nTD: Noise Adaptive Tensor Decomposition

Xinsheng Li
Arizona State University
Tempe, AZ, 85282, USA
lxinshen@asu.edu

K. Selçuk Candan
Ariona State University
Tempe, AZ, 85282, USA
candan@asu.edu

Maria Luisa Sapino
University of Torino
I-10149 Torino, Italy
marialuisa.sapino@unito.it

ABSTRACT

Tensor decomposition is increasingly being used for many data analysis operations from clustering, trend detection, anomaly detection, to correlation analysis. However, as we argue in the paper, many of the tensor decomposition schemes are sensitive to noisy data, an inevitable problem in the real world that can lead to false conclusions. The problem is compounded by over-fitting when the data is sparse. Recent research has shown that it is possible to avoid over-fitting by relying on probabilistic techniques. However, these have two major deficiencies: (a) firstly, they assume that all the data and intermediary results can fit in the main memory, and (b) they treat the entire tensor uniformly, ignoring potential non-uniformities in the noise distribution. To deal with these challenges, in this paper, we propose a *Noise Adaptive Tensor Decomposition (*`nTD`*)* method, which aims to tackle both of these challenges: in particular, `nTD` leverages a grid-based two-phase decomposition strategy for two complementary purposes: firstly, the partitioning helps ensure that the memory footprint of the decomposition is kept low; secondly (and more importantly) the noise profiles of the grid partitions enable us to develop a sample assignment strategy (or `s-strategy`) that best suits the noise distribution of the given tensor, leading to *nTD* method, which can leverage available *rough* knowledge regarding where in the tensor noise might be more prevalent. Experiments show that `nTD`'s performance is significantly better than conventional ALS-based CP decomposition on noisy tensors.

1. INTRODUCTION

Tensors are multi-dimensional arrays and are commonly used for representing multi-dimensional data, such as sensor streams and social networks [17, 31]. Thanks to the widespread availability of multi-dimensional data in many applications domains, tensor decomposition operations (such as CP [9] and Tucker [27]) are increasingly being used to implement various data analysis tasks, from clustering, anomaly detection [17], correlation analysis [24], to pattern discovery [12].

Yet, tensor decomposition process is subject to several major challenges: One major challenge with tensor decomposition is its computational complexity: decomposition algorithms have high computational costs and, in particular, incur large memory overheads (also known as the *intermediary data blow-up problem*) and, thus, basic algorithms and naive implementations are not suitable for large problems. Recently distributed/parallel implementations, such as GridParafac [21], GigaTensor [13], HaTen2 [11], TensorDB [15, 19], were proposed to deal with the high computational cost of the task. In addition the tensor decomposition process can be negatively affected from noise in the data – in particular, especially for sparse data avoiding over-fitting to the noisy data can be a significant challenge.

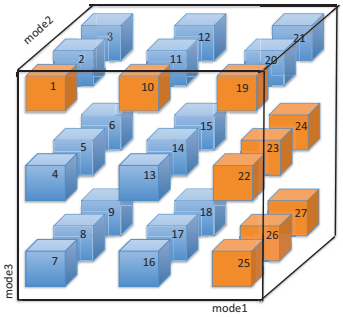


Figure 1: A sample 3-mode tensor, partitioned into a grid of sub-tensors; the figure also highlights a subset of the sub-tensor which are noisy

Recent research has shown that it is possible to avoid such over-fitting by relying on probabilistic techniques [28]. Unfortunately, existing probabilistic approaches have two major deficiencies: (a) firstly, they assume that all the data and intermediary results can fit in the main memory and (b) they treat the entire tensor uniformly, ignoring possible non-uniformities in the distribution of noise in the given tensor.

To deal with the first challenge, in this paper, we propose a *Noise Adaptive Tensor Decomposition (*`nTD`*)* method: `nTD` partitions the tensor into multiple sub-tensors and then decomposes each sub-tensor probabilistically through Bayesian factorization – the resulting decompositions are then recombined through an iterative refinement process to obtain the decomposition for the whole tensor. Simultaneously `nTD` develops a resource allocation strategy that accounts for the impact of the noise density of one sub-tensor on the decomposition accuracies of the other sub-tensors:

This provides several benefits:

- Firstly, the partitioning helps ensure that the memory footprint of the decomposition is kept low.

ACM ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

- Secondly, the probabilistic framework used in the first phase ensures that the decomposition is robust to the presence of noise in the sub-tensors.
- Thirdly,*a priori* knowledge about noise distribution among the sub-tensors (Figure 1) is used to obtain a resource assignment strategy that best suits the noise profile of the given tensor.

Experiments show that `nTD`'s accuracy and memory consumption are significantly better than conventional ALS based CP decomposition on noisy tensors.

1.1 Organization of this Paper

This paper is organized as follows: In the next section, we present the related work.Section 3 presents the relevant notations and the background. Section 4 describes the overview of the grid based probabilistic tensor decomposition (`GPTD`) scheme. Section 5 introduces the proposed sample assignment strategy (*s-strategy*) to adapt `GPTD` to different noise profiles, leading to a novel noise adaptive tensor decomposition (`nTD`) approach.Section 6 experimentally evaluates the effectiveness of the `nTD` and its alternative implementations. Experiments show that `nTD` indeed improves the decomposition accuracy of noise polluted tensors and the proposed sample assignment strategy (*s-strategy*) helps optimize the `nTD` performance under different noise scenarios. We conclude the paper in Section 7.

2. RELATED WORK

Tensor based representations of data and tensor decompositions (especially the two widely used decompositions CP [9] and Tucker [27]) are proven to be effective in multi-aspect data analysis for capturing high-order structures in multi-dimensional data. In [16], for example, the authors incorporate contextual information to the traditional HITS algorithm, formulating the task as tensor decomposition. In [1], authors introduce a tensor-based framework to identify epileptic seizures and, in [25], authors use tensors to incorporate user click information to improve web search. [8] shows that tensor decomposition of fMRI data can help in differentiating healthy and Alzheimer affected individuals.

There are two widely used toolboxes for tensor manipulation: the *Tensor Toolbox for Matlab* [3] (for sparse tensors) and *N-way Toolbox for Matlab*[2] (for dense tensors). Yet, due to the significant cost [17] of tensor decompositions, various parallel algorithms and systems have been developed. [26] proposes MACH, a randomized algorithm that speedups the Tucker decomposition while providing accuracy guarantees. More recently, [13] proposed GigaTensor, a massively distributed Map-Reduce based implementation of CP decomposition (also known as PARAFAC). Authors introduce a series of optimizations, which (in combination with the use of the Map-Reduce environment) lead to a highly scalable PARAFAC decomposition platform. In [20], authors propose PARCUBE, a sampling based, parallel and sparsity promoting, approximate PARAFAC decomposition scheme. Scalability is achieved through sketching of the tensor (using biased sampling) and parallelization of the decomposition operations onto the resulting sketches. TensorDB [15, 19] leverages a block-based framework to store and retrieve data, extends array operations to tensor operation, and introduces optimization schemes for in-database tensor decomposition. HaTen2 [11] focuses on sparse tensors and presents a scalable tensor decomposition suite of methods for Tucker and PARAFAC decompositions on a MapReduce framework. SCOUT [12] is a recent coupled matrix-tensor factorization framework, also built on MapReduce. In addition to parallelism, it also leverages computation reordering as well as data transformation and reuse to reduce the computational cost of the process.

In [7], authors develop a probabilistic framework, pTucker, for modeling structural dependency from partially observed multi-dimensional arrays. [30] implements a deterministic Bayesian inference algorithm, which formulates CP factorization with a hierarchical probabilistic model and employs Bayesian treatment by incorporating a sparsity-inducing prior over multiple latent factors and the appropriate hyper-priors over all hyperparameters, resulting in automatic rank determination. [22] proposed a Bayesian framework for low-rank decomposition of multiway missing observations tensor data. The method helps with the discovery of the decomposition rank from the data; moreover, inference scales linearly with the observation size, which helps the proposed approach scale very well. [6] proposes a loss function that helps the tensor decomposition process handle both Gaussian and grossly non-Gaussian perturbations.

