Machine Learning
https://doi.org/10.1007/s10994-023-06338-5

Tensor completion with noisy side information

Dimitris Bertsimas[1] · Colin Pawlowski[1]

Received: 3 February 2020 / Revised: 12 December 2022 / Accepted: 6 April 2023

Abstract
We develop a new model for tensor completion which incorporates noisy side information available on the rows and columns of a 3-dimensional tensor. This method learns a low rank representation of the data along with regression coefficients for the observed noisy features. Given this model, we propose an efficient alternating minimization algorithm to find high-quality solutions that scales to large data sets. Through extensive computational experiments, we demonstrate that this method leads to significant gains in out-of-sample accuracy filling in missing values in both simulated and real-world data. We consider the problem of imputing drug response in three large-scale anti-cancer drug screening data sets: the Genomics of Drug Sensitivity in Cancer (GDSC), the Cancer Cell Line Encyclopedia (CCLE), and the Genentech Cell Line Screening Initiative (GCSI). On imputation tasks with 20% to 80% missing data, we show that the proposed method `TensorGenomic` matches or outperforms state-of-the-art methods including the original tensor model and a multilevel mixed effects model. With 80% missing data, `TensorGenomic` improves the R^2 from 0.404 to 0.552 in the GDSC data set, 0.407 to 0.524 in the CCLE data set, and 0.331 to 0.453 in the GCSI data set compared to the tensor model which does not take into account genomic side information.

Keywords Tensor completion · Low-rank · 3-Dimensional data · Anti-cancer drug screens · Genomic data

1 Introduction

Mathematically, a tensor is a multidimensional array of numbers, typically with 3 or more dimensions (Kolda & Bader, 2009). A vector is a 1-dimensional tensor, a matrix is a 2-dimensional tensor, and in general there are N-dimensional tensors. For example,

Editor: Pradeep Ravikumar.

Dimitris Bertsimas
dbertim@mit.edu

Colin Pawlowski
cpawlows@mit.edu

[1] Operations Research Center, E40-111, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

suppose that we are given an e-commerce data set of n customers interacting with m products through ℓ interactions. These interactions may include things such as: “searched for the product”, “purchased the product”, and “clicked on advertisement for the product”. We can represent this data as a 3-dimensional tensor $\mathbf{Z} \in \mathbb{R}^{n \times m \times \ell}$, where $z_{ij}^{k} = 1$ if interaction k occurred for the pair (customer i, item j) and $z_{ij}^{k} = 0$ otherwise. This tensor may contain a large number of missing values, for instance because we have not shown advertisements to each (customer i, item j) pair. However, we assume that all of the entries in the tensor have some true underlying values which we would like to predict. This representation is useful because the data naturally varies along each dimension according to a different mechanism, which is the principal structure that is leveraged by mathematical models based on tensor data.

Given this tensor representation, we consider the problem of filling in the missing values of this tensor. This is known as the problem of *tensor completion*. In the e-commerce example, we would like to predict the purchase probability for each pair (customer i, item j) so that we can display personalized advertisements and search recommendations. However, in order to develop the most accurate predictive model, in many cases it is insufficient to consider the tensor data in isolation because we have additional data available. Suppose that we are given additional data on the customers $\mathbf{X} \in \mathbb{R}^{n \times p}$ and additional data on the products $\mathbf{Y} \in \mathbb{R}^{m \times q}$ which are completely known. We refer to this additional data as *side information*. In practice, this side information may be *noisy*, which means that it contains only limited predictive power for the learning task at hand. Therefore, we will avoid making any strong assumptions about the relationships between $\mathbf{X}$, $\mathbf{Y}$, and $\mathbf{Z}$. In this work, we propose a model which leverages all of this data simultaneously to fill in the missing values of $\mathbf{Z}$.

As a real world application explored in this paper, we consider the problem of personalized chemotherapy treatment for patients with cancer. Since data from human clinical trials is sparse in this area, we use data from large-scale anti-cancer drug screens, including the Genomics of Drug Sensitivity in Cancer (GDSC), the Cancer Cell Line Encyclopedia (CCLE), and the Genentech Cell Line Screening Initiative (GCSI) data sets. These data sets were generated from *in vitro* experiments on *cell lines*, which are samples of cells that have been taken from the tumors of patients with cancer and grown in the lab (Shoemaker, 2006). Suppose that we are given a data set with n patients, m anti-cancer drugs, and ℓ doses. We can represent this data set as a tensor $\mathbf{Z} \in \mathbb{R}^{n \times m \times \ell}$, where z_{ij}^{k} is the percentage reduction in tumor size after the cell line from patient i receives anti-cancer drug j at dose k. In addition, we may also be given noisy side information in the form of genomic features $\mathbf{X} \in \mathbb{R}^{n \times p}$ for the patients and drug target pathway features $\mathbf{Y} \in \mathbb{R}^{m \times q}$ for the anti-cancer drugs. Our goal is to fill in the missing values in the tensor $\mathbf{Z}$ so that we can prescribe the best anti-cancer treatment for each individual. While a lot of research has been done on this subject, accurately predicting the response of an individual to anti-cancer drugs remains a crucial challenge (Azuaje, 2016). Furthermore, to our knowledge very little work has been done trying to predict the response at particular doses. In computational experiments in Sect. 4, we test tensor completion methods on the GDSC, CCLE, and GCSI data sets, and we compare our approach to existing methods for this application.

1.1 Related work

Our work belongs to the class of statistical methods known as *collaborative filtering* algorithms (Koren et al., 2009; Koren & Bell, 2015). The objective of collaborative filtering is to learn the preferences of an individual by collecting taste information from many individuals (Candès & Tao, 2009). For instance, in the previous two examples we were interested in learning the product preferences of consumers and the drug preferences of cancer patients, respectively. Further, in this work we focus on the problem setting with *explicit feedback*, where we have direct input from users regarding their preferences, in contrast with the implicit feedback setting where direct user input is not available (Hu et al., 2008). Collaborative filtering methods designed for the explicit feedback setting include algorithms for matrix completion and tensor completion, and typically use matrix factorization methods (Koren et al., 2009).

There is extensive literature on the problem of matrix completion, with a surge in interest starting in 2006 with the Netflix Prize competition (Bennett et al., 2007). In this competition, the internet movie-streaming company Netflix asked participants to come up with a recommendation system that accurately predicts movie ratings of users, with a \$1 million dollar first prize. The winning entry used a matrix factorization approach with modifications (Bell & Koren, 2007). Over the past decade, matrix completion methods have been used in many ratings-based recommendation systems in e-commerce (Yang et al., 2016; Kluver et al., 2018).

Matrix factorization methods for matrix completion use the assumption that the underlying data is *low rank*. Intuitively, this means that the data matrix has a simpler structure than an arbitrary matrix with the same dimensions. Formally, the rank of a matrix $\mathbf{M} \in \mathbb{R}^{n \times m}$ is the smallest integer r such that it can be expressed as the product of two matrices $\mathbf{U}\mathbf{V}^T$, where $\mathbf{U} \in \mathbb{R}^{n \times r}$ and $\mathbf{V} \in \mathbb{R}^{m \times r}$. Computationally efficient methods are available for learning low rank matrix approximations, including nuclear-norm minimization, singular-value decomposition, and alternating minimization. These approaches are widely used for solving the problem of matrix completion (Candès & Recht, 2009; Candes & Plan, 2010; Cai et al., 2010; Mazumder et al., 2010; Jain et al., 2013).

In addition, matrix completion methods that incorporate side information have been studied. For example, Inductive Matrix Completion (IMC) is a method for matrix completion with exact side information (Jain & Dhillon, 2013). In this model, we assume that the data matrix $\mathbf{M} \in \mathbb{R}^{n \times m}$ can be expressed as the product of $\mathbf{X}\mathbf{S}\mathbf{Y}^T$, where $\mathbf{X} \in \mathbb{R}^{n \times p}$, $\mathbf{Y} \in \mathbb{R}^{m \times q}$ are the matrices of side information and $\mathbf{S} \in \mathbb{R}^{p \times q}$ is learned from the data. Alternatively, Chiang et al. (2015) proposed a method given noisy side information which uses weaker assumptions on the data structure. In their model, they assume that the data matrix can be expressed as $\mathbf{X}\mathbf{S}\mathbf{Y}^T + \mathbf{R}$, where $\mathbf{X}$, $\mathbf{S}$, $\mathbf{Y}$ are defined as before and $\mathbf{R} \in \mathbb{R}^{n \times m}$ is a low rank component learned from the data. In our approach, we use a similar additive model to incorporate noisy side information, but extended to the tensor setting. Finally, we note that there are several other methods, including Kernelized Bayesian Matrix Factorization which integrates side information using Bayesian priors (Gönen et al., 2013), and extensions of IMC which impose sparsity constraints upon the $\mathbf{S}$ matrix (Lu et al., 2016; Bertsimas & Li, 2018).

In order to extend the ideas from matrix completion to tensor completion, the first thing required is a generalization of the concept of matrix rank to higher dimensions. There are multiple definitions for the rank of a tensor with 3 or more dimensions, including the CP rank, Tucker rank, and Slice rank (Kolda & Bader, 2009; Tucker, 1966; Farias & Li, 2019). Some of these objects such as the Tucker rank have multiple components based upon the number of dimensions in the tensor. Similar to the matrix rank, if a tensor $\mathbf{Z} \in \mathbb{R}^{n \times m \times \ell}$ has low rank according to one of these criteria, then it has a simpler structure than an arbitrary tensor with the same dimensions. We discuss the mathematical properties of tensors in more detail in Sect. 2.1, and we provide the formal definitions of these concepts of tensor rank in Appendix 1.

Previous work has been done to find the best low rank approximation to a tensor for different definitions of the tensor rank. The CP and Tucker decompositions are well-known (Kolda & Bader, 2009). However, these methods are computationally intensive and impractical for large-scale data sets. Several promising results in recent years have focused on finding the best convex tensor approximation by minimizing the sum or a convex combination of the components of the Tucker rank (Gandy et al., 2011; Liu et al., 2013). These algorithms have recovery guarantees and generalize better than exact Tucker decomposition when the number of observed entries is greater than a certain threshold (Tomioka et al., 2011).

Newer approaches which use non-convex approaches have been shown to outperform convex methods for tensor completion. Chen et al. (2019) propose a non-convex projected gradient descent method which bounds the Tucker rank and imposes sparsity on the tensor approximation. In addition, Farias and Li (2019) propose a non-convex method for 3-dimensional tensor completion which provides stronger statistical guarantees compared to general methods for n-dimensional tensors. Their proposed algorithm learns a low Slice rank representation of the data via a hard-thresholding SVD approach which can scale to large data sets. In this paper, we also restrict our focus to the 3-dimensional tensor completion problem, and we extend the method proposed by Farias and Li (2019) to accommodate noisy side information on the rows and the columns of the tensor.

There has also been some previous work adapting tensor completion methods to accommodate side information (Acar et al., 2011; Narita et al., 2012; Rai et al., 2015; Lamba et al., 2016; Zhou et al., 2017; Wimalawarne et al., 2018). For example, Narita et al. (2012) propose a method that finds the best CP or Tucker approximation with an additional regularization term based on graph Laplacians to incorporate the side information. This method is slightly less computationally efficient compared to the original CP and Tucker decomposition algorithms. Rai et al. (2015) propose a completely Bayesian model that enriches the original CP decomposition with a second-layer tensor decomposition that incorporates side information. However, this method requires many tuning parameters, and in practice we may not have the distributional information which is required for the Bayesian priors. In general, current methods for tensor completion which account for side information are computationally intensive and difficult to implement for problems encountered in practice. In particular, there is a practical need to impute missing values in 3-dimensional data sets used for medical applications, which may include genomic side information on the patients which are noisy and high-dimensional.

Finally, we outline the literature which is related to the real-world application that we consider. Many collaborative filtering methods have been applied to predict gene-disease associations and drug response. For example, IMC and probability-based collaborative filtering have been used to discover gene-disease associations in the Online Mendelian Inheritance in Man (OMIM) database (Hamosh et al., 2005; Natarajan & Dhillon, 2014; Zeng et al., 2017). In addition, several methods have been developed specifically to predict anti-cancer drug response in the GDSC and CCLE data sets. For example, Tan (2016) extend Kernelized Bayesian Matrix Factorization to predict anti-cancer drug response in the GDSC data set leveraging drug pathway data as side information. Liu et al. (2018) propose a nearest-neighbors based method which incorporates genomic and drug side information into a low rank model. Other methods which do not rely upon low rank models have also been developed to predict anti-cancer drug response, including random forest, deep learning, and network-based methods (Rahman & Pal, 2019; Su et al., 2019; Franco et al., 2019).

Our approach for predicting anti-cancer drug response differs from the above methods because we train on the raw experimental data from the drug screens, which is the drug response of cell lines from patients with solid tumor cancers to anti-cancer treatments at particular doses. As a result, the training data is a 3-dimensional tensor with dimensions (patient, drug, dose). In contrast, the previous methods rely upon a pre-processing step to determine the sensitivity of each (patient, drug) pair first. These sensitivity values are taken as ground truth, and then matrix completion methods are fit on top. In this work, we avoid the dependence upon intermediate models by casting this as a tensor completion problem instead of a matrix completion problem.

