Optimal Recovery of Tensor Slices

Vivek F. Farias
MIT Sloan School of Management

Andrew A. Li
MIT Operations Research Center

Abstract

We consider the problem of large scale matrix recovery given side information in the form of additional matrices of conforming dimension. This is a parsimonious model that captures a number of interesting problems including context and location aware recommendations, personalized 'tag' learning, demand learning with side information, etc. 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 a tensor from noisy observations of the tensor. We provide an efficient algorithm to recover slices of structurally 'simple' tensors given noisy observations of the tensor's entries; our definition of simplicity subsumes low-rank tensors for a variety of definitions of tensor rank. Our algorithm is practical for large datasets and provides a significant performance improvement over state of the art incumbent approaches to tensor recovery. We establish theoretical recovery guarantees that under reasonable assumptions are minimax optimal for slice recovery. These guarantees also imply the first minimax optimal guarantees for recovering tensors of low Tucker rank and general noise. Experiments on data from a music streaming service demonstrate the performance and scalability of our algorithm.

1 Introduction

Consider the problem of learning the propensity of a customer for a product from, say, observations of customer transactions. One approach to this problem is to cast it as a problem of noisy matrix recovery. Specifically, as a toy example, imagine an $m \times m$ matrix P for which $P_{i,j}$ encodes the probability that customer i will buy a product j over some specific period. P is never observed; instead we observe transactions which we encode as a matrix X for which $X_{i,j} = 1$ if customer i purchases product j, and is 0 otherwise. Our task can now roughly be stated as recovering P, having observed X. This is an incredibly well studied problem of noisy matrix recovery.

Clearly, we can only hope to recover P if it is, in some sense, 'simple'. One common notion of simplicity here is the rank of P, and it is well known that the minimax error of any matrix estimator is $\Omega\left(r/m\right)$. For such a guarantee to be meaningful in a relative sense, we need that $\|P\|_1 \geq rm$, or in other words, we loosely expect to see r transactions per user on average. Unfortunately, real world datasets are incredibly sparse: the largest retailers have upwards of 10^8 active users and products, but only 10^9 products sold *yearly* (FastCompany (2015)). Given the discussion thus far, data this sparse will only let us learn very simple ground truth matrices.

The setup above paints a much bleaker picture than is the reality of the task at hand: the sales matrix is just one 'slice' of the data a retailer may collect – other slices can be generated from clickstream data, advertising, surveys, etc. that encode interactions between a customer and a product that are distinct from a transaction but may well inform the likelihood of a transaction. Succinctly, the data consists of many other 'slices' and these other slices may contain information useful to estimate the original sales matrix P in our example. *The objective of this paper is to formalize and exploit this observation.*

In analogy to the matrix recovery problem above, a reasonable formalization of the task we have laid out is via the problem of recovering a three dimensional *tensor* from its noisy observations, and there is by now, a fairly robust literature on the problem of tensor recovery. 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 certain convex surrogate to the

Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS) 2017, Fort Lauderdale, Florida, USA. JMLR: W&CP volume 54. Copyright 2017 by the author(s).

tensor rank is small. We will see that employing this approach in our setting presents a few challenges.

Computation: The approaches at hand appear difficult to scale to massive amounts of data and typically call for dense matrix operations at scales that are untenable.

Rates of Recovery: The recovery rates established for these approaches fall well short of what we are hoping for: as we will show in the sequel, in our setting of three dimensional tensors, these rates 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.

1.1 Our Contribution

The present paper proposes a simple algorithm for the problem of slice learning; applied to all slices this also results in an algorithm for the recovery of three dimensional tensors from their noisy observations. Relative to the extant literature, we make the following contributions:

Statistical Power and Near-Optimal Recovery: Under a broad set of assumptions, we show the strongest available guarantees for the recovery of a broad class of three dimensional tensors from their noisy observations. Our analysis also admits guarantees on error rates for individual slices which do not have a counterpart in the extant tensor recovery literature. In addition to the above, the requirements we place on the underlying 'ground truth' tensor and the nature on the noise are looser than those required for rigorous recovery via existing methods.