3. BACKGROUND AND NOTATIONS

We now present the relevant background and notations.

3.1 Tensors and Tensor Decompositions

Tensors are generalizations of matrices: while a matrix is essentially a 2-mode array, a tensor is an array of larger number of modes. Intuitively, the tensor model maps a relational schema with N attributes to an N-modal array (where each potential tuple is a tensor cell).

Tensor decomposition process generalizes the matrix decomposition process to tensors and rewrites the given tensor in the form of a set of *factor* matrices (one for each mode of the input tensor) and a core matrix (which, intuitively, describes the spectral structure of the given tensor). The two most popular tensor decomposition algorithms are the Tucker [27] and the CANDECOMP/PARAFAC (CP) [9] decompositions. While CP decomposes the input tensor into a sum of component rank-one tensors (leading into a diagonal core), Tucker decomposition results in a dense core. In this paper, we focus on the CP decomposition process.

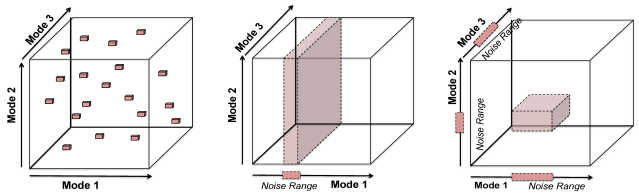
3.2 CP Decomposition

Given a tensor $\boldsymbol{\mathcal{X}}$, CP factorizes the tensor into factor matrices with F rows (where F is a user supplied non-zero integer value also referred to as the *rank* of the decomposition). For the simplicity of the discussion, let us consider a 3-mode tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I \times J \times K}$. CP would decompose $\boldsymbol{\mathcal{X}}$ into three matrices $\mathbf{A}, \mathbf{B}$, and $\mathbf{C}$, such that

$$\boldsymbol{\mathcal{X}} \approx \tilde{\boldsymbol{\mathcal{X}}} = [\mathbf{A}, \mathbf{B}, \mathbf{C}] \equiv \sum_{f=1}^{F} a_f \circ b_f \circ c_f,$$

where $a_f \in \mathbb{R}^I$, $b_f \in \mathbb{R}^J$ and $c_f \in \mathbb{R}^K$. The factor matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ are the combinations of the rank-one component vectors into matrices; e.g., $\mathbf{A} = [\, a_1 \; a_2 \cdots \; a_F \,]$. Since tensors may not always be exactly decomposed, the new tensor $\tilde{\boldsymbol{\mathcal{X}}}$ obtained by recomposing the factor matrices $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ is often different from the input tensor, $\boldsymbol{\mathcal{X}}$. The accuracy of the decomposition is often measured by considering the Frobenius norm of the difference tensor:

$$accuracy(\boldsymbol{\mathcal{X}}, \tilde{\boldsymbol{\mathcal{X}}}) = 1 - error(\boldsymbol{\mathcal{X}}, \tilde{\boldsymbol{\mathcal{X}}}) = 1 - \left(\frac{\|\tilde{\boldsymbol{\mathcal{X}}} - \boldsymbol{\mathcal{X}}\|}{\|\boldsymbol{\mathcal{X}}\|} \right).$$



(a) uni noise (b) sc noise (b) mm noise

Figure 2: Alternative noise profiles of a tensor

Alternating least squares (ALS) is the most conventional method for tensor decomposition: at each iteration, ALS estimates one factor matrix while maintaining other matrices fixed; this process is repeated for each factor matrix associated to the modes of the input tensor until convergence condition is reached.

3.3 Noise Profiles of Tensors

In a given tensor, noise can be distributed in several ways.

- In *uniform (uni) noise* (Figure 2(a)), there is no underlying pattern and noise is not clustered across any slice or region of the tensor.
- *Slice-concentrated (sc)* noise (Figure 2(b)) is clustered on one or more slices on the tensor across one or more modes. This type of noise is common in the real-world, where, for example, a sensor is mis-calibrated or damaged. The sensor may still be recording, but all observations involving that sensor are inherently noisy and these observations would be concentrated on a slice of the tensor corresponding to the faulty sensor.
- In *multi-modal (mm)* noise, again, the noise is clustered; however, in this case the noise is expected to occur when a combination of a subset of the values across two or more modes are considered together as in Figure 2(c). This may, for example, occur if a certain type of sensors (described by one mode in the tensor) are known to provide a noisy reading in certain types of environments (described by another mode of the same tensor).

In addition to the above, noise may impact the observed values in different ways: in *value-independent noise*, an existing value may be overwritten by a completely random new value, whereas in *value-independent noise* existing values may be perturbed (often with a Gaussian noise – defined by its *strength*, or standard deviation, σ). When we consider noise profiles, another relevant parameter is the *noise density*, which can be defined as the ratio of the cells that are subject to noise[1]. In general, a tensor may be subject to different sources of noise simultaneously.

Note that, during the decomposition process, information about the nature of the noise may or may not be available. In the next section, we propose a decomposition technique that does not assume the availability of any *a priori* knowledge about noise distribution, which is basic technique of nTD, *Grid Based Probabilistic Tensor Decomposition* (`GPTD`) . Then, in Section 5, we extend this to scenarios where we have very rough information (more specifically, the relative noise density of the sub-tensors) about the noise distribution and develop a noise adaptive decomposition (`nTD`) technique that leverages this information to improve decomposition accuracy.

[1]In this paper, without loss of generality, we assume noise are on cells that already have values (i.e., the observed values be faulty, but there are no spurious observations) and, thus, we define noise density as the ratio of non-null cells.

Algorithm 1 Phase 1: Monte Carlo based Bayesian decomposition of each sub-tensor

Input: Sub-tensor $\boldsymbol{\mathcal{X}}_{\vec{k}}$, sampling number L

Output: Decomposed factors $\boldsymbol{U}^{(1)}_{\vec{k}}$, $\boldsymbol{U}^{(2)}_{\vec{k}}$, ..., $\boldsymbol{U}^{(N)}_{\vec{k}}$

1. Initialize model parameters $\boldsymbol{U}^{(1)1}_{\vec{k}}$, $\boldsymbol{U}^{(2)1}_{\vec{k}}$, ..., $\boldsymbol{U}^{(N)1}_{\vec{k}}$.
2. For $l = 1, \ldots, L$
   (a) Sample the hyper-parameter, α:
   - $\alpha^l \sim p(\alpha^l | \boldsymbol{U}^{(1)l}_{\vec{k}}, \boldsymbol{U}^{(2)l}_{\vec{k}}, \ldots, \boldsymbol{U}^{(N)l}_{\vec{k}}, \boldsymbol{\mathcal{X}}_{\vec{k}})$

   (b) For each mode $j = 1, \ldots, N$,
   i. Sample the corresponding hyper-parameter, Θ:
   - $\Theta_{\boldsymbol{U}^{(j)l}_{\vec{k}}} \sim p(\Theta_{\boldsymbol{U}^{(j)l}_{\vec{k}}} | \boldsymbol{U}^{(j)l}_{\vec{k}})$

   ii. For $i_j = 1, ..., I_j$, sample the mode (in parallel):

$$\boldsymbol{U}^{(j)(l+1)}_{\vec{k}(i_j)} \sim p\Big(\boldsymbol{U}^{(j)}_{\vec{k}(i_j)} \Big| \boldsymbol{U}^{(1)l}_{\vec{k}}, \ldots, \boldsymbol{U}^{(j-1)l}_{\vec{k}}, \boldsymbol{U}^{(j+1)l}_{\vec{k}}, \ldots, \boldsymbol{U}^{(N)l}_{\vec{k}}, \Theta^l_{\boldsymbol{U}^{(j)}_{\vec{k}}}, \alpha^l, \boldsymbol{\mathcal{X}}_{\vec{k}}\Big)$$

3. For each mode $j = 1, \ldots, N$,
   - $\boldsymbol{U}^{(j)}_{\vec{k}} = \frac{\sum_{i=1}^{L} \boldsymbol{U}^{(j)i}_{\vec{k}}}{L}$

4. GRID BASED PROBABILISTIC TENSOR DECOMPOSITION (GPTD)

In the real world, information of about the nature of the noise may not be provided. *Grid Based Probabilistic Tensor Decomposition* (`GPTD`), basic technique of (`nTD`,is proposed without any priori knowledge about noise distribution. `GPTD` techniques firstly partition the given tensor into blocks, initially decompose each block independently, and then iteratively combine these decompositions into a final composition. We firstly discuss the sub-tensor decomposition process next.

