

# On the equivalence between nonnegative tensor factorization and tensorial probabilistic latent semantic analysis

Wei Peng · Tao Li

© Springer Science+Business Media, LLC 2010

**Abstract** Non-negative Matrix Factorization (NMF) and Probabilistic Latent Semantic Analysis (PLSA) are two widely used methods for non-negative data decomposition of two-way data (e.g., document-term matrices). Studies have shown that PLSA and NMF (with the Kullback-Leibler divergence objective) are different algorithms optimizing the same objective function. Recently, analyzing multi-way data (i.e., tensors), has attracted a lot of attention as multi-way data have rich intrinsic structures and naturally appear in many real-world applications. In this paper, the relationships between NMF and PLSA extensions on multi-way data, e.g., NTF (Non-negative Tensor Factorization) and T-PLSA (Tensorial Probabilistic Latent Semantic Analysis), are studied. Two types of T-PLSA models are shown to be equivalent to two well-known non-negative factorization models: PARAFAC and Tucker3 (with the KL-divergence objective). NTF and T-PLSA are also compared empirically in terms of objective functions, decomposition results, clustering quality, and computation complexity on both synthetic and real-world datasets. Finally, we show that a hybrid method by running NTF and T-PLSA alternatively can successfully jump out of each other's local minima and thus be able to achieve better clustering performance.

**Keywords** Nonnegative tensor factorization · Probabilistic latent semantic analysis · Clustering

---

W. Peng  
Xerox Innovation Group, Xerox Corporation, Rochester,  
NY 14580, USA  
e-mail: [wei.peng@xerox.com](mailto:wei.peng@xerox.com)

T. Li (✉)  
School of Computer Science, Florida International University,  
Miami, FL 33199, USA  
e-mail: [taoli@cs.fiu.edu](mailto:taoli@cs.fiu.edu)

## 1 Introduction

### 1.1 NMF and PLSA

Non-negative data have been widely studied in many applications and research areas. Typical examples include document-term co-occurrence data (count data or binary data) in text clustering and summarization; and friend-of-friend network data in social community detection and collaborative filtering. Among many non-negative data analysis methods, non-negative data decomposition is one of the most popular approaches. For decomposing non-negative data, the two most widely used methods are Non-negative Matrix Factorization (NMF) [21] from matrix computation perspective and Probabilistic Latent Semantic Analysis (PLSA) from the statistical learning perspective.

NMF factorizes an input nonnegative matrix into a product of two new matrices with lower rank. The initial work on NMF [18, 19] emphasizes that the NMF factors contain coherent parts of the original data (images). Later works [8, 22–24, 38] show the usefulness of NMF for clustering with experiments on documents collections, and recent theoretical results [5, 6] show the equivalence between NMF, K-means and spectral clustering. In contrast to NMF, PLSA uses latent class models (or aspect models) to perform a probabilistic mixture decomposition. Expectation-Maximization (EM) algorithm is a commonly used approach to maximize the log-likelihood of the latent class models. PLSA is often used in natural language processing, information retrieval, and text mining related areas [15]. It has been shown that NMF with the KL-divergence (Kullback-Leibler divergence) cost function is equivalent to PLSA [7, 11].

## 1.2 Multi-way data analysis

Recently, decomposing **multi-way** non-negative data has attracted a lot of attention as multi-way data have rich intrinsic structures and naturally appear in many real-world applications [1, 17, 36]. For example, in document clustering, the data over different time periods can be represented as a three-way dataset as *author*  $\times$  *terms*  $\times$  *time*. In email communications, the data can be represented as *sender*  $\times$  *receiver*  $\times$  *time*. In web page personalization, the data can be represented as *user*  $\times$  *query word*  $\times$  *webpage* [34]. In high-order web link analysis, the data is represented as a three-way dataset as *web page*  $\times$  *web page*  $\times$  *anchor text* [17]. Instead of performing traditional two-way data analysis (e.g., matrix decomposition) by unwrapping multi-way data into matrices and assuming only pairwise relationships between two dimensions (rows and columns), multi-way data analysis methods (e.g., tensor decomposition) in these applications consider the relationships among multiple dimensions. Typical examples of non-negative multi-way data analysis methods are extensions of NMF and PLSA to multi-way non-negative data, namely NTF (Non-negative Tensor Factorization) and T-PLSA (Tensorial Probabilistic Latent Semantic Analysis) respectively.

We note that there are generally two types of tensor factorization models: (1) **Parafac** (parallel factor analysis) [29]: The Parafac model can be thought as a multilinear form of decomposition for the objective tensor: each entry of the three-way tensor is approximated by a linear combination of three vectors [29]. (2) **Tucker**: The Tucker model, proposed by L.R.Tucker, includes Tucker1, Tucker2, Tucker3, etc. Tucker model can be thought as multi-way principle component analysis, and aims to give the optimal low rank approximation of a tensor in given dimensions [9]. HOSVD (Higher Order Singular Value Decomposition [35, 37]) is **Tucker3** (it allows rank reduction in all three modes) with orthogonality constraints on the components. Many multi-way models can be considered as the extensions or modifications of the above two types [1, 16, 39]. These multi-way models have also been applied to many applications such as web link analysis [17], web page personalization [34], social network analysis [3, 26], and others [2, 10, 27, 35]. Non-negative tensor factorization (NTF) is first introduced by Shashua and Hazan in [32] and it uses Lee and Seung's multiplicative algorithm [20] to minimize the least square errors. In this paper, we will study the non-negative extensions of the two well-known tensor factorization models, Parafac and Tucker3.

Probabilistic Latent Semantic Analysis (PLSA) applies the Expectation Maximization (EM) algorithm to perform maximum likelihood estimation in a latent class model, where the observed distribution is derived from a set of latent classes [14, 15]. EM algorithm for two-way PLSA

models can be naturally extended to higher dimensional tensors by adding more latent variables in the maximum likelihood function. A few efforts have been done to extend PLSA on multi-way data in different approaches recently. For instance, Shashanka et al. proposed probabilistic latent variable models for generalization of PLSA [31]. Chi et al. also proposed an extension of PLSA called Probabilistic Polyadic Factorization on multi-way data for personalized recommendation [4].

## 1.3 Contribution of the paper

Although PLSA and NMF are shown to be equivalent [7, 11], limited attempts have been reported to establish the connections between T-PLSA and the well-known NTF models, and to empirically demonstrate their similarities and differences. The contribution of this paper is listed as follows:

- The paper studies two types of T-PLSA (ParaAspect and TuckAspect<sup>1</sup>) and two well-known NTF methods (non-negative Parafac (NParafac) and non-negative Tucker3 (NTucker)). It shows that ParaAspect (with KL-divergence objective) and NParafac, TuckAspect (with KL-divergence objective function) and NTucker optimize the same objective functions. In addition, the factorization of non-negative Parafac and Tucker3 with column L1-normalization is equivalent to that of ParaAspect and TuckAspect, respectively.
- T-PLSA and NTF will be compared empirically in terms of objective functions, decomposition results, clustering quality, and computation complexity via extensive experiments on both synthetic and real-world datasets.
- Due to different optimization procedures, ParaAspect and NParafac, TuckAspect and NTucker are shown to converge to different local minima even though they optimize the same objective functions. This paper will demonstrate that a hybrid method, by running NTF and T-PLSA alternatively, can successfully jump out of each other's local minima and thus be able to lead to better clustering solutions.

