Robust Low-Rank Tensor Decomposition with the L2 Criterion

Qiang Heng, Eric C. Chi, and Yufeng Liu

ABSTRACT

The growing prevalence of tensor data, or multiway arrays, in science and engineering applications motivates the need for tensor decompositions that are robust against outliers. In this article, we present a robust Tucker decomposition estimator based on the L2 criterion, called the Tucker-L2E. Our numerical experiments demonstrate that Tucker-L2E has empirically stronger recovery performance in more challenging high-rank scenarios compared with existing alternatives. The appropriate Tucker-rank can be selected in a data-driven manner with cross-validation or hold-out validation. The practical effectiveness of Tucker-L2E is validated on real data applications in fMRI tensor denoising, PARAFAC analysis of fluorescence data, and feature extraction for classification of corrupted images.

1. Introduction

There has been growing interest in tensors, or multi-way arrays, since many real-world datasets have a multi-dimensional structure that is not well exploited by two-dimensional matrix-based data analysis. Some of the most important tensor-based data analysis tools are low-rank tensor decompositions, which primarily take two forms: the CANDECOMP/PARAFAC (CP) decomposition (Carroll and Chang 1970; Harshman 1970) and the Tucker decomposition (Tucker 1966). In the last decade, several new tensor decomposition paradigms based on alternative notions of tensor rank have been proposed, including low-tubal-rank factorization (Kimler and Martin 2011), tensor-train decomposition (Oseledets 2011) and tensor ring decomposition (Zhao et al. 2016). Each decomposition has suitable applications as well as limitations.

A major challenge of low-rank tensor decomposition is that the observed tensor may be grossly corrupted with outliers or sparse noise. This article addresses the robust tensor decomposition problem when the underlying tensor has low Tucker-rank. When an N-way data tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is fully observed, we assume that $\mathcal{X}$ is generated from the following model

$$\mathcal{X} = \mathcal{L} + \mathcal{S} + \mathcal{E},$$

where $\mathcal{L}$ denotes an underlying low Tucker-rank tensor, $\mathcal{S}$ denotes a sparse tensor of outlying entries, and $\mathcal{E}$ denotes a tensor of dense noise.

We also consider the case when $\mathcal{X}$ is only partially observed over a subset $\Omega$ of its indices. Let $[I]$ denote the set of consecutive integers $\{1, \ldots, I\}$. Then the set $\Omega \subset [I_1] \times \cdots \times [I_N]$ is an index set of observed entries, and we assume that

$$x_{i_1i_2\cdots i_N} = l_{i_1i_2\cdots i_N} + s_{i_1i_2\cdots i_N} + e_{i_1i_2\cdots i_N},$$

for $(i_1, i_2, \ldots, i_N) \in \Omega$.

Our goal is to recover the latent factors of the underlying low-rank tensor $\mathcal{L}$. Ideally, a robust method should remain effective in the absence of $\mathcal{S}$ or $\mathcal{E}$. If the goal is estimating the low-rank tensor $\mathcal{L}$ instead of its latent factors, we refer to robust tensor decomposition as robust tensor recovery. We focus on the Tucker decomposition in this article. Since the CP decomposition is a special case of the Tucker decomposition when the CP-rank does not exceed any of the tensor dimensions, the proposed method may also be applied to denoise or reconstruct low CP-rank tensors.

Many CP and Tucker decomposition methods have been proposed in the literature. We discuss these in Section 3. Our robust formulation adapts the L2E method (Scott 2001, 2009) to the Tucker decomposition. The L2E is a minimum distance estimator that minimizes the integrated squared error (ISE) for parametric estimation. The integrated squared error is also referred to as the L2 criterion (Hjort 1994; Terrell 1990) in nonparametric density estimation, hence, the name L2E. Consequently, we call our formulation of Tucker decomposition the Tucker-L2E. Minimum distance estimators are well known to possess robustness properties (Donoho and Liu 1988). Moreover, minimization of the L2 criterion has been employed in developing a wide range of robust statistical models including structured sparse models (Lozano, Meinshausen, and Yang 2016; Chi and Chi 2022; Liu, Chi, and Lange in press), quantile regression (Lane 2012), mixture models (Lee 2010), classification (Chi and Scott 2014), forecast aggregation (Ramos 2014), and survival analysis.
(Yang and Scott 2013). It also has successes in engineering applications including signal processing tasks such as wavelet-based image denoising (Scott 2006) and image registration (Ma et al. 2013, 2015; Yang et al. 2017). The $L_2E$ is attractive among minimum distance estimators since it strikes a good balance between robustness, efficiency, and computational tractability (Scott 2001, 2009).

The rest of this article is organized as follows. In Section 2, we introduce tensor notation and terminology. In Section 3, we briefly review several prominent CP and Tucker decomposition formulations, both nonrobust and robust, in the existing literature. In Section 4, we present our formulation of robust Tucker decomposition and its solution algorithm. In Sections 5 and 6, we demonstrate the practical effectiveness of our approach and its advantage over existing methods in terms of recovery capability with numerical experiments and real data applications.

2. Background on Tensors and their Decompositions

We review basic operations on matrices and tensors using the terminology and notation in Kolda and Bader (2009). A tensor $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is an element of the tensor product of $N$ real vector spaces. The number of dimensions, or ways, $N$ is called the order of tensor $\mathbf{X}$. Each dimension is also called a mode. A fiber of $\mathbf{X}$ is a column vector subset of $\mathbf{X}$, defined by fixing all but one of the indices. For a matrix, an order-2 tensor, a mode-1 fiber is a matrix column, and a mode-2 fiber is a matrix row. A slice of $\mathbf{X}$ is a matrix subset of $\mathbf{X}$, defined by fixing all but two of the indices.

2.1. Basic Tensor Operations

It is often convenient to reshape a tensor into a matrix or a vector. The former is referred to as matricization, while the latter is referred to as vectorization. The mode-$n$ matricization of a tensor $\mathbf{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, denoted $\mathbf{X}^{(n)} \in \mathbb{R}^{I_n \times I_1 \times \cdots \times I_{n-1} \times I_{n+1} \times \cdots \times I_N}$ with $I_n = \prod_{k=1,k\neq n}^N I_k$, arranges the mode-$n$ fibers as the columns of the matrix $\mathbf{X}^{(n)}$ in the following lexicographic order. The tensor element $x_{i_1 \cdots i_N}$ is mapped to the matrix element of $\mathbf{X}^{(n)}$ with index $(i_n, j)$ where $j = 1 + \sum_{k=1,k\neq n}^N (i_k - 1) I_k$ and

$$J_k = \begin{cases} 1 & \text{if } k = 1 \text{ or if } k = 2 \text{ and } n = 1, \\ \prod_{k=1}^{k-1} I_{k'} & \text{otherwise}. \end{cases}$$

The matricization of $\mathbf{X}$, denoted $\text{vec}(\mathbf{X})$, is the vector obtained by stacking the columns of its mode-1 matricization $\mathbf{X}^{(1)}$ on top of each other.

We will use two kinds of products involving tensors and matrices throughout this article. The elementwise Hadamard product of two tensors $\mathbf{X}$ and $\mathbf{Y}$ of identical size $I_1 \times \cdots \times I_N$ is denoted by $\mathbf{X} \odot \mathbf{Y}$ and is the tensor whose $(i_1, \ldots, i_N)$th element is given by $x_{i_1 \cdots i_N} y_{i_1 \cdots i_N}$. The $n$-mode product of a tensor $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ with a matrix $\mathbf{A} \in \mathbb{R}^{I_N \times I}$ is denoted by $\mathbf{X} \times_n \mathbf{A}$, which is a tensor of size $I_1 \times I_2 \times \cdots \times I_{n-1} \times I \times I_{n+1} \times \cdots \times I_N$ with elements

$$(\mathbf{X} \times_n \mathbf{A})_{i_1 \cdots i_{n-1} j n+1 \cdots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 \cdots i_{n-1} i_n} a_{jn}.$$ 

for $j \in [I]$. Note that the mode-$n$ matricization of the $n$-mode product $\mathbf{X} \times_n \mathbf{A}$ can be expressed as

$$[\mathbf{X} \times_n \mathbf{A}]^{(n)} = \mathbf{A} \mathbf{X}^{(n)}.$$ 

The Frobenius norm and $\ell_1$-norm of a tensor $\mathbf{X} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$ are defined as

$$\|\mathbf{X}\|_F = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} x_{i_1 i_2 \cdots i_N}^2 \text{ and}$$

$$\|\mathbf{X}\|_1 = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_N=1}^{I_N} |x_{i_1 i_2 \cdots i_N}|.$$ 

Finally, we use $\mathbf{X}^{\otimes 2}$ to denote the tensor obtained by raising each entry of $\mathbf{X}$ to the power of 2. We use $\mathbf{X} + b$ to denote the tensor obtained by multiplying every entry of $\mathbf{X}$ by $a$ and then adding $b$ to every element of the resulting scaled tensor. We denote the sum of all tensor entries as $\text{sum}(\mathbf{X})$ and the elementwise exponential of a tensor as $\exp(\mathbf{X})$.