1.2 Contributions

The contributions of this paper are as follows:

1. We propose an extension to the low rank model for tensor completion proposed by Farias and Li (2019) that leverages noisy side information. In particular, we propose a model for tensor completion with one-sided information that incorporates noisy features of the rows, and a model for tensor completion with two-sided information that incorporates noisy features of both the rows and columns. Each model is composed of a low rank component which leverages the structure of the observed values in the tensor and a regression component which leverages the noisy side information.
2. For each model, we derive fast algorithms based upon alternating minimization which find high quality solutions. In particular, we present the algorithms `TensorOneSided` and `TensorTwoSided` for tensor completion given noisy one-sided and two-sided information, respectively.
3. In experiments on simulated data, we demonstrate that the proposed method `TensorTwoSided` significantly outperforms benchmark methods for tensor completion given two-sided information with varying levels of noise. The benchmark methods considered include the original tensor completion method `Tensor` which does not incorporate side information and a regression method `TwoSided` which uses side information only.
4. In experiments on real-world data, we demonstrate that the proposed method `TensorGenomic` matches or outperforms state-of-the-art methods for predicting anti-cancer drug response in the GDSC, CCLE, and GCSI data sets with 20% to 80% missing values given genomic side information. In particular, with 80% missing data, `TensorGenomic` improves the R^2 from 0.404 to 0.552 in the GDSC data set, 0.407 to 0.524 in the CCLE data set, and 0.331 to 0.453 in the GCSI data set compared to the low rank tensor model which does not take into account genomic side information.

The structure of this paper is as follows. In Sect. 2, we describe our proposed methods for tensor completion for problems with noisy one-sided and two-sided information. In Sect. 3, we compare the performance of our methods against benchmark tensor completion methods on simulated data experiments. In Sect. 4, we test the performance of the method for tensor completion on two real-world examples predicting anti-cancer drug response with genomic side information. In Sect. 5, we discuss the results from the simulated and real-world computational experiments. We conclude in Sect. 6.

2 Methods

In Sect. 2.1, we provide some background material on tensors which is prerequisite material for this work. In Sect. 2.2, we state the problem of tensor completion with noisy side information. In Sects. 2.3 and 2.4, we introduce two basic regression models for tensor completion using one-sided and two-sided information, and we present two fast methods based upon accelerated gradient descent. In Sect. 2.5, we introduce a low rank model for tensor completion without side information, and we review the Slice Learning method. In Sect. 2.6, we introduce a low rank model for tensor completion with one-sided information that uses features on the rows, and we present the method `TensorOneSided`. In Sect. 2.7, we introduce a low rank model for tensor completion with two-sided information that uses features on both rows and columns, and we present the method `TensorTwoSided`.

2.1 Background on tensors

In this section, we cover a few preliminaries on tensors and the notation that we use to describe them. A tensor is a multidimensional array or N-way array (Kolda & Bader, 2009). In this work, we consider only 3-way tensors in the Euclidean space $\mathbb{R}^{n\times m\times \ell}$. We refer to n, m, and ℓ as the number of rows, columns, and slices of the tensor, respectively. For a given tensor $\mathbf{Z} \in \mathbb{R}^{n\times m\times \ell}$, let z_{ij}^k be the element in the ith row, jth column, and kth slice of the tensor. In addition, we refer to the matrix formed by the kth slice of the tensor as $\mathbf{Z}^k \in \mathbb{R}^{n\times m}$. If $\mathbf{Z}$ has missing values, we denote the known and missing entries in the kth slice of the tensor as

$$\Omega_k = \left\{(i,j)\ :\ z_{ij}^k \text{ is known}\right\},$$
$$\Omega_k^c = \left\{(i,j)\ :\ z_{ij}^k \text{ is missing}\right\}.$$

and across all slices of the tensor as

$$\Omega = \left\{(i,j,k)\ :\ z_{ij}^k \text{ is known}\right\},$$
$$\Omega^c = \left\{(i,j,k)\ :\ z_{ij}^k \text{ is missing}\right\}.$$

We also describe some basic notation that we use to refer to matrices. For a matrix $\mathbf{X} \in \mathbb{R}^{n\times p}$, let x_{id} be the element in the ith row and dth column. We denote the transpose of $\mathbf{X}$ as $\mathbf{X}^T \in \mathbb{R}^{p\times n}$ and the column rank as $\text{rank}(\mathbf{X})$. We also will use the Frobenius norm of a matrix, which is defined as

$$\|\mathbf{X}\|_F := \sqrt{\sum_{i,d} x_{id}^2}.$$

2.1.1 Tensor unfoldings

Next, we define the *unfoldings* of a tensor $\mathbf{Z} \in \mathbb{R}^{n\times m\times \ell}$. We define the mode-1 unfolding of a tensor to be the horizontal concatenation of the slices of the tensor into a single matrix. We denote this matrix as $\mathbf{Z}_{(1)} \in \mathbb{R}^{n \times m\ell}$. In $\mathbf{Z}_{(1)}$, the columns are equivalent to the columns of all of the slices in the original tensor. Likewise, we define the mode-2 unfolding of a tensor to be the horizontal concatenation of the transposed slices into a single matrix. We denote the mode-2 unfolding as $\mathbf{Z}_{(2)} \in \mathbb{R}^{m \times n\ell}$. In $\mathbf{Z}_{(2)}$, the columns are equivalent to the rows of all of the slices in the original tensor. Following this pattern, we can also define the mode-3 unfolding of a tensor, although forming this object requires breaking up the tensor slices. We consider the vector $\mathbf{z}_{ij}$ formed by fixing the ith row and jth column, and then varying the third dimension. The mode-3 unfolding $\mathbf{Z}_{(3)} \in \mathbb{R}^{\ell \times nm}$ is the matrix formed by horizontally concatenating all of these vectors $\mathbf{z}_{ij}$. In Fig. 1, we provide visualizations of a 3-dimensional tensor and its mode-1 and mode-2 unfoldings. For more discussion on tensor unfoldings, we refer the reader to Kolda and Bader (2009). In Appendix 1, we provide definitions for the rank of a tensor which use these concepts of tensor unfoldings.

2.2 Tensor completion problem

In this section, we state the problem of tensor completion with noisy side information.

Suppose that we are given a 3-dimensional data set $\mathbf{Z} \in \mathbb{R}^{n \times m \times \ell}$ with known and missing values Ω_k, Ω_k^c for each slice k, respectively. In addition, suppose that we are also given noisy features $\mathbf{X} \in \mathbb{R}^{n \times p}$ and $\mathbf{Y} \in \mathbb{R}^{m \times q}$ of the rows and columns of this data, respectively.

For example, consider an e-commerce data set with n customers, m products, and ℓ interactions. In this tensor $\mathbf{Z}$, $z_{ij}^k = 1$ if customer i interacted with product j via interaction k, and $z_{ij}^k = 0$ otherwise. Many of the entries of $\mathbf{Z}$ are missing because each customer typically has only a few interactions with a subset of the products. In addition, we have p features of the customers, such as the age, gender, location, and electronic device of each customer, which are captured in the row side information $\mathbf{X}$. We also have q features of the products, such as the brand, supplier, and average review score of each product, which are captured in the column side information $\mathbf{Y}$.

As another example, consider a drug screening data set with n patients, m drugs, and ℓ doses. In this tensor, z_{ij}^k is the outcome when patient i is treated with drug j at dose k. Many of the entries of $\mathbf{Z}$ are missing because each patient has only received a few (drug, dose) treatment combinations in the past. In the row side information $\mathbf{X}$, we have p features of the patients, which may include demographic or genomic data. In the column side information $\mathbf{Y}$, we have q features of the drugs, which may include drug target pathway data.

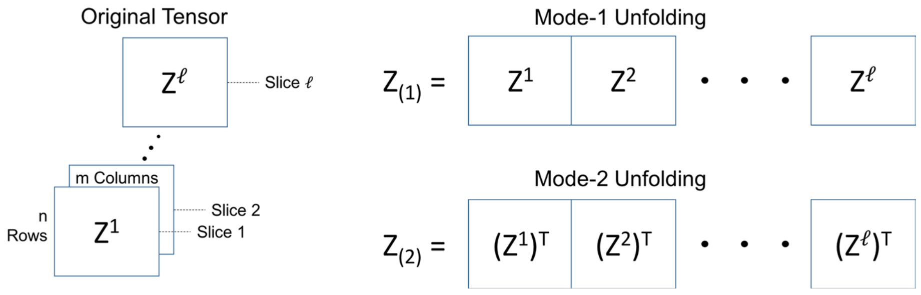

Fig. 1 Visualizations of a 3-dimensional tensor and its mode-1 and mode-2 unfoldings

Given the observed values of $\mathbf{Z}$ and the noisy features $\mathbf{X}, \mathbf{Y}$, our goal is to find the approximation $\hat{\mathbf{Z}} \in \mathbb{R}^{n \times m \times \ell}$ which is as close as possible to the original $\mathbf{Z}$. In particular, we would like to find $\hat{\mathbf{Z}}$ which minimizes the sum-of-squared errors across all of the slices:

$$\sum_{k=1}^{\ell} \|\mathbf{Z}^k - \hat{\mathbf{Z}}^k\|_F^2. \tag{1}$$

In order to find $\hat{\mathbf{Z}}$ which minimizes (1), we consider the following problem:

$$\begin{aligned} \min_{\hat{\mathbf{Z}}} \quad & \sum_{k=1}^{\ell} \sum_{(i,j) \in \Omega_k} (z_{ij}^k - \hat{z}_{ij}^k)^2 \\ \text{s.t.} \quad & \hat{\mathbf{Z}} \in \mathcal{Z}, \end{aligned} \tag{2}$$

where $\hat{\mathbf{Z}} \in \mathcal{Z}$ denotes a set of structural assumptions on $\hat{\mathbf{Z}}$. In the next few sections, we consider models based on a few different structural assumptions which are summarized in Table 1.

2.3 One-sided regression model

In this section, we introduce the one-sided regression model which uses row side information only, and we present the `OneSided` method. This is a simple model based upon linear regression which does not leverage information across multiple slices of the tensor.

Suppose that $\mathbf{x}_i \in \mathbb{R}^p$ is the vector of features for the ith row in the tensor. Consider the linear model:

$$\hat{z}_{ij}^k = \mathbf{x}_i^T \mathbf{w}_{:j}^k,$$

where $\mathbf{w}_{:j}^k \in \mathbb{R}^p$ are weights for a particular (column, slice) pair. We can interpret the weight w_{dj}^k as the amount of change in the prediction given one unit increase in x_{id}. In tensor notation, we have

$$\hat{\mathbf{Z}}^k = \mathbf{X}\mathbf{W}^k, \tag{3}$$

where $\mathbf{W}^k \in \mathbb{R}^{p \times m}$ is the matrix of weights for the kth slice. We can learn $\mathbf{W}$ by running $m\ell$ independent linear regressions, one for each (column, slice) pair. We can fit all of these linear regression models simultaneously by considering the following optimization problem:

Table 1 Structural assumptions on the slices of the tensor approximation. In these models, $\mathbf{W}^k, \mathbf{S}^k$ are weights which are different for each slice of the tensor. On the other hand, $\mathbf{U}, \mathbf{V}$ are latent features which are constant across all slices. More details are provided in Sects. 2.3-2.7

Model	Structural Assumption	Section
One-Sided Regression	$\hat{\mathbf{Z}}^k = \mathbf{X}\mathbf{W}^k$	2.3
Two-Sided Regression	$\hat{\mathbf{Z}}^k = \mathbf{X}\mathbf{W}^k\mathbf{Y}^T$	2.4
Tensor	$\hat{\mathbf{Z}}^k = \mathbf{U}\mathbf{S}^k\mathbf{V}^T$	2.5
Tensor One-Sided	$\hat{\mathbf{Z}}^k = \mathbf{U}\mathbf{S}^k\mathbf{V}^T + \mathbf{X}\mathbf{W}^k$	2.6
Tensor Two-Sided	$\hat{\mathbf{Z}}^k = \mathbf{U}\mathbf{S}^k\mathbf{V}^T + \mathbf{X}\mathbf{W}^k\mathbf{Y}^T$	2.7

$$\min_{\mathbf{W}} \sum_{k=1}^{\ell} \sum_{(i,j)\in\Omega_k} \left(z_{ij}^k - (\mathbf{X}\mathbf{W}^k)_{ij}\right)^2 + \frac{1}{\gamma}\|\mathbf{W}^k\|_F^2, \tag{4}$$

where γ is a regularization parameter. Problem (4) is a quadratic optimization problem which is efficiently solvable. In particular, we solve this subproblem using Nesterov's accelerated gradient descent method (Nesterov, 1983). Let $f(\mathbf{W};\mathbf{X},\Omega)$ be the objective function of problem (4). The partial derivative of f with respect to w_{dj}^k is

$$\frac{\partial f(\mathbf{W};\mathbf{X},\Omega)}{\partial w_{dj}^k} = \frac{2}{\gamma} w_{dj}^k + \sum_{i:(i,j)\in\Omega_k} 2x_{id}(\mathbf{x}_i^T \mathbf{w}_{:j}^k - z_{ij}^k).$$

Let $\nabla f(\mathbf{W};\mathbf{X},\Omega)$ be the full gradient of f with respect to $\mathbf{W}$. In Algorithm 1, we present an accelerated gradient descent method for solving problem (4) using this gradient. We can further speed up this method by selecting the step size ν dynamically at each step via backtracking line search (Nocedal & Wright, 2006).