4.1 Phase 1: Monte Carlo based Bayesian Decomposition of Sub-tensors

As we discussed in the introduction, in this paper, we propose to leverage probabilistic tensor factorization, an extension of the probabilistic matrix factorization, to better deal with over-fitting, which is a challenge especially when the data is noisy.

Intuitively, each entry in the factor matrices are modeled as probabilistic variables and decomposition is posed as a maximization problem where these (latent) random variables fit the observed data. In the presence of noise in the data, the observed variables may also be modeled probabilistically: since the observations cannot be precisely described, they may be considered as samples from a probability distribution. In this paper, following [23], in the presence of data uncertainty (due to noise), we describe the fit between the observed data and the predicted latent factor matrices, probabilistically, as follows:

$$\boldsymbol{\mathcal{X}}_{\vec{k}(i_1,i_2,\ldots,i_N)} \Big| \boldsymbol{U}^{(1)}_{\vec{k}}, \boldsymbol{U}^{(2)}_{\vec{k}} \ldots, \boldsymbol{U}^{(N)}_{\vec{k}} \sim \boldsymbol{\mathcal{N}}([\boldsymbol{U}^{(1)}_{\vec{k}(i_1)}, \boldsymbol{U}^{(2)}_{\vec{k}(i_2)} \ldots, \boldsymbol{U}^{(N)}_{\vec{k}(i_N)}], \alpha^{-1}), \quad (1)$$

where the conditional distribution of $\boldsymbol{\mathcal{X}}_{\vec{k}(i_1,i_2,\ldots,i_N)}$ given $\boldsymbol{U}^{(j)}_{\vec{k}}$ $(1 \leq j \leq N)$ is a Gaussian distribution with mean $[\boldsymbol{U}^{(1)}_{\vec{k}(i_1)}, \boldsymbol{U}^{(2)}_{\vec{k}(i_2)}, \ldots, \boldsymbol{U}^{(N)}_{\vec{k}(i_N)}]$ and the observation precision α. We also impose independent Gaussian priors on the modes:

$$\boldsymbol{U}^{(j)}_{\vec{k}(i_j)} \sim \boldsymbol{\mathcal{N}}(\mu_{\boldsymbol{U}^{(j)}_{\vec{k}}}, \Lambda^{-1}_{\boldsymbol{U}^{(j)}_{\vec{k}}}) \quad i_j = 1 \ldots I_j \tag{2}$$

where I_j is the dimensionality of the j^{th} mode. Given this, one can estimate the latent features $\boldsymbol{U}^{(j)}_{\vec{k}}$ by maximizing the logarithm of the posterior distribution, $\log p(\boldsymbol{U}^{(1)}_{\vec{k}}, \boldsymbol{U}^{(2)}_{\vec{k}} \ldots, \boldsymbol{U}^{(N)}_{\vec{k}} | \boldsymbol{\mathcal{X}}_{\vec{k}})$. One difficulty with the approach, however, is the tuning of the hyper-parameters of the model: α and $\Theta_{\boldsymbol{U}^{(j)}_{\vec{k}}} \equiv \{\mu_{\boldsymbol{U}^{(j)}_{\vec{k}}}, \Lambda_{\boldsymbol{U}^{(j)}_{\vec{k}}}\}$ for $1 \leq j \leq N$. [28] notes that one can avoid the difficulty underlying the estimation of these parameters through a fully Bayesian approach , complemented with a sampling-based Markov Chain Monte Carlo (MCMC) method to address the lack of the analytical solution.

The process is visualized in Algorithm 1 in pseudo-code form.

4.2 Phase 2: Iterative Refinement

Once the individual sub-tensors are decomposed, the next step is to stitch the resulting sub-factors into the full F-rank factors, $\boldsymbol{A}^{(i)}$ (each one along one mode), for the input tensor, $\boldsymbol{\mathcal{X}}$. Let us partition each factor $\boldsymbol{A}^{(i)}$ into K_i parts corresponding to the block boundaries along mode i:

$$\boldsymbol{A}^{(i)} = [\boldsymbol{A}^{(i)T}_{(1)} \boldsymbol{A}^{(i)T}_{(2)} \ldots \boldsymbol{A}^{(i)T}_{(K_i)}]^T.$$

Given this partitioning, each sub-tensor $\boldsymbol{\mathcal{X}}_{\vec{k}}$, $\vec{k} = [k_1, \ldots, k_i, \ldots, k_N] \in \mathcal{K}$ can be described in terms of these sub-factors:

$$\boldsymbol{\mathcal{X}}_{\vec{k}} \approx \boldsymbol{I} \times_1 \boldsymbol{A}^{(1)}_{(k_1)} \times_2 \boldsymbol{A}^{(2)}_{(k_2)} \cdots \times_N \boldsymbol{A}^{(N)}_{(k_N)} \tag{3}$$

The current estimate of the sub-factor $\boldsymbol{A}^{(i)}_{(k_i)}$ can be revised using the following update rule [21]:

$$\boldsymbol{A}^{(i)}_{(k_i)} \longleftarrow \quad \boldsymbol{T}^{(i)}_{(k_i)} \left(\boldsymbol{S}^{(i)}_{(k_i)}\right)^{-1} \tag{4}$$

where

$$\boldsymbol{T}^{(i)}_{(k_i)} = \sum_{\vec{m} \in \{[*,\ldots,*,k_i,*,\ldots,*]\}} \boldsymbol{U}^{(i)}_{\vec{m}} \left(\boldsymbol{P}_{\vec{m}} \oslash (\boldsymbol{U}^{(i)T}_{\vec{m}} \boldsymbol{A}^{(i)}_{(k_i)})\right)$$

$$\boldsymbol{S}^{(i)}_{(k_i)} = \sum_{\vec{m} \in \{[*,\ldots,*,k_i,*,\ldots,*]\}} \boldsymbol{Q}_{\vec{m}} \oslash \left(\boldsymbol{A}^{(i)T}_{(k_i)} \boldsymbol{A}^{(i)}_{(k_i)}\right)$$

such that, given $\vec{m} = [m_1, m_2, \ldots, m_N]$, we have

- $\boldsymbol{P}_{\vec{m}} = \circledast_{h=1}^{N} (\boldsymbol{U}^{(h)T}_{\vec{m}} \boldsymbol{A}^{(h)}_{(m_h)})$ and
- $\boldsymbol{Q}_{\vec{m}} = \circledast_{h=1}^{N} (\boldsymbol{A}^{(h)T}_{(m_h)} \boldsymbol{A}^{(h)}_{(m_h)})$.

Above, $\circledast$ denotes the Hadamart product and $\oslash$ denotes the element-wise division operation.

4.3 Overview of GPTD

The two phases of the decomposition process are visualized in Algorithm 2 and Figure 3. Note that, in the second phase of the process, each $\boldsymbol{A}^{(i)}_{(k_i)}$ is maintained incrementally by using, for all $1 \leq j \leq N$, (a) the current estimates for $\boldsymbol{A}^{(j)}_{(k_j)}$ and (b) the decompositions in $\mathfrak{U}^{(j)}$; i.e., the F-rank sub-factors of the sub-tensors in $\mathfrak{X}$ along the different modes of the tensor. This implies that a sub-tensor which is poorly decomposed due to noise may negatively impact decomposition accuracies also for other parts of the tensor. Consequently, it is important to properly allocate resources to prevent a few noisy sub-tensors among all from negatively impacting the accuracy of the whole decomposition – which is discussed next.