2.2. Tensor Decompositions and Ranks

The Tucker decomposition of $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ with rank $R = (r_1, r_2, \ldots, r_N)$ aims to find a core tensor $\mathbf{G} \in \mathbb{R}^{I_1 \times r_2 \times \cdots \times r_N}$ and factor matrices $A^{(n)} \in \mathbb{R}^{I_n \times r}$ for $n \in [N]$ such that

$$\mathbf{X} \approx \mathbf{G} \times_1 A^{(1)} \times_2 A^{(2)} \times_3 \cdots \times_N A^{(N)}$$

$$= [\mathbf{G}; A^{(1)}, \ldots, A^{(N)}],$$

where the equality uses the more compact notation $[\mathbf{G}; A^{(1)}, \ldots, A^{(N)}]$ introduced in Kolda (2006). Sometimes the columns of $A^{(n)}$ are required to be orthogonal so that the columns of $A^{(n)}$ can be interpreted as the principal components of the $n$th mode, but we do not require this in this work. The tensor $\mathbf{X}$ is said to have Tucker-rank $R = (r_1, r_2, \ldots, r_N)$ if $\text{rank}(\mathbf{X}) = r_n$ for $n \in [N]$.

The CP decomposition for $\mathbf{X}$ with rank $R = r$ aims to find $a_{i}^{(n)} \in \mathbb{R}^{I_n}$ for $n \in [N], i \in [r]$, and a weight vector $\mathbf{V} \in \mathbb{R}^r$ such that

$$\mathbf{X} \approx \sum_{i=1}^{r} \gamma_i a_{i}^{(1)} \circ a_{i}^{(2)} \circ \cdots \circ a_{i}^{(N)},$$

where $\circ$ denotes the outer product. Just as the outer product of two vectors yields a rank-1 matrix, the outer product of $N$ vectors yields an $N$-way rank-1 tensor. Thus, the CP model aims to approximate a tensor with a linear combination of rank-1 tensors. Following Kolda and Bader (2009), we write the linear combination of rank-1 tensors $\sum_{i=1}^{r} \gamma_i a_{i}^{(1)} \circ a_{i}^{(2)} \circ \cdots \circ a_{i}^{(N)}$ more compactly as $[\mathbf{V}; A^{(1)}, \ldots, A^{(N)}]$, where $A^{(n)} = [a_{1}^{(n)}, a_{2}^{(n)}, \ldots, a_{r}^{(n)}] \in \mathbb{R}^{r \times I_n}$ are the CP factor matrices. The tensor $\mathbf{X}$ is said to have CP-rank $r$ if $r$ is the smallest integer possible for the approximation to hold with equality. When $r \leq \min(I_1, I_2, \ldots, I_N)$, the CP decomposition can be viewed as a special case of Tucker decomposition. This is because if $\mathbf{G}$ has dimension $(r, r, \ldots, r)$ and is "superdiagonal," that is, its only nonzero entries are $g_{i,i-1}$ for $i \in [r]$, then

$$\mathbf{G} \times_1 A^{(1)} \times_2 A^{(2)} \times_3 \cdots \times_N A^{(N)} = [\mathbf{G}; A^{(1)}, A^{(2)}, \ldots, A^{(N)}].$$
where \( \mathbf{g} \in \mathbb{R}^r \) is a vector containing the superdiagonal of nonzero entries of \( \mathcal{G} \). A tensor with CP-rank \( r \) has Tucker-rank \((r, r, \ldots, r)\), but the converse does not hold in general.

For notational simplicity, we will often “absorb” the weight vector into one of the factor matrices when writing the CP model, for example,

\[
\mathbf{y}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}] = \left[ \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)} \right],
\]

where \( \mathbf{A}^{(1)} = \mathbf{A}^{(1)} \text{diag(} \mathbf{y} \text{)} \) and \( \text{diag(} \mathbf{y} \text{)} \) is the diagonal matrix with \( i \)th diagonal entry \( y_i \).

### 3. Related Work

#### 3.1. Tensor Decompositions based on Least Squares

Nonrobust CP and Tucker decompositions are formulated as the solutions to nonlinear least squares problems that minimize the Frobenius norm of the residual tensor. Formally, the CP decomposition solves the following optimization problem

\[
\minimize_{\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}} \left\| \mathbf{X} - \left[ \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)} \right] \right\|_F^2.
\]

Historically, the alternating least squares (ALS) algorithm (Carroll and Chang 1970; Harshman 1970) has been the “workhorse” of solving the above CP decomposition problem, which updates one of the factor matrices while holding the others fixed. Acar, Dunlavy, and Kolda (2011), however, showed that a direct optimization approach can obtain more accurate estimates of the low-rank tensor, especially when the specified rank is greater than the true rank. By direct optimization, we mean that the gradients with respect to the factor matrices are computed and all the factor matrices are updated “all-at-once” or simultaneously with a local optimization method like nonlinear conjugate gradient method (NCG) or Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm (L-BFGS). We refer to this direct optimization approach as CP-OPT.

Similarly, the Tucker decomposition, or the best rank- \((r_1, r_2, \ldots, r_N)\) approximation of \( \mathbf{X} \), is formulated as

\[
\minimize_{\mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)}} \left\| \mathbf{X} - \left[ \mathbf{G}; \mathbf{A}^{(1)}, \ldots, \mathbf{A}^{(N)} \right] \right\|_F^2.
\]

The first method to compute the Tucker decomposition introduced in Tucker (1966) was later shown by De Lathauwer, De Moor, and Vandewalle (2000a) to be a generalization of the matrix singular value decomposition (SVD), known today as Higher-order Singular Value Decomposition (HOSVD). However, it does not produce the best fit in terms of relative error. An alternating least squares algorithm named Higher-order Orthogonal Iteration (HOOI) (Kroonenberg and De Leeuw 1980; Kaptainy, Neudecker, and Wansbeek 1986; De Lathauwer, De Moor, and Vandewalle 2000b) has stronger empirical performance and is the most widely adopted method to compute the Tucker decomposition. In fact, it has also been shown that HOSVD achieves a sub-optimal rate of estimation error, while HOOI is information-theoretical optimal (Zhang and Xia 2018).

The above least squares formulations reflect a Gaussian assumption. This assumption is a natural starting point to develop a robust tensor decomposition formulation. Consequently, our Tucker-L2E method is derived under it. Nonetheless, it is important to note that there have been recent works extending tensor decomposition under non-Gaussian modeling assumptions. For example, Hong, Kolda, and Duersch (2020) studied generalized CP decomposition with various statistically motivated loss functions. Han, Willett, and Zhang (2022) studied generalized low Tucker-rank tensor estimation, which establishes an upper bound for statistical error and a linear computational convergence rate.

#### 3.2. Robust Principal Component Analysis (RPCA)

Perhaps the most classic robust matrix recovery method is Principal Component Pursuit (Candès et al. 2011), which decomposes a corrupted matrix \( \mathbf{X} \in \mathbb{R}^{m \times n} \) as the sum of a low-rank matrix \( \mathbf{L} \) and a matrix of sparse outliers \( \mathbf{S} \). This is achieved by solving the following convex optimization problem

\[
\minimize_{\mathbf{X}, \mathbf{S}} \left\| \mathbf{L} \right\|_* + \lambda \left\| \mathbf{S} \right\|_1 \quad \text{subject to} \quad \mathbf{L} + \mathbf{S} = \mathbf{X},
\]

where \( \left\| \cdot \right\|_* \) denotes the nuclear norm and \( \lambda \) is a nonnegative tuning parameter. Candès et al. (2011) proved that (3) achieves exact recovery of the low-rank component \( \mathbf{L} \) under low-rank and incoherence assumptions. Since the matricizations of a low Tucker-rank tensor are low-rank matrices, RPCA is often used as a baseline for robust tensor recovery.

