Submitted to *Operations Research*
manuscript (Please, provide the manuscript number!)

Interpretable Matrix Completion: A Discrete Optimization Approach

Dimitris Bertsimas
Sloan School of Management and Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA 02139, dbertsim@mit.edu

Michael Lingzhi Li
Operations Research Center, Massachusetts Institute of Technology, Cambridge, MA 02139, mlli@mit.edu

We consider the problem of matrix completion with side information on an $n \times m$ matrix. We formulate the problem exactly as a sparse regression problem of selecting features and show that it can be reformulated as a binary convex optimization problem. We design OptComplete, based on a novel concept of stochastic cutting planes to enable efficient scaling of the algorithm up to matrices of sizes $n = 10^6$ and $m = 10^5$. We report experiments on both synthetic and real-world datasets that show that OptComplete outperforms current state-of-the-art methods both in terms of accuracy and scalability, while providing insight on the factors that affect the ratings.

Key words: Matrix Completion, Mixed-Integer Optimization, Stochastic Approximation

1. Introduction

Low-rank matrix completion has attracted much attention after the successful application in the Netflix Competition. It is now widely utilized in far-reaching areas such as computer vision (Candes and Plan (2010)), signal processing (Ji et al. (2010)), and control theory (Boyd et al. (1994)) to generate a completed matrix from partially observed entries. Given a data matrix $\boldsymbol{A} \in \mathbb{R}^{n\times m}$, the low-rank assumption assumes that rank($\boldsymbol{A}$) is small - in other words there are only a few, but still unknown, common linear factors that affect A_{ij}. For example, in the original Netflix competition where each row represents a person and each column represents a movie, the entry A_{ij} represents the score that person i assigns to movie j. It is reasonable to postulate that there are only a few factors that affect how a person rates a movie, and thus the low-rank assumption can be used. In most applications of the matrix completion problem, there is a well-defined list of possible factors that could determine A_{ij}. Thus, there has been a considerable rise in interest for *inductive matrix completion*, where side information on the rows and columns of the matrices can be utilized. We next review the literature in this area.

Literature

Matrix completion has been applied successfully to many tasks, including recommender systems Koren et al. (2009), social network analysis Chiang et al. (2014) and clustering Chen et al. (2014b). After Candès and Tao (2010) proved a theoretical guarantee for the retrieval of the exact matrix under the nuclear norm convex relaxation, a lot of methods have focused on the nuclear norm problem (see Mazumder et al. (2010), Beck and Teboulle (2009), Jain et al. (2010), and Tanner and Wei (2013) for examples). Alternative methods include alternating projections by Recht and Ré (2013) and Grassmann manifold optimization by Keshavan et al. (2009). There has also been work where the uniform distributional assumptions required by the theoretical guarantees are violated, such as Negahban and Wainwright (2012) and Chen et al. (2014a).

Interest in inductive matrix completion intensified after Xu et al. (2013) showed that given predictive side information, one only needs $O(\log n)$ samples to retrieve the full matrix. Thus, most of this work (see Xu et al. (2013), Jain and Dhillon (2013), Farhat et al. (2013), Natarajan and Dhillon (2014)) have focused on the case in which the side information is assumed to be perfectly predictive so that the theoretical bound of $O(\log n)$ sample complexity Xu et al. (2013) can be achieved. Chiang et al. (2015) explored the case in which the side information is corrupted with noise, while Shah et al. (2017) and Si et al. (2016) incorporated nonlinear combination of factors into the side information. Surprisingly, as pointed out by a recent article Nazarov et al. (2018), there is a considerable lack of effort to introduce sparsity into inductive matrix completion, with Lu et al. (2016), Soni et al. (2016) and Nazarov et al. (2018) being among the only works that attempt to do so. Our work differs from the previous attempts to introduce sparsity in that it does not consider the heuristic convex relaxation of sparsity in the nuclear norm. Instead, we tackle the exact sparse formulation through a binary convex reformulation. Combined with the novel algorithmic advancements in stochastic cutting planes that we propose in this work, we are able to achieve exact retrieval with superior speed compared to earlier attempts.

Contributions and Structure

We use the term interpretable, as opposed to inductive, matrix completion, to highlight that our approach, like sparse linear regression, gives insights on which factors affect the estimation of the matrix $\boldsymbol{A}$. The rank condition is equivalent to the sparsity of these factors. We propose a new method, inspired by Bertsimas and Van Parys (2017) for sparse linear regression, to conduct sparse interpretable matrix completion exactly. Unlike previous methods which utilized the nuclear norm convex relaxation, we solve the exact low rank problem with side feature information by reformulating the rank minimization problem as a binary convex optimization problem. We introduce a new algorithm OptComplete, a stochastic cutting planes algorithm, to enable scalability for matrices of sizes on the order of $(n,m) = (10^6, 10^5)$. In addition, we provide empirical evidence on both synthetic and real-world data that OptComplete exceeds the current state-of-the-art convex methods on speed and accuracy. Specifically, our contributions in this paper are as follows:

1. We reformulate the low-rank interpretable matrix completion problem with side information as a binary convex optimization problem that can be solved using cutting planes methods.
2. We propose a new novel approach to cutting planes by introducing stochastic cutting planes. We prove that the new algorithm converges to an optimal solution with high probability.

3. We present computational results on both synthetic and real datasets that show that the algorithm outperforms current state-of-the-art methods in terms of both scalability and accuracy,

The structure of the paper is as follows. In Section 2, we introduce the binary convex reformulation of the low-rank interpretable matrix completion problem. In Section 3, we introduce the base cutting plane algorithm CutPlanes. In Section 4, we introduce OptComplete, a stochastic cutting planes method designed to scale the CutPlanes algorithm in Section 3. In Section 5, we report on computational experiments with synthetic data that compare OptComplete to Inductive Matrix Completion (IMC) introduced in Natarajan and Dhillon (2014) and SoftImpute-ALS by Hastie et al. (2015), two state-of-the-art matrix completion algorithms. In Section 6, we report on computational experiments on the real-world Netflix dataset. In Section 7 we provide our conclusions. Appendix A contains the proof of convergence and optimality of the OptComplete algorithm, and Appendix B contains the list of features used for the Netflix dataset.

2. Binary Convex Reformulation of Matrix Completion

The classical matrix completion problem considers a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times m}$ in which $\Omega = \{(i,j) \mid A_{ij} \text{ is known}\}$ is the set of the known entries of $\boldsymbol{A}$. We aim to recover a matrix $\boldsymbol{X} \in \mathbb{R}^{n\times m}$ of rank k that minimizes the distance between $\boldsymbol{X}$ and $\boldsymbol{A}$ on the known entries $\boldsymbol{A}$:

$$\min_{\boldsymbol{X}} \frac{1}{n} \sum_{(i,j)\in\Omega} (X_{ij} - A_{ij})^2 \quad \text{subject to} \quad \text{Rank}(\boldsymbol{X}) = k.$$

The problem we consider here is that for every column $j = 1,\ldots,m$, we have a given p-dimensional feature vector $\boldsymbol{B}_j$ with $p \geq k$ that contains the information we have on column j. In the Netflix example, column j corresponds to movie j, and thus the feature vector $\boldsymbol{B}_j$ includes information about the movie: Budget, Box Office revenue, IMDB rating, etc. We represent all this side information with a matrix $\boldsymbol{B} \in \mathbb{R}^{p\times m}$. Given side data $\boldsymbol{B}$ we next rewrite the rank condition as a sparsity condition over a set of p binary variables $\boldsymbol{s} = (s_1,\ldots,s_p) \in S_k^p$:

$$S_k^p = \left\{ \boldsymbol{s} = (s_1,\ldots,s_p)^T \in \{0,1\}^p : \sum_{i=1}^{p} s_i = k \right\}.$$

We introduce the diagonal matrix $\boldsymbol{S} = \mathrm{Diag}\{s_1, \ldots, s_p\} \in \mathbb{R}^{p \times p}$ and define the matrix $\boldsymbol{U} \in \mathbb{R}^{n \times p}$ of feature exposures. Then, the matrix completion problem with side data $\boldsymbol{B}$ can be written as :

$$\min_{\boldsymbol{s} \in S_k^p} \min_{\boldsymbol{U}} \frac{1}{n} \sum_{(i,j) \in \Omega} (X_{ij} - A_{ij})^2 \quad \text{subject to} \quad \boldsymbol{X} = \boldsymbol{USB}.$$

We note that given that $\sum_{i=1}^p s_i = k$, the rank of matrix $\boldsymbol{X}$ is indeed k.