A preliminary version of the work was presented as a 2-page poster at the 32nd Annual ACM SIGIR Conference [25]. In this journal submission, we included detailed theoretical analysis and algorithm description, and more experimental results. The rest of the paper is organized as follows: Sect. 2 introduces the notations used in the paper and presents the factorization models and objective functions of T-PLSA and NTF. Section 3 shows that T-PLSA and NTF optimize the same objective function. Section 4 describes

<sup>1</sup>Para is from Parafac, Tuck is from Tucker3, Aspect comes from Aspect model.

and compares the algorithms of T-PLSA and NTF. Section 5 empirically compares T-PLSA and NTF from various perspectives on both synthetic three-way data and real-world three-way data. Finally Sect. 6 concludes.

## 2 The factorization models and objective functions of T-PLSA and NTF

### 2.1 Notation

The following notations are used throughout the paper. Scalars are denoted by lowercase letters, e.g.  $x$ , and vectors are denoted by boldface lowercase letters, e.g.  $\mathbf{x}$ , where the  $i$ -th entry is  $x_i$ . Matrices are denoted by boldface capital letters, e.g.  $\mathbf{X}$ , where the  $i$ -th column of matrix  $\mathbf{X}$  is  $\mathbf{x}_i$ , and the  $(i, j)$ -th entry is  $x_{ij}$ . Three-way arrays are denoted by boldface underlined letters, e.g.  $\underline{\mathbf{X}}$ , which can be unfolded in the  $n$ -th mode to form a matrix denoted by  $\underline{\mathbf{X}}_{(n)}$ . The  $c$ -th frontal slice of  $\underline{\mathbf{X}}$  denoted by  $\underline{\mathbf{X}}_c$  is formed by holding the last mode of the multi-way array fixed at  $c$ . The symbol  $\otimes$  is Kronecker product that is the operation between two arbitrary matrices. The Kronecker product of the matrix  $\mathbf{A} \in \mathbb{R}^{a \times b}$  and the matrix  $\mathbf{B} \in \mathbb{R}^{c \times d}$  is the matrix  $\mathbf{C} \in \mathbb{R}^{ac \times bd}$ , where each entry is the product of two entries from  $\mathbf{A}$  and  $\mathbf{B}$  respectively. The symbol  $\odot$  denotes Khatri-Rao product.  $\mathbf{A} \odot \mathbf{B} = [\mathbf{a}_1 \otimes \mathbf{b}_1 \mathbf{a}_2 \otimes \mathbf{b}_2 \cdots \mathbf{a}_k \otimes \mathbf{b}_k]$  ( $\mathbf{A}$  and  $\mathbf{B}$  need to have the same number of columns).

### 2.2 The factorization models of T-PLSA and NTF

Let  $\mathbf{F} = (f_{ij})$  be a non-negative matrix. In document clustering, it is viewed as a document-word co-occurrence matrix. The NMF factorization model is  $\mathbf{F} \simeq \mathbf{C}\mathbf{H}^T$  with  $c_{ij} \geq 0$  and  $h_{ij} \geq 0$ . The PLSA latent variable model can be written as  $f_{ij} \simeq P(w_i, d_j) = \sum_k (P(w_i|z_k)P(d_j|z_k)P(z_k))$ , where  $P(w_i, d_j)$  is the joint probability of the  $i$ -th word and  $j$ -th document,  $P(w_i|z_k)$  and  $P(d_j|z_k)$  are probabilities of  $i$ -th word and  $j$ -th document conditioned on the class  $z_k$ .

NTF and T-PLSA are the extensions of NMF and PLSA on multi-way data, respectively. Without loss of generality, three-way data are mainly focused in this paper. Let  $\underline{\mathbf{F}} = (f_{ijl})$ , where  $1 \leq i \leq n$ ,  $1 \leq j \leq m$ , and  $1 \leq l \leq t$ , be a non-negative tensor. It can be viewed as a document-word-time co-occurrence tensor indicating the occurring frequency of a particular word appearing in a particular document at a particular time or a sender-receiver-time tensor showing the number of emails sending from a particular sender to a particular receiver in a particular time. Normalize  $\underline{\mathbf{F}}$  by L1-normalization that  $\sum_{ijl} f_{ijl} = 1$ ,  $f_{ijl}$  can be viewed as the joint probability of  $d_i$ ,  $w_j$ , and  $e_l$ .

### 2.2.1 NTF factorization models

There are generally two most well-known tensor decomposition models, **Parafac** and **Tucker**. Both of them try to reconstruct the original tensor  $\underline{\mathbf{F}} \simeq \underline{\mathbf{C}}$ . **Parafac** with non-negative constraints can be written as

$$\underline{\mathbf{C}}_1 = \mathbf{U}\mathbf{S}_1\mathbf{V}^T, \tag{1}$$

or

$$c_{ijl} = \sum_p (u_{ip}v_{jp}s_{lp}), \tag{2}$$

where  $\mathbf{U} \in \mathbb{R}^{n \times k}$ ,  $\mathbf{V} \in \mathbb{R}^{m \times k}$ ,  $\mathbf{S} \in \mathbb{R}^{t \times k}$ ,  $\mathbf{S}_1$  is a diagonal matrix with the  $l$ -th row of  $\mathbf{S}$  on the diagonal, and  $u_{ip} \geq 0$ ,  $v_{jp} \geq 0$ ,  $s_{lp} \geq 0$ . Note that it only allows the same number of factors in each mode, and the  $i$ -th factor in one mode only interacts with the  $i$ -th factors in other modes (e.g., one-to-one interactions). Rank-1 decomposition is a type of Parafac with orthogonality constraints on components [1].

The non-negative **Tucker3** can be written as

$$c_{ijl} = \sum_{pqr} (g_{pqr}u_{ip}v_{jq}s_{lr}), \tag{3}$$

where  $\mathbf{U} \in \mathbb{R}^{n \times k_1}$ ,  $\mathbf{V} \in \mathbb{R}^{m \times k_2}$ ,  $\mathbf{S} \in \mathbb{R}^{t \times k_3}$ , and  $\mathbf{G} \in \mathbb{R}^{k_1 \times k_2 \times k_3}$ . HOSVD (High Order SVD) is also called Tucker3 tensor model with orthogonality constraints on components [1].

### 2.2.2 T-PLSA factorization models

Two types of PLSA extension on multi-way data can be derived by simply adding more latent factors. In order to relate them to NParafac and NTucker, in this paper, one type of PLSA extension mentioned in [31] is referred as **ParaAspect** model and the other type introduced by [4] is called **TuckerAspect** model. **ParaAspect** factorization model can be written as

$$f_{ijl} \simeq P(d_i, w_j, e_l) = \sum_p (P(d_i|x_p)P(w_j|x_p)P(e_l|x_p)P(x_p)), \tag{4}$$

where  $(x_p|1 \leq p \leq k)$  are latent class variables/factors, and  $P(d_i, w_j, e_l)$  is the joint probability of  $d_i$ ,  $w_j$ , and  $e_l$ .  $P(d_i|x_p)$ ,  $P(w_j|x_p)$ , and  $P(e_l|x_p)$  are the probabilities of generating  $d_i$ ,  $w_j$ , and  $e_l$  respectively, when the class  $x_p$  is chosen.

**TuckAspect** has a different latent variable model from ParaAspect. It contains several sets of latent factors  $(x_p|1 \leq p \leq k_1)$ ,  $(y_q|1 \leq q \leq k_2)$ , and  $(z_r|1 \leq r \leq k_3)$ . Documents, words, and time are generated from different latent factor sets, e.g.  $P(d_i|x_p)$ ,  $P(w_j|y_q)$ , and  $P(e_l|z_r)$ . In addition, the joint probability of these factors is generated, e.g.