#### 3.3. Higher-Order RPCA

Goldfarb and Qin (2014) proposed Higher-order Robust Principal Component Analysis (HoRPCA) as a generalization of RPCA to tensors. HoRPCA comes in several different variations. We discuss the three best-performing variants in this section. The first variant is the singleton model (HoRPCA-S), formulated as

\[
\minimize_{\mathbf{L}, \mathbf{S}} \sum_{i=1}^{N} \left\| \mathbf{L}(i) \right\|_* + \lambda \left\| \mathbf{S} \right\|_1 \quad \text{subject to} \quad \mathbf{L} + \mathbf{S} = \mathbf{X}.
\]

HoRPCA-S minimizes the sum of nuclear norms of all the matricizations of \( \mathbf{L} \) to encourage each mode to be low-rank. The descriptor “singleton” is in contrast with the mixture model in (HoRPCA-W), formulated as

\[
\minimize_{\mathbf{L}, \mathbf{S}} \sum_{i=1}^{N} \left\| \mathbf{L}(i) \right\|_* + \lambda \left\| \mathbf{S} \right\|_1 \quad \text{subject to} \quad \sum_{i=1}^{N} \mathbf{L}(i) + \mathbf{S} = \mathbf{X}.
\]

HoRPCA-W represents the underlying tensor as the sum of \( N \) tensors that are only low-rank in one mode. Tomioka et al. (2011) first introduced the mixture model which can be considered a relaxation of the singleton model. Yang, Feng, and Suykens (2015) later proposed robust tensor recovery also using the mixture model along with more robust loss functions. The mixture model can automatically detect the rank-deficient modes and yields better recovery results when the underlying tensor is only low-rank in certain modes. However, in our experience, the limitation of the mixture model is that it does
not approximate the low-rank tensor well when the minimum rank in the Tucker rank tuple is relatively large. The variant with the strongest recovery performance presented in Goldfarb and Qin (2014) is the constrained nonconvex model (HoRPCA-C), formulated as

$$\begin{align*}
\text{minimize} & \quad \|S\|_1 \quad \text{subject to} \quad L + S = X, \\
& \quad \text{rank}(L_{(i)}) \leq r_i, \ i \in [N].
\end{align*}$$

(6)

The key difference between the formulation in (6) and those in (4) and (5) is that (6) enforces a hard constraint on the Tucker-rank \((r_1, r_2, \ldots, r_N)\) whereas the other formulations trade off the rank of the latent tensor \(L\) with the \(\ell_1\)-norm of the outlier tensor \(S\) via the penalty parameter \(\lambda\).

Goldfarb and Qin (2014) iteratively solves (4), (5), and (6) using the alternating direction method of multipliers (ADMM) algorithm (Boyd et al. 2011). In particular, since (6) is nonconvex, the standard convergence guarantees of ADMM for convex programs do not apply. However, (6) demonstrates strong empirical convergence in practice.

### 3.4. Bayesian Robust Tensor Factorization

Zhao et al. (2015) approached the robust CP decomposition problem in a generative manner with Bayesian Robust Tensor Factorization (BRTF). Normal-gamma priors are used to induce column sparsity of the factor matrices and elementwise sparsity of the outliers. We direct readers to Zhao et al. (2015) for the detailed hierarchical model setup. What makes BRTF attractive is that it can automatically infer the appropriate CP-rank.

### 3.5. Riemannian Gradient Descent

Recently, Cai, Li, and Xia (2022) introduced a general framework under a low-rank plus sparse tensor model. The algorithm is based on Riemannian gradient descent and a novel gradient pruning procedure, which is able to estimate both the low-rank tensor and the outlying sparse tensor. The appropriate Tucker-rank and sparsity level of outliers can be tuned with a BIC-type criterion. Performance bounds for both the low-rank and the sparse tensors are established under suitable conditions. The proposed algorithm is also applicable to Bernoulli and Poisson distributed data. We refer to the algorithm described in Cai, Li, and Xia (2022) as RGrad in this article.

### 3.6. Partial Observations

Real-world tensor data is often not fully observed. In the tensor completion literature, a binary weight tensor \(W\) whose entries are 0/1 to indicate missing or observed is often used as a mask to model missing entries. For example, CP and Tucker decomposition in the presence of missing data can be formulated as

$$\begin{align*}
\text{minimize} & \quad \|W \ast (X - [A^{(1)}, \ldots, A^{(N)}])\|_F^2, \\
& \quad \text{subject to} \quad A^{(1)}, \ldots, A^{(N)}
\end{align*}$$

(7)

and

$$\begin{align*}
\text{minimize} & \quad \|W \ast (X - [S; A^{(1)}, \ldots, A^{(N)}])\|_F^2.
\end{align*}$$

(8)

Similar to (1) and (2), (7) and (8) can also be solved with direct optimization, for example, CP-WOPT (Acar et al. 2011) and Tucker-WOPT (Filipović and Jukić 2015). For RPCA and HoRPCA, an easy way to deal with missing data is to enforce the equality constraints only on observed entries. Similarly for BRTF, we can choose to incorporate only the observed tensor entries into the hierarchical model.

It is worth mentioning that there is significant interest in tensor completion in recent research. In the last few years there are many advances on the theoretical front for low CP/Tucker-rank tensor completion. For example, Cai et al. (2019) studied the reconstruction of a low-rank symmetric tensor and proposed a two-stage nonconvex algorithm which achieves optimal \(\ell_\infty\) statistical accuracy. Building upon Cai et al. (2019), Cai, Poor, and Chen (2023) studied the nonconvex tensor completion problem from an uncertainty quantification perspective. Xia and Yuan (2019) studied the sample size requirement for the exact recovery of a low-rank tensor from a subset of its entries, using a spectral initialization method and gradient descent. Zhang (2019) proposed a novel tensor measurement scheme for low-rank tensor completion. Xia, Yuan, and Zhang (2021) proposed a procedure for low-rank tensor completion from noisy entries based on spectral initialization and power iteration that is computationally efficient and achieves the optimal rates of convergence. Recently, Tong et al. (2022) developed a scaled gradient descent approach to low-rank tensor completion and regression which converges at a linear rate independent of the condition number of the true tensor.

**Remark.** Among the previously reviewed methods, RPCA, HoRPCA-S and HoRPCA-W induce a low-rank structure using nuclear norm penalties while CP-OPT, HOOI, HoRPCA-C, and RGrad require the rank to be explicitly specified. We categorize methods that employ nuclear norm penalties as “penalized formulations” and methods requiring specification of rank as “rank-constrained formulations.” Our approach, Tucker-L2E, is an instance of the latter. In Section 5, we demonstrate that rank-constrained formulations are generally more robust and handle dense noise better than penalized formulations, provided that the ranks are appropriately specified. We also illustrate in Section 5.3 that HoRPCA-C and Tucker-L2E can tolerate some level of rank overestimation if the noise is sparse.

We acknowledge several other robust CP/Tucker decomposition methods that we have not detailed in this section due to the limitation of space. Anandkumar et al. (2016), for example, proposed an iterative thresholding algorithm for robust tensor decompositions which is designed to recover CP models with orthogonal factors. Gu, Gui, and Han (2014) studies the statistical performance of a convex formulation of robust tensor decomposition. Wu et al. (2017) uses the Cauchy distribution to handle long-tail noise in CP and Tucker decomposition.

### 4. Methodology

In this section, we introduce our proposed Tucker-L2E method. We briefly review the L2E method in Section 4.1 and then develop Tucker-L2E in Section 4.2. The algorithm and implementation details for Tucker-L2E can be found in Section 4.3.
parameter. Under this assumption, the integral consists of a mean and precision (inverse standard deviation) as this work, we assume that has a closed-form expression for many parametric models. In be ignored in the computation of a minimizer. The first integral does not involve the parameter of interest mean provides an unbiased estimate of this quantity. The third is a random variable drawn from .

