TENSOR DICTIONARY LEARNING WITH SPARSE TUCKER DECOMPOSITION

Syed Zubair and Wenwu Wang

Centre for Vision, Speech and Signal Processing
University of Surrey, Guildford, GU2 7XH, UK
Emails: s.zubair@surrey.ac.uk, w.wang@surrey.ac.uk

ABSTRACT

Dictionary learning algorithms are typically derived for dealing with one or two dimensional signals using vector-matrix operations. Little attention has been paid to the problem of dictionary learning over high dimensional tensor data. We propose a new algorithm for dictionary learning based on tensor factorization using a TUCKER model. In this algorithm, sparseness constraints are applied to the core tensor, of which the n-mode factors are learned from the input data in an alternate minimization manner using gradient descent. Simulations are provided to show the convergence and the reconstruction performance of the proposed algorithm. We also apply our algorithm to the speaker identification problem and compare the discriminative ability of the dictionaries learned with those of TUCKER and K-SVD algorithms. The results show that the classification performance of the dictionaries learned by our proposed algorithm is considerably better as compared to the two state of the art algorithms.

***Index Terms*—** Tensor Factorization, Sparse Representations, Classification, Dictionary Learning

1. INTRODUCTION

Learning the features and structures of a signal is important for obtaining a succinct representation that can be used for various applications such as source separation and signal classification. Dictionary learning algorithms emerging from sparse representations have recently been used for learning such representations as given in [1]. However, these algorithms are mostly limited to one or two dimensional signals. With content-rich applications emerging nowadays, signal dimensionality is constantly increasing e.g. in video signals. Moreover, a low-dimensional signal such as an audio signal can be cast in a higher dimensional space, e.g. in a space-time-frequency domain. This preserves the structure of the signal which may otherwise be lost when used in a low dimensional form. Hence, it becomes highly desirable for those algorithms to be able to learn signal features from higher dimensional data, such as tensor data.

Tensor factorization and decomposition have recently attracted attention in the signal processing community, for processing high dimensional signals. PARAFAC [2] and TUCKER [3] decompositions are two such classical algorithms. PARAFAC decomposes the tensor as a sum of k rank-1 tensors while the TUCKER method computes the orthonormal subspaces corresponding to each mode of the tensor. This can be treated as higher order principal component analysis. However, these methods do not explicitly enforce signal sparsity despite its benefits in signal representations for various applications.

Recent effort has therefore been on extending these two algorithms by introducing additional constraints to the models with the aim of learning sparse representations of the tensors. Both non-negativity and sparsity have been used to achieve this. Inspired by the non-negative matrix factorization (NMF) techniques due to Lee and Seung [4], the authors of [5] and [6] introduced non-negative PARAFAC decomposition with multiplicative updates and applied it to various signal and image processing applications. Similarly, non-negative versions of TUCKER decomposition have also been proposed in [7] and [8].

Sparse representations of PARAFAC and TUCKER models have also been derived. In the case of TUCKER model, which is the focus of our discussion in this paper, sparse TUCKER decomposition methods have been proposed in [7] and [9]. In [7], smoothing matrices are used for each mode of the tensor to make the core tensor as well as the TUCKER factors sparse, while in [9], sparsity is introduced by penalizing its core tensor with l_1 norm and claim that this penalty can also be applied to any of the other factors of TUCKER decomposition. In both of these works, sparsity has been applied in case of non-negative TUCKER decomposition. Hence the factors of TUCKER decomposition are learned by NMF techniques with multiplicative updates as presented in [4].

In this paper, we propose a tensor dictionary learning algorithm based on the TUCKER model with sparsity constraints over its core tensor. Unlike [7] and [9], the sparsity over the core tensor is applied here in a greedy fashion. The sparse core tensor is calculated by a tensor extended version of the greedy algorithm, Tensor Orthogonal Matching Pursuit

(TOMP) [10]. Two main reasons for introducing sparsity in the core tensor are as follows:

- Unlike the standard TUCKER representation, the sparsity of the core tensor compresses the data by considering only non-zero values of the core tensor. Moreover, the input signal is represented by only those columns of mode-n dictionaries which correspond to those non-zero elements of the core tensor.

