

A Fast Algorithm for Nonnegative Tensor Factorization using Block Coordinate Descent and an Active-set-type method

Krishnakumar Balasubramanian* Jingu Kim* Andrey Purotskiy[†] Michael W. Berry[†]
Haesun Park*

Abstract

Nonnegative factorization of tensors plays an important role in the analysis of multi-dimensional data in which each element is inherently nonnegative. It provides a meaningful lower rank approximation, which can further be used for dimensionality reduction, data compression, text mining, or visualization. In this paper, we propose a fast algorithm for nonnegative tensor factorization (NTF) based on the alternating nonnegative least squares framework. To efficiently solve the nonnegativity-constrained least squares problems, we adopted a block principal pivoting algorithm that included additional improvements exploiting the common characteristics of NTF computation. The effectiveness and efficiency of the proposed algorithm are demonstrated through experiments using various real-world and synthetic data sets. We observed that the proposed algorithm significantly outperformed existing ones in computational speed.

1 Introduction

Tensors, which refers to multi-dimensional arrays, provide a mathematical and algorithmic framework for analyzing multi-scale, multi-dimensional data and extracting meaningful information from them. Tensor factorizations [8] are multi-linear generalizations of matrix factorizations and are powerful tools for analyzing multi-dimensional data. Tensor modeling and tensor factorizations enable the analysis of complex data objects, which are not well understood by conventional matrix-based methods. Numerous data sets from various areas have been studied using tensor factorizations [14].

Recently, nonnegative tensor factorization (NTF) [21, 27] has attracted growing interest since it provides interpretable factorizations [22] when data components are inherently nonnegative. For example, pixels in digital im-

ages, chemical concentrations in chemometrics, high dimensional data in Internet traffic or large-scale social networks [24] are naturally represented by nonnegative numbers. Once nonnegative factors are obtained, each object can be understood as an additive linear combination of intrinsic parts of the data. This is one of the most important properties of nonnegative matrix factorization (NMF) [16, 17]. When we deal with tensors, a simple extension of the singular value decomposition (SVD) to a higher-order case [15] can be used, but it might fail to factorize in a meaningful way since it does not consider the intrinsic nonnegativity in the data. Instead, NTF has been successfully used to analyze such data sets. An interesting connection between NTF and latent class models in statistics has been also shown [21].

While NTF provides several advantages, modern data sets which tend to be extremely large make the problem computationally challenging. Previous approaches for computing NTF include the following. Andersson and Bro proposed an NTF algorithm based on alternating least squares framework [1], and Kim et. al. proposed an algorithm based on the active set method for the nonnegativity constrained least squares problems [11]. Welling and Webber proposed an algorithm based on multiplicative updates [27]. Friedlander and Hatz proposed an algorithm based on solving bound constrained linear least-squares problem [7]. Despite the presence of such algorithms, there exists need for developing computational faster algorithms, especially for analyzing large data sets.

In this paper, we propose a fast algorithm for NTF based on the alternating nonnegativity constrained least squares (ANLS) framework [10, 11], in which a series of nonnegativity constrained least squares (NNLS) problems are solved in each iteration. We propose to solve the NNLS problems by a fast active-set-type algorithm, called the block principal pivoting method [20, 12], which overcomes some limitations of the standard active set method. This algorithm is carefully tuned for the special characteristics of the NTF computation and is well suited for large scale data. We also extend our NTF algorithm to other NTF formulations such as regularized NTF and sparse NTF. Comparisons of algorithms using various data sets

*School of Computational Science and Engineering, Georgia Institute of Technology. {bkkumar,jingu,hpark}@cc.gatech.edu. The work of these authors was supported in part by the National Science Foundation grants CCF-0728812, CCF-0732318, and CCF-0956517. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

[†]Department of EECS, University of Tennessee. {purotski,berry}@eecs.utk.edu

show that the proposed new algorithm outperforms existing ones in terms of computational speed and approximation accuracy.

The rest of this paper is organized as follows. In Section 2, we give a brief overview of tensors and nonnegative tensor factorization. In Section 3, our computationally faster algorithm for NTF is described in detail and its extensions are presented in Section 4. In Section 5, we present extensive experimental results demonstrating the speed-up that our algorithm provides and various application results demonstrating the wide applicability of our algorithm.

2 Tensors and Tensor Factorizations

In this section, we begin with examples of tensor modeling to motivate tensor methods for data analysis. We then introduce basic mathematical notations and operations used for tensors, and we describe the definition of the tensor factorization studied in this paper.

2.1 Modeling with tensors Multi-linear algebra, the algebra of tensors, provides a flexible framework for modeling and analyzing multi-dimensional data. For example, a higher order extension of well-known Eigenfaces approach, called *Tensorfaces* [26], has been proposed for modeling faces for recognition. In Tensorfaces, a set of K images with each image of pixel size $N \times N$ is modeled as a $N \times N \times K$ tensor, rather than $N^2 \times K$ matrix, and a tensor extension of SVD is calculated for finding the low rank approximation of the data. Tensorfaces provide several advantages over the conventional Eigenfaces method. In web-link analysis, an extension of the popular HITS model was proposed [13] using tensors. The tensor model used here incorporates anchor text information in the adjacency matrix. Given a set of I web sites and K keywords, a $I \times I \times K$ tensor is constructed such that x_{ijk} is 1 if page i points to page j using term k , and 0 otherwise. Several semantic information may be incorporated by increasing the number of dimensions to more than 3. In both of the examples, the entries of the tensor are nonnegative and incorporating that constraint for factorization will provide a more meaningful lower-rank approximation of data.

Sun et. al., [24] used Tucker decomposition, another type of tensor factorization, for analysis and visualization of social networks. A tensor is formed by taking into account both the network and content aspect of the social network. Then, clustering is performed on the factors of the decomposed tensor to find out which topics people discuss, who are the experts in a given topic, etc. They also provide an efficient method for hierarchical context-specific visualization of such social network based on tensor decomposition. They demonstrate efficiency of

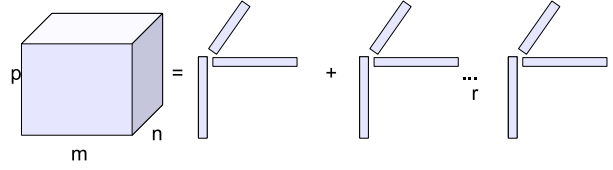


Figure 1: 3-way PARAFAC model: The tensor is represented as a linear combination of r rank-1 tensors. This will provide a rank- r approximation to the original tensor.

their approach through experiments on social networks.