Algorithm 1 `OneSided`

```
Data: Tensor $\mathbf{Z} \in \mathbb{R}^{n\times m\times \ell}$ with known entries
    $\Omega = \{(i,j,k) : z_{ij}^k \text{ is known}\}$,
    side information $\mathbf{X} \in \mathbb{R}^{n\times p}$.
Input: Warm start $\mathbf{W}_0 \in \mathbb{R}^{p\times m}$,
    regularization parameter $\gamma > 0$,
    max number of gradient steps $G \geq 1$, step size $\nu > 0$.
Output: Optimal solution to problem (4).
Procedure:
    Initialize $\mathbf{W}_1 \leftarrow \mathbf{W}_0$, $t \leftarrow 1$.
    while $t < G + 1$ do
        $\overline{\mathbf{W}} \leftarrow \mathbf{W}_t + \frac{t-1}{t+2}(\mathbf{W}_t - \mathbf{W}_{t-1})$,
        $\mathbf{W}_{t+1} \leftarrow \overline{\mathbf{W}} - \nu\nabla f(\overline{\mathbf{W}};\mathbf{X},\Omega)$,
        $t \leftarrow t + 1$.
    end while
    return Estimated value of $\mathbf{W}$.
```

2.4 Two-sided regression model

In this section, we introduce the two-sided regression model which uses both row and column side information, and we present the `TwoSided` method. Similar to the one-sided model, this model does not leverage information across multiple slices of the tensor.

Suppose that $\mathbf{x}_i \in \mathbb{R}^p$ is the vector of features for the ith row, and $\mathbf{y}_j \in \mathbb{R}^q$ is the vector of features for the jth column. Consider the bilinear model:

$$\hat{z}_{ij}^k = \mathbf{x}_i^T \mathbf{W}^k \mathbf{y}_j,$$

where $\mathbf{W}^k \in \mathbb{R}^{p\times q}$ are weights for the kth slice of the tensor. We can interpret the weight w_{de}^k as the amount of change in the prediction given one unit increase in the interaction term $x_{id}y_{je}$. In tensor notation we have

$$\hat{\mathbf{Z}}^k = \mathbf{X}\mathbf{W}^k\mathbf{Y}$$

for each slice $k = 1, \dots, \ell$. In order to find the weights $\mathbf{W}$, we consider the following optimization problem:

$$\min_{\mathbf{W}} \sum_{k=1}^{\ell} \sum_{(i,j)\in\Omega_k} \left(z_{ij}^k - (\mathbf{X}\mathbf{W}^k\mathbf{Y})_{ij}\right)^2 + \frac{1}{\gamma}\|\mathbf{W}^k\|_F^2, \tag{5}$$

where γ is a regularization parameter. This is a quadratic optimization problem nearly identical to the one-sided regression formulation. If $\mathbf{Y}$ is the $m \times m$ identity matrix, then these two formulations are equivalent. Let $g(\mathbf{W};\mathbf{X},\mathbf{Y},\Omega)$ be the objective function of problem (5), and let $\nabla g(\mathbf{W};\mathbf{X},\mathbf{Y},\Omega)$ be the gradient of g with respect to $\mathbf{W}$. In Algorithm 2, we present an accelerated gradient descent method for solving problem (5) using this gradient.

```
Algorithm 2 TwoSided
Data: Tensor $\mathbf{Z} \in \mathbb{R}^{n\times m\times \ell}$ with known entries
    $\Omega = \{(i,j,k) : z_{ij}^k$ is known$\}$,
    side information $\mathbf{X} \in \mathbb{R}^{n\times p}$, $\mathbf{Y} \in \mathbb{R}^{m\times q}$.
Input: Warm start $\mathbf{W}_0 \in \mathbb{R}^{p\times q}$,
    regularization parameter $\gamma > 0$,
    max number of gradient steps $G \geq 1$, step size $\nu > 0$.
Output: Optimal solution to problem (5).
Procedure:
    Initialize $\mathbf{W}_1 \leftarrow \mathbf{W}_0$, $t \leftarrow 1$.
    while $t < G + 1$ do
        $\overline{\mathbf{W}} \leftarrow \mathbf{W}_t + \frac{t-1}{t+2}(\mathbf{W}_t - \mathbf{W}_{t-1})$,
        $\mathbf{W}_{t+1} \leftarrow \overline{\mathbf{W}} - \nu\nabla g(\overline{\mathbf{W}};\mathbf{X},\mathbf{Y},\Omega)$,
        $t \leftarrow t + 1$.
    end while
    return  Estimated value of $\mathbf{W}$.
```

2.5 Basic tensor model

In this section, we introduce a low rank model for tensor completion without side information, and we present the `Tensor` method. This low rank model is equivalent to the one originally proposed by Farias and Li (2019).

There are two shortcomings of the one-sided and two-sided regression models that we have presented so far. First of all, these models do not leverage information across multiple slices of the tensor to impute the missing values. Second, because the observed row and column features are noisy, they are typically poor predictors for the tensor on their own. For example, in a drug screening data set, genomic features are typically poor predictors of drug response on their own. The `Tensor` model that we present next addresses these issues.

Instead of trying to improve the noisy row and column features, we will try to learn new features from scratch using only the observed values in the tensor. In the `Tensor` model, we suppose that there are a few true underlying *latent features* of the rows and columns which are constant across all slices of the tensor. We assume that there are at most r latent features for the rows and at most r latent features for the columns, and all of these latent features are unknown *a priori*. This is known as a *low rank* assumption, which is commonly used for collaborative filtering methods (Koren et al., 2009; Koren & Bell, 2015).

Let $\mathbf{u}_i \in \mathbb{R}^r$ be the latent features of row i and let $\mathbf{v}_j \in \mathbb{R}^r$ be the latent features of observation j. Given these latent features, the generative model for z_{ij}^k is

$$\hat{z}_{ij}^k = \mathbf{u}_i^T \mathbf{S}^k \mathbf{v}_j, \tag{6}$$

where $\mathbf{S}^k \in \mathbb{R}^{r\times r}$ is a matrix of fitted coefficients. Let $\mathbf{U} \in \mathbb{R}^{n\times r}$ be the matrix of row latent features and let $\mathbf{V} \in \mathbb{R}^{m\times r}$ be the matrix of column latent features. It follows that the model for the kth slice of the tensor is

$$\hat{\mathbf{Z}}^k = \mathbf{U}\mathbf{S}^k\mathbf{V}^T. \tag{7}$$

For different slices, the latent features $\mathbf{U}$ and $\mathbf{V}$ remain the same, but the fitted coefficients $\mathbf{S}^k$ are different. This structural assumption is equivalent to requiring that the Slice rank of $\hat{\mathbf{Z}}$ is at most r. In Fig. 2, we show a diagram of this tensor model for the drug screening data set.

To find $\mathbf{U}, \mathbf{S}, \mathbf{V}$, we consider the following optimization problem:

$$\min_{\mathbf{U},\mathbf{S},\mathbf{V}} \sum_{k=1}^{\ell} \sum_{(i,j)\in\Omega_k} \left(z_{ij}^k - (\mathbf{U}\mathbf{S}^k\mathbf{V}^T)_{ij} \right)^2. \tag{8}$$

Note that this formulation requires a single parameter, the tensor rank r, which we will learn via cross-validation.

Unlike the previous one-sided and two-sided regression formulations, problem (8) is non-convex, so we cannot compute the global optimal solution. However, we can find high-quality solutions via nonconvex methods. In particular, we can use an iterative procedure based upon the Slice Learning algorithm proposed by Farias and Li (2019) to find high-quality solutions. In this procedure, we begin with a warm start solution $\hat{\mathbf{Z}}$. Each iteration, we run the Slice Learning algorithm and update $\hat{\mathbf{Z}}$ in the missing entries Ω^c of the original tensor. We repeat until the tensor approximation $\hat{\mathbf{Z}}$ converges to a stationary point, or the maximum iteration limit is reached.

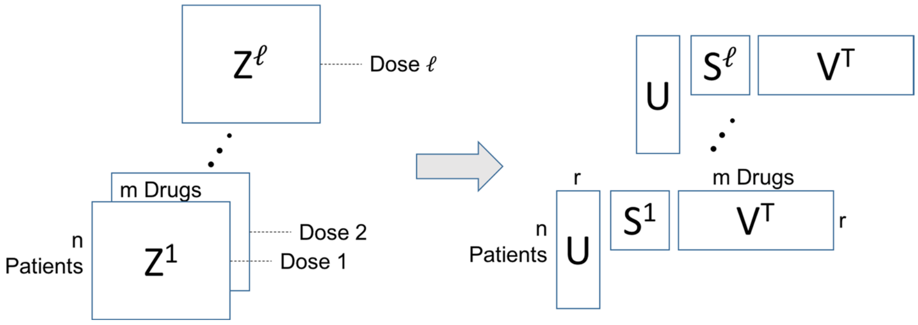

Fig. 2 Tensor model of a drug screening data set. n is the number of patients, m is the number of drugs, ℓ is the number of doses tested, and r is the number of latent features

In a single iteration, we run the Slice Learning algorithm to obtain updated estimates for $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{S}^1, \dots, \mathbf{S}^\ell$. First, we estimate the latent features of the rows $\mathbf{U}$ by taking the singular value decomposition (SVD) of the mode-1 unfolding of $\hat{\mathbf{Z}}$. Let $\mathbf{U}_1 \boldsymbol{\Sigma}_1 \mathbf{V}_1^T$ be the SVD of $\hat{\mathbf{Z}}_{(1)}$, where $\mathbf{U}_1$, $\mathbf{V}_1$ are orthonormal and $\boldsymbol{\Sigma}_1$ is diagonal. We set $\mathbf{U}$ to be the r columns of $\mathbf{U}_1$ which correspond to the top r singular values. We denote this operation as:

$$\mathbf{U} \leftarrow \text{svds}(\hat{\mathbf{Z}}_{(1)}, r).$$

Similarly, we estimate the latent features of the columns $\mathbf{V}$ by taking the SVD of the mode-2 unfolding of $\hat{\mathbf{Z}}$. The update for $\mathbf{V}$ is

$$\mathbf{V} \leftarrow \text{svds}(\hat{\mathbf{Z}}_{(2)}, r).$$

Finally, we update the estimates for $\mathbf{S}^1, \dots, \mathbf{S}^\ell$. Since $\mathbf{U}$, $\mathbf{V}$ are orthonormal, we have $\mathbf{U}^{-1} = \mathbf{U}^T$ and $\mathbf{V}^{-1} = \mathbf{V}^T$. Therefore the update for $\mathbf{S}^k$ which minimizes the squared error for slice k is

$$\mathbf{S}^k \leftarrow \mathbf{U}^T \hat{\mathbf{Z}}^k \mathbf{V}.$$

In Algorithm 3, we summarize this method for tensor completion without side information. In the next two sections, we see how this method can be modified to incorporated side information on the rows and/or columns.

Algorithm 3 `Tensor`

```
Data: Tensor $\mathbf{Z} \in \mathbb{R}^{n\times m\times \ell}$ with missing entries
  $\Omega^c = \{(i,j,k) : z_{ij}^k \text{ is missing}\}$.
Input: Rank $r$, warm start $\mathbf{Z}_0 \in \mathbb{R}^{n\times m\times \ell}$,
  max number of iterations $T \geq 1$.
Output: Locally optimal solution to problem (8).
Procedure:
  Initialize $\hat{\mathbf{Z}} \leftarrow \mathbf{Z}_0$, $t \leftarrow 0$.
  while $t < T$ do
    $\mathbf{R} \leftarrow \hat{\mathbf{Z}}$,
    $\mathbf{U} \leftarrow \text{svds}(\mathbf{R}_{(1)}, r)$,
    $\mathbf{V} \leftarrow \text{svds}(\mathbf{R}_{(2)}, r)$,
    $\mathbf{S}^k \leftarrow \mathbf{U}^T\mathbf{R}^k\mathbf{V}$          $\forall k$,
    $\hat{z}_{ij}^k \leftarrow (\mathbf{U}\mathbf{S}^k\mathbf{V}^T)_{ij}$   $\forall (i,j,k) \in \Omega^c$,
    $t \leftarrow t + 1$.
  end while
  return Estimated values of $\mathbf{U}, \mathbf{S}, \mathbf{V}$.
```

2.6 Tensor model with noisy one-sided information

In this section, we introduce a low rank model for tensor completion given noisy one-sided information, and we present the method `TensorOneSided`. This model combines components from the `Tensor` and `OneSided` models.

