Covariate-Assisted Sparse Tensor Completion

Hilda S. Ibriga and Will Wei Sun

ABSTRACT
We aim to provably complete a sparse and highly missing tensor in the presence of covariate information along tensor modes. Our motivation comes from online advertising where users’ click-through-rates (CTR) on ads over various devices form a CTR tensor that has about 96% missing entries and has many zeros on nonmissing entries, which makes the standalone tensor completion method unsatisfactory. Beside the CTR tensor, additional ad features or user characteristics are often available. In this article, we propose Covariate-assisted Sparse Tensor Completion (COSTCO) to incorporate covariate information for the recovery of the sparse tensor. The key idea is to jointly extract latent components from both the tensor and the covariate matrix to learn a synthetic representation. Theoretically, we derive the error bound for the recovered tensor components and explicitly quantify the improvements on both the reveal probability condition and the tensor recovery accuracy due to covariates. Finally, we apply COSTCO to an advertisement dataset consisting of a CTR tensor and ad covariate matrix, leading to 23% accuracy improvement over the baseline. An important by-product is that ad latent components from COSTCO reveal interesting ad clusters, which are useful for better ad targeting. Supplementary materials for this article are available online.

1. Introduction
Low-rank tensor completion aims to impute missing entries of a partially observed tensor by forming a low-rank decomposition on the observed entries. It has been widely used in various scientific and business applications, including recommender systems (Symeonidis, Nanopoulos, and Manolopoulos 2008), neuroimaging analysis (Zhou, Li, and Zhu 2013), social network analysis (Jing et al. 2020), personalized medicine (Wang et al. 2019), and time series analysis (Chen, Yang, and Zhang 2019). We refer to the recent surveys on tensors for more real applications (Song et al. 2019; Bi et al. 2020). In spite of its popularity, it is also well known that when the missing percentage of the tensor is very high, a standalone tensor completion method often fails at yielding desirable recovery results. Fortunately, in many real applications, we also have access to some side covariate information. In this article, we aim to complete a sparse and highly-missing tensor in the presence of covariate information along tensor modes.

Our motivation originates from online advertising applications, where advertisement (ad) information is usually described by both users’ click behavior data and ad characteristics data. More formally, the users’ click data refer to as the click-through rate (CTR) of the ads, quantifying the user click behavior on different ads, various platforms, different devices or over time etc. The CTR data is therefore, often represented as a tensor of three or four modes, for example, the user $\times$ ad $\times$ device tensor shown in Figure 1. The ad characteristic data on the other hand is usually represented in the form of a matrix which contains context information for each ad. Typically in online advertising not all users are presented with all ads, thus, creating many missing data in the CTR tensor. Moreover, users typically engage with a small subset of the ads that are presented to them. Low rates of ads engagement is a common phenomenon in online advertising which begets a highly sparse CTR tensor (many zero entries) with high percentage of missing entries. For instance, in our real data shown in Section 6, the ad CTR tensor has 96% missing entries and is highly sparse with only 40% of the revealed entries being nonzero. We show in Sections 5 and 6 that methods using a standalone tensor completion often fail at recovering the missing entries of a tensor with such missing percentage. On the contrary the ad characteristic data is usually relatively complete and dense. It therefore, becomes advantageous to incorporate the ad characteristic information in a model to recover the missing entries of the CTR tensor. The structure of the sparse CTR tensor with missing entries coupled with the ad characteristic data is illustrated in Figure 1. As shown in Figure 1 the two sources of data; CTR tensor and ad covariates matrix are coupled along the ad mode.

In this article, we propose Covariate-assisted Sparse Tensor Completion (COSTCO) to recover missing entries in highly sparse tensor with a large percentage of missing entries. Under the low-rank assumption on both the tensor and the covariate matrix, we assume the latent components corresponding to the coupled mode are shared by both the tensor and matrix decomposition. This model encourages a synthetic representation of the coupled mode by leveraging the additional covariate information.
information into tensor completion. Another advantage of our COSTCO is that it naturally handles the cold-start problem. For a new ad, the CTR tensor itself provides no information to estimate the corresponding CTR entries. Hence, existing standalone tensor completion based methods are not directly applicable. In contrast, our COSTCO solves this issue by incorporating additional ad covariate information. The intuition behind it is that the ad covariate matrix provides a reasonable cluster structure of ads. Therefore, the missing clicking behaviors on a new ad can be learned from the shared latent components estimated based on both the CTR tensor and the ad covariate matrix. Similarly, the cold-start problem can be addressed for a new user when we have a user covariate matrix. In algorithm, we formulate the parameter estimation as a nonconvex optimization with sparsity constraints, and propose an efficient sparse alternating least squares approach with an extra refinement step. Our algorithm jointly extracts latent features from both tensor and the covariate matrix and uses covariate information to improve the recovery accuracy of the recovered tensor components. We showcase through extensive numerical studies that our COSTCO is able to successfully recover entries for a tensor even with 98% missing entries.

In addition to the above methodological contributions, we also make theoretical contributions to the understanding of how side covariate information affects the performance of tensor completion. In particular, we derive the nonasymptotic error bound for the recovered tensor components and explicitly quantify the improvements on both the reveal probability condition and the tensor recovery accuracy due to additional covariate information. We show that COSTCO allows for a relaxation on the lower bound of the reveal probability \( p \) compared to that required in tensor completion with no covariates, see Assumption 6 for details. In the extreme case where all tensor modes are coupled with covariate matrices, we can still recover the tensor entries even when the reveal probability of the tensor is close to zero. Moreover, we present the statistical errors for the shared tensor component (corresponding to the coupled mode) and nonshared tensor components separately to demonstrate the gain brought in through the coupling of covariates information in the model. We show that given some mild assumptions on noise levels and condition numbers, our COSTCO guarantees an improved recovery accuracy for the shared component. Unlike existing theoretical analysis on low-rank tensors which assumes the error tensor to be Gaussian, we do not impose any distributional assumption on the error tensor or the error matrix. Our theoretical results depends on the error term only through its sparse spectral norm.

Finally, we apply COSTCO to the advertising data from a major internet company to demonstrate its practical advantages. COSTCO makes use of both ad CTR tensor and ad covariate matrix to extract the latent component which leads to 23% accuracy improvement in recovering the missing entries when compared to the standalone sparse tensor completion and 10% improvement over a covariate-assisted deep learning algorithm. Moreover, an important by-product from our COSTCO is to use the recovered ad latent components for better ad clustering. Ad clustering is an essential task for targeted advertising that helps lead useful ad recommendation for online platform users. Cluster analysis on our ad latent components reveals interesting and new clusters that link different product industries which are not formed in existing clustering methods. Such findings could directly help the marketing team to strategize the ad planning procedure accordingly for better ad targeting.

1.1. Related Work and Paper Organization

Tensor completion with side information: The simultaneous extraction of latent information from multiple sources of data can be interpreted as a form of data fusion (Acar, Kolda, and Dunlavy 2011; Acar et al. 2013; Zhou et al. 2017; Kishan, Makoto, and Hiroshi 2018; Choi, Jang, and Kang 2019; Huang, Liu, and Zhu 2020; Li, Zeng, and Zhang 2020). Among them, there are a few work related to tensor completion with side information. The most related work to our approach is the gradient-based all-at-once optimization method proposed by Acar, Kolda, and Dunlavy (2011) which updates the matrix and tensor components all at once. We compare it in our experiments and find that it is consistently inferior to our COSTCO. Zhou et al. (2017) proposed a Riemannian conjugate gradient descent algorithm to solve the tensor completion problem in the presence of side information. However, this procedure does not address the tensor completion problem in the presence of high percentage of missing entries combined with a high sparsity level. Choi, Jang, and Kang (2019) developed a fast and scalable algorithm for the estimation of shared latent features in coupled tensor matrix model. However, their approach does not allow missing entries and only works for complete data. Importantly, all the aforementioned works did not provide any theoretical analysis for their methods. Kishan,
Makoto, and Hiroshi (2018) proposed a convex coupled tensor-matrix completion method and Huang, Liu, and Zhu (2020) applied the tensor ring decomposition method on the coupled tensor-tensor problem. However, these two works do not account for noise in the tensor or matrix, that is, their model is noiseless, nor do they consider the sparse tensor case. To the best of our knowledge, our work is the first provably method that is tailored for completing a highly sparse and highly missing tensor in the presence of covariate information.

**Tensor completion with theoretical guarantees:** Our theoretical analysis is related to a list of recent theoretical work in standalone tensor completion that does not incorporate covariate information (Jain and Oh 2014; Zhang 2019; Cai et al. 2021; Xia, Yuan, and Zhang 2021). In particular, Jain and Oh (2014) provided recovery guarantee for symmetric and orthogonal tensors with missing entries, but did not explore recovery for the tensor completion with coupled covariates nor did they address the case of the nonorthogonal, noisy and sparse tensor. Zhang (2019) established a sharp recovery error for a special tensor completion problem, where the missing pattern was not uniformly missing but followed a cross structure. Xia, Yuan, and Zhang (2021) proposed a two-step algorithm (a spectral initialization method followed by the power method) for the noisy Tensor completion case and established the optimal statistical rate in low-rank tensor completion. Different from our model, they assumed the error tensor to be sub-Gaussian and their model introduce additional challenges. These make our theoretical analysis far from a simple extension to the standard tensor completion problem.