In this paper, we propose a fast algorithm for computing nonnegative tensor factorization, which can be subsequently used in many applications. With this motivation, we give a brief overview of tensors and the notations used in this paper. See [14, 6] for a comprehensive overview of tensors and various tensor factorizations.

2.2 Notations and operations A tensor is a multi-dimensional array $\mathbf{X} \in \mathbb{R}^{m_1 \times m_2 \times \dots \times m_N}$. The order of a tensor is the number of dimensions, also known as way or mode. Mode- n fiber of a given tensor is obtained by fixing every index except the n^{th} index.

DEFINITION 2.1. An N -way tensor $\mathbf{X} \in \mathbb{R}^{m_1 \times m_2 \times \dots \times m_N}$ is called a rank-1 tensor if it can be written as an outer product of N vectors a_1, \dots, a_N .

$$\mathbf{X} = a_1 \circ a_2 \circ a_3 \cdots \circ a_N$$

where \circ represents the vector outer product.

DEFINITION 2.2. The Kronecker product of two matrices $A \in \mathbb{R}^{I \times J}$ and $B \in \mathbb{R}^{K \times L}$ is given by:

$$A \otimes B_{(IK) \times (JL)} = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1J}B \\ a_{21}B & a_{22}B & \cdots & a_{2J}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}B & a_{I2}B & \cdots & a_{IJ}B \end{pmatrix}$$

DEFINITION 2.3. The Khatri-Rao product of two matrices $A \in \mathbb{R}^{I \times J}$ and $B \in \mathbb{R}^{K \times J}$ is given by:

$$A \odot B_{(IK) \times (J)} = [a_1 \otimes b_1 \quad a_2 \otimes b_2 \quad \cdots \quad a_J \otimes b_J]$$

The process of flattening or unfolding a tensor is the reordering of the elements of the tensor in the form of a matrix. The mode- n unfolding of a tensor $\mathbf{X} \in \mathbb{R}_+^{m_1 \times m_2 \times \dots \times m_N}$, denoted by $\mathbf{X}_{(n)}$, is obtained by arranging the mode- n fibers to be the columns of the resulting matrix. For example, consider a $3 \times 3 \times 2$ tensor $\mathbf{X} = (a_{ijk}) \in \mathbb{R}_+^{3 \times 3 \times 2}$. The mode-1, 2, and 3 unfolded matrices will be, respectively,

$$X_{(1)}^T = \begin{pmatrix} a_{1,1,1} & a_{1,2,1} & a_{1,3,1} & a_{1,1,2} & a_{1,2,2} & a_{1,3,2} \\ a_{2,1,1} & a_{2,2,1} & a_{2,3,1} & a_{2,1,2} & a_{2,2,2} & a_{2,3,2} \\ a_{3,1,1} & a_{3,2,1} & a_{3,3,1} & a_{3,1,2} & a_{3,2,2} & a_{3,3,2} \end{pmatrix}$$

$$X_{(2)}^T = \begin{pmatrix} a_{1,1,1} & a_{2,1,1} & a_{3,1,1} & a_{1,1,2} & a_{2,1,2} & a_{3,1,2} \\ a_{1,2,1} & a_{2,2,1} & a_{3,2,1} & a_{1,2,2} & a_{2,2,2} & a_{3,2,2} \\ a_{1,3,1} & a_{2,3,1} & a_{3,3,1} & a_{1,3,2} & a_{2,3,2} & a_{3,3,2} \end{pmatrix}$$

$$X_{(3)} = \begin{pmatrix} a_{1,1,1} & a_{1,1,2} \\ a_{1,2,1} & a_{1,2,2} \\ a_{1,3,1} & a_{1,3,2} \\ a_{2,1,1} & a_{2,1,2} \\ a_{2,2,1} & a_{2,2,2} \\ a_{2,3,1} & a_{2,3,2} \\ a_{3,1,1} & a_{3,1,2} \\ a_{3,2,1} & a_{3,2,2} \\ a_{3,3,1} & a_{3,3,2} \end{pmatrix}.$$

2.3 Nonnegative tensor factorization To discuss lower-rank factorization of tensors, one must define the rank of a tensor.

DEFINITION 2.4. *The rank of a tensor \mathbf{X} is defined as the minimum number of rank-one tensors needed, so that it can be represented in a polyadic form of those tensors.*

A decomposition of a given tensor as a sum of rank-one tensors is called as PARAFAC decomposition. Even though the definition of tensor rank is similar to matrix rank, its properties are rather different. In fact, determining the rank of a given tensor is NP-complete [9]. An exact PARAFAC decomposition with $r = \text{rank}(\mathbf{X})$ components is called the rank decomposition of a tensor. An important property of tensor factorization is that the rank decompositions are unique up to elementary indeterminacies of scaling and permutation. It was claimed that the problem of computing PARAFAC decomposition of a tensor is well-posed for the case when it has nonnegative constraints [23]. For the rest of the paper, we focus on nonnegative PARAFAC decomposition which we refer to as NTF.

Given an N -way nonnegative tensor, $\mathbf{X} \in \mathbb{R}_+^{m_1 \times m_2 \times \dots \times m_N}$, where \mathbb{R}_+ denotes the positive quadrant, we need to decompose it into a set of loading matrices, $\{A_1, A_2, \dots, A_N\}$, where $A_k \in \mathbb{R}_+^{m_k \times r}$ for $1 \leq k \leq N$ and r is some positive integer¹. For simplicity, we consider a three-way nonnegative tensor. Extension to higher order tensors is analogous.

DEFINITION 2.5. *Given a tensor $\mathbf{X} = (x_{ijk}) \in$*

$\mathbb{R}_+^{m \times n \times p}$, its NTF model with rank r is defined as

$$(2.1) \quad x_{ijz} = \sum_{q=1}^r a_{iq} b_{jq} c_{zq} + e_{ijz}$$

where $A = a_{iq} \in \mathbb{R}_+^{m \times r}$, $B = b_{jq} \in \mathbb{R}_+^{n \times r}$, $C = c_{zq} \in \mathbb{R}_+^{p \times r}$, are nonnegative loading matrices and $\mathbf{E} = (e_{ijz})$ is the approximation error tensor.