In this approach, we model the tensor as the sum of two components, with one component that we learn from the Slice learning decomposition and one component that we learn from the row side information. Let $\mathbf{x}_i$ be the observed features of row i. The resulting generative model for z_{ij}^k is

$$\hat{z}_{ij}^k = \mathbf{u}_i^T \mathbf{S}^k \mathbf{v}_j + \mathbf{x}_i^T \mathbf{w}_{:j}^k \tag{9}$$

where $\mathbf{u}_i$ are latent features of row i, $\mathbf{v}_j$ are latent features of row j, $\mathbf{S}^k$ are weights of the latent features for slice k, and $\mathbf{w}_{:j}^k$ are (column, slice)-specific weights. It follows that the model for the kth slice of the tensor is

$$\hat{\mathbf{Z}}^k = \mathbf{U}\mathbf{S}^k\mathbf{V}^T + \mathbf{X}\mathbf{W}^k, \tag{10}$$

where $\mathbf{U} \in \mathbb{R}^{n\times r}$, $\mathbf{V} \in \mathbb{R}^{m\times r}$, $\mathbf{S}^1, \dots, \mathbf{S}^\ell \in \mathbb{R}^{r\times r}$, and $\mathbf{W}^1, \dots, \mathbf{W}^\ell \in \mathbb{R}^{p\times m}$ are learned from the data. We can interpret model (10) as the basic tensor model (7) with an additional term to predict the residuals that is linear with respect to the observed row features. Note that if $\mathbf{W} = \mathbf{0}$, then this model reduces to the the basic tensor model exactly. This is important because in some cases the side information may not provide any additional predictive power. Similarly, if any of $\mathbf{U}$, $\mathbf{S}$, $\mathbf{V}$ are equal to zero, then this reduces to the regression model (4) using row side information only. In Fig. 3, we show a diagram of this tensor model with one-sided information for a drug screening data set.

To find $\mathbf{W}$, $\mathbf{U}$, $\mathbf{S}$, $\mathbf{V}$, we consider the following optimization problem:

$$\min_{\mathbf{W},\mathbf{U},\mathbf{S},\mathbf{V}} \sum_{k=1}^{\ell} \sum_{(i,j)\in\Omega_k} \left(z_{ij}^k - (\mathbf{U}\mathbf{S}^k\mathbf{V}^T + \mathbf{X}\mathbf{W}^k)_{ij} \right)^2 + \frac{1}{\gamma} \|\mathbf{W}^k\|_F^2, \tag{11}$$

where γ is a regularization parameter. This formulation uses two parameters γ and r which we can learn via cross-validation. Taking the limit as $\gamma \to 0$, this model reduces to the original tensor formulation (8).

We propose the following alternating optimization procedure to solve problem (11), which we refer to as `TensorOneSided`. In this approach, we alternate between running the Slice Learning algorithm and solving a quadratic optimization problem.

1. Begin with a warm start solution $\hat{\mathbf{Z}}$. Initialize all of the variables $\mathbf{W}, \mathbf{U}, \mathbf{S}, \mathbf{V}$ to zero.
2. Update $\mathbf{U}, \mathbf{S}, \mathbf{V}$ by considering the following problem:

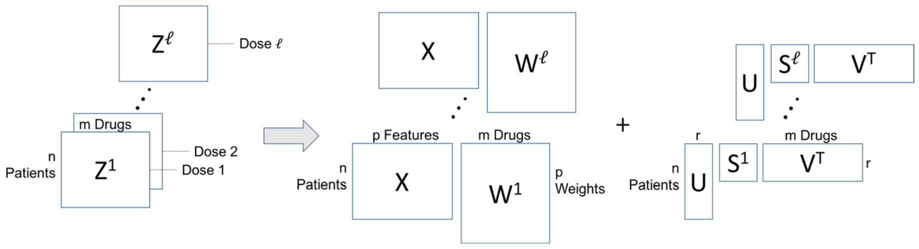

Fig. 3 Tensor model of a drug screening data set with noisy side information for the patients only. n is the number of patients, m is the number of drugs, ℓ is the number of doses tested, r is the number of latent features, and p is the number of observed patient features

$$\min_{\mathbf{U},\mathbf{S},\mathbf{V}} \sum_{k=1}^{\ell} \sum_{(i,j)\in\Omega_k} \left((\hat{\mathbf{Z}}^k - \mathbf{X}\mathbf{W}^k)_{ij} - (\mathbf{U}\mathbf{S}^k\mathbf{V}^T)_{ij} \right)^2. \tag{12}$$

We can find high-quality solutions to this problem using the Slice Learning algorithm (Farias & Li, 2019). Let $\mathbf{R}$ be the tensor of residuals, where $\mathbf{R}^k = \hat{\mathbf{Z}}^k - \mathbf{X}\mathbf{W}^k$. In this step, we find a low rank tensor approximation to $\mathbf{R}$ by taking SVDs of the mode-1 and mode-2 unfoldings.

3. Update the $\mathbf{W}$ by considering the following problem:

$$\min_{\mathbf{W}} \sum_{k=1}^{\ell} \sum_{(i,j)\in\Omega_k} \left((\hat{\mathbf{Z}}^k - \mathbf{U}\mathbf{S}^k\mathbf{V}^T)_{ij} - (\mathbf{X}\mathbf{W}^k)_{ij} \right)^2 + \frac{1}{\gamma}\|\mathbf{W}^k\|_F^2. \tag{13}$$

This is a quadratic optimization problem, so it is efficiently solvable via gradient descent. Let $\mathbf{R}$ be the tensor of residuals, where $\mathbf{R}^k = \hat{\mathbf{Z}}^k - \mathbf{U}\mathbf{S}^k\mathbf{V}$. Given a warm start solution $\mathbf{W}_0$, maximum number of gradient steps G, and step size $\nu > 0$, we denote this update compactly as

$$\mathbf{W} \leftarrow \texttt{OneSided}(\mathbf{R}, \Omega, \mathbf{X}, \mathbf{W}_0, \gamma, G, \nu),$$

which is detailed in Algorithm 1.

4. Iterate until the variables $\mathbf{W}, \mathbf{U}, \mathbf{S}, \mathbf{V}$ converge, or the maximum iteration limit is reached.

We express the steps of the complete algorithm `TensorOneSided` in Algorithm 4.

Algorithm 4 `TensorOneSided`

```
Data: Tensor $\mathbf{Z} \in \mathbb{R}^{n\times m\times \ell}$, with
    known entries $\Omega = \{(i,j,k) : z_{ij}^k \text{ is known}\}$,
    missing entries $\Omega^c = \{(i,j,k) : z_{ij}^k \text{ is missing}\}$,
    and side information $\mathbf{X} \in \mathbb{R}^{n\times p}$.
Input: Rank $r$, warm start $\mathbf{Z}_0 \in \mathbb{R}^{n\times m\times \ell}$,
    max number of iterations $T \geq 1$,
    regularization parameter $\gamma > 0$,
    max number of gradient steps $G \geq 1$, step size $\nu > 0$.
Output: Locally optimal solution to problem (11).
Procedure:
    Initialize $\hat{\mathbf{Z}} \leftarrow \mathbf{Z}_0$, $\mathbf{W} \leftarrow \mathbf{0}$, $t \leftarrow 0$.
    while $t < T$ do
        $\mathbf{R}^k \leftarrow \hat{\mathbf{Z}}^k - \mathbf{X}\mathbf{W}^k \quad \forall k$,
        $\mathbf{U} \leftarrow \text{svds}(\mathbf{R}_{(1)}, r)$,
        $\mathbf{V} \leftarrow \text{svds}(\mathbf{R}_{(2)}, r)$,
        $\mathbf{S}^k \leftarrow \mathbf{U}^T\mathbf{R}^k\mathbf{V} \quad \forall k$,
        $\mathbf{R}^k \leftarrow \hat{\mathbf{Z}}^k - \mathbf{U}\mathbf{S}^k\mathbf{V}^T \quad \forall k$,
        $\mathbf{W} \leftarrow \texttt{OneSided}(\mathbf{R}, \Omega, \mathbf{X}, \mathbf{W}, \gamma, G, \nu)$,
        $\hat{z}_{ij}^k \leftarrow (\mathbf{U}\mathbf{S}^k\mathbf{V}^T + \mathbf{X}\mathbf{W}^k)_{ij} \quad \forall (i,j,k) \in \Omega^c$,
        $t \leftarrow t + 1$.
    end while
    return Estimated values of $\mathbf{W}, \mathbf{U}, \mathbf{S}, \mathbf{V}$.
```

2.7 Tensor model with noisy two-sided information

In this section, we introduce a low rank model for tensor completion given noisy two-sided information, and we present the method `TensorTwoSided`. This model combines components from the `Tensor` and `TwoSided` models.

Let $\mathbf{x}_i$ be the features of row i, and let $\mathbf{y}_j$ be the features of column j. The generative model for z_{ij}^k is

$$\hat{z}_{ij}^k = \mathbf{u}_i^T \mathbf{S}^k \mathbf{v}_j + \mathbf{x}_i^T \mathbf{W}^k \mathbf{y}_j, \tag{14}$$

where $\mathbf{u}_i$ are latent features of row i, $\mathbf{v}_j$ are latent features of row j, $\mathbf{S}^k$ are weights of the latent features for slice k, and $\mathbf{W}^k$ are weights of the observed features for slice k. It follows that the model for the kth slice of the tensor is

$$\hat{\mathbf{Z}}^k = \mathbf{U}\mathbf{S}^k\mathbf{V}^T + \mathbf{X}\mathbf{W}^k\mathbf{Y}^T, \tag{15}$$

where $\mathbf{U} \in \mathbb{R}^{n\times r}$, $\mathbf{V} \in \mathbb{R}^{m\times r}$, $\mathbf{S}^1, \dots, \mathbf{S}^\ell \in \mathbb{R}^{r\times r}$, and $\mathbf{W}^1, \dots, \mathbf{W}^\ell \in \mathbb{R}^{p\times q}$ are learned from the data. In Fig. 4, we show a diagram of this tensor model with two-sided information for a drug screening data set.

To find $\mathbf{W}$, $\mathbf{U}$, $\mathbf{S}$, $\mathbf{V}$, we consider the following optimization problem:

$$\min_{\mathbf{W},\mathbf{U},\mathbf{S},\mathbf{V}} \sum_{k=1}^{\ell} \sum_{(i,j)\in\Omega_k} \left(z_{ij}^k - (\mathbf{U}\mathbf{S}^k\mathbf{V}^T + \mathbf{X}\mathbf{W}^k\mathbf{Y}^T)_{ij}\right)^2 + \frac{1}{\gamma}\|\mathbf{W}^k\|_F^2, \tag{16}$$

where γ is a regularization parameter. This formulation uses two parameters γ and r which we can learn via cross-validation. Instead of the Frobenius norm, it is also reasonable to consider a nuclear norm penalty on each matrix of coefficients $\mathbf{W}^k$, or add the constraint that $\mathbf{W}^k$ is low rank. We consider the Frobenius norm here because this formulation is very close to formulation (11) so we can use a similar solution method.

In Appendix 2, we provide details for an alternating optimization procedure to solve problem (16), which we refer to as the `TensorTwoSided` algorithm. This algorithm is identical to the `TensorOneSided` algorithm except for the update of $\mathbf{W}$ in Step 3.

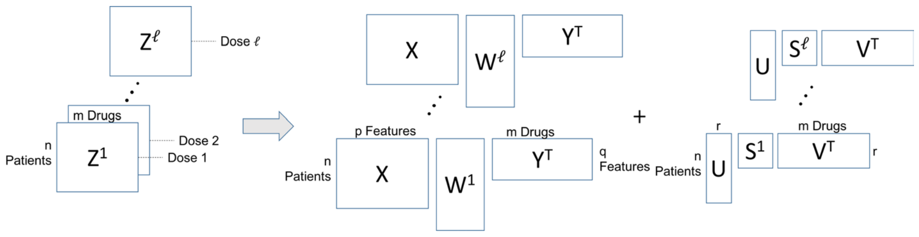

Fig. 4 Tensor model of a drug screening data set with noisy side information for both the patients and drugs. n is the number of patients, m is the number of drugs, ℓ is the number of doses tested, r is the number of latent features, p is the number of observed patient features, and q is the number of observed drug features

3 Simulated data experiments

In this section, we present computational experiments testing the proposed methods for tensor completion on simulated data. In Sect. 3.1, we describe the generation process for the simulated data sets. In Sect. 3.2, we present the experimental setup and the methods which are compared. In Sect. 3.3, we present the results from all of the simulated data experiments.

3.1 Simulated data sets

For this set of experiments, we generate complete tensors $\mathbf{Z} \in \mathbb{R}^{200\times200\times10}$ with low Slice rank. In particular, we suppose that:

$$\mathbf{Z}^k = \mathbf{U}\mathbf{S}^k\mathbf{V}^T, \quad \forall k = 1, \dots, 10,$$

where:

- $\mathbf{U} \in \mathbb{R}^{200\times20}$: ground truth latent features of the rows,
- $\mathbf{S}^k \in \mathbb{R}^{20\times20}$: ground truth weights for the kth slice,
- $\mathbf{V} \in \mathbb{R}^{200\times20}$: ground truth latent features of the columns.

We suppose that all of the entries of $\mathbf{U}, \mathbf{S}^1, \dots, \mathbf{S}^k, \mathbf{V}$ are independently identically distributed $\mathcal{N}(0, 1)$. In addition, we suppose that the matrices of row and column side information are given by:

$$\mathbf{X} = \mathbf{U} + \epsilon^1,$$
$$\mathbf{Y} = \mathbf{V} + \epsilon^2,$$

where:

- $\epsilon^1 \in \mathbb{R}^{200\times20}$: random noise for the row features,
- $\epsilon^2 \in \mathbb{R}^{200\times20}$: random noise for the column features.

We suppose that all of the entries of ϵ^1, ϵ^2 are independently identically distributed $\mathcal{N}(0, \sigma)$, where $\sigma \geq 0$ is the standard deviation of the feature noise which we will vary.