We first review the parametric estimation framework using the integral

\[
\int \left[ \hat{\phi}(x \mid \theta) - \phi(x) \right]^2 dx. \tag{9}
\]

Of course, recovering in this way is impossible in practice since is unknown. Fortunately, although we cannot minimize the L2 distance between \(\hat{\phi}(x)\) and \(\phi(x \mid \theta)\) directly, we can minimize an unbiased estimate of the distance. To do this, we first expand (9) as

\[
\int \hat{\phi}(x \mid \theta)^2 dx - 2 \int \hat{\phi}(x \mid \theta) \phi(x) dx + \int \phi(x)^2 dx.
\]

The second integral is the expectation \(E_X[\hat{\phi}(X \mid \theta)]\), where is a random variable drawn from \(\phi\). Therefore, the sample mean provides an unbiased estimate of this quantity. The third integral does not involve the parameter of interest \(\theta\) and may be ignored in the computation of a minimizer. The first integral has a closed-form expression for many parametric models. In this work, we assume that \(\hat{\phi}(x \mid \theta)\) is a normal density where \(\theta\) consists of a mean and precision (inverse standard deviation) parameter. Under this assumption, the integral \(\int \phi(x \mid \theta)^2 dx\) can be written as an explicit function of the precision parameter alone. As a concrete example, consider the univariate case and assume that \(\theta = (\mu, \tau)\) and \(\phi(x \mid \theta)\) is the density function of a normal random variable \(X \sim \mathcal{N}(\mu, \tau^{-2})\). Then the L2E for the univariate mean and precision is

\[
\hat{\theta}_{L2E} = \arg\min_{\mu, \tau} h(\mu, \tau), \tag{10}
\]

where

\[
h(\mu, \tau) = \frac{\tau}{2\sqrt{\pi}} - \frac{\tau}{n\sqrt{\pi}} \sum_{i=1}^{n} \exp \left( -\frac{\tau^2}{2} (x_i - \mu)^2 \right). \tag{11}
\]

For a fixed \(\tau\), the \(\mu\) that minimizes (11) approaches the MLE of \(\mu\) as \(\tau\) approaches zero. To see this, note that the Taylor expansion of \(- \exp(-t)\) around 0 is \(-1 + t + o(t)\). Therefore, for sufficiently small \(\tau\),

\[
h(\mu, \tau) \approx \frac{\tau}{2\sqrt{\pi}} - \frac{\tau^3}{n\sqrt{2\pi}} \sum_{i=1}^{n} (x_i - \mu)^2.
\]

We can visualize how the L2 criterion \(h(\mu, \tau)\) varies with \(\mu\) for fixed values of \(\tau\) to illustrate how the L2E achieves robustness. We consider two examples. In the first example, \(\phi(x)\) is a three-to-one mixture of the constant 0 and . The second example is identical to the first but the observations are further corrupted with additive \(\mathcal{N}(0,1)\) noise. One hundred observations \(x_1, x_2, \ldots, x_{100}\) are generated for each example. In both examples, the true value for \(\mu = 0\) and the 25% uniformly distributed observations can be regarded as outliers. Intuitively, the true value for \(\tau = 0.8\) is the closest (0.25) to the true value 0. For sufficiently small \(\tau\),

\[
h(\mu, \tau) \approx \frac{\tau}{2\sqrt{\pi}} - \frac{\tau^3}{n\sqrt{2\pi}} \sum_{i=1}^{n} (x_i - \mu)^2.
\]

4.2. Tucker-L2E

We set up the optimization problem for estimating the Tucker-L2E model in stages. We start with a natural adaptation of (10) to accommodate tensor data which replaces the location parameter...
\( \mu \) with a latent mean tensor \( \mathcal{L} \). A preliminary formulation of robust tensor estimation based on the L2 criterion is to minimize the following objective function

\[
h(\mathcal{L}, \tau) = \frac{\prod_{n=1}^{N} I_n}{2\sqrt{\pi}} \tau \exp\left(-\frac{\tau^2}{2} (\mathcal{X} - \mathcal{L})^2\right),
\]

where \( \mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N} \) is the observed noisy tensor.

When the data tensor \( \mathcal{X} \) has missing entries and is observed only on the index set \( \Omega \), similar to CP-WOPT and Tucker-WOPT, we can sum over only the observed entries in the objective to account for missing data and minimize the natural generalization of the objective function in (12)

\[
h_\Omega(\mathcal{L}, \tau) = \frac{\sum(W)}{2\sqrt{\pi}} \tau \exp\left(-\frac{\tau^2}{2} (\mathcal{X} - \mathcal{L})^2\right).
\]

Recall that the tensor \( W \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N} \) is binary and depends on \( \Omega \) in the following manner. The \((i_1, i_2, \ldots, i_N)\)th entry of \( W \) is 1 if \( x_{i_1,i_2,\ldots,i_N} \in \Omega \) and 0 otherwise. Note that \( h_\Omega(\mathcal{L}, \tau) \) and \( h(\mathcal{L}, \tau) \) coincide when \( \mathcal{X} \) is fully observed, that is, \( \Omega = [I_1] \times \cdots \times [I_N] \). Thus, we will work with the more general objective function \( h_\Omega \) moving forward.

To estimate a low Tucker-rank tensor, we parameterize \( \mathcal{L} \) as \( \mathcal{L} = \bigg[ \mathcal{S}; A^{(1)}, \ldots, A^{(N)} \bigg] \). Notice that this parameterization is equivalent to imposing the constraints \( \text{rank}(L(n)) \leq r_n, n \in [N] \) on \( \mathcal{L} \) (see Zhang and Xia (2018)). Thus, we seek the solution to the following optimization problem over the parameters \( \mathcal{S}, A^{(1)}, \ldots, A^{(N)} \) and \( \tau \).

\[
\min_{\mathcal{S}, \tau} h_\Omega \left( \bigg[ \mathcal{S}; A^{(1)}, \ldots, A^{(N)} \bigg], \tau \right) \quad \text{subject to} \quad \tau > 0.
\]

We now turn our attention to details concerning the parameter \( \tau \). Notice that \( \tau \) must be positive since it is a precision parameter and consequently introduces an additional constraint over an open set. An easy way to ensure a positive precision while at the same time eliminating the strict positivity constraint is to reparameterize \( \tau \) as \( \tau = \exp(\eta) \) and optimize over \( \eta \in \mathbb{R} \). Moreover, recall in Section 4.1, we discussed that although the \( L_2E \) is more robust when \( \tau \) is larger, the accompanying spurious local minima may make computing solutions of (12) harder. In other words, the precision parameter \( \tau \) trades off robustness and the “roughness” of the optimization landscape. We also see from the left panel of Figure 1 that the minimum value of \( h(\mu, \tau) \) always decreases as \( \tau \) increases. This suggests that (12) may not even have a finite infimum if we allow \( \tau \) to diverge to infinity. Therefore, it is reasonable to impose an upper bound on \( \tau \), or equivalently on \( \eta \). Thus, we seek the solution to the following optimization problem over the parameters \( \mathcal{S}, A^{(1)}, \ldots, A^{(N)} \) and \( \eta \).

\[
\min_{\mathcal{S}, \tau} h_\Omega \left( \bigg[ \mathcal{S}; A^{(1)}, \ldots, A^{(N)} \bigg], e^\eta \right) \quad \text{subject to} \quad \eta \leq \eta_{\text{max}}.
\]

Theorem 4.1 gives some justification for reparameterizing \( \tau \) as \( \exp(\eta) \) and placing an upper bound on \( \eta \) in (14).

Theorem 4.1. Problem (14) has a finite infimum and the infimum is negative.

We provide a proof of Theorem 4.1 in the supplementary materials. Theorem 4.1 is relevant in our problem setup since it has the following implications in our computation process. First, we have seen in the left panel of Figure 1 that if the noise is sparse, the \( L_2E \) problem may be ill-posed with no finite infimum. Theorem 4.1 assures us that if we place an upper bound on \( \eta \), we can guarantee a finite infimum. Second, when \( \tau = 0 \), it is not hard to see from (13) that the objective value will be 0. If the infimum of (14) is guaranteed to be negative, then any point with \( \tau = 0 \) is suboptimal, thus, we do not risk excluding a potential minimizer through reparameterizing \( \tau \) as \( \exp(\eta) \).