- The core tensor establishes the relationship between the elements of the dictionaries for describing the input data model. Non-sparse core tensor makes this relationship ambiguous specially in decision based applications such as classification. The sparsity of the core tensor reduces this ambiguity and clarifies the relationship between the dictionaries.

To learn tensor dictionaries of TUCKER model along each mode, we propose a gradient descent algorithm that updates the mode-n dictionaries iteratively in an alternating manner. The proposed tensor dictionary algorithm, similar to standard dictionary learning algorithms, is a two-stage iterative process: sparse coding and dictionary update. First, given initial TUCKER factors considered as dictionaries, the TOMP algorithm is used to find the sparse core tensor. Then in the second stage, the dictionaries (factors) corresponding to each mode are updated, using the gradient descent method.

The organization of the whole paper is as follows: Section 2 formulates an objective function for tensor dictionary learning problem. Section 3 presents the Tensor OMP algorithm. Section 4 describes the proposed dictionary learning method for high dimensional data, GradTensor. Section 5 shows experiments along with their results and section 6 concludes the paper.

2. PROBLEM FORMULATION AND OPTIMIZATION CRITERION

A signal of a high dimension is considered as a tensor. Here for simplicity, we consider $\underline{\mathrm{Y}}$ as a tensor of three dimensions e.g. $\underline{\mathrm{Y}} \in R^{I_1 \times I_2 \times I_3}$, where $I_n (n = 1, 2, 3)$ are the dimensions of each mode. However, the following discussion can be readily extended to the signals with a dimension greater than three. A three dimensional tensor is also called as a three-way signal. A matrix is a form of two-way signal and a vector is considered as a one-way signal. A tensor can be unfolded to a mode-n matrix form and represented as $\mathrm{Y}_{(n)}$. For a three-way tensor, the mode-n matrix can be extracted by changing all the indices in the tensor except the n-th index. Hence a three-way tensor can be unfolded into any of its mode-n matrices. For example, the mode-1 unfolded matrix of tensor $\underline{\mathrm{Y}}$, i.e. $\mathrm{Y}_{(1)}$, has a dimension $R^{I_1 \times I_2 I_3}$. Similarly the mode-2 unfolded matrix $\mathrm{Y}_{(2)}$ has a dimension $R^{I_2 \times I_1 I_3}$. Tensor decomposition for the TUCKER model is formulated as:

$$\begin{aligned} \underline{\mathrm{Y}} &= \underline{\mathrm{X}} \times_1 \mathrm{A} \times_2 \mathrm{B} \times_3 \mathrm{C} \\ &= \sum_{m_1=1}^{M_1} \sum_{m_2=1}^{M_2} \sum_{m_3=1}^{M_3} x_{m_1 m_2 m_3} \mathbf{a}_{m_1} \circ \mathbf{b}_{m_2} \circ \mathbf{c}_{m_3} \end{aligned} \quad (1)$$

where $\circ$ is the outer product between the vectors. $\mathrm{A} \in R^{I_1 \times M_1}$, $\mathrm{B} \in R^{I_2 \times M_2}$ and $\mathrm{C} \in R^{I_3 \times M_3}$ are orthogonal factor matrices composed of **a**, **b** and **c** vectors and can be considered as principal components along each mode of the tensor. $\underline{\mathrm{X}} \in R^{M_1 \times M_2 \times M_3}$ is a core tensor. This form of decomposition was suggested by [3], hence it is called TUCKER decomposition. It can be represented element-wise as

$$y_{i_1 i_2 i_3} = \sum_{m_1=1}^{M_1} \sum_{m_2=1}^{M_2} \sum_{m_3=1}^{M_3} x_{m_1 m_2 m_3}\, a_{i_1 m_1}\, b_{i_2 m_2}\, c_{i_3 m_3} \quad (2)$$
$$for \;\; i_n = 1, \ldots, I_n, n = (1, 2, 3)$$

If the core tensor $\underline{\mathrm{X}}$ is super-diagonal and $M_1 = M_2 = M_3$, then this can be considered as PARAFAC decomposition introduced by [2].