3.2 Experimental setup

In these experiments, we impute missing values in tensors of the form $\mathbf{Z} \in \mathbb{R}^{200\times200\times10}$ described in the previous section. For this task, we suppose that we are given the observed values of $\mathbf{Z}$ and side information $\mathbf{X} \in \mathbb{R}^{200\times20}, \mathbf{Y} \in \mathbb{R}^{200\times20}$ for some level of noise $\sigma \geq 0$. In each experiment, we randomly select 80% of the values in the tensor to be missing completely at random (MCAR). We then compare a variety of methods for predicting these missing values in the tensor, including:

1. **Tensor**: Implements the `Tensor` method given in Algorithm 3 to impute the missing values via the Slice Learning method (Farias & Li, 2019). This method learns a low rank representation of the 3-dimensional data, including latent features for the rows and

columns which are constant across all of the slices. Uses cross-validation to select the tensor rank r.

2. **Two-Sided**: Implements the `TwoSided` method given in Algorithm 2 to impute the missing values for each slice independently via an ℓ_2-regularized bilinear regression model. Uses interaction terms between observed features of the rows and columns as features in the model. Uses cross-validation to select the regularization parameter γ.
3. **Tensor Two-Sided**: Implements the `TensorTwoSided` method given in Algorithm 4 which incorporates both observed and latent features of the rows and columns. Uses cross-validation to select the tensor rank r with the weights of the side information $\mathbf{W} = \mathbf{0}$ fixed. Then, with the optimal value of r fixed, uses cross-validation to select the regularization parameter γ.

For each of the above methods, we tune the tensor rank r over the range $\{1, 2, \dots, 20\}$, and we tune the regularization parameter γ over the range $\{0.1, 0.01, \dots, 10^{-10}\}$. We evaluate the out-of-sample accuracy of each method and compare against a baseline which predicts the mean value of each tensor slice. For each method and missing data scenario, we compute the out-of-sample R^2 value on each slice, and then take the average of the out-of-sample R^2 values across all of the slices. We repeat all of the experiments 5 times varying the random seed which generates the ground truth tensor $\mathbf{Z} \in \mathbb{R}^{200\times200\times10}$ and the missing data scenarios.

3.3 Results

In this section, we present the results from the experiments on simulated data.

In Fig. 5, we plot the imputation accuracy of the tensor completion methods as we vary the standard deviation of the noise added to the side information. Across all levels of noise

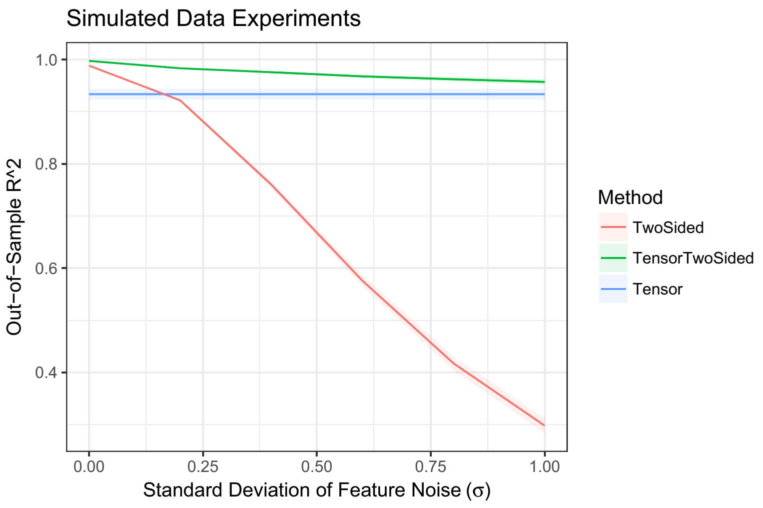

Fig. 5 Imputation accuracy for the simulated data experiments with 80% missing data, varying the standard deviation of the normally distributed feature noise

considered, the `TensorTwoSided` method significantly improves upon the next best method. As the level of noise increases, the performance of both `TensorTwoSided` and `TwoSided` decreases, while the performance of `Tensor` remains constant. At the highest noise level $\sigma = 1$, the average out-of-sample R^2 values were 0.957, 0.933, and 0.298 for the `TensorTwoSided`, `Tensor`, and `TwoSided` methods, respectively. This demonstrates that the proposed method `TensorTwoSided` can improve upon the baseline `Tensor` method even when the side information is only weakly predictive.

On the other hand, with no noise added ($\sigma = 0$), the average out-of-sample R^2 values were 0.997, 0.933, and 0.988 for the `TensorTwoSided`, `Tensor`, and `TwoSided` methods, respectively. This demonstrates that the proposed method `TensorTwoSided` can improve upon the baseline `TwoSided` regression method even when the row and column features are known exactly. Overall, these results show that the proposed method `TensorTwoSided` outperforms the best of the `Tensor` and `TwoSided` methods across all noise levels considered.

4 Real-world data experiments

In this section, we present computational experiments testing the proposed methods for tensor completion on two large-scale anti-cancer drug screens. In Sects. 4.1 and 4.2, we describe the Genomics of Drug Sensitivity in Cancer (GDSC) and the Cancer Cell Line Encyclopedia (CCLE) data sets. In Sect. 4.4, we present the experimental setup and the methods which are compared. In Sect. 4.5, we present the results from all of the real-world data experiments.

4.1 Genomics of drug sensitivity in cancer

The first anti-cancer drug screening data set that we consider is the Genomics of Drug Sensitivity in Cancer (GDSC) data set (Yang et al., 2012). We are given data $\mathbf{Z} \in \mathbb{R}^{n\times m\times \ell}$ from experiments applying anti-cancer drugs to patients, where $n = 955$, $m = 265$, and $\ell = 12$ are the numbers of patients, drugs, and doses, respectively. For each drug j, the ℓth dose corresponds the maximum concentration at which drug j was administered, and the kth dose is 1/2 times the concentration of the $(k + 1)$th dose for $k = 1, \dots, (\ell - 1)$. In addition, we have genomic data $\mathbf{X} \in \mathbb{R}^{n\times p}$ where $p = 2,004$ is the number of genomic features. These features include mutation, gain-loss, and whole exome sequence information for the oncogenes identified in the Catalogue of Somatic Mutations in Cancer (COSMIC) data set (Forbes et al., 2016), as well as tissue type and cancer classification according to The Cancer Genome Atlas (TGCA) groupings (Weinstein et al., 2013).

4.2 Cancer Cell Line Encyclopedia data set

We also consider the Cancer Cell Line Encyclopedia (CCLE) anti-cancer drug screening data set (Barretina et al., 2012). In this data set, we are given data $\mathbf{Z} \in \mathbb{R}^{n\times m\times \ell}$ from experiments applying anti-cancer drugs to patients, where $n = 461$, $m = 24$, and $\ell = 8$ are the numbers of patients, drugs, and doses, respectively. For all drugs, the ℓth dose corresponds the maximum concentration of 8μM, and the kth dose is approximately

1/3.2 times the concentration of the $(k+1)$th dose for $k = 1, \dots, (\ell - 1)$. In addition, we have genomic data $\mathbf{X} \in \mathbb{R}^{n \times p}$ where $p = 2,036$ is the number of genomic features. These features include copy number variation, mutation, and RNA expression data for the oncogenes identified in the COSMIC data set (Forbes et al., 2016).

4.3 Genentech Cell Line Screening Initiative data set

The third anti-cancer drug screening data set that we consider is the Genentech Cell Line Screening Initiative (GCSI) data set (Haverty et al., 2016). This includes data $\mathbf{Z} \in \mathbb{R}^{n \times m \times \ell}$ from experiments applying anti-cancer drugs to patients, where $n = 126$, $m = 16$, and $\ell = 9$ are the numbers of patients, drugs, and doses, respectively. For each drug j, the ℓth dose corresponds the maximum concentration at which drug j was administered, and the kth dose is 1/3 times the concentration of the $(k+1)$th dose for $k = 1, \dots, (\ell - 1)$. In this data set, the cell lines are a subset of the cell lines in the CCLE data set, and the genomic features available for these cell lines is the same. In particular, we have genomic data $\mathbf{X} \in \mathbb{R}^{n \times p}$ derived from the COSMIC data set (Forbes et al., 2016), where $p = 2,036$ is the number of genomic features. This data set was accessed via the PharmacoDB online database (version 1.1.1) of anticancer drug screens (Smirnov et al., 2017).

4.4 Experimental setup

In these experiments, we impute missing values in tensors of drug sensitivity values from the GDSC (Yang et al., 2012) and CCLE (Barretina et al., 2012) data sets. For each dose, we ignore the already missing values and hide an additional 20%, 40%, 60%, or 80% of the observed values to be the test set. We then compare a variety of methods for predicting these missing values in the tensor, including:

1. **Piecewise Linear**: Uses linear interpolation to fill in each missing (patient, drug, dose) response using the (patient, drug) responses that are available at the higher and lower doses. For (patient, drug) pairs with zero observations, this method imputes the mean of the drug response at that dose. This is a fast method that we use as a warm start for the other methods which require one.
2. **Non-Linear Mixed Effects (NLME)**: Uses a multilevel mixed effects model to simultaneously fit two-parameter sigmoidal dose response curves for all (patient, drug) pairs (Vis et al., 2016). For each sigmoidal curve, the two free parameters are assumed to be normally distributed about the mean values for the entire data set. Uses the Piecewise Linear imputation as a warm start.
3. **Matrix**: Fills in the missing values for each dose independently with matrix completion via SoftImpute (Mazumder et al., 2010). Uses the Piecewise Linear imputation as a warm start and cross-validation to select the optimal matrix rank, which may be different for each slice of the tensor.
4. **Tensor**: Implements the `Tensor` method given in Algorithm 3 to impute the missing values via the Slice Learning method (Farias & Li, 2019). This method learns a low rank representation of the 3-dimensional data, including latent features for the patients and drugs which are constant across all of the doses. Uses the Piecewise Linear imputation as a warm start and cross-validation to select the tensor rank r.

5. **Genomic**: Implements the `OneSided` method given in Algorithm 1 to impute the missing values for each dose independently via an ℓ_2-regularized regression model. Uses genomic features of the patients as the row side information. Uses cross-validation to select the regularization parameter γ.
6. **Tensor Genomic**: Implements the `TensorOneSided` method given in Algorithm 4 which incorporates both genomic features of the patients and latent features of the patients and drugs. Uses cross-validation to select the tensor rank r with the weights of the side information $\mathbf{W} = \mathbf{0}$ fixed. Then, with the optimal value of r fixed, uses cross-validation to select the regularization parameter γ. Uses the Piecewise Linear imputation as a warm start.
7. **XGBoost**: Uses the gradient boosting algorithm (Chen & Guestrin, 2016), implemented with the software package `xgboost` in R. Random grid search with 10 iterations is used to select the optimal parameters from the following parameter ranges: (1) booster: 'gblinear', 'gbtree', (2) max_depth: [lower bound = 3, upper bound = 10], (3) min_child_weight: [lower bound = 1, upper bound = 10], (4) subsample: [lower bound = 0.5, upper bound = 1.0], (5) colsample_bytree: [lower bound = 0.5, upper bound = 1.0], (6) eta: [0.01, 0.05, 0.1, 0.2, 0.3]. Once the above parameters are set, 5-fold cross-validation is used to select the optimal "nrounds" parameter value from 1 to 200 using the built-in function `xgb.cv`. Independent predictive models are fit for each of the drugs in the data set, and the features used are: dose, patient identifier, and genomic features for the patients. Dose is treated as a numeric variable, and patient identifier is treated as a categorical variable.

In the `Matrix` method, we tune the matrix ranks over the range $\{1, 2, \ldots, 20\}$ for the CCLE and GDSC data sets, and over the range $\{1, 2, \ldots, 10\}$ for the GCSI data set. For the `Tensor` and `TensorOneSided` methods, we tune the tensor rank r over the ranges $\{10, 20, \ldots, 120\}$ for the GDSC data set, $\{1, 2, \ldots, 20\}$ for the CCLE data set, and $\{1, 2, \ldots, 16\}$ for the GCSI data set. For the `TensorOneSided` method, we tune the regularization parameter γ over the range $\{0.1, 0.01, \ldots, 10^{-10}\}$.

We evaluate the out-of-sample accuracy of each method and compare against a baseline which predicts the mean value of each tensor slice. For each method and missing data scenario, we compute the out-of-sample R^2 value on each slice, and then take the average of the out-of-sample R^2 values across all of the slices. We repeat all of the experiments 5 times varying the random seed which generates the missing data scenarios.

In addition, we perform computational speed experiments recording the time and memory usage of each algorithm on the CCLE data set with 20%, 40%, 60%, and 80% missing data. We perform these experiments on a single core of a MacBook Pro (OS X El Capitan, Version 10.11.6) with a 3.1GHz Intel Core i7 processor and 16 GB 1867 MHz DDR3 memory drive. We repeat the computational speed experiments 5 times varying the random seed which generates the missing data scenarios.

4.5 Results

In this section, we present the results from the real-world experiments on the anti-cancer drug screening data sets.

