Learning Preferences with Side Information

Vivek F. Farias
Sloan School of Management
Massachusetts Institute of Technology
email: vivekf@mit.edu

Andrew A. Li
Operations Research Center
Massachusetts Institute of Technology
email: aali@mit.edu

July 18, 2017

Abstract

Product and content personalization is now ubiquitous in e-commerce. There is typically too little available transactional data for this task. As such, companies today seek to use a variety of information on the interactions between a product and a customer to drive personalization decisions. We formalize this problem as one of recovering a large-scale matrix, with side information in the form of additional matrices of conforming dimension. Viewing the matrix we seek to recover and the side information we have as slices of a tensor, we consider the problem of *Slice Recovery*, which is to recover specific slices of 'simple' tensors from noisy observations of the entire tensor. We propose a definition of simplicity that on the one hand elegantly generalizes a standard generative model for our motivating problem, and on the other subsumes low-rank tensors for a variety of existing definitions of tensor rank. We provide an efficient algorithm for slice recovery that is practical for massive datasets and provides a significant performance improvement over state of the art incumbent approaches to tensor recovery. Further, we establish near-optimal recovery guarantees that in an important regime represent an order improvement over the best available results for this problem. Experiments on data from a music streaming service demonstrate the performance and scalability of our algorithm.

Keywords: personalization; e-commerce; online retail; recommender systems; collaborative filtering; matrix recovery; tensor recovery; side information; multi-interaction data

1. Introduction

Consider the problem of learning the propensity of individual customers for different products. This is an important challenge in e-commerce as customer preferences are a core input into a number of operational and marketing activities. For example, recommender systems are pervasive throughout e-commerce where they serve as a tool to drive sales and consumption by decreasing search costs for users. These systems rely on estimates of customer-product propensities to automatically suggest products to users that they are likely to enjoy. User-item propensities are also used to offer personalized experiences in more general service settings. Search results and loyalty programs may all be personalized to a user. One-to-one marketing, including banner advertisements and targeted sales promotions all benefit from accurate estimates of user-item propensities. The list goes on.

As an example of the task above, consider the problem faced by a retailer estimating the likelihood a specific customer might purchase a given product using historical transaction data available across all customers and products in the retailer's offering. This task is already quite challenging due to its massive scale: Amazon.com has upwards of 10^8 active users and products (Export-x (2015), Statista (2016)). In addition to scale, however, this task represents a *statistical* challenge: the probability of a given customer purchasing a given product is small, so that the observed matrix of transactions has a very small number of non-zero entires. For instance, in the case of Amazon, the average active user makes *less than a single* transaction in a month (FastCompany (2015)). We will see that this is, in fact, the key problem: meaningful recovery of the underlying probabilities from their corresponding realizations is hard when these probabilities are small.

Ultimately, the only remedy available in this situation is acquiring more data, perhaps beyond historical transactions. In the retail setting, this is indeed feasible: e-commerce businesses are able to capture data from a variety of distinct *types* of observations at the customer-product granularity. For example, besides sales transactions, online retailers record users' browse and search histories (capturing respectively 'browse' and 'search' interactions between user-product pairs); clickstream data (capturing a 'click' interaction); and responses to advertisements and promotions (capturing a 'promotion' interaction), just to name a few. The ultimate task, of course, remains predicting the likelihood a customer will purchase a given product. While these 'slices' of data encode interactions between a customer and a product that are distinct from a transaction, they may well inform the likelihood of a transaction and ultimately help resolve the challenge of limited data.

We refer to this as the problem of learning preferences (e.g. the likelihood of a given customer purchasing a given product in the retail example) with side information (e.g. data beyond just historical transactions, as alluded to above). Our first objective will be to formalize this problem; there are many paths we can take here and we will ultimately propose viewing the problem at hand as one of recovering a three-dimensional *tensor* from its noisy observations. We will then seek to solve this problem and will ultimately present an efficient, near-optimal algorithm for the same that yields a dramatic improvement over existing approaches to tensor recovery.

1.1. Representing Data as a Tensor

Taking a step back, the problem of estimating customer-product purchase probabilities from a *single* type of interaction data can be formulated as a *noisy matrix recovery problem* – the goal is to estimate the matrix whose rows correspond to customers, columns correspond to products, and entries contain the purchase probabilities. If our data consisted purely of transactions, these transactions could be viewed as a 'noisy realization' of the underlying matrix which we seek to recover. As we discuss in a subsequent section, this formalization can be viewed as equivalent to assigning latent feature vectors u_i, v_j to each customer i and each product j respectively, and

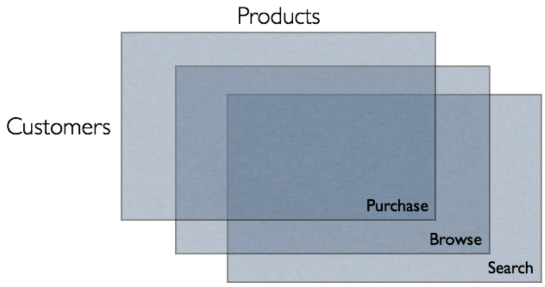


Figure 1: Customer-product interaction matrices represented as slices of a tensor.

assuming the probability of a specific customer i purchasing product j is a bilinear form in these latent vectors, $f(u_i, v_j)$. As one allows the dimension of the latent space to grow, the expressive power of such a model grows as well, ultimately becoming fully general. The problem of matrix recovery can be viewed as one of learning the latent feature vectors and the bilinear form $f(\cdot, \cdot)$ from observed data. Indeed this formalization of the problem has proved to be incredibly productive and central to the current state-of-the-art in the design of personalization algorithms, as we discuss in the literature review.

Our goal here is to consider multiple types of interactions (for example, sales transactions *along* with other customer-product interactions, such as browse, search, etc.). In analogy to matrix recovery, we may represent this data as a set of matrices, with each matrix corresponding to a single type of interaction. We model the various 'slices' of data that we have as the slices of a three dimensional tensor (see Figure 1). Tensors, which are the higher-dimensional analogues of matrices, have been used in a broad array of applications to represent high-dimensional data. The underlying generative model (that we will describe in greater detail in the next section) will then, in analogy to the matrix setting, continue to correspond to assigning latent feature vectors u_i, v_j to each customer i and each product j respectively. In addition, however, the likelihood of each interaction type k between a specific customer i and product j will be described by a distinct bilinear form in these latent vectors, $f_k(u_i, v_j)$. The problem of 'recovering' this tensor from its noisy observations is then equivalent to learning these latent feature vectors along with each of the bilinear forms associated with a specific type of interaction.

We may thus formalize the task we have laid out as the problem of recovering a three dimensional tensor from its 'noisy observations'. In fact, we will typically be interested in recovering a single *slice* of the tensor (for instance, the slice corresponding to the likelihood of a transaction) using data available across *all* slices. To clarify the observation model at hand, the observed data, as a special case, can be seen as a single realization of an underlying tensor of probabilities; i.e. a given entry of the observed tensor is a Bernoulli random variable with mean equal to the corresponding entry in the ground truth tensor. When the underlying ground truth probabilities are small, the observed tensor will be sparse (in the usual mathematical sense of the term). This observation model is

particularly relevant to the retail example discussed above where the occurrence (or nonoccurrence) of a transaction or other interactions may be thought of as Bernoulli realizations with specific, unknown probabilities. A similar generative model is also relevant to our experiments with music streaming data. This setting, which we will continue to refer to as the problem of *tensor recovery from noisy observations*, or *tensor recovery* for short, is the primary focus of our algorithm and recovery guarantees.

On the other hand, a distinct but closely related observation model is the setting where some small subset of entries of the tensor is observed exactly, and nothing is observed outside of this subset of entries. One application of this setting is the oft-cited Netflix problem: using a dataset of user-movie ratings, the task was to predict how a user might have rated a movie they had not watched. The data here is naturally represented as a matrix, with some entries observed in the form of ratings, and the other entries unobserved. Of course, we can again incorporate side information in this setting, in the form of additional partially observed matrices – this is typically referred to as the *tensor completion* problem. In our retail application this observation model would correspond to knowing the exact probability of purchase on some subset of customers and nothing outside that subset. Evidently, the observation model is less relevant to the retail application: we do not observe probabilities; we observe transactions. Moreover, we observe whether or not a given customer has purchased a given product for all customers and products; transactions, albeit sparse, are not hidden from the retailer. Nonetheless, we will describe how our algorithm can be extended to this observation model using a device originally proposed by Achlioptas and McSherry (2007).

Before summarizing our approach and contributions, we point out that there is by now, a fairly robust literature on the problem of tensor recovery that one might hope to fall back on at this stage. The mainstay of this literature is a convex optimization approach that seeks to find a tensor that is simultaneously 'close' to the observed data and 'simple' in the sense that a convex surrogate of the 'tensor rank' is small. We will formalize the notion of tensor rank in the next section, but ultimately this quantity can be thought of as restricting the dimension of the latent space of customer and product feature vectors, as well as the family of bilinear functions $f_k(\cdot, \cdot)$. Unfortunately, we will see that the convex approach falls short of expectations here. This happens for two key reasons that we will formalize in a subsequent section but explain in brief here. First and foremost, in the tensor setting, these convex optimization approaches are *difficult to scale* to massive amounts of data and will typically call for dense matrix operations at scales that are untenable. This is unlike the matrix setting where the convex approach can be shown equivalent to a simple specialized algorithm (singular value decomposition with soft-thresholding) ideally suited for massive datasets. More importantly though, the *statistical power* of these approaches appears to fall well short of what one might hope for in the tensor setting. As we will show in the sequel, in our setting of three dimensional tensors, the error rates achieved via these approaches are akin to what one would get by simply running a matrix recovery algorithm on each individual slice of the tensor (ignoring all

other slices!). In addition, *no* guarantees are available on the error of an individual slice which is particularly relevant since our original motivation is recovering a single slice (for instance, the likelihood of a transaction) using data from all slices.

1.2. Our Approach and Contributions

Our approach consists of a simple algorithm for learning the slices of a three dimensional tensor; applied to all slices this also results in an algorithm for the recovery of entire tensors from their noisy observations. Relative to the extant literature, we make the following contributions:

Statistical Power and Near-Optimal Rates for Noisy Tensor Recovery: Under a broad set of assumptions, we establish that our approach admits near-optimal guarantees on the rate of recovery for a broad class of three dimensional tensors from their noisy observations. We accomplish this by establishing both upper bounds on the estimation error incurred by using our approach as well as minimax lower bounds applicable to any approach. The guarantees we establish are stronger than those available for existing convex approaches. Simultaneously, we place looser restrictions on the underlying 'ground truth' tensor and the nature of the noise than those required for rigorous recovery via those convex approaches.

Our analysis also admits guarantees on error rates for individual slices which do not have a counterpart in the extant tensor recovery literature. From a pragmatic perspective this is particularly important in that such guarantees are relevant to our original motivation of recovering a single, specific slice while utilizing data across all slices.

Perhaps the most important aspect of our theoretical guarantees is that they quantify the precise extent to which side information can help with dealing with the problem of sparse data. In fact, a special case of our results establishes a broad set of conditions under which recovery error on any given single slice (and by implication the entire tensor) *decays linearly in the number of slices*. Colloquially this is equivalent to establishing how one may trade off sparsity in one type of data (say, transactions) for data of a different type (say, search data).

Empirical Evaluation: We conduct an extensive set of experiments that empirically demonstrate the most important advantages of our approach. In the first suite of experiments using synthetic data, we benchmark our recovery algorithm against a well-studied algorithm from the family of convex approaches. We observe that the recovery rate of our algorithm exactly matches that predicted by our theoretical results, and similarly the recovery rate of the convex algorithm matches the best known theoretical guarantee for convex approaches, which is weaker by an order of magnitude. Moreover, these experiments confirm that our approach is drastically more efficient from a computational standpoint.

In our second suite of experiments, we use real-world data from `Xiami.com`, a major online music streaming service. Much like the sales transaction data we have alluded to, this data is

extremely sparse, but our algorithm is able to leverage side information to outperform two common benchmarks by a significant margin. These experiments also demonstrate the scalability of our approach in practice. In particular, they are at a large enough scale that most existing tensor recovery algorithms, including convex approaches, are intractable. On the other hand, our algorithm is able to leverage data sparsity to operate on these large-sized tensors.

Scalability: In addition to the above, our approach is provably computationally efficient and, more importantly, easy to scale to massive datasets. Specifically, every step within our algorithm can be implemented as a sparse matrix-vector operation, and can thus scale to gigantic amounts of data using off-the-shelf computational tools. Resultantly, we find that our algorithm is typically faster than *a single iteration* of iterative 'operator splitting' algorithms used by incumbent approaches, and considerably simpler to implement.

We will also show that, with a minor modification, our approach can also apply to the *tensor completion* setting. Here the nature of the guarantee asks for the number of observed entires needed for eventual recovery as the dimensions of the tensor scale. We will see there that while our guarantee is not order-optimal (it is dominated by an alternative but computationally intractable approach), it is still the case that our approach can leverage side information. Specifically, the number of observations needed *per-slice* decreases as more slices are added for our approach.

1.3. Related Work

Our work falls into various diverse streams of literature relating to the nature of the data used, intended applications, and methodology. We describe these in the following subsections.

Practical Applications: The goal of the present paper is to use data to learn user-item propensities, which are a fundamental input to many operational and marketing activities. Perhaps the quintessential application is recommendation. There has been a great amount of work in designing recommender systems; see Ansari et al. (2000) and references in the nice survey by Adomavicius and Tuzhilin (2005). Recommendations, and more generally personalization, are used in all sorts of e-commerce activities and come in many different forms. For some examples in retail, see Linden et al. (2003) for a description of the various ways that Amazon.com recommends products to customers, including targeted emails and shopping cart recommendations. Outside of retail, Ghose et al. (2012) study hotel recommendations in the form of personalized results in online search engines. Chung et al. (2009) design an adaptive playlist of recommended songs, which is now fairly common in streaming media, e.g. Spotify's 'Discover Weekly' playlist.

The general problem of learning users' preferences extends to operational activities as well. For example, websites can drive usage, and therefore advertising revenue, through recommendations: Ansari and Mela (2003) study personalized emails and Besbes et al. (2015) study in-page recommendations to other pages. Fleder and Hosanagar (2009) analyze the impact of recommender systems

on the overall demand for all products, and Demirezen and Kumar (2016) look at ways in which recommendations and inventory should be considered jointly. Recommendations have also been studied in the context of promotions (Garfinkel et al. (2008)) and the firm's profit (Hosanagar et al. (2008)). Finally, Bodapati (2008) and Jacobs et al. (2016) seek to predict future purchase probabilities; these estimates can have important applications such as demand estimation.

Our work presents a framework for analyzing multiple, diverse data sources, specifically with user-item interaction data. Technological advances have allowed modern businesses to easily record and store massive amounts of data; Naik et al. (2008) survey the many sources of data and point out that the scale and diversity of data collected is constantly expanding. See Section A of the Appendix for a survey of articles dealing with user-item interaction data.

Statistical Methodology: In terms of methodology, our work broadly belongs to the class of collaborative filtering algorithms. Since their first introduction by Goldberg et al. (1992), collaborative filtering algorithms have become a mainstay approach in recommender systems. Our work fits within the matrix factorization approaches to these problems (Koren et al. (2009)).

A common criticism of collaborative filtering algorithms is their inability to work with sparse data (Ansari et al. (2000)). Our approach combines various sources of data to alleviate this sparsity. This fits within a recent stream of research into collaborative filtering algorithms that incorporate more data; see Shi et al. (2014) for a survey of these approaches. One such approach is matrix completion with side information, e.g. by Xu et al. (2013), Jain and Dhillon (2013), and Chiang et al. (2015). In this line of work, 'side information' refers to user and item features such as user demographics and product specifications. In contrast, our approach centers on interactions between users and items, but we will see that it is flexible enough to incorporate user and item features as well. Another approach is collective matrix factorization (Singh and Gordon (2008)), which studies matrix recovery using multiple matrices across multiple groups. In the setting of multiple matrices across two groups, our approach will in fact apply to a more general version of this problem and simultaneously allow us to leverage the modeling advantages of tensors.

Tensors have received significant attention recently in an attempt to generalize matrix recovery to higher dimensions; for a nice survey of tensor decompositions and methods, see Kolda and Bader (2009). Tensors have been applied in a wide range of areas, e.g. to model 3D images and video (Liu et al. (2013)), multivariate temporal data (Bahadori et al. (2014)), and multitask learning (Romera-Paredes et al. (2013)). See Mørup (2011) for a survey of more applications.

We will postpone a thorough review of the tensor recovery literature to §3.1, and for now briefly state that the majority of theoretical work so far has been on convex optimization formulations akin to nuclear norm minimization for matrix recovery. This approach was originally proposed by Liu et al. (2013) and Gandy et al. (2011), and has been analyzed extensively (see Romera-Paredes and Pontil (2013), Tomioka et al. (2011), Mu et al. (2013), Zheng and Tomioka (2015)). Finally, in

parallel to the noisy recovery problem, algorithms and guarantees have been shown for the tensor completion problem by Jain and Oh (2014), Huang et al. (2015), and Yuan and Zhang (2015). The goal there is typically to provide conditions under which the challenging goal of *exact* recovery is possible and those papers prove results under incoherence conditions similar to those made for matrix completion (see Candès and Tao (2010), Gross (2011), Recht (2011)).

Before proceeding, we make the disclaimer that we are focusing solely on three-dimensional tensors, so while our algorithm and guarantee may compare favorably against existing tensor recovery approaches, those approaches are specified for higher order tensors, while ours is not. On the other hand, there is a vast array of applications that makes studying 3D tensors specifically important, ranging from spatio-temporal analysis Bahadori et al. (2014) to neuroimaging Zhou et al. (2013) to the applications we analyze in the present work.

2. Model and Problem

One common problem that we have already alluded to is predicting customers' preferences for products from sales transaction data. In beginning to formalize this problem, let us label the customers $i = 1, \ldots, m_1$ and products $j = 1, \ldots, m_2$. To reduce notation, we will set $m_1 = m_2 = m$, but the entire analysis in what follows easily generalizes to a different number of customers and products. Say the data we have is a list of sales transactions that occurred over some previous time period, say the previous month. We encode this data in an $m \times m$ matrix called X, whose $(i,j)^{\text{th}}$ element X_{ij} is a binary indicator that equals 1 if customer i purchased product j in the last month, and 0 otherwise. To formulate a meaningful estimation problem, let us imagine that X_{ij} is a Bernoulli random variable with mean M_{ij}, and denote by M the matrix of these probabilities. More generally, we assume

$$X_{ij} = M_{ij} + \epsilon_{ij}$$

where ϵ_{ij} is a mean-zero noise term, and the noise terms are independent of each other but not necessarily identically distributed. To represent Bernoulli observations in this framework, we can let $\epsilon_{ij} = \text{Ber}(M_{ij}) - M_{ij}$. Our estimation problem is that of estimating M, having observed X.

Making progress on this problem clearly requires structural assumptions on M; estimating a completely general such matrix will require a prohibitive amount of data in a sense we will make precise shortly. One fairly general generative model for M is as follows: let us assume that every customer i is associated with some unknown vector of *latent* features, $u_i \in \mathbb{R}^r$, and every product j is similarly associated with an unknown latent vector $v_j \in \mathbb{R}^r$. We may then consider a generative model of the form

$$M_{i,j} = f(u_i, v_j)$$

where $f : \mathbb{R}^r \times \mathbb{R}^r \to \mathbb{R}$ is also unknown. If $f(\cdot,\cdot)$ were a bilinear form, so that $f(u_i, v_j) = u_i^\top S v_j$,

for some unknown S, then we may write M as

$$M = USV^\top$$

where $U \in \mathbb{R}^{m \times r}$ has as its ith row the vector $u_i^\top$, and similarly for V. Our task is now one of estimating a general rank r matrix M having observed X; the complexity of the underlying generative model is governed by r. For instance allowing $r = m$ permits a fully general model, but as we have noted, one that will be prohibitive to estimate from data. This generative model has been used extensively in operations and marketing to model product purchases (Grover and Srinivasan (1987)), ratings (Ansari et al. (2000)), and clickthroughs (Ansari and Mela (2003)), along with applications in myriad settings spanning information retrieval (Berry et al. (1995)) and latent semantic analysis (Papadimitriou et al. (1998)), computer vision problems such as facial recognition (Sirovich and Kirby (1987)) and background subtraction (Oliver et al. (2000)), sensor network localization (So and Ye (2007)), bioinformatics (Troyanskaya et al. (2001)), social network analysis (Liben-Nowell and Kleinberg (2007)), and web link analysis (Kleinberg (1999)), just to name a few.

