Tensor Decomposition for 3D Bars Problem

Jan Platoš, Jana Kočíbová, Pavel Krömer, Pavel Moravec, Václav Snášel

Department of Computer Science, FEECS, VŠB – Technical University of Ostrava, 17. listopadu 15, 708 33 Ostrava-Poruba, Czech Republic {jan.platos, jana.kocibova.st1, pavel.kromer.fei, pavel.moravec, vaclav.snasel}@vsb.cz

Abstract. In this paper, we compare performance of several dimension reduction techniques, namely SVD, NMF and SDD.The qualitative comparison is evaluated on a collection of bars. We compare the quality of these methods from on the base of the visual impact. We also compare dimension reduction techniques SVD and HO-SVD on tensors - 3D bars.

1 Introduction

In order to perform object recognition (no mater which one) it is necessary to learn representations of the underlying characteristic components. Such components correspond to object-parts, or features. These components can occur in different configurations to form many distinct images. Identifying the underlying components which are combined to form images is thus essential for learning the perceptual representations necessary for performing object recognition.

The application area of feature extraction on binary datasets addresses many problem areas, such as association rule mining, itemsets used for market basket analysis, discovery of regulation patterns in DNA microarray experiments, etc. For simplicity sake we used the well-known bars problem (see e.g. [2]), where we try to isolate separate horizontal and vertical bars from images containing their combinations.

In this paper we will concentrate on the last category – other feature extraction methods which use known dimension reduction techniques and clustering for automatic feature extraction.

In this paper we will use the bars collection as a benchmark collection. The bars problem (and its variations) is a benchmark task for the learning of independent image features (Föildiák [2]; Spratling [6];). In the standard version of the bars problem, as defined by Föildiák [2], training data consists of 8 by 8 pixel images in which each of the 16 possible (one-pixel wide) horizontal and vertical bars can be present with a probability of $\frac{1}{8}$. Typical examples of training images are shown in Figure 1.

One of the well-known methods of feature extraction is the *singular value decomposition* (*SVD*) which was already successfully used for automatic feature extraction.

We extended the bars problem to 3 dimensions, using planes instead of lines. The input cube contains several planes, which may or may not be parallel to x, y and z axes.

The straightforward approach to image indexing is to transform the 2D images into a single vector. This is often done by concatenating all the rows of an image into a single image vector [7] (although a more sophisticated method can be used). We will use similar approach for 3D bars and classic SVD, combining two dimensions into one,

V. Snášel, K. Richta, J. Pokorný (Eds.): Dateso 2008, pp. 49–60, ISBN 978-80-248-1746-0.

so that we can compare the original and reconstructed matrices based on the visual impact and Frobenius norm.

The rest of this paper is organized as follows. The second section explains dimension reduction methods, which were used for classic 2D bars problem. The third section mentions CubeSVD, which was originaly used for the 3D bars problem. Then in the fourth section we describe experimental results and finally in the section five we made some conclusions.

2 Dimension Reduction

We used four promising methods of dimension reduction for our comparison – *Singular Value Decomposition* (*SVD*), *Semi-Discrete Decomposition* (*SDD*) and *Non-negative Matrix Factorization* (*NMF*). All of them are briefly described bellow.

2.1 Singular Value Decomposition

SVD [1] is an algebraic extension of classical vector model. It is similar to the PCA method, which was originally used for the generation of eigenfaces in image retrieval. Informally, SVD discovers significant properties and represents the images as linear combinations of the base vectors. Moreover, the base vectors are ordered according to their significance for the reconstructed image, which allows us to consider only the first k base vectors as important (the remaining ones are interpreted as "noise" and discarded). Furthermore, SVD is often referred to as more successful in recall when compared to querying whole image vectors [1].

Formally, we decompose the matrix of images A by *singular value decomposition* (*SVD*), calculating singular values and singular vectors of A.