Similar to linear regression and for robustness purposes (see Bertsimas and Van Parys (2017) and Bertsimas and Copenhaver (2018)), we address in this paper the problem with a Tikhonov regularization term. Specifically, the matrix completion problem with side information and regularization we address is

$$\min_{\boldsymbol{s} \in S_k^p} \min_{\boldsymbol{U}} \frac{1}{n} \left(\sum_{(i,j) \in \Omega} (X_{ij} - A_{ij})^2 + \frac{1}{\gamma} \|\boldsymbol{U}\|_2^2 \right) \quad \text{subject to} \quad \boldsymbol{X} = \boldsymbol{USB}, \tag{1}$$

where $\gamma > 0$ is a given parameter that controls the strength of the regularization term. Then we have the following theorem:

Theorem 1 *Problem (1) can be reformulated as a binary convex optimization problem:*

$$\min_{\boldsymbol{s} \in S_k^p} c(\boldsymbol{s}) = \frac{1}{n} \sum_{i=1}^n \overline{\boldsymbol{a}}_i^T \left(\boldsymbol{I}_m + \gamma \boldsymbol{W}_i \left(\sum_{j=1}^p s_j \boldsymbol{K}_j \right) \boldsymbol{W}_i \right)^{-1} \overline{\boldsymbol{a}}_i, \tag{2}$$

where $\boldsymbol{W}_1, \cdots, \boldsymbol{W}_n \in \mathbb{R}^{m \times m}$ *are diagonal matrices:*

$$(\boldsymbol{W}_i)_{jj} = \begin{cases} 1, & (i,j) \in \Omega, \\ \\ 0, & (i,j) \notin \Omega, \end{cases}$$

$\overline{\boldsymbol{a}}_i = \boldsymbol{W}_i \boldsymbol{a}_i$, $i = 1 \ldots, n$, *where* $\boldsymbol{a}_i \in \mathbb{R}^{m \times 1}$ *is the* i*th row of* $\boldsymbol{A}$ *with unknown entries taken to be 0, and* $\boldsymbol{K}_j = \boldsymbol{b}_j \boldsymbol{b}_j^T \in \mathbb{R}^{m \times m}$, $j = 1, \ldots, p$ *with* $\boldsymbol{b}_j \in \mathbb{R}^{m \times 1}$ *the* j*th row of* $\boldsymbol{B}$.

W ith the diagonal matrices $\boldsymbol{W}_i$ defined above, we can rewrite the sum in (1) over known entries of $\boldsymbol{A}$, $\sum_{(i,j) \in \Omega} (X_{ij} - A_{ij})^2$, as a sum over the rows of $\boldsymbol{A}$:

$$\sum_{i=1}^n \|\boldsymbol{W}_i (\boldsymbol{x}_i - \boldsymbol{a}_i)\|_2^2,$$

where $\boldsymbol{x}_i \in \mathbb{R}^{m\times 1}$ is the ith row of $\boldsymbol{X}$. Using $\boldsymbol{X} = \boldsymbol{USB}$, then $\boldsymbol{x}_i^T = \boldsymbol{u}_i^T \boldsymbol{SB}$ where $\boldsymbol{u}_i \in \mathbb{R}^{m\times 1}$ is the ith row of U. Moreover,

$$\|\boldsymbol{U}\|_2^2 = \sum_{i=1}^n \|\boldsymbol{u}_i\|_2^2.$$

Then, Problem (1) becomes:

$$\min_{\boldsymbol{s}\in S_k^p} \; \min_{\boldsymbol{U}} \frac{1}{n}\left(\sum_{i=1}^n \left(\|\boldsymbol{W}_i(\boldsymbol{B}^T\boldsymbol{S}\boldsymbol{u}_i - \boldsymbol{a}_i)\|_2^2 + \frac{1}{\gamma}\|\boldsymbol{u}_i\|_2^2\right)\right).$$

We then notice that within the sum $\sum_{i=1}^n$ each row of $\boldsymbol{U}$ can be optimized separately, leading to:

$$\min_{\boldsymbol{s}\in S_k^p} \; \frac{1}{n}\left(\sum_{i=1}^n \min_{\boldsymbol{u}_i} \left(\|\boldsymbol{W}_i(\boldsymbol{B}^T\boldsymbol{S}\boldsymbol{u}_i - \boldsymbol{a}_i)\|_2^2 + \frac{1}{\gamma}\|\boldsymbol{u}_i\|_2^2\right)\right). \tag{3}$$

The inner optimization problem $\min_{\boldsymbol{u}_i} \|\boldsymbol{W}_i(\boldsymbol{B}^T\boldsymbol{S}\boldsymbol{u}_i - \boldsymbol{a}_i)\|_2^2 + \frac{1}{\gamma}\|\boldsymbol{u}_i\|_2^2$ can be solved in closed form given $\boldsymbol{S}$, as it is a weighted linear regression problem with Tiknorov regularization, see Bertsimas and Van Parys (2017). The closed form solution is:

$$\boldsymbol{a}_i^T \boldsymbol{W}_i (\boldsymbol{I}_m + \gamma \boldsymbol{W}_i \boldsymbol{B}^T \boldsymbol{SSBW}_i^T)^{-1} \boldsymbol{W}_i \boldsymbol{a}_i = \overline{\boldsymbol{a}}_i^T (\boldsymbol{I}_m + \gamma \boldsymbol{W}_i \boldsymbol{B}^T \boldsymbol{SBW}_i)^{-1} \overline{\boldsymbol{a}}_i.$$

So Problem (3) can be simplified to:

$$\min_{\boldsymbol{s}\in S_k^p} \; \frac{1}{n}\left(\sum_{i=1}^n \overline{\boldsymbol{a}}_i^T (\boldsymbol{I}_m + \gamma \boldsymbol{W}_i \boldsymbol{B}^T \boldsymbol{SBW}_i^T)^{-1} \overline{\boldsymbol{a}}_i\right).$$

We notice that

$$\boldsymbol{B}^T\boldsymbol{SB} = \sum_{j=1}^p s_j \boldsymbol{b}_j \boldsymbol{b}_j^T = \sum_{j=1}^p s_j \boldsymbol{K}_j$$

and therefore, Problem (1) is equivalent to (2).

2.1. Two-sided Information Case

In this section, we briefly discuss the matrix completion problem under the two-sided information case, and how it reduces to the problem of sparse linear regression. The two sided matrix completion problem with Tikhonov regularization can be stated as follows:

$$\min_{\boldsymbol{L}} \frac{1}{n}\left(\sum_{(i,j)\in\Omega} (X_{ij} - A_{ij})^2 + \frac{1}{\gamma}\|\boldsymbol{L}\|_2^2\right) \quad \text{subject to} \quad \boldsymbol{X} = \boldsymbol{ULB} \quad \|\boldsymbol{L}\|_0 = k, \tag{4}$$

where $\boldsymbol{U} \in \mathbb{R}^{n\times p_1}$ is a known matrix of p_1 features of each row, $\boldsymbol{B} \in \mathbb{R}^{p_2\times m}$ is a known matrix of p_2 features of each column as before, and $\boldsymbol{L} \in \mathbb{R}^{p_1\times p_2}$ is a sparse matrix that has k nonzero entries, ensuring that $\text{Rank}(\boldsymbol{X}) \le k$. We note that in Eq. (4) we restrict the support of matrix $\boldsymbol{L}$ to be k, that is the entries of $\boldsymbol{L}$ are not 0 or 1, as both $\boldsymbol{U}$ and $\boldsymbol{B}$ are known. In contrast, in Eq. (1), as $\boldsymbol{U}$ is unknown, we need only to restrict the matrix $\boldsymbol{S}$ to be diagonal and only containing 0 or 1 entries.