**Paper organization:** The rest of the article is organized as follows. Section 2 reviews some notations, basic definitions of algebra of tensors. Section 3 presents our model, the optimization problem and our algorithm along with procedures for initialization and parameter tuning. Section 4 presents the main theoretical results. Section 5 contains a series of simulation studies. Section 6 applies our algorithm to an advertisement dataset to illustrate its practical advantages. Interesting extensions, all proof details, lemmas and additional experiments are left in the supplementary materials.

### 2. Notation and Preliminaries

In this section, we introduce some notation, and review some background on tensors. Throughout the article we denote tensors by Euler script letters, for example, $\mathcal{T}$, $\mathcal{E}$. Matrices are denoted by boldface capital letters, for example, $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$; vectors are represented with boldface lowercase letters, for example, $\mathbf{a}$, $\mathbf{v}$, and scalars are denoted by lowercase letters, for example, $a$, $\lambda$. The $n \times n$ identity matrix $\mathbf{I}_n$ is simply written as $I$ when the dimension can be easily implied from the context.

Following Kolda and Bader (2009), we use the term tensor to refer to a multidimensional array; a concept that generalizes the notion of matrices and vectors to higher dimensions. A first-order tensor is a vector, a second-order tensor is a matrix and a third-order tensor is a three-dimensional array. Each order of a tensor is referred to as a mode. For example a matrix (second-order tensor) has two modes with mode-1 and mode-2 being the dimensions represented by the rows and columns of the matrix, respectively. Let $\mathcal{T} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ be a third-order nonsymmetric tensor. We denote its $(i, j, k)$th entry as $T_{ijk}$. A tensor fiber refers to a higher order analogue of matrix row and column and is obtained by fixing all but one of the indices of the tensor. For the tensor $\mathcal{T}$ defined above, the mode-1 fiber is given by $T_{1jk}$; the mode-2 fiber by $T_{ijk}$ and mode-3 fiber by $T_{ij}$. Next the slices of the tensor $\mathcal{T}$ are obtained by fixing all but two of the tensor indices. For example the frontal, lateral and horizontal slices of the tensor $\mathcal{T}$ are denoted as $T_{i, j}$, $T_{i, j}$, and $T_{i}$. We define three different types of tensor vector products. For vectors $\mathbf{u} \in \mathbb{R}^{n_1}, \mathbf{v} \in \mathbb{R}^{n_2}, \mathbf{w} \in \mathbb{R}^{n_3}$, the mode-1, mode-2 and mode-3, tensor-vector product is a matrix defined as a combinations of tensor slices: $\mathcal{T} \times_1 \mathbf{u} = \sum_{i=1}^{n_1} u_i T_{i, j, k}$, $\mathcal{T} \times_2 \mathbf{v} = \sum_{j=1}^{n_2} v_j T_{i, j, k}$, $\mathcal{T} \times_3 \mathbf{w} = \sum_{k=1}^{n_3} w_k T_{i, j, k}$. The tensor multiplying two vectors along its two modes is a vector defined as $\mathcal{T} \times_1 \mathbf{v} \times_3 \mathbf{w} = \sum_{j,k} v_j w_k T_{i,j,k}$, $\mathcal{T} \times_2 \mathbf{u} \times_3 \mathbf{v} = \sum_{i,j} u_i v_j T_{i,j,k}$, $\mathcal{T} \times_3 \mathbf{u} \times_2 \mathbf{v} \times_3 \mathbf{w} = \sum_{i,j,k} u_i v_j w_k T_{i,j,k}$. Finally the tensor-tensor product is a scalar defined as $\mathcal{T} \times_1 \mathbf{u} \times_2 \mathbf{v} \times_3 \mathbf{w} = \sum_{i,j,k} u_i v_j w_k T_{i,j,k}$.

We denote $\|\mathbf{M}\|$ and $\|\mathbf{M}\|_F$ to be the spectral norm and the Frobenius norm of a matrix $\mathbf{M}$, respectively. The spectral norm of a tensor $\mathcal{T}$ is defined as

$$
\|\mathcal{T}\| := \sup_{\|\mathbf{u}\|_2 = \|\mathbf{v}\|_2 = \|\mathbf{w}\|_2 = 1} |\mathcal{T} \times_1 \mathbf{u} \times_2 \mathbf{v} \times_3 \mathbf{w}|, \tag{1}
$$

and its Frobenius norm is $\|\mathcal{T}\|_F := \left(\sum_{i,j,k} T_{ijk}^2\right)^{1/2}$. Define the sparse spectral norm of a matrix $\mathbf{M}$ as $\|\mathbf{M}\|_{<d_1, d_2, d_3>} := \max_{\|\mathbf{u}\|_2 = \|\mathbf{v}\|_2 = \|\mathbf{w}\|_2 = 1} \|\mathcal{T} \times_1 \mathbf{u} \times_2 \mathbf{v} \times_3 \mathbf{w}\|_2$ and the sparse spectral norm of a tensor $\mathcal{T}$ as

$$
\|\mathcal{T}\|_{<d_1, d_2, d_3>} := \max_{\|\mathbf{u}\|_2 = \|\mathbf{v}\|_2 = \|\mathbf{w}\|_2 = 1} \|\mathcal{T} \times_1 \mathbf{u} \times_2 \mathbf{v} \times_3 \mathbf{w}\|_2,
$$

where $d_1 < n_1$, $d_2 < n_2$, $d_3 < n_3$. When $d_1 = d_2 = d_3 = d$, we simplify $\|\mathcal{T}\|_{<d_1, d_2, d_3>}$ as $\|\mathcal{T}\|_{<d, d, d>}$.

Given a third-order tensor $\mathcal{T} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, we denote its CP decomposition as

$$
\mathcal{T} = \sum_{r \in [R]} \lambda_r \mathbf{a}_r \otimes \mathbf{b}_r \otimes \mathbf{c}_r, \tag{2}
$$

where $[R]$ indicates the set of integer numbers $\{1, \ldots, R\}$, and $\otimes$ denotes the outer product of two vectors. For example, the outer product of three vectors $\mathbf{a}_r \in \mathbb{R}^{n_1}$, $\mathbf{b}_r \in \mathbb{R}^{n_2}$ and $\mathbf{c}_r \in \mathbb{R}^{n_3}$ forms a third order tensor of dimension $n_1 \times n_2 \times n_3$ whose $(i, j, k)$th entry is equal to $a_{r_i} \times b_{r_j} \times c_{r_k}$ where $a_{r_i}$ is the $i$th entry of $\mathbf{a}_r$. In (2), $\mathbf{a}_r, \mathbf{b}_r, \mathbf{c}_r$ are of unit norm; that is $\|\mathbf{a}_r\|_2 = \|\mathbf{b}_r\|_2 = \|\mathbf{c}_r\|_2 = 1$ for all $r \in [R]$; $\lambda_r \in \mathbb{R}^+$ is the $r$th decomposition weight of the tensor. We denote matrices
A ∈ ℜ^{n_1 × R}, B ∈ ℜ^{n_2 × R} and C ∈ ℜ^{n_3 × R} whose columns are a_r, b_r, and c_r, for r ∈ [R], respectively, as,

A = [a_1, a_2, ..., a_R]  B = [b_1, b_2, ..., b_R]  C = [c_1, c_2, ..., c_R].

3. Methodology

In this section we introduce our sparse tensor completion model when covariate information is available and propose a non-convex optimization for parameter estimation. Our algorithm employs an alternative updating approach and incorporates a refinement step to boost the performance.

3.1. Model

We observe a third-order tensor T ∈ ℜ^{n_1 × n_2 × n_3} and a covariate matrix M ∈ ℜ^{n_1 × n_2} corresponding to the feature information along the first mode of the tensor T. Here, without loss of generality, we consider the case where the tensor has three modes and the tensor and the matrix are coupled along the first mode. Our method can be easily extended to the case where more than one mode of the tensor has a covariates matrix. Section S.1.1 of the supplementary materials presents a general case where all tensor modes are coupled to covariate matrices.

