# Efficient Collaborative Filtering via Data Augmentation and Step-size Optimization

Xuejun Liao, Patrick Koch, Shunping Huang, Yan Xu
{xuejun.liao,patrick.koch,shunping.huang,yan.xu}@sas.com
SAS Institute Inc.
Cary, North Carolina, USA

## ABSTRACT

As a popular approach to collaborative filtering, matrix factorization (MF) models the underlying rating matrix as a product of two factor matrices, one for users and one for items. The MF model can be learned by Alternating Least Squares (ALS), which updates the two factor matrices alternately, keeping one fixed while updating the other. Although ALS improves the learning objective aggressively in each iteration, it suffers from high computational cost due to the necessity of inverting a separate matrix for every user and item. The softImpute-ALS reduces the per-iteration computation significantly using a strategy that requires only two matrix inversions; however, the computation saving leads to shrinkage of objective improvement. In this paper, we introduce a new algorithm, termed *Data Augmentation with Optimal Step-size (DAOS)*, which alleviates the drawback of softImpute-ALS while still maintaining its low cost of computation per iteration. The DAOS is presented in the context that factor matrices may include fixed columns or rows, with this allowing bias terms and/or linear models to be incorporated into the ML model. Experimental results on synthetic and MovieLens 1M Dataset demonstrate the benefits of DAOS over ALS and softImpute-ALS in terms of generalization performance and computational time.

## CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → **Factor analysis**.

## KEYWORDS

Collaborative filtering; Matrix factorization; Alternating least squares (ALS); SoftImpute-ALS; Pre-defined factors; Data augmentation (DA); Optimal step-size (OS)

**ACM Reference Format:**
Xuejun Liao, Patrick Koch, Shunping Huang, Yan Xu. 2021. Efficient Collaborative Filtering via Data Augmentation and Step-size Optimization. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3447548.3467380


*KDD '21, August 14–18, 2021, Virtual Event, Singapore*

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467380

## 1 INTRODUCTION

Recommender systems are a technology used ubiquitously in web services including Amazon, Netflix, and Pandora [8, 15]. From the perspective of users, a recommender provides personalized recommendation by helping users to find interesting items (products, movies, music, etc). From the perspective of items, a recommender performs targeted advertisement by identifying potential users that would be interested in a particular item. The information about users, items, and user-item interactions constitute the data that are used to achieve the goal of recommenders. Among the three types of information, user-item interactions are essential. Recommenders employing user-item interactions alone, without requiring the information of users or items, is based on a technique known as collaborative filtering.

For $m$ users and $n$ items, the interactions can be arranged into an $m$-by-$n$ matrix $R$ with $R_{ui}$ representing the interaction between user $u$ and item $i$. In this paper, we are concerned with explicit user feedback, in which case $R_{ui}$ is a numerical value representing the rating that $u$ gives to $i$. Typically, each user rates only a fraction of items and each item receive ratings from only a fraction of users, making $R$ an incomplete matrix with only a fraction of entries observed. In this matrix formulation, the goal of recommenders, specifically collaborative filtering, becomes predicting the missing entries of $R$ so as to locate the interesting items or potential users. The formulation has particularly motivated the work of solving collaborative filtering with matrix completion [4, 5, 12], where exciting theoretical results are established to guarantee exact matrix reconstruction under certain conditions. A major bottleneck of matrix completion is the reliance on singular value decomposition (SVD), limiting its use in large-scale applications.

An alternative approach to collaborative filtering that has gained popularity in practice is matrix factorization (MF), which models the user-item interactions as a product of two factor matrices, $R = XY$, where the rows of $X$ and the columns of $Y$ embed users and items, respectively, into an Euclidean space. With this embedding, each user or item is represented by a vector, and a rating entry of $R$ is represented by the inner product of two vectors. These vectors can be considered as a feature representation of the users and items. As they are not observed, but rather are inferred from user-item interactions, these vectors are commonly referred to as latent features or factors. Moreover, the latent features of all users and all items may be inferred simultaneously, making it possible to incorporate the benefit of multitask learning (MTL) [1, 6]. By the principle of MTL, the feature vector of each user is not only influenced by its own rating history, but also by the rating histories of other users, with the extent of influence dictated by the similarity between users. For this reason, a user may discover new interesting items from

the rating histories of its peers who share similar interests, with the similarity identified from all users' rating histories by learning algorithms.

A widely adopted algorithm for learning MF models is Alternating Least Squares (ALS) [2, 10, 11], which updates the two factor matrices alternately, keeping one fixed while updating the other. Given one matrix, ALS optimizes the other by solving a least squares (LS) problem for each user or item. As the LS solution is optimal, ALS can improve the learning objective aggressively in each iteration, leading to convergence in a small number of iterations. However, different users may have rated different items and, similarly, different items may have been rated by different users; thus, the LS problem for a user or item generally has a distinct Hessian matrix that differs from those of other users or items. As an LS solution requires inverting the Hessian matrix, this entails a separate matrix inversion for each user or item, leading to high computational cost in each iteration of ALS.

The issue is addressed in [9] with the so-called softImpute-ALS algorithm, which reduces the per-iteration computation of ALS using a strategy that requires only two matrix inversions. Instead of directly solving a LS problem for each user or item, softImpute-ALS first completes the rating matrix $R$ by imputing the missing ratings with the predictions provided by the current model (i.e., the model most recently updated). The completed $R$ matrix gives rise to a surrogate objective, which is then optimized by softImpute-ALS to yield the solution for the original objective. With the surrogate objective, the LS problems for all users or items now share the same Hessian matrix, which can be solved with a single matrix inversion. However, the optimal solution for the surrogate objective is only sub-optimal for the original objective. Therefore, improvement of the original objective in a single iteration of softImpute-ALS can be significantly smaller than that of ALS.

In this paper, we introduce a new algorithm, termed *Data Augmentation with Optimal Step-size (DAOS)*, to alleviate the drawback of softImpute-ALS. As the name indicates, DAOS first performs data augmentation, an equivalent to the imputation step of softImpute-ALS. However, DAOS goes one step further to construct a set of solutions, with the softImpute-ALS solution included in the set as special element. The solutions are parameterized by a scalar that plays the role of step-size in gradient descent [3]. The step-size is optimized by DAOS to find the solution that maximizes the original objective. The optimization guarantees a larger improvement of the original objective compared to the improvement achieved by softImpute-ALS, with this helping to alleviate the issue of slow progress per iteration and thus to speed up convergence. Thanks to the quadratic objective, the optimal step-size can be obtained in closed-form and its calculation does not introduce significant additional cost of computation; thus, DAOS has almost the same per-iteration computational complexity as softImpute-ALS in the big-O notation. With the low cost per iteration and more aggressive improvement of the learning objective, DAOS blends the advantage of softImpute-ALS into that of ALS, and is expected to achieve a high performance-to-cost ratio.

## 1.1 Contribution of This Work

The contribution of this work is summarized in detail below.

- A new algorithm called DAOS is introduced for learning matrix factorization models. The algorithm constructs a set of solutions parameterized by a step-size-like scalar, with the softImpute-ALS solution included in the set and associated with a unitary step-size. By optimizing the step-size parameter, DAOS obtains a solution that is guaranteed to yield a larger, or at least the same, improvement of the learning objective in any given iteration, compared to softImpute-ALS.
- The solution set constructed in DAOS is analyzed, showing some interesting property of the optimal step-size, and establishing closed-form formulae for the objective improvement achieved by DAOS as well as that by softImpute-ALS.
- When developing the DAOS algorithm, we consider a more general form of matrix factorization $R = XY$, in which some columns of $X$ and/or some rows of $Y$ are pre-defined and fixed. We refer to this general model as *matrix factorization with partially defined factors (MF-PDF)*. The MF-PDF allows bias terms and/or linear models to be incorporated into the ML model, which can be useful in practice. However, MF-PDF has the same basic structure as MF and can be learned by alternating algorithms (including ALS, softImpute-ALS, and DOAS) in the same way as MF is learned, except for some additional notations used to designate the adjustable factors versus the unadjustable factors, as will be clear shortly. The MF-PDF model has the practical advantage of updating bias terms along with the latent factors; it can also be used to incorporate engineered features (as opposed to inferred features) of users or items into collaborative filtering.
- Experimental results are provided to demonstrate the benefits of DAOS over ALS and softImpute-ALS, and to also identify the negative cases when DAOS is not performing as well as the competing algorithms.

## 1.2 Notational Convention

The following national convention is observed in the paper.

- Blackboard bold letters are sets of integers, e.g., $\mathbb{F}$, $\mathbb{P}$, $\mathbb{Q}$, $\mathbb{I}$.
- Capital Greek letters denote sets of integer pairs, e.g., $\Omega = \{(u, i), u \in \mathbb{P}, i \in \mathbb{Q}\}$.
- The compliment of a set is denoted as the set symbol with a bar over the head; for example, $\bar{\mathbb{P}}$ is a complement of $\mathbb{P}$.
- The cardinality of a set is denoted as $|\mathbb{F}|$.
- A universal set is shorthanded as : (i.e., a colon).
- A matrix is denoted by a capital letter with subscripts indicating rows and columns. For example,
  - $R$, $X$ and $Y$ are matrices.
  - $R_{ui}$ is the entry of $R$ with row index $u$ and column index $i$.
  - $R_{u:}$ is the $u$-th row of $R$ and $R_{:i}$ is the $i$-th column of $R$.
  - $X_{:\mathbb{F}_1}$ is a sub-matrix of $X$ obtained by extracting the columns indexed by the elements of $\mathbb{F}_1$.
  - $Y_{\mathbb{F}_2:}$ is a sub-matrix of $Y$ obtained by extracting the rows indexed by the elements of $\mathbb{F}_2$.
  - $X_{u\mathbb{P}}$ is the $u$-th row of $X_{:\mathbb{P}}$.
  - $Y_{\mathbb{P}i}$ is the $i$-th column of $Y_{\mathbb{P}:}$.