Before discussing how to compute Tucker-\( L_2E \) next, we emphasize a key feature of our approach is that it simultaneously optimizes or estimates a precision parameter as well as a latent low-rank tensor. As we saw earlier, this precision parameter controls the cutoff for when an entry is large enough to be effectively trimmed from the model fitting. As far as we are aware, Tucker-\( L_2E \) is unique in its ability to jointly optimize precision and latent low-rank tensor parameters. Other methods that employ a precision parameter treat it as a hyperparameter and consequently employ a separate estimation or setting procedure for the precision parameter. For example, Wu et al. (2017) computes an estimate of the precision parameter based on the residuals of the least squares estimate of the low-rank tensor, but this limits the recovery capability of the model as the least squares estimates are of poor quality in challenging high-rank scenarios.

### 4.3. Solution Algorithm and Implementation Details

The optimization problem in (14) involves differentiable functions of all the model parameters with a simple box constraint on \( \eta \). The high dimensionality of the parameter space renders second-order algorithms impractical. In contrast, the classic quasi-Newton method L-BFGS-B (Liu and Nocedal 1989; Byrd et al. 1995; Zhu et al. 1997) is particularly well suited for solving (14). A detailed derivation of the gradient of the objective in (14) with respect to \( A^{(n)}, \mathcal{S}, \) and \( \eta \) can be found in the supplementary materials. Since the optimization problem in (14) is nonconvex, initialization is critical. We present two initialization strategies in this article. The first is a simple strategy from Filipović and Jukić (2015):

The second is a popular initialization procedure in the recent tensor completion literature, which we call spectral initialization with diagonal deletion, see for example Cai et al. (2019), Xia and Yuan (2019), Xia, Yuan, and Zhang (2021), and Tong et al. (2022). In our experiments, we find that this initialization procedure offers some improvement over Algorithm 1 when the underlying tensor has low CP-rank and a large percentage of tensor entries is missing. However, we note that in most cases, Algorithm 1 works similarly or better, especially when the underlying Tucker-rank is relatively high. Therefore, our default initialization procedure is Algorithm 1 in our following experiments unless explicitly specified otherwise. We discuss the details of the second initialization strategy in the supplementary materials.
Algorithm 1 Mean Imputation + HOOI/HOSVD

Require: $X, W \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$, $(r_1, r_2, \ldots, r_N)$.
1: Impute missing entries with the mean of the observed entries of $X$.
2: Compute $\mathcal{G}_0$ and $A_0^{(n)}$ via HOOI/HOSVD of $X$ with rank $(r_1, r_2, \ldots, r_N)$.
3: return $(\mathcal{G}_0, A_0^{(n)})$.

Algorithm 2 Tucker-L2E

Require: $X, W \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$, $(r_1, r_2, \ldots, r_N)$, and $\eta_{\max}$.
1: Calculate the MAD of the observed entries of $X$, denoted as $s$.
2: Rescale $X$ as $\frac{1}{10s} X$.
3: Compute initial estimates $(\mathcal{G}_0, A_0^{(n)})$ with Algorithm 1 or spectral initialization with diagonal deletion.
4: Set $\eta_0 = \log(0.01)$.
5: Using (14) as the objective and $(\mathcal{G}_0, A_0^{(n)}, \eta_0)$ as the initial value, update $(\mathcal{G}, A^{(n)}, \eta)$ with L-BFGS-B until convergence of objective value or the maximum number of iterations is reached. The final iterate is denoted as $(\mathcal{G}_*, A_*^{(n)}, \eta_*)$.
6: return $(10s\mathcal{G}_*, A_*^{(n)}, \eta_*)$.

We find the initial value of $\eta$ to have minimal impact on the solution, consequently we initialize $\eta$ using a small value $\log(0.01)$, where log is the natural log function. We also observe that numerical issues can occur when the tensor entries are somewhat large in magnitude due to the exponent function in the objective of (14). Scientific computing languages compute $\exp(-x)$ via a power series so that the numeric precision degrades when $x$ becomes larger in absolute value. Therefore, as a pre-processing step, the observed tensor entries are rescaled to have a mean absolute deviation (MAD) of 0.1. We revert the estimated tensor to the original scale after the decomposition is complete. Another practical concern is the choice of Tucker-rank $(r_1, r_2, \ldots, r_N)$ and the upper bound $\eta_{\max}$. In particular, for the upper bound $\eta_{\max}$ we want to select a value such that we have sufficient robustness but the loss landscape is still smooth enough for L-BFGS-B to find a good solution. In Sections 5.3 and 6.1, we demonstrate that Tucker-rank $(r_1, r_2, \ldots, r_N)$ can be selected in a data-driven manner using cross-validation or hold-out validation if computation is intensive. For $\eta_{\max}$, we find that $\log(50)$ works well for a wide range of problems and we have used it for all of our experiments except the feature extraction application in Section 6.3 where $\eta_{\max}$ is set at log(20) for optimal performance. It can also be tuned along with the Tucker-rank using cross-validation or hold-out validation if warranted or desired.

Algorithm 2 summarizes our procedure for computing Tucker-L2E. We implement our algorithm in the MATLAB R2021a computing environment. We use the Tensor Toolbox for MATLAB version 3.2.1 (Bader and Kolda 2006, 2008) for basic tensor classes and operations. We also use the implementation of HOOI and HOSVD in the Tensor Toolbox to compute the initial estimate of $\mathcal{G}$ and $A^{(n)}$ in Algorithm 1. We use the implementation of L-BFGS-B by Becker (2015). We direct readers to Byrd et al. (1995) for the algorithmic details of L-BFGS-B. We note that since L-BFGS-B is a gradient-based local optimization algorithm, Algorithm 2 is only able to find a locally optimal or critical point of (14). Additionally, since L-BFGS-B is a descent method (Byrd et al. 1995) and Theorem 4.1 ensures that our objective in (14) is bounded below, the sequence of objective function evaluations over the iterate sequence is guaranteed to converge with L-BFGS-B updates.

5. Numerical Experiments

We consider both the CP model and the Tucker model in our simulation studies. The tensor dimension is set at (50, 50, 50). For the CP model, the entries of the factor matrices are independently drawn from $\mathcal{N}(0, 1)$. For the Tucker model, we adopt a similar data generation protocol to Cai, Li, and Xia (2022). A tensor of size (50, 50, 50) with random normal entries is first generated. Then the tensor is truncated to have Tucker-rank $(r, r, r)$ with HOSVD. We use relative error, defined as $\text{re} = \|\mathcal{L} - \mathcal{L}\|_F/\|\mathcal{L}\|_F$, as the primary goodness of fit metric. After generating the low-rank tensor $\mathcal{L}$, we randomly select a fraction $\delta$ of the tensor entries to be corrupted with outliers drawn from Unif$[-M, M]$. We use a relatively large magnitude $M = 10\text{std}(\text{vec}(\mathcal{L}))$ in the following experiments. Optionally, a layer of dense Gaussian noise $\mathcal{E}$ can also be added, whose scale is set such that $\|\mathcal{E}\|_F/\|\mathcal{L}\|_F = 0.1$. We use CP-OPT, HOOI, BRTF, HoRPCA-S, HoRPCA-C and RGrad as the baseline methods in this section. We use the implementation of CP-OPT and HOOI in the Tensor Toolbox. The software of HoRPCA-S, HoRPCA-C and BRTF can be found on the authors’ websites. The software of RGrad can be found in the supplementary materials of Cai, Li, and Xia (2022). We provide code and demo examples for our proposed method at https://github.com/qhengncsu/TuckerL2E.

5.1. Evaluating Robustness Versus Rank

The inverse problem of robust Tucker decomposition becomes more challenging as the underlying Tucker rank or outlier percentage increases. In this section, we contrast the recovery performance of the baseline methods and Tucker-L2E by generating third-order tensors with increasing CP or Tucker-rank ($R = 5, 10, \ldots, 45$ or $R = (5, 5, 5), (10, 10, 10), \ldots, (45, 45, 45)$), under outlier corruption and in the presence or absence of dense noise. Note that for a tensor with an underlying CP-rank of $r$, we can still compute a Tucker decomposition with Tucker-rank $(r, r, r)$ to reconstruct the tensor. In this section we keep the outlier sparsity at 25%. For HoRPCA-S, we tune the penalty parameter with the ground truth. For rank-constrained formulations, the specified CP or Tucker-ranks are set to the true ranks. We provide additional details of parameter settings in the supplementary materials.