We consider a widely used random sampling model (Jain and Oh 2014; Barak and Moitra 2016; Song et al. 2019; Xia and Yuan 2019; Cai, Poor, and Chen 2020; Zhang and Ng 2020; Xia, Yuan, and Zhang 2021; Cai et al. 2021) where the partially observed entries in the tensor are assumed to be uniformly random sampled from the original tensor. That is, let Ω be the subset of indices of the tensor T for which entries are not missing. Each index (i, j, k) of the tensor T is included in Ω independently with probability p. Next we define a projection function P_Ω(T) that projects the tensor onto the observed set Ω, such that

\[ P_Ω(T)_{ijk} = \begin{cases} T_{ijk} & \text{if } (i,j,k) \in Ω \\ 0 & \text{otherwise}. \end{cases} \]  (3)

In other words P_Ω(·) is a function that is applied element-wise to the tensor entries and indicates which entries of the tensor are missing. In this article, we assume a noisy observation model, where the observed tensor and matrix are noisy versions of their true counterparts. That is,

\[ P_Ω(T) = P_Ω(T^* + E_T); \quad M = M^* + E_M, \]  (4)

where E_T and E_M are the error tensor and the error matrix, respectively; T^* and M^* are the true tensor and the true matrix, which are assumed to have low-rank decomposition structures (Kolda and Bader 2009);

\[ T^* = \sum_{r \in [R]} \lambda_r a_r^* \otimes b_r^* \otimes c_r^*; \quad M^* = \sum_{r \in [R]} \sigma_r a_r^* \otimes v_r^*. \]  (5)

where \( \lambda_r^* \) and \( \sigma_r^* \) ∈ ℜ^+, and \( a_r^* \in ℜ^{n_1}, b_r^* \in ℜ^{n_2}, c_r^* \in ℜ^{n_3} \) and \( v_r^* \in ℜ^{n_v} \) with \( \|a_r^*\|_2 = \|b_r^*\|_2 = \|c_r^*\|_2 = \|v_r^*\|_2 = 1 \) for all \( r \in [R] \) with \( R \) representing the rank of the tensor and matrix. In this article we consider the case that the ranks of both tensor and matrix are the same in order to simplify the presentation and theoretical studies. In this case, the uniqueness of the decomposition is guaranteed (Sørensen and De Lathauwer 2015). However, when the tensor rank and the matrix rank are different, the recovery of low-rank components would become more challenging due to some indeterminacy issue (De Lathauwer and Kofidis 2017).

As motivated from the online advertisement application, we impose an important sparsity structure on the tensor and matrix components \( a_r^*, b_r^*, c_r^* \), and \( v_r^* \) such that they belong to the set \( S(n, d_i) \) with \( i = 1, 2, 3, v \), where

\[ S(n, d_i) := \left\{ u \in ℜ^{n_i} \|u\|_2 = 1, \sum_{j=1}^{n_i} 1_{\{u_j \neq 0\}} \leq d_i \right\}. \]  (6)

The values \( d_i \) for \( i = 1, 2, 3, v \) are considered to be the true sparsity parameters for the tensor and matrix latent components. Note that since the rank \( R \) is typically very small in low-rank tensor models, the sum of sparse rank-1 tensors in (5) still leads to a sparse tensor. To illustrate it, suppose each component \( a_r^*, b_r^*, c_r^* \) is sparse with only 10% nonzero elements, that is, \( d_1 = 0.1n_1 \), then the tensor \( T^* \) has at most \( R \times 0.001 \times n_1 n_2 n_3 \) nonzero entries. In this case, \( T^* \) is sparse as long as the rank \( R \) is not too large.

Given a tensor \( T \) with many missing entries and a covariate matrix \( M \), our goal is to recover the true tensor \( T^* \) as well as its sparse latent components. We formulate the model estimation as a joint sparse matrix and tensor decomposition problem. This comes down to finding a sparse and low-rank approximation to the tensor and matrix that are coupled in the first mode.

\[ \min_{A,B,C,V,\lambda,\sigma} \left\{ \|P_Ω(T) - P_Ω \left( \sum_{r \in [R]} \lambda_r a_r \otimes b_r \otimes c_r \right) \|_F^2 + \|M - \sum_{r \in [R]} \sigma_r a_r \otimes v_r \|_F^2 \right\} \]

subject to \( \|a_r\|_2 = \|b_r\|_2 = \|c_r\|_2 = \|v_r\|_2 = 1 \), \( \|a_r\|_0 \leq s_1, \|b_r\|_0 \leq s_2, \|c_r\|_0 \leq s_3, \|v_r\|_0 \leq s_v \).  (7)

Here \( s_i, i = 1, 2, 3, v \), are the sparsity parameters and can be tuned via a data-driven way. It is worth mentioning that in this article we consider the case where the covariate matrix \( M \) is fully observed. When \( M \) also contains missing entries, we can employ a similar projection function to solve the optimization problem on the observed entries of \( M \). In particular, let \( Ω_M \) be the subset of indices of the matrix \( M \) for which entries are not missing, and define a projection function \( P_{Ω_M}(M) \) that projects the matrix onto the observed set \( Ω_M \). When both the tensor \( T \) and the covariate matrix \( M \) contain missing entries, the objective function in (7) can be adjusted as \( \|P_Ω(T) - P_Ω \left( \sum_{r \in [R]} \lambda_r a_r \otimes b_r \otimes c_r \right) \|_F^2 + \|P_{Ω_M}(M) - P_{Ω_M} \left( \sum_{r \in [R]} \sigma_r a_r \otimes v_r \right) \|_F^2 \). The problem in (7) is a nonconvex optimization when considering all parameters at once, however, the objective function is convex in each parameter while other parameters are fixed. Such multi-convex property motivates us to consider an efficient alternative updating algorithm.
3.2. Algorithm

In order to solve the optimization problem formulated in (7), we use an alternating least-squares (ALS) approach and incorporate an extra refinement step as introduced in Jain and Oh (2014). In each iteration of ALS, all but one of the components are fixed and the optimization problem reduces to a convex least-squares problem. In order to enforce $\ell_2$ norm penalization in the optimization, we apply a truncation step after each component update similar to that used in Sun et al. (2017), Zhang and Han (2019), and Hao, Zhang, and Cheng (2020). For a vector $\mathbf{u} \in \mathbb{R}^n$ and an index set $F \subseteq [n]$ we define $\text{Truncate}(\mathbf{u}, F)$ such that its $i$th entry is

$$[\text{Truncate}(\mathbf{u}, F)]_i = \begin{cases} \mathbf{u}_i, & \text{if } i \in F \\ 0, & \text{otherwise}. \end{cases}$$

For a scalar $s < n$, we denote $\text{Truncate}(\mathbf{u}, s) = \text{Truncate}(\mathbf{u}, \text{supp}(\mathbf{u}, s))$, where $\text{supp}(\mathbf{u}, s)$ is the set of indices of $\mathbf{u}$ which have the largest $s$ absolute values. For example, consider $\mathbf{u} = (0.1, 0.2, 0.5, -0.6)^T$, we have $\text{supp}(\mathbf{u}, 2) = \{3, 4\}$ and $\text{Truncate}(\mathbf{u}, 2) = (0, 0.5, -0.6)^T$. Note that existing sparse tensor models encourage the sparsity either via a Lasso penalized approach (Pan, Mai, and Zhang 2019), dimension reduction approach (Li and Zhang 2017), or sketching (Xia and Yuan 2021). We extend the truncation-based sparsity approach in traditional high-dimensional vector models (Wang et al. 2014; Wang, Liu, and Zhang 2014) and tensor factorization (Sun et al. 2017; Zhang and Han 2019; Hao, Zhang, and Cheng 2020) to the tensor completion problem. As shown in Wang, Liu, and Zhang (2014) and Sun et al. (2017), the truncation-based sparsity approach often leads to improved estimation performance in practice.

Our COSTCO in Algorithm 1 takes a matrix $\mathbf{M}$ and a tensor $\mathcal{T}$ with missing entries as input and computes the components of the matrix and tensor. Due to the nonconvexity of the optimization problem, there could be multiple local optima. In our algorithm we initialize the tensor and matrix components using the procedure in Section 3.2.1 which is shown through extensive simulations to provide good starting values for the tensor and matrix components. Line 6 of the algorithm has an inner loop on $r \in [R]$ which loops on each tensor rank. This inner loop on $r$ performs an "extra refinement" step that was first introduced in Jain and Oh (2014) for tensor completion; and is, therein, proved to improve the error bounds of tensor recovery.

The main component updates are performed in Lines 8 and 10 which are solutions to the least-squares problem while other parameters are fixed. Note that the horizontal double line in Lines 8 and 10 indicate element-wise fraction and the squaring in the denominator applies entry-wise on the vectors. After obtaining these nonsparse components, Lines 9 and 11 perform the truncation operator to encourage the sparsity on the latent components. The detailed derivation of this algorithm is shown in Lemma 1 in the supplementary material. Finally, the algorithm stops if either the maximum number of iterations $\tau$ is reached or the normalized Frobenius norm difference of the current and previous components are below a threshold $tol$.