3 NTF using ANLS framework and Block Principal Pivoting

3.1 ANLS framework One of the problems with tensor decomposition is that there does not exist an algorithm for finding the exact number of components in the decomposition of a tensor [18]. An alternative way to proceed is to seek decompositions with multiple components and choose the best according to some criteria. For example, the loading matrices can be found by solving the following optimization problem

$$(3.2) \quad \min_{A, B, C \geq 0} \|\mathbf{X} - \llbracket ABC \rrbracket\|_F^2,$$

where $\llbracket ABC \rrbracket = \sum_{q=1}^r a_q \circ b_q \circ c_q$ and is called a *Kruskal operator*.

This nonlinear optimization problem can be solved by using the alternating nonnegative least squares (ANLS) framework. For simplicity, we describe the framework for a three-way PARAFAC model with nonnegativity constraints although the framework can be extended to higher order models. In the case of a nonnegative tensor $\mathbf{X} \in \mathbb{R}_+^{m \times n \times p}$, we want to identify three nonnegative factors or loading matrices $A \in \mathbb{R}_+^{m \times r}$, $B \in \mathbb{R}_+^{n \times r}$, and $C \in \mathbb{R}_+^{p \times r}$.

We first initialize two matrices, say B and C , and iterate solving the following nonnegativity constrained least squares (NNLS) problems until a stopping criteria is satisfied:

$$(3.3) \quad \min_{A \geq 0} \|Y_{BC} A^T - X_{(1)}\|_F^2$$

$$(3.4) \quad \min_{B \geq 0} \|Y_{AC} B^T - X_{(2)}\|_F^2$$

$$(3.5) \quad \min_{C \geq 0} \|Y_{AB} C^T - X_{(3)}\|_F^2$$

where $Y_{BC} = B \odot C$ and $X_{(1)}$ is the $(np) \times m$ unfolded matrix; $Y_{AC} = A \odot C$ and $X_{(2)}$ is the $(mp) \times n$ unfolded matrix, and $Y_{AB} = A \odot B$ and $X_{(3)}$ is the $(mn) \times p$ unfolded matrix.

This ANLS formulation has the property that for any $N > 2$, if each of the sub problems have unique solution, then the limit point of the sequence is a stationary point [4]. For $N = 2$, any limit point of the sequence is

¹Various bounds exist in the literature for rank of a tensor. See, for example, [14] for further discussion on the rank of tensors.

a stationary point [19]. The unfolding operation is critical in the sense that it turns the original problem into a sequence of NNLS problems. Another important thing to note is the structure of the NNLS problems that are obtained by the unfolding operation. Typically, r is small for a low rank approximation, and hence the matrices formed by the Khatri-rao product is long and thin. This observation is important in designing efficient algorithms for solving the NNLS problems. In the following section, we explain how we efficiently solve the NNLS problems.

Algorithm 1 NTF/ANLS

- Given a tensor $\mathbf{X} \in \mathbb{R}_+^{m_1 \times m_2 \times \dots \times m_N}$ and rank r , initialize the $N - 1$ of the loading matrices, say $A_2 \in \mathbb{R}_+^{m_2 \times r} \dots A_N \in \mathbb{R}_+^{m_N \times r}$ with nonnegative values.
- Repeat solving the following NNLS problems using the block principal pivoting method until a stopping criterion is satisfied:

$$\begin{aligned} & \min_{A_1 \geq 0} \|Y_{A_2 \dots A_N} A_1^T - X_{(1)}\|_F^2 \\ & \min_{A_2 \geq 0} \|Y_{A_1 A_3 \dots A_N} A_2^T - X_{(2)}\|_F^2 \\ & \quad \vdots \\ & \min_{A_N \geq 0} \|Y_{A_1 \dots A_{N-1}} A_N^T - X_{(N)}\|_F^2 \end{aligned}$$

3.2 Block Principal Pivoting Algorithm The computational task for NTF is now narrowed down to the NNLS problems in Eqs. (3.3)-(3.5). For the moment, suppose we want to solve a NNLS problem given as

$$(3.6) \quad \min_{X \geq 0} \|DX - E\|_F^2,$$

where $D \in \mathbb{R}^{p \times q}$, $E \in \mathbb{R}^{p \times l}$, and $X \in \mathbb{R}^{q \times l}$. Note that one can solve Eq. (3.6) by naively solving NNLS problems for each right-hand side vector, which appears as

$$(3.7) \quad \min_{x \geq 0} \|Dx - e\|_2^2.$$

Although this approach is possible, we will see that there exist efficient ways to accelerate the multiple right-hand side case.

Algorithms for solving Eq. (3.6) or (3.7) have been studied in a number of papers [5, 3, 12]. For each case of applications, the algorithm of choice depends on the

size and the structure of NNLS problems. Note that in the case of NTF, because D is typically long and thin after the unfolding operation, each column of X is rather short. In fact, the size of each column vector is the target lower dimension, r . Hence, active-set-type methods are expected to perform well compared to iterative optimization schemes. We adopted the modified active-set-type algorithm, called the block principal pivoting method, studied by Kim and Park [12]. They efficiently extended the block principal pivoting algorithm in [20] for the multiple right-hand side case. Here we briefly summarize the key ideas of [20] and [12].

The motivation of block principal pivoting methods [20] comes from the difficulty of conventional active set algorithms which occur when the number of variables increases. In active set algorithms, because typically only one variable is exchanged per iteration between the active and passive sets, the number of iterations heavily depend on number of variables. To accelerate computation, an algorithm whose iteration count does not depend on the number of variables is desirable. The block principal pivoting methods manage to do so by exchanging multiple variables at a time. To describe the method, let us consider the NNLS problem with a single right-hand side in Eq. (3.7). The Karush-Kuhn-Tucker (KKT) optimality conditions for Eq. (3.7) are given as

$$(3.8a) \quad y = D^T D x - D^T e,$$

$$(3.8b) \quad y \geq 0, x \geq 0,$$

$$(3.8c) \quad x_i y_i = 0, i = 1, \dots, q.$$

We assume that the matrix D has full column rank. In this case, a solution x that satisfies the conditions in Eqs. (3.8) is the optimal solution of Eq. (3.7).