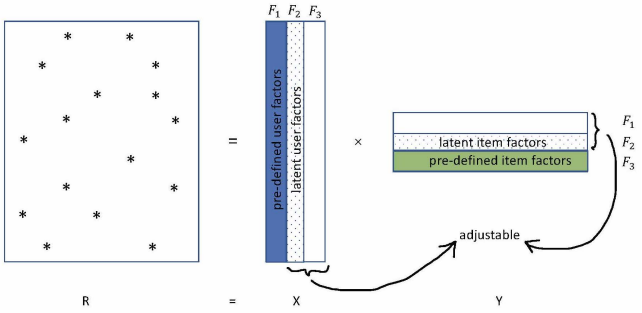


**Figure 1: The MF-PDF model, where $*$ denotes observed entries and the remaining entries of $R$ are missing. It is assumed, without loss of generality, that $k_1 < k_2 < k_3$ holds true for any $k_1 \in \mathbb{F}_1$, $k_2 \in \mathbb{F}_2$, $k_3 \in \mathbb{F}_3$. This can always be satisfied by shuffling the columns of $X$ and the rows of $Y$ appropriately.**

## 2 THE MF-PDF MODEL

Let $R$ be an $m$-by-$n$ matrix representing the user-item interactions (ratings in particular). The MF-PDF model for $R$ is illustrated in Figure 1. The model partitions each of the two factor matrices into three sub-matrices,

$$X = [X_{:\mathbb{F}_1}, X_{:\mathbb{F}_2}, X_{:\mathbb{F}_3}] \quad \text{and} \quad Y = \begin{bmatrix} Y_{\mathbb{F}_1:} \\ Y_{\mathbb{F}_2:} \\ Y_{\mathbb{F}_3:} \end{bmatrix},$$

where $\mathbb{F}_1$, $\mathbb{F}_2$ and $\mathbb{F}_3$ form a partition of $\{1, 2, \ldots, |\mathbb{F}_1| + |\mathbb{F}_2| + |\mathbb{F}_3|\}$, that is, they are mutually exclusive and $\mathbb{F}_1 \cup \mathbb{F}_2 \cup \mathbb{F}_1 = \{1, 2, \ldots, |\mathbb{F}_1| + |\mathbb{F}_2| + |\mathbb{F}_3|\}$. The partition yields a three-term representation of the user-item interactions $R$,

$$R = X_{:\mathbb{F}_1} Y_{\mathbb{F}_1:} + X_{:\mathbb{F}_2} Y_{\mathbb{F}_2:} + X_{:\mathbb{F}_3} Y_{\mathbb{F}_3:}, \tag{1}$$

with each term a sub-model as detailed below:

(1) The first term is a linear regression model, with $X_{:\mathbb{F}_1}$ the pre-defined factors (PDF) of users and $Y_{\mathbb{F}_1:}$ the associated regression parameters.
(2) The second term is a standard MF model, with $X_{:\mathbb{F}_2}$ the latent user factors and $Y_{\mathbb{F}_2:}$ the latent item factors.
(3) The third term is a linear regression model, with $Y_{\mathbb{F}_3:}$ the pre-defined factors of items and $X_{:\mathbb{F}_3}$ the associated regression parameters.

When $|\mathbb{F}_1| = |\mathbb{F}_3| = 0$, the MF-PDF model boils down to a standard MF model. In this paper, we are concerned with the case of $|\mathbb{F}_1| = |\mathbb{F}_3| = 1$, with $X_{:\mathbb{F}_1}$ being a column vector of all one's and $Y_{\mathbb{F}_3:}$ a row vector of all ones. In this spacial case, the regression parameters $X_{:\mathbb{F}_3}$ become the biases of users and $Y_{\mathbb{F}_1:}$ the biases of items. In future research, $X_{:\mathbb{F}_1}$ and $Y_{\mathbb{F}_3:}$ can be used to hold engineered features of users and items, respectively.

Although MF-PDF retains the basic structure of MF, it has an important difference from the standard MF model: the two factor matrices are only partially adjustable and the adjustable part of $X$ is not perfectly aligned with that of $Y$, as noted from Figure 1. More specifically, $X_{:\mathbb{P}}$ is the adjustable part of $X$ while $Y_{\mathbb{Q}:}$ is the adjustable part of $Y$, where $\mathbb{P} = \mathbb{F}_2 \cup \mathbb{F}_3$ and $\mathbb{Q} = \mathbb{F}_1 \cup \mathbb{F}_2$. It is clear that $\mathbb{P} \neq \mathbb{Q}$ unless $|\mathbb{F}_1| = |\mathbb{F}_3| = 0$. Therefore, $X_{:\mathbb{P}}$ and $Y_{\mathbb{Q}:}$ constitute the parameters of MF-PDF that are to be estimated.

### 2.1 The Learning Objective

As mentioned in the introduction, $R$ is typically a partially observed matrix with heavily missing entries in practice and the goal of collaborative filtering is to predict the missing (unknown) ratings based on the observed (known) ratings. With MF-PDF, we first learn model parameters $\{X_{:\mathbb{P}}, Y_{\mathbb{Q}:}\}$ using the observed entries of $R$, i.e., $\{R_{ui} : R_{ui} \text{ is known}\}$, and then employ the learned model to make predictions for the missing ratings.

The learning of MF-PDF is based on a commonly-adopted objective function, the squared error with $L_2$ regularization,

$$g(X_{:\mathbb{P}}, Y_{\mathbb{Q}:}) = \sum_{(u,i)\in\Omega} (R_{ui} - X_{u:} Y_{:i})^2 + \lambda \left(\|X_{:\mathbb{P}}\|^2 + \|Y_{\mathbb{Q}:}\|^2\right), \tag{2}$$

where it is defined that $\Omega \triangleq \{(u, i) : R_{ui} \text{ is known}\}$, $\|\cdot\|$ is the Frobenius norm, and $\lambda$ is a tuning parameter controlling the trade-off between the error term and the regularization terms.

Following [9, 11], we take the approach of updating $X_{:\mathbb{P}}$ and $Y_{\mathbb{Q}:}$ alternately, keeping one fixed while updating the other. The objective function of $X_{:\mathbb{P}}$ with fixed $Y_{\mathbb{Q}:}$ is

$$g(X_{:\mathbb{P}} | Y_{\mathbb{Q}:}) = \sum_{(u,i)\in\Omega} (R_{ui} - X_{u\bar{\mathbb{P}}} Y_{\bar{\mathbb{P}}i} - X_{u\mathbb{P}} Y_{\mathbb{P}i})^2 + \lambda \|X_{:\mathbb{P}}\|^2, \tag{3}$$

and the objective function of $Y_{\mathbb{Q}:}$ with fixed $X_{:\mathbb{P}}$ is

$$g(Y_{\mathbb{Q}:} | X_{:\mathbb{P}}) = \sum_{(u,i)\in\Omega} (R_{ui} - X_{u\bar{\mathbb{Q}}} Y_{\bar{\mathbb{Q}}i} - X_{u\mathbb{Q}} Y_{\mathbb{Q}i})^2 + \lambda \|Y_{:\mathbb{Q}}\|^2, \tag{4}$$

where $\bar{\mathbb{P}}$ is the complement of $\mathbb{P}$ in $\{1, 2, \cdots, |\mathbb{F}_1| + |\mathbb{F}_2| + |\mathbb{F}_3|\}$, as is $\bar{\mathbb{Q}}$ of $\mathbb{Q}$. As $X_{:\mathbb{P}}$ and $Y_{\mathbb{Q}:}$ play similar roles in (2), we present the details of updating $X_{:\mathbb{P}}$ given $Y_{\mathbb{Q}:}$, with the update of $Y_{\mathbb{Q}:}$ given $X_{:\mathbb{P}}$ following similarly.

### 2.2 Alternating Least Squares (ALS)

The optimal $X_{:\mathbb{P}}$ with a fixed $Y_{\mathbb{Q}:}$ is the solution to minimizing (3). The solution, which can be found by solving a least squares problem with respect to each row of $X_{:\mathbb{P}}$, is given by

$$X^{als}_{u\mathbb{P}} = (R_{u\mathbb{I}_u} - X_{u\bar{\mathbb{P}}} Y_{\bar{\mathbb{P}}\mathbb{I}_u}) Y^T_{\mathbb{P}\mathbb{I}_u} (\lambda I + Y_{\mathbb{P}\mathbb{I}_u} Y^T_{\mathbb{P}\mathbb{I}_u})^{-1}, \tag{5}$$

for $u = 1, 2, \cdots, m$, where $\mathbb{I}_u \triangleq \{i : R_{ui} \text{ is known}\}$ is the set of items for which user $u$ has provided ratings. This is exactly the ALS solution [11] rewritten for MF-PDF. The ALS solution requires inverting a $|\mathbb{P}|$-by-$|\mathbb{P}|$ matrix for each user, which can be costly when the number of users is large. The cost of ALS is on the same order of magnitude when updating $Y_{\mathbb{Q}:}$ for fixed $X_{:\mathbb{P}}$.

The matrix $\lambda I + Y_{\mathbb{P}\mathbb{I}_u} Y^T_{\mathbb{P}\mathbb{I}_u}$ is the Hessian for user $u$. A user-specific Hessian comprises only factors of the items rated by the user, as an item-specific Hessian comprises only factors of the users that have rated the item. In either case, the Hessian can be ill-conditioned when the user or item in question has few ratings associated with it. When the ill-conditioned Hessian is used to normalize the gradient, the descent direction will become problematic.

Although the LS solution is optimal for (3), it is not optimal for (2), because (3) is a local objective conditional on a fixed $Y_{\mathbb{Q}:}$. The local optimality makes $X^{als}_{u\mathbb{P}}$ a greedy solution. Compared to ALS,

the softImpute-ALS algorithm [9] has several advantages: (i) it is a non-greedy solution in each iteration; (ii) it can be calculated with less computation; (iii) it is more robust to data scarcity.