Algorithm 2 The outline of the GPTD process

Input: Input tensor $\boldsymbol{\mathcal{X}}$, partitioning pattern $\mathcal{K}$, and decomposition rank, F, and per sub-tensor sample count, L

Output: Tensor decomposition $\mathring{\boldsymbol{\mathcal{X}}}$

1. Phase 1: for all $\vec{k} \in \mathcal{K}$
   - decompose $\boldsymbol{\mathcal{X}}_{\vec{k}}$ into $\boldsymbol{U}^{(1)}_{\vec{k}}$, $\boldsymbol{U}^{(2)}_{\vec{k}}$, …, $\boldsymbol{U}^{(N)}_{\vec{k}}$

   with sample count L using Algorithm 1.
2. Phase 2: repeat
   (a) for each mode $i = 1$ to N
      i. for each modal partition, $k_i = 1$ to K_i,
         A. update $\boldsymbol{A}^{(i)}_{(k_i)}$ using $\boldsymbol{U}^{(i)}_{[*,\ldots,*,k_i,*,\ldots,*]}$, for each block $\boldsymbol{\mathcal{X}}_{[*,\ldots,*,k_i,*,\ldots,*]}$; more specifically,
            - compute $\boldsymbol{T}^{(i)}_{(k_i)}$, which involves the use of $\boldsymbol{U}^{(i)}_{[*,\ldots,*,k_i,*,\ldots,*]}$ (i.e. the mode-i factors of $\boldsymbol{\mathcal{X}}_{[*,\ldots,*,k_i,*,\ldots,*]}$)
            - revise $\boldsymbol{P}_{[*,\ldots,*,k_i,*,\ldots,*]}$ using $\boldsymbol{U}^{(i)}_{[*,\ldots,*,k_i,*,\ldots,*]}$ and $\boldsymbol{A}^{(i)}_{(k_i)}$
            - compute $\boldsymbol{S}^{(i)}_{(k_i)}$ using the above
            - update $\boldsymbol{A}^{(i)}_{(k_i)}$ using the above
            - for each $\vec{k} = [*,\ldots,*,k_i,*,\ldots,*]$
              – update $\boldsymbol{P}_{\vec{k}}$ and $\boldsymbol{Q}_{\vec{k}}$ using
              – $\boldsymbol{U}^{(i)}_{\vec{k}}$ and $\boldsymbol{A}^{(i)}_{(k_i)}$

   until stopping condition
3. Return $\mathring{\boldsymbol{\mathcal{X}}}$

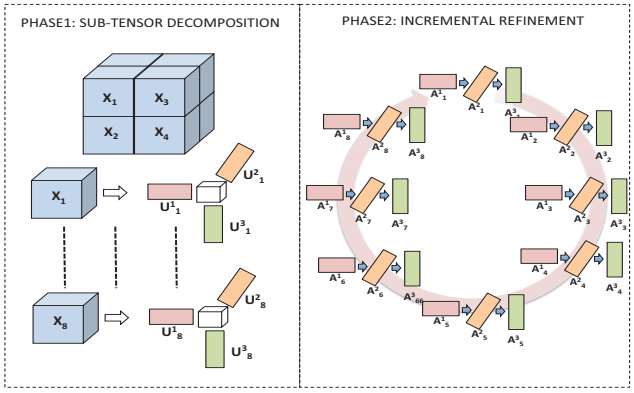


Figure 3: Illustration of sub-tensor based tensor decomposition: the input tensor is partitioned into smaller blocks, each block is decomposed (potentially in parallel), and the partial decompositions are stitched together through an iterative improvement process

5. NOISE ADAPTIVE TENSOR DECOMPOSITION

One crucial piece of information that the basic GPTD process fails to account for is the potential *a priori knowledge* about the distribution of the noise across the input tensor. In the real world, noise is rarely *uniformly* distributed. More often, we would expect that noise would be clustered across slices of the tensor (corresponding, for example, to unreliable sensors) or be clustered in a multi-modal fashion as discussed in Section 3.3. In many cases, even if we do not have precise knowledge about the cells that are subject to noise or the amount of noise they contain, we may have a rough idea about the distribution of noise across the grid partitions. In this section, we aim to answer the question "*can we leverage rough information that may be available about noise distribution in tensor decomposition?*".

This leads to a novel noise adaptive grid based probabilistic tensor decomposition (nTD) scheme. Remember from Section 4.2 that a sub-tensor which is poorly decomposed due to noise may negatively impact decomposition accuracies also for other parts of the tensor. Consequently, it is important to allocate resources in a way to prevent noisy sub-tensors from negatively impacting the decomposition accuracy for the whole decomposition. In this section, we note that grid partitioning can also provide opportunities for taking into account the noise profile of the input tensor. More specifically, rough knowledge about the noise profiles of the grid partitions enables us to develop a sample assignment strategy (or s-strategy) that best suits the noise distribution in a given tensor. In particular, nTD assigns the ranks and samples to different sub-tensors in a way that maximizes the overall decomposition accuracy of the whole tensor without negatively impacting the efficiency of the decomposition process. Since probabilistic decomposition can be costly, nTD algorithm considers *a priori* knowledge about each sub-tensor's noise density to decide, for each sub-tensor, the appropriate number of Gibbs samples to achieve good accuracy with the given number of samples.

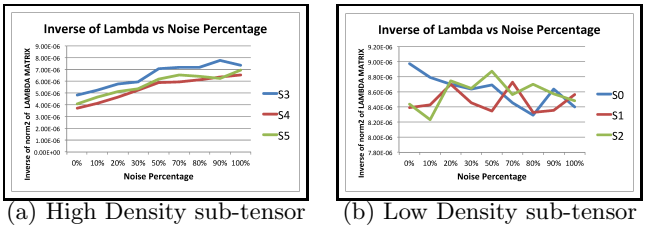
5.1 Does density affect?

Density is a critical parameter in the area of data analysis. Intuitively, It should be hard to learn the noise profile from the sparse data. As Eq. 2 stated, parameter Λ^{-1} indicates the variance of data. Λ^{-1} should have certain correlation with noise distribution. However, if the data is too sparse, while lower than certain threshold, there is no difference between real value and noise, which leads to the pattern of Λ^{-1} should be hard to learn. As shown in Fig. 4, 4, S0,S1,S2,S3,S4,S5 are sub-tensors from different region of MovieLen data, which have different densities. S0, S1, S2 density is lower than 1.0×10^{-5}, S3,S4,S5 are sub-tensors, which density is higher than 1.0×10^{-5}. So we can find out, in the fig. 4 (a), with increasing of each high density sub-tensor's noise percentage, Λ^{-1} is almost linearly increasing. However, in the fig. 4 (b), no pattern can be allocated in lower density sub-tensors.

So, intuitively, while the tensor density is low, profile of noise can't provide enough information to leverage. Instead, density is better candidate to be utilized to improve the performance of decomposition. In the section 6, experiments also verify this observation.

5.2 Noise Sensitive Sample Assignment: First Attempt

âĽd' As we experimentally show in Section 6, there is a direct relationship between the amount of noise a (sub-)tensor has and the number of Gibbs samples it requires for accurate decomposition. On the other hand, the number of samples also directly impacts the cost of the probabilistic decomposition process. Consequently, given a set of sub-tensors, with different amounts of noise, uniform assignment of the number of samples, $L = \left(\frac{L_{(total)}}{|\mathcal{K}|}\right)$, where $L_{(total)}$ is the total number of samples for the whole tensor and $|\mathcal{K}|$ is the number of sub-tensors, may not be the best choice.