We have matrix A, which is an $n \times m$ rank-r matrix (where $m \geq n$ without loss of generality) and values $\sigma_1, \ldots, \sigma_r$ are calculated from eigenvalues of matrix AA^T as $\sigma_i = \sqrt{\lambda_i}$. Based on them, we can calculate column-orthonormal matrices $U = (u_1, \ldots, u_n)$ and $V = (v_1, \ldots, v_n)$, where $U^T U = I_n$ a $V^T V = I_m$, and a diagonal matrix $\Sigma = diag(\sigma_1, \ldots, \sigma_n)$, where $\sigma_i > 0$ for $i \leq r, \sigma_i \geq \sigma_{i+1}$ and $\sigma_{r+1} = \cdots = \sigma_n = 0$.
The decomposition

$$A = U\Sigma V^T \tag{1}$$

is called *singular decomposition* of matrix A and the numbers $\sigma_1, \ldots, \sigma_r$ are *singular values* of the matrix A. Columns of U (or V) are called *left* (or *right*) singular vectors of matrix A.

Now we have a decomposition of the original matrix of images A. We get r nonzero singular numbers, where r is the rank of the original matrix A. Because the singular values usually fall quickly, we can take only k greatest singular values with the corresponding singular vector coordinates and create a *k-reduced singular decomposition* of A.

Let us have k $(0 < k < r)$ and singular value decomposition of A

$$A = U\Sigma V^T \approx A_k = (U_k U_0) \begin{pmatrix} \Sigma_k & 0 \\ 0 & \Sigma_0 \end{pmatrix} \begin{pmatrix} V_k^T \\ V_0^T \end{pmatrix} \tag{2}$$

We call $A_k = U_k \Sigma_k V_k^T$ a k-reduced singular value decomposition (rank-k SVD). Instead of the A_k matrix, a matrix of image vectors in reduced space $D_k = \Sigma_k V_k^T$ is used in SVD as the representation of image collection. The image vectors (columns in D_k) are now represented as points in k-dimensional space (the *feature-space*). represent the matrices U_k, Σ_k, V_k^T.

2.2 Semi-discrete Decomposition

The *SDD* is one of other LSI methods, proposed recently for text retrieval in [3]. As mentioned earlier, the rank-k SVD method (called *truncated SVD* by authors of semi-discrete decomposition) produces dense matrices U and V, so the resulting required storage may be even larger than the one needed by the original term-by-document matrix A.

To improve the required storage size and query time, the semi-discrete decomposition was defined as

$$A \approx A_k = X_k D_k Y_k^T, \tag{3}$$

where each coordinate of the matrices X_k and Y_k is constrained to have entries from the set $\varphi = \{-1, 0, 1\}$, and the matrix D_k is a diagonal matrix with positive coordinates.

The SDD does not reproduce A exactly, even if $k = n$, but it uses very little storage with respect to the observed accuracy of the approximation. A rank-k SDD (although from mathematical standpoint it is a sum on rank-1 matrices) requires the storage of $k(m + n)$ values from the set $\{-1, 0, 1\}$ and k scalars. The scalars need to be only single precision because the algorithm is self-correcting. The SDD approximation is formed iteratively.

The optimal choice of the triplets (x_i, d_i, y_i) for given k can be determined using greedy algorithm, based on the residual $R_k = A - A_{k-1}$ (where A_0 is a zero matrix).

2.3 Non-negative Matrix Factorization

The *NMF* [5] method calculates an approximation of the matrix A as a product of two matrices, W and H. The matrices are usually pre-filled with random values (or H is initialized to zero and W is randomly generated). During the calculation the values in W and H stay positive. The approximation of matrix A, matrix A_k, can be calculated as $A_k = WH$.

The original *NMF* method tries to minimize the Frobenius norm of the difference between A and A'_k using $\min\limits_{W,H} ||V - WH||_F^2$ as the criterion in the minimization problem.

Recently, a new method was proposed in [6], where the constrained least squares problem $\min\limits_{H_j}\{||V_j - WH_j||^2 - \lambda||H_j||_2^2\}$ is the criterion in the minimization problem. This approach is yields better results for sparse matrices.

Unlike in *SVD*, the base vectors are not ordered from the most general one and we have to calculate the decomposition for each value of k separately.