We denote by $\boldsymbol{U}_i \in \mathbb{R}^{n\times 1}$ the ith column of $\boldsymbol{U}$ and $\boldsymbol{b}_j \in \mathbb{R}^{m\times 1}$ the jth row of $\boldsymbol{B}$. We introduce the matrices $\boldsymbol{W}_i$ as in Theorem 1. Using $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{L}\boldsymbol{B}$, we can write

$$X_{ij} = \sum_{q=1}^{p_1}\sum_{\ell=1}^{p_2} L_{q,\ell} D_{ij}^{q,\ell},$$

where $D_{ij}^{q,\ell} = (\boldsymbol{U}_q \boldsymbol{b}_\ell^T)_{ij}$ is the (i,j)th entry of the matrix formed by multiplying qth column of $\boldsymbol{U}$ with ℓth row of $\boldsymbol{B}$. Then, Problem (4) becomes:

$$\min_{\boldsymbol{L}} \frac{1}{n}\left(\sum_{(i,j)\in\Omega}\left(\sum_{q=1}^{p_1}\sum_{\ell=1}^{p_2} L_{q,\ell} D_{ij}^{q,\ell} - A_{ij}\right)^2 + \frac{1}{\gamma}\|\boldsymbol{L}\|_2^2\right) \quad \text{subject to} \quad \|\boldsymbol{L}\|_0 = k. \tag{5}$$

As every $\boldsymbol{D}$ matrix is known, this becomes a sparse regression problem where there are p_1p_2 features to choose from (the $\boldsymbol{D}$ matrices), there are $|\Omega|$ samples (the $\boldsymbol{A}$ matrix), the sparsity requirement is k, the regression coefficients are $\boldsymbol{L}$, and we have Tikhonov regularization. Vectorizing $\boldsymbol{D}$, $\boldsymbol{L}$, and $\boldsymbol{A}$ reduces the problem back to the familiar form of sparse linear regression, that can be solved by the algorithm developed in Bertsimas and Van Parys (2017) at scale.

3. A Cutting-Plane Algorithm

In this section, we utilize the cutting plane algorithm first introduced by Duran and Grossmann (1986) to solve the binary convex optimization problem (2). Given a current feasible solution $\boldsymbol{s}_t$ at Step t of the algorithm, we add the hyperplane:

$$\eta \ge c(\boldsymbol{s}_t) + \nabla c(\boldsymbol{s}_t)^T(\boldsymbol{s} - \boldsymbol{s}_t), \tag{6}$$

which cuts off the current binary solution $\boldsymbol{s}_t$ unless it happens to be optimal. As the algorithm progresses, at Step t the outer approximation function c_t constructed:

$$c_t(\boldsymbol{s}) = \max_{i\in[t]} c(\boldsymbol{s}_t) + \nabla c(\boldsymbol{s}_t)^T(\boldsymbol{s} - \boldsymbol{s}_t)$$

becomes an increasingly better approximation of $c(\boldsymbol{s})$ retaining the property that $c_t(\boldsymbol{s}) \leq c(\boldsymbol{s})$. We describe the algorithm next.

Algorithm 1 Cutting-plane algorithm for matrix completion with side information.

```
procedure CUTPLANES($\boldsymbol{A}, \boldsymbol{B}$)        # masked matrix $\boldsymbol{A}$, and feature matrix $\boldsymbol{B}$
    $t \leftarrow 1$
    $\boldsymbol{s}_1 \leftarrow$ warm start                 # Heuristic Warm Start
    $\eta \leftarrow 0$                                      # Initialize feasible solution variable
    while $\eta_t < c(\boldsymbol{s}_t)$ do                  # While the current solution is not optimal
        $\boldsymbol{s}_{t+1}, \eta_{t+1} \leftarrow \arg\min_{\boldsymbol{s} \in S_k^p, \eta > 0} \eta \geq c(\boldsymbol{s}_i) + \nabla c(\boldsymbol{s}_i)^T (\boldsymbol{s} - \boldsymbol{s}_i) \, \forall i \in [t]$
        $t \leftarrow t + 1$
    end while
    $\boldsymbol{s}^* \leftarrow \boldsymbol{s}_t$
    $i \leftarrow 1$
    for $i < n$ do
        $\boldsymbol{x}_i \leftarrow \boldsymbol{B}_{\boldsymbol{s}^*} (\boldsymbol{B}_{\boldsymbol{s}^*}^T \boldsymbol{W}_i \boldsymbol{B}_{\boldsymbol{s}^*})^{-1} \overline{\boldsymbol{a}}_i$      # $\boldsymbol{B}_{\boldsymbol{s}^*}$ is $\boldsymbol{B}$ submatrix with $\boldsymbol{s}^*$ columns
    end for
    return $\boldsymbol{X}$                                  # Return the filled matrix $\boldsymbol{X}$
end procedure
```

We next outline how to implement the algorithm for improved scalability and speed.

3.1. Implementation of CutPlanes

We introduce

$$\alpha_i(\boldsymbol{s}) = \overline{\boldsymbol{a}}_i^T \left(\boldsymbol{I}_m + \gamma \boldsymbol{W}_i \left(\sum_{j=1}^{p} s_j \boldsymbol{K}_j \right) \boldsymbol{W}_i \right)^{-1} \overline{\boldsymbol{a}}_i, \quad i = 1, \ldots, n. \tag{7}$$

The function $c(\boldsymbol{s})$ in (2) can be expressed as

$$c(\boldsymbol{s}) = \frac{1}{n} \sum_{i=1}^{n} \alpha_i(\boldsymbol{s}). \tag{8}$$

Applying the Matrix Inversion Lemma Woodbury (1949) we have

$$\alpha_i(\boldsymbol{s}) = \overline{\boldsymbol{a}}_i^T \left(I_m - \boldsymbol{V} \left(\frac{\boldsymbol{I}_k}{\gamma} + \boldsymbol{V}^T \boldsymbol{W}_i \boldsymbol{V} \right)^{-1} \boldsymbol{V}^T \right) \overline{\boldsymbol{a}}_i, \tag{9}$$

where $\boldsymbol{V} \in \mathbb{R}^{k \times m}$ is the feature matrix formed by the k columns of $\boldsymbol{B}$ such that $s_j = 1$. Note that in order to compute $\alpha_i(\boldsymbol{s})$ using Eq. (7) we need to invert an $m \times m$ matrix, while from Eq. (9) we need to invert a $k \times k$ matrix $\frac{\boldsymbol{I}_k}{\gamma} + \boldsymbol{V}^T \boldsymbol{W}_i \boldsymbol{V}$, which is much smaller. Thus, we can compute $\alpha_i(\boldsymbol{s})$ in floating point complexity of $O(m^2k + k^3)$ rather than $O(m^3)$ from Eq. (7). In real world applications $m \gg k$ leading to a considerable speedup.

Further, we observe that $\boldsymbol{W}_i$ is a diagonal matrix with binary entries, so we can calculate $\boldsymbol{W}_i\boldsymbol{V}$ by zeroing out the columns in which $(\boldsymbol{W}_i)_{jj} = 0$. This allows $\boldsymbol{V}^T\boldsymbol{W}_i\boldsymbol{V}$ to be calculated in $O(mk^2)$ instead of $O(m^2k)$. Although this does not reduce the asymptotic complexity (as the pre-multiplication and post-multiplication of $\boldsymbol{V}$ and $\boldsymbol{V}^T$ necessarily requires a $O(m^2k)$ complexity), due to the high number of $\alpha_i(\boldsymbol{s})$ required in calculating the cutting plane, this provides another substantial speedup.