The above setup demonstrates the manner in which the problem of predicting customers' preferences for products from a single type of data, e.g. transactions, can be formulated precisely as one of low-rank matrix recovery, which in turn is an incredibly well studied problem. It is well known that the minimax mean squared error of any matrix estimator is $\Omega\left(r/m\right)$. That is, for any estimator $\hat{M}(\cdot)$, we can construct a rank r matrix M for which,

$$\frac{1}{m^2}\mathsf{E}\|\hat{M}(X) - M\|_F^2 = \Omega\left(r/m\right)$$

In fact, estimators that achieve this lower bound have been constructed (e.g. Koltchinskii et al. (2011)). In light of these error rates, recovery is only meaningful in a relative sense if $\|M\|_F^2$ is larger in size than this error, i.e. if $\|M\|_F^2 = \Omega(rm)$. Now, if M encodes purchase probabilities (i.e. the entries of M are in $[0, 1]$), then $\|M\|_1 \geq \|M\|_F^2$, so that for recovery to be meaningful, we require $\|M\|_1 = \Omega(rm)$. In other words, we *require on average r expected transactions per user to estimate a rank r model*. Put another way, the sparsity of observed transactions limits the complexity of the model we can estimate. This limitation is often observed in practice – consider that in 2014, the online music streaming service Spotify had 10^7 active users and songs, i.e. $m \sim 10^7$, and had observed 10^{10} user-song pairs, such that the user listened to the particular song, i.e. $\|M\|_1 \sim 10^{10}$. The most complex models Spotify estimated for use in their recommendation engine had rank $\sim 10^3$ (Bernhardsson (2014)), which matches the error bound just described. This error bound also makes clear the challenge faced by, say, an online retailer where, as we have noted previously, on average only a single transaction is observed per user. It is this challenge that our work is meant to address.

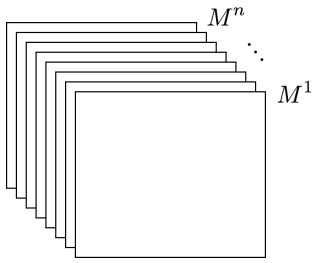


Figure 2: Matrices $M^1, \ldots, M^n$ represented as slices of a tensor

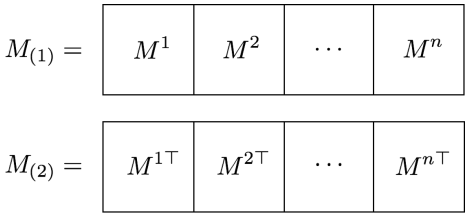


Figure 3: Graphical representation of the mode-1 and mode-2 unfoldings of M

2.1. Multiple Data Types and Tensors

Now suppose the data we have comes from observing multiple types of customer-product interactions. We label the different types of interactions $k = 1, \ldots, n$, and denote the value generated by an interaction of type k between customer i and product j as X_{ij}^k. As before, this data is a noisy observation of some underlying 'ground truth' value denoted M_{ij}^k that is never observed; for example X_{ij}^k might be a Bernoulli random variable with mean M_{ij}^k. In general, we assume

$$X_{ij}^k = M_{ij}^k + \epsilon_{ij}^k$$

where ϵ_{ij}^k is a mean-zero noise term, and the noise terms are independent of each other but not necessarily identically distributed. In order to model X_{ij}^k as a Bernoulli random variable with mean M_{ij}^k, one simply takes $\epsilon_{ij}^k = \text{Ber}(M_{ij}^k) - M_{ij}^k$. We let M^k and X^k denote the $m \times m$ matrices whose $(i,j)^{\text{th}}$ elements are M_{ij}^k and X_{ij}^k, respectively. Returning to our running example of predicting sales using data on a variety of distinct interactions, in our notation, X^1 may now be the matrix whose $(i,j)^{\text{th}}$ entry is 1 if and only if customer i purchased product j. X^2 could be the matrix whose $(i,j)^{\text{th}}$ entry is 1 if and only if customer i browsed product j, and so on. M^1 and M^2 would then encode the probabilities of those events, that we seek to estimate. Further matrices $X^3, \ldots, X^n$ would encode data observed from ratings, search, etc., taken as noisy observations of the corresponding matrices $M^3, \ldots, M^n$.

It will be convenient to introduce some basic notation for three-dimensional tensors. Let $M \in \mathbb{R}^{m \times m \times n}$ denote the three-dimensional tensor obtained by stacking the matrices $M^1, \ldots, M^n$ (see Figure 2). We call each of these matrices *slices* of the tensor, and continuing with the same notation, we denote the $(i,j)^{\text{th}}$ element of the k^{th} slice as M_{ij}^k. Without loss of generality, we will assume that every entry of M lies in $[-1, 1]$; this can always be satisfied by rescaling.

A common set of operations on tensors that we will use frequently here is that of *unfolding*, which essentially flattens the tensor into a matrix (see Figure 3). The mode-1 unfolding of M, denoted $M_{(1)}$, is the $m \times nm$ matrix whose columns are the columns of $M^1, \ldots, M^n$ (the order of the columns will not matter for us). Similarly the mode-2 unfolding, denoted $M_{(2)}$, is the $m \times nm$

matrix whose columns are the transposed rows of $M^1, \ldots, M^n$.[1]

2.2. Generative Model and Slice Rank

Just as in matrix recovery, we cannot hope to recover the tensor M without some assumption on its simplicity. In analogy with the setting of a single interaction (where we dealt with matrices), we now seek to propose a natural generative model for M. To that end, we assume that every customer i is associated with some unknown vector of *latent* features, $u_i \in \mathbb{R}^r$, and every product j is similarly associated with an unknown latent vector $v_j \in \mathbb{R}^r$. We may then consider that for each type of interaction $k = 1, 2, \ldots n$, we have

$$M^k_{i,j} = f^k(u_i, v_j)$$

where $f^k : \mathbb{R}^r \times \mathbb{R}^r \to \mathbb{R}$ is also unknown. This model says that the likelihood of an interaction between a specific user and product depends on feature vectors specific to the user and product respectively, through a function that is specific to the interaction. Given such a model, the hope is that data from any interaction can now contribute to learning the underlying user and product feature vectors, while the function determining a specific interaction is pinned down using data from just that interaction. It is this intuition that will eventually guide our recovery algorithms.

As before, if $f^k(\cdot, \cdot)$ were a bilinear form, so that $f^k(u_i, v_j) = u_i^\top S^k v_j$, then we may write each of the slices of M as

$$M^k = U S^k V^\top$$

where $U \in \mathbb{R}^{m \times r}$ has as its ith row the vector $u_i^\top$, and similarly for V. The key aspect of this model is that U and V do not depend on k; that is, the latent features are the same across matrices. This assumption relates the various matrices M^k to each other and allows for the potential of using other slices for learning. In particular, in addition to X^1, data from $X^2, \ldots, X^n$ can be used to improve the rate of recovery on M^1. Note that the possibility for some elements of S^k to be equal to zero means that different interactions do not need to involve the *same* latent factors, but rather all interactions draw from a small, shared pool of latent factors. As before the generality of this model is determined by the dimension of the latent space, r, a quantity that we will formalize as the *slice rank* of the tensor M:

Definition 1. *The slice rank of a tensor $M \in \mathbb{R}^{m \times m \times n}$, denoted* $\text{Slice}(M)$*, is the maximum of the ranks of its mode-1 and mode-2 unfoldings, i.e.* $\text{Slice}(M) = \max(\text{rank}(M_{(1)}), \text{rank}(M_{(2)}))$.

In the definition above, $\text{rank}(M_{(1)})$ is the number of latent customer features and $\text{rank}(M_{(2)})$ is the number of latent product features. Since these numbers may not be equal (corresponding

[1] For completeness, we can also define a third unfolding: for each row and column index, we can take the corresponding entries across all the slices to create a column vector in $\mathbb{R}^n$; these are the columns of the mode-3 unfolding, denoted $M_{(3)}$, an $n \times m^2$ matrix

to having a different number of customer and product features), the slice rank is defined as the maximum of the two quantities.

In what follows we will seek to recover tensors of low *slice rank* from their noisy observations – a formalization of the estimation problem that motivates this paper. Before doing so, we find it worthwhile to note that there is a long history of using tensors as a meaningful data structure to capture multivariate data, and this literature has yielded other notions of tensor rank. Specifically, the two most common of these notions are the *canonical (or CP) rank* and the *Tucker rank*, which have both been used in applications as diverse as psychometrics (Carroll and Chang (1970)), chemical analysis (Henrion (1994)), facial recognition (Vasilescu and Terzopoulos (2002)), chatroom analysis (Acar et al. (2005)), and web search (Sun et al. (2005)). As it turns out the requirement of a low slice rank is *weaker* than requiring either low CP rank or low Tucker rank. Specifically, denote by $\mathcal{CP}(r)$ the set of tensors $M \in \mathbb{R}^{m \times m \times n}$ that have CP rank at most r. Similarly denote by $\text{Tucker}(r, r, l)$ the set of tensors $M \in \mathbb{R}^{m \times m \times n}$ with Tucker rank at most (r, r, l) (see Section B of the Appendix for a formal definition of CP rank and Tucker rank). We then have:

Proposition 1. *The set of tensors $M \in \mathbb{R}^{m \times m \times n}$ with slice rank at most r,* $\text{Slice}(r)$*, contains $\mathcal{CP}(r)$ and* $\text{Tucker}(r, r, l)$*:*

$$\mathcal{CP}(r) \subseteq \text{Slice}(r), \quad \text{and} \quad \text{Tucker}(r, r, l) \subseteq \text{Slice}(r)$$

The proof of this result can be found in the Section B of the Appendix. In summary, this result establishes that our requirement of 'simplicity' subsumes important existing notions of simplicity for tensors. More importantly, as is evident from our presentation thus far, the notion of low slice rank has an elegant interpretation in our context as seeking out latent representations for customers and products along with functional forms that relate these latent representations to the likelihood of a specific interaction.

2.3. Our Problem

Our problem is to recover the tensor M from a noisy observation of its entries. In particular, our observation consists of the data tensor $X = M + \epsilon$, where the elements of the 'noise' tensor ϵ are independent with mean zero. We emphasize that, so far, we have not restricted the elements of ϵ to be Gaussian or even identically distributed, as is typically done.[2]

Our goal is to construct an estimator $\hat{M}(X)$ to minimize some loss function with respect to M. To reduce notation, we will use $\hat{M}(X)$ and $\hat{M}$ interchangeably. The usual loss function which we will take here is the mean-squared error (MSE):

$$\text{MSE}(\hat{M}) = \mathsf{E}\left[\sum_{k=1}^{n} \frac{1}{nm^2} \left\|\hat{M}^k - M^k\right\|_F^2\right].$$

[2]In our analysis, we will place substantially weaker requirements on ϵ.

We will refer to the problem of constructing an estimator to minimize MSE as *tensor recovery*. In addition, as we have noted, in many cases we might only be interested in recovering a single slice of the tensor (having observed all of X). For example, even with data from many types of customer-product interactions, we may be solely interested in predicting purchase probabilities. In these settings, the MSE is not an ideal loss function, as it measures average recovery error over all slices. Therefore, in addition to MSE, we will also consider the *slice mean-squared error* (SMSE):

$$\text{SMSE}(\hat{M}) = \max_{1 \le k \le n} \mathsf{E}\left[\frac{1}{m^2}\left\|\hat{M}^k - M^k\right\|_F^2\right],$$

and refer to the problem of constructing an estimator to minimize SMSE as *slice recovery*. SMSE is a more robust loss function, as it measures the maximum recovery error over all slices, and so a guarantee on the SMSE applies to every single slice. Also, since SMSE is always greater than or equal to MSE, any upper bound for SMSE will apply to MSE, and therefore the tensor recovery problem.

3. A Lower Bound and Incumbent Approaches

In the previous section, we recalled a minimax lower bound on the error rate one may hope to achieve under any algorithm for the problem of matrix recovery. Indeed, this bound motivated our problem of tensor recovery, where, loosely stated, we hope to use data from multiple types of customer-product interactions to improve our estimate of the likelihood of a given type of interaction. Our goal in this section will be to understand the extent to which data on additional types of interactions can help with the recovery of a specific type of interaction. Thus motivated we next establish a lower bound on the error rate that one may hope to achieve under *any* algorithm for our problem.

Proposition 2. *For any estimator $\hat{M}$, we can construct a tensor $M \in \text{Slice}(r)$ with entries in $[-1, 1]$ and a random noise ϵ with independent, zero mean entries, such that $X = M + \epsilon$ has entries in $[-1, 1]$ almost surely, and*

$$\text{SMSE}(\hat{M}) \ge \text{MSE}(\hat{M}) \ge C\left(\frac{r^2}{m^2} + \frac{r}{nm}\right),$$

where C is a universal constant.

The proof of Proposition 2 is presented in Section C of the Appendix, and relies on a carefully constructed ensemble of problem instances. The proposition lets us draw several key conclusions with regard to the power of side information:

1. **The Special Case of Matrix Recovery:** In the matrix recovery setting, i.e. $n = 1$ so that M is a matrix of rank r, we recover a minimax lower bound of $\text{MSE}(\hat{M}) = \Omega(r/m)$.

This bound is well-known (e.g. Candès and Plan (2011)) and a number of matrix recovery algorithms achieve this bound. Consequently, the naive approach of using an optimal matrix recovery algorithm *separately on each slice* achieves MSE $= O(r/m)$ and SMSE $= O(r/m)$. Beating such an approach is precisely the motivation for our work here.

2. **The Potential Benefit of Side-Information:** The impact of side-information is made precise by the dependence, on n, of our lower bound on achievable error rate. The minimax bound above suggests that additional side information might permit up to a *linear* reduction in recovery error up to a certain point beyond which additional side information cannot help. Specifically, the lower bound is dominated by an error term that scales like r/nm for n sufficiently small, and can be no smaller than r^2/m^2 irrespective of the amount of side information n. *How does side information alleviate the sparsity problem?* Recall that from our motivating example in the introduction, the minimax optimal recovery rate of $O(r/m)$ achievable with a single type of data ($n = 1$) implies that we need to observe on the order of r purchases per user, which was a problem since users make on average less than a single purchase a month. In the setting above, we may hope to get away with on the order of just r/n purchases per user as long as we recover a total of r^2 purchases across all users and products. Put another way, a retailer that can easily collect data on 10^2 types of interactions can hope to estimate a much richer model with $r = 10^2$.

In summary, the minimax lower bound established above raises the specter of dramatically increasing our ability to cope with sparse data provided we have access to sufficient side information. Do existing algorithms come close to achieving this lower bound?

3.1. Incumbent Approaches

The dominant approach to tensor recovery relies on convex optimization. To motivate this approach by example, consider the recovery of a low Tucker rank tensor from noisy observation X. We can formulate this problem as the following (hard) optimization problem:

$$\begin{aligned} \min_{Y} \quad & \|Y - X\|_F^2 \\ \text{s.t.} \quad & \operatorname{rank}(Y_{(i)}) \leq r_i, \;\; i = 1, 2, 3. \end{aligned} \tag{1}$$

That is, to choose a tensor Y that most closely 'matches' the observed data X, from the set of tensors with Tucker rank bounded by (r_1, r_2, r_3). For certain noise models, for example i.i.d. Gaussian noise, the solution to (1) would correspond to the maximum likelihood estimator.

Since this is a difficult problem (due to the rank constraints), the standard approach taken is to come up with a convex surrogate for the tensor rank. So for example, one such variant is:

$$\min_{Y} \|X - Y\|_F^2 + \sum_{i=1}^{3} \lambda_i \left\| Y_{(i)} \right\|_*, \tag{2}$$

where the ranks are replaced by the nuclear norms of the appropriate unfoldings, and in addition the rank constraints have also been dualized (in keeping with how such relaxations are presented in the literature) – the weights λ_i are chosen by the user and intuitively should encode prior knowledge of rank. This convex algorithm has been studied extensively (Gandy et al. (2011), Tomioka et al. (2011), Liu et al. (2013), and Signoretto et al. (2014)). In the case where $n = 1$, the mode-3 unfolding would naturally be removed from the objective, in which case the algorithm is precisely equivalent to the de facto convex formulation for matrix recovery, rendering the formulation a natural one. Tomioka et al. (2011) show that if the noise tensor has i.i.d. Gaussian entries with standard deviation σ, the estimator above achieves

$$\text{MSE}(\hat{M}) = O\left(\sigma^2(r/m + r/n)\right) \tag{3}$$

if the Tucker rank of M is (r,r,r). Recall that any tensor with Tucker rank (r,r,r) has slice rank at most r, so this result applies to a subset of tensors with slice rank r. There is good reason to believe that this guarantee is tight. For example, Tomioka et al. (2011) also show a very similar guarantee in a related setting where random linear combinations of the entries of M are observed, and Mu et al. (2013) show that in fact that guarantee is tight. Furthermore, in our experiments later on, we will empirically observe that the recovery rate scales as (3).

Taken together, this is disappointing – it shows that the convex approach above does not improve on recovering individual slices of the matrix via an optimal matrix completion algorithm and leaves a wide chasm between the minimax lower bound of Proposition 2 and the error rate achieved. Put simply, this approach *does not solve the data sparsity problem.* The issue of why existing convex approaches do not achieve optimality is a very interesting question. The main reason to hope that convex approaches might work in the tensor setting is because of their success in matrix recovery where the nuclear norm is easily shown as the *tightest* convex relaxation of the rank function. The major challenge in going from matrices to tensors is that there is not an obvious 'optimal' convexification of (1) above. The formulation (2) is one possible convexifiction, but it is provably *not* a tight convexification. Other convex problems have been suggested, for instance in Romera-Paredes and Pontil (2013), Mu et al. (2013), Tomioka and Suzuki (2013), but none of these proposals improve on the recovery guarantee above.

Outside of convex formulations, Suzuki (2015) recently proposed a Bayesian estimator that matches the lower bound in Proposition 2 for i.i.d. Gaussian noise, and tensors with low CP rank (as is evident from the definition of CP rank, this is significantly more restrictive than slice rank). Unfortunately, this procedure relies on a Monte Carlo approach in high dimension and its computational efficiency is unknown (i.e. we no longer have the computational efficiency guarantees that come with the convex approach).

In summary, we may conclude that employing existing tensor recovery machinery for the problem

at hand does not yield a solution to the data sparsity problem. In fact, using the de facto convex approaches for the problem cannot be expected to yield any improvement over the approach of individually recovering each slice of data. Now in contrast to taking the convex relaxation approach above, our algorithm works by directly (and efficiently) constructing a *feasible* solution to the original problem (1). The constructed solution is not necessarily an optimal solution to (1), but our core theoretical result shows that it is minimax optimal for noisy recovery.

4. An Algorithm for Slice Recovery

We will now present our algorithm for slice recovery and tensor recovery, which we will see is of a fundamentally different nature than the convex approaches, as just described. Recall from the setup that X is our observed data tensor and M is the ground truth tensor we are trying to recover. M is assumed to have slice rank r, which implies the existence of a decomposition wherein every slice M^k can be represented as $M^k = US^kV^\top$, where U, V are $m \times r$ matrices encoding latent customer and product features, and each S^k is an $r \times r$ matrix that captures the specific bilinear form for each slice M^k.