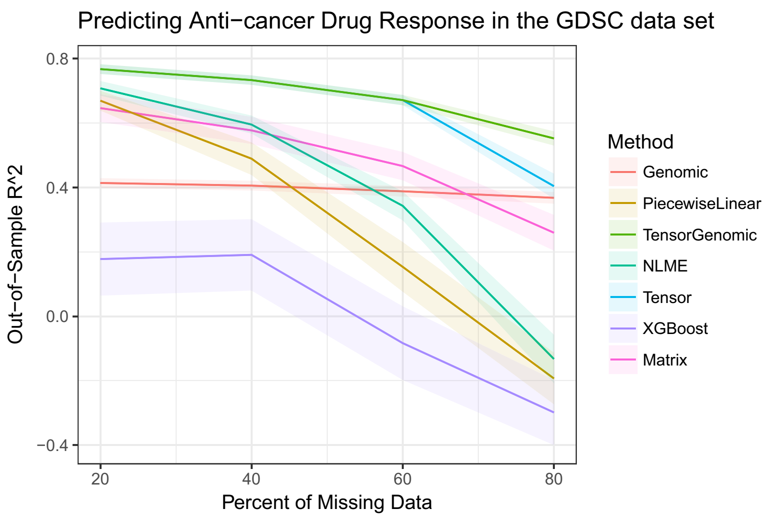

Fig. 6 Imputation accuracy on the GDSC data set varying the percentage of missing data from 20 to 80%

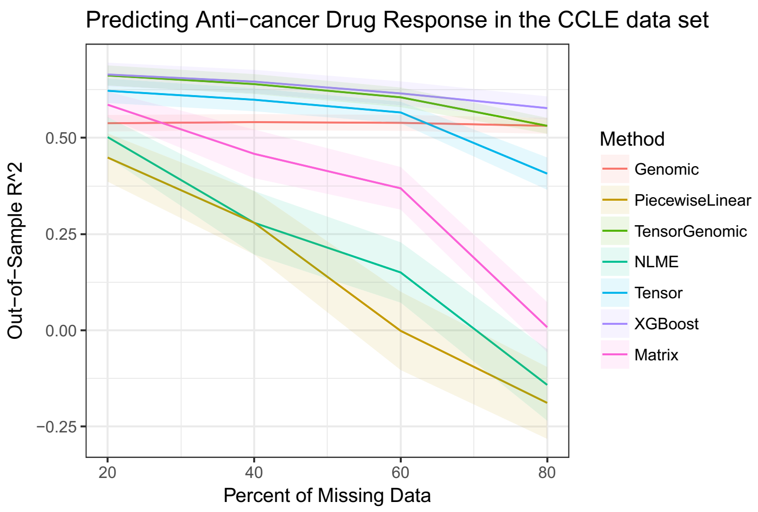

Fig. 7 Imputation accuracy on the CCLE data set varying the percentage of missing data from 20 to 80%

In Figs. 6, 7, and 8 we show the average out-of-sample R^2 for each method on the GDSC, CCLE, and GCSI data sets under different missing scenarios. For both sets of experiments, we see that the `TensorGenomic` method performs best in all missing percentages. As the percentage of missing data increases, the relative improvement over the `Tensor` method increases, while the relative improvement over the `Genomic` method decreases. This makes sense because the fully known side information becomes more important as the percentage of missing data in the tensor increases.

On the GDSC data set, we see that `Tensor` and `TensorGenomic` are equally the best methods when there is 20–60% missing data, and `TensorGenomic` outperforms

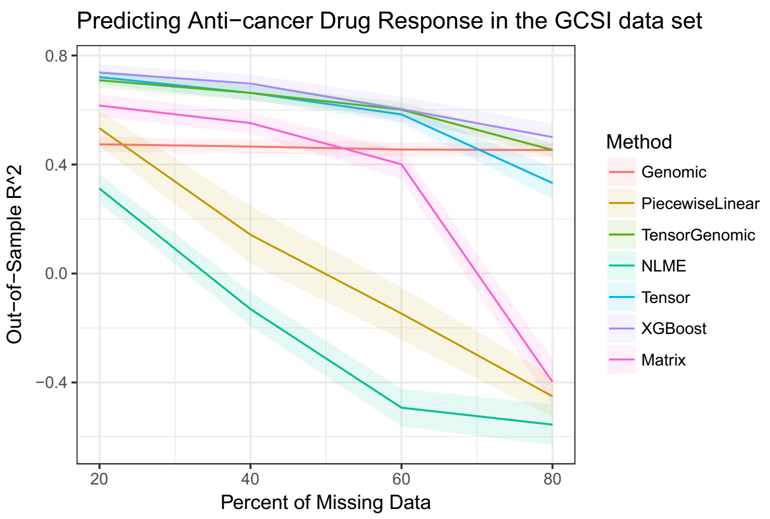

Fig. 8 Imputation accuracy on the GCSI data set varying the percentage of missing data from 20 to 80%

both `Tensor` and `Genomic` when there is 80% missing data. At low missing percentages, the third best method is `NLME`, which is the mixed-effects model to fit sigmoidal dose response curves that has been used in recent publications on the GDSC data set (Vis et al., 2016; Iorio et al., 2016). However, at high missing percentages, the performance of the `NLME` method tails off considerably and its R^2 even turns negative. In contrast, the `TensorGenomic`, `Tensor`, `Genomic`, and `Matrix` methods all maintain R^2 values of 0.25 or greater. This indicates that matrix factorization-based and regression-based models can add value over current parametric models for fitting dose response curves, especially in scenarios with lots of missing data. Finally, we observe that the `XGBoost` method, which does not take into account the tensor information, underperforms the other methods.

On the CCLE data set, we observe that `XGBoost` is the best overall method, followed closely by `TensorGenomic`. In particular, the out-of-sample R^2 are closely overlapping for the missing percentages 20–60%, and `XGBoost` has a slight edge at 80% missing data. We also observe that `Genomic` has a strong performance across the board, matching the accuracy of the `TensorGenomic` method at the 80% missing data level. This suggests that the genomic features that we selected in the CCLE data set are more predictive than the genomic features that we selected in the GDSC data set. As in the previous data set, the `NLME` method declines in performance rapidly as the percent of missing data increases, and is significantly outperformed by matrix factorization-based and regression-based models with 80% missing data.

On the GCSI data set, we observe similar trends as in the `CCLE` data set. `XGBoost` is the best overall method, followed closely by `TensorGenomic` and then `Tensor`. The accuracy of the `Genomic` method remains relatively constant across all missing percentages, and matches the accuracy of the `TensorGenomic` method with 80% missing data. The `Matrix` and `PiecewiseLinear` methods perform reasonably well with 20% missing data, but their performance rapidly declines as the percentage of missing data increases. The `NLME` performs poorly across the board for all missing percentages on this data set.

We also present the tensor ranks which were selected during cross-validation for the tensor-based methods in Figs. 9, 10, and 11 in Appendix 3. Since we select *r* first during the cross-validation procedure for `TensorGenomic`, the rank parameters selected by both `Tensor` and `TensorGenomic` are the same in each experiment. In both data sets, the average tensor rank selected decreases as the percentage of missing data increases. In addition, the average tensor rank selected is much higher in the GDSC experiments than in the CCLE or GCSI experiments, because the GDSC data set is much larger. This shows that we can fit more complicated tensor models (e.g. models with higher tensor ranks) when more data is available.

In Table 2, we show the results from the computational speed experiments. For each method, we include the time and memory costs for hyperparameter tuning required for the method as well. We observe that the methods `Matrix`, `PiecewiseLinear`, `Sigmoid`, and `Tensor` were the fastest algorithms with average runtimes under 1 min for all missing percentages. The next fastest algorithms were `Genomic` and `TensorGenomic` which had average runtimes under 20 min for all missing percentages. The slowest method was `XGBoost`, which had an average runtime ranging from 1.3 h to 4.1 h across

Table 2 Time and memory usage for each algorithm on the CCLE data set

Method	Missing %	Time (seconds)	Total memory usage (GB)
Genomic	20	177.7 (6.4)	371.0 (9.7)
Genomic	40	136.2 (3.7)	237.2 (6.1)
Genomic	60	130.9 (2.9)	184.8 (1.9)
Genomic	80	107.4 (1.4)	102.8 (1.6)
Matrix	20	7.8 (0.2)	0.4 (0.0)
Matrix	40	10.5 (1.3)	0.5 (0.0)
Matrix	60	9.7 (0.7)	0.5 (0.0)
Matrix	80	7.3 (0.4)	0.4 (0.0)
PiecewiseLinear	20	0.2 (0.1)	0.3 (0.0)
PiecewiseLinear	40	0.3 (0.1)	0.4 (0.0)
PiecewiseLinear	60	0.3 (0.1)	0.4 (0.0)
PiecewiseLinear	80	0.2 (0.1)	0.3 (0.0)
NLME	20	42.8 (3.1)	1.7 (0.0)
NLME	40	43.1 (1.6)	1.7 (0.0)
NLME	60	43.0 (1.0)	1.7 (0.0)
NLME	80	43.9 (1.5)	1.7 (0.0)
Tensor	20	10.5 (1.6)	3.5 (0.0)
Tensor	40	9.9 (1.3)	3.9 (0.0)
Tensor	60	9.8 (1.2)	4.2 (0.0)
Tensor	80	10.7 (2.7)	4.3 (0.0)
TensorGenomic	20	992.0 (5.9)	2024.1 (9.7)
TensorGenomic	40	554.7 (9.1)	1009.3 (9.7)
TensorGenomic	60	748.5 (15.1)	1229.6 (9.7)
TensorGenomic	80	604.2 (7.4)	831.9 (9.7)
XGBoost	20	14652.3 (505.7)	67.6 (0.0)
XGBoost	40	11542.9 (1090.0)	58.0 (0.0)
XGBoost	60	8283.9 (415.3)	48.3 (0.0)
XGBoost	80	4814.7 (287.0)	38.5 (0.0)

the different missing percentages. In particular, we note that the average runtimes for `XGBoost` increased as the missing percentage decreased. At lower missing percentages, the sizes of the training and validation sets for the gradient boosting models were larger, so this algorithm was more computationally intensive and slower.

While the `Genomic`, `TensorGenomic`, and `XGBoost` methods had relatively high total memory usage compared to the other methods (up to 2TB for the `TensorGenomic` algorithm with 20% missing data), peak memory usage was much lower. In particular, all computational experiments were performed on a machine with 16GB RAM, so high memory machines are not required to run any of the methods.

5 Discussion

In this section, we discuss the results from the experiments on simulated and real-world data in Sects. 3 and 4. In addition, we discuss potential applications of this algorithm to other drug response data sets and other areas for future research.

Overall, both sets of experiments demonstrate that the proposed methods for tensor completion which combine a low rank and a regression component generally match or outperform methods which have only one of these components. In the simulated data experiments, we see that the proposed method `TensorTwoSided` outperforms the low rank and regression methods across all levels of feature noise considered. In the real-world data experiments, we see that the proposed method `TensorGenomic` matches the low rank and regression methods across all percentages of missing data considered, and strictly outperforms the other methods for the GDSC data set with 80% missing data. On the CCLE and GCSI data sets, `XGBoost` has a slight edge over `TensorGenomic`, however we note that the `XGBoost` runtime is significantly slower. For example, on the CCLE data set with 20% missing data, the R^2 values of both the `XGBoost` and `TensorGenomic` methods are approximately 0.66, but the `XGBoost` method with hyperparameter tuning takes approximately 4 h while the `TensorGenomic` method with hyperparameter tuning takes approximately 16 min. Therefore, the `TensorGenomic` method may be preferable to the `XGBoost` method in certain applications where computational speed is a high priority.

In addition, the computational experiments on real-world data show that the proposed methods can outperform state-of-the-art methods for the task of predicting anti-cancer drug response. First, we observe that the tensor model on its own significantly outperforms the multilevel mixed effects model which is used in practice. We suspect that the multilevel mixed effects model generalizes poorly because the dose response curves of some patients are significantly different from a "typical" sigmoidal dose response curve. Some patients may have mutations which make them completely resistant to certain anti-cancer drugs, while other patients may be extra sensitive to certain drugs. As a result, the dose response curves of these patients may be significantly different from the population average, which goes against the probabilistic assumptions of the multilevel mixed effects model.

Furthermore, the real-world experiments demonstrate that we can improve the out-of-sample performance of the tensor model using the genomic features which are available on the patients. We see that adding genomic data side information is more useful when the percentage of missing data is high. When the missing percentage is lower, most of the predictive power comes from the original tensor model. As a result, the final method `TensorGenomic` performs better than either the `Tensor` or `Genomic` methods individually. At low levels of missing data, predictive models with more tunable parameters such as

XGBoost outperform the basic Genomic method which is based upon an ℓ_2-regularized regression model.

These results suggest that the tensor data is quite valuable when it is available. One of the best predictors of an individual's response to chemotherapy may be how this individual responded to previous rounds of chemotherapy, even at different drugs and doses. In a clinical setting, if a patient is receiving their 4th round of chemotherapy, we may be able to optimize the drug and dose depending on the results from their first 3 rounds of treatment along with their individual characteristics. However, if a patient is starting their first round of chemotherapy, then we must rely solely upon the individual characteristics to make a treatment decision.