Scalability: The approach we propose 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.

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 easily generalize for higher order tensors, while ours does not. On the other hand, there is a vast array of applications that makes studying 3D tensors specifically important, ranging from latent variable models (Anandkumar et al. (2014)) to spatio-temporal analysis (Bahadori et al. (2014)) to neuroimaging (Zhou et al. (2013)). Furthermore, consider that the tools for completing two-dimensional tensors (matrix recovery) differ substantially from the tools for one-dimensional tensors (vector recovery, compressed sensing), and in particular, state-of-the-art matrix recovery algorithms are not direct generalizations of state-of-the-art compressed sensing algorithms, nor are compressed sensing algorithms special cases of matrix completion algorithms. Therefore, it is sensible to believe that algorithms for 3D tensor recovery will be specialized.

The remainder of this paper is organized as follows. We begin by introducing our problem of slice recovery and notion of simplicity for tensors in §2. In §3, we show a minimax lower bound for tensor and slice recovery, and review incumbent approaches to tensor recovery; for space reasons, this discussion will serve as our review of the related literature, along with inline citations made throughout. We present our algorithm in §4 and a theoretical recovery guarantee in §5. Experiments are discussed in §6, and §7 concludes the paper.

2 Model and Problem

We begin by introducing some basic notation for three dimensional tensors. Let $M \in \mathbb{R}^{m_1 \times m_2 \times n}$ denote a three-dimensional tensor that represents the 'ground truth'. This tensor can be viewed as a collection of n matrices, denoted $M^1, \ldots, M^n \in \mathbb{R}^{m_1 \times m_2}$. We call each of the matrices M^k 'slices', and for ease of notation, we will assume the slices are square, so $m_1 = m_2 = m$; all our results easily extend to rectangular slices. We denote the element in the i^{th} row and j^{th} column of slice M^k by $M^k_{i,j}$. We will require that every entry of M lie in $[-1, 1]$; this can always be satisfied by rescaling.

A common operation on tensors that we will use frequently here is 'unfolding'. The mode-1 unfolding of M, denoted $M_{(1)}$, is the m-by-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 is the m-by-nm matrix whose columns are the *rows* of $M^1, \ldots, M^n$.[1]

Our problem is to recover M, or often a single slice of 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

[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

even identically distributed[2].

In *tensor recovery*, the problem is to construct an estimator $\hat{M}$, which is a function of the noisy observation X, to minimize some loss function with respect to M. We will take the loss function here to be the mean-squared error (MSE):

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

As we have discussed, oftentimes the problem is to recover a single slice of the tensor (having observed X). In this case, MSE is not an appropriate 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)*:

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

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 MSE, any upper bound for SMSE will apply to MSE, and therefore the tensor recovery problem.

2.1 Simplicity and Slice rank

Achieving low MSE or SMSE in our problem is impossible without further assumptions on the 'simplicity' of M. To this end, we introduce the notion of *slice rank* and will subsequently require that M has small slice rank. To be clear at the outset, tensors with low canonical (or CP) rank or low Tucker rank will automatically have low slice rank, so that the class of tensors of low slice rank will contain those other classes.

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

Note that the ranks of the mode-1 and mode-2 unfoldings need not be equal, and as such, their maximum is taken in the definition above.

An equivalent, and perhaps more illuminating definition is the following: if M has slice rank r, then there exist matrices $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{m\times r}$ such that each slice of M, M^j can be decomposed as

$$M^j = US^jV^\top \tag{1}$$

for some $S^j \in \mathbb{R}^{r\times r}$. This decomposition is a generalization of one proposed by Nickel et al. (2011) for three dimensional tensors with *symmetric* slices.