$P(x_p, y_q, z_r)$ . The factorization model of TuckAspect is presented as

$$f_{ijl} \simeq P(d_i, w_j, e_l) = \sum_p (P(d_i|x_p)P(w_j|y_q)P(e_l|z_r)P(x_p, y_q, z_r)). \quad (5)$$

### 2.3 The objective functions of T-PLSA and NTF

NParafac and NTucker minimize the KL-divergence between the original data and the reconstruction:

$$J_{\text{NTF}} = \sum_{ijl} \left( f_{ijl} \log \frac{f_{ijl}}{c_{ijl}} - f_{ijl} + c_{ijl} \right). \quad (6)$$

ParaAspect and TuckAspect maximize the log-likelihood

$$L = \sum_{ijl} (f_{ijl} \log P(d_i, w_j, e_l)) = -J_{\text{T-PLSA}}. \quad (7)$$

By adding a minus sign, they minimize the objective function  $J_{\text{T-PLSA}}$ .

## 3 The equivalence of T-PLSA and NTF

In this section, T-PLSA and NTF will be shown to be equivalent. First, ParaAspect (with KL-divergence objective) and NParafac, TuckAspect (with KL-divergence objective) and NTucker are shown to optimize the same objective functions. Then, we show that the factorization of NParafac and NTucker with column L1-normalization is equivalent to that of ParaAspect and TuckAspect, respectively.

### 3.1 The equivalent objective functions

Recall (7), T-PLSA minimizes

$$J_{\text{T-PLSA}} = - \sum_{ijl} (f_{ijl} \log P(d_i, w_j, e_l)).$$

If a constant  $\sum_{ijl} (f_{ijl} \log f_{ijl})$  is added (it is a constant because  $f_{ijl}$  is the input), T-PLSA minimizes

$$J_{\text{T-PLSA}} = \sum_{ijl} \left( f_{ijl} \log \frac{f_{ijl}}{P(d_i, w_j, e_l)} \right). \quad (8)$$

Then by adding

$$\begin{aligned} & \sum_{ijl} [P(d_i, w_j, e_l) - f_{ijl}] \\ &= \sum_{ijl} P(d_i, w_j, e_l) - \sum_{ijl} f_{ijl} = 1 - 1 = 0, \end{aligned}$$

T-PLSA minimizes

$$J_{\text{T-PLSA}} = \sum_{ijl} \left( f_{ijl} \log \frac{f_{ijl}}{P(d_i, w_j, e_l)} - f_{ijl} + P(d_i, w_j, e_l) \right). \quad (9)$$

Referring to (6), (9) is exactly the objective function of NTF.

### 3.2 The equivalent factorization

Unlike T-PLSA, NParafac and NTucker have an infinite number of solutions. Normalization is a way to make NParafac and NTucker invariant. In order to compare NTF and T-PLSA, we use L1-normalization for probabilistic formulations. Let the normalized input tensor  $\sum_{ijl} f_{ijl} = 1$ . In the following the NParafac factorization will be shown to be equivalent to that of ParaAspect.

**NParafac and ParaAspect:** Let  $\mathbf{D}^u$ ,  $\mathbf{D}^s$ , and  $\mathbf{D}^v$  be the square diagonal matrices with entries  $d_{jj}^u$ ,  $d_{jj}^s$ , and  $d_{jj}^v$  be  $\sum_i u_{ij}$ ,  $\sum_i s_{ij}$ , and  $\sum_i v_{ij}$ , respectively.  $\hat{\mathbf{U}} = \mathbf{U}(\mathbf{D}^u)^{-1}$ ,  $\hat{\mathbf{S}} = \mathbf{S}(\mathbf{D}^s)^{-1}$ , and  $\hat{\mathbf{V}} = \mathbf{V}(\mathbf{D}^v)^{-1}$  are denoted as the column L1-normalized matrices of  $\mathbf{U}$ ,  $\mathbf{S}$ , and  $\mathbf{V}$ . NParafac factorization of (1) can be written as

$$\underline{\mathbf{E}}_1 = \hat{\mathbf{U}}\hat{\mathbf{S}}\hat{\mathbf{H}}\hat{\mathbf{V}}^T,$$

or

$$f_{ijl} = \sum_p (h_{pp}\hat{u}_{ip}\hat{v}_{jp}\hat{s}_{lp}), \quad (10)$$

where  $\mathbf{H}$  is a diagonal matrix that

$$\mathbf{H} = \mathbf{D}^u\mathbf{D}^s\mathbf{D}^v.$$

We can observe that the factorization in (10) is equivalent to ParaAspect model in (4) with  $\hat{u}_{ip} = P(d_i|x_p)$ ,  $\hat{s}_{lp} = P(e_l|x_p)$ ,  $\hat{v}_{jp} = P(w_j|y_q)$ , and  $h_{pp} = P(x_p)$ . Similar to T-PLSA,

$$\sum_i \hat{u}_{ip} = 1, \quad \sum_j \hat{v}_{jp} = 1, \quad \sum_l \hat{s}_{lp} = 1,$$

resulted from the column L1-normalization. Moreover,  $\sum_p h_{pp} = 1$  because

$$\sum_{ijl} f_{ijl} = 1 = \sum_{ijlp} (h_{pp}\hat{u}_{ip}\hat{v}_{jp}\hat{s}_{lp}) = \sum_p h_{pp}.$$

**NTucker and TuckAspect:** The factorization model of NTucker in (3) can be written as

$$\underline{\mathbf{E}}_{(1)} = \mathbf{U}\underline{\mathbf{G}}_{(1)}(\mathbf{S} \otimes \mathbf{V})^T, \quad (11)$$

where  $\underline{\mathbf{F}}_{(1)}$  is  $\underline{\mathbf{F}}$  matricized on the first mode because of the following:

$$(\underline{\mathbf{U}}\underline{\mathbf{G}}_{(1)})_{i, k_2 \times (r-1) + q} = \sum_p (u_{ip} g_{pqr}),$$

and

$$((\mathbf{S} \otimes \mathbf{V})^T)_{k_2 \times (r-1) + q, m \times (l-1) + j} = s_{lr} v_{jq},$$

then

$$f_{ijl} = (\underline{\mathbf{F}}_{(1)})_{i, m \times (l-1) + j} = \sum_{pqr} (g_{pqr} u_{ip} v_{jq} s_{lr}).$$

Similarly let  $\mathbf{D}^u$ ,  $\mathbf{D}^s$ , and  $\mathbf{D}^v$  be the diagonal matrices with entries being column summation values of  $\mathbf{U}$ ,  $\mathbf{S}$ ,  $\mathbf{V}$ .  $\hat{\mathbf{U}}$ ,  $\hat{\mathbf{S}}$ , and  $\hat{\mathbf{V}}$  are normalized versions of  $\mathbf{U}$ ,  $\mathbf{S}$ , and  $\mathbf{V}$ . Equation (11) is reformulated as

$$\underline{\mathbf{F}}_{(1)} = \hat{\mathbf{U}}\underline{\mathbf{H}}_{(1)}(\hat{\mathbf{S}} \otimes \hat{\mathbf{V}})^T,$$

where

$$\underline{\mathbf{H}}_{(1)} = \mathbf{D}^u \underline{\mathbf{G}}_{(1)} (\mathbf{D}^s \otimes \mathbf{D}^v).$$

Then (11) can be written in the same factorization format as TuckAspect in (5)

$$f_{ijl} = \sum_{pqr} (h_{pqr} \hat{u}_{ip} \hat{v}_{jq} \hat{s}_{lr}). \tag{12}$$

Similar to the derivation between NParafac and ParaAspect,

$$\sum_i \hat{u}_{ip} = 1, \quad \sum_j \hat{v}_{jq} = 1, \quad \sum_l \hat{s}_{lr} = 1,$$

$$\sum_{pqr} h_{pqr} = 1.$$

Thus  $\hat{u}_{ip} = P(d_i|x_p)$ ,  $\hat{v}_{jq} = P(w_j|y_q)$ ,  $\hat{s}_{lr} = P(e_l|z_r)$ , and  $h_{pqr} = P(x_p, y_q, z_r)$ .

