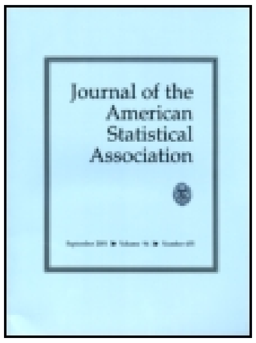

Journal of the American Statistical Association

ISSN: 0162-1459 (Print) 1537-274X (Online) Journal homepage: https://amstat.tandfonline.com/loi/uasa20

Personalized Prediction and Sparsity Pursuit in Latent Factor Models

Yunzhang Zhu, Xiaotong Shen & Changqing Ye

To cite this article: Yunzhang Zhu, Xiaotong Shen & Changqing Ye (2016) Personalized Prediction and Sparsity Pursuit in Latent Factor Models, Journal of the American Statistical Association, 111:513, 241-252, DOI: 10.1080/01621459.2014.999158

To link to this article: https://doi.org/10.1080/01621459.2014.999158

View supplementary material

Accepted author version posted online: 30 Jan 2015.
Published online: 05 May 2016.

Submit your article to this journal

Article views: 1213

View Crossmark data

Citing articles: 3 View citing articles

Full Terms & Conditions of access and use can be found at
https://amstat.tandfonline.com/action/journalInformation?journalCode=uasa20

Supplementary materials for this article are available online. Please go to www.tandfonline.com/r/JASA

Personalized Prediction and Sparsity Pursuit in Latent Factor Models

Yunzhang ZHU, Xiaotong SHEN, and Changqing YE

Personalized information filtering extracts the information specifically relevant to a user, predicting his/her preference over a large number of items, based on the opinions of users who think alike or its content. This problem is cast into the framework of regression and classification, where we integrate additional user-specific and content-specific predictors in partial latent models, for higher predictive accuracy. In particular, we factorize a user-over-item preference matrix into a product of two matrices, each representing a user's preference and an item preference by users. Then we propose a likelihood method to seek a sparsest latent factorization, from a class of overcomplete factorizations, possibly with a high percentage of missing values. This promotes additional sparsity beyond rank reduction. Computationally, we design methods based on a "decomposition and combination" strategy, to break large-scale optimization into many small subproblems to solve in a recursive and parallel manner. On this basis, we implement the proposed methods through multi-platform shared-memory parallel programming, and through Mahout, a library for scalable machine learning and data mining, for mapReduce computation. For example, our methods are scalable to a dataset consisting of three billions of observations on a single machine with sufficient memory, having good timings. Both theoretical and numerical investigations show that the proposed methods exhibit a significant improvement in accuracy over state-of-the-art scalable methods. Supplementary materials for this article are available online.

KEY WORDS: Alternating directions; Collaborative filtering; Content-based filtering; Partial latent models; Recommender; Sparse factorization.

介绍什么叫个性化信息过滤

1. INTRODUCTION

Personalized information filtering automates the process of personalized prediction of a user's preference over a large number of items. It has become increasingly important given today's explosive growth of information, having an array of applications in personalized advertising, online news personalization, consumers' recommendation, among others. Personalized information filtering often results in "BIG data" associated with the ever-increasing volume, variety, and velocity of information. Analysis of this type of data requires new statistical and computational treatments beyond those for conventional data. In this article, we cast the problem of personalized information filtering into the framework of regression and classification, argue that using predictors is essential for personalized prediction, and address core issues toward high predictive accuracy and scalability for massive data with billions of observations.

Personalized information filtering can be regarded as multi-response regression and classification involving a large number of responses and users, where it predicts the preference of a user over a large number of items, called personalized prediction, simultaneously for all relevant users. Personalized prediction is summarized by a preference matrix, whose rows and columns correspond to users and items. This problem can be phrased as estimating unknown parameters of high-dimensionality with very few observations, in the presence of high percentage of missing values. However, it differs substantially from matrix completion (Candès, Romberg, and Tao 2006), which does not usually use predictors.

Two major approaches have emerged, namely, collaborative filtering (Lee and Park 2012) and content-based filtering (Van Meteren and Van Someren 2000). The former pools the information across similar users for a specific item, whereas the latter acts on characteristics of the items that a user prefers, on which two kinds of recommender systems Grooveshark and Pandora (Breese, Heckerman, and Kadie 1998; Van Meteren and Van Someren 2000) are built. These two approaches use either nearest neighbors defined by a similarity metric or matrix factorization (Srebro, Rennie, and Jaakkola 2005). A similarity-based method is intuitive but is difficult to handle newer users and items (Park et al. 2006), while a matrix factorization method imputes or constrains out missing values. Despite success, many challenges remain, among which two salient ones are predictive accuracy and scalability. In reality, many statistical methods are unscalable thus impractical (Das, Dalar, and Garg 2007), whose predictive accuracy deteriorates rapidly as the missing percentage escalates. Consequently, how to design a method becomes critical, to yield high accuracy for a dataset of hundreds millions of observations, for instance, the famous MovieLens data consisting of about one hundred thousands (10^5) users and ten thousands (10^4) movies, which amounts to one billion (10^9) ratings with only 1% observed values, see, *http://www.grouplens.org/node/12*.

For personalized prediction, predictors may be available such as users' demographic profiles together with content information such as item-related web-browsing history. Using them in prediction is not common in recommender systems, but its importance has been recognized (Agarwal and Chen 2009; Forbes and Zhu 2011; Nguyen and Zhu 2012). In what follows, we address the aforementioned issues in a general case. In particular, we develop two novel treatments: (1) sparse latent factorizations within the framework of partial latent models, and (2) decomposable likelihood for scalable computation.

信息过滤的文献回顾

本文的两个创新点

信息过滤可以在回归框架下来建模分析

(E-mail: *zhuxx351@umn.edu*), Xiaotong Shen (E-mail: u), and Changqing Ye (E-mail: *cqye@stat.umn.edu*), School of Statistics, University of Minnesota, Minneapolis, MN 55455. Research supported in part by NSF grants DMS-1207771 and DMS-1415500, NIH grants R02GM08153 and R01HL105397. The authors would like to thank Professor Hui Jiang for technical help in OpenMP coding, and thank the editor, the associate editor and anonymous referees for helpful comments and suggestions.

© 2016 American Statistical Association
Journal of the American Statistical Association
March 2016, Vol. 111, No. 513, Theory and Methods
DOI: 10.1080/01621459.2014.999158

本文模型的解释说明，
后面直接看模型

Within the framework of partial latent models, we use user-specific and content-specific covariates, permitting a treatment of multi-response regression and classification with a large number of variables in the presence of high percentage of missing values. This is in contrast to collaborative filtering and content-based filtering. Specifically, we factorize the preference matrix into a product of a user preference matrix $\boldsymbol{A}$ and an item preference matrix $\boldsymbol{B}^T$, each having the same rank as the original matrix, where $\boldsymbol{A}$ represents overall preference as well as user-specific preference, and $\boldsymbol{B}$ denotes overall item preference as well as item-specific preference, both of which correspond to the main and interaction in factor analysis. As is relevant in many real-world situations, we assume that a preference matrix can be well represented by a product of two sparse matrices $\boldsymbol{A}\boldsymbol{B}^T$. Then we seek a sparsest factorization from a class of such overcomplete factorizations. This yields a parsimonious model for given data at hand, which may be viewed as an analogy of basis pursuit from overcomplete dictionary of bases Chen, Donoho, and Saunders (1998). In this situation, a sparse factorization suggests that user j's preference score on item i is an inner product of two sparse vectors—the jth row of $\boldsymbol{A}$ and the ith row of $\boldsymbol{B}$. As suggested by our analysis, a sparsest factorization can be identified using a proposed L_0-method, leading to high accuracy of personalized prediction by the means of sparsity pursuit.

Statistically, we propose regularized likelihood methods to pursue a sparsest factorization to minimize the number of nonzero factorization entries. Our primary method is the L_0-regularization method through a continuous surrogate, with the L_1- and L_2- methods as a by-product. For the L_0-method, we prove that it yields higher accuracy than its counterparts. Two somewhat surprising results are noted. First, the L_0-method is capable of identifying the rank of the true preference matrix, whereas its counterparts are not. Moreover, the L_0-method yields a more parsimonious representation, due to sparsity pursuit beyond rank identification, thus leading to higher accuracy than matrix completion through low rank approximation without covariates. Second, the proposed computational methods are most effective for high missing data, which is unlike an imputation method geared toward low missing data.

Computationally, we decompose the log-likelihood and regularizers over users and items so that nonconvex minimization for maximum likelihood estimation is solved by treating many small subproblems recursively, alleviating high storage costs and permitting parallel computation. Then the proposed methods are implemented through multi-platform shared-memory parallel programming (OpenMP, see, *http://www.openmp.org*), and implemented and tested in Mahout (see, *http://mahout.apache.org*), a library for scalable machine learning and data mining. This enables the methods to be scalable to a dataset with one billion (10^9) observations on a single machine, with good timings.

With respect to latent factors, the proposed method is somewhat related to yet different from that in Agarwal and Chen (2009). Whereas Agarwal and Chen (2009) assumed that latent factors are linear combinations of user/item features, as described in their Equation (5), we directly model classification/regression functions through linear combinations of user/item features together with latent factors. On this ground, sparsity pursuit for an low-rank approximation of the latent factors can be made without assuming such linear dependence that may destroy the sparsity and rank structures. As a result, further dimension reduction is possible. Moreover, a treatment of Monte Carlo EM algorithm for latent factors could be much slower than the proposed optimization-based treatment for a large-scale problem.

和文献做比较

This article is organized in seven parts. Section 2 introduces sparse factorizations in partial latent models, followed by computational methods to identify a sparsest factorization for prediction with missing values in Section 3. Section 4 establishes some theoretical results concerning estimation accuracy of the proposed methods. Section 5 presents some simulated and two real benchmark examples, including 1 M MovieLens with movie categorization and demographic information, and 10 M MovieLens data without demographic information. Section 6 discusses the methodology. Section 7 contains technical proofs.