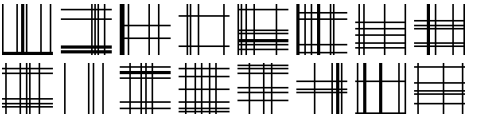

Fig. 1. Some more complex 2D bars

3 3D Theory

A tensor is a higher order generalization of a vector. Vector is a first order tensor and a matrix is a second order tensor. The order of a tensor $\mathcal{A} \in \mathcal{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is N. Elements of $\mathcal{A}$ is denoted as $a_{i_1 \ldots i_n \ldots i_N}$ where $1 \leq i_n \leq I_N$. Two basic operations are for calculation of CubeSVD: the unfolding of a tensor $\mathcal{A}\left(\mathcal{A}_{(n)}\right)$ and the mode-n product of a tensor $\mathcal{A}$ and matrix $\mathcal{M}\left(\mathcal{A} \times_{(n)} \mathcal{M}\right)$.

The operation unfolding unfolds the tensor $\mathcal{A}$ into matrix $\mathcal{A}_{(n)}$ along order N. Each column of tensor $\mathcal{A}_{(n)}$ is composed of $a_{i_1 \ldots i_n \ldots i_N}$ where i_n varies and the order indices are fixed. The operations unfolding are illustrated in Figure 2 for third order tensor. See [4] for details on operation unfolding of a tensor $\mathcal{A}$.

The n-mode product of a tensor $\mathcal{A} \in R^{I_1 \times I_2 \times \cdots \times I_N}$ by a matrix $M \in R^{J_n \times I_n}$ is an $I_1 \times I_2 \times \cdots \times I_{n-1} \times J_n \times I_{n+1} \times \cdots \times I_N$-tensor of which the entries are given by

$$(\mathcal{A} \times_n M)_{i_1 \cdots i_{n-1} j_n i_{n+1} \cdots i_N} = \sum_{i_n} a_{i_1 \cdots i_{n-1} i_n i_{n+1} \cdots i_N} m_{j_n i_n}$$

See [4] for details on operation mode-n product of a tensor $\mathcal{A}$ and matrix $\mathcal{M}$.

Matrix SVD can be rewritten as $A = \Sigma \times_1 V^{(1)} \times_2 V^{(2)}$ in terms of n-mode products. CubeSVD is a generalization of SVD and was described in [4]. Tensor $\mathcal{A}$ can be written as the n-mode product [4]

$$\mathcal{A} = \mathcal{S} \times_1 \mathcal{V}_1 \times_2 \mathcal{V}_2 \times_3 \cdots \times_N \mathcal{V}_N$$

as illustrated in Figure 3 for $N = 3$.

$\mathcal{S}$ is called core tensor. S is in general a full tensor, instead of being pseudodiagonal (this would mean that nonzero elements could only occur when the indices $i_1 = i_2 = \cdots = i_N$). $\mathcal{S}$ has the property of all-orthogonality [4]. $\mathcal{V}_\backslash$ contains the orthonormal vectors. They called n-mode singular vectors. The Frobenius-norms $\|\mathcal{S}_{in=i}\|$ are n-mode singular values of A. Their order is

$$\|\mathcal{S}_{in=1}\| \geq \|\mathcal{S}_{in=2}\| \geq \cdots \geq \|\mathcal{S}_{in=In}\| \geq 0$$

4 Experimental Results - 2D Bars

For testing of above mentioned methods, we used generic collection of 1600 32×32 black-and-white images containing different combinations of horizontal and vertical lines (bars). The probabilities of bars to occur in images were the same and equal to