To calculate the derivative $\nabla(\alpha_i(\boldsymbol{s}))$, we follow the same derivation as detailed in Bertsimas and Van Parys (2017). We write

$$\alpha_i(\boldsymbol{s}) = \overline{\boldsymbol{a}}_i^T \left(I_m - \boldsymbol{V} \left(\frac{\boldsymbol{I}_k}{\gamma} + \boldsymbol{V}^T \boldsymbol{W}_i \boldsymbol{V} \right)^{-1} \boldsymbol{V}^T \right) \overline{\boldsymbol{a}}_i = \overline{\boldsymbol{a}}_i^T \boldsymbol{\gamma}_i(\boldsymbol{s}),$$

and, by algebraic manipulations, we obtain

$$(\nabla \alpha_i(\boldsymbol{s}))_j = -\gamma \left(\boldsymbol{B}^T \boldsymbol{W}_i \boldsymbol{\gamma}_i(\boldsymbol{s}) \right)_j^2.$$

Therefore, most of the calculation for the objective can be reused for the derivative, and exploiting the structure of $\boldsymbol{W}_i$, the complexity of the derivative calculation is only $O(pm)$. Thus, the total complexity of generating a full cutting plane is:

$$O(nm^2k + nk^3 + npm). \tag{10}$$

4. The Stochastic Cutting Planes Algorithm

In this section, we introduce the stochastic cutting planes algorithm that enables us to scale the algorithm CutPlanes to very high dimensions of n, m. Specifically, at each instance where cutting planes are generated, we randomly select r rows and s columns of $\boldsymbol{A}$. We denote by $\boldsymbol{V}_s \in \mathbb{R}^{p\times s}$ the feature matrix with the s selected columns, $\boldsymbol{W}_i^s \in \mathbb{R}^{s\times s}$ diagonal matrix and $\overline{\boldsymbol{a}}_i^s \in \mathbb{R}^{s\times 1}$ that are defined similarly as before corresponding to the s selected columns. Then, the approximate function we want to minimize is

$$\tilde{c}_r(\boldsymbol{s}) = \frac{1}{r}\sum_{i=1}^{r} \tilde{\alpha}_i^s(\boldsymbol{s}), \tag{11}$$

where

$$\tilde{\alpha}_i^s(\boldsymbol{s}) = \overline{\boldsymbol{a}}_i^{sT}\left(I_s - \boldsymbol{V}_s\left(\frac{\boldsymbol{I}_k}{\gamma} + \boldsymbol{V}_s^T\boldsymbol{W}_i^s\boldsymbol{V}_s\right)^{-1}\boldsymbol{V}_s^T\right)\overline{\boldsymbol{a}}_i^s.$$

With this, the complexity of the cutting plane is now

$$O(rs^2k + rk^3 + prs).$$

To select the appropriate r and s, we use the observation from Candès and Tao (2010), who show that to complete a square $N\times N$ matrix of rank k, we need at least $O(kN\log N)$ elements. Assuming an average missing rate of μ, the expected number of known (not missing) elements if we sample r rows and s columns from matrix $\boldsymbol{A}$ will be $r\cdot s\cdot(1-\mu)$. Using $N^2 = n\cdot m$, we need

$$r\cdot s\cdot(1-\mu) \geq c\cdot k\sqrt{nm}\log(\sqrt{nm}),$$

where c is a numerical constant that we selected experimentally to be $c = \frac{1}{8}$. As the CutPlanes algorithm scales linearly in n and quadratically in m (see 10), we select s as small as possible. We thus selected

$$s = \min(s_0, m), \qquad r = \min\left(\frac{ck\sqrt{mn}\log(\sqrt{mn})}{(1-\mu)\min(s_0, m)}, n\right), \tag{12}$$

where s_0 is some appropriate lower bound for the minimum number of columns chosen - empirically we have selected $s_0 = 500$ in our experiments. The stochastic cutting plane algorithm, we call OptComplete is as follows.

Algorithm 2 Stochastic Cutting-plane algorithm for matrix completion with side information.

```
procedure OptComplete($\boldsymbol{A}, \boldsymbol{B}$)          # masked matrix $\boldsymbol{A}$, and feature matrix $\boldsymbol{B}$
    $t \leftarrow 1$
    $\boldsymbol{s}_1 \leftarrow$ warm start                         # Heuristic Warm Start
    $\eta \leftarrow 0$                                       # Initialize feasible solution variable
    $s \leftarrow \min(s_0, m)$                               # $s_0$ is pre-determined
    $r \leftarrow \min\left(\frac{ck\sqrt{mn}\log(\sqrt{mn})}{(1-\mu)\min(s_0,m)}, n\right)$
    while $\eta_t < c(\boldsymbol{s}_t)$ do                            # While the current solution is not optimal
        $\boldsymbol{s}_{t+1}, \eta_{t+1} \leftarrow \arg\min_{\boldsymbol{s}\in S_k^p, \eta>0} \eta \geq \tilde{c}_r(\boldsymbol{s}_i) + \nabla \tilde{c}_r(\boldsymbol{s}_i)^T(\boldsymbol{s}-\boldsymbol{s}_i)\ \forall i \in [t]$
                                                          # We randomly sample $r$ new rows
                                                          # and $s$ new columns to calculate
                                                          # $\tilde{c}_r(\boldsymbol{s}_i)$ and $\nabla \tilde{c}_r(\boldsymbol{s}_i)$
        $t \leftarrow t+1$
    end while
    $\boldsymbol{s}^* \leftarrow \boldsymbol{s}_t$
    $i \leftarrow 1$
    for $i < n$ do
        $\boldsymbol{x}_i \leftarrow \boldsymbol{B}_{\boldsymbol{s}^*}(\boldsymbol{B}_{\boldsymbol{s}^*}^T \boldsymbol{W}_i \boldsymbol{B}_{\boldsymbol{s}^*})^{-1}\overline{\boldsymbol{a}}_i$    # $\boldsymbol{B}_{\boldsymbol{s}^*}$ is $\boldsymbol{B}$ submatrix with $\boldsymbol{s}^*$ columns
    end for
    return $\boldsymbol{X}$                                  # Return the filled matrix $\boldsymbol{X}$
end procedure
```

The OptComplete algorithm enjoys theoretical guarantees. Let us define the following:

Definition 1 *The* ***convexity parameter*** *a of the set of functions $\tilde{\alpha}_i^s(\boldsymbol{s})$ is defined as the largest positive number for which the following statement is true:*

$$\tilde{\alpha}_i^s(\boldsymbol{s}) \geq \tilde{\alpha}_i^s(\boldsymbol{s}_0) + \nabla \tilde{\alpha}_i^s(\boldsymbol{s}_0)^T(\boldsymbol{s}-\boldsymbol{s}_0) + \frac{a^2}{2}(\boldsymbol{s}-\boldsymbol{s}_0)^T(\boldsymbol{s}-\boldsymbol{s}_0) \quad \forall \boldsymbol{s}, \boldsymbol{s}_0\ \forall i \tag{13}$$

Note we can always find such $a > 0$, as $\tilde{\alpha}_i^s(\boldsymbol{s})$ are strongly convex functions for all i. Then, the following theorem provides a theoretical lower bound for the probability that OptComplete finds an optimal solution:

Theorem 2 *For the matrix completion problem (1), we assume the rows of $\boldsymbol{U}$, are independent and identically distributed (iid) from a probability distribution with finite third moment and the rows of $\boldsymbol{B}$ are iid draws from a p-dimensional sub-Gaussian distribution. OptComplete satisfies:*

(a) *OptComplete terminates in a finite number of steps C.*

(b) *OptComplete finds an optimal solution of (1) with probability at least $1 - \frac{KC}{a^4}\left(\frac{1}{r} + \frac{1}{s^{1/2}}\right)$ where K is a constant independent of C, r, s, a, and a is the convexity parameter of the functions $\tilde{\alpha}_i^s(\boldsymbol{s})$.*

The proof of the theorem is in Appendix A.