#### 4 The algorithm comparison of T-PLSA and NTF

Although T-PLSA and NTF are similar in terms of the objective functions and factorization, they have different computational algorithms because they have different ways to optimize the objective function. T-PLSA use EM algorithm and NTF use multiplicative update rules proposed by [21]. In this section, we compare computational algorithms of T-PLSA and NTF.

#### 4.1 Algorithm of NParafac and Tucker3

Non-negative Parafac and Tucker3 use multiplicative update rules proposed by [21] to alternatively update each component by minimizing the KL-divergence. For updating the component matrix  $\mathbf{U}$  in non-negative Parafac, let

$$\mathbf{Z} = (\mathbf{S} \odot \mathbf{V})^T, \tag{13}$$

and  $\underline{\mathbf{F}}_{(1)} = \mathbf{O}$ , then

$$u_{ip} = u_{ip} \frac{\sum_j (z_{pj} o_{ij} / (\mathbf{UZ})_{ij})}{\sum_j z_{pj}}. \tag{14}$$

For updating other components of NParafac, we just need to simply matricize  $\underline{\mathbf{F}}$  on the other modes, and  $\mathbf{Z}$  is changed to be  $(\mathbf{S} \odot \mathbf{U})^T$  and  $(\mathbf{V} \odot \mathbf{U})^T$  respectively. Similarly they can be updated according to (14).

As for updating the component matrix  $\mathbf{U}$  in non-negative Tucker3,

$$\mathbf{Z} = ((\mathbf{S} \otimes \mathbf{V})\underline{\mathbf{G}}_{(1)}^T)^T. \tag{15}$$

The updating algorithm is still following (14). Other components can be solved by matricizing  $\underline{\mathbf{F}}$  and  $\underline{\mathbf{G}}$  on the other modes.  $\mathbf{Z}$  is changed to be  $((\mathbf{S} \otimes \mathbf{U})\underline{\mathbf{G}}_{(2)}^T)^T$  and  $((\mathbf{V} \otimes \mathbf{U})\underline{\mathbf{G}}_{(3)}^T)^T$ .  $\underline{\mathbf{G}}$  is a close form solution that  $\underline{\mathbf{G}}_{(1)} = \mathbf{U}^T \underline{\mathbf{X}}_{(1)} (\mathbf{S} \otimes \mathbf{V})$ .

#### 4.2 Algorithm of ParaAspect and TuckAspect

ParaAspect and TuckAspect use the standard EM algorithm to maximize log-likelihood functions. In Expectation step, the posterior probabilities of latent variables are computed based on the current parameters. For ParaAspect, the posterior probability is calculated as

$$P(x|d, w, e) = \frac{P(x)P(d|x)P(w|x)P(s|x)}{\sum_x (P(x)P(d|x)P(w|x)P(e|x))}. \tag{16}$$

For TuckAspect, the posterior probability is

$$P(x, y, z|d, w, e) = \frac{P(x, y, z)P(d|x)P(w|y)P(s|z)}{\sum_{x,y,z} (P(x, y, z)P(d|x)P(w|y)P(s|z))}. \tag{17}$$

In Maximization step, the parameters are estimated based on the computed posterior probabilities of the latent variables and the original input tensor. For example  $P(d|x)$  is updated for ParaAspect as following

$$P(d|z) = \frac{\sum_{w,e} (f_{d,w,e} P(x|d, w, e))}{\sum_{d,w,e} (f_{d,w,e} P(x|d, w, e))}. \tag{18}$$

The updating algorithms for ParaAspect and TuckAspect are omitted here. Detailed algorithms can be found in [31] and [4].

### 4.3 Summary

From the above algorithm descriptions, we observe the following differences between NTF and T-PLSA:

- NTF updates one component by fixing other components, and updated component can be subsequently used to update the other components. However, T-PLSA updates the components only based on the posterior probability of the latent variables.
- NParafac and ParaAspect have the same computational complexity, which is  $O(mntk)$ . On the other hand, TuckAspect has larger computational complexity  $O(nmtk_1k_2k_3)$  than NTucker with  $O(mtk_1k_2k_3 + nmtk_1)$ , or  $O(nmtk_1)$  when  $n \gg k_2k_3$ .
- Though NParafac and ParaAspect have the same computational complexity, ParaAspect requires more space for calculating the posterior probabilities of the latent variables, which needs  $O(mntk)$  number of space units (if saving all entries). NParafac only requires  $O(nmt)$  with  $k \ll \min(n, m, t)$ . Moreover, TuckAspect needs  $O(nmtk_1k_2k_3)$  space units while NTucker only needs space  $O(nmt + mtk_2k_3 + ntk_1k_3 + nmk_1k_2)$ , or  $O(nmt)$  when  $n \gg k_2k_3$ ,  $m \gg k_1k_3$ , and  $t \gg k_1k_2$ .

In summary, we can see that NTF and T-PLSA have different algorithms to solve their models. From our experimental results in Sect. 5, we note that NTF and T-PLSA have similar computation convergence rate, that is, they take almost the same number of rounds to converge. Since NTF requires less computation and space than T-PLSA for each round, NTF is more efficient than T-PLSA.

## 5 Empirical comparison between T-PLSA and NTF

### 5.1 Discussion

It is actually very advantageous to know that NTF and T-PLSA are equivalent and have different pros and cons. As discussed in Sect. 4.3, the NTF algorithm has less computational and space complexity than T-PLSA. Thus if the computation speed of T-PLSA is an issue for solving some problems, NTF can be the alternative for T-PLSA by viewing the component factors as probability parameters. On the contrary, T-PLSA has a more solid statistical foundation as a probabilistic mixture model. Its statistical properties enable T-PLSA to perform model selection and to assess the number of classes in the model. However, there is limited literature on how to determine the number of factors for NTF. Now knowing the equivalence between T-PLSA and NTF, we could borrow the model selection technique from T-PLSA for determining the number of factors for NTF. For example, most model selection criteria like BIC

(Bayesian Information Criterion) [30] for probabilistic models are equal to the log-likelihood of the model offset by the number of parameters. The “log-likelihood value” of NTF can be easily calculated because NTF and T-PLSA have the equivalent objective functions and factorization as proven in Sect. 3. In order to further understand the equivalence and difference between T-PLSA and NTF, in the following they will be compared empirically in terms of objective functions, decomposition results, clustering quality, and computation complexity via extensive experiments on both synthetic datasets and real-world datasets.

### 5.2 Experiment setup

#### 5.2.1 Data description

Three real-world datasets are used in our experiments. Two are from the DBLP computer science bibliography, one from Enron emails.

**DBLP Datasets:** DBLP datasets are extracted from the DBLP computer science bibliography that can be downloaded at <http://www.informatik.uni-trier.de/~ley/db/>. We extract author names, publication titles and the corresponding years of the publications. Among these records, 1000 active researchers with their publication titles for the last 20 years (from 1988 to 2007) are chosen for our experiments. The researchers are selected from the program committee members from the flagship conferences in computer science and they are divided into 9 different research areas: *Database*, *Data Mining*, *Software Engineering*, *Theory*, *Computer Vision*, *Operating System*, *Machine Learning*, *Networking*, and *Natural Language Processing* based on the authors’ major activities (e.g., the conference venues of the publications and the conference program committee memberships). These different areas will serve as the ground-truth labels for our experimental comparison purpose. The data are preprocessed by using standard text preprocessing techniques. The terms are extracted from the publication titles. Stop words are removed and we do not perform stemming. For each year, a binary matrix with each entry denoting the co-occurrence of the corresponding author and the term in that year is constructed. Then, the data are actually a three-way array with the author, term, and year modes. We conduct experiments on two such three-way datasets: DBLP9 which contains 20 years publication titles of 1000 authors from all 9 areas and 1000 key terms with the highest occurrence frequency, and DBLP4 which contains 20 years publication titles of 250 authors who are randomly chosen from 1000 authors in 4 areas *Data Mining*, *Software Engineering*, *Theory*, and *Computer Vision*. 200 key terms are also extracted. DBLP4 and DBLP9 have different cluster structures: the clusters in DBLP4 are more well-separated than those in DBLP9.