Algorithm 1 handles two possible sources of identifiability issues. First, after obtaining the sparse update $\tilde{\mathbf{a}}_r, \tilde{\mathbf{b}}_r, \tilde{\mathbf{c}}_r, \tilde{\mathbf{v}}_r$, it normalizes these components by its Euclidean norm so that all factor vectors $\mathbf{a}_r, \mathbf{b}_r, \mathbf{c}_r, \mathbf{v}_r$ (Lines 9 and 13 of Algorithm 1) are scaling-identifiable. Second, when there are a few entries of the same largest absolute value in a vector, the Truncate operator in Lines 9 and 11 ensures that the same entries will be kept. To illustrate it, consider $\mathbf{u} = (0.5, 0.5, 0.5, 0.4, 0.3)^T$ and the sparsity parameter $s = 2$, $\text{Truncate}(\mathbf{u}, 2)$ always returns a sparse vector $(0.5, 0.5, 0, 0, 0)^T$, that is, only the first appear $s$ largest absolute values are kept.

Figure 2 is an illustration of COSTCO that reveals the intuition behind the working of Algorithm 1. As the percentage of missing entries in the tensor increases, recovering the tensor components using only the observed tensor entries leads to a reduction in the accuracy of the recovered tensor components. However, with COSTCO, we leverage the additional latent information coming from the covariates on the shared mode. The signal obtained from the matrix components in contributes in improving the recovery of the shared components and indirectly that of the nonshared components as well. This observation is reflected on Line 8 of Algorithm 1 for the shared component update, where we see in the denominator that even when $P_{\mathcal{I}}(\mathbf{I}, \mathbf{b}_r^2, \mathbf{c}_r^2)$ is close to zero (meaning most entries of the tensor are missing) the denominator remains a nonzero value due to the signal from the covariate matrix. In this case we are still able to estimate the shared component $\mathbf{a}_r$. This would not be the case without the addition of the covariates matrix information, where the denominator for the update would only be $P_{\mathcal{I}}(\mathbf{I}, \mathbf{b}_r^2, \mathbf{c}_r^2)$ which is close to zero. Therefore, a standalone tensor completion algorithm would become unstable. In the
3.2.2. Initialization Procedure

This section presents details about the method used for the initialization procedure on Line 2 of Algorithm 1. Unlike matrix completion, success in designing an efficient and accurate algorithm for the tensor completion problem is contingent to starting with a good initial estimates. In fact, the convergence rate of low-rank tensor algorithms is typically written as a function of the tensor components weights as well as the initialization error (Anandkumar et al. 2014; Jain and Oh 2014; Sun et al. 2017; Cai et al. 2021; Xia, Yuan, and Zhang 2021). It is therefore, imperative to design an initialization procedure efficient enough to help rule out local stationary points.

We use to our advantage, the fact that in our model, the tensor and matrix share at least one mode and use the singular value decomposition (SVD) (Stewart 1990; Ipsen 1998) of the observed matrix M to initialize the shared components of the tensor A along with the matrix weights \( \sigma_1, \ldots, \sigma_R \) and matrix component V, respectively. We then use the robust tensor power method (RTPM) from Anandkumar et al. (2014) to initialize the nonshared components B and C and the tensor weights. This is done by setting all missing entries in the tensor to be zero before running RTPM. In practice we show in our simulations in Section 5 that this is an adequate initialization procedure and produces much better initials compared to a random initialization scheme. In the more general case where all tensor modes have covariate matrices, the SVD on the covariate matrices can be used to initialize all the tensor components. In this case, the RTPM for nonshared components initialization would not be needed.

3.2.3. Rank and Cardinality Tuning

Our COSTCO method relies on two key parameters: the rank \( R \) and the sparsity parameters. It has been shown that exact tensor rank calculation is a NP-hard problem (Kolda and Bader 2009). In this section, following the tuning method in Allen (2012) and Sun et al. (2017), we provide a BIC-type criterion to tune these parameters. Given a prespecified set of rank values \( \mathcal{R} \) and a prespecified set of cardinality values \( S \), we choose the parameters which minimizes

\[
BIC = \log \left( \frac{\|P_{\Omega}(T - \sum_{r \in \mathcal{R}} \lambda_r a_r \otimes b_r \otimes c_r)\|_F^2}{n_1 n_2 n_3} \right) + \frac{\|M - \sum_{r \in \mathcal{R}} \sigma_r a_r \otimes v_r\|_F^2}{n_1 n_r} + \log \left( \frac{(n_1 n_2 n_3 + n_1 n_r) \sum_{r \in \mathcal{R}} (\|a_r\|_0 + \|b_r\|_0 + \|c_r\|_0 + \|v_r\|_0)}{n_1 n_2 n_3 + n_1 n_r} \right).
\]

To further speed up the computation, in practice, we tune these parameters sequentially. That is, we first fix \( s_1 = n_1 \) and tune the rank \( R \) via (8). Then given the tuned rank, we tune the sparsity parameters. This tuning procedure works very well through simulation studies in Section 5.

4. Theoretical Analysis

In this section, we derive the error bound of the recovered tensor components obtained from Algorithm 1. We only provide the results for tensor components as our ultimate goal is to recover the missing entries in the tensor. We present the recovery results for the estimated shared components \( a_r \) and nonshared tensor components \( b_r \) and \( c_r \) separately to highlight the sharp
improvement in recovery accuracy of the tensor resulting from incorporating the covariate information.

The theory is presented in two phases, first we focus on a simplified case in which the true tensor and matrix components \( a^*_r, b^*_r, c^*_r, \) and \( v^*_r \) are nonspare and both tensor and matrix weights are equal (i.e., \( \sigma_r = \lambda_r^*, \forall r \in [R] \)). Presenting this simplified case allows us to showcase clearly the interplay between the reveal probability, the tensor and matrix dimensions as well as how the noises in the tensor and matrix affect the statistical and computational errors of the algorithm. In the second case, we then present the results for the general scenario where the tensor and matrix weights are allowed to be unequal and the tensor and matrix components are assumed to be spare.

### 4.1. Case 1: Nonsparse Tensor and Matrix with Equal Weights

Before presenting the theorem for the simplified case, we introduce assumptions on the true tensor \( T^* \) and matrix \( M^* \) and then discuss their utility. Denote \( n := \max(n_1, n_2, n_3, n_r) \).

**Assumption 1 (Tensor and matrix structure).**

(i) Assume \( T^* \) and \( M^* \) are specified as in (5) with unique low-rank decomposition up to a permutation, and assume rank \( R = o(n^{1/2}) \) and \( \lambda^*_r = \sigma^*_r \) (equal weight), \( \forall r \in [R] \).

(ii) The entries of the decomposed components for both \( T^* \) and \( M^* \) satisfy the \( \mu \)-mass condition,

\[
\max_r \{ \|a^*_r\|_\infty, \|b^*_r\|_\infty, \|c^*_r\|_\infty, \|v^*_r\|_\infty \} \leq \frac{\mu}{\sqrt{n}},
\]

where \( \mu \) is a constant.

(iii) The components across ranks for both \( T^* \) and \( M^* \) meet the incoherence condition,

\[
\max_{i \neq j} \left\{ \|a^*_i - a^*_j\|, \|b^*_i - b^*_j\|, \|c^*_i - c^*_j\|, \|v^*_i - v^*_j\| \right\} \leq \frac{c_0}{\sqrt{n}},
\]

where \( c_0 \) is a constant.

**Assumption 1(i) is a common assumption in the tensor decomposition literature to ensure identifiability Kolda and Bader (2009), Anandkumar et al. (2014), Jain and Oh (2014), and Sun et al. (2017).** It imposes the condition that the tensor admits a low rank CP decomposition that is unique. This is the case of the undercomplete tensor decomposition, where the rank of the tensor is assumed to be lower than the dimension of the component. The condition \( \lambda^*_r = \sigma^*_r \) is a simplification of the problem that allows us to simplify the derivation and showcase clearly the interplay between important parameters. The same results (up to a constant) in **Theorem 1** would hold if \( \sigma^*_r \) is of the same order as \( \lambda^*_r \). The general weight case is described in **Section 4.2. Assumption 1(ii)** ensures that the mass of the tensor is not contained in only a few entries and is necessary if one hopes to recover any of the nonshare components of the tensor with acceptable accuracy. **Assumption 1(iii)** is related to the nonorthogonality of the tensor components and imposes a soft orthogonality condition on the tensor and matrix components. That is, the tensor components are allowed to be correlated only to a certain degree. Anandkumar, Ge, and Janzamin (2014) and Sun et al. (2017) show that such a condition is met when the tensor and matrix components are randomly generated from a Gaussian distribution. Both the \( \mu \)-mass condition and the incoherence conditions have been commonly assumed in low-rank tensor models (Anandkumar et al. 2014; Jain and Oh 2014; Sun et al. 2017; Xia and Yuan 2019; Cai, Poor, and Chen 2020; Cai et al. 2021).