Up to this point, it has been convenient to think of each column of U and V as encoding a specific, possibly interpretable feature (e.g. customer demographics, product specifications), but in fact our algorithm is best understood when U and V are viewed as latent feature *spaces*. In particular, U and V each encode a linear subspace of $\mathbb{R}^m$, the subspaces spanned by their respective columns. Note that because our model places no restriction on the bilinear interaction terms S^k, the terms U and V in the decomposition $M^k = US^kV^\top$ are only unique up to the subspaces they span, i.e. we could replace U and V with any set of features that span the same feature space. For this reason, we will refer to U and V as features and feature spaces interchangeably, and for mathematical convenience, we will assume without loss of generality that the columns of U and V are orthonormal.

The algorithm proceeds in two stages, both of which we motivate from first principles:

Stage 1: Learning Subspaces: In the first stage, we use data from every slice to estimate the latent feature spaces U and V. Let us first focus on the procedure for learning U. Our first step in this regard is to construct the mode-1 unfolding $X_{(1)}$, which recall is the $m \times nm$ matrix whose columns are the columns of $X^1, \ldots, X^n$ (the order of the columns does not matter). We then compute $\hat{U}$, our estimate of U, as the first r left singular vectors of $X_{(1)}$. More precisely, assuming that $X_{(1)}$ admits the singular value decomposition $X_{(1)} = U_1\Sigma_1V_1^\top$, we set $\hat{U}$ to be the columns of U_1 corresponding to the r largest singular values. We denote this entire procedure with the shorthand

$$\hat{U} = \text{svds}(X_{(1)}, r). \tag{4}$$

Under very mild assumptions, it will turn out that $\hat{U}$ is a 'good' estimate of U. To see this, we can view $X_{(1)}$ as a noisy observation of $M_{(1)}$, which is a wide matrix with a row for each customer and a column for each product-interaction type pair. The key point is that the columns of $M_{(1)}$ span the feature space U, so if we were allowed to observe $M_{(1)}$, we could easily find U as the space spanned by the columns of $M_{(1)}$, i.e. its 'column space.' Instead, we observe $X_{(1)}$, which because of the added noise, does not have column space U; in fact, its columns are likely to span all of $\mathbb{R}^m$. Still, under mild assumptions on the nature of the noise, we can expect that the columns of $X_{(1)}$ lie approximately in U, and therefore estimate $\hat{U}$ as the orthonormal matrix that minimizes the function f defined as

$$f(U_1) = \min_{R_1 \in \mathbb{R}^{r \times mn}} \left\| U_1 R_1 - X_{(1)} \right\|_F .$$

For any r-dimensional subspace U_1, $f(U_1)$ measures how closely $X_{(1)}$ can be approximated by a matrix with column space U_1, and its minimizer $\hat{U}$ is precisely (4). This stage is exactly where we take advantage of having multiple slices of data, as the accuracy of $\hat{U}$ as an estimate of U is better the more slices we have, or the wider $M_{(1)}$ is. We will quantify this exactly in the next section.

To estimate V, we apply a similar procedure using the mode-2 unfolding $X_{(2)}$, which recall is the $m \times nm$ matrix whose columns are the transposed *rows* of $X^1, \ldots, X^n$. Just as in the previous discussion, $X_{(2)}$ is a noisy observation of $M_{(2)}$, whose column space is V. It follows that, under some assumptions on the noise, the columns of $X_{(2)}$ lie approximately in V and a natural estimate of V is

$$\hat{V} = \text{svds}(X_{(2)}, r).$$

Stage 2: Projection: The second stage works on each slice separately. Having estimated U and V as $\hat{U}$ and $\hat{V}$, it remains to estimate the bilinear terms S^k for each slice M^k. We do this by solving the following optimization problem for each slice:

$$\hat{S}^k = \operatorname*{argmin}_S \left\| \hat{U} S \hat{V}^\top - X^k \right\|_F . \tag{5}$$

To motivate this, suppose that instead of $\hat{U}$ and $\hat{V}$, we had access to U and V exactly. Then knowing that M^k takes the form $US^kV^\top$ for some S^k, our estimate of M^k should take this same form. Therefore, we would use the closest approximation to X^k of this form, essentially 'projecting' X^k onto the feature spaces U and V. Since we only have $\hat{U}$ and $\hat{V}$, we use the closest approximation of the form $\hat{U}S^k\hat{V}^\top$ instead, as in (5).

The above is a least-squares problem and as such admits a closed-form solution: assuming that $\hat{U}$ and $\hat{V}$ are orthonormal, the solution of (5) is $\hat{S}^k = \hat{U}^\top X^k \hat{V}$, and so our final estimate of M^k can be written as

$$\hat{M}^k = \hat{U}\hat{U}^\top X^k \hat{V}\hat{V}^\top.$$

One nice interpretation here is that $\hat{U}\hat{U}^\top$ and $\hat{V}\hat{V}^\top$, which we will denote $P_{\hat{U}}$ and $P_{\hat{V}}$, are projection

operators: left multiplying a matrix by $P_{\hat{U}}$ replaces each of its columns with the orthogonal projection onto the space $\hat{U}$, and similarly right multiplying by $P_{\hat{V}}$ replaces rows by the orthogonal projection onto $\hat{V}$. The resulting matrix lies in the feature spaces $\hat{U}$ and $\hat{V}$.

The entire slice learning algorithm is outlined in **Algorithm 1**. The algorithm and our forthcoming analysis are easily extended to using different values of r when forming $\hat{U}$ and $\hat{V}$. This may be advantageous when the latent customer and product feature spaces of M are of significantly different dimension, but we use a single value r here for ease of notation.

Algorithm 1: Slice Learning

Input : X, r

1. $\hat{U} \leftarrow \text{svds}(X_{(1)}, r)$
2. $\hat{V} \leftarrow \text{svds}(X_{(2)}, r)$
3. $\hat{M}^k \leftarrow P_{\hat{U}} X^k P_{\hat{V}}, \quad k = 1, \ldots, n$

Output : $\hat{M}^k, \quad k = 1, \ldots, n$

4.1. Practical Considerations

We conclude this section with a discussion of practical implementation and computation issues:

Knowing r: The slice learning algorithm above takes the slice rank r as input, but in some settings we may not know r in advance. In particular, we do not know the ranks of the two unfoldings $M_{(1)}$ and $M_{(2)}$ in advance. This is a common challenge in low-rank matrix recovery, though in that setting the problem has proven to be relatively benign. Specifically, there exists a wide array of rank estimation methods that we may borrow from, including cross-validation (Wold (1978), Owen and Perry (2009)), visual inspection of plotted singular values (Cattell (1966)), and bayesian methods (Hoff (2012)).

Perhaps the simplest approach when r is not known is a 'universal' thresholding scheme where we only preserve singular vectors corresponding to singular values above a certain easily precalculated threshold. This has been shown to work in the matrix recovery setting (Chatterjee (2014), Gavish and Donoho (2014)), and we anticipate that such a scheme will work just as effectively here. In particular, $X_{(1)}$ is the sum of the signal $M_{(1)}$ and the noise $\epsilon_{(1)}$, so if the singular values of $\epsilon_{(1)}$ are all significantly lower than the r non-zero singular values of $M_{(1)}$, then by Weyl's inequality, $X_{(1)}$ will have r singular values that are much larger than the rest.

To make this more precise, consider the following argument, which also serves as an introduction to the type of arguments used in the next section. Suppose the terms of $\epsilon_{(1)}$ are i.i.d. with unit variance and bounded fourth moment; we will assume a more general noise model later. To fix the signal-to-noise ratio, assume that the terms of $M_{(1)}$ are of constant order, so $\|M_{(1)}\|_F^2 = \Theta(m^2 n)$. First, $M_{(1)}$ has rank r (for some unknown r), so if the non-zero singular values of $M_{(1)}$ are all within a constant order of each other, then these singular values will be of $\Theta(\|M_{(1)}\|_F/\sqrt{r})$, or $\Theta(m\sqrt{n/r})$.

At the same time, by the Bai-Yin law (Yin et al. (1988)), the largest singular value of $\epsilon_{(1)}$ does not scale greater than $O(\sqrt{mn})$. Therefore, a clear separation forms asymptotically between the non-zero singular values of $M_{(1)}$ and the singular values of $\epsilon_{(1)}$. By Weyl's inequality, $X_{(1)}$ will have r singular values of $\Theta(m\sqrt{n/r})$ and its remaining singular values are of $O(\sqrt{mn})$. It follows that choosing a threshold in this gap and retaining the singular values greater than this threshold will closely approximate using the correct value of r. This result can be formalized as a theorem, but we do not do so here.

Computation with large tensors: An important consideration is the scale at which the slice learning algorithm needs to operate – the nature of the applications necessitates that predictions be made rapidly, as the data is constantly changing and up-to-date output is required. Therefore, computational efficiency is of paramount importance in practice.

The only computationally intensive step in the slice learning algorithm is in Stage 1, which requires the computation of two partial SVDs. Again, the largest retailers have upwards of 10^8 unique customers and products ($m \sim 10^8$), and can easily collect data on hundreds of interactions ($n \sim 100$), at the very least. For dense matrices, this massive scale renders generic SVD algorithms intractable. Fortunately, while the ambient dimensions of the input data are large, the data is itself typically quite sparse. Data sparsity has so far been treated as a disadvantage, since it limits the complexity of the models we can learn, but it is a key advantage from a computational standpoint – there exist mature linear algebraic algorithms that compute the top singular vectors while exploiting data sparsity, for example using Lanczos iterations as in the PROPACK algorithm (Larsen (1998)). Specifically, these algorithms rely on a power method that repeatedly applies a matrix-vector multiplication subroutine. Since the matrix in question (the unfolding of the sparse tensor) is sparse, this operation can be implemented with running time linear in the number of non-zero elements of the matrix. This is already drastically less computation than convex approaches to this task, which by and large are solved with iterative algorithms (Gandy et al. (2011)) that require performing dense SVDs multiple times.

Customer/Product Side Information: Our generative model includes a set of latent customer and product features U and V, and Stage 1 of our algorithm essentially works by estimating these features jointly across all slices of data. We will see that the performance of our algorithm boils down to how well U and V can be estimated, and that going from a single slice of data to many slices interpolates between standard matrix recovery (with no prior knowledge of U or V), and recovery given U and V exactly.

In many cases, we may have side information in the form of explicit features about customers or products that are believed to be relevant, e.g. customer demographics and product specifications. This is the subject of a line of work in matrix recovery (Xu et al. (2013), Soni et al. (2016)) that seeks to recover a single slice, assuming that the feature spaces U and V are known exactly.

The approach we have laid out is flexible enough to incorporate the same kind of customer and product features, without requiring the assumption of knowing U and V exactly. Suppose that in addition to the data tensor X, we have ℓ customer features which we encode in the $m \times \ell$ matrix A. We can include this information in our algorithm by expanding the estimated feature space $\hat{U}$ with these extra features, i.e. after producing the estimate $\hat{U}$ in Stage 1 as usual, we can replace $\hat{U}$ with the subspace spanned by the columns of $\hat{U}$ and A, and then execute Stage 2 normally. The equivalent procedure can be done with $\hat{V}$ if we have product features.

5. Recovery Guarantees for Slice Learning

The goal of this section is to provide a statistical recovery guarantee for our slice learning algorithm. In particular, we are looking for a guarantee that improves upon the naive approach of ignoring side information and recovering each slice separately. As discussed in Section 3, that naive approach, which is effectively a matrix recovery algorithm, can achieve SMSE $= O(r/m)$.

So far, we have made the assumption that the ground truth tensor has low slice rank, which as we have seen, restricts the complexity of the underlying generative model and offers the possibility of improving on the matrix recovery rate. However, beating this rate is impossible without making further assumptions. Specifically, consider the case where M has $n-1$ slices, all of whose entries are identically 0, and a single non-trivial, low-rank slice, which we take without loss of generality as the first slice. Further, assume that the noise tensor has independent unit variance Gaussian entries on the first slice, and is identically zero on the remaining $n-1$ slices. Though the ground truth tensor has low slice rank, the problem of recovering the first slice is now literally no different than the problem of matrix recovery on the first slice since the remaining slices are superfluous.

To this end, we define a structural parameter for M that we will eventually see controls the rate at which learning across slices is possible. Letting $\sigma_r^2(M_{(1)})$ and $\sigma_r^2(M_{(2)})$ denote the r^{th} largest singular values of $M_{(1)}$ and $M_{(2)}$, respectively, we define the *learning rate* as follows:

Definition 2. *The learning rate of a tensor $M \in \mathbb{R}^{m \times m \times n}$ with slice rank r, denoted γ_M, is defined as*

$$\gamma_M = \frac{r}{m^2 n} \min\left(\sigma_r^2(M_{(1)}), \sigma_r^2(M_{(2)})\right)$$

We will shortly see how γ_M plays the role of a rate of learning. For now, we merely comment on the range of values one might reasonably expect this quantity to take. On the low end observe that by Marčenko and Pastur (1967), if the noise tensor were i.i.d., the (squared) singular values of the noise tensor unfoldings $\epsilon_{(1)}$ and $\epsilon_{(2)}$ are $O(mn)$. A minimal requirement is that the singular values of $M_{(1)}$ and $M_{(2)}$ dominate those of the noise unfoldings which would in turn imply that $\gamma_M = \Omega(r/m)$, so the loosest requirement we can reasonably place on γ_M is to require $\gamma_M = \Omega(r/m)$.

At the other end of the spectrum, given that the entries of M are required to be bounded,

we must have that $\|M_{(1)}\|_F^2 = O(m^2 n)$, so that $\sigma_r^2(M_{(1)}) = O(m^2 n/r)$. This in turn implies that $\gamma_M = O(1)$, and the strongest requirement we can place is to require γ_M be a constant. In fact, in terms of scaling, it is reasonable to treat γ_M as constant: the condition $\|M_{(1)}\|_F^2 = \Theta(m^2 n)$ is satisfied for all but trivial examples like the one just described, and then $\sigma_r^2(M_{(1)}) = \Theta(m^2 n/r)$ follows as long as the largest and smallest non-zero singular values of $M_{(1)}$ are not significantly different, i.e. a parameter akin to the condition number is bounded.[3]

Before proceeding with a statement of our main result, we will place an assumption on the noise (our only non-trivial assumption thus far). Specifically, we will require the noise to be 'balanced' in a certain sense.

Assumption 1 (Balanced Noise). *Let v be the tensor whose entries are the variances of the corresponding entries of ϵ. Specifically, $v_{i,j}^k = \mathsf{E}[(\epsilon_{i,j}^k)^2]$. The noise ϵ is said to be* balanced *if the row-sums of $v_{(1)}$ are all equal and the row-sums of $v_{(2)}$ are all equal.*

The assumption above is trivially satisfied in the case of i.i.d. noise, and in particular the case of i.i.d. Gaussian noise that is often studied in the matrix and tensor recovery setting. Refinements of our result allow for weaker versions of this assumption; we will see in the next subsection that we can permit a certain amount of discrepancy in the row sums of $v_{(1)}$ and $v_{(2)}$, and allow this to grow with m and n. We are now ready to state our main result.

Theorem 1 (Balanced Noise). *Assume the entries of M lie in $[-1, 1]$. If the entries of ϵ are independent, mean-zero, and $\mathsf{E}[(\epsilon_{ij}^k)^6] \leq K^6$, and if furthermore ϵ is balanced, then there exists a constant $c(K)$ that depends only on K such that for the slice learning algorithm,*

$$\mathrm{MSE}(\hat{M}) \leq \mathrm{SMSE}(\hat{M}) \leq c(K) \left[\frac{r^2}{m^2} + \frac{r^2}{\gamma_M^2 mn} \right].$$

The proof of Theorem 1 can be found in Section D in the Appendix. We next evaluate this result in light of the minimax guarantee established in Proposition 2, and more generally, our broader goal of slice recovery:

1. Learning from slices: As discussed earlier, at the very least we expect $\gamma_M = \Omega(r/m)$. Even in this setting, we see that provided n is sufficiently large, Theorem 1 guarantees an SMSE (and consequently MSE as well) that is $O(r^2/m^2)$. In particular, for n sufficiently large, we obtain a recovery rate that meets the leading term of the minimax guarantee in Proposition 2. Of course, this is *substantially* better than the available guarantee for the naive approach which was $O(r/m)$.
2. High learning rate: As γ_M grows, so does our ability to learn across slices. Specifically, for $\gamma_M = \Theta(1)$, Theorem 1 guarantees MSE $\leq$ SMSE $= O(r^2/m^2 + r^2/mn)$. This is within a

[3]The condition number of a matrix can be defined as the ratio of its largest singular value to its smallest singular value. The ratio we describe here is defined only for the *non-zero* singular values.

factor of r off from the lower bound of MSE $= \Omega(r^2/m^2 + r/mn)$ in Proposition 2. Put a different way, we achieve SMSE $= O(r^2/m^2)$ with only $n = \Omega(m)$ slices of side information.

3. The recovery rate in Theorem 1 is for low slice rank tensors, which includes tensors with low Tucker rank and CP rank. We emphasize that the rate MSE $= O(r^2/m^2 + r^2/mn)$ greatly improves upon the best known theoretical guarantees for noisy recovery of low Tucker rank tensors, and for convex optimization approaches to recovering low CP rank tensors.

5.1. Relaxing the Balanced Noise Assumption

While the balanced noise assumption is already a generalization of many frequently studied noise models, it is worth considering how our algorithm performs when this assumption does not hold. To do so, we will present a more general version of Theorem 1 that relaxes the balanced noise assumption and reflects the recovery error caused by 'unbalanced' noise.

To proceed, we need to precisely quantify the concept of 'unbalanced' noise. Recall that if v is the tensor whose entries are the variances of the corresponding entries of ϵ, i.e. $v_{ij}^k = \mathsf{E}[(\epsilon_{ij}^k)^2]$, then ϵ is balanced if the row-sums of $v_{(1)}$ are equal and the row-sums of $v_{(2)}$ are equal. An equivalent way to state this assumption is $\mathsf{E}[\epsilon_{(1)}\epsilon_{(1)}^\top] = \rho_1 I_m$ and $\mathsf{E}[\epsilon_{(2)}\epsilon_{(2)}^\top] = \rho_2 I_m$ for some constants ρ_1 and ρ_2, where I_m is the $m \times m$ identity matrix. To see this, note that the off-diagonal elements of $\mathsf{E}[\epsilon_{(1)}\epsilon_{(1)}^\top]$ and $\mathsf{E}[\epsilon_{(2)}\epsilon_{(2)}^\top]$ are always equal to zero when the noise terms are independent, and the diagonal elements are exactly the row sums of $v_{(1)}$ and $v_{(2)}$, so the balanced noise assumption states that $\mathsf{E}[\epsilon_{(1)}\epsilon_{(1)}^\top]$ and $\mathsf{E}[\epsilon_{(2)}\epsilon_{(2)}^\top]$ are multiples of the identity matrix.