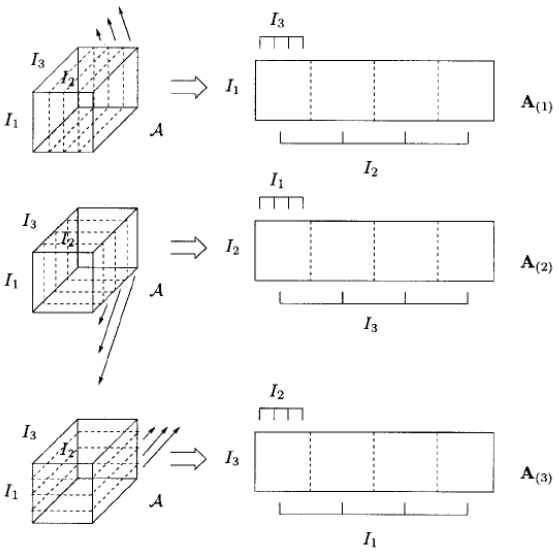

Fig. 2. Unfolding of third order of a tensor $\mathcal{A}$

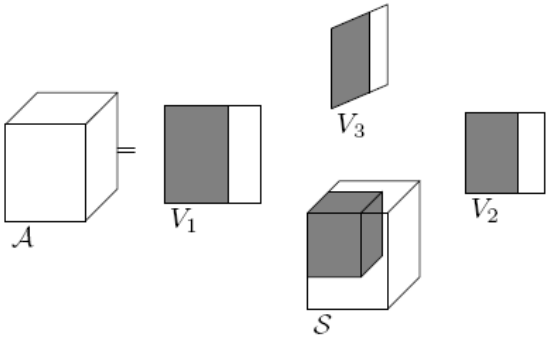

Fig. 3. Order-3 Singular Value Decomposition $\mathcal{A}$

$10/64$, i.e. images contain 10 bars in average. An example of several images from generated collection is shown in Figure 4.

Many of tested methods were able to generate a set of base images or factors, which should ideally record all possible bar positions. However, not all methods were truly successful in this.

With SVD, we obtain classic singular vectors, the most general being among the first. The first few are shown in Figure 5. We can se, that the bars are not separated and different shades of gray appear.

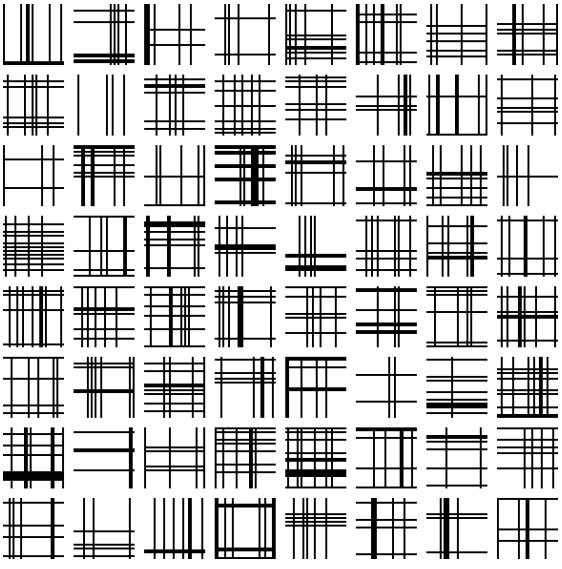

Fig. 4. Some generated images from bars collection

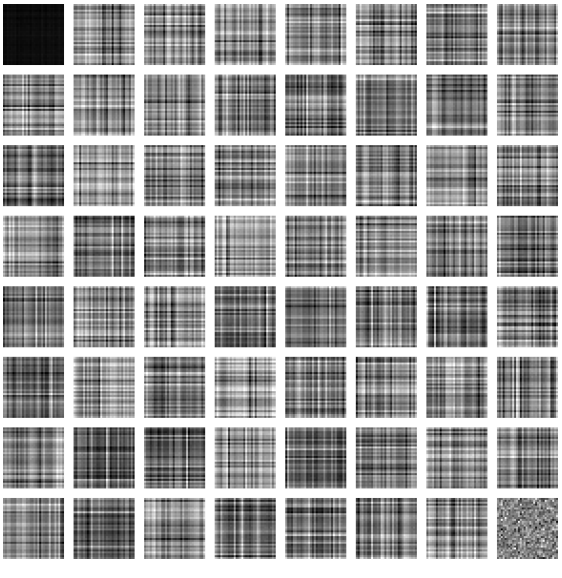

Fig. 5. First 64 base images – SVD method

The NMF methods yield different results. The original NMF method, based on the adjustment of random matrices W and H provides hardly-recognizable images even for