(a) High Density sub-tensor (b) Low Density sub-tensor

Figure 4: Comparison of density influences about the noise profile learning

In fact, the numbers of Gibbs samples allocated to different sub-tensors $\boldsymbol{\mathcal{X}}_{\vec{k}}$ in Algorithm 1 *do not need to be the same.* As we have seen in Section 4.1, Phase 1 decomposition of each sub-tensor is independent from the others and, thus, the number of Gibbs samples of different sub-tensors can be different. This observation, along with observation that more samples can provide better accuracy for noisy sub-tensors, can be used to improve the overall decomposition accuracy for a given number of Gibbs samples. More specifically, the number of samples a noisy sub-tensor, $\boldsymbol{\mathcal{X}}_{\vec{k}}$, is allocated should be proportional to the density, $nd_{\vec{k}}$, of noise it contains:

$$L(\boldsymbol{\mathcal{X}}_{\vec{k}}) = \lceil \gamma \times nd_{\vec{k}} \rceil + L_{min}, \tag{5}$$

where L_{min} is the minimum number of samples a (non-noisy) tensor of the given size would need for accurate decomposition and γ is a control parameter. Note that the value of γ is selected such that the total number of samples needed is equal to the number, $L_{(total)}$, of samples allocated for the whole tensor:

$$L_{(total)} = \sum_{\vec{k} \in \mathcal{K}} L(\boldsymbol{\mathcal{X}}_{\vec{k}}). \tag{6}$$

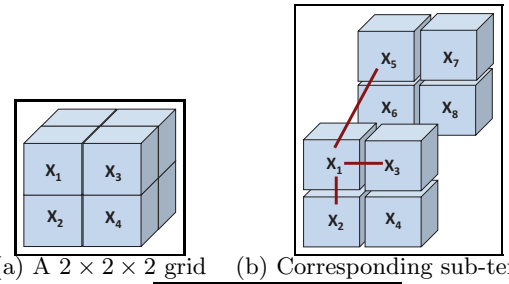
5.3 Noise Sensitive Sample Assignment: Second Attempt

Equations 5 and 6, above, help allocate samples across sub-tensors based on their noise densities. However, they ignore the relationships among the sub-tensors. In Section 4.2, we have seen that, during the iterative refinement process of Phase 2, inaccuracies in decomposition of one sub-tensor can propagate across the rest of the sub-tensors. Therefore, a better approach could be to consider how errors can propagate across sub-tensors when allocating samples.

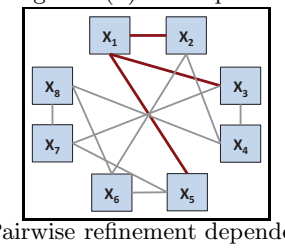
5.3.1 Accounting for Accuracy Inter-dependencies among Sub-Tensors

More specifically, in this section, we note that if we could assign a significance score to each sub-tensor, $\boldsymbol{\mathcal{X}}_{\vec{k}}$, that takes into account not only its noise density, but also the position of the sub-tensor relative to other sub-tensors, we could use this information to allocate samples.

Let $\boldsymbol{\mathcal{X}}$ be a tensor partitioned into a set (or grid) of sub-tensors $\mathfrak{X} = \{\boldsymbol{\mathcal{X}}_{\vec{k}} \mid \vec{k} \in \mathcal{K}\}$. According to the update rule

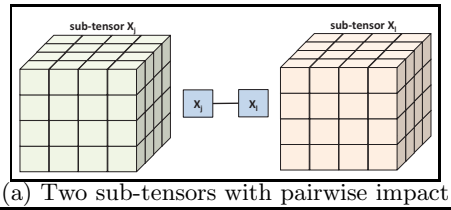


(a) A $2 \times 2 \times 2$ grid (b) Corresponding sub-tensors

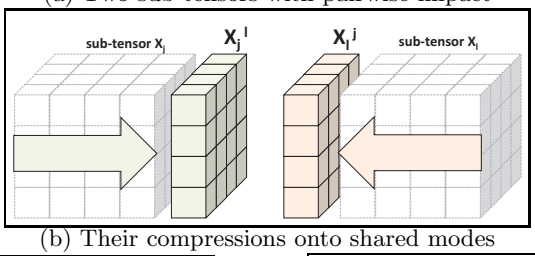


(c) Pairwise refinement dependencies

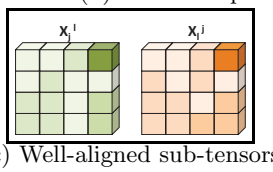
Figure 5: A sample grid and the corresponding pairwise refinement dependencies among the sub-tensors per Equation 4



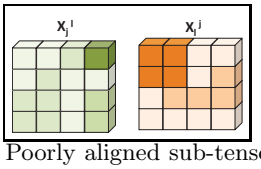
(a) Two sub-tensors with pairwise impact



(b) Their compressions onto shared modes



(c) Well-aligned sub-tensors



(d) Poorly aligned sub-tensors

Figure 6: Measuring the alignment of two sub-tensors: (a) the sub-tensors with pairwise impact, (b) their compressions onto their shared modes, (c) well-aligned tensors have similar distributions on this compressed representation, whereas (d) poorly aligned tensors have dissimilar distributions

(Equation 4) in Section 4.2, if two sub-tensors are lined up along one of the modes of the tensor, they can be used to revise each other's estimates. This means that the update rule ties each sub-tensor's accuracy directly to $\sum_{1 \leq i \leq N} K_i$ other sub-tensors (that line up with the given sub-tensor along one of the N modes – see Figure 5).

Moreover, we see that if the two sub-tensors are similarly distributed along the modes that they share, then they are likely to have high impacts on each other's decomposition; in contrast, if they are dissimilar, their impacts on each other will also be minimal. In other words, given two sub-tensors $\boldsymbol{\mathcal{X}}_{\vec{j}}$ and $\boldsymbol{\mathcal{X}}_{\vec{l}}$, we can compute an *alignment* score, $align(\boldsymbol{\mathcal{X}}_{\vec{j}}, \boldsymbol{\mathcal{X}}_{\vec{l}})$, between $\boldsymbol{\mathcal{X}}_{\vec{j}}$ and $\boldsymbol{\mathcal{X}}_{\vec{l}}$ as

$$align(\boldsymbol{\mathcal{X}}_{\vec{j}}, \boldsymbol{\mathcal{X}}_{\vec{l}}) = cos(\boldsymbol{\mathcal{X}}_{\vec{j}}^{\vec{l}} \boldsymbol{\mathcal{X}}_{\vec{l}}^{\vec{j}}),$$

where $cos()$ is the cosine similarity function and $\boldsymbol{\mathcal{X}}_{\vec{a}}^{\vec{b}}$ is the version of the sub-tensor $\boldsymbol{\mathcal{X}}_{\vec{a}}$ compressed, using the standard Frobenius norm, onto the modes along which the sub-tensor $\boldsymbol{\mathcal{X}}_{\vec{a}}$ and $\boldsymbol{\mathcal{X}}_{\vec{b}}$ are aligned (Figure 6). Intuitively, this pairwise alignment score describes how the decomposition of one sub-tensor will impact another. A sub-tensor which is not aligned with the other sub-tensors is likely to have minimal impact on the accuracy of the overall decomposition even if it contains significant amount of noise. In contrast, a sub-tensor which is well-aligned with a larger portion of other sub-tensors may have a large impact on the other sub-tensors, and consequently, on the whole tensor. Consequently, while the former sub-tensor may not deserve a significant amount of resources, the accuracy of the latter sub-tensor is critical and hence that tensor should be allocated more resources to ensure better overall accuracy.

5.3.2 *Sub-Tensor Centrality based Sample Assignment*

Therefore, given pairwise alignment scores among the sub-tensors, one option is to measure the significance of a sub-tensor relative to other sub-tensors using a centrality measure like PageRank (PR [4]), which computes the significance of each node in a (weighted) graph relative to the other nodes. More specifically, given a graph, $G(V, E)$, the PageRank score $\vec{p}[i]$, of a node $v_i \in V$ is obtained by solving $\vec{p} = (1 - \beta)\mathbf{A}\ \vec{p} + \beta\vec{s}$, where $\mathbf{A}$ denotes the transition matrix, β is a parameter controlling the random walk likelihood , and $\vec{s}$ is a teleportation vector such that for $v_j \in V$, $\vec{s}[j] = \frac{1}{\|V\|}$. Therefore, given (a) the set (or grid) of sub-tensors $\boldsymbol{\mathcal{X}} = \{\boldsymbol{\mathcal{X}}_{\vec{k}} \mid \vec{k} \in \mathcal{K}\}$ and (b) their pairwise alignment scores, we can associate a significance score,

$$\tau_{\vec{k}} = \frac{\vec{p}[\vec{k}] - min_{\vec{j} \in \mathcal{K}}(\vec{p}[\vec{j}])]}{max_{\vec{j} \in \mathcal{K}}(\vec{p}[\vec{j}]) - min_{\vec{j} \in \mathcal{K}}(\vec{p}[\vec{j}])},$$

to each sub-tensor $\boldsymbol{\mathcal{X}}_{\vec{k}}$ by computing PageRank scores described by the vector $\vec{p}$. Given this score, we can then rewrite Equation 5 as

$$L(\boldsymbol{\mathcal{X}}_{\vec{k}}) = \lceil \gamma \times \tau_{\vec{k}} \times nd_{\vec{k}} \rceil + L_{min}, \tag{7}$$

taking into account both the noise density of the sub-tensor along with its relationship to other sub-tensors.