**Assumption 2 (Reveal probability).** Denote \( \lambda^*_n := \min_{r \in [R]} \{\lambda^*_r\} \) and \( \lambda^*_\max := \max_{r \in [R]} \{\lambda^*_r\} \). We assume that each entry \((i, j, k)\) of the tensor \( T^* \) for all \( i \in [n_1], j \in [n_2] \), and \( k \in [n_3] \) is observed with equal probability \( p \) which satisfies,

\[
P \geq \frac{C R^2 \mu^3 \lambda^*_\max^2 \log^2 (n)}{(\lambda^*_n + \sigma^*_n)^2 n^{3/2}},
\]

where \( C \) is a constant.

**Assumption 2** guarantees that the tensor entries are revealed uniformly at random with probability \( p \). The lower bound on \( p \) is an increasing function of the tensor rank since recovering tensors with a larger rank is a harder problem which requires more observed entries. The bound on \( p \) is also an increasing function of the \( \mu \)-mass parameter since a larger \( \mu \)-mass parameter in **Assumption 1(ii)** indicates a smaller signal in each tensor entry and hence more reveal entries for accurate component recovery would be needed. Moreover, the bound on \( p \) is a decreasing function of the tensor component dimension \( n \) and relates as \( n^{-3/2} \) up to a logarithm term. This is the optimal dependence on the dimension in tensor completion literature (Jain and Oh 2014; Xia and Yuan 2019). Most importantly, the lower bound on \( p \) is relaxed when the minimal weight \( \lambda^*_n \) of the tensor or the minimal weight \( \sigma^*_n \) of the matrix increases. This reflects a critical difference when compared to the lower bound condition required in traditional tensor completion (Jain and Oh 2014; Xia and Yuan 2019) which corresponds to the case \( \sigma^*_n = 0 \). It shows the advantage of coupling the matrix of covariates for the tensor completion. This new lower bound on \( p \) translates to requiring less observed entries for the tensor recovery in the presence of covariates. Note that in the present simplified case \( \sigma^*_r = \lambda^*_r \), we still choose to write \( \sigma^*_n \), explicitly in the lower bound condition to showcase the effect of the covariate information. The improvement on \( p \) over existing literature will be clearer in **Assumption 6** for the general weight case.

**Assumption 3 (Initialization error).** Define the initialization errors for the tensor components as \( \epsilon_0_T := \max_{r \in [R]} \|a^0_r - a^*_r\|_2, \|b^0_r - b^*_r\|_2, \|c^0_r - c^*_r\|_2, \|v^0_r - v^*_r\|_2 \) and the initialization error for the matrix components as \( \epsilon_0_M := \max_{r \in [R]} \|v^0_r - v^*_r\|_2, \|a^0_r - a^*_r\|_2, \|b^0_r - b^*_r\|_2, \|c^0_r - c^*_r\|_2 \) and the initialization error for the matrix components as \( \epsilon_0_M := \max_{r \in [R]} \|v^0_r - v^*_r\|_2, \|a^0_r - a^*_r\|_2, \|b^0_r - b^*_r\|_2, \|c^0_r - c^*_r\|_2 \) and the initialization error for the matrix components as \( \epsilon_0_M := \max_{r \in [R]} \|v^0_r - v^*_r\|_2, \|a^0_r - a^*_r\|_2, \|b^0_r - b^*_r\|_2, \|c^0_r - c^*_r\|_2 \) and the initialization error for the matrix components as \( \epsilon_0_M := \max_{r \in [R]} \|v^0_r - v^*_r\|_2, \|a^0_r - a^*_r\|_2, \|b^0_r - b^*_r\|_2, \|c^0_r - c^*_r\|_2 \). Assume that

\[
\epsilon_0 := \max(\epsilon_0_T, \epsilon_0_M) \leq \frac{\lambda^*_n}{100 R \lambda^*_\max} - \frac{c_0}{3 \sqrt{n}}.
\]

Here the component \( c_0 / \sqrt{n} \) is due to the nonorthogonality of the tensor factors. When the components are orthogonal, we allow a larger initialization error. This observation aligns with the common knowledge in tensor recovery as the problem is known to be harder for nonorthogonal tensor factorization (Anandkumar, Ge, and Janzamin 2014). Similarly, a larger
rank $R$ of the tensor leads to a harder problem and a stronger condition on the initialization error. Under Assumption 1(i) $R = o(n^{1/2})$, when the condition number $\lambda_{\text{max}} / \lambda_{\text{min}} = O(1)$, this initial condition reduces to $\varepsilon_0 = O(1/R)$, as shown in Anandkumar, Ge, and Janzamin (2014); Jain and Oh (2014), the robust tensor power method initialization procedure used in our Algorithm satisfies $O(1/R)$ error bound.

**Assumption 4 (Signal-to-noise ratio condition).** Denote $\|\xi_T\|$, $\|\xi_M\|$ as the spectral norm of the error tensor and error matrix, respectively. We assume that

\[
\frac{\|\xi_T\|}{\sqrt{p} \lambda_{\text{min}}^*} = o(1) \quad \text{and} \quad \frac{\|\xi_M\|}{(p+1)\lambda_{\text{min}}^*} = o(1). \tag{10}
\]

Assumption 4 can be considered as the commonly used signal-to-noise ratio condition in noisy tensor decomposition (Sun et al. 2017; Sun and Li 2019; Cai et al. 2021; Xia, Yuan, and Zhang 2021). It ensures that the estimators for both shared and nonshared components contract in each iteration and the corresponding final statistical errors converge to zero. Note that when all mode of the tensors are coupled with covariate matrices, the condition on $\|\xi_T\|$ can be relaxed to $\sqrt{\frac{\|\xi_T\|}{p+1})\lambda_{\text{min}}^*} = o(1)$ due to the incorporation of covariate matrices for all shared components.

**Theorem 1 (Nonshared tensor and matrix components with equal weights).** Assuming Assumptions 1, 2, 3, and 4 are met. After running $\Omega\left(\log_2 \left(\frac{(p+1)\lambda_{\text{min}}^*}{\sqrt{p} \lambda_{\text{min}}^*} + \frac{\sqrt{p} \lambda_{\text{min}}^*}{\lambda_{\text{min}}^*} \right)\right)$ iterations of Algorithm 1 with $s_i = n_i$, for $i = 1, 2, 3, v$, we have

- **Shared Component $a_i$:**

  \[
  \max_{r \in [R]} \left(\|a_r^{s} - a_r^{*}\|_2\right) = O_p \left(\frac{\sqrt{p} \|\xi_T\| + \|\xi_M\|}{(p+1)\lambda_{\text{min}}^*}\right). \tag{11}
  \]

- **Nonshared Components $b_r$, $c_r$:**

  \[
  \max_{r \in [R]} \left(\|b_r^{c} - b_r^{*}\|_2, \|c_r^{c} - c_r^{*}\|_2, \|\lambda_r - \lambda_r^{*}\|_2\right) = O_p \left(\frac{\|\xi_T\|}{\sqrt{p} \lambda_{\text{min}}^*}\right). \tag{12}
  \]

Theorem 1 indicates that the shared component error is a weighted average of the spectral norm of the error tensor and error matrix. Whereas the nonshared component error is simply a function of the error tensor. In the extreme case in which the covariates matrix $M$ is noiseless, then the recovery error of the shared component becomes $\frac{\sqrt{p} \lambda_{\text{min}}^*}{(p+1)\lambda_{\text{min}}^*}$, which is much smaller than the recovery error of the nonshared component $\frac{\|\xi_M\|}{\sqrt{p} \lambda_{\text{min}}^*}$, especially when the observation probability $p$ is very small. Moreover even in the case in which the coupled covariates matrix is not noiseless, since $p \leq 1$ we notice an improvement in the statistical error of the recovered shared component compared to that of the nonshared components as long as the spectral norm of the error matrix is no larger than the spectral norm of the error tensor.

**Remark 1 (Sub-Gaussian noise).** In Theorem 1, we consider the noisy model with a general error tensor and error matrix. When the entries of the error tensor $\xi_T$ and the error matrix $\xi_M$ are iid sub-Gaussian with mean zero and variance proxy $\sigma^2$, we can further simplify the statistical error. For simplicity, consider $E_T \in \mathbb{R}^{n \times n \times n}$ and $E_M \in \mathbb{R}^{n \times n}$. According to Tomioka and Suzuki (2014) and Vershynin (2018), $\|\xi_T\| = O_p(\sigma \sqrt{n \log(n)})$ and $\|\xi_M\| = O_p(\sigma \sqrt{n \log(n)})$. Therefore, the errors of the shared component in (11) and that of the nonshared component in (12) can be simplified as

\[
(11) = O_p \left(\frac{\sigma}{\lambda_{\text{min}}^*} \left(\sqrt{p} + 1\right)\frac{\sqrt{n \log(n)}}{(p+1)}\right);
\]

\[
(12) = O_p \left(\frac{\sigma}{\lambda_{\text{min}}^*} \frac{\sqrt{n \log(n)}}{p}\right). \tag{12}
\]

The estimation error for the nonshared component matches with that in the standalone tensor completion (Cai et al. 2021), while the estimation error for the shared component largely improves due to the incorporation of the covariate matrix. The improvement is more significant especially when the observation probability $p$ is small as $(\sqrt{p} + 1)/(p+1) \approx 1/\sqrt{p}$.