## 3 THE SOFTIMPUTE-ALS FOR MF-PDF

We re-derive the softImpute-ALS algorithm for MF-PDF. The re-derivation takes into account the pre-defined factors in $X$ and $Y$, leading to update equations of $X_{:\mathbb{P}}$ and $Y_{\mathbb{Q}:}$. In comparison, the original derivation of [9] updates $X$ and $Y$, without considering the pre-defined factors. Moreover, the re-derivation offers a different perspective and new insight into softImpute-ALS. In particular, we introduce the *surrogate gap*, defined as the difference between the surrogate objective and the original objective, and use it to motivate the DAOS algorithm. The surrogate gap is also expressed explicitly to facilitate the analysis in Section 4.2.2.

### 3.1 Data Augmentation (DA)

The re-derivation starts with the surrogate objective function,

$$g(X_{:\mathbb{P}}|Y_{\mathbb{Q}:},\tilde{X}_{:\mathbb{P}}) \triangleq g(X_{:\mathbb{P}}|Y_{\mathbb{Q}:}) + \sum_{(u,i)\in\bar{\Omega}} (\tilde{X}_{u\mathbb{P}}Y_{\mathbb{P}i} - X_{u\mathbb{P}}Y_{\mathbb{P}i})^2, \quad (6)$$

where $\bar{\Omega} \triangleq \{(u,i) : R_{ui} \text{ is missing}\}$ and $\tilde{X}_{:\mathbb{P}}$ is the current estimate of $X_{:\mathbb{P}}$. The surrogate objective has an additional term compared to the original objective in (3), which becomes zero at $X_{:\mathbb{P}} = \tilde{X}_{:\mathbb{P}}$; thus the surrogate objective coincides with the original objective at $X_{:\mathbb{P}} = \tilde{X}_{:\mathbb{P}}$. The new term in (6) appears to make the optimization harder to solve; however, we show shortly that it actually makes the optimization easier and more efficient.

To verify that optimization of (6) improves (3), we consider an arbitrary $\hat{X}_{:\mathbb{P}}$ satisfying $g(\hat{X}_{:\mathbb{P}}|Y_{\mathbb{Q}:},\tilde{X}_{:\mathbb{P}}) \le g(\tilde{X}_{:\mathbb{P}}|Y_{\mathbb{Q}:},\tilde{X}_{:\mathbb{P}})$. Substituting (6) into this inequality, we obtain

$$\begin{aligned} &g(\hat{X}_{:\mathbb{P}}|Y_{\mathbb{Q}:}) + \sum_{(u,i)\in\bar{\Omega}} (\tilde{X}_{u\mathbb{P}}Y_{\mathbb{P}i} - \hat{X}_{u\mathbb{P}}Y_{\mathbb{P}i})^2 \\ \le\ & g(\tilde{X}_{:\mathbb{P}}|Y_{\mathbb{Q}:}) + \sum_{(u,i)\in\bar{\Omega}} (\tilde{X}_{u\mathbb{P}}Y_{\mathbb{P}i} - \tilde{X}_{u\mathbb{P}}Y_{\mathbb{P}i})^2. \end{aligned}$$

Noticing the sum on the right side vanishes, we have

$$\begin{aligned} g(\hat{X}_{:\mathbb{P}}|Y_{\mathbb{Q}:}) &\le g(\tilde{X}_{:\mathbb{P}}|Y_{\mathbb{Q}:}) - \sum_{(u,i)\in\bar{\Omega}} (\tilde{X}_{u\mathbb{P}}Y_{\mathbb{P}i} - \hat{X}_{u\mathbb{P}}Y_{\mathbb{P}i})^2, \quad (7)\\ &\le g(\tilde{X}_{:\mathbb{P}}|Y_{\mathbb{Q}:}). \end{aligned}$$

Therefore, if $\hat{X}_{:\mathbb{P}}$ is a better estimate than $\tilde{X}_{:\mathbb{P}}$ for (6), it is also a better estimate for (3).

A close examination shows that the second term in (6) is basically the squared error summed over the entries of $R$ which are originally missing but are filled with

$$R_{ui} = X_{u\bar{\mathbb{P}}}Y_{\bar{\mathbb{P}}i} + \tilde{X}_{u\mathbb{P}}Y_{\mathbb{P}i}, \quad \forall\ (u,i) \in \bar{\Omega}. \quad (8)$$

This step of filling or imputing the missing values of $R$ is the application of a statistical technique known as *data-augmentation (DA)* [14]. With the data-augmentation, the fill-in entries completes the matrix $R$ and we can write the surrogate objective concisely as

$$g(X_{:\mathbb{P}}|Y_{\mathbb{Q}:},\tilde{X}_{:\mathbb{P}}) = \left\|R - X_{:\bar{\mathbb{P}}}Y_{\bar{\mathbb{P}}:} - X_{:\mathbb{P}}Y_{\mathbb{P}:}\right\|^2 + \lambda\|X_{:\mathbb{P}}\|^2. \quad (9)$$

The minimizer of (9) leads to the softImpute-ALS solution,

$$\begin{aligned} X_{:\mathbb{P}}^{\text{si-als}} &= \min_{X_{:\mathbb{P}}} g(X_{:\mathbb{P}}|Y_{\mathbb{Q}:},\tilde{X}_{:\mathbb{P}})\\ &= (R - X_{:\bar{\mathbb{P}}}Y_{\bar{\mathbb{P}}:})Y_{\mathbb{P}:}^T\left(\lambda I + Y_{\mathbb{P}:}Y_{\mathbb{P}:}^T\right)^{-1}, \quad (10) \end{aligned}$$

where "si-als" is used as a shorthand for softImpute-ALS. Although $X_{:\mathbb{P}}^{\text{si-als}}$ is minimizer of the surrogate objective (9), it is also a better estimate than $\tilde{X}_{:\mathbb{P}}$ for the original objective in (3), according to the arguments made above. Compared to (5), the solution in (10) can be calculated by inverting one $|\mathbb{P}|$-by-$|\mathbb{P}|$ matrix only, instead of inverting $m$ matrices. This is the major advantage of softImpute-ALS over ALS that is claimed in [9].

### 3.2 Sparse Representation of the DA Solution

Since $R - X_{:\bar{\mathbb{P}}}Y_{\bar{\mathbb{P}}:}$ is a huge dense matrix, direct evaluation of the solution (10) is not appealing. Fortunately, there exists a strategy of representing $\hat{X}$ using only the observed entries $\{R_{ui} : (u,i) \in \Omega\}$, avoiding the fill-in entries $\{R_{ui} : (u,i) \in \bar{\Omega}\}$. To see this, we expand (10) to get

$$\begin{aligned} X_{:\mathbb{P}}^{\text{si-als}} &= (R - X_{:\bar{\mathbb{P}}}Y_{\bar{\mathbb{P}}:} - \tilde{X}_{:\mathbb{P}}Y_{\mathbb{P}:} + \tilde{X}_{:\mathbb{P}}Y_{\mathbb{P}:})Y_{\mathbb{P}:}^T\left(\lambda I + Y_{\mathbb{P}:}Y_{\mathbb{P}:}^T\right)^{-1},\\ &= \tilde{X}_{:\mathbb{P}} - \lambda\tilde{X}_{:\mathbb{P}}\left(\lambda I + Y_{\mathbb{P}:}Y_{\mathbb{P}:}^T\right)^{-1}\\ &\quad + (R - X_{:\bar{\mathbb{P}}}Y_{\bar{\mathbb{P}}:} - \tilde{X}_{:\mathbb{P}}Y_{\mathbb{P}:})Y_{\mathbb{P}:}^T\left(\lambda I + Y_{\mathbb{P}:}Y_{\mathbb{P}:}^T\right)^{-1}, \quad (11) \end{aligned}$$

It is important to note that $(R - X_{:\bar{\mathbb{P}}}Y_{\bar{\mathbb{P}}:} - \tilde{X}_{:\mathbb{P}}Y_{\mathbb{P}:})$ is a sparse matrix because its $(u,i)$-th entry is zero for any $(u,i) \in \bar{\Omega}$ due to the data augmentation in (8). Removing the terms involving the zero entries of this matrix, we can write (11) in an equivalent row-wise manner,

$$\begin{aligned} X_{u\mathbb{P}}^{\text{si-als}} &= \tilde{X}_{u\mathbb{P}} - \lambda\tilde{X}_{u\mathbb{P}}\left(\lambda I + Y_{\mathbb{P}:}Y_{\mathbb{P}:}^T\right)^{-1}\\ &\quad + (R_{u\mathbb{I}_u} - X_{u\bar{\mathbb{P}}}Y_{\bar{\mathbb{P}}\mathbb{I}_u} - \tilde{X}_{u\mathbb{P}}Y_{\mathbb{P}\mathbb{I}_u})Y_{\mathbb{P}\mathbb{I}_u}^T\left(\lambda I + Y_{\mathbb{P}:}Y_{\mathbb{P}:}^T\right)^{-1}, \quad (12) \end{aligned}$$

for $u = 1, 2, \ldots, m$, where $\mathcal{I}_u = \{i : R_{ui} \text{ is known}\}$ as defined earlier. It is clear from (12) that calculation of $\hat{X}_{:\mathbb{P}}$ does not involve the missing entries of $R$ and can thus be computed efficiently.

### 3.3 New Insights into SoftImpute-ALS

*3.3.1 Data augmentation as a new regularizer.* It is worthwhile to point out an additional advantage of softImpute-ALS over ALS, which is not discussed in [9]. The Hessian matrix $\lambda I + Y_{\mathbb{P}:}Y_{\mathbb{P}:}^T$ in (10) is shared by all users and comprises the factors from all items, as opposed to the Hessian matrix $\lambda I + Y_{\mathbb{P}\mathbb{I}_u}Y_{\mathbb{P}\mathbb{I}_u}^T$ in (5), which is user-dependent and includes only the items rated by each particular user $u$. For a user that has rated very few items, the matrix used by ALS can become ill-conditioned unless the $L_2$ regularization has a sufficient strength. For softImpute-ALS, however, all users use the same matrix and that matrix is unlikely to be ill-conditioned unless the union of items rated by all users is small and the regularization is not strong.