5.4 Noise Sensitive Sample Assignment: S-Strategy

The above formulation considers the position of each sub-tensor in the whole sub-tensor to compute its significance and then multiplies this with the corresponding noise density to decide how much resources to allocate to that sub-tensor. This, however, may not properly take into account the relationship among the noisy sub-tensors as well as the positioning of the other sub-tensors relative to the noisy ones.

In this paper, we note that a better approach would be to consider the noise densities of the sub-tensors directly when evaluating the significance of each sub-tensor. More specifically, instead of relying on PageRank, we propose to use a measure like personalized PageRank (PPR [5]), which computes the significance of each node in a (weighted) graph relative to a given set of seed nodes. Given a graph, G, and a set, $S \subseteq G$, of seed nodes, the PPR score $\vec{p}[i]$, of a node $v_i \in G$ is obtained by solving $\vec{p} = (1 - \beta)\mathbf{A}\ \vec{p} + \beta\vec{s}$,

Algorithm 3 Overview of nTD: noise adaptive decomposition (with noise based resource allocation)

Input: original tensor $\boldsymbol{\mathcal{X}}$, partitioning pattern $\mathcal{K}$, noisy sub-tensor $\mathcal{K}_P$, and decomposition rank, F and total sampling number L

Output: tensor decomposition, $\hat{\boldsymbol{\mathcal{X}}}$

1. obtain the noise profile of the sub-tensors of $\boldsymbol{\mathcal{X}}$,
2. for sub-tensor $\vec{k} \in \mathcal{K}$, assign a decomposition rank $F_{\vec{k}} = F$ and a sampling number $L_{\vec{k}}$ based on noise-sensitive sample allocation strategy, described in Section 5.4.
3. obtain the decomposition, $\hat{\boldsymbol{\mathcal{X}}}$, of $\boldsymbol{\mathcal{X}}$ using the GPTD algorithm (Algorithm 2), given partitioning pattern $\mathcal{K}$ and the initial decomposition ranks $\{F_{\vec{k}} \mid \vec{k} \in \mathcal{K}\}$ and sampling number $\{L_{\vec{k}} \mid \vec{k} \in \mathcal{K}\}$,
4. Return $\hat{\boldsymbol{\mathcal{X}}}$

where $\mathbf{A}$ denotes the transition matrix, β is a parameter controlling the overall importance of the seeds, and $\vec{s}$ is a seeding vector such that if $v_i \in S$, then $\vec{s}[i] = \frac{1}{\|S\|}$ and $\vec{s}[i] = 0$, otherwise. Therefore, given (a) the set (or grid) of sub-tensors $\boldsymbol{\mathfrak{X}} = \{\boldsymbol{\mathcal{X}}_{\vec{k}} \mid \vec{k} \in \mathcal{K}\}$, (b) their pairwise alignment scores, and (c) a seeding vector

$$\vec{s}[\vec{k}] = \frac{nd_{\vec{k}}}{\sum_{\vec{j} \in \mathcal{K}} nd_{\vec{j}}},$$

we associate a noise sensitive significance score,

$$\eta_{\vec{k}} = \frac{\vec{p}[\vec{k}] - min_{\vec{j} \in \mathcal{K}}(\vec{p}[\vec{j}])}{max_{\vec{j} \in \mathcal{K}}(\vec{p}[\vec{j}]) - min_{\vec{j} \in \mathcal{K}}(\vec{p}[\vec{j}])},$$

to each sub-tensor $\boldsymbol{\mathcal{X}}_{\vec{k}}$ based on the PPR scores, described by the vector $\vec{p}$, relative to the noisy tensors. Given this score, we rewrite Equation 5 as

$$L(\boldsymbol{\mathcal{X}}_{\vec{k}}) = \lceil \gamma \times \eta_{\vec{k}} \rceil + L_{min}. \quad (8)$$

5.5 Overview of nTD

The pseudo-code of the proposed noise adaptive tensor decomposition (nTD) process is visualized in Algorithm 3.

6. EXPERIMENTAL EVALUATION

In this section, we report experiments that aim to assess the effectiveness of the proposed *noise adaptive tensor decomposition* approach. In particular, we compare the decomposition (GPTD, Section 4) ALS-based grid tensor decomposition (GridParafac [21]). We further assess the proposed noise-sensitive sample assignment strategy (**s-strategy**) by comparing the performance of nTD, which leverages this strategy, against GPTD with uniform assignment.

6.1 Experiment Setup

Key parameters and their values are reported in Table 1.

Data Sets. In these experiments, we used three real datasets: *Epinions* [32], *Ciao* [32], and *MovieLens* [10, 31]. The first two of these are comparable in terms of their sizes and semantics: they are represented in the form of $5000 \times 5000 \times 999$ (density 1.4×10^{-6}) and $5000 \times 5000 \times 996$ (density 1.7×10^{-6}) tensors, respectively, and both have the schema $\langle user, item, category \rangle$. The *MovieLens* data set ($943 \times 1682 \times 2001$, density 3.15×10^{-5}) is denser and has a different schema, $\langle user, movie, time \rangle$. In all three data sets, the tensor cells contain rating values between 1 and 5 or (if the rating does not exist) a special "null" symbol.

Parameters	*Alternative values*
Noise Density	**10%**; 30%; 50%; 80%
# partitions	$2 \times 2 \times 2$; $\mathbf{4 \times 4 \times 4}$
Per sub-tensor Gibbs sample count	1; **3**; 5; 10; 30; 80
Target Rank (F)	**10**

Table 1: Parameters – default values, used unless otherwise specified, are highlighted

Noise. In these experiments, uniform type of noise were introduced by modifying the existing ratings in the data set (Because of space limitation, sc and mm types noise aren't introduced in this paper, but the performance of those two types are similar as uniform type noise). More specifically, given a uniform noise profile and density, we have selected a subset of the existing ratings (ignoring "null" values) and altered the existing values – either by selecting a completely new rating (which we refer to as *value-independent noise*).

Evaluation Criteria. We use the *root mean squares error* (RMSE) inaccuracy measure to assess the decomposition effectiveness. We also report the decomposition times and memory consumptions. Unless otherwise reported, the execution time of the overall process is reported as if sub-tensor decompositions in Phase 1 and Phase 2 are all executed serially, without leveraging any sub-tensor parallelism. Each experiment was run 10 times with different random noise distributions and averages are reported.

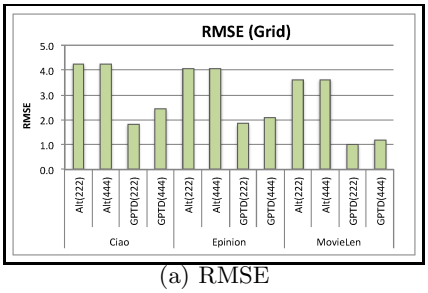
Hardware and Software. We ran experiments on a quad-core CPU Nehalem Node with 12.00GB RAM. All codes were implemented in Matlab and run using Matlab R2015b. For conventional ALS-based CP decomposition, we used MATLAB Tensor Toolbox Version 2.6 [3].