We divide the index set $\{1, \dots, q\}$ into two sub-groups F and G where $F \cup G = \{1, \dots, q\}$ and $F \cap G = \emptyset$. Let x_F, x_G, y_F , and y_G denote the subsets of variables with corresponding indices, and let D_F and D_G denote the submatrices of D with corresponding column indices. Initially, we assign $x_G = 0$ and $y_F = 0$. Then, by construction, $x = (x_F, x_G)$ and $y = (y_F, y_G)$ always satisfy Eq. (3.8c) for any x_F and y_G . Now, we compute x_F and y_G using Eq. (3.8a) and check whether the computed values of x_F and y_G satisfy Eq. (3.8b). Computation of x_F and y_G is done as follows:

$$(3.9a) \quad D_F^T D_F x_F = D_F^T e,$$

$$(3.9b) \quad y_G = D_G^T (D_F x_F - e).$$

One can first solve for x_F in Eq. (3.9a) and use it to compute y_G in Eq. (3.9b). We call the computed pair (x_F, y_G) a complementary basic solution.

If a complementary basic solution (x_F, y_G) satisfies $x_F \geq 0$ and $y_G \geq 0$, then it is called *feasible*. In this case,

$x = (x_F, 0)$ is the optimal solution of Eq. (3.7), and the algorithm terminates. Otherwise, a complementary basic solution (x_F, y_G) is *infeasible*, and we need to update F and G by exchanging variables for which Eq. (3.8b) or Eq. (3.8c) does not hold. Formally, we define the following index sets

$$(3.10a) \quad H_1 = \{i \in F : x_i < 0\}$$

$$(3.10b) \quad H_2 = \{i \in G : y_i < 0\},$$

and update F and G by the following rules:

$$(3.11a) \quad F = (F - H_1) \cup H_2$$

$$(3.11b) \quad G = (G - H_2) \cup H_1.$$

The finite termination property of this strategy with careful modifications is discussed in [20].

For a multiple right-hand side case in Eq. (3.6), Kim and Park [12] significantly improved this algorithm by employing two important improvements. Observe that the sets F and G change over iterations, and Eqs. (3.9) has to be solved for different F and G every time. The first improvement is based on the observation that the matrix D is typically very long and thin. In this case, constructing matrices $D_F^T D_F$, $D_F^T e$, $D_G^T D_F$, and $D_G^T e$ before solving Eqs. (3.9) is computationally very expensive. To ease this difficulty, $D^T D$ and $D^T E$ can be computed in the beginning and reused in later iterations. One can easily see that $D_F^T D_F$, $D_F^T e_j$, $D_G^T D_F$, and $D_G^T e_j$, $j \in \{1, \dots, l\}$, can be directly retrieved as a submatrix of $D^T D$ and $D^T E$. Because the column size of C is small, storage needed for $D^T D$ and $D^T E$ is also small.

The second improvement involves exploiting common computations in solving Eq. 3.9a. Here we simultaneously run the block principal pivoting approach mentioned above for multiple right-hand side vectors. At each iteration, we have the index sets F_j and G_j for each column $j \in \{1, \dots, l\}$, and we must compute x_{F_j} and y_{G_j} using Eqs. (3.9). The idea is to find groups of columns that share the same index sets F_j and G_j . We reorder the columns with respect to these groups and solve Eqs. (3.9) for the columns in the same group. By doing so, we avoid repeated Cholesky factorization computations required for solving Eq. (3.9a). With these modifications, the block principal pivoting method appeared very efficient in solving NNLS problems with multiple right hand sides [12].

4 Extensions to Regularized and Sparse NTF

In this section, we show how the algorithm can be extended to regularized and sparse NTF. There are multiple interpretations for regularization. It aids in numerical stability by solving a different but more stable problem. On

the other hand, from a Bayesian viewpoint, regularization is a maximum a-posteriori (MAP) approximation, and it enforces prior beliefs provided by domain experts.

For regularized NTF, the objective function to be minimized is

$$(4.12) \quad \min_{A, B, C \geq 0} \|\mathbf{X} - \llbracket ABC \rrbracket\|_F^2 + \alpha \|A\|_F^2 + \beta \|B\|_F^2 + \gamma \|C\|_F^2.$$

Here we iterate solving the following NNLS problems until a stopping criterion is satisfied:

$$\begin{aligned} \min_{A \geq 0} & \left\| \begin{pmatrix} Y_{BC} \\ \sqrt{\alpha} I_{r \times r} \end{pmatrix} A^T - \begin{pmatrix} \mathbf{X}^{(1)} \\ 0_{r \times m} \end{pmatrix} \right\|_F^2, \\ \min_{B \geq 0} & \left\| \begin{pmatrix} Y_{AC} \\ \sqrt{\beta} I_{r \times r} \end{pmatrix} B^T - \begin{pmatrix} \mathbf{X}^{(2)} \\ 0_{r \times n} \end{pmatrix} \right\|_F^2, \\ \min_{C \geq 0} & \left\| \begin{pmatrix} Y_{AB} \\ \sqrt{\gamma} I_{r \times r} \end{pmatrix} C^T - \begin{pmatrix} \mathbf{X}^{(3)} \\ 0_{r \times p} \end{pmatrix} \right\|_F^2, \end{aligned}$$

where α, β, γ are regularization coefficients, $I_{r \times r}$ is an $r \times r$ identity matrix, and $0_{x \times y}$ is a zero matrix of dimensions $x \times y$. The role of the parameters α, β, γ with small values is to impose full rank on the matrices on the left hand side of variable matrices in the NNLS subproblems.