### 4.2. Case 2: Sparse Tensor and Matrix with General Weights

We now present the result for the general case with sparse tensor and matrix $T^*$ and $M^*$ and the weights of the tensor and matrix are allowed to be unequal. The theoretical analysis for the general case is much more challenging than that covered in Theorem 1. For example, unlike the setting in Case 1, we are no longer able to derive the closed form solution to the optimization problem in (7) for the shared tensor component. Instead, we construct an intermediate estimate in the analysis of the shared component recovery. Fortunately, this general result allows us to explicitly quantify the improvement due to the covariates on the missing percentage requirement and the final error bound.

The following conditions are needed for the general scenario. Recall that $d = \max\{d_1, d_2, d_3, d_v\}$ is the maximal true sparsity parameter defined in (6) and define $s := \max\{s_1, s_2, s_3, s_v\}$.

**Assumption 5 (sparse tensor and matrix structure).**

(i) Assume $T^*$ and $M^*$ have the sparse structure in (5) and (6) with unique low-rank decomposition up to a permutation, and assume rank $R = o(d^{1/2})$.

(ii) The entries of the decomposed components for $T^*$ satisfy the following $\mu$-mass condition

\[
\max_{r} \|a_r^{s}\|_{\infty}, \|b_r^{s}\|_{\infty}, \|c_r^{s}\|_{\infty}, \|v_r^{s}\|_{\infty} \leq \frac{\mu}{\sqrt{d}}.
\]

(iii) The components across ranks for both $T^*$ and $M^*$ meet the incoherence condition,

\[
\max_{i \neq j} \left(\|a_r^{s}a_r^{s}\|, \|b_r^{s}b_r^{s}\|, \|c_r^{s}c_r^{s}\|, \|v_r^{s}v_r^{s}\|\right) \leq \frac{\varepsilon_0}{\sqrt{d}}.
\]

Notice that since the components of tensor and matrix are assumed to be sparse, the $\mu$-mass and incoherence condition are functions of the maximum number of nonzero elements $d$ in the tensor and matrix components rather than the dimension $n$. In the case in which $d \ll n$, this constitutes a milder assumption compared to Assumptions 1(ii) and 1(iii).
Assumption 6 (Reveal probability). We assume that each tensor entry \((i, j, k)\) for all \(i \in [r_1], j \in [r_2] \text{ and } k \in [r_3]\) is observed with equal probability \(p\) which satisfies,

\[
p \geq CR^2\mu_1^2\lambda_{\min}^6 \log^3(d)/(\lambda_{\min}^2 + \sigma_{\min}^2 d^{3/2}).
\]  

(13)

Similar to the equal-weight case, the required lower bound on the reveal probability in (13) improves the established lower bound for the tensor completion with no covariates matrix. Specifically, Jain and Oh (2014), Montanari and Sun (2018), and Xia and Yuan (2019) show that the lower bound for nonspars tensor completion is of the order \(\lambda_{\min}^2 \log^2(n)/(\lambda_{\min}^2 + \sigma_{\min}^2 n^{1/2})\), while our lower bound is of the order \(\lambda_{\min}^2 \log^2(n)/(\lambda_{\min}^2 + \sigma_{\min}^2 n^{1/2})\) when the components are not sparse \((d = n)\). This highlights the fact that a weaker assumption on the reveal probability is required in the presence of covariates matrix than in the case with no covariates. An interesting phenomenon is that when the minimal weight of the matrix \(\lambda_{\min}\) is very large, we could allow the reveal probability to be even close to zero. For example, in the nonspars case, when \(\lambda_{\max} = O(\lambda_{\min})\) and \(\sigma_{\min}/\lambda_{\max} = \sqrt{n}\), our lower bound on \(p\) is relaxed to \(O(n^{-1/2})\) up to a logarithmic order. In fact, as long as \(\lambda_{\max} = o(\sigma_{\min})\) and \(\lambda_{\min} = O(\sigma_{\min})\), the lower bound would be smaller than \(O(n^{-3/2})\). This is a major advantage of our method and this property does not exist in existing standalone tensor completion which requires \(n^{-3/2}\) lower bound on \(p\). As demonstrated in our simulations, our COSTCO is still satisfactory even when 98% of the tensor entries are missing, while the traditional tensor completion method start to fail when there are more than 90% missing entries. Moreover, in the sparse case, the lower bound is a decreasing function of the sparsity parameter \(d\). This is intuitive as when \(d\) decreases, the nonzero tensor components will concentrate on fewer dimensions which makes the tensor recovery problem harder.

Assumption 7 (Initialization error). Assume that

\[
\epsilon_0 := \max{\epsilon_{0r}, \epsilon_{0s}} \leq 95/96\lambda_{\min}^2 + \sigma_{\min}^2/144R(\lambda_{\max}^2 + \sigma_{\max}^2) - c_0/3\sqrt{d},
\]  

(14)

with \(\epsilon_{0r}\) and \(\epsilon_{0s}\) as defined in Assumption 3.

Compared to that in Assumption 3, the initialization condition for Case 2 is slightly stronger. This is reflected on two parts. First, the term \(c_0/\sqrt{d}\) is due to the nonorthogonality of sparse tensor components and is larger in the sparse case. This requires a stronger condition on the rank \(R\) as shown in Assumption 1(i) in order to ensure the positivity of the right-hand side of (14). Second, the ratio \((95/96\lambda_{\min}^2 + \sigma_{\min}^2)/(144(\lambda_{\max}^2 + \sigma_{\max}^2))\) is smaller than \(\lambda_{\min}/(100\lambda_{\max})\) in Assumption 3. Even when \(\lambda_{\max} = \sigma_{\min}\) and \(d = n\), this condition is still slightly stronger than Assumption 3 since \(\lambda_{\min}^2/\lambda_{\max}^2 < \lambda_{\min}/\lambda_{\max}\). This additional term is due to handling the nonequal weights. Fortunately, when condition numbers \(\lambda_{\max}/\lambda_{\min} = O(1)\) and \(\sigma_{\max}/\sigma_{\min} = O(1)\), we have \(\epsilon_0 = O(1/R)\), which is again satisfied by the initialization procedure in our algorithm.

Assumption 8 (Signal-to-noise ratio condition). Denote \(\|\mathcal{E}_T\|_{<s>}\), \(\|\mathcal{E}_M\|_{<s>}\) as the sparse spectral norm of the error tensor and error matrix defined in Section 2. We assume that

\[
\frac{\|\mathcal{E}_T\|_{<s>}}{\sqrt{p}\lambda_{\min}^*} = o(1) \quad \text{and} \quad \frac{\sigma_{\max}^*\|\mathcal{E}_M\|_{<s>}}{p\lambda_{\min}^* + \sigma_{\min}^2} = o(1).
\]  

(15)

Assumption 8 extends the signal-to-noise ratio condition in Assumption 4 to the sparse and general nonequal weight case.

Theorem 2 (Sparse tensor and matrix components with general weights). Assuming Assumptions 5, 6, 7, and 8 are met. After running \(\Omega\left(\frac{\log^2}{\sqrt{p}\lambda_{\min}^* + \sigma_{\min}^2}\right)\) iterations of Algorithm 1 with \(s_{i} \geq d_{i}\) for \(i = 1, 2, 3, \nu\), we have

- Shared Component \(a_{r}\):

\[
\max_{r \in [k]} (\|a_{r} - a_{r}^*\|_2) = \mathcal{O}_p\left(\frac{\sqrt{p}\lambda_{\max}^*\|\mathcal{E}_T\|_{<s>} + \sigma_{\max}^*\|\mathcal{E}_M\|_{<s>}}{p\lambda_{\min}^* + \sigma_{\min}^2}\right).
\]  

(16)

- Nonshared Components \(b_{r}, c_{r}\):

\[
\max_{r \in [k]} (\|b_{r} - b_{r}^*\|_2, \|c_{r} - c_{r}^*\|_2, \frac{|r - \lambda_{r}^*|}{\lambda_{r}^*}) = \mathcal{O}_p\left(\frac{\|\mathcal{E}_T\|_{<s>}^\prime}{\sqrt{p}\lambda_{\min}^*}\right).
\]  

(17)

Similar to that in Theorem 1, the statistical error for the shared tensor component in Theorem 2 is a weighed age of the sparse spectral norm of the error tensor \(\mathcal{E}_T\) and error matrix \(\mathcal{E}_M\). The key difference is that the weight is now related to \(\lambda_{\max}^*\) and \(\sigma_{\max}^*\) and the spectral norm is now much smaller than the nonshared counterparts in Theorem 1 since typically \(s < n\) and hence \(\|\mathcal{E}_T\|_{<s>} < \|\mathcal{E}_T\|\) and \(\|\mathcal{E}_M\|_{<s>} < \|\mathcal{E}_M\|\). Similarly, the recovery error for the nonshared tensor component in the general case is also smaller than that in (12) due to a smaller spectral norm. This observation highlights the advantage of considering sparse tensor components. In addition, we highlight a few important scenarios in Table 1 where the error of shared tensor component is smaller than that of the nonshared component. Such scenario indicates when the additional covariate information is useful to reduce the estimation error of the tensor components. In summary, such improvement is observed when the sparse spectral norm of the error matrix is smaller than or comparable to that of the error tensor.