Figure 2 shows that the convex penalized formulation HoRPCA-S works well when the tensor rank is low but loses
accuracy as the rank increases. HoRPCA-C, RGrad, and Tucker-L2E demonstrate competitive recovery performance in most cases, except when the rank is too close to the data dimension, particularly at rank 35–45. Notably, Tucker-L2E appears to be able to tolerate a higher Tucker-rank than HoRPCA-C and RGrad, both in the CP model and the Tucker model. More specifically, Tucker-L2E is able to provide a reasonably good reconstruction at rank 35 and 40 while HoRPCA-C and RGrad break down.

5.2. Phase Transition of Rank and Outlier Sparsity

In the previous section, we saw that compared with HoRPCA-C and RGrad, Tucker-L2E appears to be able to tolerate a higher rank at the given outlier percentage. Although RGrad handles dense noise better than HoRPCA-C, HoRPCA-C appears to be a slightly stronger baseline when it comes to stability in high-rank scenarios. To further investigate the phase transition behavior of HoRPCA-C and Tucker-L2E, we generate tensors with high rank (\( R = 25, 27, \ldots, 45 \)) and vary the percentage of outliers taking \( \delta = 0, 0.05, \ldots, 0.5 \). Full observations are used and no dense noise is added. The specified Tucker-ranks are set to be equal to the true CP or Tucker-ranks.

Figure 3 shows that as the underlying tensor rank increases, the percentage of outliers that can be tolerated decreases for both methods. However, at a given rank, Tucker-L2E can often handle a greater level of corruption. At rank 25, both methods are able to obtain an accurate reconstruction for any outlier percentage no greater than 0.5. From rank 29 to 41, Tucker-L2E can generally handle 10%–20% more outliers. At rank 43 or 45, Tucker-L2E can handle 5% more outliers. Interestingly, the advantage of Tucker-L2E over HoRPCA-C is notably more significant on data generated by the CP model, especially at ranks 43 and 45. This is potentially because a CP-rank of 45 is a more constrained low-rank structure than a general Tucker-rank of (45,45,45).

5.3. Rank Misspecification and Cross-Validation

In the previous two sections, we set the specified ranks to be equal to the true ranks for the rank-constrained formulations. In practice, however, such prior knowledge of tensor rank may
not always be available. In this section, we investigate how rank-constrained formulations behave when the tensor rank is underestimated or overestimated. We also demonstrate the application of cross-validation to choose the appropriate rank for HoRPCA-C and Tucker-L2E. We consider three scenarios: (a) the noiseless tensor has CP-rank 15; (b) the noiseless tensor has Tucker-rank (30,10,5); (c) the noiseless tensor has Tucker-rank (35,35,35). After generating the true low-rank tensor, 25% of tensor entries are corrupted with outliers. The specified CP-ranks are 5, 10, \ldots, 45 and the specified Tucker-ranks are (5, 5, 5), (10, 10, 10), \ldots, (45, 45, 45). Notice that although in the second scenario, the noiseless tensor is not equally low-rank in every mode, we still set the specified Tucker-ranks to be equal in every mode simply to limit the number of Tucker-rank tuples that we need to consider.

The cross-validation scheme can be described as follows: the tensor entries are randomly split into 10 folds; robust tensor decomposition methods (HoRPCA-C and Tucker-L2E) are applied to 9 out of 10 folds, treating the hold-out fold as missing data. This process is repeated for each train/test split; we use the estimated values for the hold-out fold to form a new tensor which we call the predicted tensor; cross-validation error is computed as the MAD between entries of the predicted tensor and entries of the original noisy tensor. The MAD is chosen over the more common mean squared error (MSE) to make the cross-validation error less sensitive to large residuals, which likely coincide with outlying entries.

Figure 4 shows that the nonrobust methods (CP-OPT and HOOI) will greatly overfit to the outliers if the tensor rank is overestimated. For the first scenario \((R = 15)\), both HoRPCA-C and Tucker-L2E exhibit a certain level of overfitting resistance. Remarkably, Tucker-L2E remains unaffected by outliers even if the Tucker-rank is grossly overestimated to be (45,45,45). In the second scenario \((R = (30,10,5))\), unlike Tucker-L2E, HoRPCA-C is not able to achieve perfect recovery if the Tucker-rank is specified to (30,30,30). The first two scenarios demonstrate that Tucker-L2E is more robust to rank overestimation than HoRPCA-C. The third scenario \((R = (35,35,35))\) is chosen to be challenging. Contrary to our expectation, the relative error and cross-validation error for Tucker-L2E will first increase before it decreases. It is reassuring that the cross-validation error still achieves its minimum at the true Tucker-rank. Another surprising observation is that HoRPCA-C attains its minimum relative error at Tucker-rank (30, 30, 30) instead of...
the true rank (35, 35, 35). This is likely because while at Tucker-rank (30, 30, 30), HoRPCA-C is only capable of an approximate reconstruction, it is still better than the true Tucker-rank (35, 35, 35) where HoRPCA-C becomes unstable.

### 5.4. Varying Degrees of Missingness

In this section we investigate the recovery performance of HoRPCA-S, HoRPCA-C and Tucker-L2E under varying degrees of missingness. RGrad is not considered in this section since its current form does not allow missing entries. We generate (50, 50, 50) tensors with CP-rank 15, 30 or Tucker-rank (15, 15, 15), (30, 30, 30). After generating the low-rank tensor, 25% of the entries are corrupted with outliers and dense noise of relative scale 0.1 is added. Then $\rho \times 100\%$ of the tensor entries are set to be missing. The missingness is assumed to be completely random and independent from the outlier corruption. We vary the missing ratio $\rho$ from 0.05 to 0.9 in this section. We present the recovery results of both initialization methods.

In Figure 5, “Tucker-L2E Diagonal Deletion” refers to the recovery results using spectral initialization with diagonal deletion and “Tucker-L2E Mean Imputation” refers to the recovery results using Algorithm 1 as the initialization. We can see that the diagonal deletion procedure offers some improvement over Algorithm 1 when the underlying tensor is of low CP-rank. Tucker-L2E appears to be less stable than HoRPCA-C when the rank is low and the missing percentage is very high (over 80%). This may be attributed to the fact that HoRPCA-C models missing data with an equality constraint so that the unobserved entries are still penalized for having a large magnitude, while for Tucker-L2E the unobserved entries are masked and unconstrained. When the rank is relatively high, we can see that Tucker-L2E still enjoys an empirical advantage over HoRPCA-C in terms of recovery capability.

### 6. Real Data Applications

#### 6.1. Tensor Denoising on 3D fMRI Data

We consider a 3D MRI dataset INCISIX from the OsiriX repository, which contains 166 slices through a human brain, each having dimension $512 \times 512$. The dataset was first analyzed in Gandy, Recht, and Yamada (2011) from a tensor completion perspective by randomly setting voxels to be missing. We approach this dataset from a tensor denoising perspective. Following Goldfarb and Qin (2014), we extract the first 50 slices and downsample each of them to have size $128 \times 128$. Therefore, the noiseless tensor $X$ has dimension $128 \times 128 \times 50$. We then corrupt 25% of the tensor entries with outliers/noise drawn from $\text{Unif}[0, 2\text{std}(\text{vec}(X))]$. In addition to HoRPCA-C and RGrad, we consider a classic low-tubal-rank robust tensor recovery method called tensor robust principal component analysis (Lu et al. 2019) as another baseline. Scree plots of the different matrix unfoldings revealed that the mode-3 singular values decay rapidly, which indicates that the data

---

3https://www.osirix-viewer.com
Figure 5. Recovery results under varying degrees of missingness. Outlier sparsity is set at 25%. Data points are averaged over 20 random replicates. Average relative errors larger than 1 are capped at 1 to suit the display.

Figure 6. tRPCA, HoRPCA-C, RGrad and Tucker-L2E.