The resistance of softImpute-ALS to data scarcity of individual users builds upon the sharing of information across different users, which is essentially the benefit of data augmentation combined with the benefit of collaborative filtering. This can be clearly seen from the fact that, even though user $u$ has never rated item $i$, it still gets a sample $R_{ui} = X_{u\bar{\mathbb{P}}}Y_{\bar{\mathbb{P}}i} + \tilde{X}_{u\mathbb{P}}Y_{\mathbb{P}i}$ from data augmentation

using $Y_{\mathbb{P}i}$ which is learned with the ratings of other users for the item. It is noted that one can always increase the regularization strength to prevent an ill-conditioned Hessian. As true as it is, data augmentation helps to prevent ill-conditioning regardless of the regularization. In this sense, data augmentation plays a similar role as regularization. This observation is in good agreement with the role of preventing overfitting played by broader types of data augmentation [16]. Despite of the different details, the message conveyed is essentially the same: an appropriate enhancement of training samples can prevent models from overfitting.

*3.3.2 Surrogate gap.* The above derivation of softImpute-ALS can be interpreted as an instance of handling incomplete data using *expectation maximization (EM)* [7], under the assumption that the entries of $R$ are independently and identically distributed (iid) as a standard normal distribution with unitary variance and mean $X_{u:}Y_{:i}$, $\forall\ (u,i)$. The basic idea of EM is to treat the missing values as hidden variables and consider their distribution conditional on the current model and the observed variables (the observed entries of $R$). The first step of EM is to take expectation of the hidden variables with respect to this conditional distribution, which results in the surrogate objective function in (6). Due to the simplicity of the present problem, however, we are able to obtain the surrogate objective in a more straightforward way.

The principle of EM guarantees that the surrogate objective is an upper-bound to the original objective in (3). The difference between the two objectives or the surrogate gap, as defined at the beginning of Section 3, is indicated by the second term in (6). The non-negativity of the term proves the surrogate is an upper bound.

## 4 THE DAOS ALGORITHM

A crucial fact about softImpute-ALS is that the solution in (10) optimizes the surrogate objective (6), but it does not optimize the original objective (3). This fact suggests the possibility that the solution may be further improved to achieve a larger decrease of the original objective. This is the major motivation of the DAOS algorithm. As noted in (7), softImpute-ALS improves (3) through two ways: i) minimization of (10), and ii) the nonnegative surrogate gap. While the surrogate gap is dictated by the surrogate objective and cannot be changed, we show that it can be enhanced in some sense to further improve (3).

### 4.1 Solution Set and Optimization of Step-size

The basic idea of DAOS is to construct a set of solutions with the softImpute-ALS solution included as a special solution. We start with the definition

$$\begin{aligned} D &= (R - X_{:\bar{\mathbb{P}}}Y_{\bar{\mathbb{P}}:} - \tilde{X}_{:\mathbb{P}}Y_{\mathbb{P}:})Y_{\mathbb{P}:}^T\left(\lambda I + Y_{\mathbb{P}:}Y_{\mathbb{P}:}^T\right)^{-1} \\ &\quad -\lambda\tilde{X}_{:\mathbb{P}}\left(\lambda I + Y_{\mathbb{P}:}Y_{\mathbb{P}:}^T\right)^{-1}, \end{aligned} \tag{13}$$

and then the set of solutions is constructed as

$$\hat{X}_{:\mathbb{P}} = \tilde{X}_{:\mathbb{P}} + \eta D, \quad \eta \in \mathbb{R} \tag{14}$$

where $\eta$ is a real-number parameter of the solution set. The solution set has the same form as gradient descent [3], except that the direction $D$ has a particular definition, rather than the gradient, Newton direction or other popular choices. The parameter $\eta$ may be seen as a step-size controlling in which direction and how far to move along direction $D$, starting from the current estimate $\tilde{X}_{:\mathbb{P}}$. It can be verified that the softImpute-ALS solution (11) is included in the set with $\eta = 1$ associated with it.

As indicted in (10), $X_{:\mathbb{P}}^{\text{si-als}}$ is the minimizer of the surrogate objective $g(X_{:\mathbb{P}}|Y_{\mathbb{Q}:}, \tilde{X}_{:\mathbb{P}})$ and $\eta = 1$ is therefore the optimal step-size as far as the surrogate objective is concerned. However, we show next that $\eta = 1$ is actually not the optimal step-size for the original objective in (3). Further improvement can be achieved for the original objective by optimizing $\eta$. Substituting $X_{:\mathbb{P}} = \tilde{X}_{:\mathbb{P}} + \eta D$ into (3), one obtains

$$\begin{aligned} g_{\text{daos}}(\eta) &= \sum_{(u,i)\in\Omega}(R_{ui} - X_{u\bar{\mathbb{P}}}Y_{\bar{\mathbb{P}}i} - \tilde{X}_{u\mathbb{P}}Y_{\mathbb{P}i} - \eta D_{u:}Y_{\mathbb{P}i})^2 \\ &\quad +\lambda\|\tilde{X}_{:\mathbb{P}} + \eta D\|^2, \end{aligned} \tag{15}$$

which is a function of $\eta$ for fixed $Y_{\mathbb{P}:}$, $\tilde{X}_{:\mathbb{P}}$, and $D$. This is a univariate quadratic function and admits a closed-form solution. As the solution is also required by the subsequent analysis for a similar function, we give the solution for a general form of (15) in Lemma 1. Minimization of (15) leads to the optimal step-size which, by Lemma 1, is given by $\eta_{\text{daos}} = \alpha_{\text{daos}}/\beta_{\text{daos}}$, where

$$\begin{aligned} \alpha_{\text{daos}} &= \sum_{(u,i)\in\Omega}(R_{ui} - X_{u\bar{\mathbb{P}}}Y_{\bar{\mathbb{P}}i} - \tilde{X}_{u\mathbb{P}}Y_{\mathbb{P}i})(D_{u:}Y_{\mathbb{P}i}) - \lambda\,\text{tr}(\tilde{X}_{:\mathbb{P}}^T D) \\ \beta_{\text{daos}} &= \sum_{(u,i)\in\Omega}(D_{u:}Y_{\mathbb{P}i})^2 + \lambda\|D\|^2. \end{aligned} \tag{16}$$

We refer to $X_{:\mathbb{P}}^{\text{daos}} \triangleq \tilde{X}_{:\mathbb{P}} + \eta_{\text{daos}}D$ as the DAOS solution. A complete description of the DAOS algorithm and its time complexity is given in the Appendix.

Lemma 1. *Let $A, B, C, D$ be fixed matrices of appropriate sizes and*

$$g(\eta) = \textstyle\sum_{u,i}(C_{ui} - A_{u:}B_{:i} - \eta D_{u:}B_{:i})^2 + \lambda\|A + \eta D\|^2. \tag{17}$$

*Then* $\arg\min_\eta g(\eta) = \frac{\alpha}{\beta}$ *and* $\min_\eta g(\eta) = g(0) - \frac{\alpha^2}{\beta}$, *with*

$$\begin{aligned} \alpha &= \textstyle\sum_{u,i}(C_{ui} - A_{u:}B_{:i})(D_{u:}B_{:i}) - \lambda\,\text{tr}(A^T D). \\ \beta &= \textstyle\sum_{u,i}(D_{u:}B_{:i})^2 + \lambda\|D\|^2, \end{aligned}$$

*where* tr() *stands for the trace of a matrix. In the special case that the summation $\sum_{u,i}$ traverses every entry of $C$ in (17), one has $\alpha = \text{tr}((C - AB)^T DB) - \lambda\,\text{tr}(A^T D)$ and $\beta = \|DB\|^2 + \lambda\|D\|^2$.*

### 4.2 Analysis of DAOS versus SoftImpute-ALS

By virtue of the construction in (14), $X_{:\mathbb{P}}^{\text{daos}}$ and $X_{:\mathbb{P}}^{\text{si-als}}$ can both be represented as $\tilde{X}_{:\mathbb{P}} + \eta D$ with $\eta$ taking different step-size values, which we respectively denote as $\eta_{\text{daos}}$ and $\eta_{\text{si-als}}$. Moreover, these step-size values can each be represented as $\eta_{\text{daos}} = \alpha_{\text{daos}}/\beta_{\text{daos}}$ and $\eta_{\text{si-als}} = \alpha_{\text{si-als}}/\beta_{\text{si-als}}$, according to Lemma 1. An analysis of these $\alpha$'s and $\beta$'s reveals interesting relations between DAOS and softImpute-ALS, which consequently provide insights into why DAOS provides larger improvement to the original objective than softImpute-ALS and how much the additional improvement can be.