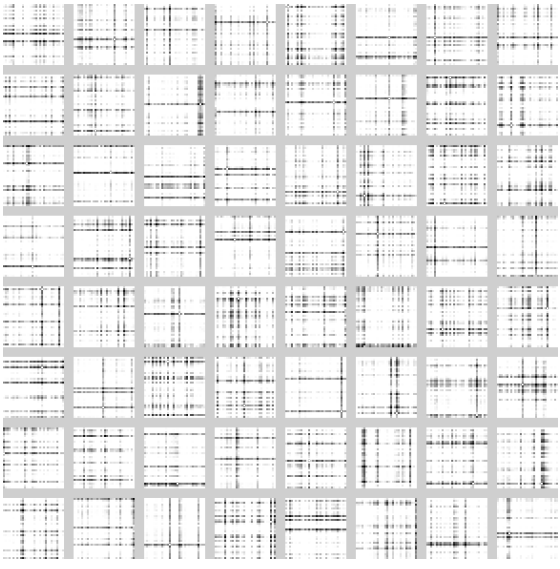

Fig. 6. First 64 factors – original NMF method

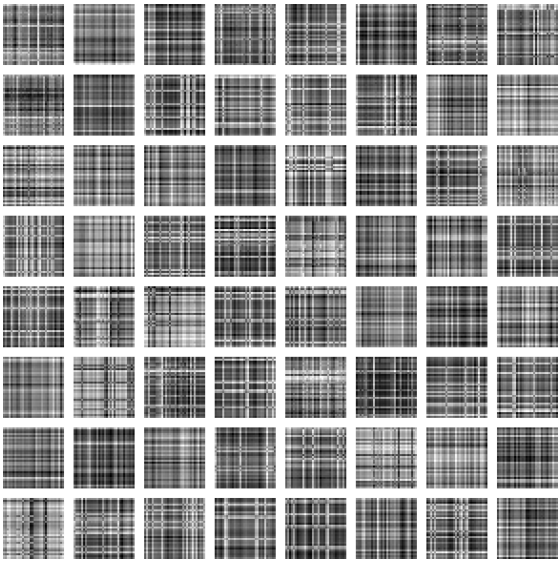

Fig. 7. First 64 factors for GD-CLS NMF method (0.01)

$k = 100$ and 1000 iterations (we used 100 iterations for other experiments). Moreover, these base images still contain significant salt and pepper noise and have a bad contrast.

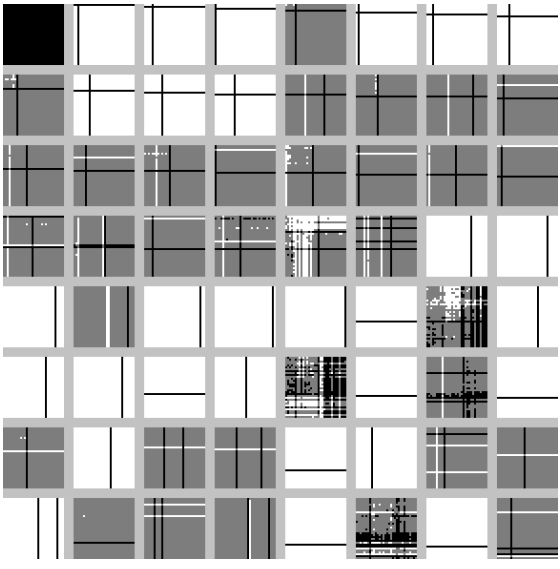

Fig. 8. First 64 base vectors – SDD method

The factors are shown in Figure 6. We must also note, that the NMF decomposition will yield slightly different results each time it is run, because the matrix(es) are pre-filled with random values.

The SDD method differs slightly from previous methods, since each factor contains only values $\{-1, 0, 1\}$. Gray in the factors shown in Figure 8 represents 0; -1 and 1 are represented with black and white respectively.