To learn tensor dictionaries with a sparsity constraint on the core tensor $\underline{\mathrm{X}}$, our objective function for model (1) takes the form:

$$\mathcal{F}(\underline{\mathrm{X}}, \mathrm{A}, \mathrm{B}, \mathrm{C}) = \min_{\underline{\mathrm{X}}, \mathrm{A}, \mathrm{B}, \mathrm{C}} \| \underline{\mathrm{Y}} - \underline{\mathrm{X}} \times_1 \mathrm{A} \times_2 \mathrm{B} \times_3 \mathrm{C} \|_F^2$$
$$s.t. \;\; x_{m_1 m_2 m_3} = 0 \; \forall \; (m_1, m_2, m_3) \notin \mathcal{M}_1 \times \mathcal{M}_2 \times \mathcal{M}_3 \quad (3)$$

where $\| \cdot \|_F$ is the Frobenius norm, $\mathcal{M}_n = [m_n^1, \ldots, m_n^{s_n}]$ denotes the subset of indices of non-zero values in the core tensor for mode n ($n = 1, 2, 3$), and s_n represents the mode-n sparsity, showing the number of selected columns of each dictionary required for the TUCKER representation. In this way, the sparsity structure of the core tensor is ***block-sparse*** and the total sparsity (i.e. the number of non-zeros) of the three way core tensor is denoted by $s = s_1 \times s_2 \times s_3$. Here we assume that the size of the core tensor $\underline{\mathrm{X}}$ is larger than or equal to the size of $\underline{\mathrm{Y}}$ ($M_n \geq I_n$).

3. TENSOR OMP

Tensor OMP (TOMP) [10] is based on the equivalence of equation (1) to the vectorized version of the tensor representation in terms of Kronecker dictionaries, i.e.

$$\begin{aligned} vec(\underline{\mathrm{Y}}) &= (\mathrm{C} \otimes \mathrm{B} \otimes \mathrm{A}) vec(\underline{\mathrm{X}}) \quad (4) \\ \mathbf{y} &= (\mathrm{C} \otimes \mathrm{B} \otimes \mathrm{A}) \mathbf{x} \quad (5) \end{aligned}$$

where $\otimes$ is the Kronecker product. $vec(\cdot)$ is obtained by stacking all the columns of mode-1 tensor $\mathrm{Y}_{(1)}$ in a single vector $\mathbf{y} \in R^{I_1 I_2 I_3}$. Equation (5) is similar to the conventional linear (matrix) sparse representation formulation where

x is a sparse vector with s number of non-zero elements. This is one of the reasons for assuming $M_n \geq I_n$ because sparse signal model is formulated with an overcomplete dictionary. The TOMP algorithm is given in Algorithm 1.