2. PARTIAL LATENT MODELS AND SPARSE FACTORIZATIONS

2.1 Partial Latent Models

介绍部分潜因子模型

Given an observed preference matrix $\boldsymbol{R} = (r_{ji})_{U\times M}$, with its jith element r_{ji} measuring the jth user's preference on the ith item, we link $r_{ji} = G(\theta_{ji})$ with preference probability or mean parameter $\theta_{ji} = Er_{ji}$ in a generalized linear model, where E denotes the expectation, and $G(\cdot)$ is a link function (McCullagh and Nelder 1990), for instance, the logit function in logistic regression. Associated with each r_{ji} are user-specific and content-specific predictor vectors $\boldsymbol{x}_j = (x_{j1}, \ldots, x_{jU_0})^T$ and $\boldsymbol{y}_i = (y_{1i}, \ldots, y_{M_0 i})^T$. Now we model $\{\theta_{ji}\}$ as a function of predictors and latent factors in an additive fashion in partial latent models:

$$\theta_{ji} = \boldsymbol{x}_j^T\boldsymbol{\alpha} + \boldsymbol{\beta}^T\boldsymbol{y}_i + \boldsymbol{a}_j^T\boldsymbol{b}_i; \tag{1}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_{U_o})^T$ and $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_{M_o})^T$ are vectors of regression parameters, respectively for $\boldsymbol{x}_j$ and $\boldsymbol{y}_i$, and $\boldsymbol{a}_j, \boldsymbol{b}_i$ are K-dimensional unobserved latent vectors, and K is an upper bound of the number of informative latent factors of $\boldsymbol{\Theta}$ explaining variability in $\boldsymbol{R}$, with $\boldsymbol{\Theta}$ a $U \times M$ user-over-item preference matrix defined in (2). Here missing is allowed for $\{\boldsymbol{x}_j, \boldsymbol{y}_i\}$ in (2) in real data analysis. In particular, the corresponding values are set to zero in (1) when some components of $\boldsymbol{x}_j$ and/or $\boldsymbol{y}_i$ are missing. Model (1) can be expressed in a matrix form

$$\boldsymbol{\Theta} = \boldsymbol{A}\boldsymbol{B}^T, \quad \boldsymbol{A} = \begin{pmatrix} \boldsymbol{x_1}^T & \boldsymbol{\beta}^T & \boldsymbol{a_1}^T \\ \boldsymbol{x_2}^T & \boldsymbol{\beta}^T & \boldsymbol{a_2}^T \\ \vdots & \vdots & \vdots \\ \boldsymbol{x_U}^T & \boldsymbol{\beta}^T & \boldsymbol{a_U}^T \end{pmatrix},$$

$$\boldsymbol{B} = \begin{pmatrix} \boldsymbol{\alpha}^T & \boldsymbol{y_1}^T & \boldsymbol{b_1}^T \\ \boldsymbol{\alpha}^T & \boldsymbol{y_2}^T & \boldsymbol{b_2}^T \\ \vdots & \vdots & \vdots \\ \boldsymbol{\alpha}^T & \boldsymbol{y_I}^T & \boldsymbol{b_I}^T \end{pmatrix}. \tag{2}$$

In (1), $\boldsymbol{R}$ is only partially observed over subset Ω of indices, which is indicated by a binary variable $z_{ji} \in \{0, 1\}$, with 1 indicating being observed and $P(z_{ji} = 1) = \delta_{ji}$, where the z_{ji}'s

随机缺失假设

模型解释

are independent when missing is at random. Our goal is to estimate $\mathbf{\Theta}$ as well as the number of informative latent factors, based on $\{r_{ji}, z_{ji}, \mathbf{x}_j, \mathbf{y}_i\}$. Note that selection of informative predictors from a set of candidate predictors $\{\mathbf{x}_j, \mathbf{y}_i\}$ can be performed as well. However, we will not pursue this direction in this article.

Model (1) is highly interpretable in that $\mathbf{x}_j$ and $\mathbf{y}_i$ represent overall or global information regarding users and items, whereas $\mathbf{a}_j$ and $\mathbf{b}_i$ reflect specific or local information for the jth user over the ith item. They can be regarded as the main effects and unobserved interactions as in analysis of variance (ANOVA). Model (1) is useful in several aspects. First, it leverages additional information from $\{\mathbf{x}_j, \mathbf{y}_i\}$, which pushes the latent model (Tipping and Bishop 1999) to the next level by using covariates for prediction. Second, (1) yields models for multi-response regression and classification with a large number of responses, for personalized prediction, in lieu of the latent models without covariates. Statistically, regression and classification of this sort are extremely challenging for personalized prediction due to overparametrization and lack of repeated measurements, user-specific, and item-specific information, where each r_{ji} may be observed at most once or missing. This is in contrast to traditional statistical analysis such as two-way ANOVA. Third, (1) makes personalized prediction possible through pooling information across users and items, which is advantageous over collaborative filtering without content-specific information and content-based filtering without user-specific information.

In (2), $\mathbf{A}$ and $\mathbf{B}$ are called a user preference matrix and an item preference matrix, respectively. Nonzero-columns of $\mathbf{A}$ and $\mathbf{B}$ can be thought of as features for predicting outcome of $\{r_{ji}\}$. In personalized information filtering, each nonzero-column of $\mathbf{A}$ is either a user-specific predictor or an unobserved latent factor governing a user's preference over items, whereas each nonzero-column of $\mathbf{B}$ is either a content-specific predictor or an unobserved latent factor governing an item's preference by users. In a sense, matrices $\mathbf{A}$ and $\mathbf{B}$ work together to yield personalized prediction of a user's preference over an item simultaneously for all users and items, where the number of nonzero-columns of $\mathbf{A}$ or $\mathbf{B}$ leads to an estimated number of informative latent factors.

先考虑无协变量的稀疏因子分解

2.2 Sparse Latent Factorizations

For motivation, consider latent factor models (1) without predictors $(\mathbf{x}_j, \mathbf{y}_i)$. Let $\mathbf{\Theta}_0$ be the true parameter matrix. A factorization of $\mathbf{\Theta}_0 = \bar{\mathbf{A}}\bar{\mathbf{B}}^T$ is said to be a latent factorization if

$$\mathbf{\Theta}_0 = \bar{\mathbf{A}}\bar{\mathbf{B}}^T, \bar{\mathbf{A}} = (\mathbf{a}_1, \dots, \mathbf{a}_U)^T, \bar{\mathbf{B}} = (\mathbf{b}_1, \dots, \mathbf{b}_M)^T,$$
$$r(\bar{\mathbf{A}}) = r(\bar{\mathbf{B}}) = r(\mathbf{\Theta}_0) \equiv r_0 \le K, \tag{3}$$

where $r(\cdot)$ denotes the rank of a matrix, and $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$ are $U \times K$ and $M \times K$ matrices that have the same locations of zero-columns simultaneously, that is, if the jth column of $\bar{\mathbf{A}}$ is identical to zero, so is the corresponding one of $\bar{\mathbf{B}}$, and vice versa. Note that representations in (3) are overcomplete and involve myriad factorizations, including the one defined by the singular value decomposition (SVD) of $\mathbf{\Theta}_0$. For instance, $\mathbf{\Theta}_0$ can be expressed as follows

$$\mathbf{\Theta}_0 = \begin{pmatrix} 2 & 1 \\ 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$$
$$= \begin{pmatrix} -1.376 & -0.325 \\ -0.851 & 0.526 \end{pmatrix}\begin{pmatrix} -1.376 & -0.851 \\ -0.325 & 0.526 \end{pmatrix}. \tag{4}$$

The first latent factorization with six nonzero entries is sparser than the second one involving eight, which is preferred in estimation of $\mathbf{\Theta}_0$ from a point of view of dimension reduction. As showed in Lemma 4, the number of nonzero elements in a sparsest factorization is no greater than $(M + U - r(\mathbf{\Theta}_0) + 1)r(\mathbf{\Theta}_0)$, which is substantially less than the dimension of $\mathbf{\Theta}_0$, (U, M) when U and M are large. This is in contrast to principal component analysis (PCA) in a different context, where orthogonality is imposed to $\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}$ for an unique factorization, that is, $\bar{\mathbf{A}}^T\bar{\mathbf{A}}$ and $\bar{\mathbf{B}}^T\bar{\mathbf{B}}$ are diagonal. Therefore, we seek, among many factorizations in (3), a sparsest factorization $(\mathbf{A}_0, \mathbf{B}_0) = \arg\min_{\{\mathbf{\Theta}_0=\bar{\mathbf{A}}\bar{\mathbf{B}}^T, (\bar{\mathbf{A}},\bar{\mathbf{B}}) \text{ satisfies } (3)\}}(\|\bar{\mathbf{A}}\|_0 + \|\bar{\mathbf{B}}\|_0)$ to minimize $\|\bar{\mathbf{A}}\|_0 + \|\bar{\mathbf{B}}\|_0$, where $\|\mathbf{A}\|_q$ denotes the L_q norm of the vectorization of a matrix $\mathbf{A}$. This permits a sparser factorization than the SVD factorization imposed by orthogonality. In general, this treatment for (3) without $(\mathbf{x}_j, \mathbf{y}_i)$ is readily generalized to (3) with $(\mathbf{x}_j, \mathbf{y}_i)$. In short, seeking a sparsest factorization in (3) leads to further dimension reduction thus higher accuracy of prediction.

最稀疏分解的三大好处

In (3), benefits of pursuit of a sparsest factorization are threefolded. First, a sparser factorization in terms of $(\mathbf{A}_0, \mathbf{B}_0)$ can be obtained by removing additional redundant entries to realize further sparsity within nonzero-columns of $\mathbf{A}_0$ and $\mathbf{B}_0$. This leads to a sparser factorization than one without doing so. In this sense, a factorization of this type is more preferable over a matrix factorization defined by low rank approximation and PCA. Second, estimation of $r(\mathbf{\Theta}_0)$ can be performed by removing redundant columns of $\mathbf{A}_0$ and $\mathbf{B}_0$ simultaneously given $\mathbf{\Theta}_0 = \mathbf{A}_0\mathbf{B}_0^T$, which also yields an estimated $r(\mathbf{\Theta}_0)$ by-product. This is especially useful in the presence of high percentage of missing ratings. Third, a general matrix $\mathbf{\Theta}_0$, sparse or nonsparse, can be expressed in terms of a sparse factorization as illustrated in (4), hence that it can be estimated through pursuit of a sparsest factorization. In summary, pursuit of a sparsest factorization is achieved by the means of identifying zero entries of $\mathbf{A}_0$ and $\mathbf{B}_0$.

The factorization (3) can be also interpreted an overcomplete dictionary $\mathbf{A}$ with K atoms, multiplied by $\mathbf{B}^T$ whose columns corresponding to decomposition coefficients of columns of $\mathbf{R}$ against the dictionary $\mathbf{A}$. In this regard, this interpretation brings (3) close to sparse dictionary learning in Mairal et al. (2009), where $\mathbf{A}$ is a known dictionary consisting of possibly some base functions, which requires $K > \min(U, M)$ for an overcomplete dictionary.

3. METHODS FOR MISSING VALUES

Given (1) and (3), assume, without of loss of generality, that the marginal likelihood of $\{z_{ji}, \mathbf{x}_j, \mathbf{y}_i\}$ is independent of the parameters $\{\boldsymbol{\alpha}, \boldsymbol{\beta}, \mathbf{a}_j, \mathbf{b}_i\}$. Assuming ignorable missing, or the conditional probability of $(r_{ji}, \mathbf{x}_j, \mathbf{y}_i)$ given z_{ji} is the same as the unconditional probability of $(r_{ji}, \mathbf{x}_j, \mathbf{y}_i)$, we obtain a negative log-likelihood or empirical loss function of $\{r_{ji}\}_{j=1,i=1}^{U,M}$

本文主要内容和目标函数及求解

given $\{z_{ji}, \boldsymbol{x}_j, \boldsymbol{y}_i\}_{j=1,i=1}^{U,M}$, after ignoring parameter-independent terms involving the marginal distribution of $\{z_{ji}, \boldsymbol{x}_j, \boldsymbol{y}_i\}$, which can be written as

$$\sum_{(j,i)\in\Omega} l\left(r_{ji}, \boldsymbol{x}_j^T\boldsymbol{\alpha} + \boldsymbol{\beta}^T\boldsymbol{y}_i + \boldsymbol{a}_j\boldsymbol{b}_i^T\right)$$

注意L(.)与j无关系，r_ji属于同一type

$$= \sum_{j=1}^{U}\sum_{i=1}^{M} w_{ji} l\left(r_{ji}, \boldsymbol{x}_j\boldsymbol{\alpha}^T + \boldsymbol{y}_i\boldsymbol{\beta}^T + \boldsymbol{a}_j^T\boldsymbol{b}_i\right), \qquad (5)$$

where $l(r_{ji}, \theta_{ji})$ is the negative log-likelihood (loss function) of r_{ji} given $\{z_{ji}, \boldsymbol{x}_j, \boldsymbol{y}_i\}$, and $w_{ji} = 1$ if $(j,i) \in \Omega$, $w_{ji} = 0$ otherwise. For nonignorable missing, modeling the distribution of z_{ji} given $(r_{ji}, \boldsymbol{x}_j, \boldsymbol{y}_i)$ may be required. We refer to Little and Rubin (2002) for a more discussion about relevant issues.