The base vectors in Figure 8 can be divided into three categories:

1. Base vectors containing only one bar.
2. Base vectors containing one horizontal and one vertical bar.
3. Other base vectors, containing several bars and in some cases even noise.

5 Experimental result - 3D Bars

For testing of CubeSVD method, we use several collections of $8 \times 8 \times 8$ 3-dimensional cubes. We create 2 types of test collections. The first type contains 2 collections with 15 cubes each which were used for local feature extraction (each cube was decomposed separately). The first collection contains cubes which are crossed by 2 perpendicular planes (Figure 9a). The second collection contains cubes crossed by 3 planes - 2 perpendicular and 1 skewed (Figure 9b). The resulting number of singular values was between 1 and 8 (full CubeSVD).

The second type contains 4 collections with 1000 cubes each which were used for collection-based feature extraction. The first collection contains cubes with one skewed

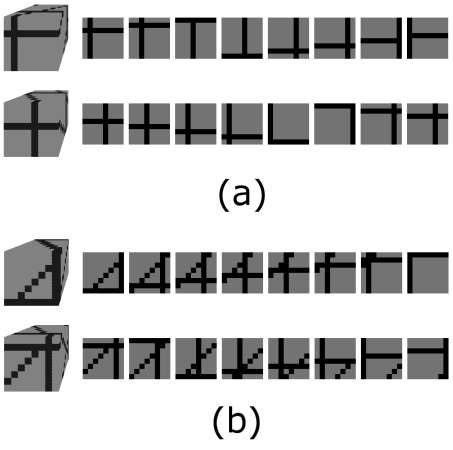

Fig. 9. Collections for local features extraction. (a) is example from first collection, (b) is example from second collection

plane, second collection contains cubes with 2 skewed planes and so on. Example cubes for each collection is depicted in Figure 10.

As a measure for comparing similarity between original and reduced cubes we used the Frobenius norm (without calculating the square root)

$$F^2(O, R) = \sum_i \sum_j \sum_k (O[i, j, k] - R[i, j, k])^2$$

5.1 Local feature extraction

In the first experiment we applied CubeSVD and SVD algorithm on the first two collections for extracting local features. Results for the first collection with 2 perpendicular planes are depicted in Figure 11 for 6 singular values and are depicted on Figure 12 for 2 singular values. Values of Frobenius norm are shown in tables 1 and 2.

It can be seen CubeSVD extracts all of the original bars in Figure 11, while the SVD with the same rank ignored one of the bars, while reconstructing the other two more sharply. This is even more visible in Figure 12, where all bars in CubeSVD are slightly recorded, but classic SVD reconstructs one of the bars perfectly, adding noise to other areas. We see, that this behavior leads to lower Frobenius norm in Tables 1 and 2 for th classic SVD, which satisfies the condition that the value for SVD should be minimal for given rank k.

6 Conclusion

Since the CubeSVD method provided only one singular value for planes parallel to the axes, which was to be expected, the experiments were done on planes both perpendicular and skewed.

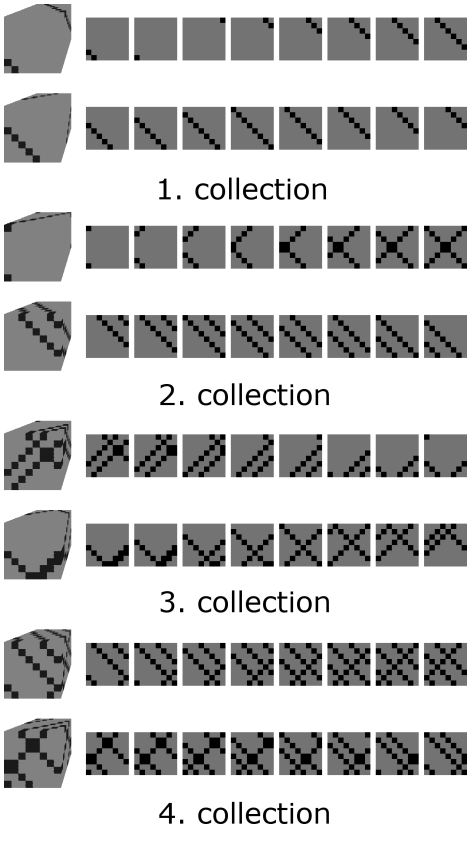

Fig. 10. Collections for collection-based features extraction

It seems, that the original SVD performed better than CubeSVD, based on the Frobenius norm, but the visual inspection of reduced tensors shows the reason – whilst the horizontal bars were reconstructed nearly perfectly, the vertical ones deteriorated more quickly. On the other hand, the CubeSVD tried to minimize the overall error.