Remark 2 (Sub-Gaussian noise). Similar to Remark 1, when the entries of the error tensor \(\mathcal{E}_T\) and the error matrix \(\mathcal{E}_M\) are iid sub-Gaussian with mean zero and variance proxy \(\sigma^2\), we can further simply the statistical error in Theorem 2. Using a similar covering number argument in Tomioka and Suzuki (2014) and Zhou et al. (2021) show that the sparse spectral norm of \(\mathcal{E}_T\) and \(\mathcal{E}_M\) satisfies \(\|\mathcal{E}_T\|_{<s>} = \mathcal{O}_p(\sigma\sqrt{s\log(n)})\) and \(\|\mathcal{E}_M\|_{<s>} = \mathcal{O}_p(\sigma\sqrt{s\log(n)})\). Therefore, the errors of the shared component in (16) and that of the nonshared component in (17) can be simplified as

\[
(16) = \mathcal{O}_p\left(\frac{(\sqrt{p}\lambda_{\max}^* + \sigma_{\max}^*)\sqrt{s\log(n)}}{p\lambda_{\min}^* + \sigma_{\min}^2}\right);
\]  

(17) = \mathcal{O}_p\left(\frac{\sigma_{\max}^*}{\sqrt{p}}\sqrt{s\log(n)}\right).
The estimation error for the nonshared component matches with the rate in the sparse tensor model (Zhou et al. 2021), while the estimation error for the shared component again largely improves due to the incorporation of the covariate matrix.

5. Simulations

In this section we evaluate the performance of our COSTCO algorithm via a series of simulations. We compare it with two competing state-of-the-art methods tenALSsparse by Jain and Oh (2014) and OPT by Acar, Kolda, and Dunlavy (2011). tenALSsparse is an alternating minimization based method for tensor completion which incorporates a refinement step in the standard ALS method. In contrast to our method, tenALSsparse does not incorporate side covariate information in tensor completion. Comparing our algorithm to tenALSsparse helps to highlight the impact of incorporating additional information through coupling with a covariate matrix. It is worth noting that the original algorithm from Jain and Oh (2014) was built for the recovery of nonsparse tensors. In order to allow a fair comparison between our algorithm and theirs, we modify their original algorithm by introducing the same truncation scheme presented in Algorithm 1 to generate the sparse version of their algorithm. The second comparison method is the OPT algorithm by Acar, Kolda, and Dunlavy (2011), which approaches the coupled matrix and tensor component recovery by solving for all components simultaneously using a gradient-based optimization approach. The all-at-once optimization method is known to be robust to rank misspecification (Song et al. 2019), however, it is computationally less efficient then ALS based methods specially when the tensor is highly missing (Tomasi and Bro 2006).

In the aforementioned sections, we discuss our models and theories via a third-order tensor to simply the presentation. Note that our COSTCO is applicable to the tensor with more than three modes. In the simulation, we generate a fourth-order tensor $T^*$ $\in \mathbb{R}^{d_1 \times 30 \times 30 \times 30}$ and a matrix $M^*$ $\in \mathbb{R}^{d_1 \times 30}$. We assume that the matrix and the tensor share components across the first mode just as is the case in the aforementioned sections. In order to form the tensor $T^*$ and the matrix $M^*$, we draw each entry of $A^*$ $\in \mathbb{R}^{d_1 \times 30}$, $B^*$ $\in \mathbb{R}^{30 \times 30}$, $C^*$ $\in \mathbb{R}^{30 \times 30}$, $D^*$ $\in \mathbb{R}^{30 \times 30}$ and $V^*$ $\in \mathbb{R}^{30 \times 30}$, from the iid standard normal distribution. We enforce sparsity to the tensor components by keeping only the top 40% of the entries in each column in $B^*$, $C^*$, and $D^*$ and set the rest of the entries to zero. In all of our simulations we consider the coupled modes $A^*$ to be dense to mimic the real data scenario in Section 6 where the coupled matrix is dense. We define $\lambda_1^*, ..., \lambda_R^*$ and $\sigma_1^*, ..., \sigma_R^*$ as the product of the nonnormalized component norms in each mode, that is, $\lambda_i^* = \|a_i^*\|_2 \times \|b_i^*\|_2 \times \|c_i^*\|_2 \times \|d_i^*\|_2$ and $\sigma_i^* = \|a_i^*\|_2 \times \|v_i^*\|_2$. We then normalize each of the columns of $A^*$, $B^*$, $C^*$, $D^*$, $V^*$ to unit norm. To illustrate, the first mode component matrix $A^*$ becomes $A^* = [a_1^* / \|a_1^*\|_2, ..., a_R^* / \|a_R^*\|_2]$. The sparse tensor $T^*$ and matrix $M^*$ are then formed as $T^* = \sum_{r \in [R]} \lambda_r^* a_r^* \otimes b_r^* \otimes c_r^* \otimes d_r^*$ and $M^* = \sum_{r \in [R]} \sigma_r^* a_r^* \otimes v_r^*$. We then add noise to the tensor and matrix using the following setup $T = T^* + \eta_T N_T / \|N_T\|_F$ and $M = M^* + \eta_M N_M / \|N_M\|_F$, where $N_T$ and $N_M$ are a tensor and a matrix of the same size as $T^*$ and $M^*$ respectively, whose entries are generated from the standard normal distribution. A similar noise generation procedure has been considered in Acar, Kolda, and Dunlavy (2011). We simulate the uniformly missing at random pattern in the tensor data by generating entries of the reveal tensor $\Omega \in \mathbb{R}^{d_1 \times 30 \times 30 \times 30}$ from the binomial distribution with reveal probability $p$. The sparse and noisy tensor $P_\Omega(T)$ with missing data is finally obtained as $P_\Omega(T) = T * \Omega$, where $*$ is the element-wise multiplication.

To assess the goodness of fit for the tensor and tensor component recovery, we use the normalized Frobenius norm of the difference between the recovered component and the true component. We compute the tensor estimation error, the tensor component error and tensor weights error as

$$
tensor error := \|T^* - T\|_F / \|T^*\|_F;
$$

$$
component error := \|U^* - U\|_F / \|U^*\|_F;
$$

$$
weight error := \|\lambda^* - \lambda\|_2 / \|\lambda^*\|_2, \tag{18}
$$

where $T$, $U$, are the estimated tensor and tensor components with $U \in \{A, B, C, D\}$, and $\lambda := (\lambda_1, ..., \lambda_R)^T$ is the vector of estimated tensor weights returned by Algorithm 1. In all simulations we return the mean error of 30 replicates of each experiment. Throughout all the experiments, we set the maximum number of iterations $\tau$ to be 200, the tolerance $tol$ in Algorithm 1 is set to be $1e^{-7}$. To avoid bad local solutions, we conduct 10 initializations for each replicate in all methods. We set the tuning range for the rank $R$ to be $\{1, 2, 3, 4, 5\}$. The tuning range for the sparsity is set to be $\{20\%$, $40\%$, $60\%$, $80\%$, $90\%$, $100\%\}$, each value representing the percentage of nonzero entries in the latent components as performed on Lines 9 and 11 of Algorithm 1. Note that in addition to a series of simulations considered here, in Section S.5 of the supplementary material, we provide two additional simulations to investigate the practical
The estimation errors with varying missing percentages are shown in Table 2. The table indicates that our algorithm COSTCO outperforms both tensor completion methods, tenALSsparse and OPT, in terms of estimation error, especially when the missing percentage is higher. This is evident in the table, where COSTCO consistently has lower estimation errors compared to the other methods.

5.1. Missing Percentage

In this first simulation, we consider the case with varying levels of missing percentages. We set the dimension of the shared component to be \( d_1 = 30 \) and generate \( P_2(T) \in \mathbb{R}^{30 \times 30 \times 30} \). We set the rank to be \( R = 2 \) and the noise level \( \eta_T, \eta_M \) to be both 0.001. We measure the recovery error under four different settings of the reveal probability parameter \( \eta \in \{0.2, 0.1, 0.05, 0.01\} \). In other words, 80%, 90%, 95%, and 99% of the tensor entries are missing in each setting.

5.2. Noise Level

In the next set of experiments, we vary the noise level parameter for the tensor \( \eta_T \) and noise level for the matrix \( \eta_M \) to test algorithms’ robustness to noise. These two parameters control the signal-to-noise ratio in the model. The missing probability for these experiments is set to 90% and tensor rank and sparsity of the true tensor are set to \( R = 2 \) and 60%, respectively.