The choice of $l(\cdot,\cdot)$ depends on models underlying the observed data. If $l(r_{ji}, \theta_{ji}) = (r_{ji} - \theta_{ji})^2$ in (5), then it yields the SVD of $\boldsymbol{R}$, which is useful for continuous response r_{ij}. For ordinal response r_{ji}, a model such as the proportional odd model (McCullagh and Nelder 1990) may be useful, where

$$l(r_{ji}, \theta_{ji}) = -\sum_{t=1}^{L} \delta_t(r_{ji}) \log\left(\frac{\exp(\mu_t(r_{ji}))}{1+\exp(\mu_t(r_{ji}))} - \frac{\exp(\mu_{t-1}(r_{ji}))}{1+\exp(\mu_{t-1}(r_{ji}))}\right), (j,i) \in \Omega, \qquad (6)$$

where $\delta_t(r_{ji}) = I(r_{ji} = t)$ and $\mu_t(r_{ji}) = \mu_t + \boldsymbol{x}_j^T\boldsymbol{\alpha} + \boldsymbol{\beta}^T\boldsymbol{y}_i + \boldsymbol{a}_j^T\boldsymbol{b}_i$; $\mu_0(r_{ji}) = 0$, and $P(r_{ji} \le t) = \frac{\exp(\mu_t(r_{ji}))}{1+\exp(\mu_t(r_{ji}))}$; $t = 1, \ldots, L$.

For incomplete data, a common treatment is imputation of missing values to construct the likelihood of pseudo complete data, which is mainly for convenience (Little and Rubin 2002). Unfortunately, however, such a treatment may not be scalable and feasible with high percentage of missing values, as in our situation. Our strategy is to work with the likelihood of incomplete data (5) and adopt an approach of "decomposing and combining" to break (5) into many nearly independent subproblems to solve recursively. Details are given in what follows.

3.1 Sparsity Pursuit

This section introduces our methods for estimating a sparsest factorization as well as reconstruction of $\boldsymbol{\Theta}$, which is a nonconvex problem itself. These methods are designed so that they can be implemented through mapReduce for distributed computation (Dean and Ghemawat 2008), which is an important consideration for scalability. Other competing methods such as their constrained counterpart will not be considered.

Statistically, we develop methods to achieve three objectives simultaneously: (1) identifying a sparsest factorization of $\boldsymbol{\Theta}$ in terms of $(\boldsymbol{A}, \boldsymbol{B})$; (2) estimating $\boldsymbol{\Theta}$ through sparse $(\boldsymbol{A}, \boldsymbol{B})$ estimates; (3) determining the number of informative latent factors. For estimation, the number of parameters amounts to $U_0 + M_0 + UM$, which greatly exceeds the sample size $|\Omega|$ due to a high missing percentage in $\boldsymbol{R}$. To pursue sparsity and prevent overfitting, we regularize (5) through row by row regularization:

提出的方法实现3个目标

$$S_1(\boldsymbol{\alpha}, \boldsymbol{\beta}, \tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}) = \sum_{(j,i)\in\Omega} l\left(r_{ji}, \boldsymbol{x}_j^T\boldsymbol{\alpha} + \boldsymbol{\beta}^T\boldsymbol{y}_i + \boldsymbol{a}_j^T\boldsymbol{b}_i\right) + \lambda\left(\sum_{j=1}^{U}\|\boldsymbol{a}_j\|_1 + \sum_{i=1}^{M}\|\boldsymbol{b}_i\|_1\right) \qquad (7)$$

$$S_0(\boldsymbol{\alpha}, \boldsymbol{\beta}, \tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}) = \sum_{(j,i)\in\Omega} l\left(r_{ji}, \boldsymbol{x}_j^T\boldsymbol{\alpha} + \boldsymbol{\beta}^T\boldsymbol{y}_i + \boldsymbol{a}_j^T\boldsymbol{b}_i\right) + \lambda\left(\sum_{j=1}^{U}\|\boldsymbol{a}_j\|_0 + \sum_{i=1}^{M}\|\boldsymbol{b}_i\|_0\right), \qquad (8)$$

where $\tilde{\boldsymbol{A}} = (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_U)^T$, $\tilde{\boldsymbol{B}} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_M)^T$, $\|\cdot\|_q$ denotes the L_q-norm of a vector of length K for row by row regularization, and λ is a nonnegative tuning parameter. For efficient computation, we replace the L_0-function in (8) by its continuous surrogate, the truncated L_1-function $J(u) = \frac{1}{\tau}\min(|u|, \tau)$ Shen, Pan, and Zhu (2012), to yield our cost function:

$$S_0(\boldsymbol{\alpha}, \boldsymbol{\beta}, \tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}}) = \sum_{(j,i)\in\Omega} l\left(r_{ji}, \boldsymbol{x}_j^T\boldsymbol{\alpha} + \boldsymbol{\beta}^T\boldsymbol{y}_i + \boldsymbol{a}_j^T\boldsymbol{b}_i\right) + \lambda\left(\sum_{j=1}^{U}\sum_{k=1}^{K} J(|a_{jk}|) + \sum_{i=1}^{M}\sum_{k=1}^{K} J(|b_{ik}|)\right), \qquad (9)$$

where τ is another tuning parameter, and $J(u)$ approximates the L_0-function as $\tau \to 0^+$. Minimization of (7) and (9) with respect to $\{\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{a}_j, \boldsymbol{b}_i\}$ yields the regularized maximum likelihood estimates $(\hat{\boldsymbol{A}}^{L_1}, \hat{\boldsymbol{B}}^{L_1})$ and $(\hat{\boldsymbol{A}}^{L_0}, \hat{\boldsymbol{B}}^{L_0})$. Importantly, regularization in (7)–(9) is imposed to $\boldsymbol{A}$ and $\boldsymbol{B}$ in the exactly same fashion so that the latent factorization property in (3) is satisfied by our estimates.

Computationally, we develop strategies to solve large-scale nonconvex minimization described in (7) and (9). For (7), as in the foregoing discussion, we decompose it into many small subproblems to solve recursively by a blockwise coordinate decent method with a maximum block improvement (Chen et al. 2012). This strategy yields parallelization and mapReduce (Dean and Ghemawat 2008) for fast computation, reducing memory requirement. In particular, we employ a blockwise updating scheme with three blocks, corresponding to the main effects $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ involving all observations, individual user preference vectors $\{\boldsymbol{a}_j\}$, and individual item preference vectors $\{\boldsymbol{b}_j\}$. For each block, the estimates are updated while the other blocks are held fixed at the current values. Then we circle through the three blocks until convergence, where circling proceeds with the largest amount of block improvement.

To update $\hat{\boldsymbol{a}}_j$ given the rest; $j = 1, \ldots, U$, we treat $\boldsymbol{a}_j$ as unknown parameters and minimize, after ignoring terms independent of $\boldsymbol{a}_j$ in (7),

$$\sum_{i\in R_{j\cdot}} l\left(r_{ji}, \boldsymbol{x}_j^T\hat{\boldsymbol{\alpha}} + \hat{\boldsymbol{\beta}}^T\boldsymbol{y}_i + \boldsymbol{a}_j^T\hat{\boldsymbol{b}}_i\right) + \lambda\|\boldsymbol{a}_j\|_1, \qquad (10)$$

over a set $R_{j\cdot}$ of items that user j's preference scores have observed, which is a K-dimensional generalized Lasso problem with sample size $|R_{j\cdot}|$. Importantly, (10) involves user j alone, which separates this user from the rest, and becomes ideal for parallelization and mapReduce. Similarly, from (7), we update $\hat{\boldsymbol{b}}_i$ in (11); $i = 1, \ldots, M$, by minimizing

$$\sum_{j \in R_{\cdot i}} l\big(r_{ji}, \boldsymbol{x}_j^T \hat{\boldsymbol{\alpha}} + \hat{\boldsymbol{\beta}}^T \boldsymbol{y}_i + \hat{\boldsymbol{a}}_j^T \boldsymbol{b}_i\big) + \lambda \|\boldsymbol{b}_i\|_1, \qquad (11)$$

over a set $R_{\cdot i}$ of users who have indicated their preferences on item i. From the view of efficient computation, the above decomposition permits distributed computation over users and items, with each user or each item as one building block. Therefore, (10) and (11) can be parallelized in j and i, respectively over users and items, without cross-referencing to other users and items.

For the main effects updating, we treat $(\boldsymbol{\alpha}, \boldsymbol{\beta})$ as unknown parameters and minimize

$$\sum_{(j,i) \in \Omega} l\big(r_{ji}, \boldsymbol{x}_j^T \boldsymbol{\alpha} + \boldsymbol{\beta}^T \boldsymbol{y}_i + \hat{\boldsymbol{a}}_j^T \hat{\boldsymbol{b}}_i\big) \qquad (12)$$

to yield estimates $(\hat{\boldsymbol{\alpha}}, \hat{\boldsymbol{\beta}})$. This is $(U_0 + M_0)$-dimensional regression, which is a convex problem when $\{\hat{\boldsymbol{a}}_j\}$ and $\{\hat{\boldsymbol{b}}_i\}$ are held fixed.

For (12), we use an analytic formula integrated with sequential updating over observations by regressing $r_{ji} - \hat{\boldsymbol{a}}_j^T \hat{\boldsymbol{b}}_i$ on $\{\boldsymbol{x}_j, \boldsymbol{y}_i\}$ for the l_2-loss, where the process of sequential updating can be parallelized over observations. For a general loss, we consider use a second-order method involving the gradient and the Hessian of the cost function in (12), which is less efficient than the l_2-case. For (10) and (11), we use fast Lasso implementation of Liu and Ye (2009) for the l_2-loss $l(r_{ji}, \theta_{ji}) = (r_{ji} - \theta_{ji})^2$, where standardization of covariates is applied as usual.

For efficient computation of (10) and (11) and reducing memory requirement, we store data as a row-based sparse matrix and a column-based matrix, respectively for user (row) updating and item (column) updating. This enables efficient updating with searching user- or item-specific information. Algorithm 1 summarizes our computational strategy for the L_1-method, where parallel computation is implemented inside each of the three blocks through OpenMP.