**Table 1** The dimensions and the number of classes of real-world datasets

Data	Mode 1	Mode 2	Mode 3	Classes
DBLP4	100	200	20	4
DBLP9	1000	1000	20	9
Enron	184	184	44	4

**Table 2** The descriptions of 3 synthetic datasets

Data	Mode 1	Mode 2	Mode 3	Noise 2	Noise 3	Cln
Syn1	200	20	10	0	0	5
Syn2	50	100	10	50	5	3
Syn3	1000	200	30	100	10	6

**Enron Dataset:** The Enron email dataset containing email communications among 184 users is obtained from [28]. We remove the emails before 13-Nov-1998 and after 21-Jun-2002 and our data is a 184 users × 184 users × 44 months three-way data. The class labels are unknown. The number of classes is set to 4 according to [3], which defined 4 roles for Enron email senders/receivers.

**Synthetic Datasets:** Three sets of synthetic data are generated for experiments. The purpose of using synthetic data is that the detailed cluster structures are known, hence we can adjust different factors such as noise levels and cluster structures systematically. Synthetic data are generated by using the algorithm proposed by Milligan [12]. The clusters are nonoverlapping and truncated multivariate normal mixtures. Basically the dimension with clustering structures (called cluster dimensions) is created based on a cluster length chosen from a uniform distribution. The mean and standard deviation of the cluster on this dimension can be derived from this cluster length. The dimension without clustering structures (called noise dimensions) is created from a uniform distribution in some generated range. Thus cluster dimensions and noise dimensions are independent from each other.

The brief descriptions of the real-world datasets is listed in Table 1. Table 2 summarizes the characteristics of the synthetic datasets. The first column lists the dataset names. The next three columns list the number of cluster dimensions in mode 1, mode 2, and mode 3. The fifth and sixth columns contain the number of noise dimensions in mode 2 and mode 3. The last column provides the number of clusters.

### 5.2.2 Clustering quality evaluation measures

In order to compare the clustering performance, we use Normalized Mutual Information (NMI), Adjusted Rand Index (ARI), and Accuracy (ACC) as our performance measures.

These measures provide good insights on how the clustering results agree with the true labels.

Normalized Mutual Information measures how clustering results share the information with the ground-truth label [33]. Generally, larger the NMI value, better the clustering quality is. Its value is between [0, 1]. The NMI of the entire clustering solution is computed as:

$$NMI = \frac{\sum_{i,j} (P(i,j) \log_2 \frac{P(i,j)}{P(i)P(j)})}{\sqrt{\sum_i (-P(i) \log_2 P(i)) \sum_j (-P(j) \log_2 P(j))}}, \quad (19)$$

where  $P(i)$  is the probability that an arbitrary data point belongs to cluster  $i$ , and  $P(j)$  is the probability that an arbitrary data point belongs to ground-truth class  $i$ .  $P(i, j)$  is the joint probability that an arbitrary data point belongs to cluster  $i$  and also class  $j$ . NMI is actually the mutual information between clustering and ground-truth class knowledge divided by the maximum value of clustering entropy and class entropy.

The Rand Index is defined as the number of pairs of objects that are both located in the same cluster and the same class, or both in different clusters and different classes, divided by the total number of objects. Adjusted Rand Index (ARI) adjusts Rand Index by setting the value between [0, 1] [13].

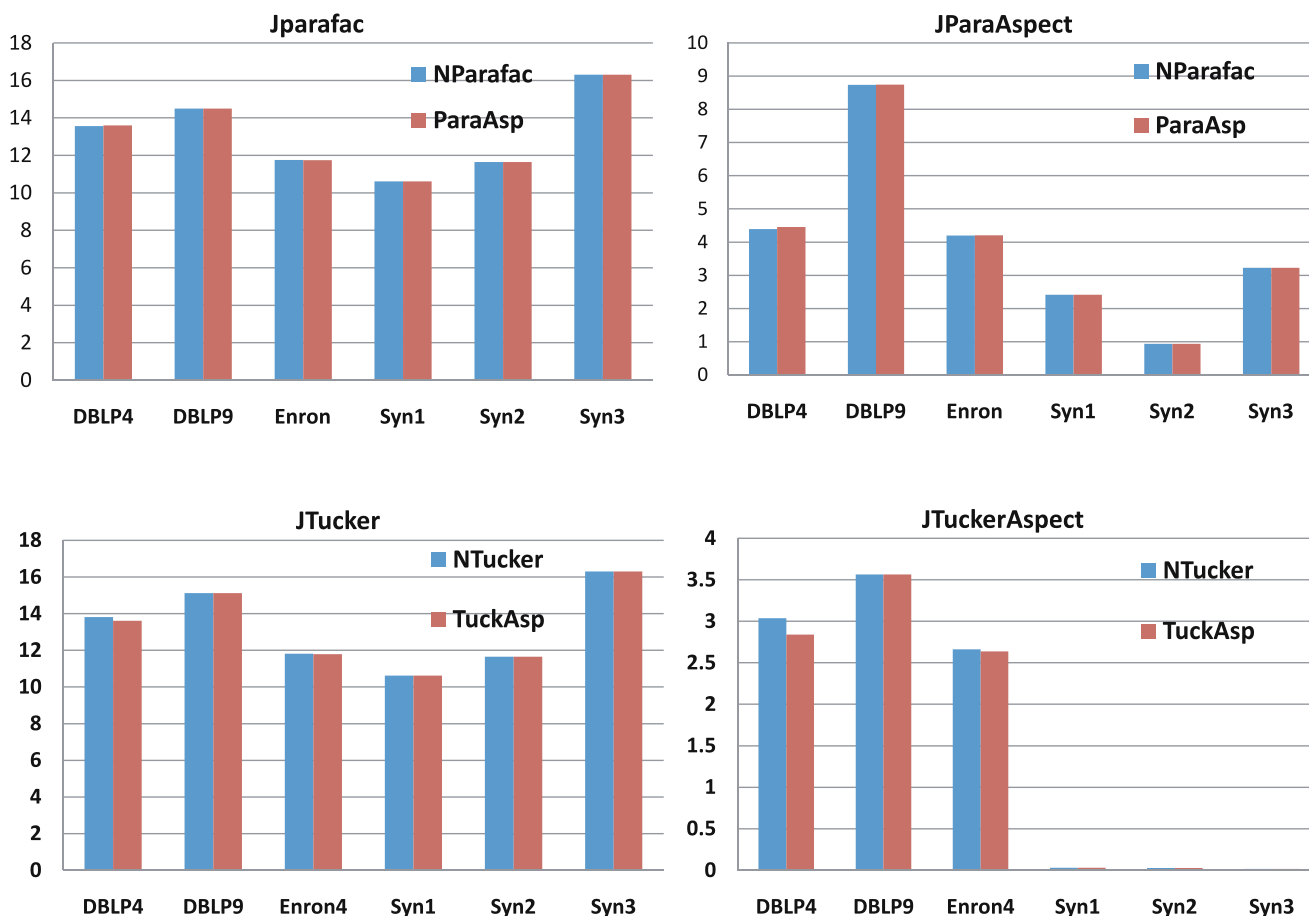
$$ARI = \frac{a - \frac{bc}{n(n-1)/2}}{(1/2)(b+c) - \frac{bc}{n(n-1)/2}}, \quad (20)$$

where  $a = \sum_{i,j} \frac{V_{ij}(V_{ij}-1)}{2}$ ,  $b = \sum_i \frac{V_i(V_i-1)}{2}$ ,  $c = \sum_j \frac{V^j(V^j-1)}{2}$ ,  $V_{ij}$  is the number of objects that are in both the class  $i$  and cluster  $j$ ,  $V_i$  is the number of objects in class  $i$ , and  $V^j$  is the number of objects in cluster  $j$ . Higher the Adjusted Rand Index, more resemblance between the clustering results and the labels is.