Our sparse NTF formulation incorporates ℓ_1 -norm regularization. The idea of using ℓ_1 -norm regularization for the purpose of achieving sparsity has been successfully utilized in a variety of problems [25]. Without loss of generality, we assume that one of the loading matrix, say A , is to be sparse. In that case, sparse NTF can be viewed as minimizing the following objective function:

$$(4.13) \quad \min_{A, B, C \geq 0} \|\mathbf{X} - \llbracket ABC \rrbracket\|_F^2 + \alpha \sum_{j=1}^r \|A(:, j)\|_1^2 + \beta \|B\|_F^2 + \gamma \|C\|_F^2.$$

In this case, we iterate solving the following NNLS problems until a stopping criterion is satisfied:

$$\begin{aligned} \min_{A \geq 0} & \left\| \begin{pmatrix} Y_{BC} \\ \sqrt{\alpha} e_{1 \times r} \end{pmatrix} A^T - \begin{pmatrix} \mathbf{X}^{(1)} \\ 0_{1 \times m} \end{pmatrix} \right\|_F^2, \\ \min_{B \geq 0} & \left\| \begin{pmatrix} Y_{AC} \\ \sqrt{\beta} I_{r \times r} \end{pmatrix} B^T - \begin{pmatrix} \mathbf{X}^{(2)} \\ 0_{r \times n} \end{pmatrix} \right\|_F^2, \\ \min_{C \geq 0} & \left\| \begin{pmatrix} Y_{AB} \\ \sqrt{\gamma} I_{r \times r} \end{pmatrix} C^T - \begin{pmatrix} \mathbf{X}^{(3)} \\ 0_{r \times p} \end{pmatrix} \right\|_F^2, \end{aligned}$$

where α is a coefficient controlling sparsity, $e_{1 \times r}$ is a vector of ones.

Data set	Dimensions	Speed up
Synthetic	$100 \times 433 \times 200$	3.6
Amino-acid	$5 \times 201 \times 61$	7.4
CMU face images	$128 \times 120 \times 640$	9.8
Enron (3-way)	$3000 \times 141 \times 141$	12.2
Enron (4-way)	$39573 \times 197 \times 197 \times 357$	11.3
IEEE Vast 2007	$12121 \times 7141 \times 15$	8.9

Table 1: Data sets used, their sizes, and amount of speed-up: The values in “Speed up” column represent the amount of average speedup the **BPP** algorithm has over the second fastest algorithm.

5 Experiments and Results

We experimented with various data sets from different application domains. The data sets and their size are shown in Table 1, and each of the data sets are explained in more detail below. We compared the following algorithms for NTF:

1. (**BPP**) NTF using the block principal pivoting method proposed in this paper
2. (**ACTSET**) NTF using the active set method [11]
3. (**AB**) Andersson and Bro’s NTF [1]
4. (**MU**) NTF using multiplicative updates [17, 27]

For fair timing comparison, we present the relative residual as a function of time to see the convergence and the rate of convergence of each algorithm. Below, we give a brief description of each data set used, and we proceed to timing and application results. All the experiments were performed with MATLAB version 7.8 in a 3.2 GHZ Pentium 4 machine with Linux OS.

5.1 Description of data sets We used data sets from various domains including text mining, social network (email network) analysis, image processing, and bioinformatics. This illustrates the wide applicability of the proposed NTF algorithm.

Synthetic tensor: For this data set, multiple tensors of size $100 \times 433 \times 200$ were generated. Each entry of the tensor was sampled independently from a uniform distribution $U(0, 1)$ making the random tensor inherently nonnegative.

*Amino acid data set*²: This data set consists of fluorescence data (AMINO) containing five samples with different amounts of tryptophan, phenylalanine, and tyrosine. This data set has a small number of negative values which came from the intrinsic uncertainty in real experimental measurements that included noise. Hence, having

²<http://www.models.life.ku.dk/research/data/AminoAcid/fluo/index.asp>

such small negative values is not contradictory to the non-negativity assumption; in other words, true parameters are still nonnegative. It is a relatively small 3-way data set, with dimensionality $5 \times 201 \times 61$. Each sample was excited at 61 wavelengths (240 - 300 nm in 1 nm interval), and fluorescence emission intensities are measured at 201 wavelengths (250 - 450 nm in 1 nm interval). Each element of the tensor represents fluorescence emission signal intensity. A three-component PARAFAC model ($k = 3$) was chosen for this data set since we already know that each signal intensity comes from three analytes. The values in the first, second, and third loading matrices represent the sample mode loadings, the emission mode loadings, and the excitation mode loadings, respectively.

*CMU Face data set*³: This data set consists of 640 facial images taken with varying pose, expression, eyes, and size. Each image is of dimension 128×120 . Standard image processing approaches treat each image as a 15,360 dimensional vector, but tensors provide a way to model images without the need for vectorizing them, thereby preserving the inherent spatial relationship in the image. In addition, all the pixel values are inherently non-negative, and hence we impose the non-negativity constraint on the factors.

*Enron email data set*⁴: The Enron email data set consists of email exchanges between employees of Enron corporation. The raw Enron corpus contains 619,446 messages belonging to 158 users. For experimental purposes, a subset of Enron emails was selected. We created two types of tensor data sets from the raw data set. The base directory in both cases consisted of 121,393 terms.

For the first experiment, a 4-way term-author-recipient-day tensor with $39,573 \times 197 \times 197 \times 357$ dimensions was constructed. This can be useful for tracking discussion between the users on a day-by-day level about particular topics. The $ijkl^{th}$ element of the tensor represents the count of the term i used by author j while emailing recipient k on the l^{th} day.

For next experiment, we created a 3-way author-recipient-keyword tensor with $3,000 \times 141 \times 141$ dimensions. This tensor did not take into account the time stamp of the emails. The ijk^{th} entry represents how many times the user i^{th} had used the k^{th} keyword, while conversing with user j . This tensor was created to see which group of people talked to each other about what particular topic.

*IEEE VAST 2007 Contest data set*⁵: This data set consists of a tensor formed out of 1,455 text files corresponding to news stories, email messages, or blog posts from

³<http://mlr.cs.umass.edu/ml/datasets/CMU+Face+Images>

⁴<http://www.cs.cmu.edu/~enron/>

⁵<http://www.cs.umd.edu/hcil/VASTcontest07>

the VAST data set. In addition to the plain text versions of these files, the data set includes tag information such as date, person, location, organization, and money. The date tag was used to help extract the time stamp. The four remaining tags all represent entities of interest and were used to create the tensor model. We considered term-by-entity associations in the news stories over monthly time intervals, which corresponds to a sparse tensor in $12, 121 \times 7, 141 \times 15$ dimensions with 1, 142, 077 nonzeros. Detailed expressions for calculating the entries of the tensor can be found in [2].