For general, possibly unbalanced noise, it turns out that the appropriate quantities to measure the level of 'unbalance' in the noise are

$$\min_\rho \frac{1}{m} \left\| \mathsf{E}[\epsilon_{(1)}\epsilon_{(1)}^\top] - \rho I_m \right\|_F^2 \quad \text{and} \quad \min_\rho \frac{1}{m} \left\| \mathsf{E}[\epsilon_{(2)}\epsilon_{(2)}^\top] - \rho I_m \right\|_F^2 .$$

These quantities measure how far $\mathsf{E}[\epsilon_{(1)}\epsilon_{(1)}^\top]$ and $\mathsf{E}[\epsilon_{(2)}\epsilon_{(2)}^\top]$ are from a multiple of the identity matrix. One nice interpretation is that the quantities correspond to population variances, one each for the row-sums of $v_{(1)}$ and the row-sums of $v_{(2)}$. We denote the maximum of these two quantities as δ^2, and can now state our more general result:

Theorem 2. *Assume the entries of M lie in $[-1, 1]$. If the entries of ϵ are independent, mean-zero, and $\mathsf{E}[(\epsilon_{ij}^k)^6] \le K^6$, then there exists a constant $c(K)$ that depends only on K such that for the slice learning algorithm,*

$$\text{SMSE}(\hat{M}) \le c(K) \left[\frac{r^2}{m^2} + \frac{r^2}{\gamma_M^2 mn} + \frac{r^2\delta^2}{\gamma_M^2 m^3 n^2} \right].$$

To interpret the guarantee in Theorem 2, consider the range of values that δ^2 might take. The lowest possible value is $\delta^2 = 0$, which corresponds to the balanced noise setting, and in this case we recover Theorem 1 exactly. On the other hand, each row of $v_{(1)}$ and $v_{(2)}$ contains mn elements,

so their row-sums may scale as $O(mn)$, and thus in the worst case, $\delta^2 = O(m^2n^2)$. In this worst case setting, Theorem 2 does not improve upon the guarantee for the naive approach. However, the recovery rate in Theorem 2 matches that in Theorem 1 as long as $\delta^2 = O(m^2n)$. Also, since δ^2 measures the population variances of the two sets of row-sums, it is highly robust to settings where only a few row-sums are significantly 'unbalanced'. All of this suggests that *Theorem 1 holds even if the balanced noise assumption is significantly relaxed.*

To this point, our work has applied to the problem of noisy tensor recoveryb, a framework that addresses settings such as the retail example and our experiment with music streaming data. As discussed in Section 1, there are settings and applications where the existing data instead can be represented as a partially observed tensor, i.e. the tensor completion problem. The challenge here is to design algorithms that are optimal in terms of the *number of observed entries* required for exact recovery. Now, under the assumption that entries are observed uniformly at random, it is possible to map completion problems to noisy recovery problems using a technique developed for matrix completion (Achlioptas and McSherry (2007), Keshavan et al. (2010), Chatterjee (2014)). We discuss this topic in Section G of the Appendix, where we (a) use this same device to adapt the slice learning algorithm to tensor completion, (b) state a Corollary to Theorem 2 that characterizes the requisite number of observed entries, (c) compare this result to existing tensor completion algorithms, and (d) show results of synthetic tensor completion experiments.

6. Experiments

We performed two sets of experiments to evaluate the slice learning algorithm, the first using randomly generated tensors, and the second using a real-world dataset. This section describes these experiments and their results in detail, wherein the following points emerge:

1. The slice learning algorithm drastically outperforms convex algorithms by an order of magnitude, even though convex algorithms require drastically greater computation. On tensors with low slice rank, the slice learning algorithm outperforms a convex benchmark we propose that is in the spirit of incumbent approaches; this is a 'home court' setting, as the recovery guarantees from the previous section show that our algorithm is expected to recover these tensors. More surprisingly, on tensors with low Tucker rank, the slice learning algorithm also outperforms an existing convex benchmark designed specifically for low Tucker rank tensors.
2. On real-world, sparse data, the slice learning algorithm outperforms two common benchmark approaches. This suggests that real data, when represented as a tensor, indeed exhibits the type of structure that the slice learning algorithm is able to exploit.
3. By leveraging sparsity, the slice learning algorithm can be used on large datasets, in regimes where common operations such as taking the SVD of a dense matrix are intractable.

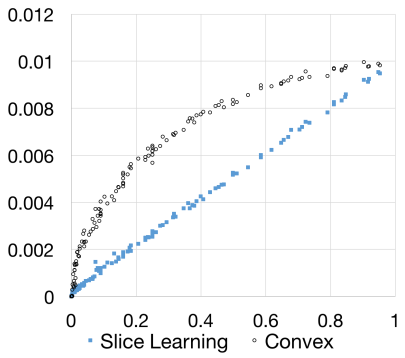


Figure 4: Comparison of slice learning and convex approach for noisy recovery of low Tucker rank tensors. MSE vs. $(r/m)^2$ is plotted for each replication.

6.1. Synthetic Recovery Experiments

We conducted a number of experiments on randomly generated tensors to test the performance of the slice learning algorithm, for learning individual slices and the tensor as a whole. In all of our experiments, we compare against a benchmark convex optimization approach.

6.1.1. Noisy Recovery and Tucker Rank

For our first experiment, we randomly generated $m \times m \times m$ tensors with Tucker rank (r, r, r), where in each replication, m was drawn uniformly from the integers between 10 and 50, and r was then drawn uniformly from the integers between 1 and m. In each replication, we randomly drew orthonormal $m \times r$ matrices U, V, and W, along with an $r \times r \times r$ tensor S with entries drawn from the standard normal distribution. The ground truth tensor was then constructed in the canonical way from U, V, W, and S (see Appendix, Section B), and the observed data tensor X was constructed by adding independent mean-zero gaussian noise of standard deviation 0.1 to each entry. This is nearly identical to the experimental setup in Tomioka et al. (2011).

We compared the slice learning algorithm against the convex algorithm that minimizes (2) in §3.1. Recall that this is one of the few well-studied algorithms with a theoretical recovery guarantee (see Tomioka et al. (2011)). We solved this via the Douglas-Rachford splitting method, as described by Gandy et al. (2011). Note that the slice learning algorithm requires an estimated rank as input, and in this experiment, the algorithm was given the true rank r in each replication. On the other hand, the convex objective (2) has a parameter λ that encodes knowledge of the rank; to level the playing field, in each iteration we solved (2) for values of λ ranging from 2^{-2} to 2^5 and reported the best performance among all of these.

We performed 100 replications. The results are depicted in Figure 4, where each replication is represented by two points, one for each algorithm. Each point represents the MSE versus the value $(r/m)^2$ for that particular replication. The reason we plot the MSE against $(r/m)^2$ is that

Theorem 1 predicts the MSE of the slice learning algorithm should scale linearly with this value, which appears to be the case in Figure 4. On the other hand, the best theoretical results for the convex algorithm state that the MSE scales linearly with r/m, i.e. sublinear in $(r/m)^2$; Figure 4 confirms that this is indeed the case. That is to say the slice learning algorithm outperforms the convex algorithm in recovering tensors of low Tucker rank, even though the convex algorithm is *suited specifically for tensors of low Tucker rank*.

In terms of computation, the slice learning algorithm is also superior. The first order Douglas-Rachford splitting method for solving (2) requires three SVDs in each iteration. Compared to the two SVDs required by the entire slice learning algorithm, this means *each iteration is slower than the entire slice learning algorithm*, and in our experiments, the whole algorithm was consistently at least 10 times slower. Along these lines, there is ongoing progress in improving the computational efficiency of convex optimization approaches, e.g. by Liu et al. (2014).

6.1.2. The Value of Side Information

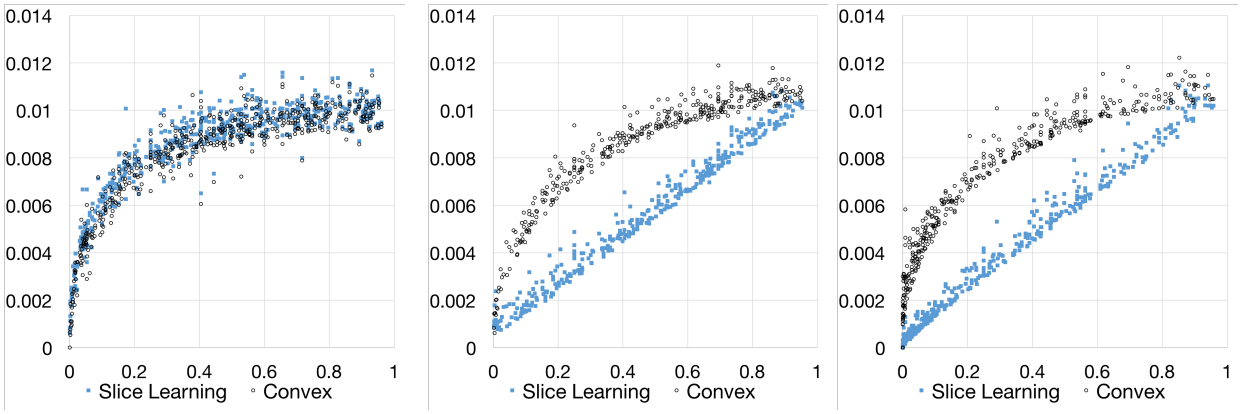
We performed a similar experiment to test the recovery of tensors with varying numbers of slices, and particularly recovery in terms of SMSE. We randomly generated $m \times m \times n$ tensors of slice rank r where in each replication, m was drawn randomly between 10 and 50, then r was drawn randomly between 1 and m, and finally n was set equal to either 1, r, or m. In each replication, we randomly drew orthonormal $m \times r$ matrices U and V, along with $r \times r$ matrices $S^1, \ldots, S^n$ with entries drawn from the standard normal distribution, and we set the slices of the ground truth tensor to be $M^k = US^kV^\top$. Independent mean-zero gaussian noise of standard deviation 0.1 was then added as before.

We used a similar convex algorithm as a benchmark:

$$\operatorname*{argmin}_Y \sum_{i=1}^{2} \lambda \left\| Y_{(i)} \right\|_* + \|Y - X\|_F^2 . \tag{6}$$

This objective is almost identical to (2), except that it does not include the nuclear norm of the mode-3 unfolding. This is catered to recovering low slice rank tensors, as it imposes no penalty on the complexity between slices. We solved this with a similar Douglas-Rachford splitting method that requires two singular value decompositions in each iteration. Just as in (2), the parameter λ encodes knowledge of rank, and so we solve (6) for λ ranging from 2^{-2} to 2^5 and report the best performance among all of these.

We performed 400 replications. In each replication, we measured the SMSE of the slice learning algorithm and the convex algorithm (6). The results for each of the three cases ($n = 1, r, m$) are depicted in three separate plots in Figure 5. Figure 5a shows that with a single slice, both algorithms have SMSE sublinear in $(r/m)^2$; this is to be expected as the exercise is equivalent to matrix recovery where the best achievable rate is r/m (Proposition 2). Figure 5c shows that the slice



(a) $n = 1$ **(b)** $n = r$ **(c)** $n = m$

Figure 5: Comparison of slice learning and convex approach for noisy slice recovery of low slice rank tensors with varying numbers of slices. SMSE vs. $(r/m)^2$ is plotted for each replication.

learning algorithm achieves the gold standard performance of SMSE linear in $(r/m)^2$ with $n = m$ slices, while the convex algorithm is still sublinear. The good performance of the slice learning algorithm is to be expected since in Theorem 1 we were able to show that $n = m$ slices are sufficient to achieve the $(r/m)^2$ rate. The surprising result is that Figure 5b is almost identical to Figure 5c, implying that $n = r$ slices are sufficient to achieve this same rate. This suggests that there are settings where slice learning can greatly outperform standard matrix learning, with only *very little* side information.

6.2. Experiments on Real Data

In addition to synthetic experiments, we also performed experiments using real-world data to address a number of important challenges that occur in practice. In particular, these experiments differ greatly from the previous synthetic experiments in that the data is very large and sparse. The data is from `Xiami.com`, a major online music streaming service where users may listen to songs and share their own music. Within the service, users can interact with songs in different ways: they can 'Listen' to, 'Download', and 'Collect' any song offered by the service. The collect interaction is especially important, as it is a strong signal of a user's affinity for a song, but is performed with the least frequency in our data. Our dataset[4] is a sample of all three interaction types between users and songs over a six month period in 2015. For the experiments, we represent this data as a three-dimensional tensor X with three slices, one for each type of interaction, with binary entries indicating whether that particular user-song interaction occurred during the six month period. Just as in our motivating retail example, we model this as Bernoulli noise, i.e. we assume that the data X is a Bernoulli observation of some ground-truth tensor M of probabilities: $X^k_{ij} \sim \text{Ber}(M^k_{ij})$.

We compared the slice learning algorithm against two benchmarks. The first benchmark is the

[4] `https://tianchi.shuju.aliyun.com/`

naive approach of using a matrix recovery algorithm separately on each slice, which is still today a typical approach to collaborative filtering. The second benchmark is a more sophisticated approach: form one of the unfoldings and use a matrix recovery algorithm on the unfolding; we will refer to this algorithm as the *matrix* approach. This approach nicely exploits tensor structure and can be performed at large scale; Mu et al. (2013) studied such algorithms and demonstrated theoretical guarantees for high-dimensional tensors.

Both benchmarks require a matrix recovery algorithm, and for our experiments, we recovered matrices by replacing hidden entries with zeros and then calculating a low-rank approximation of this modified matrix (Achlioptas and McSherry (2007), Keshavan et al. (2010)). This procedure requires roughly the same computational budget as the slice learning algorithm. Finally, since our experiments were quite large (see Table 1), calculating SVDs using standard libraries meant for dense matrices was not feasible. Instead, we used the off-the-shelf software package PROPACK[5] (Larsen (1998)), which exploits data sparsity through the iterative algorithms described in Section 4.1.

6.2.1. Experimental Setup

Since the data is binary, we evaluated performance vis-à-vis a binary classification task, i.e. the task of classifying entries as being equal to zero or one, on half of the entries. Since each entry of the tensor corresponds to whether or not a particular user interacted with a particular song, we will refer to the values zero and one as 'did not occur' and 'did occur', respectively.

Our algorithm and the two benchmark algorithms return complete tensors with continuous values. To convert these continuous values to classifications of occurring and not occurring, we chose a fixed threshold θ and classified any entry exceeding θ as occurring, and the remaining entries as not occurring. To evaluate an algorithm's performance, we can calculate two important performance metrics: the true positive rate (TPR) and the false positive rate (FPR). Out of all the hidden entries that occurred, the TPR is the proportion that the algorithm correctly classified as occurring, and out of all the hidden entries that did not occur, the FPR is the proportion that the algorithm incorrectly classified as occurring.

A 'good' classification has a high TPR and low FPR. This might inform the choice of the threshold θ, but unfortunately there is a tradeoff: both the TPR and FPR are non-increasing in θ, so it is impossible to improve on both metrics just by changing θ. In particular, by varying θ, a given algorithm produces a set of classifications ranging from classifying all entries as not occurred (TPR and FPR equal to 0) to classifying all entries as occurred (TPR and FPR equal to 1). To evaluate the performance of an algorithm independently from the choice of θ, we can plot each of these classifications in receiver operating characteristic (ROC) space, and calculate the area under

[5] `http://sun.stanford.edu/~rmunk/PROPACK/`

the resulting curve (AUC), as in Figure 6. The AUC is always in $[0, 1]$, and a higher value generally signifies greater accuracy; as a benchmark, the AUC of a random classification is expected to be 0.5.

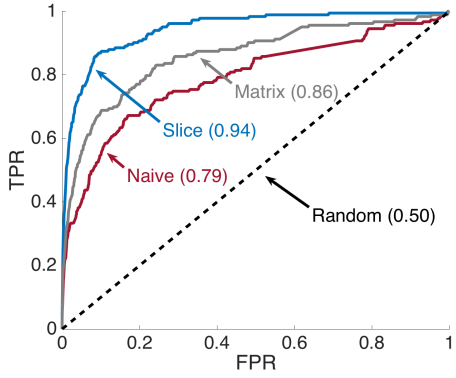


Figure 6: Sample ROC curves for recovering the Collect slice. These curves were generated from a single replication of the experiment in the first row of Table 1. For each algorithm, TPR vs. FPR is plotted, and the AUC is reported.

Users	Songs	Sparse	Naive	Matrix	Slice
2,412	1,541	5.7	0.76 (4)	0.83 (7)	0.91 (14)
4,951	2,049	4.1	0.73 (7)	0.78 (12)	0.91 (15)
27,411	3,472	2.0	0.66 (9)	0.67 (19)	0.87 (20)
23,300	10,106	1.0	0.86 (1)	0.87 (1)	0.95 (18)
53,713	10,199	0.6	0.82 (3)	0.84 (1)	0.95 (13)

Table 1: Summary of experiments on Xiami data for recovering the Collect slice. Each row corresponds to an experiment on a subset of the data. Columns 'Users' and 'Songs' show the number of users and songs in each experiment, and 'Sparse' gives the average number of collects per user in the data. Results for the naive benchmark, the matrix-based benchmark, and the slice learning algorithm are shown in the last three columns. The average AUC over 10 replications is reported, along with the rank in parentheses.

6.2.2. Summary of Results

The results of the experiment, in terms of recovering the Collect slice, are summarized in Table 1. We performed experiments on tensors of five different sizes, described in the first two columns. These tensors were created by selecting subsets of the densest rows and columns of the original data set. The third column shows the sparsity of the Collect slice in each of the five tensors, measured in average number of collects per user. Note that the data is extremely sparse – in the largest tensor, we observe less than a single collect per user.

For each tensor, 10 replications were performed, where each replication included a resampling of observed entries, followed by performing all three algorithms with ranks ranging from 1 to 20. For each algorithm and rank, the AUC was averaged over all 10 replications. The best average AUC of each algorithm is reported in the last three columns of Table 1, and the best ranks are given in parentheses.

In absolute terms, the slice learning algorithm performs very well, with an AUC consistently above 0.87. The algorithm also consistently outperforms both benchmark approaches by a significant margin. For example, in the largest experiment with approximately 50K users and 10K songs, the slice learning algorithm has an average AUC of 0.95, while the naive and matrix algorithms have AUCs of 0.82 and 0.84, respectively. Part of this strong performance comes from the fact that the slice learning algorithm is able to estimate more complex models, which is demonstrated by the consistently higher rank values.

The equivalent tables for recovering the Listen and Download slices can be found in Section F in the Appendix. To summarize those results succinctly, the slice learning algorithm again consistently outperforms both benchmarks across all experiments. The margin of improvement is lower, but that is to be expected as those slices are less sparse than the Collect slice, and so the Collect slice does not offer as much side-information.

7. Conclusion

This paper introduced a new approach to modeling and learning with side information. Motivated by settings in e-commerce, where data is sparse but multiple interactions occur, we formulated the problem of recovering slices of a three-dimensional tensor from a noisy observation of the tensor. We proposed the slice learning algorithm, a computationally efficient algorithm that scales to enormous size by leveraging sparsity. Theoretically, we showed that the algorithm achieves the minimax lower bound for recovery with sufficiently many slices; this guarantee is the best known guarantee for noisy recovery of tensors and the first guarantee of its kind for recovery of specific slices. Synthetic experiments further supported the fact that this algorithm outperforms existing convex methods in both efficiency and accuracy. Experiments on real-world data from the music streaming service `Xiami.com` demonstrated the scalability of the approach and provided real empirical evidence that having side information is advantageous and that our approach utilizes side information effectively.

Our work points to a number of interesting, exciting directions for future work:

1. Different forms of side information: side information may come in the form of data specific to the row space or the column space. For example, retailers have demographic information on their customers, and basic information about their products. As we discussed in §4, the slice learning algorithm can incorporate this kind of side information. A deeper analysis of this procedure is an important next step.
2. Trimming: our algorithm performs best under the balanced noise assumption. We defined precisely how to measure the level of unbalance, and quantified the penalty of unbalance. When the noise is known to be significantly unbalanced, it may be possible to weight the rows and columns of the tensor in such a way that induces balanced noise. This weighted tensor can be estimated and then unweighted to recover the original tensor. Such 'trimming' procedures need to be analyzed.
3. Higher-dimensional tensors: there are many applications for tensors of dimension greater than three. For example, retailers might view their sales transactions over time, producing a three-dimensional tensor where time is the third dimension. Time-series data for multiple interactions may then be viewed as a four-dimensional tensor. There may be ways that the slice learning algorithm can be generalized to higher dimensions.

References

Acar E, Çamtepe SA, Krishnamoorthy MS, Yener B (2005) Modeling and multiway analysis of chatroom tensors. *International Conference on Intelligence and Security Informatics*, 256–268 (Springer).