Accuracy (ACC) discovers the one-to-one relationship between clusters and classes, and measures the extent to which each cluster contains data points from the corresponding class. It sums up the whole matching degree between all class-cluster pairs. Its value is also between [0, 1]. Accuracy can be represented as:

$$ACC = \max \left( \sum_{C_k, L_m} T(C_k, L_m) \right) / N, \quad (21)$$

where  $C_k$  denotes the  $k$ -th cluster, and  $L_m$  is the  $m$ -th class.  $T(C_k, L_m)$  is the number of entities that belong to class  $m$  are assigned to cluster  $k$ . Accuracy computes the maximum sum of  $T(C_k, L_m)$  for all pairs of clusters and classes, and these pairs have no overlaps. Generally, greater the accuracy, better the clustering performance is.



**Fig. 1** (a) and (b) are comparisons of ParaAspect and non-negative Parafac on  $J_{\text{Parafac}}$  and  $J_{\text{ParaAspect}}$ , and (c) and (d) are comparisons of TuckAspect and non-negative Tucker3 on  $J_{\text{Tucker}}$  and  $J_{\text{TuckAspect}}$  as referenced to (6) and (7)

### 5.3 Result analysis

#### 5.3.1 Objective function and clustering performance comparison

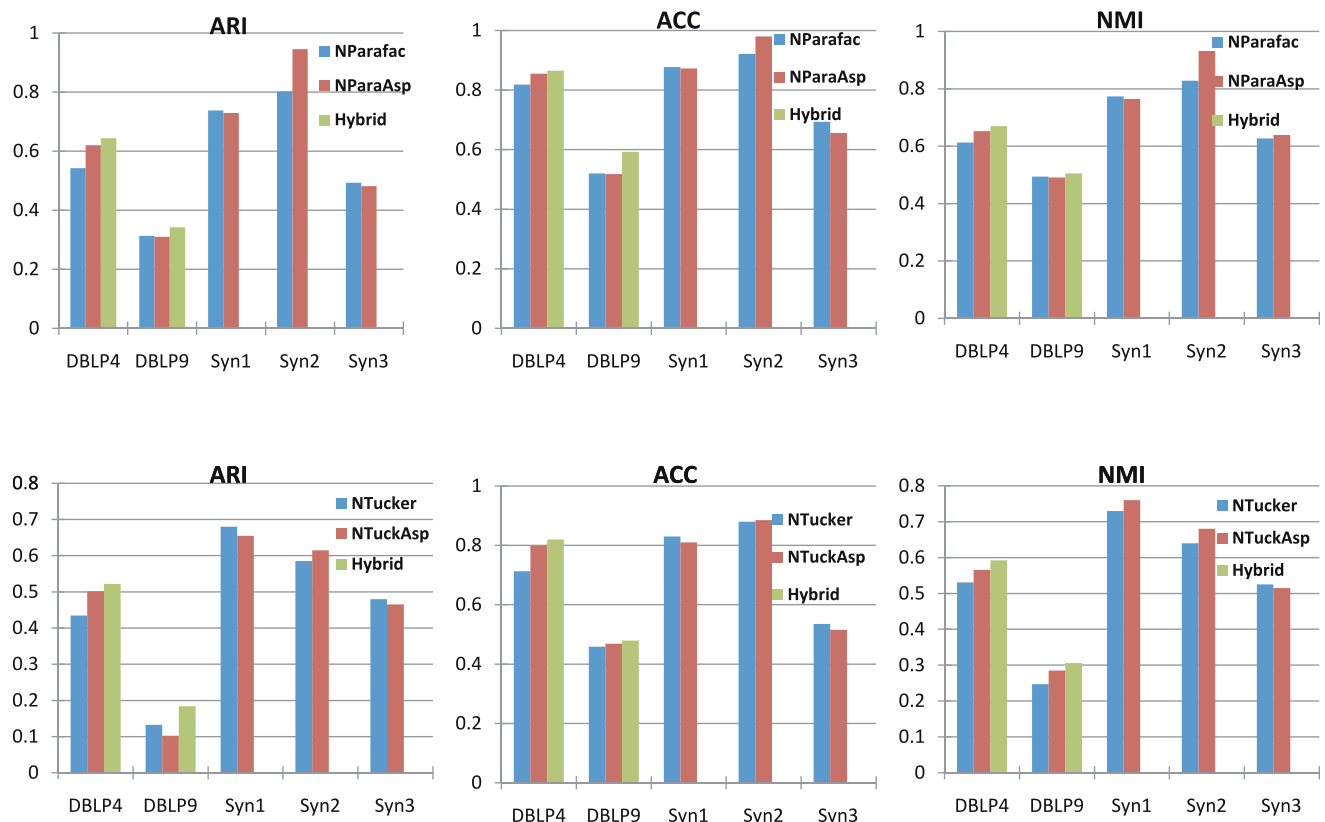
For all synthetic datasets and real-world datasets, NTF (NParafac and NTucker) and T-PLSA (ParaAspect and TuckAspect) are initialized from the same random starting points and are executed until convergence. All experimental results are obtained by averaging 10 trials. The objective functions  $J_{\text{Parafac}}$  (6) and  $J_{\text{ParaAspect}}$  (7) are calculated for both non-negative Parafac and ParaAspect. We observe that they have very similar values (as their difference is less than 0.1) for all datasets as shown in parts (a) and (b) of Fig. 1. Similarly, non-negative Tucker3 and TuckAspect have almost equal values for  $J_{\text{Tucker}}$  (6) and  $J_{\text{TuckAspect}}$  (7) for all datasets as shown in parts (c) and (d) of Fig. 1.

In addition, the clustering results of NTF and T-PLSA are compared in terms of Adjust Rand Index, Accuracy, and NMI on DBLP and synthetic datasets where the class labels are given as shown in Fig. 2. We note that NTF and T-PLSA have similar clustering performance in most cases

because they try to minimize the same objective functions. On the other hand, they also show the different clustering performance values because they use different computational algorithms to optimize the objective functions and thus may end up in different local minima as discussed in Sect. 4.