With $\alpha_{\text{daos}}$ and $\beta_{\text{daos}}$ given by (16), the analysis starts with finding the counterparts for softImpute-ALS. We can get an expression of $\alpha_{\text{si-als}}$ and $\beta_{\text{si-als}}$ by substituting $\tilde{X}_{:\mathbb{P}} + \eta D$ into (9) and applying Lemma 1 to the resulting objective function,

$$g_{\text{si-als}}(\eta) = \|R - X_{:\bar{\mathbb{P}}}Y_{\bar{\mathbb{P}}:} - \tilde{X}_{:\mathbb{P}}Y_{\mathbb{P}:} - \eta D Y_{\mathbb{P}:}\|^2 + \lambda\,\|\tilde{X}_{:\mathbb{P}} + \eta D\|^2. \tag{18}$$

This leads to the expression

$$\begin{aligned}\alpha_{\text{si-als}} &= \operatorname{tr}((R - X_{:\bar{\mathbb{P}}} Y_{\bar{\mathbb{P}}:} - \tilde{X}_{:\mathbb{P}} Y_{\mathbb{P}:})^T D Y_{\mathbb{P}:}) - \lambda \operatorname{tr}(\tilde{X}_{:\mathbb{P}}^T D),\\ \beta_{\text{si-als}} &= \|DY_{\mathbb{P}:}\|^2 + \lambda\|D\|^2,\end{aligned} \tag{19}$$

with the step-size accordingly given by $\eta_{\text{si-als}} = \alpha_{\text{si-als}}/\beta_{\text{si-als}}$. As $\eta_{\text{si-als}} = 1$ by (11), one must have $\alpha_{\text{si-als}} = \beta_{\text{si-als}}$. Meanwhile, a comparison (19) with (16) reveals

$$\begin{aligned}\alpha_{\text{si-als}} &= \alpha_{\text{daos}} + \textstyle\sum_{(u,i)\in\bar{\Omega}} (R_{ui} - X_{u\bar{\mathbb{P}}} Y_{\bar{\mathbb{P}}i} - \tilde{X}_{u\mathbb{P}} Y_{\mathbb{P}i})(D_{u:} Y_{\mathbb{P}i}),\\ &= \alpha_{\text{daos}}\\ \beta_{\text{si-als}} &= \beta_{\text{daos}} + \textstyle\sum_{(u,i)\in\bar{\Omega}} (D_{u:} Y_{\mathbb{P}i})^2.\end{aligned} \tag{20}$$

where the second equality arises because $R_{ui} - X_{:\bar{\mathbb{P}}} Y_{\bar{\mathbb{P}}:} - \tilde{X}_{:\mathbb{P}} Y_{\mathbb{P}:} = 0$ for any $(u,i) \in \bar{\Omega}$ according to (8). With these preparatory equations, we are now ready to do some interesting analysis of DAOS and softImpute-ALS, making a theoretical comparison of them from two perspectives.

*4.2.1 The Step-size Parameter.* We first analyze the step-size parameter of DAOS. Starting from the original form of $\eta_{\text{doas}}$, we can make a sequence of derivations to arrive at an interesting form,

$$\begin{aligned}\eta_{\text{daos}} &= \frac{\alpha_{\text{daos}}}{\beta_{\text{daos}}} = \frac{\alpha_{\text{si-als}}}{\beta_{\text{daos}}} = \frac{\beta_{\text{si-als}}}{\beta_{\text{daos}}} = \frac{\beta_{\text{daos}} + \sum_{(u,i)\in\bar{\Omega}} (D_{u:} Y_{\mathbb{P}i})^2}{\beta_{\text{daos}}},\\ &= 1 + \frac{\sum_{(u,i)\in\bar{\Omega}} (D_{u:} Y_{\mathbb{P}i})^2}{\beta_{\text{daos}}} = \eta_{\text{si-als}} + \frac{\sum_{(u,i)\in\bar{\Omega}} (D_{u:} Y_{\mathbb{P}i})^2}{\beta_{\text{daos}}}.\end{aligned}$$

Substituting the expression of $\beta_{\text{daos}}$ (16) into the rightmost side, we obtain an interesting representation,

$$\eta_{\text{daos}} = \eta_{\text{si-als}} + \frac{\sum_{(u,i)\in\bar{\Omega}} (D_{u:} Y_{\mathbb{P}i})^2}{\sum_{(u,i)\in\Omega} (D_{u:} Y_{\mathbb{P}i})^2 + \lambda\|D\|^2}. \tag{21}$$

This equation establishes the relation between DAOS and softImpute-ALS through the step-size parameter $\eta$. The second term, which is nonnegative, indicates that DAOS is at least as aggressive as, and can be much more aggressive than, softImpute-ALS in moving down direction $D$. Geometrically, this implies that the bottom of the original objective could lie further down $D$ beyond the bottom of the surrogate objective, noting that both objective functions are quadratic and thus bowl-shaped.

*4.2.2 Improvements of the Original Objective.* We now analyze the improvement that softImpute-ALS and DAOS each makes in a single iteration for the original objective. By (18), the improvement made by softImpute-ALS for the surrogate objective is

$$\begin{aligned} g_{\text{si-als}}(\eta_{\text{si-als}}) - g_{\text{si-als}}(0) &= -\alpha_{\text{si-als}}^2/\beta_{\text{si-als}} = -\beta_{\text{si-als}}\\ &= -\|DY_{\mathbb{P}:}\|^2 - \lambda\|D\|^2,\end{aligned}$$

where the first equality is due to Lemma 1 and the second equality due to $\alpha_{\text{si-als}} = \beta_{\text{si-als}}$. Then, according to (6), the improvement for the original objective is then

$$-\|DY_{\mathbb{P}:}\|^2 - \lambda\|D\|^2 - \textstyle\sum_{(u,i)\in\bar{\Omega}} (D_{u:} Y_{\mathbb{P}i})^2, \tag{22}$$

where the last term is the surrogate gap.

To find the improvement made by DAOS for the original objective, we apply Lemma 1 to (15) and then perform a sequence of derivations using $\alpha_{\text{daos}} = \beta_{\text{si-als}}$ and other equations that have been established earlier in this section,

$$\begin{aligned} g_{\text{daos}}(\eta_{\text{daos}}) - g_{\text{daos}}(0) &= -\frac{\alpha_{\text{daos}}^2}{\beta_{\text{daos}}} = -\eta_{\text{daos}}\alpha_{\text{daos}}\\ &= -\eta_{\text{daos}}\beta_{\text{si-als}} = -\beta_{\text{si-als}} - \frac{\beta_{\text{si-als}}}{\beta_{\text{daos}}} \textstyle\sum_{(u,i)\in\bar{\Omega}} (D_{u:} Y_{\mathbb{P}i})^2,\\ &= -\beta_{\text{si-als}} - \eta_{\text{daos}} \textstyle\sum_{(u,i)\in\bar{\Omega}} (D_{u:} Y_{\mathbb{P}i})^2,\\ &= -\|DY_{\mathbb{P}:}\|^2 - \lambda\|D\|^2 - \eta_{\text{daos}} \textstyle\sum_{(u,i)\in\bar{\Omega}} (D_{u:} Y_{\mathbb{P}i})^2.\end{aligned} \tag{23}$$

A comparison of (22) and (23) leads to more insight into DAOS. While the surrogate gap of softImpute-ALS is $\sum_{(u,i)\in\bar{\Omega}} (D_{u:} Y_{\mathbb{P}i})^2$, the counterpart of DAOS is $\eta_{\text{daos}} \sum_{(u,i)\in\bar{\Omega}} (D_{u:} Y_{\mathbb{P}i})^2$. As $\eta_{\text{daos}} \geq 1$, one may interpret $\eta_{\text{daos}} \sum_{(u,i)\in\bar{\Omega}} (D_{u:} Y_{\mathbb{P}i})^2$ as an enhancement or amplification of the surrogate gap. In this sense, one may think of DAOS as using the enhanced or amplified surrogate gap to achieve additional improvement based upon the improvement already achieved by softImpute-ALS. This interpretation agrees with the geometric intuition mentioned at the end of Section 4.2.1.

## 5 EXPERIMENTAL RESULTS

We perform a comparative study of DAOS versus softImpute-ALS and ALS, through use of synthetic data and MovieLens 1M Dataset. All three algorithms are implemented in C and executed on the same machine in each experiment.

### 5.1 Synthetic Data

A $1000 \times 2000$ rating matrix is synthesized as $R = X_{:\mathbb{F}_2} Y_{\mathbb{F}_2:} + 0.01N$, where the entries of $X_{:\mathbb{F}_2}$, $Y_{\mathbb{F}_2:}$ and $N$ are all independently drawn from a standard normal distribution. Thus the synthetic data follow the MF-PDF model in (1), with $m = 1000$, $n = 2000$, $|\mathbb{F}_1| = |\mathbb{F}_3| = 0$ and additive white Gaussian noise. The sampler $\Omega$ is constituted by the first $|\Omega|$ elements of a random permutation of $\Omega_0 \triangleq \{(u,i) : u = 1 \ldots m, i = 1 \ldots n\}$, with $\{R_{u,i} : (u,i) \in \Omega\}$ used as training data and $\{R_{u,i} : (u,i) \in \bar{\Omega}\}$ as test data, where $\bar{\Omega}$ is the compliment of $\Omega$ in $\Omega_0$. We learn the factor matrices $X_{:\mathbb{F}_2}$ and $Y_{\mathbb{F}_2:}$ by minimizing the objective in (2) with $\lambda$ set to 0.01; since $|\mathbb{F}_1| = |\mathbb{F}_3| = 0$, we have $\mathbb{P} = \mathbb{Q} = \mathbb{F}_2$ with $\bar{\mathbb{P}}$ and $\bar{\mathbb{Q}}$ being null sets. Moreover, we assume the truth of $|\mathbb{F}_2|$ is known and do not tune it.

By construction, $R$ is a rank-$|\mathbb{F}_2|$ matrix plus Gaussian noise and hence its rank is approximately $|\mathbb{F}_2|$. Since the size of $R$ and the noise level are fixed, the complexity of $R$ is mainly determined by $|\mathbb{F}_2|$. The more complex $R$ is, the more training samples it requires to obtain a good estimate of $X_{:\mathbb{F}_2}$ and $Y_{\mathbb{F}_2:}$. Motivated by this, we vary $|\mathbb{F}_2|$ and $|\Omega|$ to examine the performances of the participating algorithms, in hope of finding their merits and drawbacks in the different scenarios and providing insight into the algorithms from the experimental perspective.