As can be seen in Table 3, when the tensor noise \( \eta_T \) is greater than that of the matrix noise \( \eta_M \), our algorithm outperforms the two competing methods with a large gap in recovery error. Even when the matrix has a slightly larger noise level than the tensor \( (\eta_M = 0.01, \eta_T = 0.001) \), COSTCO still outperforms the other two algorithms. It shows that in high missing data regime coupling a matrix that has a slightly larger noise than the tensor still provides enough information to improve the tensor recovery rate. On the other hand, when the matrix noise level is much higher than that of the tensor \( (\eta_M = 0.1, \eta_T = 0.001) \) in Table 3, we observe that our algorithm COSTCO and the other
coupled algorithm OPT are inferior compared to tenALSsparse. In this case, the recovery of the shared component A suffers the most in COSTCO and OPT and is responsible for the inferior tensor recovery error compared to tenALSsparse which does not use the coupled matrix. This is expected as a matrix with much larger noise than that of a tensor no longer brings in enough signals in the coupling and therefore makes the tensor completion problem harder than when the matrix is completed omitted from the model. Finally, an interesting phenomenon is that the noise level of the error matrix \( \eta_M \) only affects the estimation error of the shared component but not those of the nonshared components. To see it, in the last two settings in Table 3, when \( \eta_T \) is fixed and \( \eta_M \) increases, only the recovery accuracy of the shared component A significantly drops, but those of the nonshared components have no significant changes. However, in the first two settings in Table 3, when \( \eta_M \) is fixed and \( \eta_T \) increases, the recovery accuracy of both shared and nonshared components significantly drops. These findings agree well with our theoretical results in Theorem 2.

### 6. Real Data Analysis

We apply our COSTCO method to an advertisement (ad) data to showcase its practical advantages. COSTCO makes use of multiple sources of ad data to extract the ad latent component which is a comprehensive representation of ads. We demonstrate that the obtained ad latent components are able to deliver interesting ad clustering results that are not achievable by a stand-alone method.

Online advertising is a type of marketing strategy which uses internet to promote a given product to potential customers. Extracting patterns in data gathered from online advertisement allows ad platforms and companies to churn data into knowledge which is then used to improve customer satisfaction. Clustering algorithms have been applied to the ad data to discover ad or user clusters for better ad targeting. After computing the similarity between the new ad and each ad cluster, the ad agency can determine whether a new ad should be assigned to a specific user group. Most ad-user clustering research focuses on a single correlation data. What makes our method different is that we not only have a third-order user-by-ad-by-device click tensor data but we also possess additional information which describe specific features of ads. Our COSTCO algorithm uses both click tensor data and ad matrix data to extract the ad latent component for better ad clustering.

The data we analyze in this section is advertising data collected from a major internet company for four weeks in May–June 2016. A user preference tensor was obtained by tracking the behavior of 1000 users on 140 ads accessed through three different devices. The 1000 \( \times \) 140 \( \times \) 3 tensor is formed by computing the click-through-rate (CTR) of each (user, ad, device) triplet over the four weeks period; which is the number of times a user has clicked an ad from a certain device divided by the number of times the user has seen that ad from the specific device. Each CTR tensor entry was aggregated over the four weeks period; which is then the number of times a user has clicked an ad from a certain device.

| Noise level | Component | COSTCO | tenALSsparse | OPT |
|-------------|-----------|--------|--------------|-----|
| \( \eta_M = 0.001 \) | \( w \) | 2.74e-04 (7.31e-10) | 5.37e-04 (1.00e-09) | 4.74e-04 (7.31e-10) |
| \( \eta_T = 0.01 \) | | 1.05e-04 (2.24e-10) | 3.17e-04 (1.13e-09) | 1.05e-04 (2.24e-10) |
| \( \eta_M = 0.01 \) | Comp A | 2.13e-04 (8.03e-10) | 3.10e-04 (4.72e-10) | 3.13e-04 (8.03e-10) |
| \( \eta_T = 0.1 \) | Comp B | 2.15e-04 (1.33e-09) | 3.14e-04 (1.35e-09) | 3.15e-04 (1.33e-09) |
| \( \eta_M = 0.01 \) | Comp C | 2.21e-04 (1.43e-09) | 3.22e-04 (1.69e-09) | 3.21e-04 (1.43e-09) |
| \( \eta_T = 0.001 \) | Comp D | 1.41e-05 (6.77e-11) | 1.48e-05 (7.44e-11) | 1.41e-05 (6.77e-11) |

NOTE: Reported values are the average and standard error (in parentheses) of estimation errors. COSTCO: the proposed method; tenALSsparse: sparse version of the tensor completion method by Jain and Oh (2014); OPT: the gradient based all at once optimization method of Acar, Kolda, and Dunlavy (2011). The bold values mean the smallest error among different algorithms.
Beside the ad CTR tensor, we also have access to the ad text raw data that store the content of all ads. We use Latent Dirichlet Allocation (LDA) (Blei, Ng, and Jordan 2003) to process the ad text data. LDA is an unsupervised topic modeling algorithm that attempts to describe a set of text observations as a mixture of different topics. We first follow Blei, Ng, and Jordan (2003) to tune the parameters of LDA such as the number of topics and the Dirichlet distribution parameter that give the best tradeoff between low perplexity value and efficient computing time. The best perplexity is obtained for 20 topics. This means that all the 140 advertisement data can be considered as a combination of 20 topics. Due to space constraints, we illustrate an example of advertisement data can be considered as a combination of 20 topics. Due to space constraints, we illustrate an example of

We then compare the ad latent components returned from COSTCO and tenALSsparse in Figure 4. As a comparison, we also include the result of SVD which directly decomposes the ad covariate matrix data. The ad clusters shown in Figure 4 are obtained by applying the K-means clustering algorithm to the five columns of the latent components returned from our COSTCO show a clear clustering structure with five clusters. On the other hand, the ad components extracted from tenALSsparse are all clustered around zeros. This is because the ad CTR tensor is highly sparse and the latent components based on decomposing the tensor itself contain many small values. Therefore, ad clusters generated using tenALSsparse tend to have very large and very small clusters.

Finally, after obtaining the ad clusters, we visualize the ad topics from the each cluster in Figure 5. Specifically, for all ads assigned in each cluster, we apply the topic modeling method LDA to these ad texts to obtain their topics. For example, the ad cluster 1 from our COSTCO algorithm consists of four interesting topics, represented as four boxes in the first row of Figure 5. Within each topic, the top five words are highlighted in green in our COSTCO method.

Figure 5 demonstrates some interesting ad clustering results obtained from our COSTCO algorithm which links different ad industries into the same cluster. For example based on cluster 1 from COSTCO, ads about male and female online dating are clustered together with ads about women retail stores and man clothing accessories. In cluster 2 from COSTCO, ads about weight lost and weight loss surgery are clustered together with

### Table 4. Top 10 words for seven chosen topics.

| Topics | Ride | Gaming | Security | Mortgage | Insurance | Online dating | Fashion retail |
|--------|------|--------|----------|----------|-----------|---------------|----------------|
| Top words | uber pay | game controller | vivid mortgage | get single | buy car | experience front payment | less mean gym | click sale | tradeoff between low perplexity value and efficient computing time |
| | | | | | | | | | | |
| NOTE: Top words were obtained through LDA.
Figure 4. Scatterplot of the ad latent components obtained from three methods. Different clusters are represented via different colors.

Figure 5. Result of ad clusters obtained using different methods.

ads about gourmet cuisine and restaurant which indicates that users who interact with weight loss ads are also interested in nutrition related ads. Cluster 3 of COSTCO contains ads about house mortgage, home security devices, auto, home and auto insurance, house weather control devices which indicates that users that are homeowners tend to be interested in home and auto related things. These interesting clusters are not obtained in the SVD method nor the tenALSsparse method. The clusters from SVD are solely related to the topic of each ad as shown in Figure 5 and the clusters from tenALSsparse are highly unbalanced and do not contain any understandable relationship between ads. These clustering results illustrate the practical value of our COSTCO method. By incorporating ad covariate matrix into the completion of the ad CTR tensor, we are able to obtain a more synthetic description of ads and find interesting links between different advertising industries, which directly helps the marketing team to strategize the ad planning procedure accordingly for better ad targeting.

Supplementary Materials
The supplementary material contains interesting extensions to our current framework, proof of main theorems, additional simulation results, and the implementation details of a competitive covariate-assisted neural tensor factorization compared in the real data analysis.

Acknowledgments
The authors thank the editor Professor Ian McKeague, the associate editor and two anonymous reviewers for their valuable comments and suggestions which led to a much improved article.

Funding
Will Wei Sun's research was partially supported by ONR grant N00014-18-1-2759. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Office of Naval Research.