For alternating updating between $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$, it seems that there is one identifiability issue due to scaling with respect to the product $\boldsymbol{A}\boldsymbol{B}^T$, that is, $\tilde{\boldsymbol{A}}\tilde{\boldsymbol{B}}^T = (k\tilde{\boldsymbol{A}})(k^{-1}\tilde{\boldsymbol{B}})^T$ for any nonzero constant k. To circumvent this difficulty, we propose a equal-scaling strategy performing an additional scaling step after each alternating updating. This is to ensure that column-wise scales of $\tilde{\boldsymbol{A}}$ are the same as those of $\tilde{\boldsymbol{B}}$. This strategy is performed for each column of $\tilde{\boldsymbol{A}} = (\tilde{\boldsymbol{A}}_1, \ldots, \tilde{\boldsymbol{A}}_K)$ and $\tilde{\boldsymbol{B}} = (\tilde{\boldsymbol{B}}_1, \ldots, \tilde{\boldsymbol{B}}_K)$. Specifically, we update the solution $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}})$ at the present step: $\hat{\boldsymbol{A}}_i = \tilde{\boldsymbol{A}}_i \sqrt{\frac{\|\tilde{\boldsymbol{B}}_i\|_1}{\|\tilde{\boldsymbol{A}}_i\|_1}}; \hat{\boldsymbol{B}}_i = \tilde{\boldsymbol{B}}_i \sqrt{\frac{\|\tilde{\boldsymbol{A}}_i\|_1}{\|\tilde{\boldsymbol{B}}_i\|_1}}$, $i : 1 \le i \le K$, $\|\tilde{\boldsymbol{A}}_i\|_1, \|\tilde{\boldsymbol{B}}_i\|_1 \neq 0$, and $\hat{\boldsymbol{A}}_i = \hat{\boldsymbol{B}}_i = \boldsymbol{0}$ otherwise, such that $\tilde{\boldsymbol{A}}\tilde{\boldsymbol{B}}^T = \hat{\boldsymbol{A}}\hat{\boldsymbol{B}}^T$, and $\|\hat{\boldsymbol{A}}_i\|_1 = \|\hat{\boldsymbol{B}}_i\|_1, i = 1, \ldots, K$. The following lemma ensures that this rescaling does not increase the objective function.

Lemma 1. The equal scaling strategy does not increase the cost function value. The same property holds for the L_2-penalty, where $\|\cdot\|_2^2$ is used rather than $\|\cdot\|_1$.

Algorithm 1 allows for path-following computation, where λ increases from a small λ-value. Several implementation details need attention. First, the initial value for $(\boldsymbol{A}, \boldsymbol{B})$ seems important for computation efficiency. Based on our limited experience, a better initial value results in faster convergence than a random starting point. In our case, we may use the solution of the L_2-method, as described in Section 3.2, to be an initial value. For the L_2 method, the initial value for $(\boldsymbol{A}, \boldsymbol{B})$ at the first λ-value can be determined as follows. The first U_0 columns of $\boldsymbol{A}$ and the first M_0 columns of $\boldsymbol{B}$ are set to the row averages of observed predictors, and the $U_0 + M_0 + 1$ column of $\boldsymbol{A}$ and $\boldsymbol{B}$ are set to the row averages of $\boldsymbol{R}$, with other columns to be random. Then, they are set to the solution at the adjacent λ-value to use the so called "warm-start." This strategy is employed to speed up convergence, and to avoid the solution being trapped at a stationary point. Second, the order of updating over users or items appears to be important. In fact, one may choose updating in a random order in the absence of any prior knowledge, but higher priority should be given to more active users and items, as in online prediction. Finally, K is usually set between 30 and 50 with $K = 200$ for the most extreme cases.

[illegible]

①初值选择

Algorithm 1. Parallel computation for the L_1-method with missing values

```
1: (Initialization) Input ratings $r_{ji}$, the upper bound $K$, tuning
   parameter $\lambda$, stopping criterion $\epsilon = 10^{-4}$, $m = 1$, initial
   value for $(\boldsymbol{\alpha}^{(0)}, \boldsymbol{\beta}^{(0)}, \tilde{\boldsymbol{A}}^{(0)}, \tilde{\boldsymbol{B}}^{(0)})$.
2: (Block Improvement) At $m$th iteration, perform parallel
   computation for each block:
     • For each user $j$, solve (10) with $\hat{\boldsymbol{\alpha}} = \boldsymbol{\alpha}^{(m-1)}$,
       $\hat{\boldsymbol{\beta}} = \boldsymbol{\beta}^{(m-1)}$, $\hat{\boldsymbol{b}}_i = \tilde{\boldsymbol{b}}_i^{(m-1)}$, and denote the solution
       as $\tilde{\boldsymbol{a}}_j^*$; $j = 1, \ldots, U$. Also let $I_1^{(m)} = 1 - \frac{S_1(\boldsymbol{\alpha}^{(m-1)}, \boldsymbol{\beta}^{(m-1)}, \tilde{\boldsymbol{A}}^*, \tilde{\boldsymbol{B}}^{(m-1)})}{S_1(\boldsymbol{\alpha}^{(m-1)}, \boldsymbol{\beta}^{(m-1)}, \tilde{\boldsymbol{A}}^{(m-1)}, \tilde{\boldsymbol{B}}^{(m-1)})}$.
     • For each item $i$, solve (11) with $\hat{\boldsymbol{\alpha}} = \boldsymbol{\alpha}^{(m-1)}$,
       $\hat{\boldsymbol{\beta}} = \boldsymbol{\beta}^{(m-1)}$, $\hat{\boldsymbol{a}}_j = \tilde{\boldsymbol{a}}_j^{(m-1)}$, and denote the solution
       as $\tilde{\boldsymbol{b}}_i^*$; $i = 1, \ldots, M$. Also let $I_2^{(m)} = 1 - \frac{S_1(\boldsymbol{\alpha}^{(m-1)}, \boldsymbol{\beta}^{(m-1)}, \tilde{\boldsymbol{A}}^{(m-1)}, \tilde{\boldsymbol{B}}^*)}{S_1(\boldsymbol{\alpha}^{(m-1)}, \boldsymbol{\beta}^{(m-1)}, \tilde{\boldsymbol{A}}^{(m-1)}, \tilde{\boldsymbol{B}}^{(m-1)})}$.
     • Solve (12) with $\hat{\boldsymbol{a}}_j = \tilde{\boldsymbol{a}}_j^{(m-1)}$, $\hat{\boldsymbol{b}}_i = \tilde{\boldsymbol{b}}_i^{(m-1)}$, and denote
       that solution as $(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*)$. Also let $I_3^{(m)} = 1 - \frac{S_1(\boldsymbol{\alpha}^*, \boldsymbol{\beta}^*, \tilde{\boldsymbol{A}}^{(m-1)}, \tilde{\boldsymbol{B}}^{(m-1)})}{S_1(\boldsymbol{\alpha}^{(m-1)}, \boldsymbol{\beta}^{(m-1)}, \tilde{\boldsymbol{A}}^{(m-1)}, \tilde{\boldsymbol{B}}^{(m-1)})}$.
3: (Maximum Improvement)
     • if $I_1^{(m)} = \max\{I_1^{(m)}, I_2^{(m)}, I_3^{(m)}\}$, only update $\tilde{\boldsymbol{A}}^{(m)} = \tilde{\boldsymbol{A}}^*$
       and the other components remain the same.
     • if $I_2^{(m)} = \max\{I_1^{(m)}, I_2^{(m)}, I_3^{(m)}\}$, only update $\tilde{\boldsymbol{B}}^{(m)} =$
       $\tilde{\boldsymbol{B}}^*$ and the other components remain the same.
     • if $I_3^{(m)} = \max\{I_1^{(m)}, I_2^{(m)}, I_3^{(m)}\}$, only update $(\tilde{\boldsymbol{\alpha}}^{(m)},$
       $\tilde{\boldsymbol{\beta}}^{(m)}) = (\tilde{\boldsymbol{\alpha}}^*, \tilde{\boldsymbol{\beta}}^*)$ and the other components remain the
       same.
4: (Equal Scaling) Apply the equal scaling strategy for the
   $L_1$-method,as described above if $\tilde{\boldsymbol{A}}$ or $\tilde{\boldsymbol{B}}$ is updated in
   Step 3.
5: (Stopping Criterion) if $\max\{I_1^{(m)}, I_2^{(m)}, I_3^{(m)}\} < \epsilon$, stop
   and return$(\boldsymbol{\alpha}^{L_1}, \boldsymbol{\beta}^{L_1}, \tilde{\mathbf{A}}^{L_1}, \tilde{\mathbf{B}}^{L_1}) = (\hat{\boldsymbol{\alpha}}^{(m)}, \hat{\boldsymbol{\beta}}^{(m)}, \tilde{\mathbf{A}}^{(m)}, \tilde{\mathbf{B}}^{(m)})$.
   Otherwise, set $m = m + 1$ and go to Step 2.
```

Concerning memory requirement, (10) and (11) are the jth user-specific and the ith item-specific, which are solved separately for small subproblems.

For the L_0-method, we employ the same strategy as in (7), except that we replace (10) and (11) by (13) and (14), respectively. In particular, we solve

$$\underset{\boldsymbol{a}_j}{\text{minimize}} \quad \sum_{i\in R_{j\cdot}} l\left(r_{ji}, \boldsymbol{x}_j^T\hat{\boldsymbol{\alpha}} + \hat{\boldsymbol{\beta}}^T\boldsymbol{y}_i + \boldsymbol{a}_j^T\hat{\boldsymbol{b}}_i\right) + \lambda\sum_{k=1}^K J(|a_{jk}|), \tag{13}$$

and

$$\underset{\boldsymbol{b}_i}{\text{minimize}} \quad \sum_{j\in R_{\cdot i}} l\left(r_{ji}, \boldsymbol{x}_j^T\hat{\boldsymbol{\alpha}} + \hat{\boldsymbol{\beta}}^T\boldsymbol{y}_i + \hat{\boldsymbol{a}}_j^T\boldsymbol{b}_i\right) + \lambda\sum_{k=1}^K J(|b_{ik}|). \tag{14}$$

To solve nonconvex problems (13) and (14), we employ difference convex (DC) programming to decompose (13) or (14) into a difference of two convex functions, based on which an iterative scheme is obtained. The reader may consult (Shen, Pan, and Zhu 2012) for details about a DC algorithm. More specifically, for each DC iteration m, given the solution $(\hat{\boldsymbol{a}}_j^{(m-1)}, \hat{\boldsymbol{b}}_i^{(m-1)})$ at DC iteration $m-1$, we solve (15) and (16) alternatively until convergence, to yield $(\hat{\boldsymbol{a}}_j^{(m)}, \hat{\boldsymbol{b}}_i^{(m)})$. For $\hat{\boldsymbol{a}}_j^{(m)}$, we treat $\boldsymbol{a}_j$ as unknown parameters and solve

$$\underset{\boldsymbol{a}_j}{\text{minimize}} \sum_{i\in R_{j\cdot}} l(r_{ji}, \boldsymbol{x}_j^T\boldsymbol{\alpha} + \hat{\boldsymbol{\beta}}^T\boldsymbol{y}_i + \boldsymbol{a}_j^T\hat{\boldsymbol{b}}_i) + \frac{\lambda}{\tau} \times \sum_{k=1}^K w_{Ajk}|a_{jk}|, \tag{15}$$

where $w_{Aj1} = w_{Aj2} = 0$ and $w_{Ajk} = I(\|\hat{a}_{jk}^{(m-1)}\|_2 \le \tau)$; $U_0 + M_0 \le k \le U$. Similarly, we obtain $\hat{\boldsymbol{b}}_i^{(m)}$ by minimizing

$$\sum_{j\in R_{\cdot i}} l(r_{ji}, \boldsymbol{x}_j^T\hat{\boldsymbol{\alpha}} + \boldsymbol{\beta}^T\boldsymbol{y}_i + \hat{\boldsymbol{a}}_j^T\boldsymbol{b}_i) + \frac{\lambda}{\tau}\sum_{k=1}^K w_{Bik}|b_{ik}|, \tag{16}$$

with respect to $\boldsymbol{b}_i$, where $w_{Bi1} = w_{Bi2} = 0$ and $w_{Bik} = I(\|\hat{b}_{ik}^{(m-1)}\|_2 \le \tau)$; $U_0 + M_0 \le k \le M$.