In this paper, we performed our computational experiments on the CCLE, GDSC, and GCSI data sets due to their large size and public availability. At the time of its release, the GDSC data set was the largest publicly available resource for cancer drug response information (Yang et al., 2012). The CCLE data set is a similarly sized publicly available cancer drug response data set (Barretina et al., 2012). Both of these data sets have enabled significant research studies since their release (Ghandi et al., 2019; Suphavilai et al., 2018; Tan, 2016; Liu et al., 2018; Wang et al., 2017). The GCSI data set was developed independently on a subset of cell lines and drugs included in the GDSC and CCLE data sets to address inconsistencies in the previous two data sets (Haverty et al., 2016).

We note that the tensor-based algorithms developed in this work may be applied to other anticancer drug screens as well. For example, the NCI-60 anticancer drug screen developed by the National Cancer Institute in the 1980's includes 60 cell lines (Shoemaker, 2006). The Cancer Therapeutics Response Portal (CTRP) data set developed by the Broad Institute includes 481 small molecules and drugs applied to 860 cancer cell lines (Rees et al., 2016). These data sets also include genomic side information for the cell lines and features of the drugs, so the TensorOneSided and TensorTwoSided algorithms could be applied here. Furthermore, because they are scalable, these algorithms may be applied to larger anticancer drug screening data sets which become available in future years. Since this paper only considers two data sets for the real-world computational experiments, follow-up experimental studies are required to demonstrate that the Tensor-based methods perform consistently well across a wide range of problems. In particular, while we show the potential performance gain from the TensorTwoSided algorithm in the synthetic data experiments, it remains to show that this method leads to a significant performance gain on a real-world prediction task.

In addition to applications on more data sets, there is opportunity for more theoretical work to understand the convergence properties of these algorithms. For example, it would be informative to characterize the stationary points that the TensorOneSided and TensorTwoSided algorithms may achieve. Moreover, it would be interesting to determine the statistical error bounds of these methods as well. Prior work has been done to characterize the convergence properties and statistical error bounds for similar non-convex algorithms for robust PCA and low-rank matrix completion tasks (Yi et al., 2016; Chen & Wainwright, 2015). Convergence and statistical analysis of tensor-based alternating minimization algorithms is a promising area for future work.

6 Conclusions

In this paper, we propose a new approach for tensor completion with noisy side information, and we introduce two methods which take into account noisy features of the rows and/or columns of the tensor, respectively. In computational experiments on real-world data sets, we show that the proposed method `TensorGenomic` works well in practice imputing missing values in the GDSC, CCLE, and GCSI data sets leveraging genomic side information. For this particular application, our work demonstrates that tensor-based models are effective tools representing data from large-scale anti-cancer drug screens. More broadly, our work demonstrates that tensor-based models are powerful tools representing real-world data from complex systems, and these models can be easily augmented and improved with noisy side information.

Appendix

This appendix contains supplementary material for this paper. In Appendix 1, we provide formal definitions of tensor rank. In Appendix 2, we present the details of the `TensorTwoSided` algorithm to solve the tensor completion problem given noisy two-sided information which is presented in Sect. 2.7. In Appendix 3, we provide plots of the tensor rank which is selected for the `Tensor` and `TensorOneSided` methods for the computational experiments in Sect. 4.

Appendix 1: Definitions of Tensor Rank

In this section, we provide several definitions for the rank of a 3-dimensional tensor, including the CP rank, Tucker rank, and Slice rank. The definitions of CP rank and Tucker rank are well-known, and these are also described by Kolda and Bader (2009). The definition of Slice rank was introduced in recent work by Farias and Li (2019).

1. **CP rank:** A tensor $\mathbf{Z} \in \mathbb{R}^{n \times m \times \ell}$ is CP rank-1 if and only if it can be directly expressed as the outer product of vectors. In other words, there exists vectors $\mathbf{u} \in \mathbb{R}^n$, $\mathbf{v} \in \mathbb{R}^m$, $\mathbf{w} \in \mathbb{R}^\ell$ such that $z_{ij}^k = u_i v_j w_k$ for all i, j, k. In general, the CP rank of a tensor $\mathbf{Z}$ is the minimum number r such that $\mathbf{Z}$ can be expressed as the sum of r CP rank-1 tensors.
2. **Tucker rank:** The Tucker rank is the tuple (r_1, r_2, r_3) of column ranks of the mode-1, mode-2, and mode-3 unfoldings of the tensor, or equivalently:
$$\text{Tucker}(\mathbf{Z}) := (\text{rank}(\mathbf{Z}_{(1)}), \text{rank}(\mathbf{Z}_{(2)}), \text{rank}(\mathbf{Z}_{(3)})).$$
3. **Slice rank:** The slice rank is the maximum of the column ranks of the mode-1 and mode-2 unfoldings of the tensor, or equivalently:
$$\text{Slice}(\mathbf{Z}) := \max\{\text{rank}(\mathbf{Z}_{(1)}), \text{rank}(\mathbf{Z}_{(2)})\}.$$
Further, if $\mathbf{Z}$ has Slice rank equal to r, then we can find a decomposition such that $\mathbf{Z}^k = \mathbf{U}\mathbf{S}^k\mathbf{V}^T, k = 1, \dots, \ell$ for some matrices $\mathbf{U} \in \mathbb{R}^{n \times r}$, $\mathbf{V} \in \mathbb{R}^{m \times r}$, and $\mathbf{S}^1, \dots, \mathbf{S}^\ell \in \mathbb{R}^{r \times r}$.

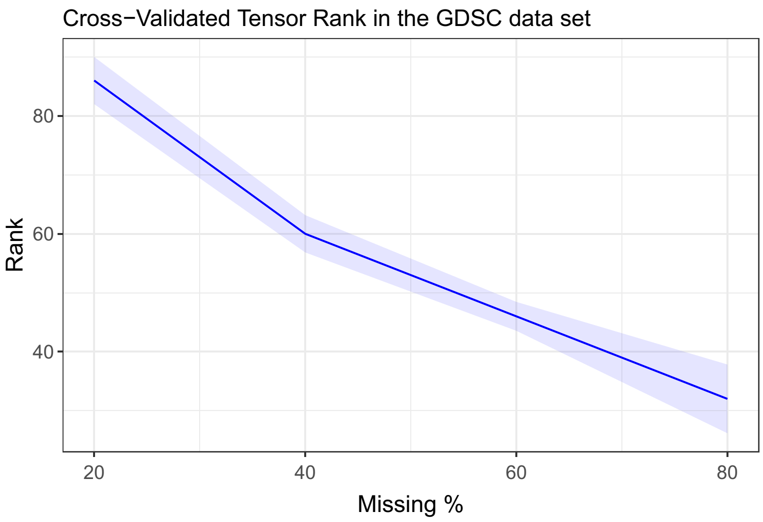

Fig. 9 Average Slice rank for the Tensor model on the GDSC data set at varying missing percentages. In each experiment, the rank is selected via cross-validation from the range $\{10, 20, \dots, 120\}$

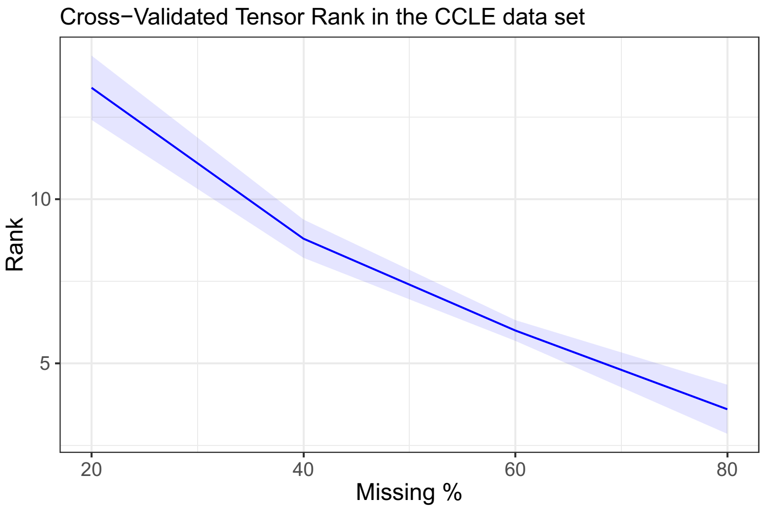

Fig. 10 Average Slice rank for the Tensor model on the CCLE data set at varying missing percentages. In each experiment, the rank is selected via cross-validation from the range $\{1, 2, \dots, 20\}$

Appendix 2: `TensorTwoSided` Algorithm

In this section, we present the alternating minimization algorithm `TensorTwoSided` to solve the tensor completion problem given noisy two-sided information. This algorithm finds high-quality solutions to problem (16) which was introduced in Sect. 2.7. It is identical to the `TensorOneSided` algorithm except for the update of $\mathbf{W}$ in Step 3.

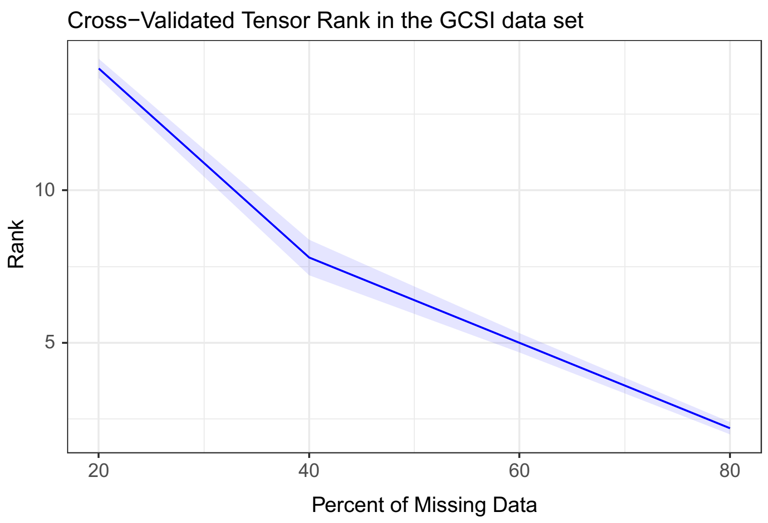

Fig. 11 Average Slice rank for the Tensor model on the GCSI data set at varying missing percentages. In each experiment, the rank is selected via cross-validation from the range $\{1, 2, \ldots, 16\}$

1. Begin with a warm start solution $\hat{\mathbf{Z}}$. Initialize all of the variables $\mathbf{W}, \mathbf{U}, \mathbf{S}, \mathbf{V}$ to zero.
2. Update $\mathbf{U}, \mathbf{S}, \mathbf{V}$ by considering the following problem:

$$\min_{\mathbf{U},\mathbf{S},\mathbf{V}} \sum_{k=1}^{\ell} \sum_{(i,j)\in\Omega_k} \left((\hat{\mathbf{Z}}^k - \mathbf{X}\mathbf{W}^k\mathbf{Y}^T)_{ij} - (\mathbf{U}\mathbf{S}^k\mathbf{V}^T)_{ij} \right)^2. \tag{17}$$

   We can find high-quality solutions to this problem using the Slice Learning algorithm (Farias & Li, 2019). Let $\mathbf{R}$ be the tensor of residuals, where $\mathbf{R}^k = \hat{\mathbf{Z}}^k - \mathbf{X}\mathbf{W}^k\mathbf{Y}^T$. In this step, we find a low rank tensor approximation to $\mathbf{R}$ by taking SVDs of the mode-1 and mode-2 unfoldings.
3. Update the $\mathbf{W}$ by considering the following problem:

$$\min_{\mathbf{W}} \sum_{k=1}^{\ell} \sum_{(i,j)\in\Omega_k} \left((\hat{\mathbf{Z}}^k - \mathbf{U}\mathbf{S}^k\mathbf{V}^T)_{ij} - (\mathbf{X}\mathbf{W}^k\mathbf{Y}^T)_{ij} \right)^2 + \frac{1}{\gamma}\|\mathbf{W}^k\|_F^2. \tag{18}$$

   This is a quadratic optimization problem, so it is efficiently solvable via gradient descent. Let $\mathbf{R}$ be the tensor of residuals, where $\mathbf{R}^k = \hat{\mathbf{Z}}^k - \mathbf{U}\mathbf{S}^k\mathbf{V}^T$. Given a warm start solution $\mathbf{W}_0$, maximum number of gradient steps G, and step size $\nu > 0$, we denote this update compactly as

$$\mathbf{W} \leftarrow \texttt{TwoSided}(\mathbf{R}, \Omega, \mathbf{X}, \mathbf{Y}, \mathbf{W}_0, \gamma, G, \nu),$$

   which is detailed in Algorithm 2.
4. Iterate until the variables $\mathbf{W}, \mathbf{U}, \mathbf{S}, \mathbf{V}$ converge, or the maximum iteration limit is reached.

We express the steps of the complete algorithm `TensorTwoSided` in Algorithm 5.