Achlioptas D, McSherry F (2007) Fast computation of low-rank matrix approximations. *Journal of the ACM (JACM)* 54(2):9.

Adomavicius G, Tuzhilin A (2005) Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering* 17(6):734–749.

Ansari A, Essegaier S, Kohli R (2000) Internet recommendation systems. *Journal of Marketing research* 37(3):363–375.

Ansari A, Mela CF (2003) E-customization. *Journal of marketing research* 40(2):131–145.

Bahadori MT, Yu QR, Liu Y (2014) Fast multivariate spatio-temporal analysis via low rank tensor learning. *Advances in Neural Information Processing Systems*, 3491–3499.

Bernhardsson E (2014) Music discovery at spotify. *MLConf NYC*.

Berry MW, Dumais ST, O'Brien GW (1995) Using linear algebra for intelligent information retrieval. *SIAM review* 37(4):573–595.

Besbes O, Gur Y, Zeevi A (2015) Optimization in online content recommendation services: Beyond click-through rates. *Manufacturing & Service Operations Management* 18(1):15–33.

Bodapati AV (2008) Recommendation systems with purchase data. *Journal of marketing research* 45(1):77–93.

Candès EJ, Plan Y (2011) Tight oracle inequalities for low-rank matrix recovery from a minimal number of noisy random measurements. *Information Theory, IEEE Transactions on* 57(4):2342–2359.

Candès EJ, Tao T (2010) The power of convex relaxation: Near-optimal matrix completion. *Information Theory, IEEE Transactions on* 56(5):2053–2080.

Carroll JD, Chang JJ (1970) Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika* 35(3):283–319.

Cattell RB (1966) The scree test for the number of factors. *Multivariate behavioral research* 1(2):245–276.

Chatterjee S (2014) Matrix estimation by universal singular value thresholding. *The Annals of Statistics* 43(1):177–214.

Chiang KY, Hsieh CJ, Dhillon IS (2015) Matrix completion with noisy side information. *Advances in Neural Information Processing Systems*, 3429–3437.

Chung TS, Rust RT, Wedel M (2009) My mobile music: An adaptive personalization system for digital audio players. *Marketing Science* 28(1):52–68.

Demirezen EM, Kumar S (2016) Optimization of recommender systems based on inventory. *Production and Operations Management* .

Export-x (2015) How many products does amazon sell? `https://export-x.com/2015/12/11/how-many-products-does-amazon-sell-2015/`.

FastCompany (2015) Amazon sold 5 billion items in 2014. `http://www.fastcompany.com/3040445/fast-feed/amazon-sold-5-billion-items-in-2014`.

Fleder D, Hosanagar K (2009) Blockbuster culture's next rise or fall: The impact of recommender systems on sales diversity. *Management science* 55(5):697–712.

Gandy S, Recht B, Yamada I (2011) Tensor completion and low-n-rank tensor recovery via convex optimization. *Inverse Problems* 27(2):025010.

Garfinkel R, Gopal R, Pathak B, Yin F (2008) Shopbot 2.0: Integrating recommendations and promotions with comparison shopping. *Decision Support Systems* 46(1):61–69.

Gavish M, Donoho DL (2014) The optimal hard threshold for singular values is. *IEEE Transactions on Information Theory* 60(8):5040–5053.

Ghose A, Ipeirotis PG, Li B (2012) Designing ranking systems for hotels on travel search engines by mining user-generated and crowdsourced content. *Marketing Science* 31(3):493–520.

Goldberg D, Nichols D, Oki BM, Terry D (1992) Using collaborative filtering to weave an information tapestry. *Communications of the ACM* 35(12):61–70.

Gross D (2011) Recovering low-rank matrices from few coefficients in any basis. *Information Theory, IEEE Transactions on* 57(3):1548–1566.

Grover R, Srinivasan V (1987) A simultaneous approach to market segmentation and market structuring. *Journal of Marketing Research* 139–153.

Henrion R (1994) N-way principal component analysis theory, algorithms and applications. *Chemometrics and intelligent laboratory systems* 25(1):1–23.

Hoff PD (2012) Model averaging and dimension selection for the singular value decomposition. *Journal of the American Statistical Association* .

Hosanagar K, Krishnan R, Ma L (2008) Recomended for you: The impact of profit incentives on the relevance of online recommendations. *ICIS 2008 Proceedings* 31.

Huang B, Mu C, Goldfarb D, Wright J (2015) Provable models for robust low-rank tensor completion. *Pacific Journal of Optimization* 11:339–364.

Jacobs BJ, Donkers B, Fok D (2016) Model-based purchase predictions for large assortments. *Marketing Science* 35(3):389–404.

Jain P, Dhillon IS (2013) Provable inductive matrix completion. *arXiv preprint arXiv:1306.0626* .

Jain P, Oh S (2014) Provable tensor factorization with missing data. *Advances in Neural Information Processing Systems*, 1431–1439.

Keshavan RH, Montanari A, Oh S (2010) Matrix completion from a few entries. *Information Theory, IEEE Transactions on* 56(6):2980–2998.

Kleinberg JM (1999) Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)* 46(5):604–632.

Kolda TG, Bader BW (2009) Tensor decompositions and applications. *SIAM review* 51(3):455–500.

Koltchinskii V, Lounici K, Tsybakov AB, et al. (2011) Nuclear-norm penalization and optimal rates for noisy low-rank matrix completion. *The Annals of Statistics* 39(5):2302–2329.

Koren Y, Bell R, Volinsky C, et al. (2009) Matrix factorization techniques for recommender systems. *Computer* 42(8):30–37.

Larsen RM (1998) Lanczos bidiagonalization with partial reorthogonalization. *DAIMI Report Series* 27(537).

Liben-Nowell D, Kleinberg J (2007) The link-prediction problem for social networks. *Journal of the American society for information science and technology* 58(7):1019–1031.

Linden G, Smith B, York J (2003) Amazon. com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 7(1):76–80.

Liu J, Musialski P, Wonka P, Ye J (2013) Tensor completion for estimating missing values in visual data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 35(1):208–220.

Liu Y, Shang F, Fan W, Cheng J, Cheng H (2014) Generalized higher-order orthogonal iteration for tensor decomposition and completion. *Advances in Neural Information Processing Systems*, 1763–1771.

Marčenko VA, Pastur LA (1967) Distribution of eigenvalues for some sets of random matrices. *Mathematics of the USSR-Sbornik* 1(4):457.

Mørup M (2011) Applications of tensor (multiway array) factorizations and decompositions in data mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1(1):24–40.

Mu C, Huang B, Wright J, Goldfarb D (2013) Square deal: Lower bounds and improved relaxations for tensor recovery. *arXiv preprint arXiv:1307.5870* .

Naik P, Wedel M, Bacon L, Bodapati A, Bradlow E, Kamakura W, Kreulen J, Lenk P, Madigan DM, Montgomery A (2008) Challenges and opportunities in high-dimensional choice data analyses. *Marketing Letters* 19(3-4):201–213.

Oliver NM, Rosario B, Pentland AP (2000) A bayesian computer vision system for modeling human interactions. *IEEE transactions on pattern analysis and machine intelligence* 22(8):831–843.

Owen AB, Perry PO (2009) Bi-cross-validation of the svd and the nonnegative matrix factorization. *The annals of applied statistics* 564–594.

Papadimitriou CH, Tamaki H, Raghavan P, Vempala S (1998) Latent semantic indexing: A probabilistic analysis. *Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, 159–168 (ACM).

Recht B (2011) A simpler approach to matrix completion. *The Journal of Machine Learning Research* 12:3413–3430.

Romera-Paredes B, Aung H, Bianchi-Berthouze N, Pontil M (2013) Multilinear multitask learning. *Proceedings of the 30th International Conference on Machine Learning*, 1444–1452.

Romera-Paredes B, Pontil M (2013) A new convex relaxation for tensor completion. *Advances in Neural Information Processing Systems*, 2967–2975.

Shi Y, Larson M, Hanjalic A (2014) Collaborative filtering beyond the user-item matrix: A survey of the state of the art and future challenges. *ACM Computing Surveys (CSUR)* 47(1):3.

Signoretto M, Dinh QT, De Lathauwer L, Suykens JA (2014) Learning with tensors: a framework based on convex optimization and spectral regularization. *Machine Learning* 94(3):303–351.

Singh AP, Gordon GJ (2008) Relational learning via collective matrix factorization. *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, 650–658 (ACM).

Sirovich L, Kirby M (1987) Low-dimensional procedure for the characterization of human faces. *Josa a* 4(3):519–524.

So AMC, Ye Y (2007) Theory of semidefinite programming for sensor network localization. *Mathematical Programming* 109(2-3):367–384.

Soni A, Jain S, Haupt J, Gonella S (2016) Noisy matrix completion under sparse factor models. *IEEE Transactions on Information Theory* 62(6):3636–3661.

Statista (2016) Annual number of worldwide active amazon customer accounts from 1997 to 2015. `http://www.statista.com/statistics/237810/number-of-active-amazon-customer-accounts-worldwide/`.

Sun JT, Zeng HJ, Liu H, Lu Y, Chen Z (2005) Cubesvd: a novel approach to personalized web search. *Proceedings of the 14th international conference on World Wide Web*, 382–390 (ACM).

Suzuki T (2015) Convergence rate of bayesian tensor estimator and its minimax optimality. *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 1273–1282.

Tomioka R, Suzuki T (2013) Convex tensor decomposition via structured schatten norm regularization. *Advances in neural information processing systems*, 1331–1339.

Tomioka R, Suzuki T, Hayashi K, Kashima H (2011) Statistical performance of convex tensor decomposition. *Advances in Neural Information Processing Systems*, 972–980.

Troyanskaya O, Cantor M, Sherlock G, Brown P, Hastie T, Tibshirani R, Botstein D, Altman RB (2001) Missing value estimation methods for dna microarrays. *Bioinformatics* 17(6):520–525.

Vasilescu MAO, Terzopoulos D (2002) Multilinear analysis of image ensembles: Tensorfaces. *European Conference on Computer Vision*, 447–460 (Springer).

Wold S (1978) Cross-validatory estimation of the number of components in factor and principal components models. *Technometrics* 20(4):397–405.

Xu M, Jin R, Zhou ZH (2013) Speedup matrix completion with side information: Application to multi-label learning. *Advances in Neural Information Processing Systems*, 2301–2309.

Yin YQ, Bai ZD, Krishnaiah PR (1988) On the limit of the largest eigenvalue of the large dimensional sample covariance matrix. *Probability theory and related fields* 78(4):509–521.

Yuan M, Zhang CH (2015) On tensor completion via nuclear norm minimization. *Foundations of Computational Mathematics* 1–38.

Zheng Q, Tomioka R (2015) Interpolating convex and non-convex tensor decompositions via the subspace norm. *arXiv preprint arXiv:1503.05479* .

Zhou H, Li L, Zhu H (2013) Tensor regression with applications in neuroimaging data analysis. *Journal of the American Statistical Association* 108(502):540–552.

A. Survey of works employing user-item interaction data

Table 2 summarizes the different forms of user-item interaction data that have previously been studied.

Activity	Interaction	Representative articles
Direct Feedback	Numerical ratings	Breese et al. (1998)
		Ansari et al. (2000)
		Herlocker et al. (2002)
	Text reviews	Hu and Liu (2004)
		Das and Chen (2007)
		Archak et al. (2011)
Purchases	In-store sales transactions	Fader and Hardie (1996)
		Chong et al. (2001)
		Chintagunta et al. (2005)
		Chintagunta and Dube (2005)
	Online sales transactions	Bodapati (2008)
		Moon and Russell (2008)
In-site actions	Add to cart, search	Wu and Rangaswamy (2003)
	Product view	Moe (2006)
	Clicks on email links sent by site	Ansari and Mela (2003)
	Product customization	Sismeiro and Bucklin (2004)
	Browsing	Bucklin and Sismeiro (2003)
		Montgomery et al. (2004)
		Besbes et al. (2015)
Outside online	Twitter, Google, Wiki, IMDB	Liu et al. (2016)
	Blogs	Gopinath et al. (2013)
	Browsing	Trusov et al. (2016)
	Tagging	Ghose et al. (2012)
Physical actions	Try-on, facial expressions	Lu et al. (2016)
	Movement, direction faced, gaze	Hui et al. (2013)
Usage	Song listening time	Chung et al. (2009)

Table 2: List of user-item interactions.

A classic user-item interaction is users' direct feedback in the form of numerical ratings. Numerical ratings have been the traditional subject of study for recommender system researchers (Breese et al. (1998), Herlocker et al. (2002) and Ansari et al. (2000)). The prototypical example of this is the Netflix Prize competition (Bennett and Lanning (2007)), where the data consisted of users' movie grades on a scale from 1 to 5. Advances in text mining techniques have also allowed analysis of users' text reviews; see Hu and Liu (2004) and Archak et al. (2011).

Bodapati (2008) points out that 'in real-world systems, explicit self-reports of ratings are not observed as frequently as behavioral data in the form of purchases.' Along these lines, another well-studied type of interaction is purchases. Sales transaction data at the customer-product granularity has existed for some time now in many forms, e.g. customer transactions have been tracked in brick-and-mortar retail with the use of scanning devices and loyalty programs. There have been studies dealing with this type of data; for

example, Chong et al. (2001), Fader and Hardie (1996), Chintagunta et al. (2005), and Chintagunta and Dube (2005) all analyze purchase data among households at brick-and-mortar stores. Online retail has made sales transactions even easier to record. Both Bodapati (2008) and Moon and Russell (2008) use online purchase data to make product recommendations.

Beyond purchases, advanced tracking software allows for all types of online behavior to be recorded. Within a business' website, a variety of user-item interactions may be recorded, including adding an item to a virtual shopping cart (Wu and Rangaswamy (2003)), viewing an item's page (Moe (2006)), and clicking on personalized email links (Ansari and Mela (2003)). These interactions also extend beyond a business' own website into general online behavior, including user-generated content such as blogs (Gopinath et al. (2013)) and social media (Liu et al. (2016)).

Increasingly sophisticated technology has even allowed for collection and analysis of new kinds of data in the brick-and-mortar setting. Data is generated for example through cell phone tracking and in-store video: Macy's encourages shoppers to scan products through their mobile app (MobileCommerceDaily (2013)), and video can be used to record when customers slow down and look at a product (Hui et al. (2013)); Lu et al. (2016) were even able to record and analyze customers' facial expressions while trying on clothing items.

B. Comparison of slice rank to existing tensor ranks

There are many definitions of rank for tensors that have already been studied. The two most common, which tensor recovery has focused on, are referred to here as CP rank and Tucker rank. We review the canonical definitions of these ranks here. See Kolda and Bader (2009) for a more thorough treatment of these concepts.

CP rank The CP rank of a tensor relates to its orthogonal decompositions. A rank-one tensor is any tensor $M \in \mathbb{R}^{m \times m \times n}$ that is the outer product of three vectors, i.e. $M = u \otimes v \otimes w$ for some $u \in \mathbb{R}^m$, $v \in \mathbb{R}^m$, and $w \in \mathbb{R}^n$, or equivalently, $M^k_{i,j} = u_i v_j w_k$. For any tensor M, we denote its CP rank as $\text{CP}(M)$, which is the minimum number r such that M can be expressed as the sum of r rank-one tensors.

Tucker rank The Tucker rank of a tensor M, denoted $\text{Tucker}(M)$, is the vector (r_1, r_2, r_3), where r_d is the rank of its mode-d unfolding. This relates to its higher order singular value decomposition: given a tensor of Tucker rank (r_1, r_2, r_3), there exist vectors $u^1, \ldots, u^{r_1} \in \mathbb{R}^m$, $v^1, \ldots, v^{r_2} \in \mathbb{R}^m$, and $w^1, \ldots, w^{r_3} \in \mathbb{R}^n$, and a smaller tensor $S \in \mathbb{R}^{r_1 \times r_2 \times r_3}$, such that $M = \sum_{\ell_1=1}^{r_1} \sum_{\ell_2=1}^{r_2} \sum_{\ell_3=1}^{r_3} S^{\ell_3}_{\ell_1,\ell_2} u^{\ell_1} \otimes v^{\ell_2} \otimes w^{\ell_3}$.

Proposition 1 establishes that the slice rank is a less restrictive measure of complexity than either of these two rank definitions. Recall that $\mathcal{CP}(r)$ is the set of tensors with CP rank at most r, $\text{Tucker}(r, r, l)$ is the set of tensors whose Tucker rank is component-wise at most (r, r, l), and $\text{Slice}(r)$ is the set of tensors whose slice rank is at most r.

Proof of Proposition 1. We first prove that $\mathcal{CP}(r) \subseteq \text{Slice}(r)$. Suppose $M \in \mathcal{CP}(r)$. By definition there exist vectors $u^1, \ldots, u^r \in \mathbb{R}^m$, $v^1, \ldots, v^r \in \mathbb{R}^m$, and $w^1, \ldots, w^r \in \mathbb{R}^n$, such that each entry of M can be expressed as

$$M^k_{i,j} = \sum_{\ell=1}^{r} u^\ell_i v^\ell_j w^\ell_k.$$

Let U and V be the matrices with columns $u^1, \ldots, u^r$ and $v^1, \ldots, v^r$, respectively. Then we can equivalently

write the above expression in matrix form for slices as

$$M^k = \sum_{\ell=1}^{r} w_k^\ell \left(u^\ell v^{\ell\top}\right) = US^kV^\top,$$

where S^k is the diagonal matrix whose diagonal elements are $w_k^1, \ldots, w_k^r$. This equivalent expression for the slices of M reveals that the slice rank of M is at most r, and so $M \in \text{Slice}(r)$.

Now we prove that $\text{Tucker}(r,r,l) \subseteq \text{Slice}(r)$. Suppose $M \in \text{Tucker}(r,r,l)$. By definition there exist vectors $u^1, \ldots, u^r \in \mathbb{R}^m$, $v^1, \ldots, v^r \in \mathbb{R}^m$, and $w^1, \ldots, w^l \in \mathbb{R}^n$, and a tensor $S \in \mathbb{R}^{r\times r\times l}$, such that $M = \sum_{\ell_1=1}^r \sum_{\ell_2=1}^r \sum_{\ell_3=1}^l S_{\ell_1,\ell_2}^{\ell_3} u^{\ell_1} \otimes v^{\ell_2} \otimes w^{\ell_3}$. Equivalently, each entry of M can be expressed as

$$M_{i,j}^k = \sum_{\ell_1=1}^{r}\sum_{\ell_2=1}^{r}\sum_{\ell_3=1}^{l} S_{\ell_1,\ell_2}^{\ell_3} u_i^{\ell_1} v_j^{\ell_2} w_k^{\ell_3}.$$

Let U and V be the matrices with columns $u^1, \ldots, u^r$ and $v^1, \ldots, v^r$, respectively. Then we can equivalently write the above expression in matrix form for slices as

$$M^k = \sum_{\ell_3=1}^{l} w_k^{\ell_3} \sum_{\ell_1=1}^{r}\sum_{\ell_2=1}^{r} S_{\ell_1,\ell_2}^{\ell_3} \left(u^{\ell_1} v^{\ell_2\top}\right) = \sum_{\ell_3=1}^{l} w_k^{\ell_3} (US^{\ell_3}V^\top) = U\left(\sum_{\ell_3=1}^{l} w_k^{\ell_3} S^{\ell_3}\right)V^\top.$$

Once again, this equivalent expression for the slices of M reveals that the slice rank of M is at most r, and so $M \in \text{Slice}(r)$. ■

C. Proof of Proposition 2

Our proof follows a standard Bayesian argument for minimax lower bounds; for example, see the proof of Theorem 1.2 in Chatterjee (2014). We will separately show that $\text{MSE}(\hat{M}) \geq C(r^2/m^2)$ and $\text{MSE}(\hat{M}) \geq C(r/mn)$. We first give a detailed proof that $\text{MSE}(\hat{M}) \geq C(r^2/m^2)$. For each ground truth slice M^k, let the elements sitting in the first r rows and first r columns be drawn independently from a uniform distribution, and the remaining elements set equal to 0:

$$M_{ij}^k \sim \begin{cases} \text{Uniform}[0,1] & \text{if } i \leq r \text{ and } j \leq r \\ 0 & \text{if } i > r \text{ or } j > r \end{cases}. \tag{7}$$

Note that all slices of M share the same column and row spaces, both with dimension at most r. Finally, conditional on M, each entry $X_{i,j}^k$ of X is drawn from the following two point distribution:

$$X_{ij}^k \sim \text{Ber}(M_{ij}^k). \tag{8}$$

Then for each $i \leq r$ and $j \leq r$, we have

$$\begin{aligned}
\mathsf{E}\left[\mathrm{Var}\left(M_{ij}^k \middle| X\right)\right] &= \mathrm{Var}\left(M_{ij}^k\right) - \mathrm{Var}\left(\mathsf{E}\left[M_{ij}^k \middle| X\right]\right) \\
&= \mathrm{Var}\left(M_{ij}^k\right) - \mathrm{Var}\left(\mathsf{E}\left[M_{ij}^k \middle| X_{ij}^k\right]\right) \\
&= \mathrm{Var}\left(M_{ij}^k\right) - \mathrm{Var}\left(\frac{1+X_{ij}^k}{3}\right) \\
&= \frac{1}{12} - \frac{1}{36} = \frac{1}{18} \qquad (9)
\end{aligned}$$

The first equality is the law of total variance. For the second equality, observe that M_{ij}^k is independent of all entries of X except for its corresponding entry X_{ij}^k. The third equality comes from the fact that, having defined M_{ij}^k to be distributed as Uniform$[0,1]$ (or equivalently Beta$(1,1)$), its distribution conditional on X_{ij}^k is Beta$(1+X_{ij}^k, 2-X_{ij}^k)$.