4.1. Warm Starts

For warm starts, we utilize ideas from Bertsimas and Van Parys (2017) to find warm starts by solving a dual problem. Specifically, we extend in a straightforward manner the method in Bertsimas and Van Parys (2017) to show that (1) admits a continuous relaxation in the dual of the following form:

$$\begin{aligned}
&\max -\sum_{i=1}^{n}\left(\frac{1}{2}\boldsymbol{\alpha}^T\boldsymbol{\alpha} + \boldsymbol{W}_i\boldsymbol{Y}^T\boldsymbol{\alpha} - \boldsymbol{1}^T\boldsymbol{u} - kt\right) \\
&\text{s.t. } \boldsymbol{\alpha} \in \mathbb{R}^m, \quad t \in \mathbb{R}, \quad \boldsymbol{u} \in \mathbb{R}_+^p, \qquad (14) \\
&\frac{2n}{\gamma}u_j \geq \sum_{i=1}^{n}\left(\boldsymbol{\alpha}^T\boldsymbol{W}_i\boldsymbol{K}_j\boldsymbol{W}_i^T\boldsymbol{\alpha} - \frac{2}{\gamma}t\right), \qquad \forall j \in 1, \cdots, p
\end{aligned}$$

Then we utilize the following procedure to provide the warmstart:

- Given the matrix $\boldsymbol{A}$ and the feature matrix $\boldsymbol{B}$, we randomly sample r rows and s columns from both matrices in accordance with (12) to formulate a smaller problem $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{B}}$.
- We solve the Problem (14) on the smaller input $\tilde{\boldsymbol{A}}$ and $\tilde{\boldsymbol{B}}$, and return the solution to the original problem with $\boldsymbol{A}$ and $\boldsymbol{B}$ as the warmstart.

5. Synthetic Data Experiments

We assume that the matrix $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{V} + \boldsymbol{E}$, where $\boldsymbol{U} \in \mathbb{R}^{n\times k}$, $\boldsymbol{V} \in \mathbb{R}^{k\times m}$, and $\boldsymbol{E}$ is an error matrix with individual elements sampled from $N(0, 0.01)$. We sample the elements of $\boldsymbol{U}$ and $\boldsymbol{V}$ from a uniform distribution of $[0, 1]$, and then randomly select a fraction μ to be missing. We formulate the feature matrix $\boldsymbol{B}$ by combining $\boldsymbol{V} \in \mathbb{R}^{k\times m}$ with a confounding matrix $\boldsymbol{Z} \in \mathbb{R}^{(p-k)\times m}$ that contains unnecessary factors sampled similarly from the Uniform $[0, 1]$ distribution. We run OptComplete on a server with 16 CPU cores. For each combination (m, n, p, k, μ), we ran 10 tests and report the median value for every statistic.

We report the following statistics with $\boldsymbol{s}^*$ being the ground-truth factor vector, and $\overline{\boldsymbol{s}}$ the estimated factor vector.

- n, m - the dimensions of $\boldsymbol{A}$.
- p - the number of features in the feature matrix.
- k - the true number of features.
- T - the total time taken for the algorithm.
- μ - The fraction of missing entries in $\boldsymbol{A}$.
- $A\%$ - the percentage of factors in the ground truth we identify correctly:

$$A\% = \frac{\text{Supp}(\boldsymbol{s}^*) \cap \text{Supp}(\overline{\boldsymbol{s}})}{\text{Supp}(\boldsymbol{s}^*)}.$$

- $F\%$ - the percentage of factors recovered that are not present in the ground truth:

$$F\% = \frac{\text{Supp}(\overline{\boldsymbol{s}}) \setminus \text{Supp}(\boldsymbol{s}^*)}{\text{Supp}(\overline{\boldsymbol{s}})}.$$

- MAPE - the Mean Absolute Percentage Error (MAPE) for the retrieved matrix $\hat{\boldsymbol{A}}$:

$$\text{MAPE} = \frac{1}{|\mathcal{S}|} \sum_{(i,j)\in\mathcal{S}} \frac{\hat{A}_{ij} - A_{ij}}{A_{ij}},$$

where $\mathcal{S} = \Omega^c$ is the set of missing data in $\boldsymbol{A}$.

We compare OptComplete using the choice of parameters in (12) and calling the state of the art commerical solver Gurobi 8.0 to solve the integer optimization subproblems with:

- IMC by Natarajan and Dhillon (2014) - This algorithm is a well-accepted benchmark for testing Inductive Matrix Completion algorithms.
- SoftImpute-ALS (SIALS) by Hastie et al. (2015) - This is widely recognized as a state-of-the-art matrix completion method without feature information. It has among the best scaling behavior across all classes of matrix completion algorithms as it utilizes fast alternating least squares to achieve scalability.

We randomly selected 20% of those elements masked to serve as a validation set. The regularization parameter γ of OptComplete, the rank parameter of IMC and the penalization parameter λ of IMC and SIALS are selected using the validation set.

The results are separated into sections below. The first five sections modify one single variable out of n, m, p, k, μ to investigate OptComplete's scalability, where the leftmost column indicates the variable modified. The last section compares the three algorithms scalability for a variety of parameter combinations.

	n	m	p	k	μ%	**OptComplete**				**IMC**		**SIALS**	
						T	A%	F%	**MAPE**	T	**MAPE**	T	**MAPE**
μ	100	100	15	5	20%	0.3s	100%	0%	0.1%	0.03s	0.01%	0.02s	0.3%
	100	100	15	5	50%	0.4s	100%	0%	0.02%	0.07s	0.5%	0.03s	0.9%
	100	100	15	5	80%	0.6s	100%	0%	0.03%	0.09s	1.3%	0.06s	5.6%
	100	100	15	5	95%	0.9s	100%	0%	0.04%	0.12s	12.1%	0.12s	7.4%
n	100	100	15	5	50%	0.4s	100%	0%	0.02%	0.07s	0.5%	0.03s	0.9%
	10^3	100	15	5	50%	2.7s	100%	0%	0.01%	0.6s	0.4%	0.1s	0.2%
	10^4	100	15	5	50%	6.4s	100%	0%	0.004%	4.5s	0.3%	6.5s	0.5%
	10^5	100	15	5	50%	15.3s	100%	0%	0.003%	32.7s	0.1%	38s	3.0%
m	100	100	15	5	50%	0.4s	100%	0%	0.02%	0.07s	0.5%	0.03s	0.9%
	100	10^3	15	5	50%	1.6s	100%	0%	0.01%	0.8s	0.3%	0.1s	0.5%
	100	10^4	15	5	50%	8.6s	100%	0%	0.004%	6.2s	0.2%	0.8s	0.3%
	100	10^5	15	5	50%	62.4s	100%	0%	0.002%	56.2s	0.1%	12.7s	0.8%
p	100	100	15	5	50%	0.4s	100%	0%	0.02%	0.07s	0.5%	0.03s	0.9%
	100	100	50	5	50%	1.0s	100%	0%	0.02%	0.3s	0.6%	0.03s	0.9%
	100	100	200	5	50%	2.6s	100%	0%	0.02%	1.9s	0.8%	0.03s	0.9%
	100	100	10^3	5	50%	16.5s	100%	0%	0.02%	10.4s	1.0%	0.03s	0.9%
k	100	100	15	5	50%	0.4s	100%	0%	0.02%	0.07s	0.5%	0.03s	0.9%
	100	100	50	10	50%	10.2s	100%	0%	0.06%	0.20s	1.2%	0.1s	0.8%
	100	100	50	20	50%	198s	100%	0%	0.07%	0.35s	2.1%	0.21s	1.0%
	100	100	50	30	50%	632s	100%	0%	0.09%	0.5s	3.3%	0.43s	2.8%
	100	100	15	5	95%	0.9s	100%	0%	0.04%	0.12s	12.1%	0.12s	7.4%
	10^3	10^3	50	5	95%	3.6s	100%	0%	0.006%	4.6s	4.7%	2.8s	12.5%
	10^4	10^3	100	5	95%	28.4s	100%	0%	0.002%	18s	2.5%	20.7s	12.6%
	10^5	10^3	200	10	95%	272s	100%	0%	0.001%	295s	1.7%	420s	4.6%
	10^5	10^4	200	10	95%	1240s	100%	0%	0.001%	1750s	0.5%	4042s	4.1%
	10^6	10^4	200	10	95%	4412s	100%	0%	0.001%	13750s	0.3%	25094s	2.5%
	10^6	10^5	200	10	95%	19854s	100%	0%	0.001%	N/A	N/A	N/A	N/A