6.2 Discussion of the Results

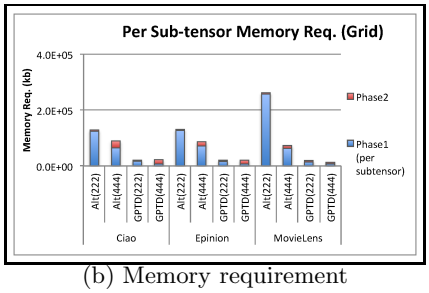
GPTD vs. ALS-based Decomposition. We start the discussion of the results by comparing the proposed grid probabilistic tensor decomposition (GPTD) against the more conventional ALS based grid decomposition, as well as monolithic (no-grid) probabilistic and ALS decompositions. As we see in Figure 7, GPTD provides significantly better accuracy than the conventional ALS based approaches and also requires significantly lesser ALS based grid decompositions. As we expected, we also see that increasing the number of sub-tensors results in a significant drop in the per-sub-tensor memory requirement (therefore improving the scalability of the tensor decomposition process) – though the execution time of the second phase of the process (where the initial decompositions of the sub-tensors are stitched together) increases due to the existence of more sub-tensors to consider.

An important observation in Figures 7 (c) is that the memory requirement for the ALS-based techniques is very sensitive to data density: While the *MovieLens* tensor has smaller dimensionality then the other two, it has a slightly higher density (3.15×10^{-5} vs. 1.7×10^{-6}). Consequently, for this data set, the memory consumptions of the ALS-based techniques (especially when the number of grid partitions used are low) are significantly higher than their memory consumptions for the other two data sets. In contrast, the results show that the probabilistic approach is not sensitive to data density and GPTD has similar memory usage for all three data sets.

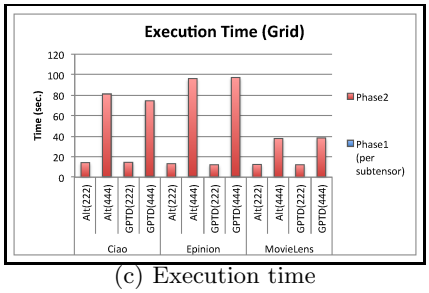
Impact of Noise Density. These results are confirmed in Figure 8(a) & (c), where we vary the noise density between 10% and 80%: as we see here, for all considered noise densities and for all three data sets, the RMSE provided by GPTD is significantly better than the RMSE provided by the



(a) RMSE

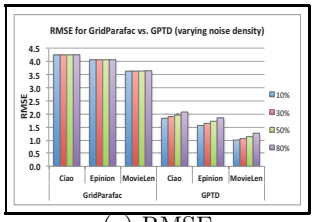


(b) Memory requirement

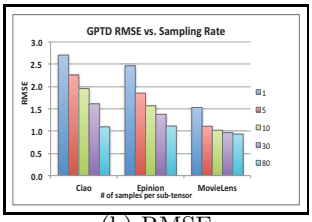


(c) Execution time

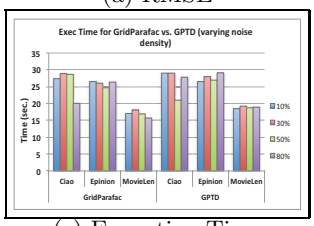
Figure 7: Comparison of different approaches (uniform noise; value independent noise; noise density 10%; $F = 10$; num. Gibbs samples per sub-tensor = 3; max. num. of P2 iteration = 1000; 4 sub-tensors with noise)



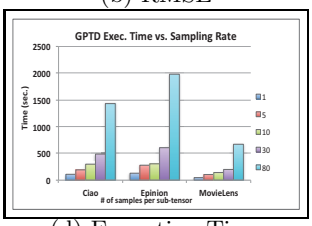
(a) RMSE



(b) RMSE



(c) Execution Time



(d) Execution Time

Figure 8: **(a)**`GPTD` vs. GridParafac ($2 \times 2 \times 2$ grid; varying noise density; uniform noise; value independent noise; num. Gibbs samples per sub-tensor = 3; $F = 10$; max. num. of P2 iteration = 1000); **(b)** `GPTD` with different num. of Gibbs samples ($4 \times 4 \times 4$ grid; uniform noise; value independent noise; noise density 10%; $F = 10$; max. num. of P2 iteration = 1000

conventional ALS-based GridParafac and this RMSE gain does not come with a significant execution time penalty.

Impact of Numbers of Samples. A key parameter of the `GPTD` algorithm is the number of Gibbs samples used per sub-tensor in Phase 1. As we see in Figure 8(b)&(d), as we would expect, increasing the number of Gibbs samples helps reduce the decomposition error (measured using RMSE) ; however having more samples increases the execution time of the algorithm.

It is important to note that, when the number of Gibbs samples is low, the algorithm is very fast, indicating that the worst case complexity of the Bayesian iterations arises only when the number of Gibbs samples is very high. Most critically, as we have already seen in Figures 7 and 8(a), the `GPTD` algorithm does not need too many Gibbs samples: using a few (in these experiments, even just 1) Gibbs samples per sub-tensor is sufficient to provide significantly better accuracy than ALS, reported in Figures 7 (a), with similar or better time overhead as reported in Figures 7 (b).

Impact of the Proposed s-strategy. In Figure 9, we compare the performance of `nTD` with noise-sensitive sample assignment (i.e., `s-strategy`) against `GPTD` with *uniform* sample assignment and the two naive noise adaptations, presented in Sections 5.2 and 5.3.1, respectively. Note that in the scenario considered in this figure, we have 640 total Gibbs samples for 64 sub-tensors, providing on the average 10 samples per sub-tensor. In these experiments, we set L_{min} to 9 (i.e. very close to this average), thus requiring that $576 (= 64 \times 9)$ samples are uniformly distributed across the sub-tensors – this leaves only 64 samples to be distributed adaptively across the sub-tensors based on the noise profiles of the sub-tensors and their relationships to other sub-tensors. As we see in this figure, the proposed `nTD` is able to leverage these 64 uncommitted samples to significantly reduced RMSE relative to `GPTD` with *uniform* sample assignment. Moreover, we also see that naive noise adaptations can actually hurt the overall accuracy. These together show that the proposed `s-strategy` is highly effective in leveraging rough knowledge about noise distributions to better allocate the Gibbs samples across the tensor.

data and density	nd	`nd` × `std`	std
MovieLen (3.15×10^{-5})	1.0221	1.0238	1.0261
Epinion (1.7×10^{-6})	1.5562	1.5744	1.5265

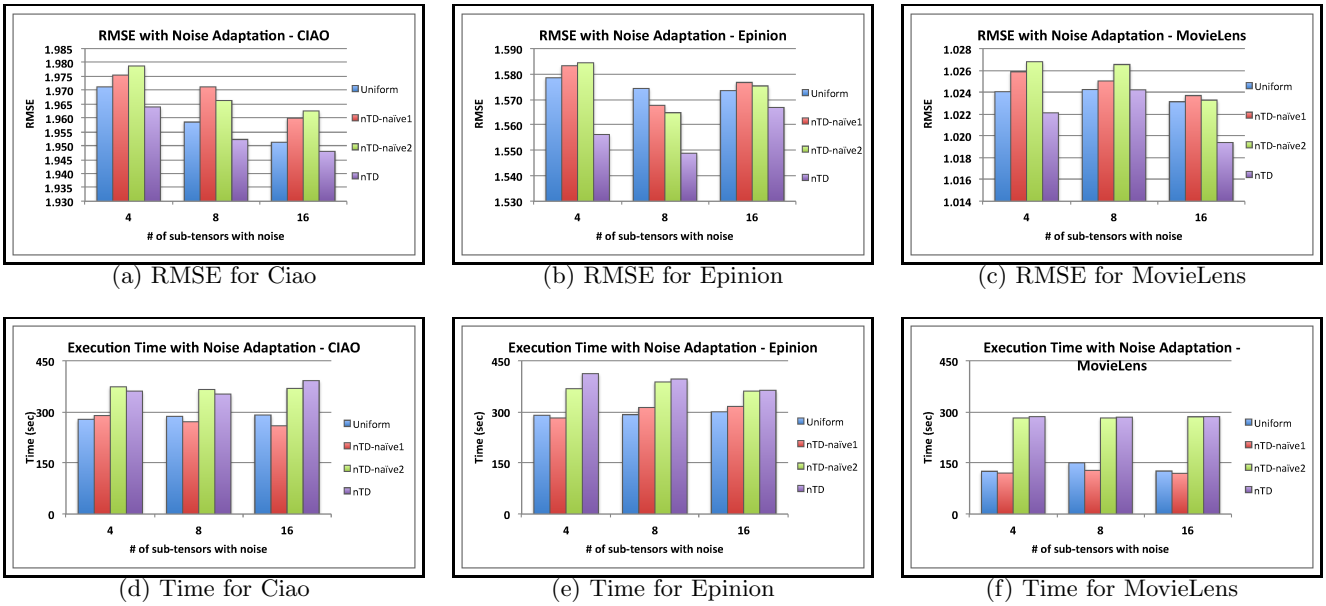
Table 2: Comparison of RMSE of nTD algorithm on different density tensors, with 4 num. of noisy sub-tensors ($4 \times 4 \times 4$ grid; uniform noise; value independent noise; noise density 10%; total num. of samples = 640; $L_{min} = 9$, $F = 10$; max. num. of P2 iteration = 1000)

As expected, `nTD` is costlier than `GPTD` as it requires additional preprocessing to compute sub-tensor alignments in Phase 2. However, the required pre-processing is trivially parallelizable and, as we detail later in the appendix, this overhead can be significantly reduced by allocating additional computational resources.