5.2 Timing comparison The algorithms were compared for speed and the relative sum of squares residual error (RSSR):

$$(5.14) \quad RSSR = \frac{\sum_{i,j,z} e_{ijz}^2}{\sum_{i,j,z} x_{i,j,z}^2},$$

where e_{ijz} and $x_{i,j,z}$ are as shown in Eq. (2.1). We denote the $RSSR$ value of t -th iteration by $RSSR(t)$.

For Enron, VAST, CMU face, and synthetic data sets, the results are reported in Figure 3. The graphs show the $RSSR$ values with respect to computation time for several algorithms. From the figures, it is clear that our new method outperforms other existing algorithms: Regardless of the duration that the algorithms are run and stopped, the **BPP** method would provide the lowest $RSSR$ value. The advantage of block principal pivoting method is generally greater when the number of factors to be recovered (r) is larger. For the amino acid data set, the results are reported in Table 2. The table shows the $RSSR$ values achieved at the specified amount of time. Note that the **BPP** method achieved a comparable $RSSR$ value at a smaller amount of time.

Results for regularized and sparse NTF are reported in Table 3 for **BPP** and **ACTSET** algorithms. The timing values in the table are calculated as follows. The **BPP** algorithm was run until $(RSSR(t-1) - RSSR(t))$ was less than 10^{-6} , and the value of $RSSR$ at the iteration t was noted. The corresponding time was reported for **BPP** algorithm. Next, other algorithms were run until they achieve same value of $RSSR$. **ACTSET** algorithm achieved the $RSSR$ value at the time mentioned, which is greater than the timing reported for **BPP**. Other algorithms took longer than **ACTSET** algorithm.

To summarize the relative efficiency, we report in Table 1 the average speedup of our algorithm over the second fastest algorithm. We first picked 30 $RSSR$ values randomly from a uniform distribution $U(RSSR_{min}, 1)$ where $RSSR_{min}$ was the smallest observed $RSSR$ value for each data set. Then, we measured time required to achieve each of the $RSSR$ values by each algorithm. For

each case, we calculated the speedup of the **BPP** algorithm over the second fastest one, and the average speedup over 30 $RSSR$ values are shown in the table.

5.3 Factor recovery in presence of noise To show that our algorithm correctly recovers the factors, we adopted a visual illustration. Three 64×64 images were treated as the loading matrices, and the original tensor was formed from these loading matrices according to Eq. (2.1). For example, the first image corresponds to loading matrix A , the second image to B , and the third image to C . An additional Gaussian noise with variance 1 was also added to the tensor. Afterwards, we ran each of the decomposition algorithms on the constructed tensor to recover the loading matrices. It should be noted that, this is an experiment for visual illustration of the factor recovery using NTF decomposition and is not related to Tensorfaces or similar face recognition experiment. Figure 2 shows the original and the recovered loading matrices by various algorithms when they were executed for the same amount of time. It can be seen that by the time the **BPP** algorithm has recovered the factors successfully, other algorithms were yet to compute the factors.

Numerical measures of the recovery in the presence of noise are also reported in Table 4. Timings shown in the table were obtained as follows. As **BPP** shows fastest trend in reducing the $RSSR$ value as demonstrated in Figure 3, we first ran **BPP** until $(RSSR(t-1) - RSSR(t))$ becomes smaller than 10^{-6} . Then, all other algorithms were executed for the same amount of time, and their $RSSR$ values are shown in the table. The results imply that under limited amount of computation time, **BPP** shows better recovery in general.

5.4 Topic identification using NTF In this subsection, we describe results of applying our NTF algorithm for topic identification.

Group discussions in Enron data set: We applied sparse NTF on the 3-way tensor derived from the Enron data set to identify topics and their participants. Sparsity constraints were enforced on two (sender and receiver) out of the three modes. Incorporating sparsity helps us to remove noise in the groups of users who discuss about a particular topic. For example, some users who were only once involved in the discussion on a particular topic should not be counted as a participant of the topic. Using sparse NTF helps in making the particular component in the factor zero. Sparse NTF with $r = 10$ was applied, and the keywords of the topics and percentage of the users participating in each topic are summarized in Table 5.

Scenario Discovery in IEEE VAST data set: We

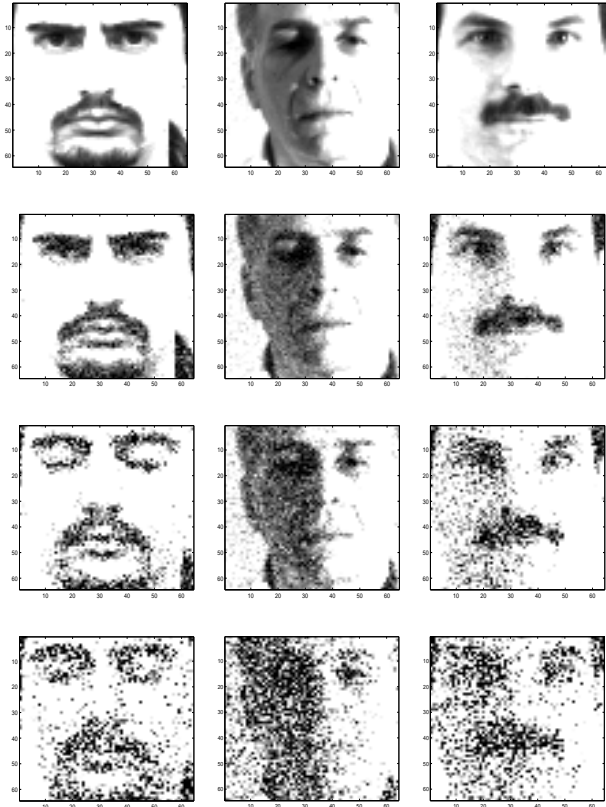


Figure 2: Original images (1st row) and recovered images using **BPP** (2nd row), **AB** (3rd row), and **MU** (4th row) algorithms when they were computed for the same amount of time.

used our NTF algorithm for scenario discovery in the IEEE Vast 2007 Contest data set to illustrate how term-entity-month based NTF model can be used for scenario discovery. The data set consist of news stories and blog entries related to wildlife law enforcement, but mixed with some noisy information. Participants are asked to discover a major law enforcement/counter-terrorism scenario, form their hypotheses, and collect supporting evidences. In order to perform the task, the raw data was processed into a tensor format as described in [2]. We used a rank 25 approximation of the original tensor and listed the keywords of the discovered scenario in Table 6. We note that the detected topic is consistent with the ground truth revealed after the end of the contest. More details about this experiment can be found in [2].