We are currently studying the collection-based feature extraction of 3D bars, where the number of singular values ranges from 1 to 512 for $8 \times 8 \times 8$ cubes, compared to 8 singular values for high-order SVD. The classic SVD results for 1 to 8 singular values mentioned in previous section are not satisfactory.

We are currently extending our CubeSVD implementation to support tensors of 4 and mode dimensions and preparing to test the High-order SDD and NMF methods against their 2D counterparts.

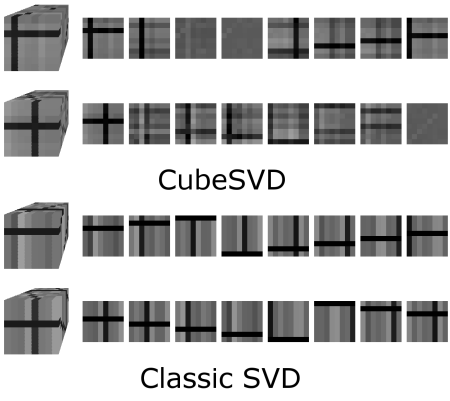

Fig. 11. Results for first collection and 6 singular values

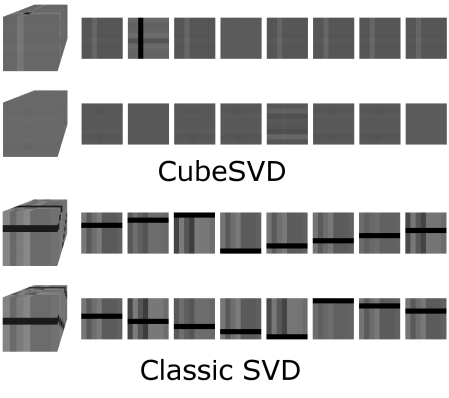

Fig. 12. Results for first collection and 2 singular values

References

1. M. Berry, S. Dumais, and T. Letsche. Computational Methods for Intelligent Information Access. In *Proceedings of the 1995 ACM/IEEE Supercomputing Conference*, San Diego, California, USA, 1995.
2. P. Földiák. Forming sparse representations by local anti-Hebbian learning. *Biological cybernetics, 64:22*, pages 165–170, 1990.
3. T. G. Kolda and D. P. O'Leary. Computation and uses of the semidiscrete matrix decomposition. In *ACM Transactions on Information Processing*, 2000.
4. L. D. Lathauwer, B. D. Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM J. Matrix Anal. Appl.*, 21(4):1253–1278, 2000.
5. F. Shahnaz, M. Berry, P. Pauca, and R. Plemmons. Document clustering using nonnegative matrix factorization. *Journal on Information Processing and Management*, 42:373–386, 2006.

Table 1. Cumulative Frobenius norm for first collection

Method	CubeSVD	SVD
K=1	1484.31	735
K=2	1356.21	630
K=3	1304.02	525
K=4	1169.81	420
K=5	976.62	315
K=6	708.54	210
K=7	379.29	105
K=8	0	0

Table 2. Cumulative Frobenius norm for second collection

Method	CubeSVD	SVD
K=1	1768.84	1045.91
K=2	1587.14	854.01
K=3	1489.88	677.87
K=4	1340.44	514.32
K=5	1091.12	364.01
K=6	813.81	228.41
K=7	438.31	102.49
K=8	0	0

6. M. W. Spratling. Learning Image Components for Object Recognition. *Journal of Machine Learning Research*, 7:793–815, 2006.
7. M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.