Algorithm 1: Tensor-OMP

```
Require: Dictionaries $A \in R^{I_1 \times M_1}$, $B \in R^{I_2 \times M_2}$ and C $\in R^{I_3 \times M_3}$, input signal $\underline{Y}$, maximum number of non-zeros coefficients $t_{max} \leq s$, tolerance $\epsilon$.
Output: $\underline{X}(\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3) = \underline{E}, \{\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3\}$.
Ensure: Sparse representation $\underline{Y} = \underline{X} \times_1 A \times_2 B \times_3 C$ with $x_{m_1 m_2 m_3} = 0 \; \forall \; (m_1, m_2, m_3) \notin \mathcal{M}_1 \times \mathcal{M}_2 \times \mathcal{M}_3$. $(\underline{X}(\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3)) = \underline{E}$.
1. $\mathcal{M}_n = [\emptyset](n = 1, 2, 3), \underline{R} = \underline{Y}, \underline{X} = \underline{0}, t = 1$;
2. while $|\mathcal{M}_1||\mathcal{M}_2||\mathcal{M}_3| < t_{max}$ and $||\underline{R}||_F > \epsilon$ do
3.     $[m_1^t m_2^t m_3^t] = \arg\max_{[m_1 m_2 m_3]} |\underline{R} \times_1 A^T(:, m_1) \times_2 B^T(:, m_2) \times_3 C^T(:, m_3)|$;
4.     $\mathcal{M}_n = \mathcal{M}_n \cup [[m_n^t]](n = 1, 2, 3), D_1 = A(:, \mathcal{M}_1), D_2 = B(:, \mathcal{M}_2), D_3 = C(:, \mathcal{M}_3)$;
5.     $\mathbf{e} = \arg\min_{\mathbf{u}} ||(D_3 \otimes D_2 \otimes D_1)\mathbf{u} - \mathbf{y}||_2^2$;
6.     $\underline{R} = \underline{Y} - \underline{E} \times_1 D_1 \times_2 D_2 \times_3 D_3$;
7.     $t = t + 1$;
8. end while
9. return $\{\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3\}, \underline{E}$;
```

4. PROPOSED METHOD: GRADTENSOR

The tensor dictionaries and the core tensor are computed in a two-step process. In the first step, the sparse core tensor is computed using TOMP with tensor dictionaries initialized by M_n left leading singular vectors of the mode-n matrices of input tensor $\underline{Y}$. Once the sparse core tensor is obtained, the tensor dictionaries are computed iteratively by gradient descent in an alternating manner.

Mathematically, equation (1) can be represented in an unfolded form as

$$\begin{aligned} Y_{(1)} &= AX_{(1)}(C \otimes B)^T \\ Y_{(2)} &= BX_{(2)}(C \otimes A)^T \\ Y_{(3)} &= CX_{(3)}(B \otimes A)^T \end{aligned} \tag{6}$$

To calculate mode-1 dictionary A in the unfolded form, the minimization of equation (3) can be written as

$$\min_{A} \parallel Y_{(1)} - AX_{(1)}(C \otimes B)^T \parallel_F^2 \tag{7}$$

From (7), the gradient of the error norm with respect to A can be calculated by

$$\nabla \mathcal{F}_A = (Y_{(1)} - AX_{(1)}(C \otimes B)) \left\{X_{(1)}(C \otimes B)^T\right\}^\dagger \tag{8}$$

where † is the pseudo-inverse of the matrix. Similarly, the gradients of the objective function with respect to B and C can be calculated as

$$\begin{aligned} \nabla \mathcal{F}_B &= (Y_{(2)} - BX_{(2)}(C \otimes A)) \left\{X_{(2)}(C \otimes A)\right\}^\dagger \\ \nabla \mathcal{F}_C &= (Y_{(3)} - CX_{(3)}(B \otimes A)) \left\{X_{(3)}(B \otimes A)\right\}^\dagger \end{aligned} \tag{9}$$

These gradients are then used to update the tensor dictionaries. The updates are given by

$$\begin{aligned} A^{(k+1)} &= A^{(k)} - \gamma \nabla \mathcal{F}_A \\ B^{(k+1)} &= B^{(k)} - \gamma \nabla \mathcal{F}_B \\ C^{(k+1)} &= C^{(k)} - \gamma \nabla \mathcal{F}_C \end{aligned} \tag{10}$$

where γ is the step size and k is the current step of the gradient descent algorithm. These tensor dictionaries are learned in an alternate minimization manner such that when learning one dictionary like A, all the other dictionaries and the core tensor are held fixed. In this way, all the dictionaries are updated. In the next iteration, these learned dictionaries are used to find out the sparse core tensor in the sparse coding stage. This two-stage learning process alternates between tensor dictionaries learning and sparse core tensor update until a stopping criterion is reached. Algorithm 1 gives the summary of the whole algorithm.