6 Conclusions

In this paper, we presented a fast algorithm for nonnegative tensor factorization by decomposing the original minimization problem as a sequence of the NNLS problems.

	r	BPP	ACTSET	AB	MU
Time(sec)	3	0.955	1.827	2.654	12.551
$RSSR$	3	0.098	0.098	0.099	0.142

Table 2: Timing comparison on amino acid data set: $\mathbf{X} \in \mathbb{R}_+^{5 \times 201 \times 61}$. The $RSSR$ values were achieved by several algorithms at the corresponding amount of time.

r	Regularized NTF		Sparse NTF	
	BPP	ACTSET	BPP	ACTSET
10	1.568	3.144	1.486	2.921
50	11.112	23.091	10.055	21.991
100	59.032	92.543	58.185	90.321

Table 3: Timing comparison on a synthetic tensor $\mathbf{X} \in \mathbb{R}_+^{173 \times 234 \times 854}$. Parameters used: $\alpha = 0.4, \beta = 0.2$, and $\gamma = 0.06$ for regularized NTF, and $\alpha = 0.5, \beta = 0.04$, and $\gamma = 0.2$ for sparse NTF.

We use the block principal pivoting algorithm to efficiently solve the NNLS problems. Our algorithm provides a faster way of computing nonnegative tensor factorization and its regularized and sparse extensions. Experimental results show that the new algorithm is much faster than existing ones. We also demonstrate the applicability of our fast algorithm in analyzing large-scale multi-dimensional text data sets.

References

- [1] C. Andersson and R. Bro. The N-way toolbox for MATLAB. *Chemometrics and Intelligent Laboratory Systems*, 52(1):1–4, 2000.
- [2] B. Bader, A. Pureskiy, and M. Berry. Scenario discovery using nonnegative tensor factorization. In *Progress in Pattern Recognition, Image Analysis and Applications, Proceedings of the Thirteenth Iberoamerican Congress on Pattern Recognition, CIARP*, pages 791–805. Springer, 2008.
- [3] M. H. V. Benthem and M. R. Keenan. Fast algorithm for the solution of large-scale non-negativity-constrained least squares problems. *Journal of Chemometrics*, 18:441–450, 2004.
- [4] D. Bertsekas, M. Homer, D. Logan, and S. Patek. Nonlinear programming. *Athena scientific*, 1995.
- [5] R. Bro and S. De Jong. A fast non-negativity-constrained least squares algorithm. *Journal of Chemometrics*, 11(5), 1997.
- [6] A. Cichocki, R. Zdunek, and S. Amari. Nonnegative matrix and tensor factorization. *John Wiley and Sons, Ltd*, 2009.
- [7] M. P. Friedlander and K. Hatz. Computing non-negative tensor factorizations. *Optimization Methods and Software*, 23(4):631, 2008.
- [8] R. Harshman. Foundations of the PARAFAC procedure: Models and conditions for an explanatory multi-modal

r	σ^2	BPP	ACTSET	AB	MU
30	1.5	0.239	0.244	0.254	0.425
60	1.5	0.063	0.063	0.064	0.085
100	1.5	0.009	0.009	0.011	0.021
30	10	0.515	0.518	0.513	0.687
60	10	0.438	0.439	0.442	0.493
100	10	0.173	0.174	0.190	0.390

Table 4: Factor recovery under noise on two synthetic tensors in $\mathbb{R}_+^{100 \times 221 \times 631}$ with Gaussian noise with two different variances. A smaller the $RSSR$ value means better the recovery.

Topics	% of users involved
California legislature	4.3
India Dabhol Power Company	2.2
Downfall	16.3
Downfall(newsfeed)	9.3
El Paso Energy	5.3
Cal energy	7.8
TruOrange/College football	4.2
Enrononline	4.3
911 Sept 2001	5.6
NFL	2.3

Table 5: List of 10 topics identified by sparse NTF along with the percentage of people involved in the communication on each topic.

factor analysis. *UCLA working papers in phonetics*, 16:1–84, 1970.

- [9] J. Håstad. Tensor rank is np-complete. *J. Algorithms*, 11(4):644–654, 1990.
- [10] H. Kim and H. Park. Non-negative matrix factorization based on alternating non-negativity constrained least squares and active set method. *SIAM Journal in Matrix Analysis and Applications*, 30(2):713–730, 2008.
- [11] H. Kim, H. Park, and L. Elden. Non-negative Tensor Factorization Based on Alternating Large-scale Non-negativity-constrained. In *Proceedings of IEEE 7th International Conference on Bioinformatics and Bioengineering (BIBE07)*, pages 1147–1151, 2007.
- [12] J. Kim and H. Park. Toward faster nonnegative matrix factorization: A new algorithm and comparisons. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (ICDM)*, pages 353–362, 2008.
- [13] T. G. Kolda and B. W. Bader. The TOPHITS model for higher-order web link analysis. In *Proceedings of the Workshop on Link Analysis, Counterterrorism and Security*, 2006.
- [14] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.
- [15] L. D. Lathauwer, B. D. Moor, and J. Vandewalle. A multilinear singular value decomposition. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.
- [16] D. Lee and H. Seung. Learning the parts of objects by

Topics
People for the Ethical Treatment of Animals, Animal rights, attacks on pet store
Monkeypox, Chinchilla
Arsons, Fire investigation
Drug trafficking, Exotic animal, Illegal trade
Animal Justice League, Protest activities
Animal treatment standards, Meat alternatives, Benefits for consumer health
Conservation of tigers, leopards
Wild chinchilla, Harvesting in Chile
Bullfighting, Cockfighting

Table 6: List of 9 topics identified by NTF from the VAST 2007 contest data set.