The slice-wise form of (1) suggests that in fact the tensor representation we have used so far may not be needed, as the entire structure is well described as a set of matrices. However, the value in viewing this as a tensor is that this structure subsumes two popularly used notions of tensor rank, the canonical (or CP) rank and the Tucker rank, and therefore our recovery results apply in those settings as well. In the language of the definition above, these notions of rank can be defined as follows:

1. If M has canonical rank r, then there exist matrices $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{m\times r}$ such that each slice of M, M^j can be decomposed as $M^j = US^jV^\top$, where each S^j is a *diagonal* matrix.
2. If M has Tucker rank (r, r, l), then there exist matrices $U \in \mathbb{R}^{m\times r}$ and $V \in \mathbb{R}^{m\times r}$ such that each slice of M, M^j can be decomposed as $M^j = US^jV^\top$, wherein $\mathrm{rank}\left(\mathrm{span}\left(s^1, s^2, \dots, s^n\right)\right) = l$ where $s^j \in \mathbb{R}^{r^2}$ is a vectorization of S^j.

The equivalence of both the above definitions to the typical definitions employed for canonical and Tucker rank respectively require proof; this can be found in the Appendix as Proposition 2. These definitions make it apparent that the sets of tensors with canonical rank bounded by r, $\mathcal{CP}(r)$, Tucker rank bounded component-wise by (r, r, l), $\mathrm{Tucker}(r, r, l)$, and slice rank bounded by r, $\mathrm{Slice}(r)$, satisfy:

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

which makes precise the notion that requiring low slice rank is a weaker requirement than requiring either low canonical rank, or low Tucker rank.

3 A Lower Bound and Incumbent Approaches

We begin this section with establishing a minimax result for the recovery of three dimensional tensors:

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

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

where C *is a universal constant.*

[2] In our analysis, we will require that the row norms of the unfoldings $\epsilon_{(1)}$ and $\epsilon_{(2)}$ be balanced in a sense we will make precise later. This will also yield an interesting algorithmic insight related to trimming procedures.

An immediate consequences of Proposition 1, which we alluded to in the introduction, is that in the matrix recovery setting, i.e. $n = 1$ so that M is a matrix of rank r, we have a minimax lower bound of $\mathrm{MSE}(\hat{M}) = \Omega(r/m)$. 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 $\mathrm{MSE} = O(r/m)$ and $\mathrm{SMSE} = O(r/m)$. Beating such an approach is precisely the motivation for our work here. Fortunately, Proposition 1 *also* suggests that we may be able to outperform this naive approach. Specifically, the minimax bound suggests that with sufficiently many slices (i.e. large n), it might be possible to achieve $\mathrm{MSE} = O(r^2/m^2)$, which is *substantially* smaller. Where on this spectrum do existing approaches to tensor recovery fall?

3.1 Incumbent Approaches

The dominant approach to tensor recovery relies on convex optimization. Variants of this approach attempt to find a tensor that is simultaneously 'close' to X while also minimizing a a convex surrogate for the tensor rank. Specialized to our setting, one such variant would consider the estimator $\hat{M}$ that is given by the optimal solution to the problem

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

where we use the sum of the nuclear norms of each tensor unfolding as a scalarized, convex surrogate of the Tucker rank. The parameter λ is a weight chosen by the user that 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)). Tomioka et al. (2011) show that if the noise tensor has i.i.d. Gaussian entries with standard deviation σ, the estimator above achieves

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

if the Tucker rank of M is (r,r,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 special setting where random linear combinations of the entries of M are observed, and Mu et al. (2014) 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). Mu et al. (2014) also propose another convex relaxation that works well for higher-dimensional tensors, though for three dimensional tensors, it does not improve on (3). Tomioka and Suzuki (2013) propose another convex problem, but again their result for tensors of Tucker rank (r, r, r) is the same as (3).

In short, (3) is the best known noisy recovery guarantee for any convex approach. It should be emphasized that the guarantee in (3) is not any stronger than the naive benchmark, no matter the number of slices, and furthermore, there is no guarantee for SMSE (while there is one for the naive approach).

Outside of convex formulations, Suzuki (2015) recently proposed a Bayesian estimator that matches the lower bound in Proposition 1 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).

Finally, some 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 understanding conditions under which *exact* recovery is possible, though this requires incoherence conditions similar to those made for matrix completion (see Candès and Tao (2010), Gross (2011), Recht (2011)).

