leftarrow \boldsymbol{U}^{(n)} - \mathrm{diag}\{\boldsymbol{\lambda}^{(n)}\}\boldsymbol{D}^{(n)}$;
- $\boldsymbol{\alpha}^{(n)} \leftarrow \max\{\alpha_{min}, \min\{\alpha_{max}, \bar{\boldsymbol{\alpha}}^{(n)}\}\}$; `// where` $\bar{\boldsymbol{\alpha}}^{(n)}$ `is given by (9)`

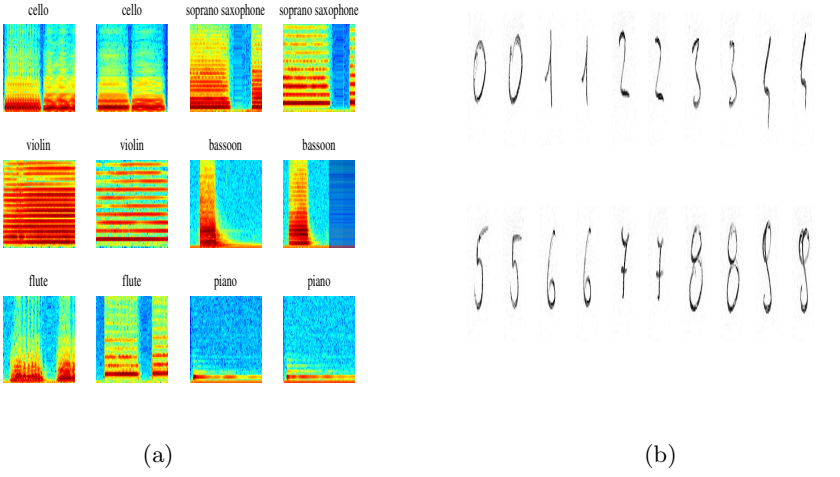
the factor matrices with the rule (3) and the core tensor with the update (4) is chosen. We analyze three classification problems of: (A) musical instruments, (B) hand-written digits, (C) facial images.

For analyzing the problem A, the audio recordings of 6 musical instruments (cello, soprano saxophone, violin, bassoon, flute, and piano) are selected from the MIS database[1] of the University of Iowa. Each audio recording at the sampling rate of 44.1kHz is restricted to contain meaningful information of about 4 sec long. The training and testing sets contain totally 56 and 12 samples, respectively. All the samples are transformed to log-magnitude spectrograms into the frequency range from 86Hz to 10.9kHz, and the time window from 0 do 4 seconds. Then, the spectrogram are downsampled to 64 frequencies $\times$ 128 time intervals. Thus $\mathcal{Y} \in \mathbb{R}^{64\times128\times56}$ and $\mathcal{T} \in \mathbb{R}^{64\times128\times12}$. For this case, we set $J_1 = J_2 = 20$, and $J_3 = 6$. The spectrograms of the testing samples are depicted in Fig. 1(a).

The samples for the problem B are images of hand-written digits. Each class in the training set is represented by 8 images of the resolution downsampled to 64×64 pixels. This gives $\mathcal{Y} \in \mathbb{R}^{64\times64\times80}$ for 10 digits from 0 to 9. The testing set consists of 20 images (2 by each class) - thus $\mathcal{T} \in \mathbb{R}^{64\times64\times20}$. We set $J_1 = J_2 = 20$, and $J_3 = 10$. The testing samples are illustrated in Fig. 1(b).