Table 1. Hold-out MAD for HoRPCA-C and Tucker-L2E at different Tucker-ranks.

| Tucker-rank | HoRPCA-C Hold-out MAD | Tucker-L2E Hold-out MAD |
|-------------|------------------------|------------------------|
| (64,64,10)  | 223.16                 | 236.51                 |
| (96,96,15)  | 216.85                 | 211.12                 |
| (128,128,20)| 276.60                 | 206.60                 |

tensor is approximately low-rank along mode-3. Therefore, we considered the following three Tucker-rank tuples, (64,64,10), (96,96,15) and (128,128,20), as candidate Tucker-rank tuples for HoRPCA-C, RGrad and Tucker-L2E. For RGrad, the proposed BIC criterion selected the parameters $R = (96,96,15)$, $\alpha = 0.27$ and $\mu_0 = 1$. For HoRPCA-C and Tucker-L2E, we use a hold-out validation approach to identify the best Tucker-rank tuple. We randomly sample 10% of the entries as the validation set and use the remaining 90% of entries to impute the missing 10% of entries. Then we can use the MAD between the imputed values and the actual values to determine the appropriate Tucker-rank. From Table 1, we see that the best performing Tucker-rank for HoRPCA-C is $(96,96,15)$, while for Tucker-L2E it is $(128,128,20)$. We then reapply HoRPCA-C and Tucker-L2E with the selected ranks to the fully observed tensor.

We visualize the recovery results of tRPCA, HoRPCA-C, RGrad and Tucker-L2E in Figure 6. The advantage of Tucker-L2E over HoRPCA-C and RGrad is that at rank (128,128,20), HoRPCA-C and RGrad overfit to the sparse noise while Tucker-L2E remains largely unaffected, in line with our observations in Section 5. By remaining stable at a larger rank, Tucker-L2E is able to capture more structural information, resulting in the smallest relative error and the best perceptual quality.

6.2. PARAFAC Analysis of Fluorescence Data

Parallel Factor analysis (PARAFAC) is a widely used tool in Chemometrics for decomposing fluorescence excitation-emission matrices (EEMs) into their underlying chemical components (Murphy et al. 2013). The CP model is particularly suitable for the analysis of EEMs since this type of data mostly conforms to a trilinear structure due to Beer’s law, which states that absorbance is the product of molar concentration, molar absorption coefficient, and optical path length. Certain regions of the fluorescence landscape, however, may be corrupted by Raman and Rayleigh scattering. Therefore, EEM data is a
natural candidate for the low CP-rank tensor plus sparse outliers model.

We consider a standard EEM dataset, the Dorrit data, originally introduced in Baunsgaard (1999). We use a preprocessed version4 (Riu and Bro 2003) of the Dorrit data, which consists of 27 mixed samples containing different concentrations of hydroquinone, tryptophan, phenylalanine, and dopa. Each sample has 121 emission wavelengths (241–481 nm) and 24 excitation wavelengths (200–315 nm). Following Riu and Bro (2003) and Goldfarb and Qin (2014), we exclude samples 2, 3, 5, and 10 as well as data corresponding to excitation wavelengths from 200 nm to 225 nm since this portion of the data is believed to be noisy for reasons other than scattering and amounts to slice-wise corruption, which greatly affect the global properties of the data. Therefore, the tensor data to be analyzed have dimension $23 \times 121 \times 18$. The truncated fluorescence landscape of sample 1 is visualized in Figure 7. We set the CP-rank to 4 and the Tucker-rank to (4, 4, 4) since we have prior knowledge that there are 4 pure compounds in the samples.

Figure 8 displays the recovered emission/excitation loadings (mode-2 and mode-3 CP factors) produced by CP-OPT, RGrad, HoRPCA-C, and Tucker-L2E. The last three methods are in fact applied to denoise the tensors. The CP factors are then extracted by applying CP-OPT to the reconstructed tensors. We assign the CP factors to the four analytes based on proximity to the pure component emission/excitation profiles presented in Baunsgaard (1999). For this dataset, with rank fixed at (4, 4, 4), BIC suggests $\alpha = 0.11$ and $\mu_0 = 5$ for RGrad. This has an interesting implication that there are approximately 11% of entries that are affected by scattering. The emission/excitation loadings produced by HoRPCA-C and Tucker-L2E appear more similar to the pure component profile than the ones produced by RGrad. Although the difference between HoRPCA-C and Tucker-L2E is minuscule, the emission/excitation loadings of phenylalanine (purple lines in Figure 8) produced by Tucker-L2E appear to be slightly more regular than those produced by HoRPCA-C.

---

4http://www.models.life.ku.dk/dorrit
6.3. Feature Extraction for Classification

Tucker decompositions are useful for extracting features from high-dimensional multi-way datasets for classification. The extracted features can then be used as input to standard classifiers such as k-nearest-neighbors (k-NN) or support vector machines (SVM). In this section we adopt the feature extraction framework based on Tucker decomposition originally presented in Phan and Cichocki (2010). Chachlakis, Prater-Bennette, and Markopoulos (2019) demonstrated that if the dataset is corrupted with sparse noise, strategic dimensionality reduction by a Tucker decomposition can reduce the impact of noise and leads to improved classification accuracy. Below we briefly describe the feature extraction framework. Suppose that we have \( K_1 \) tensor-valued training samples of size \( I_1 \times I_2 \times \cdots \times I_N \), which can be classified into \( C \) categories. We concatenate training samples across mode \( N + 1 \) to obtain \( X_1 \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N \times K_1} \), which we call the “training tensor.” Then \( X_1 \) is Tucker decomposed with rank \((d_1, d_2, \ldots, d_N, K_1)\) to obtain the factor matrices \( U_n \in \mathbb{R}^{I_n \times d_n} \) for \( n \in [N] \). We compress the training samples as follows:

\[
Z_1 = X_1 \times_1 U_1^T \times_2 U_2^T \times_3 \cdots \times_N U_N^T \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_N \times K_1}.
\]

Then \( Z_1 \) is matricized in mode \( N + 1 \) to become a data matrix \( Z_1 \) of size \( K_1 \times \prod_{n=1}^{N} d_n \) with labeled rows. We similarly concatenate testing samples to obtain the “testing tensor” \( X_2 \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_N \times K_2} \), which we compress to obtain

\[
Z_2 = X_2 \times_1 U_1^T \times_2 U_2^T \times_3 \cdots \times_N U_N^T \in \mathbb{R}^{d_1 \times d_2 \times \cdots \times d_N \times K_2},
\]

which is then matricized in mode \( N + 1 \) to become \( Z_2 \in \mathbb{R}^{K_2 \times \prod_{n=1}^{N} d_n} \). A suitable classifier is then trained on \( Z_1 \) and tested on \( Z_2 \). We apply the above feature extraction framework, with an added aspect of robustness, to image classification. We consider two classic image classification datasets, namely MNIST (Deng 2012) and COIL-20 (Nene, Nayar, and Murase et al. 1996), which are also studied in Phan and Cichocki (2010). MNIST consists
of 28 × 28 images of hand-written digits. COIL-20 consists of 128 × 128 images for 20 different objects with each object having 72 images. Figure 9 depicts samples of images from both datasets.

As a preprocessing step, we remove the four-pixel padding for MNIST since those regions contain no information. Images from COIL-20 are downsampled to 32 × 32 to speed up computation. To construct a training sample, we randomly sample 50 images for each digit from MNIST and 20 images for each object from COIL-20. The training images are then corrupted with salt-and-pepper noise added to a randomly selected sample of 25% of the pixels. Thus, for MNIST, \(X_1\) has dimension 20 × 20 × 500 while for COIL-20, \(X_1\) has dimension 32 × 32 × 400. We set \(d_1 = d_2 = d\) so that the total number of features is \(d^2\). We randomly sample another 500 images for each digit and use the remaining 52 images for each object as the testing points. The classifier of choice is nearest-neighbor. We repeat the described procedure on 50 different random realizations of training and testing sets. The Tucker decomposition methods considered here are HOOI, HoRPCA-C, and Tucker-L2E. In particular, HoRPCA-C is only used to approximate the training tensor. The factor matrices are obtained by applying HOOI to the output tensor of HoRPCA-C.

Figure 10 highlights that by applying Tucker decomposition for feature extraction and dimensionality reduction, all methods achieve a substantial gain in accuracy compared with directly using the corrupted training images. As \(d\) increases, initially the accuracy of all Tucker decomposition methods will increase due to being able to create more meaningful features. However, eventually the feature extraction framework will overfit to the sparse noise and the accuracy starts decreasing. The testing accuracy on MNIST is generally much lower than on COIL-20 despite having fewer categories and more training images per category. This suggests that MNIST is a more challenging dataset to classify. Tucker-L2E again exhibits greater stability in high-rank scenarios, especially in the case of COIL-20 with its accuracy steadily increasing throughout the range of \(d\) that we investigated. When \(d\) is small, Tucker-L2E may not be advantageous. However, as the number of features increases, the best attainable accuracy of Tucker-L2E outperforms that of HOOI and HoRPCA-C.