For a given setting of $(|\mathbb{F}_2|, |\Omega|)$, we run each learning algorithm on the training data. At each iteration of an algorithm, we compute the cumulative squared error on test data, $\sum_{(u,i)\in\bar{\Omega}} (R_{ui} - X_{u\mathbb{F}_2} Y_{\mathbb{F}_2 i})^2$, using it as a performance metric to evaluate how well the model obtained so far generalizes to data unseen in training. We are interested in examining how fast each algorithm converges in learning and how well the model it produces performs in generalization, as learning proceeds through the iterations.

The results are summarized in Figure 2 in a matrix-like format, with the rows grouped into three duos each associated with a specific value of $|\mathbb{F}_2|$, and with each column associated with a specific value of $|\Omega|$. The values of $|\mathbb{F}_2|$ increase from top to bottom and those of $|\Omega|$ decrease from left to right. This arrangement makes

an interesting pattern: as one moves from the top-left towards the bottom-right, one has less and less training data to learn a more and more complex rating matrix and thus the problem becomes more difficult.

In each duo of rows, the top row plots the learning objective and the bottom row plots the generalization performance, both as a function of the cumulative time over the iterations of each learning algorithm. The three algorithms in comparison are distinguished by color as well as line type as explained in more detail in the caption.

A first inspection of Figure 2 leads to observation of several general pattens: (i) ALS makes the most aggressive progress between iterations, followed by DAOS and then softImpute-ALS. This agrees perfectly well with our theories in Sections 3 and 4; (ii) the difference between the algorithms decreases with $|\Omega|$, demonstrating that a highly observed $R$ is not an interesting case; (iii) in terms of learning speed, the area towards the top-left is won by softImpute-ALS while the area towards the bottom-right is won ALS, which, in light of the arrangement of the results, indicates softImpute-ALS wins the easy problems and ALS wins the difficult problems as far as learning speed is concerned; however, a fast learner does not necessarily generalize well, as discussed shortly.

To obtain a more in-depth understanding, we classify the results into three cases according to the sufficiency of training data and discuss the algorithms' convergence and performance in each case.

*5.1.1 Case I - The training data are overly sufficient.* This case is represented by the first column of figures, in which 80% of the entries in $R$ are used as training data. With this high percentage of observability, $R$ is close to a complete matrix; therefore the surrogate gap, i.e., the second term in (6), is small, and consequently the surrogate objective is close to the original objective. Recall that softImpute-ALS solution minimizes the surrogate objective; then it also approximately minimizes the original objective, given the small difference between the two. This makes softImpute-ALS the fastest-convergent algorithm in this case.

In this case, moreover, the more an algorithm converges in learning, the better it performs in generalization, demonstrating that over-fitting is unlikely to happen when training data are sufficient. However, overly-sufficient training data are wasteful. due to the effort of collecting the unnecessary extra data. As a result, this case is not interesting in practice.

*5.1.2 Case II - The training data are overly insufficient.* This case is represented by the last column of figures, in which 1.25% of the entries in $R$ are used as training data for $|\mathbb{F}_2| = 18$, 2.5% for $|\mathbb{F}_2| = 38$ and 5% for $|\mathbb{F}_2| = 78$. The different training percentages reflect the fact that a larger $|\mathbb{F}_2|$ indicates a more complex $R$ matrix which requires more training data to learn a meaningful model.

In this case, $R$ has heavily missing entries and hence deviates significantly from a complete matrix. This leaves each user and/or item few ratings to train on. More specifically, the number of training ratings per user and/or item could be smaller than $|\mathbb{F}_2|$, which makes (3) and/or (4) close to under-determined problems, considering $\lambda = 0.01$ may not provide enough regularization. As noted in Figure 2, ALS has the learning objective decreased to below $10^{-3}$ in just one or a few iterations, confirming this case is happening in these experiments.

An under-determined problem requires additional information to compensate for the insufficient training data [13]. Such addition information could be encoded by regularizers, constraints, or priors in a Bayesian setting. Unless the additional information is incorporated, the solution will overfit to training data and cannot generalize well to new data. What is worse in this case is that the more an algorithm converges in learning, the poorer it performs in generalization. Although ALS converges faster than softImpute-ASL and DAOS in this case, it performs worse in generalization, the reason being that softImpute-ASL and DAOS do not follow the training data as closely as ALS during learning.

*5.1.3 Case III - The amount of training data is reasonable.* This case is represented by all the other figures that are not included in Case I or Case II. It is therefore a typical case covering a wide range of scenarios, with the training percentage ranging from 5% to 40% for $|\mathbb{F}_2| = 18$ and $|\mathbb{F}_2| = 38$, from 7.5% to 40% for $|\mathbb{F}_2| = 78$.

In this typical case, DAOS converges the fastest in learning and spends the least time to reach a given generalization performance when $|\Omega|$ is large enough to avoid over-fitting. When $|\Omega|$ is too small and over-fitting is inevitable, DAOS behaves more closely to softImpute than to ALS in generalization.

A careful examination of Figure 2 further shows that, in all three cases, the advantage of DAOS over softImpute-ALS becomes more conspicuous as $|\Omega|$ decreases. This can be explained by a comparison of (22) and (23), which indicate that DAOS achieves an additional improvement in each iteration for the original objective by the amount of $(\eta_{\text{daos}} - 1)\sum_{(u,i)\in\hat{\Omega}}(D_{u:}Y_{\mathbb{P}i})^2$, based upon the improvement already acheived by softImpute-ALS. Using (21) and noting $\eta_{\text{si-als}} = 1$, we have $(\eta_{\text{daos}} - 1) = \frac{\sum_{(u,i)\in\hat{\Omega}}(D_{u:}Y_{\mathbb{P}i})^2}{\sum_{(u,i)\in\Omega}(D_{u:}Y_{\mathbb{P}i})^2+\lambda\|D\|^2}$. As $|\Omega|$ decreases, $|\bar{\Omega}| = mn - |\Omega|$ increases, which could lead to significant increase for the additional improvement. Thus, the theoretical results from the analysis in Section 4.2 is verified by the experimental results in Figure 2.

## 5.2 MovieLens Data

We consider the MovieLens 1M Dataset, which is a public-domain dataset available at https://grouplens.org/datasets/movielens/. The dataset contains 1,000,209 ratings of 3,706 movies from 6,040 users. Therefore, in the notation of this paper, $R$ is a $m$-by-$n$ matrix with $m = 6040$ and $n = 3706$. Unlike the case of synthetic data, we do not have access to the full matrix. So we randomly split the available ratings in half, with one half used as training data and the other half as test data. Accordingly we have $|\Omega| = 1000209/2 = 500104$ and $|\Omega|/(mn) \approx 2.234$. We do not attempt to change the training percentage for the consideration that the available ratings constitute only about 4.468% of the full entries of $R$. The low percentage makes the problem fall into Case II defined in Section 5.1 and thus adjusting $|\Omega|$ makes little change in this regard.

The same consideration also motivates us to adjust $\lambda$ to see the effects of the $L_2$ regularizer on helping to improve generalization. Since this is a real dataset, we do not know the true rank of $R$. Therefore, we also want to adjust $|\mathbb{F}_2|$ to examine how it affects the results. Moreover, we use the full MF-PDF model in (1) with user and item biases, i.e., $|\mathbb{F}_1| = |\mathbb{F}_3| = 1$. In summary, our goal in this experiment is to examine the learning convergences and