Table 1 Comparison of OptComplete, IMC and SIALS on synthetic data. N/A means the algorithm did not complete running in 20 hours, corresponding to 72000 seconds.

Overall, we see that OptComplete achieves near-exact retrieval on all datasets evaluated. For the realistic data sizes in the last panel, OptComplete achieves near-exact retrieval, while requiring less time than IMC and SIALS at the same time. At the scale of $n = 10^6$ and $m = 10^4$, OptComplete is triple the speed of IMC and over 80% faster than SIALS. At the scale of $n = 10^6$ and $m = 10^5$, IMC and SIALS did not finish running within 20 hours, while OptComplete completed in just over 3 hours. We analyze the scaling of OptComplete as a function of:

1. μ - The algorithm is able to retrieve the exact factors used even with 95% of missing data. The time scaling behavior was also very similar to that of SoftImpute.

2. n - The algorithm has good scalability in n, reflecting its $O(n^{\frac{1}{2}}\log n)$ type complexity. This allows the algorithm to support matrices with n in the 10^6 range. Its scaling behavior is superior to both IMC and SoftImpute-ALS.

3. m - The algorithm has good scalability in m, similar to n, which is expected as it has the same complexity dependency. Note that the algorithm is not fully symmetric with respect to m and n even though the asymptotic complexity is the same as we try to minimize the dependency of m due to its quadratic dependency as explored in (12). It is comparable with SoftImpute-ALS and IMC.

4. p - The algorithm scales relatively well in p, which reflects the performance of the Gurobi solver. We empirically observe that Gurobi is generating roughly $O(p)$ cutting planes. Thus, as each cutting plane is $O(p)$, we expect $O(p^2)$ dependence. However, the linear timing scaling here reflects the high quality of the warm start solutions, which is short cutting most of the work. Thus, OptComplete achieves similar scaling behavior as IMC in p. Note here the SoftImpute algorithm does not utilize feature information and thus a change in p does not affect the algorithm's run speed.

5. k - The algorithm does not scale very well in k. We empirically observe that Gurobi solver is roughly generating $O(k)$ cutting planes and each cutting plane has cubic dependence on k. It appears that SoftImpute and IMC almost have a linear scaling behavior. However, in most applications, such as recommendation systems or low-rank retrieval, k is usually kept very low ($k \leq 30$), so this is not a particular concern. Moreover, with realistic n and m, the warm start usually will pre-solve the problem before Gurobi even starts.

6. Real Dataset Experiments

In this section, we report on the performance of OptComplete on the Netflix Prize dataset. This dataset was released in a competition to predict ratings of customers on unseen movies, given over

10 million ratings scattered across $500,000$ people and $16,000$ movies. Thus, when presented in a matrix $\boldsymbol{A}$ where A_{ij} represents the rating of individual i on movie j, the goal is to complete the matrix $\boldsymbol{A}$ under a low-rank assumption.

The feature matrix $\boldsymbol{B}$ of OptComplete is constructed using data from the TMDB Database, and covers 59 features that measure geography, popularity, top actors/actresses, box office, runtime, genre and more. The full list of 59 features is contained in Appendix B.

For this experiment, we included movies where all 59 features are available, and people who had at least 5 ratings present. This gives a matrix of $471,268$ people and $14,538$ movies. The slight reduction of size from the original data is due to the lack of features for about $2,000$ niche movies. To observe the scalability of OptComplete, we created five data sets:

1. Base - $\boldsymbol{A}_1$ has dimensions $3,923 \times 103$.
2. Small - $\boldsymbol{A}_2$ has dimensions $18,227 \times 323$.
3. Medium - $\boldsymbol{A}_3$ has dimensions $96,601 \times 788$.
4. Large - $\boldsymbol{A}_4$ has dimensions $471,268 \times 1760$.
5. Full - $\boldsymbol{A}$ has dimensions $471,268 \times 14,538$.

These sizes are constructed such that the total number of elements in $\boldsymbol{A}$ in the successive sizes are approximately different by approximately an order of magnitude.

For each individual matrix, we uniformly randomly withhold 20% of the ratings as a test set $\mathcal{S}$, and use the remaining 80% of ratings to impute a complete matrix $\hat{\boldsymbol{A}}$ - we perform cross-validation on the appropriate hyperparameters. Then, we report MAPE.

For comparison, we again use IMC and SIALS. We set the maximum rank of SIALS to be k - the rank optimized for in OptComplete. The results are listed below:

n	m	p	k	μ%	**OptComplete**		**IMC**		**SIALS**	
					T	**MAPE**	T	**MAPE**	T	**MAPE**
3,923	103	59	5	92.6%	3.7s	29.6%	0.6s	34.2%	0.3s	31.2%
18,227	323	59	5	94.8%	15.1s	22.2%	5.2s	29.1%	4.1s	24.1%
96,601	788	59	5	94.2%	69.3s	20.7%	38.1s	28.7%	30.4s	21.3%
471,268	1,760	59	5	93.6%	380s	18.6%	460s	24.6%	430s	19.8%
471,268	14,538	59	5	94.1%	1667s	15.3%	3921s	21.5%	5300s	16.7%

Table 2 Comparison of methods on Netflix data for $k = 5$.

n	m	p	k	μ%	**OptComplete**		**IMC**		**SIALS**	
					T	**MAPE**	T	**MAPE**	T	**MAPE**
3,923	103	59	10	92.6%	26s	30.1%	1.4s	36.7%	0.8s	35.8%
18,227	323	59	10	94.8%	122s	24.2%	12.5s	32.5%	7.0s	28.9%
96,601	788	59	10	94.2%	413s	22.4%	84.2s	29.6%	50.7s	22.8%
471,268	1,760	59	10	93.6%	2574s	20.5%	1022s	24.8%	870s	20.7%
471,268	14,538	59	10	94.1%	12865s	19.3%	8704s	23.1%	10240s	20.0%

Table 3 Comparison of methods on Netflix data for $k = 10$.

We can see that OptComplete outperforms both IMC and SoftImpute-ALS in accuracy across the datasets under different k; furthermore in the two largest datasets OptComplete runs faster under $k = 5$ and comparable under $k = 10$. Here we see that an increase from $k = 5$ to $k = 10$ actually decreased out-of-sample performance as additional factors are actually not very helpful in predictive customer tastes. The decline for OptComplete and IMC were especially higher due to the fact that the possible factors are fixed and thus an increase in the number of factors caused some non-predictive factors to be included.

For the $k = 5$ case, OptComplete identified the following as the top factors that influences an individual's rating:

- IMDB Rating
- Genre: Drama
- Released within last 10 years
- Number of Top 100 Actors

- Produced in US

These factors provide an intuitive explanation of the individual ratings of each customer in terms of a small number of factors, while exceeding the high predictive accuracy of SoftImpute.