4 An Algorithm for Slice Learning

We will now present our algorithm for slice recovery and tensor recovery. Recall that X is our observed data tensor, and that the ground truth tensor, M being of Slice rank r permits a decomposition wherein every slice M^j can be represented as $M^j = US^jV^\top$ where S^j is an $r \times r$ matrix, and U, V are both $m \times r$ matrices. Our slice learning algorithm proceeds in two stages. In the first stage, we use data from every slice to estimate the column space and row space common to all slices – specifically we will estimate the subspaces spanned by the columns of U and V. In the second stage we use these learned subspaces to then estimate M^j by projecting the observed slice X^j onto these learned spaces.

Learning Subspaces: Without loss let us focus on learning the subspace spanned by the columns of U. Our first step in this regard is to compute the unfolding $X_{(1)}$, and to then 'trim' the columns in this unfolding which we may view as dividing each column by a scalar; in other words we post multiply $X_{(1)}$ with an $mn \times mn$ diagonal matrix W_1. The analysis section will specify an idealized trimming procedure; under many conditions we can simply take W_1 to be the identity.

We now compute $\hat{U}$ as the first r left singular vectors of $X_{(1)}W_1$. More precisely, assuming that $X_{(1)}W_1$ admits the singular value decomposition $X_{(1)}W_1 = U_1\Sigma_1 V_1^\top$, we set $\hat{U}$ to be the columns of U_1 corresponding to the r largest singular values. We treat the column space of $\hat{U}$ as our estimate of the column space of U. We denote this entire procedure with the shorthand

$$\hat{U} = \text{svds}(X_{(1)}W_1, r).$$

We apply a similar procedure to learn the column space of V, except this time we use the unfolding $X_{(2)}$, so that $\hat{V} = \text{svds}(X_{(2)}W_2, r)$.

Projection: The second stage is equally simple – we use $\hat{U}$ and $\hat{V}$ to construct our estimate of each slice M^j: our estimate $\hat{M}^j$ is the projection of X^j onto these spaces, i.e.

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

The algorithm above is entirely reminiscent of hard thresholding for matrices, albeit applied to three dimensional tensors which is quite attractive. The entire slice learning algorithm is outlined in **Algorithm 1**. We note again that an 'idealized' value for the inputs W_1 and W_2 will be discussed in the next section, but they can safely be taken as the identity for a broad class of problems.

```
Algorithm 1: Slice Learning
Input  : $X$, $r$, $W_1$, $W_2$
1. $\hat{U} \leftarrow \text{svds}(X_{(1)}W_1, r)$
2. $\hat{V} \leftarrow \text{svds}(X_{(2)}W_2, r)$
3. $\hat{M}^j \leftarrow P_{\hat{U}} X^j P_{\hat{V}}$  $j = 1, \dots, n$
Output : $\hat{M}^j, j = 1, \dots, n$
```

4.1 Practical Considerations

We conclude this section with a few comments regarding practical implementation and computational issues:

Knowing r: The algorithm above takes the value r as input, but in practice this may not be known. Here we anticipate that as opposed to preserving r singular values in steps 1 and 2 of the algorithm, a 'universal' thresholding scheme where we only preserve singular values above a certain easily calculated threshold will work just as effectively. Specifically, by Marčenko and Pastur (1967) we expect that the largest singular values due to noise in either of the unfoldings will be on the order $O(\sqrt{mn})$, whereas if the singular values of $M_{(1)}$ are all within constant order of each other, these singular values will be $O(m\sqrt{n/r})$; a clear separation.

Computation with large tensors: While the ambient dimensions of the input data are large, the data is itself typically quite sparse. The only computationally intensive step in the algorithm above is the computation of a (partial) svd. Specifically, there exist mature linear algebraic algorithms that *sequentially* compute the singular vectors while exploiting data sparsity. An alternative to computing the svd altogether is employing alternating least squares; see Hastie et al. (2015) for a similar approach in matrix completion. Of course, all 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.