Impact of the tensor density. As we discussed in section 5.1, tensor density affect noise pattern learning along with the noise percentage for probabilistic decomposition process. To verify the impact of sub-tensor density. We replace current the adaptive parameter, noise density (`nd`), with sub-tensor density(`std`) and the product of `nd` and `std` (`nd` × `std`). Then we consider two different density dataset Moive-Len and Epinion and compare RMSE of `nTD` algorithm. Because only replacing the adaptive parameter, the execution time won't be affected, which is hidden in this paper. Table 2 indicates there is a trade off between noise density and sub-tensor density. While tensor density is higher than specific threshold (in this experiment, it is around 1.0×10^{-5}), noise density will domain the RMSE gain. Otherwise, leveraging sub-tensor density as adaptive parameter can make better RMSE performance.

7. CONCLUSIONS

Real-world data can be noisy. Recent research has shown that it is possible to improve the resilience of the tensor decomposition process to overfitting (an important challenge in the presence of noisy data) by relying on probabilistic techniques. However, existing techniques assume that all



(a) RMSE for Ciao (b) RMSE for Epinion (c) RMSE for MovieLens

(d) Time for Ciao (e) Time for Epinion (f) Time for MovieLens

Figure 9: RMSE and execution time (without sub-tensor parallelism) for `nTD` with different num. of noisy sub-tensors ($4\times4\times4$ grid; uniform noise; value independent noise; noise density 10%; total num. of samples = 640; $L_{min} = 9$, $F = 10$; max. num. of P2 iteration = 1000)

the data and intermediary results can fit in the main memory and (more critically) they treat the entire tensor uniformly, ignoring potential non-uniformities in the noise distribution. In this paper, we noted that, even if we do not have precise knowledge about the cells of the tensor that are subject to noise or the amount of noise they contain, we may have a rough idea about the distribution of noise across the tensor. Given this, we proposed a novel *noise-adaptive decomposition* (`nTD`) technique that leverages rough information about noise distribution to improve the tensor decomposition performance. `nTD` partitions the tensor into multiple sub-tensors and then decomposes each sub-tensor probabilistically through Bayesian factorization. The noise profiles of the grid partitions and their alignments are then leveraged to develop a sample assignment strategy (or `s-strategy`) that best suits the noise profile of a given tensor. Experiments show that `nTD` is significantly better than conventional CP decomposition on noisy tensors.

8. REFERENCES

[1] E. Acar, *et al.* Multiway Analysis of Epilepsy Tensors. Bioinformatics, pages 10-18, 2007.

[2] C. A. Andersson and R. Bro. The N-Way Toolbox for Matlab. Chem. and Intel. Lab. Systems, 52(1):1-4, 2000.

[3] B. W. Bader, T. G. Kolda, *et al.* MATLAB Tensor Toolbox Version 2.5, Available online, January 2012. URL: http://www.sandia.gov/∼tgkolda/TensorToolbox.

[4] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. WWW, 1998.

[5] S. Chakrabarti, Dynamic personalized pagerank in entity-relation graphs. WWW 2007.

[6] E. C. Chi and T. G. Kolda Making Tensor Factorizations Robust to Non-Gaussian Noise. tech. report, arXiv: 1010.3043v1, 2010.

[7] W. Chu, Z. Ghahramani, Probabilistic Models for Incomplete Multi-dimensional Arrays. AISTATS 2009.

[8] I. Davidson, *et al.* Network discovery via constrained tensor analysis of fmri data. KDD 2013.

[9] R. A. Harshman, Foundations of the PARAFAC procedure: Model and conditions for an explanatory multi-mode factor analysis. UCLA Working Papers in Phonetics, 16:1-84, 1970.

[10] F. M. Harper and J. A. Konstan. The MovieLens Datasets: History and Context. TiiS 5, 4, Article 19, 2015.

[11] I. Jeon, E. Papalexakis, U. Kang, and C. Faloutsos. HaTen2: Billionscale tensor decompositions. ICDE'15, 1047-1058, 2015

[12] B. Jeon, *et al.* SCouT: Scalable Coupled Matrix-Tensor Factorization - Algorithm and Discoveries. ICDE 2016.

[13] U. Kang, *et al.* Gigatensor: scaling tensor analysis up by 100 times algorithms and discoveries. KDD 2012

[14] H. Kim and H. Park, Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method, SIAM J. Matrix Anal. Appl., vol. 30, no. 2, pages 713-730, 2008

[15] M. Kim and K.S. Candan. Efficient Static and Dynamic In-Database Tensor Decompositions on Chunk-Based Array Stores. CIKM, 2014.

[16] T.G. Kolda and B.W. Bader. The tophits model for higher-order web link analysis. Workshop on Link Analysis, Counterterrorism and Security, 2006

[17] T. G. Kolda, J. Sun. Scalable tensor decompositions for multi-aspect data mining. ICDM, 363-372 2008.

[18] X. Li, *et al.* Focusing Decomposition Accuracy by Personalizing Tensor Decomposition (PTD). CIKM 2014.

[19] X. Li, *et al.* 2PCP: Two-phase CP decomposition for billion-scale dense tensors. ICDE 2016.

[20] E. Papalexakis, C. Faloutsos, and N. Sidiropoulos. Parcube: Sparse parallelizable tensor decompositions. ECML PKDD, pp. 521-536, 2012.

[21] A.H. Phan and A. Cichocki, PARAFAC algorithms for large-scale problems, Neurocomputing, vol. 74, no. 11, pp. 1970-1984, 2011.

[22] P. Rai *et al.* Scalable Bayesian Low-Rank Decomposition of Incomplete Multiway Tensors. ICML 2014

[23] R. Salakhutdinov and A. Mnih, Probabilistic Matrix Factorization. NIPS 07, pages 1257-1264

[24] J. Sun, S. Papadimitriou, and P. S. Yu. Window based tensor analysis on high dimensional and multi aspect streams. ICDM, pages 1076-1080, 2006.

[25] J.T. Sun, H.J. Zeng, H. Liu, Y. Lu, and Z. Chen. Cubesvd: a novel approach to personalized web search. WWW 2005

[26] C. E. Tsourakakis, Mach: Fast randomized tensor decompositions. Arxiv preprint arXiv:0909.4969, 2009

[27] L. Tucker, Some mathematical notes on three-mode factor analysis. Psychometrika, 31:279-311, 1966.

[28] L. Xiong, *et al.* Temporal collaborative filtering with bayesian probabilistic tensor factorization. SDM 2010.

[29] Q. Zhang, M. Berry, B. Lamb, and T. Samuel. A parallel nonnegative tensor factorization algorithm for mining global climate data. ICCS'09, pages 405-415, 2009.

[30] Q. Zhao, L. Zhang, A. Cichoki, Bayesian CP Factorization of Incomplete Tensors with Automatic Rank Determination. IEEE Trans. on Pat. Analysis and Mach. Intel. 2014.

[31] http://grouplens.org/datasets/movielens/

[32] http://www.public.asu.edu/~jtang20/datasetcode/truststudy.htm