7. Conclusions

We have presented OptComplete, a scalable algorithm to retrieve a low-rank matrix in the presence of side information. Compared with state of the art algorithms for matrix completion OptComplete exceeds current benchmarks on both scalability and accuracy and provides insight on the factors that affect the ratings.

Appendix A: Proof of Convergence of OptComplete

In this section, we provide the proof of Theorem 2. We first introduce a lemma, proven in Dhillon et al. (2013).

Lemma 1 *If the rows of the feature matrix $\boldsymbol{B}$ are iid draws from a p-dimensional sub-Gaussian distribution, then we have:*

$$\mathbb{E}[\tilde{\alpha}_i^s(\boldsymbol{s})] = \alpha_i(\boldsymbol{s}), \qquad \mathbb{E}[\nabla\tilde{\alpha}_i^s(\boldsymbol{s})] = \nabla\alpha_i(\boldsymbol{s}) \tag{15}$$

$$|\tilde{\alpha}_i^s(\boldsymbol{s}) - \alpha_i(\boldsymbol{s})| \leq \frac{K_1}{\boldsymbol{s}^{1/2}} \tag{16}$$

$$|\nabla\tilde{\alpha}_i^s(\boldsymbol{s})^T\boldsymbol{a} - \nabla\alpha_i(\boldsymbol{s})^T\boldsymbol{a}| \leq \frac{K_2\|\boldsymbol{a}\|^2}{\boldsymbol{s}^{1/2}}, \quad \forall \boldsymbol{a} \tag{17}$$

Proof of Theorem 2

(a) OptComplete is a specific implementation of the outer approximation algorithm. R and S (1994) have proven that it always terminates in finite number of steps C.

(b) Given that we assumed that the problem is feasible (we assumed that $\boldsymbol{u}_i$'s exist and follow some distribution P), for OptComplete to not return an optimal solution, it would have to cut it off during the course of its execution.

Let $\boldsymbol{s}^*$ be an optimal solution for Problem (1). Let $\boldsymbol{s}_t$ be an optimal solution at the t-th iteration of OptComplete, $t \in [C]$. The cutting plane constraint for OptComplete at the point of an optimal solution $\boldsymbol{s}^*$ is

$$\eta_t \geq \tilde{c}_r(\boldsymbol{s}_t) + \nabla \tilde{c}_r(\boldsymbol{s}_t)^T(\boldsymbol{s}^* - \boldsymbol{s}_t).$$

If $c(\boldsymbol{s}^*) < \eta_t$, then $\boldsymbol{s}^*$ will be cut off, and OptComplete will not find $\boldsymbol{s}^*$. Applying the definition of the convexity parameter (13) and letting $\|\boldsymbol{s}^* - \boldsymbol{s}_t\| = \theta_t$ we obtain

$$c(\boldsymbol{s}^*) \geq c(\boldsymbol{s}_t) + \nabla c(\boldsymbol{s}_t)^T(\boldsymbol{s}^* - \boldsymbol{s}_t) + \frac{\theta_t^2 a^2}{2}. \tag{18}$$

Therefore, if

$$c(\boldsymbol{s}_t) + \nabla c(\boldsymbol{s}_t)^T(\boldsymbol{s}^* - \boldsymbol{s}_t) + \frac{\theta_t^2 a^2}{2} \leq c(\boldsymbol{s}^*) < \tilde{c}_r(\boldsymbol{s}_t) + \nabla \tilde{c}_r(\boldsymbol{s}_t)^T(\boldsymbol{s}^* - \boldsymbol{s}_t),$$

or equivalently if

$$\zeta_t := [\tilde{c}_r(\boldsymbol{s}_t) - c(\boldsymbol{s}_t)] + [\nabla \tilde{c}_r(\boldsymbol{s}_t) - \nabla c(\boldsymbol{s}_t)]^T(\boldsymbol{s}^* - \boldsymbol{s}_t) > \frac{\theta_t^2 a^2}{2}, \tag{19}$$

then OptComplete will not find $\boldsymbol{s}^*$. Therefore, for OptComplete to succeed, ζ_t should satisfy $\zeta_t \leq \theta_t^2 a^2/2$ for all $t \in [C]$.

Let $F =$ the event that OptComplete succeeds in finding $\boldsymbol{s}^*$. Then,

$$\mathrm{P}(F) \geq \mathrm{P}\left(\zeta_t \leq \frac{\theta_t^2 a^2}{2} \;\; t \in [C]\right).$$

Since at each step of OptComplete we randomly sample r new rows and s new columns, the events $\zeta_t \leq \frac{\theta_t^2 a^2}{2}$ are independent for different $t \in [C]$, and hence

$$\mathrm{P}(F) \geq \prod_{t=1}^{C}\left(1 - \mathrm{P}\left(\zeta_t > \frac{\theta_t^2 a^2}{2}\right)\right).$$

In order to calculate $\mathrm{P}(\zeta_t \geq \theta_t^2 a^2/2)$ we first calculate the mean and variance of ζ_t. Using Eqs. (8) and (11) and applying Eq. (15) we obtain

$$\mathbb{E}[\tilde{c}_r(\boldsymbol{s}_t) - c(\boldsymbol{s}_t)] = 0, \qquad \mathbb{E}[\nabla \tilde{c}_r(\boldsymbol{s}_t) - \nabla c(\boldsymbol{s}_t)] = 0, \tag{20}$$

. leading to $\mathbb{E}[\zeta_t]=0$. To calculate the variance, we have

$$\begin{aligned}
\mathbb{V}[\tilde{c}_r(\boldsymbol{s}_t)-c(\boldsymbol{s}_t)] &= \mathbb{V}\left[\frac{1}{r}\sum_{i=1}^{r}\tilde{\alpha}_i^s(\boldsymbol{s}_t)-\frac{1}{n}\sum_{i=1}^{n}\alpha_i(\boldsymbol{s}_t)\right]\\
&= \mathbb{V}\left[\frac{1}{r}\sum_{i=1}^{r}\tilde{\alpha}_i^s(\boldsymbol{s}_t)-\frac{1}{r}\sum_{i=1}^{r}\alpha_i(\boldsymbol{s})\right]+\mathbb{V}\left[\frac{1}{r}\sum_{i=r}^{n}\alpha_i(\boldsymbol{s}_t)-\frac{1}{n}\sum_{i=1}^{n}\alpha_i(\boldsymbol{s}_t)\right]\\
&\quad \text{(rows and columns are sampled independently)}\\
&\le \frac{K_1}{s^{1/2}}+\mathbb{V}\left[\frac{1}{r}\sum_{i=1}^{r}\alpha_i(\boldsymbol{s}_t)-\frac{1}{n}\sum_{i=1}^{n}\alpha_i(\boldsymbol{s}_t)\right] \quad \text{(From (16))}\\
&\le \frac{K_1}{s^{1/2}}+\frac{K_2}{r}\\
&\quad \text{(Using the convergence rate of sample mean, Berry (1941))}\\
&\le K_0\left(\frac{1}{r}+\frac{1}{s^{1/2}}\right).
\end{aligned}$$

A similar derivation and using (17) leads to

$$\mathbb{V}[(\nabla\tilde{c}_r(\boldsymbol{s}_t)-\nabla c(\boldsymbol{s}_t))^T(\boldsymbol{s}^*-\boldsymbol{s}_t)]\le K_3\theta_t^2\left(\frac{1}{r}+\frac{1}{s^{1/2}}\right). \tag{21}$$