In addition, the equal scaling strategy, as described by the L_1-method, is employed similarly with the L_1-norm replaced by the L_2-norm such that $\tilde{\boldsymbol{A}}\tilde{\boldsymbol{B}}^T = \hat{\boldsymbol{A}}\hat{\boldsymbol{B}}^T$, and $\|\hat{\boldsymbol{A}}_i\|_2 = \|\hat{\boldsymbol{B}}_i\|_2$; $i = 1, \dots, K$, where $(\hat{\boldsymbol{A}}, \hat{\boldsymbol{B}})$ is updated from the present solution $(\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}})$. Now let $(\boldsymbol{W}_A, \boldsymbol{W}_B)$ be weight matrices whose jkth entries are w_{Ajk} and w_{Bjk}.

Next, we establish the convergence properties of the algorithms as well as properties of the estimates. Before proceeding, let $S_l(\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3)$ be the cost function for the L_l-method, where $\boldsymbol{s}_1 = \{\boldsymbol{\alpha}, \boldsymbol{\beta}\}$, $\boldsymbol{s}_2 = \{\boldsymbol{a}_j\}$, and $\boldsymbol{s}_3 = \{\boldsymbol{b}_i\}$ represent the corresponding three blocks. Then $(\boldsymbol{s}_1^*, \boldsymbol{s}_2^*, \boldsymbol{s}_3^*)$ is a stationary point of $S_j(\boldsymbol{s}_1, \boldsymbol{s}_2, \boldsymbol{s}_3)$ if

$$s_l^* = \underset{s_l\in D_l, l=1,2,3}{\text{argmin}} \; S_j(s_1^*, \dots, s_{l-1}^*, s_l, s_{l+1}^*, \dots, s_3^*); \quad l = 1, 2, 3,$$

where D_l is a compact domain for s_l. As indicated by lemma below, the algorithms converge to a stationary point of the cost function.

Lemma 2 (Statistical properties). The estimates $(\boldsymbol{\alpha}^{L_1}, \boldsymbol{\beta}^{L_1}, \tilde{\mathbf{A}}^{L_1}, \tilde{\mathbf{B}}^{L_1})$ and $(\boldsymbol{\alpha}^{L_0}, \boldsymbol{\beta}^{L_0}, \tilde{\mathbf{A}}^{L_0}, \tilde{\mathbf{B}}^{L_0})$, computed from Algorithms 1 and a L_0-version of Algorithm 1, are stationary points of $S_1(\boldsymbol{\alpha}, \boldsymbol{\beta}, \tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}})$ and $S_0(\boldsymbol{\alpha}, \boldsymbol{\beta}, \tilde{\boldsymbol{A}}, \tilde{\boldsymbol{B}})$, respectively.

The storage cost for our methods is minimal as only user-specific or item-specific information is stored for solving (10) and (13) or (11) and (14). The computational complexity is $(2(U+M)\text{Las} + 2\text{Reg})I_1I_2$, where Las denotes the complexity of solving single weighted Lasso with K variables and $\max(M, U)$ observations, Reg is that of solving single U_0-dimensional or M_0-dimensional regression, and I_1 and I_2 are the number of blockwise iteration and DC iteration, respectively. Based on our experience, I_1 and I_2 are about $30 \sim 40$ and $3 \sim 4$.

3.2 The L_2-Method

In the literature, (17) without predictors is studied using a maximum margin matrix factorization method proposed in Srebro, Rennie, and Jaakkola (2005). The cost function is $\min_{\boldsymbol{A},\boldsymbol{B}}(\sum_{j=1}^U\sum_{i=1}^M w_{ji}(r_{ji} - \boldsymbol{a}_j^T\boldsymbol{b}_i)^2 + \lambda(\sum_{j=1}^U\|\boldsymbol{a}_j\|^2 + \sum_{i=1}^M\|\boldsymbol{b}_i\|^2))$, where a ridge regularizer is employed. The corresponding nonconvex minimization problem is solved by an alternate least squares algorithm (Zhou et al. 2008) or a stochastic gradient descent algorithm (Takács et al. 2009).

In this section, we generalize our partial latent factor models to this situation to leverage predictors. This L_2 method will be studied further and compared with the L_0- and L_1-methods. The cost function for the L_2-method is written as

$$\min_{\boldsymbol{A},\boldsymbol{B}}\left(\sum_{j=1}^U\sum_{i=1}^M w_{ji}l\left(r_{ji}, \boldsymbol{x}_j^T\boldsymbol{\alpha} + \boldsymbol{\beta}^T\boldsymbol{y}_i + \boldsymbol{a}_j^T\boldsymbol{b}_i\right) + \lambda\left(\sum_{j=1}^U\|\boldsymbol{a}_j\|^2 + \sum_{i=1}^M\|\boldsymbol{b}_i\|^2\right)\right), \tag{17}$$

where $\lambda > 0$ is a regularization parameter controlling predictive accuracy. Note that no sparse solutions are expected for $(\boldsymbol{A}, \boldsymbol{B})$ in (17), as in ridge regression. For (17), we employ the same computational strategy as before except (10) and (11) are replaced by a ridge penalty.

Lemma 3 establishes a connection between (17) with complete data and (5) with trace-norm minimization. It says that in the case of complete data the L_2-method estimates $r(\boldsymbol{\Theta})$ through the trace-norm that is an upper convex envelope of $r(\boldsymbol{\Theta})$. Note, however, such a result may not be expected for (17) with missing observations, suggested by our simulations.

Lemma 3. In the case of complete data, when $K = \min(U, M)$, minimizing (5) with respect to $(\boldsymbol{A}, \boldsymbol{B})$ reduces to

$$\min_{\boldsymbol{\Theta}}\left(\|\boldsymbol{\Theta} - \boldsymbol{R}\|_F^2 + 2\lambda\|\boldsymbol{\Theta}\|_*\right), \tag{18}$$

where $\boldsymbol{\Theta} = \boldsymbol{A}\boldsymbol{B}^T$, and $\|\cdot\|_F$ and $\|\boldsymbol{\Theta}\|_* = \sum_{i=1}^{\min(U,M)}\sigma_i$ are the Frobenius-norm and trace-norm, respectively, and σ_i is the ith singular value of $\boldsymbol{\Theta}$. Hence the solution of (5) is unique with respect to $\boldsymbol{A}\boldsymbol{B}^T$ although it may not be so in $(\boldsymbol{A}, \boldsymbol{B})$.

4. THEORY

This section is devoted to theoretical investigation of pursuit of a sparsest factorization in terms of accuracy of reconstructing $\boldsymbol{\Theta}$ with missing values. In particular, we derive recovery error rate for reconstructing $\boldsymbol{\Theta}$, for the L_0-, L_1-, and L_2-methods, as a function of the sample size $|\Omega|$, tuning parameter λ or (λ, τ), and a quantity that we call the degree of sparseness. In addition, we will compare these methods to understand the role of a various regularizer plays in pursuing a sparse factorization.

4.1 Main Results

First, we introduce the degree of sparseness given factorizations in (3). The degree of sparseness s_q for the L_q-norm is defined to be $s_q = \min_{\{\boldsymbol{\Theta}_0=\boldsymbol{A}\boldsymbol{B}^T,(\boldsymbol{A},\boldsymbol{B})\text{ satisfying (3)}\}}(\|\boldsymbol{A}\|_q + \|\boldsymbol{B}\|_q)$; $q = 0, 1,$ and $s_2 \equiv \min_{\{\boldsymbol{\Theta}_0=\boldsymbol{A}\boldsymbol{B}^T,(\boldsymbol{A},\boldsymbol{B})\text{ satisfying }\in(3)\}}(\|\boldsymbol{A}\|_2^2 + \|\boldsymbol{B}\|_2^2)$. See Lemma 5 for a connection among s_0, s_1, and s_2. In a sense, these methods aim to different aspects of factorization, resulting dramatically different statistical properties of the estimates.

Second, we define our parameter space $\mathcal{F}$. Let $L > 0$ be a constant controlling the scale of $\boldsymbol{A}$ and $\boldsymbol{B}$. For personalized information filtering, the support of $\{r_{ji}\}$ such as a preference score is usually finite. It is then sensible to assume that $\|\boldsymbol{A}\|_\infty \le L$ and $\|\boldsymbol{B}\|_\infty \le L$, which is equivalent to that $\max_{j,i}(\|\boldsymbol{x}_j\|_\infty, \|\boldsymbol{y}_i\|_\infty, \|\boldsymbol{a}_j\|_\infty, \|\boldsymbol{b}_i\|_\infty) \le L$ and $\max(\|\boldsymbol{\alpha}\|_\infty, \|\boldsymbol{\beta}\|_\infty) \le L$. Then $\mathcal{F}$ is defined as $\{\boldsymbol{\Theta} = \boldsymbol{A}\boldsymbol{B}^T : (\boldsymbol{A}, \boldsymbol{B})$ satisfies (3), $\|\boldsymbol{A}\|_\infty \le L$, $\|\boldsymbol{B}\|_\infty \le L$, $\boldsymbol{A} \in \mathbb{M}(M, K)$, $\boldsymbol{B} \in \mathbb{M}(U, K)\}$, where $\mathbb{M}(M, K)$ is a class of $M \times K$ matrices, and $\|\cdot\|_\infty$ denotes the sup-norm of a matrix.

Third, we define a distance over $\mathcal{F}$. For any $\boldsymbol{\Theta}_l = (\theta_{ji}^l)$; $l = 1, 2$, define the Hellinger-distance $h(\boldsymbol{\Theta}_1, \boldsymbol{\Theta}_2)$ to be $(MU)^{-1}\sum_{j=1}^M\sum_{i=1}^U h(\theta_{ji}^1, \theta_{ji}^2)$, where $h(\theta_{ji}^1, \theta_{ji}^2) \equiv \int(f^{1/2}(r_{ji}, z_{ji}, \theta_{ji}^1) - f^{1/2}(r_{ji}, z_{ji}, \theta_{ji}^2))^2 d\mu(r_{ji}, z_{ji})$ is the Hellinger-distance for the ji component, $f(r_{ji}, z_{ji}, \theta_{ji})$ is the probability density of (r_{ji}, z_{ji}) given $(\boldsymbol{x}_j, \boldsymbol{y}_i)$, and $\mu(\cdot)$ is a dominating measure.

The following assumption is made to require that the likelihood function is smooth.

Assumption A (Smoothness of likelihood). For some constant $d_0 > 0$, any θ_{ji}^1 and θ_{ji}^2,

$$\begin{aligned}&\left|f^{1/2}\left(r_{ji}, z_{ji}, \theta_{ji}^1\right) - f^{1/2}\left(r_{ji}, z_{ji}, \theta_{ji}^2\right)\right| \\ &\quad \le G(r_{ji}, \delta_{ji})\left|\theta_{ji}^1 - \theta_{ji}^2\right|; \quad j = 1, \ldots, M, i = 1, \ldots, U,\end{aligned}$$

with $\sup_{1\le j\le U, 1\le i\le M} EG(r_{ji}, \delta_{ji}) \le d_0$.

Assume the existence of the maximum likelihood estimates $\hat{\boldsymbol{\Theta}}^{L_0}$, $\hat{\boldsymbol{\Theta}}^{L_1}$, and $\hat{\boldsymbol{\Theta}}^{L_2}$, corresponding to L_0-, L_1-, and L_2-norm regularization by minimizing (7), (8), and (17) over $\mathcal{F}$, respectively. Note that existence of an approximated maximum likelihood estimate is assured if the log-likelihood function is bounded above.