7. Conclusion

This article describes a new formulation of the robust Tucker decomposition problem based on the \(L_2\) criterion, Tucker-L2E. We present two initialization strategies and a solution algorithm based on L-BFGS-B. Numerical experiments and real data applications demonstrate that Tucker-L2E exhibits stronger recovery capability in challenging high-rank scenarios compared with existing alternatives. This empirical property is useful since real-world tensors are often nearly low-rank instead of perfectly low-rank. By remaining stable at a higher rank, Tucker-L2E is able to provide a more expressive reconstruction of the underlying low-rank tensor in the presence of sparse perturbations.

In this article we used an off-the-self local optimization algorithm L-BFGS-B as the main computational tool. A projected-gradient type algorithm will likely have a smaller memory footprint, which presents an interesting venue for future work. We also note that the proposed robust tensor recovery paradigm can be adapted to other formats of low-rank tensor recovery, for example the low-tubal-rank format and the tensor-train format, with suitably designed computation algorithms.

Supplementary Materials

Supplement: A pdf file that contains derivation of gradient, an alternative initialization strategy named spectral initialization with diagonal deletion, proof of Theorem 4.1, details of parameter choices, and a run time comparison with the baseline methods.
Software: Matlab code of the described method, along with scripts to reproduce some of the figures in Sections 5 and 6.

Acknowledgments
We are grateful to the associate editor, editor, and anonymous referee for their valuable comments and suggestions, which greatly improved the presentation of this article. We thank Haixu Ma and Xiaojian Liu for their assistance in testing the software.

Disclosure Statement
The authors report there are no competing interests to declare.

Funding
This research was partially funded by grants from the National Institute of General Medical Sciences (R01GM135928: EC, R01GM126550: YL) and the National Science Foundation (DMS-2201136: EC, DMS-2100729: YL).

ORCID
Qianga Heng  http://orcid.org/0000-0002-4042-6773
Eric C. Chi  http://orcid.org/0000-0003-4647-0895
Yufeng Liu  http://orcid.org/0000-0002-1686-0545

References
Acar, E., Dunlavy, D. M., and Kolda, T. G. (2011), “A Scalable Optimization Approach for Fitting Canonical Tensor Decompositions,” Journal of Chemometrics, 25, 67–86. [539]
Acar, E., Dunlavy, D. M., Kolda, T. G., and Morup, M. (2011), “Scalable Tensor Factorizations for Incomplete Data,” Chemometrics and Intelligent Laboratory Systems, 106, 41–56. [540]
Anandkumar, A., Jain, P., Shi, Y., and Niranjan, U. N. (2016), “Tensor vs. Matrix Methods: Robust Tensor Decomposition Under Block Sparse Perturbations,” in Artificial Intelligence and Statistics, PMLR, pp. 268–276. [540]
Bader, B. W., and Kolda, T. G. (2006), “Algorithm 862: MATLAB Tensor Classes for Fast Algorithm Prototyping,” ACM Transactions on Mathematical Software (TOMS), 32, 653–655. [543]
——— (2008), “Efficient MATLAB Computations with Sparse and factored Tensors,” SIAM Journal on Scientific Computing, 30, 205–231. [543]
Baunsgaard, D. (1999), “Factors Affecting 3-way Modelling (PARAFAC) of Fluorescence Landscapes,” Internal Report, Department of Dairy and Food Science, The Royal Veterinary and Agricultural University Denmark. [548]
Becker, S. (2015), “L-BFGS-B-C,” GitHub, available at https://github.com/stephenbeck/L-BFGS-B-C. [543]
Boyd, S., Parikh, N., Chu, E., Peleato, B., Eckstein, J., et al. (2011), “Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers,” Foundations and Trends® in Machine Learning, 3, 1–122. [540]
Byrd, R. H., Lu, P., Nocedal, J., and Zhu, C. (1995), “A Limited Memory Algorithm for Bound Constrained Optimization,” SIAM Journal on Scientific Computing, 16, 1190–1208. [542,543]
Cai, C., Li, G., Poor, H. V., and Chen, Y. (2019), “Nonconvex Low-Rank Tensor Completion from Noisy Data,” in Advances in Neural Information Processing Systems (Vol. 32). [540,542]
Cai, C., Poor, H. V., and Chen, Y. (2023), “Uncertainty Quantification for Nonconvex Tensor Completion: Confidence Intervals, Heteroscedasticity and Optimality,” IEEE Transactions on Information Theory, 69, 407–452. [540]
Cai, J.-F., Li, J., and Xia, D. (2022), “Generalized Low-Rank Plus Sparse Tensor Estimation by Fast Riemannian Optimization,” Journal of the American Statistical Association, DOI: 10.1080/01621459.2022.2063131. [540,543]
Candès, E. J., Li, X., Ma, Y., and Wright, J. (2011), “Robust Principal Component Analysis? Journal of the ACM, 58, 1–37. [539]
Lu, C., Feng, J., Chen, Y., Liu, W., Lin, Z., and Yan, S. (2019), “Tensor Robust Principal Component Analysis with a New Tensor Nuclear Norm,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42, 925–938.

Ma, J., Qiu, W., Zhao, J., Ma, Y., Yuille, A. L., and Tu, Z. (2015), “Robust $L_2/E$ Estimation of Transformation for Non-rigid Registration,” *IEEE Transactions on Signal Processing*, 63, 1115–1129. [538]

Phan, A. H., and Cichocki, A. (2010), “Tensor Decompositions for Feature Extraction and Classification of High Dimensional Datasets,” *PARAFAC*,” *Analytical Methods*, 5, 6557–6566. [547]

Eseledets, I. V. (2011), “Tensor-Train Decomposition,” *SIAM Journal on Scientific Computing*, 33, 2295–2317. [537]

Ramos, J. J. (2014), “Robust Methods for Forecast Aggregation,” Ph.D. thesis. [537]

Scott, A. I. (2006), “Denoising by Wavelet Thresholding Using Multivariate Minimum Distance Partial Density Estimation,” Ph.D. thesis. [538]

Scott, D. W. (2001), “Parametric Statistical Modeling by Minimum Integrated Square Error,” *Technometrics*, 43, 274–285. [537,538,541]

— (2009), “The L2E Method,” *Wiley Interdisciplinary Reviews: Computational Statistics*, 1, 45–51. [537,538,541]

Terrell, G. R. (1990), *Linear Density Estimates*, Department of Statistics, Virginia Polytechnic Institute and State University. [537]

Tomioka, R., Suzuki, T., Hayashi, K., and Kashima, H. (2011), “Statistical Performance of Convex Tensor Decomposition,” in *Advances in Neural Information Processing Systems* (Vol. 24), pp. 972–980. [539]

Tong, T., Ma, C., Prater-Bennette, A., Tripp, E., and Chi, Y. (2022), “Scaling and Scalability: Provable Nonconvex Low-Rank Tensor Estimation from Incomplete Measurements,” *Journal of Machine Learning Research*, 23, 1–77. [540,542]

Tucker, L. R. (1966), “Some Mathematical Notes on Three-Mode Factor Analysis,” *Psychometrika*, 31, 279–311. [537,539]

Wu, Y., Tan, H., Li, Y., Li, F., and He, H. (2017), “Robust Tensor Decomposition based on Cauchy Distribution and its Applications,” *Neurocomputing*, 223, 107–117. [540,542]

Zhang, A. (2019), “Cross: Efficient Low-Rank Tensor Completion,” *The Annals of Statistics*, 47, 936–964. [540]

Zhao, Q., Zhou, G., Zhang, L., and Cichocki, A. (2016), “Tensor Ring Decomposition,” arXiv preprint arXiv:1606.05535. [537]

Zhao, Q., Zhou, G., Zhang, L., Cichocki, A., and Amari, S.-I. (2015), “Bayesian Robust Tensor Factorization for Incomplete Multiway Data,” *IEEE Transactions on Neural Networks and Learning Systems*, 27, 736–748. [540]