5 Recovery Guarantees for Slice Learning

The goal of this section is to provide a recovery guarantee for our slice learning algorithm for tensors with low slice rank. In particular, we are looking for a guarantee that improves upon the naive algorithm of recovering each slice separately. As discussed in Section 3, that naive algorithm achieves SMSE $= O(r/m)$. It is fairly clear that 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 slice with bounded entires, which we take without loss 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. The problem of recovering the first slice is now literally no different than matrix completion 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. Specifically, define the 'learning rate' of M, γ_M according to

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

We will shortly see how γ_M plays the role of a learning rate. 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 , 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 $\gamma_M = \Omega(r/m)$. At the other end of the spectrum, given that the entires 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 is turn implies that $\gamma_M = O(1)$. In summary, the loosest requirement we can reasonably place on γ_M is to require $\gamma_M = \Omega(r/m)$, and the strongest requirement we can place is to require it be a constant.

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 variance of the corresponding entries of ϵ. Specifically, $v^j_{k,l} = \mathsf{E}[(\epsilon^j_{k,l})^2]$. The noise ϵ is said to be* balanced *if the row-norms of $v_{(1)}$ are all equal and the row-norms of $v_{(2)}$ are all equal.*

The assumption above is trivially satisfied in the case of i.i.d. noise. Refinements of our result will allow for weaker versions of this assumption; we can permit a certain amount of discrepancy in the row norms of $v_{(1)}$ and $v_{(2)}$, and allow this to grow with m, n and r. In addition, we will see that in the event that the assumption were violated altogether, an oracle based trimming procedure could compensate for the violation. We are now ready to state our main result.

Theorem 1. *Assume the entries of M lie in $[-1, 1]$. If the entries of ϵ are independent, mean-zero, and $\mathsf{E}[\epsilon^6_{ij}] \le K^6$, and if furthermore ϵ is balanced, then there exists a universal constant c such that for the slice learning algorithm (without trimming),*

$$\mathrm{MSE}(\hat{M}) \le \mathrm{SMSE}(\hat{M}) \le c\left[\frac{K^2 r^2}{m^2} + \frac{K^2(K^4+1)r^2}{\gamma_M^2 mn}\right].$$

We next evaluate this result in light of the minimax guarantee established in Proposition 1, and more generally, our broader goal of slice recovery:

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 the Theorem above 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 1. Of course, this is *substantially* better than the available guarantees for the Naive approach which was $O(r/m)$.

High learning rate: As γ_M grows, so does our ability to learn across slices. Specifically, for $\gamma_M = \Theta(1)$, the theorem guarantees $\mathrm{MSE} \le \mathrm{SMSE} = O(r^2/m^2 + r^2/mn)$. This is within a factor of r off from the lower bound of $\mathrm{MSE} = \Omega(r^2/m^2 + r/mn)$ in Proposition 1. Put a different way, we achieve $\mathrm{SMSE} = O(r^2/m^2)$ with only $n = \Omega(m)$ slices of side information.

Tensor rate: This recovery rate here is for low slice rank tensors, which includes tensors with low Tucker rank and CP rank. We emphasize that the rate $\mathrm{SMSE} = O(r^2/m^2 + r^2/mn)$ 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.

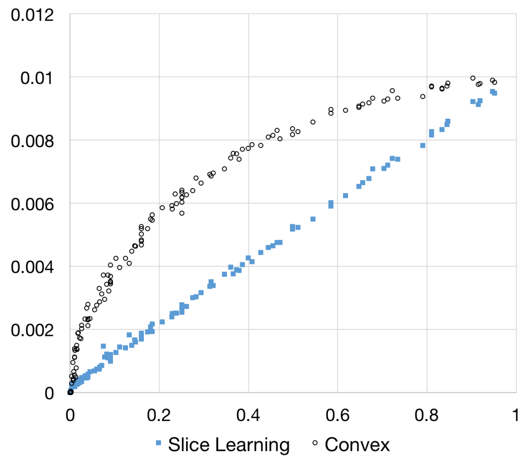


Figure 1: Comparison of slice learning and convex approach for noisy recovery of low Tucker rank tensors. MSE vs. $(r/m)^2$ is plotted for each replication.