As these dictionaries are learned by the gradient descent, the minimization of the objective function may lead to local minima. To improve convergence, all the dictionaries are initialized by the left leading factors of the input tensor data. Though we don't have an explicit proof for the convergence of the algorithm, yet the simulations on synthetic data show the good convergence of the algorithm as confirmed in the next section.

Algorithm 2: GradTensor

```
Task: Find mode-n dictionaries $A \in R^{I_1 \times M_1}$, $B \in R^{I_2 \times M_2}$ and $C \in R^{I_3 \times M_3}$ and sparse core tensor $\underline{X} \in R^{M_1 \times M_2 \times M_3}$ that give sparsest representation of input signal tensor $\underline{Y} \in R^{I_1 \times I_2 \times I_3}$ with predefined sparsity $s = s_1 \times s_2 \times s_3$.

Require: Input signal $\underline{Y}$, sparse core tensor $\underline{X}$, maximum sparsity value (total number of non-zeros) $s$, step size $\gamma$, tolerance $\epsilon_1$ and $\epsilon_2$,
Output: A,B,C.
Initialization: Mode-n dictionaries A, B and C each initialized by $M_n$ left leading vectors of $Y_n$, where $n = 1, 2, 3$ is the index of the modes of a tensor.

Repeat until convergence:( i.e. $\mathcal{F} \leq \epsilon_2$ )
1. Sparse Coding Stage: Use TOMP to find sparse core tensor $\underline{X}$ by solving eq (3).
```

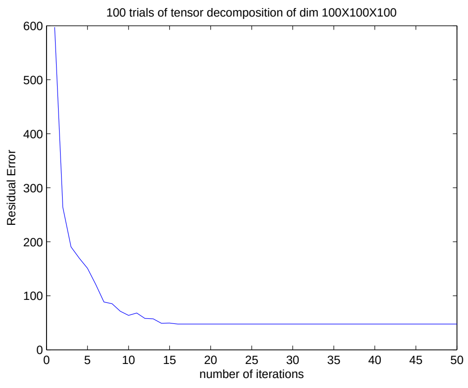

Fig. 1. Convergence of GradTensor over 100 trials.

2. *Dictionary Learning Stage*: Learn dictionaries A, B, C for each mode by the gradient descent.
   - Calculate gradient for A, $\nabla\mathcal{F}_A$. While fixing all the other dictionaries and sparse core tensor, update A by (8) and (10) for A until the error between two consecutive iterations reaches below or equal to ϵ_1.
   - Calculate gradient for B, $\nabla\mathcal{F}_B$. While fixing all the other dictionaries and sparse core tensor, update B by using (9) and (10) for B until the error between two consecutive iterations reaches below or equal to ϵ_1.
   - Calculate gradient for C, $\nabla\mathcal{F}_C$. While fixing all the other dictionaries and sparse core tensor, update C by using (9) and (10) for C until the error between two consecutive iterations reaches below or equal to ϵ_1.

5. EXPERIMENTS AND RESULTS

We perform three different experiments to analyse our algorithm for different applications. Synthetic data is used to examine the convergence of the algorithm. The second experiment includes image reconstruction by the learned tensor dictionaries with the sparse core tensor and the third experiment provides the classification performance of the algorithm for speaker identification in comparison with the TUCKER and the K-SVD [1] algorithms.

5.1. Simulation based on Synthetic Data

In the first experiment, we test the convergence of our proposed method by applying it on a synthetically generated tensor of size $I_1 \times I_2 \times I_3 = 100 \times 100 \times 100$. The tensor is generated from the mode dictionaries of size 100×100 and the sparse core tensor of size $M_n = 1.5I_n (n = 1, 2, 3)$ whose elements are obtained from Gaussian distributions. The sparse core tensor has a fixed mode sparsity of $\mu = s_n/M_n = 1/6$. The value of the step size γ is 0.3. There are two threshold parameters for stopping the algorithm, ϵ_1 for the gradient descent and ϵ_2 for the whole algorithm. In the dictionary update stage, when the error between two consecutive iterations reaches below or equal to ϵ_1, dictionary update stops. In a similar way, the whole algorithm stops when the error between the input tensor and the reconstructed one reaches below or equal to ϵ_2. Typically, ϵ_1 and ϵ_2 are chosen as 10^{-6} and 10^{-4} respectively. The algorithm convergence curve shown in Figure 1 is obtained by averaging the curve over 100 independent trials (experiments).