Then for any estimator $\hat{M}$, the definition of variance implies that

$$\mathsf{E}\left[\left(\hat{M}_{ij}^k - M_{ij}^k\right)^2 \middle| X\right] \geq \mathrm{Var}\left(M_{ij}^k \middle| X\right).$$

Taking expectations of both sides, and applying (9), we have

$$\mathsf{E}\left[\left(\hat{M}_{ij}^k - M_{ij}^k\right)^2\right] \geq \frac{1}{18}.$$

The proof concludes by summing both sides over all entries of M in the first r rows and first r columns (i.e. nr^2 entries in total) and dividing by $m^2 n$.

A nearly identical argument shows that $\mathrm{MSE}(\hat{M}) \geq C(r/mn)$. For the first slice M^1, let the elements in the first r rows be drawn independently from a uniform distribution, and the remaining elements set equal to 0:

$$M_{ij}^1 \sim \begin{cases} \text{Uniform}[0,1] & \text{if } i \leq r \\ 0 & \text{if } i > r \end{cases}. \qquad (10)$$

Set the entries of the remaining slices equal to the corresponding entries in the first slice, i.e. $M_{ij}^k = M_{ij}^1$ for all k. Once again, conditional on M, each entry X_{ij}^k is drawn independently from the distribution in (8), so while the slices of M are copies of each other, the slices of X are not. Then for each $i \leq r$, we have

$$\begin{aligned}
\mathsf{E}\left[\mathrm{Var}\left(M_{ij}^k \middle| X\right)\right] &= \mathrm{Var}\left(M_{ij}^k\right) - \mathrm{Var}\left(\mathsf{E}\left[M_{ij}^k \middle| X\right]\right) \\
&= \mathrm{Var}\left(M_{ij}^k\right) - \mathrm{Var}\left(\mathsf{E}\left[M_{ij}^k \middle| X_{ij}^1, \ldots X_{ij}^n\right]\right) \\
&= \mathrm{Var}\left(M_{ij}^k\right) - \mathrm{Var}\left(\frac{1+X_{ij}^1+\cdots+X_{ij}^n}{n+2}\right) \\
&= \frac{1}{12} - \frac{n}{12(n+2)} = \frac{1}{6(n+2)} \qquad (11)
\end{aligned}$$

Again, the first equality is the law of total variance, and for the second equality, observe that M_{ij}^k is independent of all entries of X except for the $(i,j)^{\text{th}}$ entry of each slice. For the third equality, the distribution of M_{ij}^k conditional on $X_{ij}^1, \ldots, X_{ij}^n$ is Beta$(1+X_{ij}^1+\cdots+X_{ij}^n, n+1-(X_{ij}^1+\cdots+X_{ij}^n))$.

Therefore, for any estimator $\hat{M}$ we have

$$\mathsf{E}\left[\left(\hat{M}_{ij}^k - M_{ij}^k\right)^2\right] \geq \frac{1}{6(n+2)}.$$

The proof concludes by summing both sides over all entries of M in the first r rows (i.e. nmr entries in total) and dividing by $m^2 n$. ■

D. Proofs of Theorems 1 and 2 and Corollary 1

Before proceeding with the proofs, it will be convenient to review and introduce some additional notation. For $n \in \mathbb{Z}^+$, $[n]$ denotes the set $\{1, \ldots, n\}$. If X is a matrix, then $\|X\|_2$, $\|X\|_F$, and $\|X\|_*$ are respectively the operator, frobenius, and nuclear norms of X. $\sigma_i(X)$ is the i^{th} largest singular value of X. For a matrix $U \in \mathbb{R}^{m \times r}$ with orthonormal columns, we will refer to U as a *matrix* and *subspace* interchangeably, where the subspace is the space in $\mathbb{R}^m$ spanned by the columns of U; $P_U = UU^\top$ is the projection operator onto the subspace U. We use $d(U, \hat{U}) = \left\|P_U - P_{\hat{U}}\right\|_F$ as a metric for subspaces.

It will suffice to prove Theorem 2; Theorem 1 is just the special case when $\delta = 0$. The proof of Theorem 2 involves two steps, corresponding to the two stages of the algorithm: learning subspaces and projection. In the first step, we show that we are able to closely estimate the column and row spaces, and in the second step, we show that if our estimates of the 'true' column and row spaces are close, then our estimate of each slice is close.

D.1. Step 1: Column and Row Space Estimation

To estimate the column space (and similarly the row space), we take the top column singular vectors of $X_{(1)} = M_{(1)} + \epsilon_{(1)}$, so it is important to understand the extent to which $\epsilon_{(1)}$ changes the singular vectors of $M_{(1)}$. Lemma 1 bounds the error of this step. The first result in Lemma 1 is an upper bound on $\mathsf{E}\left[d(U, \hat{U})^2\right]$, which is the expected error of our subspace estimate. Because of the decomposition we make later on, we also need to bound $\mathsf{E}\left[\left\|\epsilon^k\right\|_F^2 d(U, \hat{U})^2\right]$ for any slice of the noise tensor ϵ^k, which is complicated by the fact that ϵ^k and $d(U, \hat{U})$ are not independent. The second result in Lemma 1 controls this term.

Lemma 1. *Let $M \in \mathbb{R}^{m \times mn}$ be a matrix with column space $U \in \mathbb{R}^{m \times r}$. Suppose $\epsilon \in \mathbb{R}^{m \times mn}$ is a random matrix with independent elements, where each element ϵ_{ij} is mean-zero, and $\mathsf{E}[\epsilon_{ij}^2]$, $\mathsf{E}[\epsilon_{ij}^4]$, and $\mathsf{E}[\epsilon_{ij}^6]$ are respectively bounded by K_2, K_4, and K_6. Let $X = M + \epsilon$, and let $\hat{U} \in \mathbb{R}^{m \times r}$ be the column singular vectors of X corresponding to its largest r singular values. Then taking expectation over ϵ, we have*

$$\mathsf{E}\left[d(U, \hat{U})^2\right] \leq 24 \frac{4K_2 m \|M\|_F^2 + K_4 m^2 n + K_2^2 m^3 n + \min_\rho \left\|\mathsf{E}[\epsilon\epsilon^\top] - \rho I_m\right\|_F^2}{\sigma_r^4(M)}, \quad \textit{and}$$

$$\begin{aligned}\mathsf{E}\left[\left\|\epsilon^1\right\|_F^2 d(U, \hat{U})^2\right] \leq 24m^2 &\frac{4K_2^2 m \|M\|_F^2 + 3K_4 K_2 m^2 n + K_2^3 m^3 n + K_2 \min_\rho \left\|\mathsf{E}[\epsilon\epsilon^\top] - \rho I_m\right\|_F^2}{\sigma_r^4(M)} \\ &+ 24m^2 \frac{4K_4 \left\|M^1\right\|_F^2 / m + K_6}{\sigma_r^4(M)},\end{aligned}$$

where M^1 and ϵ^1 are the $m \times m$ submatrices consisting of the first m columns of M and ϵ, respectively.

Note that M^1 and ϵ^1 in the statement of Lemma 1 can in fact be any $m \times m$ submatrices of M and ϵ; they are taken to be the first M columns to save on notation, and without loss of generality. The proof of Lemma 1 relies on the Davis-Kahan Theorem (Davis and Kahan (1970)), via a recent extension by Yu et al. (2015), which we reproduce as Lemma 2. Note that Lemma 2 is a statement about symmetric matrices, which we adapt to our setting where the matrices are not symmetric or even square; Yu et al. (2015) also show a version of Lemma 2 for rectangular matrices that is a similar modification to Wedin's Theorem (Wedin (1972)), but applying that directly would not yield as strong a bound as Lemma 1. However, this stronger bound requires the noise to be balanced.

Lemma 2 (Davis-Kahan Variant; Yu et al. (2015), Theorem 2)**.** *Suppose S and $\hat{S}$ are symmetric matrices, and let U and $\hat{U}$ be the eigenvectors corresponding to the r largest eigenvalues of S and $\hat{S}$, respectively. Let $\lambda_r(S)$ and $\lambda_{r+1}(S)$ be the r^{th} and $r+1^{th}$ largest eigenvalues of S. Then assuming $\lambda_r(S) \neq \lambda_{r+1}(S)$, we have*

$$d(U, \hat{U}) \leq \frac{2\sqrt{2}\left\|S - \hat{S}\right\|_F}{\lambda_r(S) - \lambda_{r+1}(S)}.$$

Proof of Lemma 1. First note that the column singular vectors of M and X are identical to the eigenvectors of $MM^\top$ and $XX^\top$, respectively, and further, the eigenvectors of $XX^\top - \rho I_m$ are the same for any $\rho \in \mathbb{R}$. Thus, Lemma 2 can be applied directly with $S = MM^\top$, and $\hat{S} = XX^\top - \rho I_m$ for any $\rho \in \mathbb{R}$, and $\lambda_r(MM^\top) - \lambda_{r+1}(MM^\top) = \sigma_r(M)^2 - \sigma_{r+1}(M)^2 = \sigma_r(M)^2$:

$$d(U, \hat{U})^2 \leq \frac{8 \min_\rho \left\|MM^\top - (XX^\top - \rho I_m)\right\|_F^2}{\sigma_r^4(M)}. \tag{12}$$

To upper bound the numerator, we make the following decomposition:

$$\begin{aligned}
\min_\rho \left\|MM^\top - (XX^\top - \rho I_m)\right\|_F &\leq 2\left\|M\epsilon^\top\right\|_F + \min_\rho \left\|\epsilon\epsilon^\top - \rho I_m\right\|_F \\
&\leq 2\left\|M\epsilon^\top\right\|_F + \left\|\epsilon\epsilon^\top - \mathsf{E}[\epsilon\epsilon^\top]\right\|_F + \min_\rho \left\|\mathsf{E}[\epsilon\epsilon^\top] - \rho I_m\right\|_F,
\end{aligned}$$

and since $(a+b+c)^2 \leq 3(a^2+b^2+c^2)$ for any $a, b, c \in \mathbb{R}$, we have

$$\min_\rho \left\|MM^\top - (XX^\top - \rho I_m)\right\|_F^2 \leq 3\left(4\left\|M\epsilon^\top\right\|_F^2 + \left\|\epsilon\epsilon^\top - \mathsf{E}[\epsilon\epsilon^\top]\right\|_F^2 + \min_\rho \left\|\mathsf{E}[\epsilon\epsilon^\top] - \rho I_m\right\|_F^2\right).$$

We have decomposed the numerator of (12) into three terms. The last term is a deterministic quantity. The proof concludes by bounding the expectation of the first two terms. All of the following calculations proceed in the same manner: the equalities come from rewriting expressions in expanded form and setting any summands with a lone $\mathsf{E}[\epsilon_{ij}]$ to zero, and the inequality applies the assumptions on the moments of the noise terms.

$$\begin{aligned}
\mathsf{E}\left[\left\|M\epsilon^\top\right\|_F^2\right] &= \sum_{i_1 \in [m], i_2 \in [m]} \mathsf{E}\left[\left(\sum_{j \in [mn]} M_{i_1 j}\epsilon_{i_2 j}\right)^2\right] \\
&= \sum_{i_1 \in [m], i_2 \in [m]} \sum_{j \in [mn]} M_{i_1 j}^2 \mathsf{E}\left[\epsilon_{i_2 j}^2\right] \leq K_2 m \left\|M\right\|_F^2
\end{aligned} \tag{13}$$

$$
\begin{aligned}
(14) \qquad \mathsf{E}\left[\left\|\epsilon^1\right\|_F^2 \left\|M\epsilon^\top\right\|_F^2\right] &= \mathsf{E}\left[\left\|\epsilon^1\right\|_F^2 \sum_{i_1,i_2\in[m]} \left(\sum_{j\in[mn]} M_{i_1 j}\epsilon_{i_2 j}\right)^2\right] \\
&= \sum_{i_1,i_2\in[m]} \sum_{j\in[mn]} M_{i_1 j}^2 \mathsf{E}\left[\epsilon_{i_2 j}^2 \left\|\epsilon^1\right\|_F^2\right] \\
&= \sum_{i_1,i_2\in[m]} \sum_{j\in[m]} M_{i_1 j}^2 \mathsf{E}\left[\epsilon_{i_2 j}^4\right] + \sum_{i_1,i_2,i_3\in[m]} \sum_{\substack{j_1\in[mn],j_2\in[m] \\ (i_2,j_1)\neq(i_3,j_2)}} M_{i_1 j_1}^2 \mathsf{E}\left[\epsilon_{i_2 j_1}^2\right] \mathsf{E}\left[\epsilon_{i_3 j_2}^2\right] \\
&\leq K_4 m \left\|M^1\right\|_F^2 + K_2^2 m^3 \left\|M\right\|_F^2
\end{aligned}
$$

$$
\begin{aligned}
(15) \qquad \mathsf{E}\left[\left\|\epsilon\epsilon^\top - \mathsf{E}[\epsilon\epsilon^\top]\right\|_F^2\right] &= \sum_{i\in[m]} \mathrm{Var}\left(\sum_{j\in[mn]} \epsilon_{ij}^2\right) + \sum_{\substack{i_1\in[m],i_2\in[m] \\ i_1\neq i_2}} \mathsf{E}\left[\left(\sum_{j\in[mn]} \epsilon_{i_1 j}\epsilon_{i_2 j}\right)^2\right] \\
&= \sum_{i\in[m]} \sum_{j\in[mn]} \mathrm{Var}\left[\epsilon_{ij}^2\right] + \sum_{\substack{i_1\in[m],i_2\in[m] \\ i_1\neq i_2}} \sum_{j\in[mn]} \mathsf{E}\left[\epsilon_{i_1 j}^2\right] \mathsf{E}\left[\epsilon_{i_2 j}^2\right] \\
&\leq K_4 m^2 n + K_2^2 m^3 n
\end{aligned}
$$

(16)

$$
\begin{aligned}
\mathsf{E}\left[\left\|\epsilon^1\right\|_F^2 \left\|\epsilon\epsilon^\top - \mathsf{E}[\epsilon\epsilon^\top]\right\|_F^2\right] &= \mathsf{E}\left[\left\|\epsilon^1\right\|_F^2 \sum_{i\in[m]} \left(\sum_{j\in[mn]} \epsilon_{ij}^2 - \mathsf{E}\left[\epsilon_{ij}^2\right]\right)^2 + \left\|\epsilon^1\right\|_F^2 \sum_{\substack{i_1,i_2\in[m] \\ i_1\neq i_2}} \left(\sum_{j\in[mn]} \epsilon_{i_1 j}\epsilon_{i_2 j}\right)^2\right] \\
&= \sum_{i\in[m]} \sum_{j\in[mn]} \mathsf{E}\left[\left\|\epsilon^1\right\|_F^2 \left(\epsilon_{ij}^2 - \mathsf{E}\left[\epsilon_{ij}^2\right]\right)^2\right] + \sum_{\substack{i_1,i_2\in[m] \\ i_1\neq i_2}} \sum_{j\in[mn]} \mathsf{E}\left[\left\|\epsilon^1\right\|_F^2 \epsilon_{i_1 j}^2 \epsilon_{i_2 j}^2\right] \\
&= \sum_{i\in[m]} \sum_{j\in[m]} \mathsf{E}\left[\epsilon_{ij}^2 \left(\epsilon_{ij}^2 - \mathsf{E}\left[\epsilon_{ij}^2\right]\right)^2\right] + \sum_{i_1,i_2\in[m]} \sum_{\substack{j_1\in[mn],j_2\in[m] \\ (i_1,j_1)\neq(i_2,j_2)}} \mathsf{E}\left[\epsilon_{i_2 j_2}^2\right] \mathsf{E}\left[\left(\epsilon_{i_1 j_1}^2 - \mathsf{E}\left[\epsilon_{i_1 j_1}^2\right]\right)^2\right] \\
&+ \sum_{\substack{i_1,i_2\in[m] \\ i_1\neq i_2}} \sum_{j\in[m]} 2\mathsf{E}\left[\epsilon_{i_1 j}^4\right] \mathsf{E}\left[\epsilon_{i_2 j}^2\right] + \sum_{\substack{i_1,i_2,i_3\in[m] \\ i_1\neq i_2}} \sum_{\substack{j_1\in[mn],j_2\in[m] \\ (i_3,j_2)\neq(i_1,j_1) \\ (i_3,j_2)\neq(i_2,j_1)}} \mathsf{E}\left[\epsilon_{i_3 j_2}^2\right] \mathsf{E}\left[\epsilon_{i_1 j_1}^2\right] \mathsf{E}\left[\epsilon_{i_2 j_1}^2\right] \\
&\leq K_6 m^2 + K_4 K_2 m^4 n + 2K_4 K_2 m^3 + K_2^3 m^5 n \\
&\leq K_6 m^2 + 3K_4 K_2 m^4 n + K_2^3 m^5 n
\end{aligned}
$$

Combining (13) and (15) completes the first result, and combining (14) and (16), along with the fact that $\mathsf{E}\left[\left\|\epsilon^1\right\|_F^2\right] \leq K_2 m^2$, completes the second.

■

D.2. Step 2: Projection onto Estimated Spaces

Lemma 3 decomposes the error of the projection step in terms of the error of our column and row space estimates. For any slice M^k, our estimate of this slice is the projection of X^k onto the estimated subspaces $\hat{U}$ and $\hat{V}$, i.e. $P_{\hat{U}} M^k P_{\hat{V}} + P_{\hat{U}} \epsilon^k P_{\hat{V}}$. If $\hat{U}$ and $\hat{V}$ are close to U and V, then $P_{\hat{U}} M^k P_{\hat{V}} \approx P_U M^k P_V = M^k$.

Furthermore, since $\hat{U}$ and $\hat{V}$ are low-dimensional subspaces, $P_{\hat{U}}\epsilon^k P_{\hat{V}}$ will be small (this argument needs to be made carefully as $\hat{U}$ and $\hat{V}$ depend on ϵ^k).