- non-negative matrix factorization. *Nature*, 401:788–791, 1999.
- [17] D. Lee and H. Seung. Algorithms for non-negative matrix factorization. In *Advances in Neural Information Processing Systems*, pages 556–562, 2001.
- [18] L. Lim and P. Comon. Nonnegative approximations of nonnegative tensors. *Journal of Chemometrics*, 23:432–441, 2009.
- [19] C. Lin. Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, 19(10):2756–2779, 2007.
- [20] L. Portugal, J. Judice, and L. Vicente. A comparison of block pivoting and interior-point algorithms for linear least squares problems with nonnegative variables. *Mathematics of Computation*, pages 625–643, 1994.
- [21] A. Shashua and T. Hazan. Non-negative tensor factorization with applications to statistics and computer vision. In *Proceedings of the 22nd international conference on Machine learning (ICML)*, 2005.
- [22] A. Shashua and A. Levin. Linear image coding for regression and classification using the tensor-rank principle. In *Proceedings of the 2001 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001.
- [23] V. D. Silva and L.-H. Lim. Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1084–1127, 2008.
- [24] J. Sun, S. Papadimitriou, C. Lin, N. Cao, S. Liu, and W. Qian. Multivis: Content-based social network exploration through multi-way visual analysis. In *Proceedings of the 2009 SIAM International Conference on Data Mining (SDM)*, volume 9, pages 1063–1074, 2009.
- [25] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.
- [26] M. Vasilescu and D. Terzopoulos. Multilinear analysis of image ensembles: Tensorfaces. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 447–460, 2002.
- [27] M. Welling and M. Weber. Positive tensor factorization. *Pattern Recognition Letters*, 22(12):1255–1261, 2001.

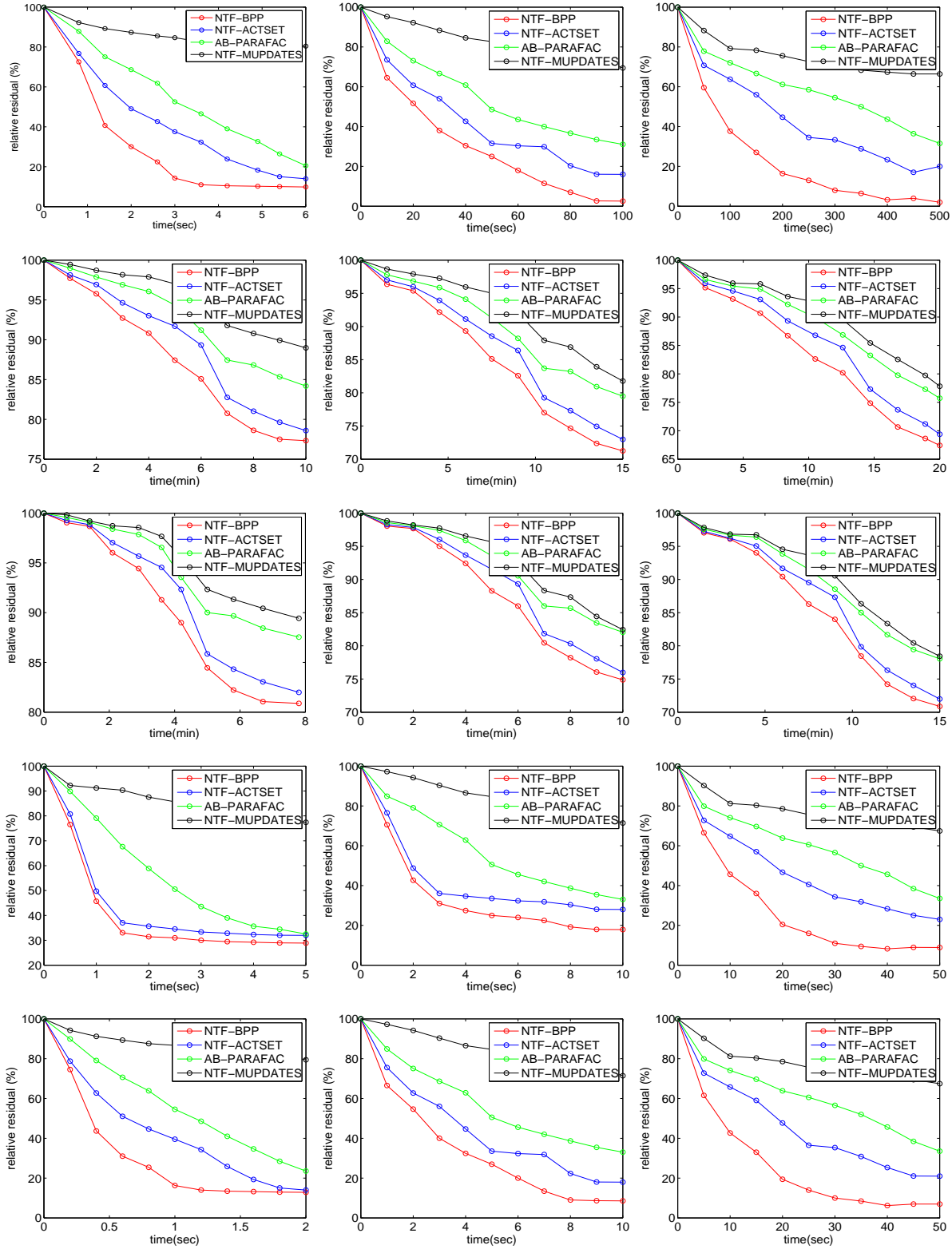


Figure 3: Timing-vs-RSSR graphs. The rows represent data sets: Enron 3-way (1st row), Enron 4-way (2nd row), VAST (3rd row), CMU facial (4th row), and synthetic (5th row) tensors. The columns represent reduced ranks: $r = 10$ (left), $r = 50$ (middle) and $r = 90$ (right). The initialization of loading matrices were done randomly. The values reported in the graphs represent average value over 1,000 initializations for all data sets. For the synthetic data set (5th), the averaging was done over 1,000 synthetic tensor instances and 1,000 random initializations of loading matrices.