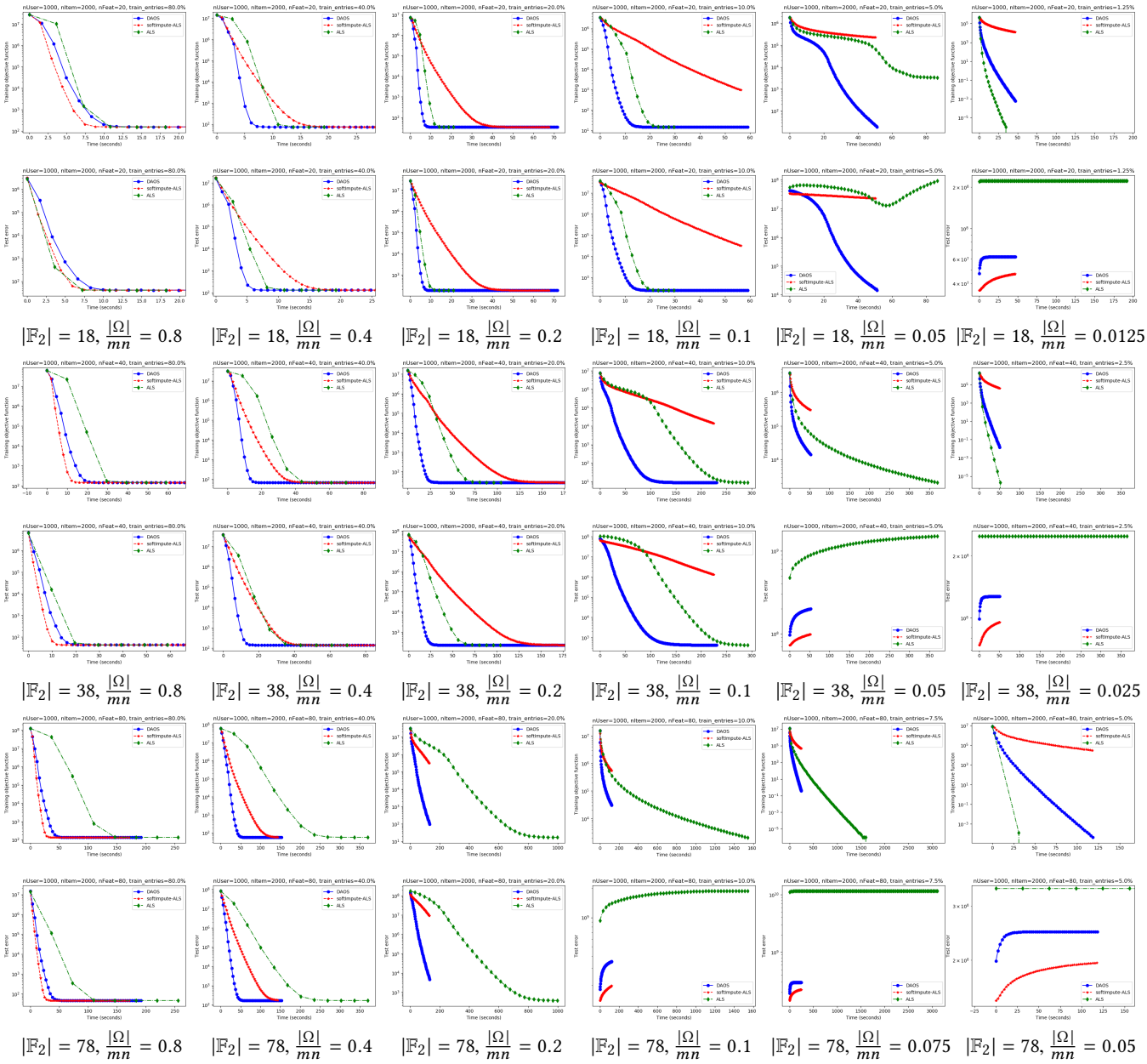


**Figure 2: Results on synthetic data. $|\mathbb{F}_1| = |\mathbb{F}_3| = 0$ and $\lambda = 0.01$. Horizontal axis: computational time in seconds. Vertical axis: training objective (the top row in each duo), or total squared error on test data (the bottom row in each duo). Legend: green diamonds = ALS, red stars = softImpute-ALS, blue circles = DAOS.**

generalization performances of the participating algorithms under different settings of $|\mathbb{F}_2|$ and $\lambda$.

For a given setting of $(|\mathbb{F}_2|, \lambda)$, we run each learning algorithm on the training data. The model produced at each iteration is evaluated using the normalized squared error, $\frac{\sum_{(u,i)}(R_{ui} - X_{u\mathbb{F}_2} Y_{\mathbb{F}_2 i})^2}{\sum_{(u,i)} R_{ui}^2}$, on both the test data and the training data, where the sum is taken over the training set and test set, respectively. Note that the normalizers are constant and do not change the shape of the resulting curves. The results are summarized in Figure 3, with the top row showing the training error and the bottom row showing the test error.

Figure 3 shows that, across all experimental settings, ALS converges the fastest in training error, followed by DAOS and then softImpute-ALS, which is similar to Case II in Section 5.1. It is further noted that ALS achieves the greatest drop in the first iteration, indicating the algorithm is overfitting to training data. However,

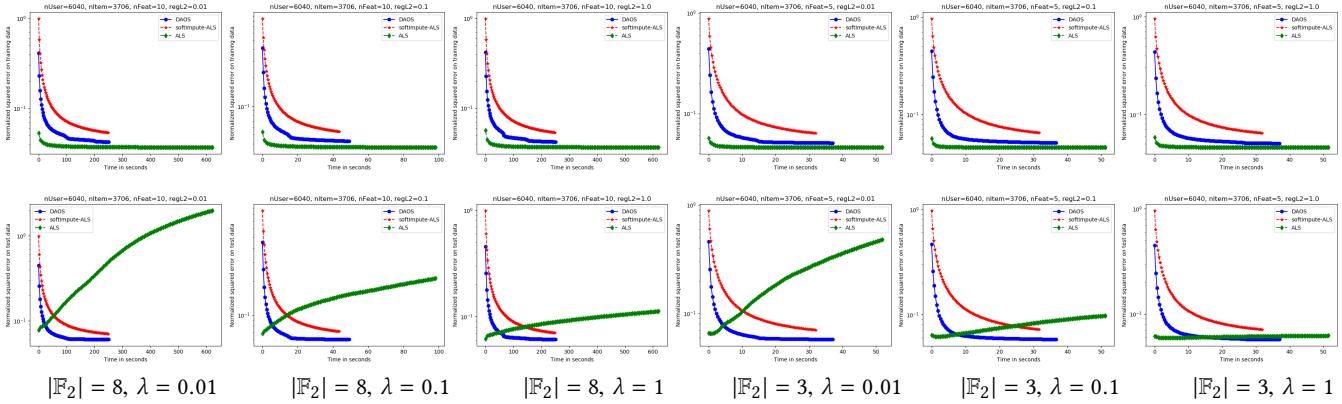


**Figure 3: Results on Movie-Lens 1M Data. $|\mathbb{F}_1| = |\mathbb{F}_3| = 1$ and $|\Omega|/(mn) \approx 0.02234$. Horizontal axis: computational time in seconds. Vertical axis: normalized squared error on training data (top row), or normalized squared error on test data (bottom row). Legend: green diamonds = ALS, red stars = softImpute-ALS, blue circles = DAOS.**

the test error of ALS exhibits great variation and is heavily influenced by $\lambda$ and $|\mathbb{F}_2|$. When the regularization is weak, the model produced by ALS deteriorates in generalization even though it is improving the training error, with this again signaling overfitting. As the regularization becomes stronger, the test error worsens at a lower speed. The overfitting of ALS is mitigated when we fit a simpler model to training data: with $|\mathbb{F}_2|$ decreased from 8 to 3 and $\lambda = 1$, the generalization performance of ALS is able to be maintained at a similar level to that of DAOS.

In contrast to ALS, softImpute-ALS and DAOS exhibit great resistance to overfitting for this dataset. To explain this, we first estimate that the average number of training ratings is $|\Omega|/m = 83$ per user and $|\Omega|/n = 135$ per item, which are both much greater than $|\mathbb{F}_2|$; thus the problems in (3) and (4) are unlikely to be under-determined and what is seen in Case II in Section 5.1 can be prevented here. But still, the training set may not have enough data for each user or item to make generalization happen. The fact that ALS solves (3) and (4) for each user or item independently makes it sensitive to the data sufficiency at the level of users and items. SoftImpute-ALS and DAOS do not suffer from this, because they solve (3) and (4) simultaneously for all users or items. Through data augmentation, each user can exploit the data of similar users, and it is this information transfer that makes softImpute-ALS and DAOS resistant to data scarcity at the user or item level. As long as similar users (items) have enough data in total, the data can be utilized to the benefit of all users or items in question.

## 6 CONCLUSIONS AND FUTURE WORK

We have presented a new algorithm, named *Data Augmentation with Optimal Step-size (DAOS)*, to solve collaborative filtering problems. The algorithm is designed to learn MF-PDF models, a generalized version of matrix factorization to allow simultaneous update of bias terms and factor matrices. The algorithm builds upon softImpute-ALS, maintaining almost the same computational complexity and yet achieving greater objective improvement in each iteration. Theoretical analysis shows that the mechanism underlying the additional improvement is amplification of the surrogate gap. The improvement is roughly proportional to the number of missing rating entries, making DAOS to approach ALS in making big progress per iteration. With this combined with the low computational complexity per iteration, DAOS is able to outperform ALS and softImpute-ALS in most typical problem settings. Future work includes using MF-PDF for content-based collaborative filtering.

## REFERENCES


[1] B. Bakker and T. Heskes. 2003. Task clustering and gating for Bayesian multitask learning. *Journal of Machine Learning Research* 4 (2003), 83–99.

[2] RM Bell and Y Koren. 2007. Scalable collaborative filtering with jointly derived neighborhood interpolation weights. In *Seventh IEEE International Conference on Data Mining (ICDM 2007)*. IEEE CS Press, 43–52.

[3] D. P. Bertsekas. 1999. *Nonlinear Programming (2nd Edition)*. Athena Scientific.

[4] Emmanuel J Candes and Yaniv Plan. 2010. Matrix completion with noise. *Proc. IEEE* 98, 6 (2010), 925–936.

[5] E.J. Candès and B. Recht. 2009. Exact Matrix Completion via Convex Optimization. *Found Comput Math* 9 (2009), 717–772.

[6] R. Caruana. 1997. Multi-task Learning. *Machine Learning* 28 (1997), 41–75.

[7] A. Dempster, N. Laird, and D. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society B* 39 (1977), 1–38.

[8] Michael D. Ekstrand, John T. Riedl, and Joseph A. Konstan. 2011. Collaborative Filtering Recommender Systems. *Foundations and Trends® in Human–Computer Interaction* 4, 2 (2011), 81–173. https://doi.org/10.1561/1100000009

[9] Trevor Hastie, Rahul Mazumder, Jason D. Lee, and Reza Zadeh. 2015. Matrix Completion and Low-Rank SVD via Fast Alternating Least Squares. *Journal of Machine Learning Research* 16, 104 (2015), 3367–3402.

[10] Y Koren and R Bell. 2015. Advances in collaborative filtering. In *Recommender systems handbook*. Springer, Boston, MA, 77–118.

[11] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *IEEE Computer* (2009), 42–49.

[12] Guangcan Liu, Qingshan Liu, and Xiaotong Yuan. 2017. A New Theory for Matrix Completion. In *Advances in Neural Information Processing Systems*, Vol. 30. 785–794.

[13] J. F. C. Mota, N. Deligiannis, and M. R. D. Rodrigues. 2017. Compressed Sensing With Prior Information: Strategies, Geometry, and Bounds. *IEEE Transactions on Information Theory* 63, 7 (2017), 4472–4496.

[14] Peter Neal and Theodore Kypraios. 2015. Exact Bayesian inference via data augmentation. *Statistics and Computing* 25 (2015), 333–347.

[15] A. Ramlatchan, M. Yang, Q. Liu, M. Li, J. Wang, and Y. Li. 2018. A survey of matrix completion methods for recommendation systems. *Big Data Mining and Analytics* 1, 4 (2018), 308–323.

[16] Connor Shorten and Taghi M. Khoshgoftaar. 2019. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data* 6, 60 (2019).