Lemma 3. *Let $M^1 \in \mathbb{R}^{m\times m}$ be a matrix with column and row spaces $U, V \in \mathbb{R}^{m\times r}$. Let $\epsilon^1 \in \mathbb{R}^{m\times m}$ be a random matrix, and let $\hat{U}, \hat{V} \in \mathbb{R}^{m\times r}$ be random subspaces, where none of these variables are required to be independent. If $\hat{M}^1 = P_{\hat{U}}(M^1 + \epsilon^1)P_{\hat{V}}$, then taking expectation over $\epsilon^1, \hat{U}$, and $\hat{V}$:*

$$\mathsf{E}\left[\left\|\hat{M}^1 - M^1\right\|_F^2\right] \le 9\mathsf{E}\left[\left\|P_U\epsilon^1 P_V\right\|_F^2\right] + 3\left\|M^1\right\|_F^2 \mathsf{E}\left[4d(U,\hat{U})^2 + d(V,\hat{V})^2\right] + 9\mathsf{E}\left[\left\|\epsilon^1\right\|_F^2\left(4d(U,\hat{U})^2 + d(V,\hat{V})^2\right)\right].$$

Proof of Lemma 3. We begin by making the following decomposition, where the first two inequalities rely on the sub-multiplicative and sub-additive properties of the frobenius norm, and the first inequality also relies on the fact that $\left\|P_{\hat{V}} - P_V\right\|_2 \le 1$. The final inequality comes from $(a+b+c)^2 \le 3(a^2+b^2+c^2)$.

$$\begin{aligned}
\left\|\hat{M}^1 - M^1\right\|_F^2 &= \left\|P_{\hat{U}}(M^1+\epsilon^1)P_{\hat{V}} - M^1\right\|_F^2 \\
&= \left\|[P_U + (P_{\hat{U}} - P_U)]M^1[P_V + (P_{\hat{V}} - P_V)] - M^1 + P_{\hat{U}}\epsilon^1 P_{\hat{V}}\right\|_F^2 \\
&= \left\|M^1(P_{\hat{V}} - P_V) + (P_{\hat{U}} - P_U)\left[M^1 + M^1(P_{\hat{V}} - P_V)\right] + P_{\hat{U}}\epsilon^1 P_{\hat{V}}\right\|_F^2 \\
&\le \left(\left\|M^1(P_{\hat{V}} - P_V)\right\|_F + 2\left\|(P_{\hat{U}} - P_U)M^1\right\|_F + \left\|P_{\hat{U}}\epsilon^1 P_{\hat{V}}\right\|_F\right)^2 \\
&\le \left(\left\|M^1\right\|_F\left(2d(U,\hat{U}) + d(V,\hat{V})\right) + \left\|P_{\hat{U}}\epsilon^1 P_{\hat{V}}\right\|_F\right)^2 \\
&\le 3\left\|M^1\right\|_F^2\left(4d(U,\hat{U})^2 + d(V,\hat{V})^2\right) + 3\left\|P_{\hat{U}}\epsilon^1 P_{\hat{V}}\right\|_F^2.
\end{aligned}$$

The last term is decomposed further in a similar way:

$$\begin{aligned}
\left\|P_{\hat{U}}\epsilon^1 P_{\hat{V}}\right\|_F^2 &= \left\|[P_U + (P_{\hat{U}} - P_U)]\epsilon^1[P_V + (P_{\hat{V}} - P_V)]\right\|_F^2 \\
&\le \left(\left\|P_U\epsilon^1 P_V\right\|_F + \left\|\epsilon^1(P_{\hat{V}} - P_V)\right\|_F + 2\left\|(P_{\hat{U}} - P_U)\epsilon^1\right\|_F\right)^2 \\
&\le \left(\left\|P_U\epsilon^1 P_V\right\|_F + \left\|\epsilon^1\right\|_F\left(2d(U,\hat{U}) + d(V,\hat{V})\right)\right)^2 \\
&\le 3\left\|P_U\epsilon^1 P_V\right\|_F^2 + 3\left\|\epsilon^1\right\|_F^2\left(4d(U,\hat{U})^2 + d(V,\hat{V})^2\right).
\end{aligned}$$

We conclude the proof by taking expectations. ■

D.3. Final Steps

We are now ready to conclude the proof. Fix any slice $k \in [n]$. Lemma 3 first gives us:

$$\mathsf{E}\left[\left\|\hat{M}^k - M^k\right\|_F^2\right] \le 9\mathsf{E}\left[\left\|P_U\epsilon^k P_V\right\|_F^2\right] + 3\left\|M^k\right\|_F^2 \mathsf{E}\left[4d(U,\hat{U})^2 + d(V,\hat{V})^2\right] + 9\mathsf{E}\left[\left\|\epsilon^k\right\|_F^2\left(4d(U,\hat{U})^2 + d(V,\hat{V})^2\right)\right].$$

The first term is bounded as follows:

$$\mathsf{E}\left[\left\|P_U\epsilon^k P_V\right\|_F^2\right] = \mathsf{E}\left[\left\|UU^\top\epsilon^k VV^\top\right\|_F^2\right] = \mathsf{E}\left[\left\|U^\top\epsilon^k V\right\|_F^2\right] = \sum_{i_1\in[r], i_2\in[r]} \mathsf{E}\left[\left(U_{i_1}^\top\epsilon^k V_{i_2}\right)^2\right] \le r^2 K_2.$$

The remaining terms are bounded by applying Lemma 1 directly. Replacing running constants with c, we have

$$\begin{aligned}\left\|M^k\right\|_F^2 \mathsf{E}\left[d(U,\hat{U})^2\right] &\leq 24\left\|M^k\right\|_F^2 \frac{4K_2 m\left\|M_{(1)}\right\|_F^2 + K_4 m^2 n + K_2^2 m^3 n + m\delta^2}{\sigma_r^4(M_{(1)})} \\ &\leq cm^2 \frac{(K_2+K_2^2)m^3 n + K_4 m^2 n + m\delta^2}{\gamma_M^2 m^4 n^2/r^2},\end{aligned}$$

$$\begin{aligned}\mathsf{E}\left[\left\|\epsilon^k\right\|_F^2 d(U,\hat{U})^2\right] &\leq 24m^2 \frac{4K_4\left\|M^k\right\|_F^2/m + 4K_2^2 m\left\|M_{(1)}\right\|_F^2 + K_6 + 3K_4K_2m^2n + K_2^3m^3n + K_2m\delta^2}{\sigma_r^4(M_{(1)})} \\ &\leq cm^2 \frac{K_4 m + (K_2^2+K_2^3)m^3n + K_6 + K_4K_2m^2n + K_2m\delta^2}{\gamma_M^2 m^4 n^2/r^2},\end{aligned}$$

and similarly for $\mathsf{E}\left[d(V,\hat{V})^2\right]$ and $\mathsf{E}\left[\left\|\epsilon^k\right\|_F^2 d(V,\hat{V})^2\right]$. Note that in both calculations above, in the second inequality, we plug in our definition of γ_M, $\min_{i=1,2}\{\sigma_r^2(M_{(i)})\} \geq \gamma_M m^2 n/r$, and use the facts that $\left\|M^k\right\|_F^2 \leq m^2$ and $\left\|M_{(1)}\right\|_F^2 \leq m^2 n$.

Putting all of this together and rearranging terms, we have

$$\begin{aligned}\frac{1}{m^2}\mathsf{E}\left[\left\|\hat{M}^k - M^k\right\|_F^2\right] &\leq c\left[\frac{K_2 r^2}{m^2} + \frac{(K_2+K_2^2+K_2^3)m^3n + K_6 + K_4(K_2+1)m^2n + (K_2+1)m\delta^2}{\gamma_M^2 m^4 n^2/r^2}\right] \\ &\leq c\left[\frac{K_2 r^2}{m^2} + \frac{(K_2+K_2^2+K_2^3)r^2}{\gamma_M^2 mn} + \frac{K_6 r^2}{\gamma_M^2 m^4 n^2} + \frac{K_4(K_2+1)r^2}{\gamma_M^2 m^2 n} + \frac{(K_2+1)r^2\delta^2}{\gamma_M^2 m^3 n^2}\right] \\ &\leq c\left[\frac{K^2 r^2}{m^2} + \frac{K^2(K^4+1)r^2}{\gamma_M^2 mn} + \frac{(K^2+1)r^2\delta^2}{\gamma_M^2 m^3 n^2}\right].\end{aligned} \tag{17}$$

In the last step above, we consolidated terms by using the fact that K_2, K_4 and K_6 are respectively bounded by K^2, K^4 and K^6. Note that this entire analysis holds for any $k \in [n]$, so

$$\mathrm{SMSE}(\hat{M}) = \max_{k\in[n]} \frac{1}{m^2}\mathsf{E}\left[\left\|\hat{M}^k - M^k\right\|_F^2\right] \leq c\left[\frac{K^2 r^2}{m^2} + \frac{K^2(K^4+1)r^2}{\gamma_M^2 mn} + \frac{(K^2+1)r^2\delta^2}{\gamma_M^2 m^3 n^2}\right].$$

This concludes the proofs of Theorems 1 and 2. Corollary 1 follows by returning to (17), and applying (22) from Section G:

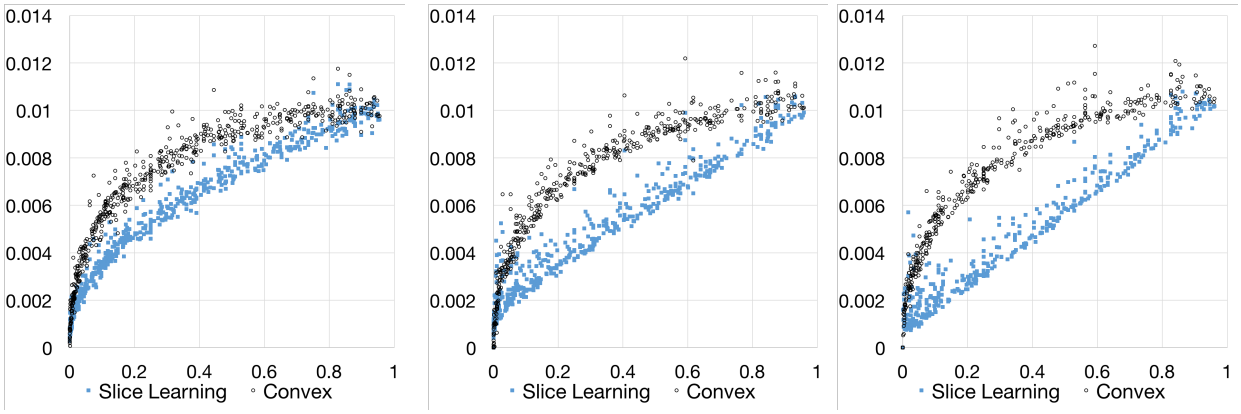
$$\begin{aligned}\frac{1}{m^2}\mathsf{E}\left[\left\|\hat{M}^k - M^k\right\|_F^2\right] &\leq c\left[\frac{K_2 r^2}{m^2} + \frac{(K_2+K_2^2+K_2^3)r^2}{\gamma_M^2 mn} + \frac{K_6 r^2}{\gamma_M^2 m^4 n^2} + \frac{K_4(K_2+1)r^2}{\gamma_M^2 m^2 n} + \frac{(K_2+1)r^2\delta^2}{\gamma_M^2 m^3 n^2}\right] \\ &\leq c\left[\frac{r^2}{m^2 p} + \frac{r^2}{\gamma_M^2 mnp^3} + \frac{r^2}{\gamma_M^2 m^4 n^2 p^5} + \frac{r^2}{\gamma_M^2 m^2 np^4} + \frac{r^2\delta^2}{\gamma_M^2 m^3 n^2 p}\right].\end{aligned}$$

■

E. Additional Synthetic Experiments

E.1. Additional Results for Section 6.1.2

Figure 7 depicts results from an extension of the experiment described in Section 6.1.2. This extension is meant to show the behavior of the slice learning and convex algorithms when the number of slices n falls in between the cases depicted in Figures 5a and 5b. We used the exact same experimental setup for three more values of n: 2, $\sqrt{r}$, and $r-1$. Figure 7a shows that the slice learning algorithm starts to show improvement even when $n = 2$, i.e. a single additional slice. Figures 7b and 7c reveal a progression as n increases until the slice learning algorithm achieves the $(r/m)^2$ rate.



(a) $n = 2$ **(b)** $n = \sqrt{r}$ **(c)** $n = r - 1$

Figure 7: Comparison of slice learning and convex approach for noisy slice recovery of low slice rank tensors with varying numbers of slices. SMSE vs. $(r/m)^2$ is plotted for each replication.

E.2. The Effect of Unbalanced Noise

To test the effect of unbalanced noise, we randomly generated ground truth tensors with varying sizes and low slice rank exactly as described in Section 6.1.2. Each tensor had $n = m$ slices. We again added mean-zero gaussian noise, but this time with varying standard deviations. In all cases, the total noise was kept constant, i.e. the noise model ϵ satisfied

$$\mathsf{E}\left[\left\|\epsilon^k\right\|_F^2\right] = .01m^2, \quad k = 1, \ldots, n. \tag{18}$$

For example, the experiments in Section 6.1.2 had all noise terms set with standard deviation 0.1, which satisfies (18).

In our first experiment, the top half of each slice ϵ^k had variance 0.2 and the bottom half had zero variance, i.e.

$$\mathsf{E}\left[\left(\epsilon_{ij}^k\right)^2\right] = \begin{cases} .02 & \text{if } i \leq m/2 \\ 0 & \text{if } i > m/2 \end{cases}.$$

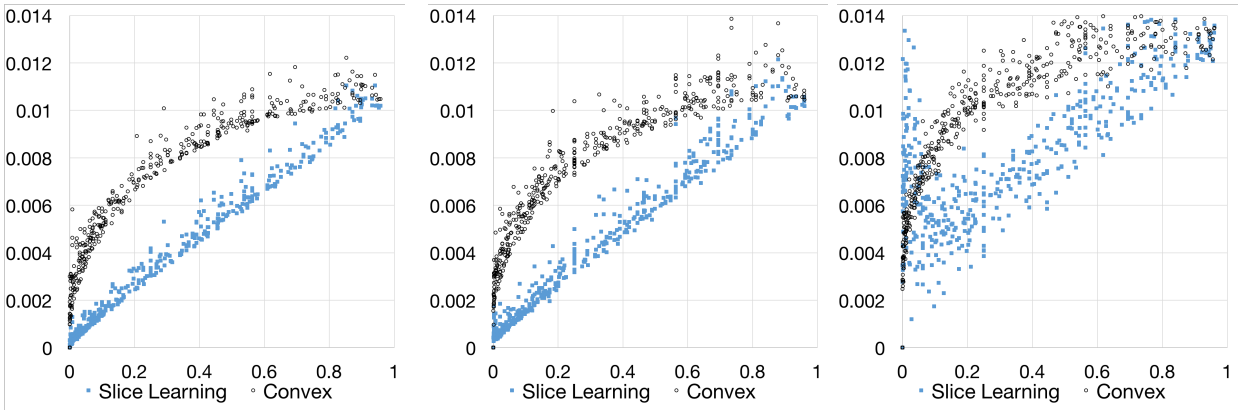
In Section 5.1, we defined an unbalance term δ, and here this corresponds to $\delta^2 = O(m^2n^2)$, which is the highest scaling for δ when the variances are bounded. The results are summarized in Figure 8b, and compared with Figure 8a, which is a reproduction of the corresponding balanced noise experiment from Section 6.1.2,

they show that the slice learning algorithm still performs well, even though Theorem 2 no longer makes such a guarantee.

For our second experiment, only the top two rows have noise, but this noise is allowed to grow unbounded:

$$\mathsf{E}\left[\left(\epsilon_{ij}^{k}\right)^{2}\right] = \begin{cases} .005m & \text{if } i = 1, 2 \\ 0 & \text{if } i \geq 3 \end{cases}.$$

This case corresponds to $\delta^2 = O(m^3 n^2)$, and is summarized in Figure 8c. At this point, the slice learning algorithm performs poorly when r/m is small.



(a) $\delta = 0$ **(b)** $\delta = O(m^2 n^2)$ **(c)** $\delta = O(m^3 n^2)$

Figure 8: Comparison of slice learning and convex approach for noisy slice recovery of low slice rank tensors with varying levels of unbalanced noise. SMSE vs. $(r/m)^2$ is plotted for each replication. Display (a) is a reproduction of Figure 5c.

For another view into what is going on here, we fixed a particular tensor size and rank, and varied the unbalance level further. We randomly generated tensors of size $30 \times 30 \times 30$ with slice rank 5. For the added noise, as before, the bottom rows had zero variance, and the variance of the top rows were set to satisfy (18), so that the unbalance level can be varied by changing the number of these non-zero variance rows. For each level of unbalance, we ran 15 replications. The results shown in Figure 9 reveal that the performance of both algorithms worsens as the unbalance increases, and that the slice learning algorithm outperforms the convex algorithm for lower levels.

E.3. The Effect of Correlated Noise

The goal of our final set of synthetic experiments is to evaluate what happens when the independence assumption is relaxed. Following the same experimental setup as in the previous section, we randomly generated tensors of size $30 \times 30 \times 5$ with slice rank 5. We will focus on the setting where the noise between slices is correlated. To do so, each noise term was a zero-mean gaussian with standard deviation 0.1, but the covariances between corresponding entries across slices were allowed to vary:

$$\mathrm{Cov}(\epsilon_{ij}^{k}, \epsilon_{ij}^{\ell}) = c, \quad k \neq \ell,$$

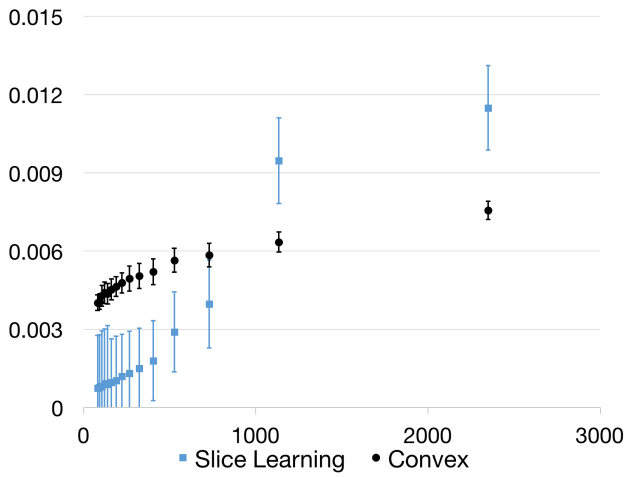
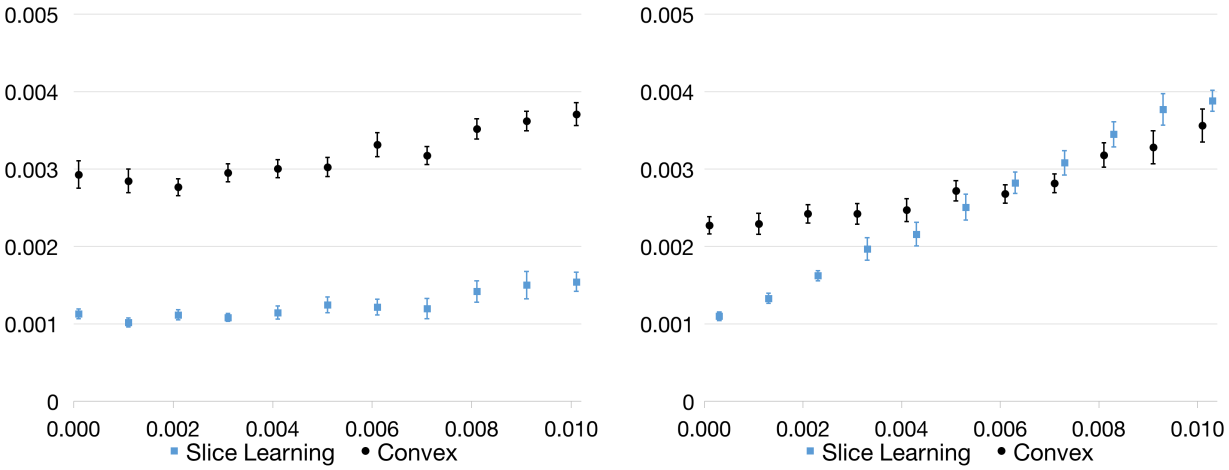


Figure 9: Comparison of slice learning and convex approach for tensors of size $30 \times 30 \times 30$ with slice rank 5 and varying levels of unbalanced noise. SMSE vs. δ^2 is plotted. Each point is the aggregate of 15 replications.

where c takes values between 0 and its maximum possible value of 0.01. Put another way, for each i, j, the vector of corresponding noise terms $(\epsilon_{ij}^1, \ldots, \epsilon_{ij}^n)$ was an independently generated mean-zero multivariate gaussian, with the above covariance structure.