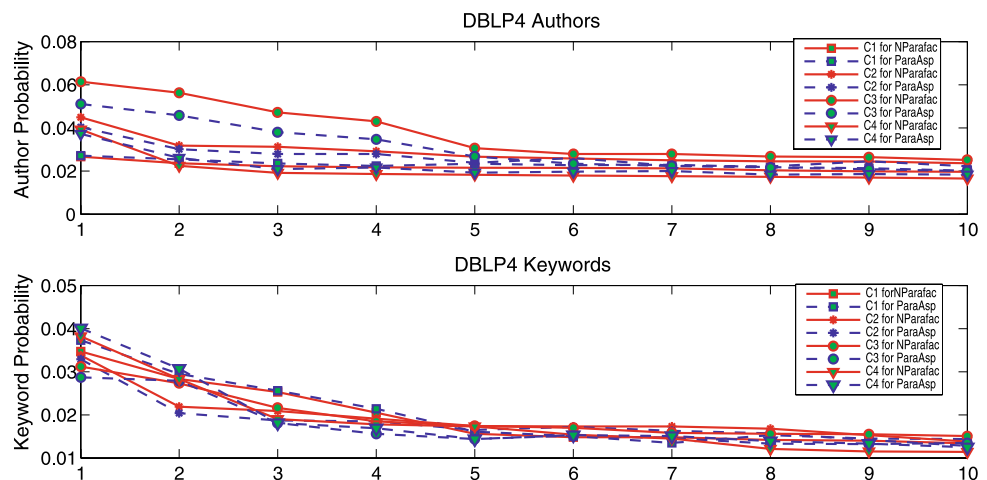
#### 5.3.2 Author and word occurrence probability comparison

In order to obtain a better understanding that NTF and T-PLSA have equivalent objective functions and factorizations, we further compare the top author and word occurrence probabilities of four clusters by running non-negative Parafac and ParaAspect on DBLP4. The results are shown in Fig. 3. In this figure, the X axis are 10 authors (upper figure) and 10 words (lower figure) with the highest occurrence probability  $\hat{u}_{ip}$  and  $\hat{v}_{jp}$  from (10) in 4 clusters. Note that they are illustrated as the declining solid lines along the Y axis. The dotted lines present the probabilities of the same authors and words  $P(d_i|x_p)$ ,  $P(w_j|x_p)$  of the corresponding clusters resulting from the ParaAspect algorithm. The solid lines and the dotted lines with the same symbols are from the corresponding clusters of Parafac and ParaAspect.



**Fig. 2** The clustering performance comparison between ParaAspect, non-negative Parafac, and their hybrid method (the upper row), and between TuckAspect, non-negative Tucker3, and their hybrid method (the lower row)

**Fig. 3** Top authors and words probabilities of 4 clusters by running non-negative Parafac and ParaAspect on DBLP4



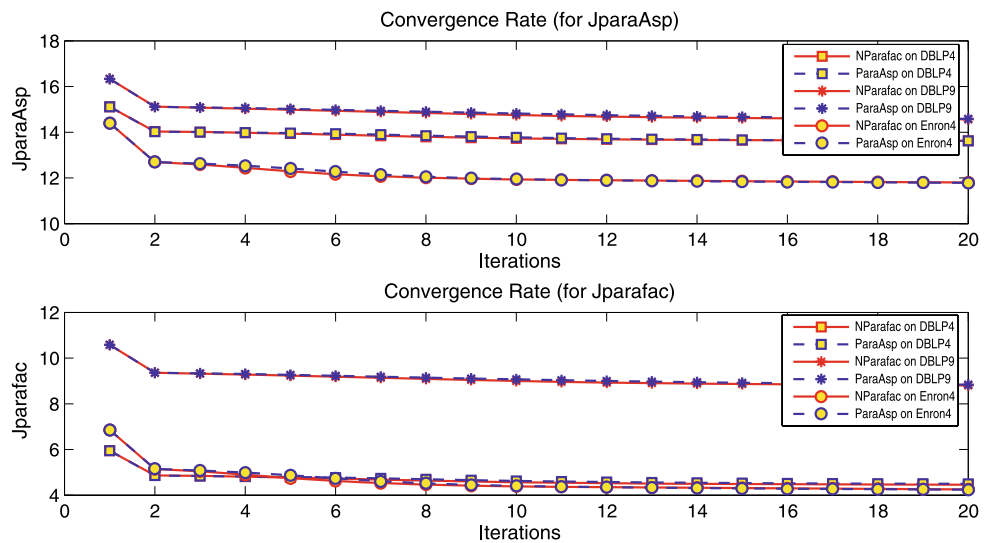
We observe that the top 10 authors and words of each cluster from NParafac are also the top 10 authors and words of clusters from ParaAspect, and every pair of solid curves and dotted curves tightly overlap with each other with very similar slopes. Basically it empirically verifies that  $\hat{u}_{ip} = P(d_i|x_p)$  and  $\hat{v}_{jp} = P(w_j|x_p)$  as shown in Sect. 3.2.

### 5.3.3 Algorithm computation comparison

As shown in Fig. 1, NTF and T-PLSA have similar objective function values upon convergence. Since they adopt differ-

ent algorithms for optimization, one natural question is how their convergence rates are different. We observe that NTF and T-PLSA have similar convergence rate in our experiments as shown in Fig. 4. Figure 4 shows the convergence rate of non-negative Parafac and ParaAspect on  $J_{\text{Parafac}}$  and  $J_{\text{ParaAspect}}$  on 3 datasets. The X axis in both subfigures indicates 20 iterations, the Y axis in the upper subfigure is the objective function value  $J_{\text{ParaAspect}}$ , and the Y axis in the lower subfigure is the objective function value  $J_{\text{Parafac}}$ . Each curve shows the declining objective function values

**Fig. 4** The convergence rate of non-negative Parafac and ParaAspect on  $J_{\text{Parafac}}$  and  $J_{\text{ParaAspect}}$



**Table 3** The time spent on running T-PLSA and NTF for 5 iterations on 6 datasets

	DBLP4	DBLP9	Enron	Syn1	Syn2	Syn3
NParafac	1.59	6.59	0.77	0.026	0.08	7.20
ParaAsp	3.90	25.12	1.85	0.04	0.09	26.28
NTucker	2.16	121.92	0.91	0.04	0.16	8.02
TuckAsp	47.84	1420	15.57	0.53	0.35	642.74

on one dataset, the solid line curves are obtained from running non-negative Parafac, and the dotted line curves are obtained from running ParaAspect. We observe that each pair of the solid line curve and the dotted line curve with the same symbol are largely overlapped. This means NParafac and ParaAspect demonstrate similar convergence rates on the datasets.

Although NTF and T-PLSA show similar computation convergence rates, they have different computational complexity as discussed in Sect. 4. The time (seconds) spent on 5 iterations of running non-negative Parafac, ParaAspect, non-negative Tucker3, and TuckAspect algorithms on DBLP4, DBLP9, Enron, Syn1, Syn2, and Syn3 are listed in Table 3. We can observe that T-PLSA spend more time than their counterparts in NTF. Furthermore, we notice that T-PLSA requires much more time than NTF when the experimental datasets become larger, and that TuckAspect has much higher time complexity than NTucker compared with ParaAspect versus NParafac. Basically, the experimental results are consistent with the discussion in Sect. 4.

#### 5.4 A hybrid algorithm of NTF and T-PLSA

We have seen from the previous sections that NTF and T-PLSA optimize the same objective functions and have equivalent factorization forms as well. However, they have different algorithms so that they would converge into dif-

**Table 4** The clustering performance of the hybrid algorithms on DBLP4 and DBLP9

		DBLP4	DBLP9
ARI	Hybrid(NParafac&ParaAsp)	0.6438	0.3426
	Hybrid(NTucker&TuckAsp)	0.5216	0.1839
Accuracy	Hybrid(NParafac&ParaAsp)	0.8647	0.5924
	Hybrid(NTucker&TuckAsp)	0.8197	0.4793
NMI	Hybrid(NParafac&ParaAsp)	0.6702	0.5051
	Hybrid(NTucker&TuckAsp)	0.5924	0.306

ferent local minima. Inspired by the hybrid NMF-PLSA algorithm proposed in [7], similar hybrid T-PLSA-NTF algorithm can be designed as following: running NTF and T-PLSA iteratively on the solutions of each other to help jump out their local minima until convergence. This hybrid algorithm is guaranteed to converge because T-PLSA and NTF will lower the objective function value and thus the process is monotonic. In the experiments, we observe that the hybrid algorithm can help NTF and T-PLSA jump out of their local minima and achieve a better clustering performance. The clustering performance based on ARI, Accuracy, and NMI of the hybrid algorithms (hybrid of NParafac and ParaAspect and hybrid of NTucker and TuckAspect) on DBLP4 and DBLP9 is presented in Table 4 and Fig. 2. It shows better clustering performance than by only running NTF or T-PLSA indicated in Fig. 2.