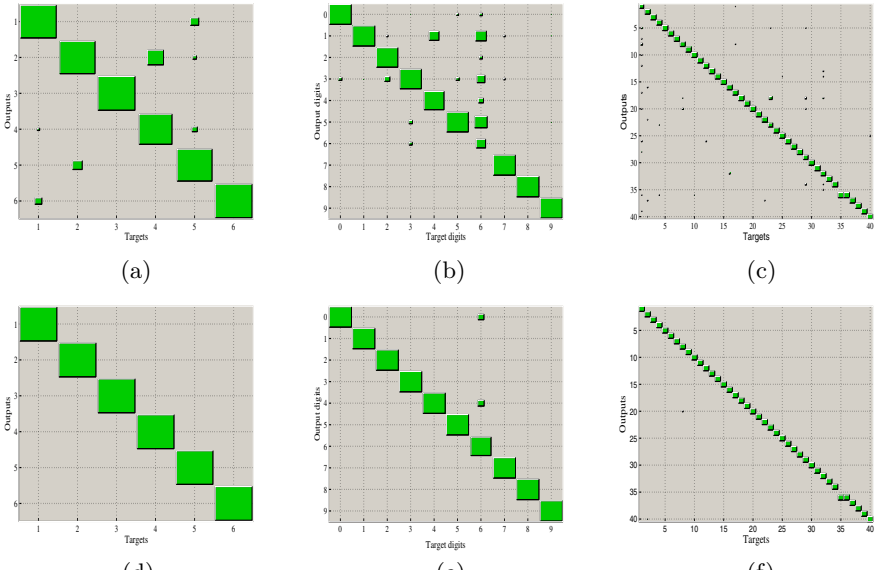
The problem C is concerned with classification of facial images from the ORL database [2] that contains 400 frontal face images of 40 people (10 pictures per person). The images were taken at different times (between April 1992 and April 1994 at the AT&T Laboratories Cambridge), varying the lighting, facial expressions (open / closed eyes, smiling / not smiling) and facial details (glasses / no glasses). All the images have a dark homogeneous background with the subjects in an upright, frontal position. The whole set is randomly divided into 320 training images containing all the classes and 80 testing images. The resolution of the images is 112×92 pixels. Despite the number of classes is 40, we noticed that setting $J_1 = J_2 = J_3 = 20$ gives nearly the same recognition rate as for $J_3 = 40$ but in a considerably shorter time.

[1] http://theremin.music.uiowa.edu
[2] http://people.cs.uchicago.edu/∼dinoj/vis/orl/



(a) (b)

Fig. 1. Testing samples: (a) spectrograms for the problem C; (b) hand-written digits



(a) (b) (c)

(d) (e) (f)

Fig. 2. Confusion matrices: (a) ALS-NTD: 6 musical instruments; (b) ALS-NTD: 10 hand-written digits; (c) ALS-NTD: 40 facial images; (d) UO-NTD: 6 musical instruments; (e) UO-NTD: 10 hand-written digits; (f) UO-NTD: 40 facial images

Each tested algorithm is initiated 100 times, starting from a random initial guess for factor matrices and a core tensor. Both algorithms are run for 20 iterations, and the GPSR-BB algorithm is executed with the settings: $k_{max} = 2$, $\alpha_{min} = 10^{-8}$, $\alpha_{max} = 1$. The averaged results of supervised classification are illustrated in Fig. 2 with the Hinton graph of the confusion matrix. Both the Hinton graph and the confusion matrix are obtained with the Matlab functions from the *Neural Network Toolbox*. Furthermore, the recognition rates and elapsed time averaged over 100 trials are presented in Table 1.

Table 1. Mean recognition rates, standard deviations, and 20 iteration elapsed time averaged over 100 runs of the tested algorithms for problems A, B, and C

Problem	ALS-NTD			UO-NTD		
	Rec. rate [%]	Std. [%]	Time [sec]	Rec. rate [%]	Std. [%]	Time [sec]
Problem A	91.92	8.41	3.67	99.67	1.64	3.27
Problem B	86.35	6.85	2.57	98.25	2.86	2.34
Problem C	92.5	2	20.8	97.36	0.39	19.8

6 Conclusions

The UO-NTD algorithm classifies all the tested images more accurately and with a lower variation of the results than the standard ALS-NTD algorithm. The elapsed time for the UO-NTD is only slightly shorter than for the others. When an outer-class correlation is very strong both algorithms are not able to classify such results. This occurs, e.g. between the subjects 34 and 35 in the problem C (see Figs. 2(c,f)) or the digits 6 and 4 in the problem B (see Figs. 2(b,e)). To tackle this problem, one may incorporate, e.g. Fisher discriminant information to the training process, however, this results in difficulty in determining inner- and outer-class correlations. The underlying iterative algorithms (GPSR-BB or ALS) update the factors with inconsistent and usually ill-posed training data, especially as the clusters are partially overlapping, and hence regularization by truncated iterations is essential. We set up 20 iterations and a considerable increase in this number may lead to over-training behavior.