(a) Different Slices

(b) Identical Slices

Figure 10: Comparison of slice learning and convex approach for tensors of size $30 \times 30 \times 5$ with slice rank 5 and varying levels correlated noise. SMSE vs. covariance is plotted for each replication. Each point is the aggregate of 15 replications, with 95% error bars. Points in Figure (b) are slightly offset horizontally to show overlapping error bars more clearly.

The results in Figure 10a show that both algorithms are only slightly affected by increasing covariance; note that at the most extreme case of $c = .01$, the noise terms across slices are identical! However, this does not necessarily mean that correlated noise is always innocuous, at least without further assumptions beyond slice rank. For example, for the experiment shown in Figure 10b, we used randomly generated tensors whose slices are identical. These tensors still have low slice rank, but the effect of correlation on noise is much

stronger. In particular, the extreme case $c = .01$ is now identical to having only a single slice of data.

F. Additional Results for Xiami Experiments

Tables 3 and 4 summarize the results of the experiments using data from `Xiami.com`, in terms of recovering the Download and Listen slices. See Section 6.2 for a detailed description of the experiment.

Users	Songs	Sparsity	Naive	Matrix	Slice
2,412	1,541	9.6	0.84 (11)	0.87 (7)	0.91 (12)
4,951	2,049	7.9	0.83 (14)	0.85 (9)	0.91 (12)
27,411	3,472	3.2	0.83 (11)	0.86 (8)	0.91 (14)
23,300	10,106	14.2	0.94 (18)	0.93 (13)	0.94 (18)
53,713	10,199	8.2	0.93 (10)	0.93 (7)	0.94 (20)

Table 3: Summary of experiments on Xiami data for recovering the Download slice. Each row corresponds to an experiment on a subset of the data. Columns 'Users' and 'Songs' show the number of users and songs in each experiment, and 'Sparsity' gives the average number of downloads per user in the data. Results for the naive benchmark, the matrix-based benchmark, and the slice learning algorithm are shown in the last three columns. The average AUC over 10 replications is reported, along with the rank in parentheses.

Users	Songs	Sparsity	Naive	Matrix	Slice
2,412	1,541	14.8	0.88 (6)	0.88 (7)	0.91 (11)
4,951	2,049	12.6	0.88 (7)	0.87 (11)	0.91 (11)
27,411	3,472	7.5	0.87 (6)	0.87 (3)	0.90 (9)
23,300	10,106	21.3	0.94 (7)	0.92 (8)	0.94 (15)
53,713	10,199	14.1	0.92 (5)	0.92 (12)	0.93 (7)

Table 4: Summary of experiments on Xiami data for recovering the Listen slice. Each row corresponds to an experiment on a subset of the data. Columns 'Users' and 'Songs' show the number of users and songs in each experiment, and 'Sparsity' gives the average number of listens per user in the data. Results for the naive benchmark, the matrix-based benchmark, and the slice learning algorithm are shown in the last three columns. The average AUC over 10 replications is reported, along with the rank in parentheses.

G. Commentary on Tensor Completion

To this point, our work has applied to the problem of noisy tensor recovery, a framework that addresses settings such as the retail example and our experiment with music streaming data. As discussed in Section 1, there are settings and applications where the existing data instead can be represented as a partially observed tensor, i.e. the tensor completion problem. To model the tensor completion setting mathematically, let Ω be the set of indices (i, j, k) of the observed entries, so that our data consists of the set of values $\{M^k_{ij} : (i, j, k) \in \Omega\}$, where $M \in \mathbb{R}^{m \times m \times n}$ is the ground truth tensor; for this discussion, we are assuming that the observed entries are observed without noise. Moreover, assume that Ω is generated randomly such

that each entry is observed independently with probability $p > 0$. The goal then is to design an estimator $\hat{M}$, which is now a function of only the observed entries.

Theoretical guarantees in this area should address performance in terms of four parameters now: m, n, r and p. In particular, the quantity of m^2p is of interest as it is equal to the expected number of observed entries per slice. Consider that a necessary condition for any algorithm to complete a tensor is

$$m^2p = \Omega(r^2 + mr/n). \tag{19}$$

This lower bound comes from the fact that having slice rank r allows for $O(nr^2 + mr)$ degrees of freedom, and that the quantity m^2np is the expected total number of observed entries. Dividing through by n gives the lower bound in terms of the number of observed entries per slice. Achieving this bound would mean that (1) in the best case, the per-slice data requirement is r^2, which does not depend on the size of the slices, and (2) as sources of side information are added, the per-slice data requirement decreases linearly until r^2. In contrast, a matrix-based approach such as using a matrix completion algorithm on each slice separately would reduce to completing n matrices of size $m \times m$ and rank r. In that case, the best known guarantees (e.g. Gross (2011)), which have matching lower bounds, are exact recovery with high probability given that

$$m^2p = \Omega(mr),$$

where we have omitted polylogarithmic factors. Another matrix-based approach of applying a matrix completion algorithm on a single unfolding would not improve on this guarantee.

There has been much work in designing algorithm tailored to the tensor completion problem. The strongest existing guarantee is by Yuan and Zhang (2015), who propose an algorithm that recovers M exactly, with high probability, when

$$m^2p = \Omega(r^2 + mr^2/n + m\sqrt{r}/\sqrt{n}).$$

This result makes significant progress toward the lower bound (19), but unfortunately the proposed algorithm is computationally intractable. Huang et al. (2015) analyze a tractable algorithm and show that a sufficient condition for recovery is

$$m^2p = \Omega(mr). \tag{20}$$

Now, when entries are observed uniformly at random, it is possible to map completion problems to noisy recovery problems by dividing the observed entries by p and treating unobserved entries as zero. This technique has been applied in the matrix completion setting (Achlioptas and McSherry (2007), Keshavan et al. (2010), Chatterjee (2014)). Following the same arguments, we could use the slice learning algorithm for tensor completion: we (1) divide each observed entry by the proportion of entries observed, i.e. $|\Omega|/m^2n$, (2) treat all hidden entries as observations of the value zero, and (3) execute the slice learning algorithm as usual on this modified tensor. In other words, we execute the slice learning algorithm on tensor $(m^2n/|\Omega|)M_\Omega$, where M_Ω is the tensor defined as

$$(M_\Omega)^k_{ij} = \begin{cases} M^k_{ij} & \text{if } (i,j,k) \in \Omega \\ 0 & \text{if } (i,j,k) \notin \Omega \end{cases}.$$

The modified tensor $(1/p)M_\Omega$ is a noisy observation of M. To see this, note that we can define the additive

noise term of the $(i,j,k)^{\text{th}}$ entry as

$$\epsilon_{ij}^k \sim \frac{\text{Ber}(p)}{p} M_{ij}^k - M_{ij}^k. \tag{21}$$

These noise terms are independent with mean zero, and $(1/p)M_\Omega = M + \epsilon$. It follows that M could be estimated by applying our slice learning algorithm to $(1/p)M_\Omega$. Since we do now know p, we use the proportion of observed entries as an estimate of p.

Since we can reformulate the completion problem as a noisy recovery problem, a natural question then is what Theorems 1 and 2 tell us about the performance of the slice learning algorithm, where again performance is measured in terms of p. To analyze the slice learning algorithm as in Theorems 1 and 2, assume that the entries of M lie in $[-1,1]$. It follows directly from (21) that

$$\mathsf{E}[(\epsilon_{ij}^k)^d] = (M_{ij}^k)^d \left(\frac{(1-p)^d}{p^{d-1}} + (1-p) \right) \leq \frac{1}{p^{d-1}}, \tag{22}$$

for even values of d. Now in the statements of Theorems 1 and 2, the guarantee is parameterized by K, where it is assumed that $\mathsf{E}[(\epsilon_{ij}^k)^6] \leq K^6$. This is actually a compact version of a more specific guarantee we show (see (17) in the Appendix) that is parameterized by the second, fourth, and sixth moments of the noise terms. Combining (22) and that guarantee, we have the following result:

Corollary 1. *Assume the entries of M lie in $[-1,1]$. Suppose Ω is randomly chosen such that each index is included independently with probability $p > 0$. Let $\hat{M}$ be the result of applying the slice learning algorithm to $(1/p)M_\Omega$. Then there exists a constant c such that*

$$\text{SMSE}(\hat{M}) \leq c \left[\frac{r^2}{m^2 p} + \frac{r^2}{\gamma_M^2 mnp^3} + \frac{r^2}{\gamma_M^2 m^4 n^2 p^5} + \frac{r^2}{\gamma_M^2 m^2 np^4} + \frac{r^2 \delta^2}{\gamma_M^2 m^3 n^2 p} \right].$$

Corollary 1 implies that we can expect to recover M using the slice learning algorithm as long as the denominator of each term in the guarantee is much larger than the numerator. To make more sense of this sufficient condition, let us assume that the noise is balanced ($\delta = 0$) and that γ_M scales as a constant, in which case after some algebraic contortion, the condition can be expressed as

$$m^2 p = \Omega(r^2 + m^{5/3} r^{2/3}/n^{1/3} + m^{3/2} r^{1/2}/n^{1/4}).$$

Unlike the guarantee (20) of Huang et al. (2015), this guarantee decreases with n and achieves the final r^2 value for sufficiently large n. On the other hand, the scaling with m is worse, and so is only an improvement when n grows sufficiently faster than m. Finally, our algorithm is dominated by that of Yuan and Zhang (2015), but is computationally efficient. The guarantees we have described here are summarized in Table 5.

G.1. Experiment

We ran a set of synthetic experiments to test the performance of the slice learning algorithm in the tensor completion setting. Our first experiment is an exact replication of one of the experiments from Gandy et al. (2011): we randomly generated tensors of size $20 \times 30 \times 40$ with Tucker rank $(2,2,2)$, using the same procedure described in §6.1.1, and observed each entry with probability 0.6. We benchmarked against the

Method	Per-Slice Observations
Lower Bound	$r^2 + mr/n$
Matrix-Complete Slices	mr
Matrix-Complete Unfolding	mr
Huang et al. (2015)	mr
Yuan and Zhang (2015)	$r^2 + mr^2/n + mr^{1/2}/n^{1/2}$
Slice Learning	$r^2 + m^{5/3}r^{2/3}/n^{1/3} + m^{3/2}r^{1/2}/n^{1/4}$

Table 5: Comparison of guarantees for tensor completion algorithms, in terms of the number of per-slice observations sufficient for completion. Logarithmic terms are omitted.

convex algorithm

$$\underset{Y}{\operatorname{argmin}} \sum_{i=1}^{3} \lambda \left\| Y_{(i)} \right\|_* + \|Y_\Omega - X_\Omega\|_F^2 , \tag{23}$$

which we solved via the Douglas-Rachford splitting method described in Gandy et al. (2011), using their recommended values for the step size and index of the proximal operator, and solving (23) multiple times with increasing values of λ until the solutions converged. This procedure was repeated for 60 replications, and the results are summarized in Figure 11a, where we give the root mean squared error (RMSE, square root of MSE) averaged over the 60 replications for both the slice learning algorithm and the convex algorithm (23). Since Douglas-Rachford is an iterative method, we report the average RMSE at various points in the procedure, i.e. various numbers of iterations. On the other hand, the slice learning algorithm consists of only a single 'iteration', and so a single value is reported. We also performed a second experiment that is a larger version of the first. We randomly generated tensors of size $200 \times 200 \times 200$ with Tucker rank $(10, 10, 10)$, with the rest of the experiment remaining the same. The results are summarized in Figure 11b.

Method	Iterations	RMSE
Slice Learning	1	1.6×10^{-3}
Convex	1	1.5×10^{-2}
	10	5.7×10^{-3}
	*24	1.2×10^{-3}
	100	8.2×10^{-6}

(a) $20 \times 30 \times 40$ tensors of Tucker rank $(2, 2, 2)$

Method	Iterations	RMSE
Slice Learning	1	3.9×10^{-4}
Convex	1	9.7×10^{-3}
	10	4.8×10^{-3}
	*22	3.2×10^{-4}
	50	1.1×10^{-4}

(b) $200 \times 200 \times 200$ tensors of Tucker rank $(10, 10, 10)$

Figure 11: Results of synthetic completion experiments. Entries were observed with probability 0.6. RMSE and iteration count are reported, averaged over 60 replications. Starred (*) rows correspond to the iteration of the convex algorithm in which, for the first time, the average RMSE falls below that of the slice learning algorithm.

The slice learning algorithm performs reasonably well, achieving RMSE on the order of 10^{-3} to 10^{-4}. To put this into perspective, the size of the elements of the randomly generated tensors is on the order of 10^{-2},

so this RMSE amounts to a relative error of about 1% to 10%. Ignoring computational costs, the convex algorithm outperforms the slice learning algorithm, achieving a lower average RMSE in both experiments after approximately 20 iterations. Moreover, we observe the RMSE continuing to decrease with each iteration, and indeed with enough iterations the RMSE may go to zero (or machine precision), corresponding to *exact* recovery of the original tensor. On the other hand, when factoring in computational costs, the slice learning algorithm performs very well. Each Douglas-Rachford iteration requires singular value decompositions of all three (dense) unfoldings of the tensor, which means each iteration is more computationally expensive than the entire slice learning algorithm, which only requires SVDs of two (sparse) unfoldings. In concrete terms, this meant that for our larger experiment, the slice learning algorithm ran in less than a minute, while the convex approach took upwards of one hour to reach the same level of accuracy.

References

Achlioptas D, McSherry F (2007) Fast computation of low-rank matrix approximations. *Journal of the ACM (JACM)* 54(2):9.

Ansari A, Essegaier S, Kohli R (2000) Internet recommendation systems. *Journal of Marketing research* 37(3):363–375.

Ansari A, Mela CF (2003) E-customization. *Journal of marketing research* 40(2):131–145.

Archak N, Ghose A, Ipeirotis PG (2011) Deriving the pricing power of product features by mining consumer reviews. *Management Science* 57(8):1485–1509.

Bennett J, Lanning S (2007) The netflix prize. *Proceedings of KDD cup and workshop*, volume 2007, 35.

Besbes O, Gur Y, Zeevi A (2015) Optimization in online content recommendation services: Beyond click-through rates. *Manufacturing & Service Operations Management* 18(1):15–33.

Bodapati AV (2008) Recommendation systems with purchase data. *Journal of marketing research* 45(1):77–93.

Breese JS, Heckerman D, Kadie C (1998) Empirical analysis of predictive algorithms for collaborative filtering. *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, 43–52 (Morgan Kaufmann Publishers Inc.).

Bucklin RE, Sismeiro C (2003) A model of web site browsing behavior estimated on clickstream data. *Journal of marketing research* 40(3):249–267.

Chatterjee S (2014) Matrix estimation by universal singular value thresholding. *The Annals of Statistics* 43(1):177–214.

Chintagunta P, Dubé JP, Goh KY (2005) Beyond the endogeneity bias: The effect of unmeasured brand characteristics on household-level brand choice models. *Management Science* 51(5):832–849.

Chintagunta PK, Dube JP (2005) Estimating a stockkeeping-unit-level brand choice model that combines household panel data and store data. *Journal of Marketing Research* 42(3):368–379.

Chong JK, Ho TH, Tang CS (2001) A modeling framework for category assortment planning. *Manufacturing & Service Operations Management* 3(3):191–210.

Chung TS, Rust RT, Wedel M (2009) My mobile music: An adaptive personalization system for digital audio players. *Marketing Science* 28(1):52–68.

Das SR, Chen MY (2007) Yahoo! for amazon: Sentiment extraction from small talk on the web. *Management Science* 53(9):1375–1388.

Davis C, Kahan WM (1970) The rotation of eigenvectors by a perturbation. iii. *SIAM Journal on Numerical Analysis* 7(1):1–46.

Fader PS, Hardie BG (1996) Modeling consumer choice among skus. *Journal of marketing Research* 442–452.

Gandy S, Recht B, Yamada I (2011) Tensor completion and low-n-rank tensor recovery via convex optimization. *Inverse Problems* 27(2):025010.

Ghose A, Ipeirotis PG, Li B (2012) Designing ranking systems for hotels on travel search engines by mining user-generated and crowdsourced content. *Marketing Science* 31(3):493–520.

Gopinath S, Chintagunta PK, Venkataraman S (2013) Blogs, advertising, and local-market movie box office performance. *Management Science* 59(12):2635–2654.

Gross D (2011) Recovering low-rank matrices from few coefficients in any basis. *Information Theory, IEEE Transactions on* 57(3):1548–1566.

Herlocker J, Konstan JA, Riedl J (2002) An empirical analysis of design choices in neighborhood-based collaborative filtering algorithms. *Information retrieval* 5(4):287–310.

Hu M, Liu B (2004) Mining and summarizing customer reviews. *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, 168–177 (ACM).

Huang B, Mu C, Goldfarb D, Wright J (2015) Provable models for robust low-rank tensor completion. *Pacific Journal of Optimization* 11:339–364.

Hui SK, Huang Y, Suher J, Inman JJ (2013) Deconstructing the "first moment of truth": Understanding unplanned consideration and purchase conversion using in-store video tracking. *Journal of Marketing Research* 50(4):445–462.

Keshavan RH, Montanari A, Oh S (2010) Matrix completion from a few entries. *Information Theory, IEEE Transactions on* 56(6):2980–2998.

Kolda TG, Bader BW (2009) Tensor decompositions and applications. *SIAM review* 51(3):455–500.

Liu X, Singh PV, Srinivasan K (2016) A structured analysis of unstructured big data by leveraging cloud computing. *Marketing Science* 35(3):363–388.

Lu S, Xiao L, Ding M (2016) A video-based automated recommender (var) system for garments. *Marketing Science* 35(3):484–510.

MobileCommerceDaily (2013) Macy's exec underpins navigation, in-store scanning for holiday marketing success. `http://www.mobilecommercedaily.com/macy's-exec-underpins-navigation-in-store-scanning-for-holiday-marketing-success`.

Moe WW (2006) An empirical two-stage choice model with varying decision rules applied to internet clickstream data. *Journal of Marketing Research* 43(4):680–692.

Montgomery AL, Li S, Srinivasan K, Liechty JC (2004) Modeling online browsing and path analysis using clickstream data. *Marketing Science* 23(4):579–595.

Moon S, Russell GJ (2008) Predicting product purchase from inferred customer similarity: An autologistic model approach. *Management Science* 54(1):71–82.

Sismeiro C, Bucklin RE (2004) Modeling purchase behavior at an e-commerce web site: A task-completion approach. *Journal of marketing research* 41(3):306–323.

Trusov M, Ma L, Jamal Z (2016) Crumbs of the cookie: User profiling in customer-base analysis and behavioral targeting. *Marketing Science* 35(3):405–426.

Wedin PÅ (1972) Perturbation bounds in connection with singular value decomposition. *BIT Numerical Mathematics* 12(1):99–111.

Wu J, Rangaswamy A (2003) A fuzzy set model of search and consideration with an application to an online market. *Marketing Science* 22(3):411–434.

Yu Y, Wang T, Samworth RJ (2015) A useful variant of the davis–kahan theorem for statisticians. *Biometrika* 102(2):315–323.

Yuan M, Zhang CH (2015) On tensor completion via nuclear norm minimization. *Foundations of Computational Mathematics* 1–38.