## 6 Conclusion

In this paper, non-negative Tensor Factorization and Tensorial Probabilistic Latent Semantic Analysis are related formally for the first time. Two types of T-PLSA models are shown to be equivalent to non-negative PARAFAC

and Tucker3 since their objective functions and factorizations are the same. Extensive experiments are conducted to compare NTF and T-PLSA empirically in terms of objective functions, decomposition results, clustering quality, and computation complexity on both synthetic datasets and real-world datasets. In addition, a hybrid algorithm is proposed by running NTF and T-PLSA alternatively to jump out of each other's local minima to achieve better clustering solutions.

**Acknowledgements** The work is partially supported by NSF grants IIS-0546280, CCF-0830659, and DMS-0915110.

## References

- Acar E, Yener B (2007) Unsupervised multiway data analysis: a literature survey. Technical report, Computer Science Department, Rensselaer Polytechnic Institute
- Smilde A, Bro R, Geladi P (2004) Multi-way analysis: applications in the chemical sciences. Wiley, New York
- Bader BW, Harshman RA, Kolda TG (2007) Temporal analysis of semantic graphs using alsan. In: Proceedings of the ICDM07, pp 33–42, October 2007
- Chi Y, Zhu S, Gong Y, Zhang Y (2008) Probabilistic polyadic factorization and its application to personalized recommendation. In: Proceedings of the 17th ACM conference on information and knowledge management (CIKM 2008), pp 941–950. ACM Press, New York
- Ding C, He X, Simon H (2005) On the equivalence of nonnegative matrix factorization and spectral clustering. In: Proceedings of the SIAM international conference on data mining (SDM 2005)
- Ding C, Li T, Jordan MI (2009) Convex and semi-nonnegative matrix factorizations. *IEEE Trans Pattern Anal Mach Intell* 32(99):1
- Ding C, Li T, Peng W (2008) On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. *Comput Statist Data Anal* 52(8):3913–3927
- Ding C, Li T, Peng W, Park H (2006) Orthogonal nonnegative matrix  $t$ -factorizations for clustering. In: Proceedings of the 12th ACM SIGKDD international conference on knowledge discovery and data mining, pp 126–135
- Duda RO, Hart PE, Stork DG (2000) Pattern classification (2nd edn). Wiley-Interscience, New York
- Evrin Acar MK, Camtepe SA, Yener B (2005) Modeling multi-way analysis of chatroom tensors. *Proc IEEE Int Conf Intell Secur Inform* 3495:256–268
- Gaussier E, Goutte C (2005) Relation between PLSA and NMF and implications. In: Proceeding of the annual international ACM SIGIR conference (SIGIR 2005). ACM Press, New York, pp 601–602
- Milligan GW (1985) An algorithm for generating artificial test clusters. *Psychometrika* 50:123–127
- Milligan GW, Cooper MC (1986) A study of the comparability of external criteria for hierarchical cluster analysis. *Multivar Behav Res* 21:846–850
- Hofmann T (1999) Probabilistic latent semantic indexing. In: Proceedings of the annual international ACM SIGIR conference (SIGIR 1999). ACM Press, New York, pp 50–57
- Hofmann T (2001) Unsupervised learning by probabilistic latent semantic analysis. *Mach Learn* 42(1–2):177–196
- Kolda T (2001) Orthogonal tensor decomposition. *SIAM J Matrix Anal Appl* 23:243–255
- Kolda TG, Bader BW (2006) The tophits model for higher-order web link analysis. In: Workshop on link analysis, counterterrorism and security
- Lee D, Seung HS (1999) Learning the parts of objects by non-negative matrix factorization. *Nature* 401:788–791
- Lee D, Seung HS (2001) Algorithms for non-negative matrix factorization. In: Dietterich TG, Tresp V (eds) *Advances in neural information processing systems*, vol 13. MIT Press, Cambridge
- Lee DD, Seung HS (2000) Algorithms for non-negative matrix factorization. In: NIPS, pp 556–562
- Lee DD, Seung SH (1999) Learning the parts of objects by non-negative matrix factorization. *Nature* 401(6755):788–791
- Li T (2008) Clustering based on matrix approximation: a unifying view. *Knowl Inform Syst J* 17(1):1–15
- Li T, Ding C (2006) The relationships among various nonnegative matrix factorization methods for clustering. In: Proceedings of the 2006 IEEE international conference on data mining (ICDM 2006), pp 362–371
- Pauca VP, Shahnaz F, Berry M, Plemmons R (2004) Text mining using non-negative matrix factorization. In: Proceedings of the SIAM international conference on data mining (SDM 2004), pp 452–456
- Peng W (2009) Equivalence between nonnegative tensor factorization and tensorial probabilistic latent semantic analysis. In: Proceedings of the annual international ACM SIGIR conference (SIGIR 2009), pp 668–669
- Peng W, Li T (2008) Author-topic evolution analysis using three-way non-negative paratucker. In: Proceedings of the annual international ACM SIGIR conference (SIGIR 2008). ACM Press, New York, pp 819–820
- Peng W, Li T, Shao B (2008) Clustering multi-way data via adaptive subspace iteration. In: Proceedings of the 17th ACM conference on information and knowledge management (CIKM 2008). ACM Press, New York, pp 1519–1520
- Priebe C, Conroy J, Marchette D, Park Y (2006) Enron data set. <http://cis.jhu.edu/~parky/Enron/enron.html>
- Harshman RA (1970) Foundations of the parafac procedure: models and conditions for an ‘explanatory’ multi-modal factor analysis. *UCLA Work Pap Phon* 16:1–84
- Schwarz G (1978) Estimating the dimension of a model. *Ann Stat* 6(2):461–464
- Shashanka M, Raj B, Smaragdís P (2008) Probabilistic latent variable models as nonnegative factorizations. *Comput Intell Neurosci* doi:10.1155/2008/947438
- Shashua A, Hazan T (2005) Non-negative tensor factorization with applications to statistics and computer vision. In: Proceedings of the 22nd international conference on machine learning (ICML 2005). ACM Press, New York, pp 792–799
- Strehl A, Ghosh J (2002) Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *J Mach Learn Res* 3:583–617
- Sun J, Zeng H, Liu H, Lu Y, Chen Z (2005) Cubesvd: a novel approach to personalized web search. In: Proceedings of the 14th international conference on World Wide Web, pp 652–662
- Vasilescu MAO, Terzopoulos D (2002) Multilinear analysis of image ensembles: tensorfaces. In: Proceedings of the 7th European conference on computer vision—part I (ECCV’02), pp 447–460
- Vichi M, Rocci R, Kiers HAL (2007) Simultaneous component and clustering models for three-way data: within and between approaches. *J Classif* 24(1):71–98
- Wang H, Ahuja N (2005) Rank- $r$  approximation of tensors: using image-as-matrix representation. In: CVPR ’05, vol 2, pp 346–353
- Xu W, Liu X, Gong Y (2003) Document clustering based on non-negative matrix factorization. In: Proceedings of ACM conference of research and development in IR (SIGIR), pp 267–273, Toronto, Canada
- Zhang T, Golub G (2001) Rank-one approximation to high order tensor. *SIAM J Matrix Anal Appl* 23:534–550