5.2. Image Reconstruction

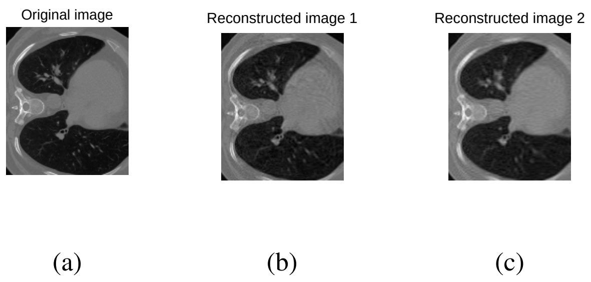

Fig. 2. Comparison between the original image and the reconstructed images using two different sparsity levels of the core tensor. (a) The original Image. (b) The reconstructed image with the core tensor sparsity of 31%. (c) The reconstructed image with the core tensor sparsity of 12%.

For the second experiment, we learn high dimensional signal features for a 3-D human abdomen image of size $I_1 \times I_2 \times I_3$ $151 \times 125 \times 141$ by our proposed algorithm. This dataset is given by [11]. Since we are interested in investigating the effect of the core tensor sparsity on the signal reconstruction, not its dimensions with respect to the input tensor, hence we set $M_n = 1.5I_n$ and the fixed mode sparsity as $\mu = 1/2.2$ and $\mu = 1/3$ respectively, which is equal to 31% and 12% of the total sparsity level (number of non-zeros) of the core tensor, respectively. These sparsity levels compactly represent the input data even though $M_n > I_n$. The 50th slice of the image is reconstructed by the learned dictionaries and the sparse core tensor, as shown in Figure 2, where the original image slice can be compared with the reconstructed slices using the two different levels of sparsity. It can be observed that the reconstructed image using the atoms learned by the proposed sparse tensor learning algorithm resembles the original image very nicely.

5.3. Speaker Identification

To compare the discriminative power of our proposed algorithm, we apply it for the multi-class classification problem of speaker identification and compare its classification performance with that of the TUCKER algorithm [3] and K-SVD algorithm [1]. The signal classification is performed by projecting feature matrix of test signals on to the basis learned by the learning algorithms. The basis in each case of GradTensor as well as the TUCKER algorithm is computed by

$$\mathcal{D} = \underline{\mathrm{X}} \times_1 \mathrm{A} \times_2 B \tag{11}$$

where $\mathcal{D}$ is the learned basis tensor. This basis tensor $\mathcal{D}$ is used to classify the test feature matrix and is determined during the training phase. By following the equation (6), the input signal $\underline{\mathrm{Y}}$ can be represented in terms of learned basis tensor in an unfolded form as

$$Y_{(1)} = I_A D_{(1)} (C \otimes I_B)^T \tag{12}$$

where I_A and I_B are the identity matrices of the same size as A and B. For a 5-class speaker identification problem, the test feature matrix Y^{test} is projected on to each class basis tensor learned in the training phase. The class label of the basis tensor that gives the minimum residual error is the label of the test signal.

For this classification problem, a subset of the TIMIT corpus is selected for speaker identification of 5 speakers with 10 utterances (sentences) per speaker, resulting in a total of 50 utterances. For different numbers of utterances per speaker, we perform classification in such a way that the training and testing examples do not overlap with each other. The class specific basis which acts as the classifier, is learned in the training phase on Linear Predictive Coding (LPC) features of the training signals. The classification performance of the basis learned by three different algorithms (GradTensor, TUCKER and K-SVD) is shown in Figure 3. In case of tensors, $M_n = I_n$ and the fixed mode sparsity in GradTensor is $\mu = s_n/M_n = 1/10$. In the case of K-SVD which learns the dictionary from two dimensional training features, the size of the input training signal, dictionary size and sparsity level are same as those of GradTensor.