Algorithm 5 `TensorTwoSided`

```
Data: Tensor $\mathbf{Z} \in \mathbb{R}^{n\times m\times \ell}$, with
    known entries $\Omega = \{(i,j,k) : z_{ij}^k \text{ is known}\}$,
    missing entries $\Omega^c = \{(i,j,k) : z_{ij}^k \text{ is missing}\}$,
    and side information $\mathbf{X} \in \mathbb{R}^{n\times p}$, $\mathbf{Y} \in \mathbb{R}^{m\times q}$.
Input: Rank $r$, warm start $\mathbf{Z}_0 \in \mathbb{R}^{n\times m\times \ell}$,
    regularization parameter $\gamma > 0$,
    max number of iterations $T \geq 1$.
Output: Locally optimal solution to problem (16).
Procedure:
    Initialize $\hat{\mathbf{Z}} \leftarrow \mathbf{Z}$, $\mathbf{W} \leftarrow \mathbf{0}$, $t \leftarrow 0$.
    while $t < T$ do
        $\mathbf{R}^k \leftarrow \hat{\mathbf{Z}}^k - \mathbf{X}\mathbf{W}^k\mathbf{Y}^T \quad \forall k$,
        $\mathbf{U} \leftarrow \text{svds}(\mathbf{R}_{(1)}, r)$,
        $\mathbf{V} \leftarrow \text{svds}(\mathbf{R}_{(2)}, r)$,
        $\mathbf{S}^k \leftarrow \mathbf{U}^T\mathbf{R}^k\mathbf{V} \quad \forall k$,
        $\mathbf{R}^k \leftarrow \hat{\mathbf{Z}}^k - \mathbf{U}\mathbf{S}^k\mathbf{V}^T \quad \forall k$,
        $\mathbf{W} \leftarrow \texttt{TwoSided}(\mathbf{R}, \Omega, \mathbf{X}, \mathbf{Y}, \mathbf{W}, \gamma, G, \nu)$,
        $\hat{z}_{ij}^k \leftarrow (\mathbf{U}\mathbf{S}^k\mathbf{V}^T + \mathbf{X}\mathbf{W}^k\mathbf{Y}^T)_{ij} \quad \forall (i,j,k) \in \Omega^c$,
        $t \leftarrow t + 1$.
    end while
    return Estimated values of $\mathbf{W}, \mathbf{U}, \mathbf{S}, \mathbf{V}$.
```

Appendix 3: Plots of Cross-validated Tensor Rank

In this section, we provide plots of the average cross-validated tensor rank selected by the `Tensor` and `TensorOneSided` methods in the computational experiments in Sect. 4.

Author Contributions DB contributed to the Conceptualization, Methodology, Validation, Writing, Editing, and Supervision for this manuscript. CP contributed to the Conceptualization, Methodology, Validation, Writing, Editing, Formal Analysis, Investigation, Software, Data Curation, and Visualization for this manuscript.

Funding 'Open Access funding provided by the MIT Libraries'. Not applicable.

Data availability The GDSC, CCLE, and GCSI anticancer drug screening data sets are available for download at: https://pharmacodb.ca/. The COSMIC data sets of genomic features for cell lines are available for download at: https://cancer.sanger.ac.uk/cell_lines/download.

Code availability The code for this study may be made available upon request to the authors.

Declarations

Conflict of interest The authors declare that they have no conflicts of interest.

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

References

Acar, E., Kolda, T.G., & Dunlavy, D.M. (2011). All-at-once optimization for coupled matrix and tensor factorizations. arXiv preprint arXiv:1105.3422

Azuaje, F. (2016). Computational models for predicting drug responses in cancer research. *Briefings in Bioinformatics, 18*(5), 820–829.

Barretina, J., Caponigro, G., Stransky, N., Venkatesan, K., Margolin, A. A., Kim, S., Wilson, C. J., Lehár, J., Kryukov, G. V., Sonkin, D., et al. (2012). The Cancer Cell Line Encyclopedia enables predictive modelling of anticancer drug sensitivity. *Nature, 483*(7391), 603.

Bell, R. M., & Koren, Y. (2007). Lessons from the Netflix prize challenge. *SiGKDD Explorations, 9*(2), 75–79.

Bennett, J., & Lanning, S., et al. (2007). The Netflix prize. In: *Proceedings of KDD Cup and Workshop*, New York, NY, USA, vol 2007, p 35

Bertsimas, D., & Li, M.L. (2018). Interpretable matrix completion: A discrete optimization approach. arXiv preprint arXiv:1812.06647.

Cai, J. F., Candès, E. J., & Shen, Z. (2010). A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization, 20*(4), 1956–1982.

Candes, E. J., & Plan, Y. (2010). Matrix completion with noise. *Proceedings of the IEEE, 98*(6), 925–936.

Candès, E. J., & Recht, B. (2009). Exact matrix completion via convex optimization. *Foundations of Computational Mathematics, 9*(6), 717.

Candès, E.J., & Tao, T. (2009). The power of convex relaxation: Near-optimal matrix completion. arXiv preprint arXiv:0903.1476.

Chen, H., Raskutti, G., & Yuan, M. (2019). Non-convex projected gradient descent for generalized low-rank tensor regression. *Journal of Machine Learning Research, 20*(5), 1–37.

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp 785–794.

Chen, Y., & Wainwright, M.J. (2015). Fast low-rank estimation by projected gradient descent: General statistical and algorithmic guarantees. arXiv preprint arXiv:1509.03025.

Chiang, K. Y., Hsieh, C. J., & Dhillon, I. S. (2015). Matrix completion with noisy side information. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems.* Berlin: Curran Associates Inc.

Farias, V. F., & Li, A. A. (2019). Learning preferences with side information. *Management Science, 65*(7), 3131–3149.

Forbes, S. A., Beare, D., Boutselakis, H., Bamford, S., Bindal, N., Tate, J., Cole, C. G., Ward, S., Dawson, E., Ponting, L., et al. (2016). Cosmic: Somatic cancer genetics at high-resolution. *Nucleic Acids Research, 45*(D1), D777–D783.

Franco, M., Jeggari, A., Peuget, S., Böttger, F., Selivanova, G., & Alexeyenko, A. (2019). Prediction of response to anti-cancer drugs becomes robust via network integration of molecular data. *Scientific Reports, 9*(1), 2379.

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

Ghandi, M., Huang, F. W., Jané-Valbuena, J., Kryukov, G. V., Lo, C. C., McDonald, E. R., Barretina, J., Gelfand, E. T., Bielski, C. M., Li, H., et al. (2019). Next-generation characterization of the cancer cell line encyclopedia. *Nature, 569*(7757), 503–508.

Gönen, M., Khan, S., & Kaski, S. (2013). Kernelized Bayesian matrix factorization. In *International Conference on Machine Learning*, pp 864–872.

Hamosh, A., Scott, A. F., Amberger, J. S., Bocchini, C. A., & McKusick, V. A. (2005). Online Mendelian Inheritance in Man (OMIM), a knowledgebase of human genes and genetic disorders. *Nucleic Acids Research, 33*(1), D514–D517.

Haverty, P. M., Lin, E., Tan, J., Yu, Y., Lam, B., Lianoglou, S., Neve, R. M., Martin, S., Settleman, J., Yauch, R. L., et al. (2016). Reproducible pharmacogenomic profiling of cancer cell line panels. *Nature, 533*(7603), 333–337.

Hu, Y., Koren, Y., & Volinsky, C. (2008). Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*, IEEE, pp 263–272.

Iorio, F., Knijnenburg, T. A., Vis, D. J., Bignell, G. R., Menden, M. P., Schubert, M., Aben, N., Gonçalves, E., Barthorpe, S., Lightfoot, H., et al. (2016). A landscape of pharmacogenomic interactions in cancer. *Cell, 166*(3), 740–754.

Jain, P., & Dhillon, I.S. (2013). Provable inductive matrix completion. arXiv preprint arXiv:1306.0626.

Jain, P., Netrapalli, P., & Sanghavi, S. (2013). Low-rank matrix completion using alternating minimization. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, ACM, pp 665–674.

Kluver, D., Ekstrand, M.D., & Konstan, J.A. (2018). Rating-based collaborative filtering: algorithms and evaluation. In *Social Information Access*, Springer, pp 344–390.

Kolda, T. G., & Bader, B. W. (2009). Tensor decompositions and applications. *SIAM Review, 51*(3), 455–500.

Koren, Y., & Bell, R. (2015). Advances in collaborative filtering. *Recommender Systems Handbook*. https://doi.org/10.1007/978-1-0716-2197-4_3

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

Lamba, H., Nagarajan, V., Shin, K., & Shajarisales, N. (2016). Incorporating side information in tensor completion. In *Proceedings of the 25th International Conference Companion on World Wide Web*, pp 65–66.

Liu, H., Zhao, Y., Zhang, L., & Chen, X. (2018). Anti-cancer drug response prediction using neighbor-based collaborative filtering with global effect removal. *Molecular Therapy-Nucleic Acids, 13*, 303–311.

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

Lu, J., Liang, G., Sun, J., & Bi, J. (2016). A sparse interactive model for matrix completion with side information. *Advances in Neural Information Processing Systems*, *29*, 4071–4079

Mazumder, R., Hastie, T., & Tibshirani, R. (2010). Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research, 11*, 2287–2322.

Narita, A., Hayashi, K., Tomioka, R., & Kashima, H. (2012). Tensor factorization using auxiliary information. *Data Mining and Knowledge Discovery, 25*(2), 298–324.

Natarajan, N., & Dhillon, I. S. (2014). Inductive matrix completion for predicting gene-disease associations. *Bioinformatics, 30*(12), i60–i68.

Nesterov, Y. E. (1983). A method for solving the convex programming problem with convergence rate O(1/kˆ2). *Doklady Akademii Nauk SSSR, 269*, 543–547.

Nocedal, J., & Wright, S. (2006). *Numerical optimization*. Berlin: Springer.

Rahman, R., & Pal, R. (2019). Predictive modeling of anti-cancer drug sensitivity from genetic characterizations. *Bioinformatics*. https://doi.org/10.1007/978-1-4939-8868-6_14

Rai, P., Wang, Y., & Carin, L. (2015). Leveraging features and networks for probabilistic tensor decomposition. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Rees, M. G., Seashore-Ludlow, B., Cheah, J. H., Adams, D. J., Price, E. V., Gill, S., Javaid, S., Coletti, M. E., Jones, V. L., Bodycombe, N. E., et al. (2016). Correlating chemical sensitivity and basal gene expression reveals mechanism of action. *Nature Chemical Biology, 12*(2), 109–116.

Shoemaker, R. H. (2006). The NCI60 human tumour cell line anticancer drug screen. *Nature Reviews Cancer, 6*(10), 813.

Smirnov, P., Kofia, V., Maru, A., Freeman, M., Ho, C., El-Hachem, N., Adam, G. A., Ba-alawi, W., Safikhani, Z., & Haibe-Kains, B. (2017). PharmacoDB: An integrative database for mining in vitro anticancer drug screening studies. *Nucleic Acids Research, 46*(D1), D994–D1002. https://doi.org/10.1093/nar/gkx911

Su, R., Liu, X., Wei, L., & Zou, Q. (2019). Deep-Resp-Forest: A deep forest model to predict anti-cancer drug response. *Methods, 166*, 91–102.

Suphavilai, C., Bertrand, D., & Nagarajan, N. (2018). Predicting cancer drug response using a recommender system. *Bioinformatics, 34*(22), 3907–3914.

Tan, M. (2016). Prediction of anti-cancer drug response by kernelized multi-task learning. *Artificial Intelligence in Medicine, 73*, 70–77.

Tomioka, R., Suzuki, T., Hayashi, K., & Kashima, H. (2011). Statistical performance of convex tensor decomposition. *Advances in Neural Information Processing Systems*, *24*, 972–980.

Tucker, L. R. (1966). Some mathematical notes on three-mode factor analysis. *Psychometrika, 31*(3), 279–311.

Vis, D. J., Bombardelli, L., Lightfoot, H., Iorio, F., Garnett, M. J., & Wessels, L. F. (2016). Multilevel models improve precision and speed of IC50 estimates. *Pharmacogenomics, 17*(7), 691–700.

Wang, L., Li, X., Zhang, L., & Gao, Q. (2017). Improved anticancer drug response prediction in cell lines using matrix factorization with similarity regularization. *BMC Cancer, 17*(1), 1–12.

Weinstein, J. N., Collisson, E. A., Mills, G. B., Shaw, K. R. M., Ozenberger, B. A., Ellrott, K., Shmulevich, I., Sander, C., Stuart, J. M., Network, C. G. A. R., et al. (2013). The cancer genome atlas pan-cancer analysis project. *Nature Genetics, 45*(10), 1113.

Wimalawarne, K., Yamada, M., & Mamitsuka, H. (2018). Convex coupled matrix and tensor completion. *Neural Computation, 30*(11), 3095–3127.

Yang, W., Soares, J., Greninger, P., Edelman, E. J., Lightfoot, H., Forbes, S., Bindal, N., Beare, D., Smith, J. A., Thompson, I. R., et al. (2012). Genomics of Drug Sensitivity in Cancer (GDSC): A resource for therapeutic biomarker discovery in cancer cells. *Nucleic Acids Research, 41*(D1), D955–D961.

Yang, Z., Wu, B., Zheng, K., Wang, X., & Lei, L. (2016). A survey of collaborative filtering-based recommender systems for mobile internet applications. *IEEE Access, 4*, 3273–3287.

Yi, X., Park, D., Chen, Y., & Caramanis, C. (2016). Fast algorithms for robust pca via gradient descent. arXiv preprint arXiv:1605.07784.

Zeng, X., Ding, N., Rodríguez-Patón, A., & Zou, Q. (2017). Probability-based collaborative filtering model for predicting gene-disease associations. *BMC Medical Genomics, 10*(5), 76.

Zhou, T., Qian, H., Shen, Z., Zhang, C., & Xu, C. (2017). Tensor completion with side information: A riemannian manifold approach. In *IJCAI*, pp 3539–3545.