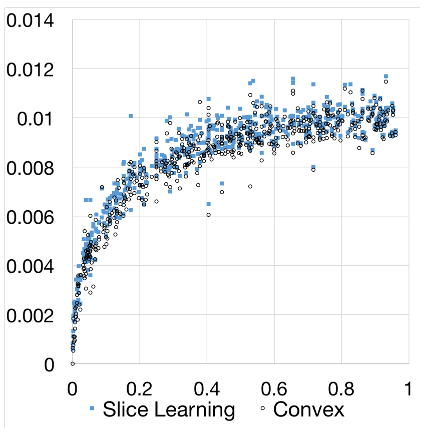
5.1 Unbalanced noise and trimming

Recall from Section 4 that in the first stage of the slice learning algorithm, we can 'trim' the columns in the unfolding $X_{(1)}$, i.e. post multiply $X_{(1)}$ with an $mn \times mn$ diagonal matrix W_1, before taking the singular value decomposition; the trimming procedure can also be done on $X_{(2)}$ with W_2.

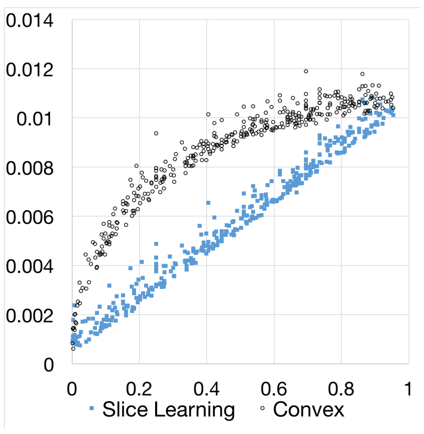
One reason to trim is to 'rebalance' the noise ϵ when it does not satisfy the balanced assumption, which suggests an ideal choice for trimming weights: suppose that we have diagonal matrices W_1 and W_2 such that $\epsilon_{(1)}W_1$ and $\epsilon_{(2)}W_2$ are balanced. Then the result of Theorem 1 effectively still holds (in fact, if the elements of $M_{(1)}W_1$ and $M_{(2)}W_2$ lie in $[-1, 1]$, and $\min_{i=1,2} \sigma_r^2(M_{(i)}W_i) \ge \gamma_M m^2 n/r$, then Theorem 1 remains exactly the same).

6 Experiments

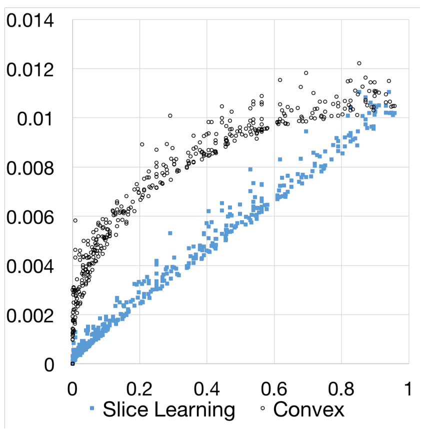
We performed two sets of experiments to evaluate the slice learning algorithm. The first set, using randomly generated tensors, reveals that the slice learning algorithm drastically outperforms convex algorithms by an order of magnitude, even though convex algorithms require drastically greater computation. The second set, using a large real-world dataset, demonstrates the scalability and performance of our approach in practice.



(a) $n = 1$



(b) $n = r$



(c) $n = m$

Figure 2: 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.

6.1 Synthetic Recovery Experiments

Tensor learning and Tucker rank: For our first experiment, we randomly generated m-by-m-by-m tensors of Tucker rank (r, r, r). In each replication, we randomly drew orthonormal m-by-r matrices U, V, and W, along with an r-by-r-by-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 data tensor X was observed with mean-zero gaussian noise of standard deviation 0.1.