Theorem 1 presents finite-sample error bounds for $\hat{\boldsymbol{\Theta}}^{L_0}$ to reconstruct $\boldsymbol{\Theta}_0$ in terms of the Hellinger-distance. Let P be the probability of (r_{ji}, z_{ji}) under $\boldsymbol{\Theta}_0$ given $\{\boldsymbol{x}_j, \boldsymbol{y}_i\}$.

Theorem 1 (Error bound for the L_0-method). Under Assumption A, for $\hat{\boldsymbol{\Theta}}^{L_0}$, if $K \ge r_0 \equiv r(\boldsymbol{\Theta}_0)$, then there exists a constant $c_1 > 0$, such that for $(|\Omega|, M, U)$,

$$P\left(h\left(\hat{\boldsymbol{\Theta}}^{L_0}, \boldsymbol{\Theta}_0\right) \ge \varepsilon_{0,|\Omega|}\right) \le 4\exp\left(-c_1|\Omega|\varepsilon_{0,|\Omega|}^2\right), \qquad (19)$$

provided that $\lambda = s_0^{-1}c_3\varepsilon_{0,|\Omega|}^2$, where $\varepsilon_{0,|\Omega|}^2 = \log(\frac{(M+U)K}{s_0})\frac{s_0}{|\Omega|}$, which is $\log(\frac{(M+U)r_0}{s_0})\frac{s_0}{|\Omega|}$ when $K = r_0$ is tuned. As $|\Omega|, M, U \to \infty$,

$$h\left(\hat{\boldsymbol{\Theta}}^{L_0}, \boldsymbol{\Theta}_0\right) = O_p(\varepsilon_{0,|\Omega|}), \text{ and } Eh^q(\hat{\boldsymbol{\Theta}}^{L_0}, \boldsymbol{\Theta}_0) = O\left(\varepsilon_{0,|\Omega|}^q\right),$$

for real number $q \ge 1$, where $O_p(\cdot)$ denotes the stochastic order under P.

Theorem 1 says that $\hat{\boldsymbol{\Theta}}^{L_0}$ reconstructs $\boldsymbol{\Theta}_0$ at a rate $\varepsilon_{0,|\Omega|} \to 0$ in probability P, provided that (1) the sample size is sufficiently large so that $\varepsilon_{0,|\Omega|} \to 0$, equivalently, $|\Omega| >> s_0$, and (2) the chance to observe r_{ji} at any location needs to be bounded away from zero, or $\inf_{1\le j\le U, 1\le i\le M}\delta_{j,i} > 0$. Most importantly, the recovery rate is of order $\varepsilon_{0,|\Omega|} = \sqrt{\log(\frac{(M+U)r_0}{s_0})\frac{s_0}{|\Omega|}}$ when $K = r_0$ is optimized through tuning. In contrast to the recovery rate $\varepsilon_{|\Omega|}^{\text{mat}}$ of matrix completion through low-rank approximation based on ± 1 ratings, see theorem 8 of Srebro, Alon, and Jaakkola (2005), $\varepsilon_{0,|\Omega|} \preceq \varepsilon_{|\Omega|}^{\text{mat}}$, when specializing to the case without predictors with $M_0 = U_0 = 0$, as the latter is not applicable to the case with predictors. Here $a_n \preceq b_n$ means $a_n \le cb_n$ for some $c > 0$, for all sufficiently large n. In particular,

$$\begin{aligned}\varepsilon_{0,|\Omega|} &= \sqrt{\log\left(\frac{(M+U)r_0}{s_0}\right)\frac{s_0}{|\Omega|}} \\ &\le \sqrt{\log\left(\frac{(M+U)r_0}{(M+U-r_0+)r_0}\right)\frac{(M+U-r_0+1)r_0}{|\Omega|}} \\ &\sim \sqrt{\frac{(M+U-r_0+1)r_0}{|\Omega|}} \\ &\preceq \sqrt{\frac{(M+U)r_0}{|\Omega|}\log\frac{r_0|\Omega|}{(M+U)}} = \varepsilon_{|\Omega|}^{\text{mat}}.\end{aligned}$$

This is because $\varepsilon_{0,|\Omega|}$ decreases in s_0, where $s_0 \le (M + U - r_0 + 1)r_0$ by Lemma 4. This is expected because additional dimension reduction is achieved after a low rank approximation is identified, whereas a method of matrix completion through low-rank approximation does not share this property.

The next lemma says that the degree of sparseness defined by the L_0-norm s_0 is upper bounded by the effective degree of freedom for rank estimation.

Lemma 4 (Connection between sparse factorization and rank estimation). Let $r(\boldsymbol{\Theta}_0) = r_0$. Then $s_0 = \|\boldsymbol{A}_0\|_0 + \|\boldsymbol{B}_0\|_0 \le (M + U - r_0 + 1)r_0$, where $(\boldsymbol{A}_0, \boldsymbol{B}_0)$ is as defined in (3).

Theorem 2 presents a parallel result of Theorem 1 for the L_1- and L_2- methods.

Theorem 2 (Error bounds for the L_1- and L_2-methods). Under Assumption A, for $\hat{\boldsymbol{\Theta}}^{L_1}$ and $\hat{\boldsymbol{\Theta}}^{L_2}$, if $K \geq r_0$, there exists a constant $c_2 > 0$, such that

$$P\big(h(\hat{\boldsymbol{\Theta}}^{L_1}, \boldsymbol{\Theta}_0) \geq \varepsilon_{1,|\Omega|}\big) \leq 4\exp\big(-c_2|\Omega|\varepsilon^2_{1,|\Omega|}\big), \quad (20)$$

$$P\big(h(\hat{\boldsymbol{\Theta}}^{L_2}, \boldsymbol{\Theta}_0) \geq \varepsilon_{2,|\Omega|}\big) \leq 4\exp\big(-c_2|\Omega|\varepsilon^2_{2,|\Omega|}\big), \quad (21)$$

provided that $\lambda = s_1^{-1}c_3\varepsilon^2_{1,|\Omega|}$ and $\lambda = s_2^{-1}c_3\varepsilon^2_{2,|\Omega|}$ in (20) and (21), respectively, where $\varepsilon_{1,|\Omega|} = \sqrt{\frac{s_1^2\log((M+U)K)}{|\Omega|}}$ and $\varepsilon_{2,|\Omega|} = \sqrt{\frac{(M+U)K\log s_2}{|\Omega|}}$, which reduce to $\varepsilon_{1,|\Omega|} = \sqrt{\frac{s_1^2\log((M+U)r_0)}{|\Omega|}}$ and $\varepsilon_{2,|\Omega|} = \sqrt{\frac{(M+U)r_0\log s_2}{|\Omega|}}$ when $K = r_0$ is tuned. Then the results of Theorem 1 continue to hold in this case with $\hat{\boldsymbol{\Theta}}^{L_0}$, $\varepsilon_{0,|\Omega|}$ replaced by $\hat{\boldsymbol{\Theta}}^{L_q}$, $\varepsilon_{q,|\Omega|}$; $q = 1, 2$.

To contrast the L_1-method with the L_0-method, we note that $\varepsilon_{0,|\Omega|} = \sqrt{\frac{s_1^2\log((M+U)K)}{|\Omega|}} \preceq \varepsilon_{1,|\Omega|} = \sqrt{\frac{s_1^2\log((M+U)r_0)}{|\Omega|}}$ in view of Lemma 5. Moreover, the L_0-method enables to recover $r(\boldsymbol{\Theta}_0)$, whereas the L_1-method does not. This aspect is confirmed by our simulation study in Example 1. Moreover, $\varepsilon_{1,|\Omega|} \preceq \varepsilon_{2,|\Omega|}$, which is anticipated due to sparsity for the L_1-norm.

The following connection between the degree of sparseness measured by the L_0-norm s_0 and the L_1-metric s_1 can be established, when $\boldsymbol{\Theta}_0$ is assumed to be bounded away from zero, which is sensible in personalized information filtering with preference scores.

Lemma 5 (Connection among the degree of sparseness s_q measured by the L_q-norm). The following results hold: $s_1 \geq s_0 c_{\min}$ and $s_2 \geq s_1 c_{\min}$, where $c_{\min}$ is the minimal of nonzero entries of $\tilde{\boldsymbol{A}}_0$ and $\tilde{\boldsymbol{B}}_0$ in the best L_1-factorization $\boldsymbol{\Theta}_0 = \tilde{\boldsymbol{A}}_0\tilde{\boldsymbol{B}}_0^T$. If the entries of $\boldsymbol{\Theta}_0$ are bounded away from zero, so is $c_{\min}$.

In summary, the theory suggests the following recovery rate relation based on the effective degree of freedom determined by each method, that is $\varepsilon_{0,|\Omega|} \preceq \varepsilon^{\text{mat}}_{|\Omega|} \preceq \varepsilon_{1,|\Omega|} \preceq \varepsilon_{2,|\Omega|}$. As suggested by simulation studies in Section 5, the L_0-method is expected to deliver higher accuracy as compared with the other methods.

4.2 Hellinger-Distance and the Kullback–Leibler Pseudo-Distance

In the literature, convergence speed is usually measured by the Kullback–Leibler pseudo distance, for example (Srebro, Alon, and Jaakkola 2005). Here we use the Hellinger-distance as a performance metric. To compare with the existing result, we next establish the equivalence between these two metrics.

First, it is known that $h^2(\boldsymbol{\Theta}_0, \boldsymbol{\Theta}) \leq K(\boldsymbol{\Theta}_0, \boldsymbol{\Theta})$. Second, in the setting of (1), $\theta_{ji} = \boldsymbol{x}_j^T\boldsymbol{\alpha} + \boldsymbol{\beta}^T\boldsymbol{y}_i + \boldsymbol{a}_j^T\boldsymbol{b}_i + \varepsilon_{ji}$; $\varepsilon_{ji} \overset{\text{iid}}{\sim} N(0, \sigma^2)$, where $h^2(\boldsymbol{\Theta}_0, \boldsymbol{\Theta}) = 1 - \exp(-\frac{1}{8UM\sigma^2}\sum_{j=1}^U \sum_{i=1}^M \delta_{ji}(\theta^1_{ji} - \theta^2_{ji})^2)$; $-\log(1 - h^2(\boldsymbol{\Theta}_0, \boldsymbol{\Theta})) = \frac{1}{4}K(\boldsymbol{\Theta}_0, \boldsymbol{\Theta})$ with $K(\boldsymbol{\Theta}^0, \boldsymbol{\Theta}) = (2\sigma^2UM)^{-1}\sum_{j=1}^U\sum_{i=1}^M \delta_{ji}(\theta^0_{ji} - \theta_{ji})^2$. Hence, $h^2(\boldsymbol{\Theta}, \boldsymbol{\Theta}_0) \leq K(\boldsymbol{\Theta}^0, \boldsymbol{\Theta}) \leq \exp(\frac{L^2}{2\sigma^2})h^2(\boldsymbol{\Theta}_0, \boldsymbol{\Theta})$, forany$\boldsymbol{\Theta} \in \mathcal{F}$. Consequently, $h^2(\boldsymbol{\Theta}, \boldsymbol{\Theta}_0)$ and $K(\boldsymbol{\Theta}_0, \boldsymbol{\Theta})$ are equivalent in this normal case. In view of the foregoing result, the convergence rates in Theorems 1 and 2 can be compared to those in Srebro, Alon, and Jaakkola (2005).