Acknowledgment. This work was partially supported by the habilitation grant N N515 603139 (2010-2012) from the Ministry of Science and Higher Education, Poland.

References

[1] Tucker, L.R.: Some mathematical notes on three-mode factor analysis. Psychometrika 31, 279–311 (1966)

[2] De Lathauwer, L., de Moor, B., Vandewalle, J.: A multilinear singular value decomposition. SIAM Journal of Matrix Analysis and Applications 21, 1253–1278 (2001)

[3] Kiers, H.A.L.: A three-step algorithm for CANDECOMP/PARAFAC analysis of large data sets with multicollinearity. Journal of Chemometrics 12(3), 155–171 (1998)
[4] Smilde, A., Bro, R., Geladi, P.: Multi-way Analysis: Applications in the Chemical Sciences. John Wiley and Sons, New York (2004)
[5] Cichocki, A., Zdunek, R., Phan, A.H., Amari, S.I.: Nonnegative Matrix and Tensor Factorizations: Applications to Exploratory Multi-way Data Analysis and Blind Source Separation. Wiley and Sons, Chichester (2009)
[6] Kolda, T.G., Bader, W.: Tensor decompositions and applications. SIAM Review 51(3), 455–500 (2009)
[7] Mørup, M., Hansen, L.K., Arnfred, S.M.: Algorithms for sparse nonnegative Tucker decompositions. Neural Computation 20(8), 2112–2131 (2008)
[8] Vasilescu, M.A.O., Terzopoulos, D.: Multilinear analysis of image ensembles: TensorFaces. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2350, pp. 447–460. Springer, Heidelberg (2002)
[9] Wang, H., Ahuja, N.: Facial expression decomposition. In: Proc. of the Ninth IEEE International Conference on Computer Vision, vol. 2, pp. 958–965 (2003)
[10] Vlasic, D., Brand, M., Phister, H., Popovic, J.: Face transfer with multilinear models. ACM Transactions on Graphics 24, 426–433 (2005)
[11] Savas, B., Eldén, L.: Handwritten digit classification using higher order singular value decomposition. Pattern Recognition 40(3), 993–1003 (2007)
[12] Kim, Y.D., Choi, S.: Nonnegative tensor decomposition. In: Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2007), Minneapolis, MN, pp. 1–8 (2007)
[13] Phan, A.H., Cichocki, A.: Tensor decompositions for feature extraction and classification of high dimensional datasets. IEICE Nonlinear Theory and Its Applications 1(1), 37–68 (2010)
[14] Phan, A.H., Cichocki, A., Vu-Dinh, T.: Classification of scenes based on multiway feature extraction. In: Proc. 2010 International Conference on Advanced Technologies for Communications, Ho Chi Minh City, Vietnam, pp. 142–145 (2010)
[15] Lee, D.D., Seung, H.S.: Learning of the parts of objects by non-negative matrix factorization. Nature 401, 788–791 (1999)
[16] Kim, H., Park, H., Elden, L.: Non-negative tensor factorization based on alternating large-scale non-negativity-constrained least squares. In: Proc. BIBE 2007, pp. 1147–1151 (2007)
[17] Figueiredo, M.A.T., Nowak, R.D., Wright, S.J.: Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems. IEEE Journal of Selected Topics in Signal Processing 1(4), 586–597 (2007)
[18] Barzilai, J., Borwein, J.M.: Two-point step size gradient methods. IMA Journal of Numerical Analysis 8(1), 141–148 (1988)
[19] Dai, Y.H., Fletcher, R.: Projected Barzilai-Borwein methods for large-scale box-constrained quadratic programming. Numer. Math. 100(1), 21–47 (2005)
[20] Zdunek, R., Cichocki, A.: Fast nonnegative matrix factorization algorithms using projected gradient approaches for large-scale problems. In: Computational Intelligence and Neuroscience 2008(939567) (2008)
[21] Birgin, E.G., Martínez, J.M., Raydan, M.: Nonmonotone spectral projected gradient methods on convex sets. SIAM Journal on Control and Optimization 10, 1196–1211 (2000)