We compared the slice learning algorithm against the convex algorithm that minimizes (2) in §3.1. We solved this via the Douglas-Rachford splitting method, as described in 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 must be selected; 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 simulations. In each simulation, m was drawn randomly between 10 and 50, and r was drawn randomly between 1 and m. The MSE of both the slice learning algorithm and (2) are reported in Figure 1, where each simulation is represented by two points representing the MSE of both algorithms, and the MSE's are plotted against the value $(r/m)^2$ for that particular replication.

In Figure 1, the MSE of the slice learning algorithm appears to scale linearly with $(r/m)^2$, as predicted by Theorem 1, while the convex algorithm is sublinear in $(r/m)^2$. 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 singular value decompositions, meaning *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 Liu et al. (2014).

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-by-m-by-n tensors of slice rank r. In each replication, we randomly drew orthonormal m-by-r matrices U and V, along with r-by-r matrices $S^1, \ldots, S^n$ from the standard normal distribution, and we set the slices of the tensor to be $M^j = US^jV^\top$. Gaussian noise of standard deviation 0.1 was used as before.

We used a similar convex algorithm as a benchmark:

$$\operatorname*{argmin}_{Y} \sum_{i=1}^{2} \lambda \left\|Y_{(i)}\right\|_* + \|Y - X\|_F^2 . \tag{4}$$

This convex algorithm was catered to recovering low slice rank tensors, as the mode-3 unfolding is not included in the objective. We solved this with a similar Douglas-Rachford splitting method.

We performed 400 replications, where in each replication, m was drawn randomly between 10 and 50, and r drawn randomly between 1 and m, and finally n was set equal to either 1, r, or m. We measured

Users	Songs	Sparsity	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. Columns 'Users' and 'Songs' show the number of users and songs in each experiment, and 'Sparsity' 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.

the SMSE of the slice learning algorithm and (4). The results for each of the three cases ($n = 1, r, m$) are depicted in three separate plots in Figure 2. Figure 2a 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 1). Figure 2c shows that the slice 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 2b is almost identical to Figure 2c, 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

We also performed experiments using real-world data 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. We represent this data as a tensor with three slices, with binary entries indicating whether that particular user-song interaction occurred during the data's observation period.

We conducted a set of completion experiments: in each experiment, approximately half of the entries (chosen uniformly at random) were observed without noise, and the remaining entries were hidden and meant to be recovered. We compared the slice learning algorithm against two benchmarks. The first benchmark is the *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. Mu et al. (2014) studied such algorithms and demonstrated theoretical guarantees for high-dimensional tensors. 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, which exploits data sparsity.

We treated the experiment as a binary classification task. The results of the experiment in terms of recovering the Collect slice, measured by area under ROC curve (AUC), are summarized in Table 1. We performed experiments on tensors of five different sizes, created by selecting subsets of the densest rows and columns of the original data set. Note that the data is extremely sparse – in the largest tensor, we observe less than a single collect per user.

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 results for recovering the Listen and Download slices can be found in the Appendix.

7 Conclusion

We introduced the problem of slice recovery, with the goal of learning particular slices of a three-dimensional tensor. We proposed the slice learning algorithm, a fast, scalable algorithm that achieves the minimax lower bound for recovery with sufficiently many slices. Experiments supported the performance and scalability of the algorithm.

References

Anandkumar A, Ge R, Hsu D, Kakade SM, Telgarsky M (2014) Tensor decompositions for learning latent variable models. *The Journal of Machine Learning Research* 15(1):2773–2832.

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.

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.

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

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

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

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

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

Hastie T, Mazumder R, Lee JD, Zadeh R (2015) Matrix completion and low-rank svd via fast alternating least squares. *J. Mach. Learn. Res* 16(1):3367–3402.

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

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

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

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.

Mu C, Huang B, Wright J, Goldfarb D (2014) Square deal: Lower bounds and improved relaxations for tensor recovery. *ICML*, 73–81.

Nickel M, Tresp V, Kriegel HP (2011) A three-way model for collective learning on multi-relational data. *Proceedings of the 28th international conference on machine learning (ICML-11)*, 809–816.

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

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.

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.

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

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.

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.