Therefore,

$$\begin{aligned}
\mathbb{V}(\zeta_t) &= \mathbb{V}\left[\tilde{c}_r(\boldsymbol{s}_t)-c(\boldsymbol{s}_t)+\nabla\tilde{c}_r(\boldsymbol{s}_t)-\nabla c(\boldsymbol{s}_t)]^T(\boldsymbol{s}^*-\boldsymbol{s}_t)\right]\\
&\le 2\mathbb{V}\left[\tilde{c}_r(\boldsymbol{s}_t)-c(\boldsymbol{s}_t)\right]+2\mathbb{V}\left[\nabla\tilde{c}_r(\boldsymbol{s}_t)-\nabla c(\boldsymbol{s}_t)\right]^T(\boldsymbol{s}^*-\boldsymbol{s}_t)\Big],
\end{aligned}$$

since for every two random variables H,G we have $\mathbb{V}(H+G)\le 2\mathbb{V}(H)+2\mathbb{V}(G)$. Thus,

$$\begin{aligned}
\mathbb{V}(\zeta_t) &\le 2K_0\left(\frac{1}{r}+\frac{1}{s^{1/2}}\right)+2K_3\theta_t^2\left(\frac{1}{r}+\frac{1}{s^{1/2}}\right)\\
&\le 2(K_0+K_3)\theta_t^2\left(\frac{1}{r}+\frac{1}{s^{1/2}}\right) \quad (\theta_t\ge 1)\\
&\le K_4\theta_t^2\left(\frac{1}{r}+\frac{1}{s^{1/2}}\right),
\end{aligned}$$

with $K_4=2(K_0+K_3)$. Applying Chebeshev's inequality, we have

$$\begin{aligned}
\mathrm{P}\left(\zeta_t>\frac{\theta_t^2a^2}{2}\right) &\le \frac{\mathbb{V}(\zeta_t)}{\left(\frac{\theta_t^2a^2}{2}\right)^2}\\
&\le \frac{4K_4\theta_t^2}{a^4\theta_t^4}\left(\frac{1}{r}+\frac{1}{s^{1/2}}\right)\\
&= \frac{K}{a^4}\left(\frac{1}{r}+\frac{1}{s^{1/2}}\right) \quad (\theta_t\ge 1).
\end{aligned}$$

We thus have

$$\begin{aligned}
\mathrm{P}(F) &\ge \left(1-\frac{K}{a^4}\left(\frac{1}{r}+\frac{1}{s^{1/2}}\right)\right)^C\\
&\ge 1-\frac{KC}{a^4}\left(\frac{1}{r}+\frac{1}{s^{1/2}}\right).
\end{aligned}$$

Appendix B: List of Features Used in the Netflix Problem

- 24 Indicator Variables for Genres: Action, Adventure, Animation, Biography, Comedy, Crime, Documentary, Drama, Family, Fantasy, Film Noir, History, Horror, Music, Musical, Mystery, Romance, Sci-Fi, Short, Sport, Superhero, Thriller, War, Western
- 5 Indicator Variables for Release Date: Within last 10 years, Between 10-20 years, Between 20-30 years, Between 30-40 years, Between 40-50 Years
- 6 Indicator Variables for Top Actors/Actresses defined by their Influence Score at time of release: Top 100 Actors, Top 100 Actresses, Top 250 Actors, Top 250 Actresses, Top 1000 Actors, Top 1000 Actresses
- IMDB Rating
- Number of Reviews
- Total Production Budget
- Total Runtime
- Total Box Office Revenue
- Indicator Variable for whether it is US produced
- 11 Indicator Variables for Month of Year Released (January removed to prevent multicollinearity)
- Number of Original Music Score
- Number of Male Actors
- Number of Female Factors
- 3 Indicator Variables for Film Language: English, French, Japanese
- Constant

References

Beck A, Teboulle M (2009) A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences* 2(1):183–202.

Berry AC (1941) The accuracy of the gaussian approximation to the sum of independent variates. *Transactions of the American Mathematical Society* 49(1):122–136.

Bertsimas D, Copenhaver MS (2018) Characterization of the equivalence of robustification and regularization in linear, median, and matrix regression. *European Journal of Operations Research* 270:931–942.

Bertsimas D, Van Parys B (2017) Sparse high-dimensional regression: Exact scalable algorithms and phase transitions. *arXiv preprint arXiv:1709.10029* .

Boyd S, El Ghaoui L, Feron E, Balakrishnan V (1994) *Linear matrix inequalities in system and control theory*, volume 15 (SIAM).

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

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

Chen Y, Bhojanapalli S, Sanghavi S, Ward R (2014a) Coherent matrix completion. *International Conference on Machine Learning*, 674–682.

Chen Y, Jalali A, Sanghavi S, Xu H (2014b) Clustering partially observed graphs via convex optimization. *The Journal of Machine Learning Research* 15(1):2213–2238.

Chiang KY, Hsieh CJ, Dhillon IS (2015) Matrix completion with noisy side information. *Advances in Neural Information Processing Systems*, 3447–3455.

Chiang KY, Hsieh CJ, Natarajan N, Dhillon IS, Tewari A (2014) Prediction and clustering in signed networks: a local to global perspective. *The Journal of Machine Learning Research* 15(1):1177–1213.

Dhillon P, Lu Y, Foster DP, Ungar L (2013) New subsampling algorithms for fast least squares regression. *Advances in Neural Information Processing Systems*, 360–368.

Duran MA, Grossmann IE (1986) An outer-approximation algorithm for a class of mixed-integer nonlinear programs. *Mathematical Programming* 36(3):307–339.

Farhat MR, Shapiro BJ, Kieser KJ, Sultana R, Jacobson KR, Victor TC, Warren RM, Streicher EM, Calver A, Sloutsky A, et al. (2013) Genomic analysis identifies targets of convergent positive selection in drug-resistant mycobacterium tuberculosis. *Nature Genetics* 45(10):1183.

Hastie T, Mazumder R, Lee JD, Zadeh R (2015) Matrix completion and low-rank svd via fast alternating least squares. *The Journal of Machine Learning Research* 16(1):3367–3402.

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

Jain P, Meka R, Dhillon IS (2010) Guaranteed rank minimization via singular value projection. *Advances in Neural Information Processing Systems*, 937–945.

Ji H, Liu C, Shen Z, Xu Y (2010) Robust video denoising using low rank matrix completion. *Computer Society Conference on Computer Vision and Pattern Recognition* (IEEE).

Keshavan RH, Oh S, Montanari A (2009) Matrix completion from a few entries. *Information Theory, 2009. ISIT 2009. IEEE International Symposium on*, 324–328 (IEEE).

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

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

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

Natarajan N, Dhillon IS (2014) Inductive matrix completion for predicting gene–disease associations. *Bioinformatics* 30(12):i60–i68.

Nazarov I, Shirokikh B, Burkina M, Fedonin G, Panov M (2018) Sparse group inductive matrix completion. *arXiv preprint arXiv:1804.10653* .

Negahban S, Wainwright MJ (2012) Restricted strong convexity and weighted matrix completion: Optimal bounds with noise. *Journal of Machine Learning Research* 13(May):1665–1697.

R F, S L (1994) Solving mixed integer nonlinear programs by outer approximation. *Mathematical Programming* 66(3):327–349.

Recht B, Ré C (2013) Parallel stochastic gradient algorithms for large-scale matrix completion. *Mathematical Programming Computation* 5(2):201–226.

Shah V, Rao N, Ding W (2017) Matrix factorization with side and higher order information. *Stat* 1050:4.

Si S, Chiang KY, Hsieh CJ, Rao N, Dhillon IS (2016) Goal-directed inductive matrix completion. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1165–1174 (ACM).

Soni A, Chevalier T, Jain S (2016) Noisy inductive matrix completion under sparse factor models. *arXiv preprint arXiv:1609.03958* .

Tanner J, Wei K (2013) Normalized iterative hard thresholding for matrix completion. *SIAM Journal on Scientific Computing* 35(5):S104–S125.

Woodbury MA (1949) The stability of out-input matrices. *Chicago, IL* 93.

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