5. NUMERICAL EXAMPLES

This section investigates numerical aspects of the proposed methods through simulated and real data. In particular, Section 5.1 performs some simulation studies to examine operating characteristics of the proposed methods, and contrast against three scalable methods—a similarity-based neighborhood recommender system, a trace-norm matrix completion method (Soft-Impute; Mazumder, Hastie, and Tibshirani 2010), and a regression-based latent factor models (Agarwal and Chen 2009). Note that a trace-norm matrix completion method is not generally equivalent to the L_2-method in the presence of missing values, although some equivalence is established for complete data in Lemma 2. Most critically, neither similarity-based neighborhood recommender systems nor the trace-norm matrix completion method enable to use predictors by design. The regression-based latent factor models incorporate user/item features into the modeling of latent factors. Section 5.2 concerns MovieLens data.

For parallel computation, we code the L_2-, L_1-, and L_0-methods in C++ through OpenMP, taking the advantage of multiple threads automatically. For mapReduce computation, we code in JAVA to integrate it through Mahout for Hadoop implementation. Based on our limited numerical experience, our JAVA version is about 6–7 times slower than our C++ version, where slowness may be due to Hadoop implementation of Mahout. In what follows, we shall use our C++ version (OpenMP) on an Intel(R) machine (Core(TM) i7 CPU @ 9500 @ 3.07 GHz) with eight threads.

For similarity-based neighborhood recommender systems, we use routines in Apache Hadoop using mapReduce paradigm in Mahout, where the following similarity metrics are used, including cosine, Pearson's correlation, Euclidean, Spearman, log-likelihood. We shall present only the results for Pearson's correlation similarity measure, because other metrics perform similarly. Here Pearson's correlation similarity measure is $\frac{\langle \boldsymbol{r}_{\cdot i}, \boldsymbol{r}_{\cdot j}\rangle}{\|\boldsymbol{r}_{\cdot i}\|\|\boldsymbol{r}_{\cdot j}\|}$, $\boldsymbol{r}_{\cdot i}$ denotes the ith column vector. For Soft-Imput (Mazumder, Hastie, and Tibshirani 2010) and Agarwal and Chen (2009), we use the MATLAB code in Mazumder, Hastie, and Tibshirani (2010) and C++ code in *https://github.com/beechung/Latent-Factor-Models*, respectively.

Our performance metric is predictive accuracy for unseen observations, measured as the root mean square error RMSE $= \sqrt{\frac{1}{|\Omega^c|}\sum_{(u,i)\in\Omega^c} l(r_{ui}, \hat{r}_{ui})}$, where $l(\cdot,\cdot)$ is a loss for prediction, r_{ui} and $\hat{r}_{ui}$ are the observed and predicted preferences over each user u and item i. In the numerical examples, the l_2 loss is used for estimation and evaluation.

For tuning, we minimize the RMSE over a tuning set with respect to the tuning parameter(s) over a set of uniform grid points with λ in $[0, \lambda_{\max}]$ for L_1 and L_2 methods, and $\tau \in \{0.001, 0.01, 0.1\}$ additionally for the L_0 method, where $\lambda_{\max}$ is the minimal λ value at which all coefficients become zero. For testing, the RMSE is computed for the models evaluated at the estimated tuning parameters.

5.1 Simulated Data

Example 1: Sparse factors. This example generates data according to (2) with $r_{ji} = \alpha_1 x_{i1} + \beta_1 y_{j1} + \beta_2 y_{j2} + \boldsymbol{a}_j \boldsymbol{b}_i^T + \epsilon_{ji}$, where $\alpha_1 = \beta_1 = \beta_2 = 1$ and $\epsilon_{ji}, x_{i1}, y_{j1}, y_{j2}$ follow $N(0, 1)$. Moreover, $\boldsymbol{a}_j$ and $\boldsymbol{b}_i$ are the jth and ith rows of $\boldsymbol{A}$, a 600×3 matrix, and $\boldsymbol{B}$, a 600×3 matrix, where $\boldsymbol{A} = (\underbrace{\boldsymbol{I}_{3\times3}, \ldots, \boldsymbol{I}_{3\times3}}_{100}, \underbrace{-\boldsymbol{I}_{3\times3}, \ldots, -\boldsymbol{I}_{3\times3}}_{100})^T$, $\boldsymbol{I}_{3\times3}$ denotes 3×3 identity matrix and $\boldsymbol{B} = \boldsymbol{A}$. By design, the columns of $\boldsymbol{A}$ and $\boldsymbol{B}$ are orthogonal to each other and sparse. Finally, missing occurs at random with missing probability 0.8, 0.7, which yields roughly about 20%, 30% of the matrix size 600×600, or 72,000, 120,000 training observations.

Example 2: Dense factors. This example mimics the data generating model considered by Agarwal and Chen (2009) in which latent factors are modeled as a linear combination of covariates. Specifically, $r_{ji} = \eta_i + \gamma_j + \alpha_1 x_{i1} + \beta_1 y_{j1} + \beta_2 y_{j2} + (\boldsymbol{a}_j + y_{j1}\boldsymbol{d}_1 + y_{j2}\boldsymbol{d}_2)(\boldsymbol{b}_i + x_{i1}\boldsymbol{g}_1)^T + \epsilon_{ji}$, where $\alpha_1 = \beta_1 = \beta_2 = 4$, and $\epsilon_{ji}, x_{i1}, y_{j1}$, and y_{j2} are sampled from $N(0, 1)$. Moreover, $\eta_i = \gamma_j = 4$, $\boldsymbol{d}_1 = \boldsymbol{d}_2 = \boldsymbol{g}_1 = (4, 4, 4)$, and $\boldsymbol{a}_j$ and $\boldsymbol{b}_i$ are the jth and ith rows of $\boldsymbol{A}$, a 100×3 matrix, and $\boldsymbol{B}$, a 100×3 matrix with entries following $N(0, 1)$.

The following situations will be examined, including (1) prediction with predictors versus without predictors, (2) latent factor rank estimation, and (3) methods with different choices of K, when a true model contains covariates. Moreover, the l_2-loss $l(r, u) = (r - u)^2$ is considered. In simulations, the RMSE value for each method is reported, in addition to the estimated rank $\hat{r}$ and the degree of sparseness $\hat{s}$ for $(\boldsymbol{a}_j, \boldsymbol{b}_i)$.

With regard to prediction, the L_0-method with predictors performs the best across almost all situations. Importantly, inclusion of the three predictors is critical for the L_q-method, as suggested by Tables 1 and 2, with the amount of improvement with predictors over without predictors about 1.3%–2.0% and 1.2%–2.0% respectively for the L_1- and L_0-methods in Example 1, and 12.2%–23.3% and 11.4%–22.5% in Example 2. Moreover, the L_0-method outperforms its L_1 counterpart, which is consistent with the theoretical results in Theorems 2 and 3. Moreover, with predictors, the L_1- and L_0-methods outperform the trace-norm matrix completion method Soft-Impute (Mazumder, Hastie, and Tibshirani 2010) with the amount of improvement about 4.0%–5.7% and 3.5%–5.2% in Example 1, and the regression-based latent factor method—Agarwal & Chen (Agarwal and Chen 2009) about 1742% and 1922% in Example 2, respectively. Excluding the predictors, the L_1-, L_0-methods continue to outperform Soft-Impute and Pearson's similarity-based method in Example 1 and Agarwal and Chen in Example 2. Interestingly, the L_0-method yields a sparser representation as compared to its competitors across all the situations. With regard to rank estimation, the L_0-method enables to estimate $r_0 = r(\boldsymbol{\Theta}_0)$ precisely, whereas the L_1-method does not well estimate $r(\boldsymbol{\Theta}_0)$, and the L_2-method, and Soft-Impute completely miss the mark partly because they are not designed for rank estimation. This partly explains higher performance of L_0-method against its competitors.

Finally, we observe that the value K plays a role of an upper bound for estimating $r(\Theta_0)$. As a matter of matter, estimation for the L_0- and L_1-methods is not sensitive to the choice of K as long as $K \geq r(\Theta_0)$. Moreover, the proposed method outperforms Agarwal and Chen by a wide margin when the model is misspecified in view of the results in Examples 2. This might be due to unstableness of Monte Carlo EM algorithm used by Agarwal and Chen.

Overall, the L_0-method performs the best across all the situations. The L_2-method without predictors and conventional similarity-based collaborative filtering methods are not competitive. Using informative predictors is essential to predictive performance in view of the results for the methods including predictors, including Agarwal and Chen.

5.2 Benchmark: MovieLens Data

This subsection compares the proposed methods against Soft-Imput and Agarwal and Chen as well as a constant model without predictors in terms of predictive performance on two benchmark datasets for personalized prediction, namely, the 1M MovieLens and 10M MovieLens. The 1M and 10M MovieLens datasets were collected by GroupLens Research Project at the University of Minnesota, during a seven-month period from September 19th, 1997 to April 22nd, 1998.

The data are comprised of movie ratings and are available at *http://www.grouplens.org/node/12*. The 1M MovieLens data consist of 1,000,209 anonymous ratings on a five-star scale from 6040 users on 3900 movies, whereas the 10M MovieLens data consist of 10,000,054 ratings from 95,580 users on 10,681 movies. For the 1M data, there are four categorical and one continuous covariates, including four user-related covariates, gender, age, occupation and zip-code, as well as one content-related covariate, genres, with each user having at least 20 ratings. For simplicity, we treat gender, age and occupation as continuous variables, in addition to 19 different movie genres that are reparameterized into 19 binary covariates encoding if certain movie belongs a particular genre, where only the first digit is used for zipcode. For the 10M data, except the demographic information, other variables are available.

For prediction, we split each dataset into three sets, with 60%, 15%, 25%, according to the timestamp of the ratings, following Agarwal and Chen (2009). This splitting is more realistic, but creates a cold-start problem for latent factor model, because the testing set may involve a large number of new user/movie. For a method involving tuning, we use the three parts for training, tuning and testing. For a method that does not require tuning, we use the combined the first two parts to have 75% of the data for training and the remaining 25% for testing. In particular, for tuning, the RMSE over the tuning set is minimized w.r.t. the tuning parameter(s) over a set of grid points in a parallel fashion as in the simulated example, where 50 uniformly grid spaced points over $(0, \lambda_{\max}]$ are used to tune λ, $\tau = 0.001, 0.01, 0.1$ are used to tuning τ. For testing, the RMSE is computed for the models evaluated at the estimated tuning parameters.

For benchmark comparisons, we include more benchmark models denoted "Constant only" model and "Predictor only" model in addition to aforementioned competing methods, where the former and latter use the average ratings and simple linear regression based on user/movie predictors.