This classification example clearly shows the discriminative power of the class basis learned by the GradTensor over those learned by the TUCKER and the K-SVD. This also signifies the importance of learning a sparse core tensor. Since the core tensor in TUCKER decomposition establishes the relationship between decomposition factors along each mode, sparsity constraint on the core tensor applied in the GradTensor learning algorithm clarifies this relationship and reduces the ambiguity in the interpretation of this relationship. The classification results show that the sparsity constraint also helps to maintain the discriminative ability of the learned dictionaries.

6. CONCLUSION

We have designed a tensor dictionary learning algorithm for the TUCKER model that incorporates sparsity constraints on the core tensor. We show the convergence property of the proposed algorithm along with experiments on signal reconstruction and classification. The reconstruction and classification results clearly show the ability of our algorithms for maintaining the discerning features of the signals while retaining the signal reconstruction. In future, we will explore the possibilities for further improving its discriminative ability by incorporating additional constraints to the cost function, such as inter-class correlations, in order to learn class discriminative dictionaries.

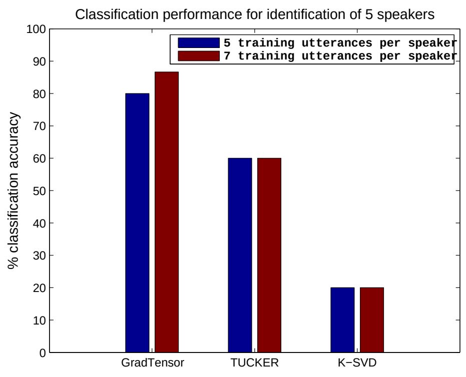

Fig. 3. Classification performances for the identification of 5 speakers for different decomposition algorithms with different number of utterances per speaker.

7. REFERENCES

[1] M. Aharon, M. Elad, and A. Bruckstein, “K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation,” *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[2] R. A. Harshman, “Foundations of the parafac procedure: Models and conditions for an “explanatory” multimodal factor analysis.,” *UCLA working papers in phonetics*, vol. 16, pp. 1–84, 1970.

[3] L. R. Tucker, “Some mathematical notes on 3-mode factor analysis.,” *Psychometrika*, vol. 31, pp. 279–311, 1966.

[4] D. D. Lee and H. S. Seung, “Algorithms for non-negative matrix factorization,” in *Proc. Neural Information Processing Systems*, 2001, pp. 556–562.

[5] T. Hazan, S. Polak, and A. Shashua, “Sparse image coding using a 3d non-negative tensor factorization,” in *Proc. of IEEE International Conference on Computer Vision*, October 2005, vol. 1, pp. 50 – 57.

[6] E. Benetos and C. Kotropoulos, “Non-negative tensor factorization applied to music genre classification,”

IEEE Transactions on Audio, Speech and Signal Processing, vol. 18, no. 8, pp. 1955–1967, November 2010.

[7] Y-D. Kim and S. Choi, “Nonnegative tucker decomposition,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition.*, june 2007, pp. 1 –8.

[8] Y-D. Kim, A. Cichocki, and S. Choi, “Nonnegative tucker decomposition with alpha-divergence,” in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing.*, 2008, pp. 1829 –1832.

[9] M. Mørup, L. K. Hansen, and S. M. Arnfred, “Algorithms for sparse nonnegative tucker decompositions,” *Neural Computing*, vol. 20, no. 8, pp. 2112–2131, 2008.

[10] C. F. Caiafa and A. Cichocki, “Block sparse representation of tensors using kronecker bases.,” in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, 2012, pp. 2709–2712.

[11] J. Vandemeulebroucke, D. Sarrut, and P. Clarysse, “The popi-model, a point-validated pixel-based breathing thorax model,” in *Proc. of the XVth ICCR Conference*, 2007.