**Table 1: A complete description of the DAOS algorithm**

Input:

- Training ratings $\{R_{ui} : (u,i) \in \Omega\}$
- Pre-defined factors (PDF) $X_{:\bar{\mathbb{P}}}$ and $Y_{\bar{\mathbb{Q}}:}$

Output: Model parameters $X_{:\mathbb{P}}$ and $Y_{\mathbb{Q}:}$

Initialization:

- Initial values of model parameters $X_{:\mathbb{P}}$ and $Y_{\mathbb{Q}:}$
- Iteration number $t = 0$

Repeat the following steps:

1. $D \leftarrow \left[(R - XY)Y_{\mathbb{P}:}^T - \lambda X_{:\mathbb{P}}\right]\left(\lambda I + Y_{\mathbb{P}:}Y_{\mathbb{P}:}^T\right)^{-1}$
2. $\alpha \leftarrow \sum_{(u,i)\in\Omega}(R_{ui} - X_{u:}Y_{:i})(D_{u:}Y_{\mathbb{P}i}) - \lambda \operatorname{tr}(X_{:\mathbb{P}}^T D)$
3. $\beta \leftarrow \sum_{(u,i)\in\Omega}(D_{u:}Y_{\mathbb{P}i})^2 + \lambda\|D\|^2$
4. $g_{2t+1} \leftarrow \sum_{(u,i)\in\Omega}(R_{ui} - X_{u:}Y_{:i})^2 + \lambda\left(\|X_{:\mathbb{P}}\|^2 + \|Y_{\mathbb{Q}:}\|^2\right)$
5. $X_{:\mathbb{P}} \leftarrow X_{:\mathbb{P}} + \frac{\alpha}{\beta}D$
6. $Z \leftarrow \left(\lambda I + X_{:\mathbb{Q}}^T X_{:\mathbb{Q}}\right)^{-1}\left[X_{:\mathbb{Q}}^T(R - XY) - \lambda Y_{\mathbb{Q}:}\right]$
7. $\alpha \leftarrow \sum_{(u,i)\in\Omega}(R_{ui} - X_{u:}Y_{:i})(X_{u\mathbb{Q}}Z_{:i}) - \lambda \operatorname{tr}(Y_{\mathbb{Q}:}Z^T)$
8. $\beta \leftarrow \sum_{(u,i)\in\Omega}(X_{u\mathbb{Q}}Z_{:i})^2 + \lambda\|Z\|^2$
9. $g_{2t+2} \leftarrow \sum_{(u,i)\in\Omega}(R_{ui} - X_{u:}Y_{:i})^2 + \lambda\left(\|X_{:\mathbb{P}}\|^2 + \|Y_{\mathbb{Q}:}\|^2\right)$
10. $Y_{\mathbb{Q}:} \leftarrow Y_{\mathbb{Q}:} + \frac{\alpha}{\beta}Z$
11. Break if any of the following conditions is satisfied
    - The number of iteration $t$ reaches the desired value
    - $g_{2t+2}$ is small enough
    - The sequence $\{t_1 \ldots g_{2t+2}\}$ converges
12. $t \leftarrow t + 1$

# A IMPLEMENTATION AND COMPLEXITY

## A.1 Algorithmic Description

A complete description of the DAOS algorithm is given in Table 1, where steps 1-5 describe the update of user parameters $X_{:\mathbb{P}}$ (including latent factors and biases) while steps 6-10 describe the update of item parameters $Y_{\mathbb{Q}:}$. The convergence is monitored in step 11.

It should be noted that $X_{:\bar{\mathbb{P}}}$ holds the pre-defined user factors (here a column of ones) and $Y_{\bar{\mathbb{Q}}:}$ holds the pre-defined item factors (here a row of ones). However, $[X_{:\bar{\mathbb{P}}}, X_{:\mathbb{P}}] = X$ and $[Y_{\mathbb{Q}:}, Y_{\bar{\mathbb{Q}}:}] = Y$; hence we can write $R - X_{:\bar{\mathbb{P}}}Y_{\bar{\mathbb{P}}:} - X_{:\mathbb{P}}Y_{\mathbb{P}:} = R - XY$, with this concise notation used throughout Table 1. This notation is possible because we are using $(X_{:\mathbb{P}}, Y_{\mathbb{Q}:})$ to denote both the current parameters and the updated parameters (the current parameters are denoted as $(\tilde{X}_{:\mathbb{P}}, \tilde{Y}_{\mathbb{Q}:})$ in previous sections). Accordingly we are using "$\leftarrow$" instead of "$=$" to represent the update.

Removing steps 2-3 and step 7-8 from Table 1 and fixing $\alpha = \beta = 1$, one obtains the softImpute-ALS algorithm.

## A.2 Time Complexity

The complete list of computations performed in Table 1 is provided in Table 2, where the computational tasks are listed in such an order that all the computations are performed before they are first used in subsequent tasks. This guarantees that the same computation is not performed repeatedly.

The computational complexity of DAOS can be obtained by adding up all the computations in Table 2, collecting terms of the same order, and ignoring the lower-order terms. The computational complexity of softImpute-ALS can be obtained similarly, except that the computations numbered 1.5-1.7 and 2.5-2.7 should be excluded because they are not required by softImpute-ALS. The results are summarized in Table 3, where the complexity of ALS is also listed for comparison.

| No. | Computation | Time | Notes |
|---|---|---|---|
| 1.1 | $R - XY$ | $O(k\|\Omega\|)$ | $R_{ui} - X_{u:}Y_{:i} = 0, \forall\ (u,i) \in \bar{\Omega}$ |
| 1.2 | $(R - XY)Y_{\mathbb{P}:}^T$ | $O(\|\mathbb{P}\|\|\Omega\|)$ | $R - XY$ is already computed |
| 1.3 | $\left(\lambda I + Y_{\mathbb{P}:}Y_{\mathbb{P}:}^T\right)^{-1}$ | $O(\|\mathbb{P}\|^2 n + \|\mathbb{P}\|^3)$ | |
| 1.4 | $D$ | $O(m\|\mathbb{P}\|^2)$ | All intermediate terms are already computed |
| 1.5 | $\{D_{u:}Y_{\mathbb{P}i}, (u,i) \in \Omega\}$ | $O(\|\mathbb{P}\|\|\Omega\|)$ | |
| 1.6 | $\alpha$ in Step 2 | $O(\|\Omega\| + m\|\mathbb{P}\|)$ | All intermediate terms are already computed |
| 1.7 | $\beta$ in Step 3 | $O(\|\Omega\| + m\|\mathbb{P}\|)$ | All intermediate terms are already computed |
| 1.8 | Step 4 | $O(\|\Omega\| + m\|\mathbb{P}\| + \|\mathbb{Q}\|n)$ | All intermediate terms are already computed |
| 1.9 | Step 5 | $O(m\|\mathbb{P}\|)$ | All intermediate terms are already computed |
| 2.1 | $R - XY$ | $O(k\|\Omega\|)$ | $R_{ui} - X_{u:}Y_{:i} = 0, \forall\ (u,i) \in \bar{\Omega}$ |
| 2.2 | $X_{:\mathbb{Q}}^T(R - XY)$ | $O(\|\mathbb{Q}\|\|\Omega\|)$ | $R - XY$ is already computed |
| 2.3 | $\left(\lambda I + X_{:\mathbb{Q}}^T X_{:\mathbb{Q}}\right)^{-1}$ | $O(m\|\mathbb{Q}\|^2 + \|\mathbb{P}\|^3)$ | |
| 2.4 | $Z$ | $O(\|\mathbb{Q}\|^2 n)$ | All intermediate terms are already computed |
| 2.5 | $\{X_{u\mathbb{Q}}Z_{:i}, (u,i) \in \Omega\}$ | $O(\|\mathbb{Q}\|\|\Omega\|)$ | |
| 2.6 | $\alpha$ in Step 7 | $O(\|\Omega\| + \|\mathbb{Q}\|n)$ | All intermediate terms are already computed |
| 2.7 | $\beta$ in Step 8 | $O(\|\Omega\| + \|\mathbb{Q}\|n)$ | All intermediate terms are already computed |
| 2.8 | Step 9 | $O(\|\Omega\| + m\|\mathbb{P}\| + \|\mathbb{Q}\|n)$ | All intermediate terms are already computed |
| 2.9 | Step 10 | $O(\|\mathbb{Q}\|n)$ | All intermediate terms are already computed |

**Table 2: The complete list of computations performed in Table 1, where $k = |\mathbb{F}_1| + |\mathbb{F}_2| + |\mathbb{F}_3|$, $|\mathbb{P}| = |\mathbb{F}_2| + |\mathbb{F}_3|$, $|\mathbb{Q}| = |\mathbb{F}_1| + |\mathbb{F}_2|$.**

| Algorithm | Time per iteration |
|---|---|
| ALS | $O\left((\|\mathbb{Q}\|^2 + \|\mathbb{Q}\|^2)\|\Omega\| + m\|\mathbb{P}\|^3 + \|\mathbb{Q}\|^3 n\right)$ |
| softImpute-ALS | $O\left((2(k+1) + \|\mathbb{P}\| + \|\mathbb{Q}\|)\,\|\Omega\| + (m+n)(\|\mathbb{P}\|^2 + \|\mathbb{Q}\|^2) + \|\mathbb{P}\|^3 + \|\mathbb{Q}\|^3\right)$ |
| DAOS | $O\left((2(k+3) + 2\|\mathbb{P}\| + 2\|\mathbb{Q}\|)\,\|\Omega\| + (m+n)(\|\mathbb{P}\|^2 + \|\mathbb{Q}\|^2) + \|\mathbb{P}\|^3 + \|\mathbb{Q}\|^3\right)$ |

**Table 3: A comparison of time complexity, where $k = |\mathbb{F}_1| + |\mathbb{F}_2| + |\mathbb{F}_3|$, $|\mathbb{P}| = |\mathbb{F}_2| + |\mathbb{F}_3|$, $|\mathbb{Q}| = |\mathbb{F}_1| + |\mathbb{F}_2|$.**