Table 1. Averaged RMSE's in the l_2-loss, estimated rank, as well as the estimated degree of sparseness (standard error in parentheses), for various methods, over 100 simulation replications under the l_2-loss in Example 1, with $U = 600$, $M = 600$, $r(\boldsymbol{\Theta}_0) = 3$ and $s_0 = 10\%, 5\%$ denoting the sparsity percentage of $(\boldsymbol{a}_j, \boldsymbol{b}_i)$ for $K = 10$ and $K = 20$, respectively. Here "Soft-Impute," "Pearson," "L_q w predictors," "L_q w/o predictors;" $q = 0, 1, 2$, denotes the trace-norm matrix completion method (Mazumder, Hastie, and Tibshirani 2010) without predictors, the regression-based latent factor models (Agarwal and Chen 2009), Pearson-similarity based collaborative filtering without predictors, the L_q-methods with and without $\{x_1, x_2, y_1\}$, and "NA" means that it is not available or computationally infeasible for the software. The precision is set to 10^{-4} for our methods, and to the default value for a competing method

(K, % training)	Method	RMSE	$\hat{r}$	$\hat{s}$ for $(\boldsymbol{a}_j, \boldsymbol{b}_i)$
	Soft-Impute	1.062 (0.002)	63.27 (7.23)	NA
	Pearson	1.510 (0.002)	NA	NA
(10, 30%)	L_2 w/o predictors	1.040 (0.002)	NA	NA
	L_1 w/o predictors	1.026 (0.005)	7.66 (1.52)	38.28% (1.67%)
	L_0 w/o predictors	1.019 (0.002)	5.0 (0.000)	29.30% (0.12%)
	L_2 w predictors	1.021 (0.002)	NA	NA
	L_1 w predictors	1.012 (0.004)	5.55 (1.60)	18.34% (1.39%)
	L_0 w predictors	1.007 (0.002)	3.0 (0.000)	10.49% (0.06%)
(20, 30%)	L_2 w/o predictors	1.041 (0.002)	NA	NA
	L_1 w/o predictors	1.025 (0.004)	9.82 (2.97)	19.59% (1.12%)
	L_0 w/o predictors	1.021 (0.009)	5.00 (0.000)	14.59% (0.30%)
	L_2 w predictors	1.021 (0.002)	NA	NA
	L_1 w predictors	1.012 (0.004)	6.9 (2.90)	9.46% (0.05%)
	L_0 w predictors	1.007 (0.002)	3.0 (0.000)	5.25% (0.03%)
	Soft-Impute	1.092 (0.003)	68.07 (6.45)	NA
	Pearson	1.518 (0.002)	NA	NA
(10, 20%)	L_2 w/o predictors	1.048 (0.002)	NA	NA
	L_1 w/o predictors	1.038 (0.003)	7.93 (1.42)	37.03% (1.38%)
	L_0 w/o predictors	1.033 (0.002)	5.00 (0.000)	29.95% (0.16%)
	L_2 w predictors	1.025 (0.002)	NA	NA
	L_1 w predictors	1.018 (0.003)	6.47 (1.94)	17.80% (1.31%)
	L_0 w predictors	1.013 (0.001)	3.00 (0.00)	11.10% (0.11%)
(20, 20%)	L_2 w/o predictors	1.049 (0.002)	NA	NA
	L_1 w/o predictors	1.038 (0.003)	10.7 (2.97)	19.02% (0.88%)
	L_0 w/o predictors	1.033 (0.002)	5.00 (0.00)	15.00% (0.01%)
	L_2 w predictors	1.025 (0.002)	NA	NA
	L_1 w predictors	1.018 (0.002)	7.66 (2.49)	9.03% (0.63%)
	L_0 w predictors	1.013 (0.002)	3.00 (0.000)	5.56% (0.05%)

Table 2. Averaged RMSE's in the l_2-loss, estimated rank, as well as the estimated degree of sparseness (Standard Error in parentheses), for various methods, over 100 simulation replications under the l_2-loss in Example 2, with $U = 100$, $M = 100$. Here "Agarwal and Chen," "L_q w predictors," "L_q w/o predictors;" $q = 0, 1, 2$, denote the regression-based latent factor models (Agarwal and Chen 2009), and the L_q-methods with and without $\{x_1, x_2, y_1\}$, and "NA" means that it is not available or computationally infeasible for the software. The precision is set to 10^{-4} for our methods, and to the default value for a competing method

(K, % training)	Method	RMSE	$\hat{r}$	$\hat{s}$ for $(\boldsymbol{a}_j, \boldsymbol{b}_i)$
(3, 50%)	Agarwal and Chen	22.77 (2.13)	NA	NA
(10, 30%)	L_2 w/o predictors	1.465 (0.071)	NA	NA
	L_1 w/o predictors	1.274 (0.027)	6.63 (1.21)	30.6% (1.3%)
	L_0 w/o predictors	1.255 (0.020)	5.10 (0.22)	29.2% (1.7%)
	L_2 w predictors	1.219 (0.037)	NA	NA
	L_1 w predictors	1.135 (0.016)	4.73 (1.41)	30.6% (1.4%)
	L_0 w predictors	1.126 (0.015)	3.04 (0.48)	29.2% (.9%)
(3, 40%)	Agarwal and Chen	22.72 (2.08)	NA	NA
(10, 20%)	L_2 w/o predictors	2.725 (0.077)	NA	NA
	L_1 w/o predictors	1.561 (0.083)	7.32 (1.38)	42.2% (2.8%)
	L_0 w/o predictors	1.510 (0.060)	4.44 (0.55)	40.3% (2.5%)
	L_2 w predictors	1.748 (0.048)	NA	NA
	L_1 w predictors	1.266 (0.049)	4.78 (1.52)	30.1% (1.0%)
	L_0 w predictors	1.233 (0.043)	3.10 (0.33)	29.0% (0.4%)

Table 3. RMSE's for various methods in benchmark data examples based on partitions in terms of rating time of the original data with 60%, 15% and 25% for training, tuning and testing. The 1M data include four users covariates, age, gender and occupation, and one content covariate, genres, whereas the 10M data has one content covariate, genres. Here "Constant only," "Predictor only," "Soft-Impute," "Agarwal and Chen," "L_q w predictors," "L_q w/o predictors;" $q = 0, 1, 2$, denote the constant model, the latent factor model with predictors, the trace-norm matrix completion method (Mazumder, Hastie, and Tibshirani 2010) without predictors, the regression-based latent factor models (Agarwal and Chen 2009), the L_q-methods with and without $\{x_1, x_2, y_1\}$, and and "NA" means that it is not available or computationally infeasible for the software. The precision is set to 10^{-4} for our methods, and to the default value for a competing method

Method	1M MovieLens	10M MovieLens
Constant only	1.1186	1.0214
Predictors only	1.0960	1.0025
Soft-Imput	1.0656	1.0175
Agarwal & Chen	1.0520	1.0185
L_2 w/o predictors	1.0577	1.0152
L_1 w/o predictors	1.0523	1.0105
L_0 w/o predictors	1.0502	1.0104
L_2 w predictors	1.0516	1.0023
L_1 w predictors	1.0480	0.9998
L_0 w predictors	1.0478	0.9995

We report our results in Table 3. With regard to reconstruction of $\mathbf{\Theta}$, the L_0 method outperforms the L_1- and L_2-methods, as well as Soft-Imput, Agarwal and Chen, the "Constant only" model and "Predictor only" model, with the amount of improvement 0.02%–0.03%, 0.04%–0.28%, 1.7%–1.8%, 0.40%–1.9%, 6.8%–2.2%, and 4.6%–0.30% for the 1M and 10M data, respectively. Moreover, the L_q-method with predictors perform better than its counterpart without predictors; $q = 0, 1, 2$, although the amount of improvement is not seen as large as in the simulated example. This may be partly due to that predictors in real datasets are less informative. Moreover, the "Predictor only" model even outperforms the latent factor models w/o predictors for the 10M data, although the R^2-values for the "Predictor only" models using user/movie predictors are only about 3.9% and 3.7% for the 1M and 10M data. In this sense, predictors are critical in prediction, especially in a cold-start situation. More informative predictors may be constructed to further ameliorate prediction. This will be a topic of future research.

With regard to runtime, for the 1M and 10M MovieLens datasets, training and tuning for 50 grid λ-points requires about $20 \sim 30$ seconds and $7 \sim 8$ minutes on average for the L_1-method on a 8-core computer with Intel(R) Core(TM) i7-3770 processors and 16GB of RAM. The L_0-method is about 4 times slower, and the L_2-method is about the same speed if K is relatively small. For large K, however, the L_2-method becomes much slower than the L_1- and L_0- methods. In contrast, Agarwal and Chen based on the Monte Carlo EM algorithm is about 6–10 times slower, which requires about 200 seconds and one hour for the default setting ($K = 3$ and 10 EM iterations), respectively. Moreover, the predictive performance of Agarwal and Chen is sensitive to both K and the number of EM iterations. It is, however, unclear how to choose them without an external tuning set. Of course, adding a tuning set does increase the computational cost. In conclusion, the L_0-method with predictors delivers higher predictive accuracy and runs much faster as compared to existing scalable algorithms.

6. DISCUSSION

This article formulates the problem of personalized information filtering as multi-response regression and classification through partial latent factor models. In addition, a sparse matrix factorization is introduced to reconstruct the user-over-time preference matrix regardless if it is sparse. Several regularization methods are proposed using decomposable cost functions for parallel and mapReduce computation, permitting a treatment of massive data, which a conventional method is incapable to deal with. A general theory is developed to quantify predictive accuracy of the reconstruction in presence of missing observations, where the proposed L_0-method promotes additional sparsity beyond rank estimation, a property other competing methods may not share.

For parallel computation, we implement the l_2-loss for least squares regression. Our methods are scalable to a dataset consisting of billions of observations on a single machine with sufficient memory, whose capability can be further expanded with the help of Hadoop infrastructure. Further investigation is necessary with regard to integration of implicit feedback and explicit feedback into latent factor models, in addition to issues relevant to nonignorable missing.

7. SUPPLEMENTARY MATERIALS

The online supplementary materials contain additional proofs.

[Received July 2013. Revised December 2014.]

REFERENCES

Agarwal, D., and Chen, B. C. (2009), "Regression-Based Latent Factor Models," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 19–28. [241,242,248,249]

Breese, J., Heckerman, D., and Kadie, C. (1998), "Empirical Analysis of Predictive Algorithms for Collaborative Filtering," in *Proceedings of 14th Conference on Uncertainty in Artificial Intelligence*, pp. 43–52. [241]

Candès, E. J., Romberg, J. K., and Tao, T. (2006), "Stable Signal Recovery From Incomplete and Inaccurate Measurements," *Communications on Pure and Applied Mathematics*, 59, 1207–1223. [241]

Chen, B., He, S., Li, Z., and Zhang, S. (2012), "Maximum Block Improvement and Polynomial Optimization," *SIAM Journal on Optimization*, 22, 87–107. [244]

Chen, S., Donoho, D., and Saunders, M. (1998), "Automatic Decomposition by Basis Pursuit," *SIAM Review*, 43, 129–159. [242]

Das, A., Dalar, M., and Garg, A. (2007), "Google News Personalization: Scalable Online Collaborative Filtering," in *Proceedings of the 16th International Conference on World Wide Web*, pp. 271–280. [241]

Dean, J., and Ghemawat, S. (2008), "MapReduce: Simplified Data Processing on Large Clusters," *Communications of the ACM*, 51, 107–113. [244]

Forbes, P., and Zhu, M. (2011), "Content-Boosted Matrix Factorization for Recommender Systems: Experiments With Recipe Recommendation," in *Proceedings of the 5th ACM Conference on Recommender Systems*, pp. 261–264. [241]

Lee, T. Q., and Park, Y. (2012), "A Time-Based Recommender System Using Implicit Feedback," in *Proceeding of the 22nd International World Wide Web Conference*, Lyon, France. [241]

Little, J. A., and Rubin, D. (2002), *Statistical Analysis With Missing Data* (2nd ed.), Hoboken, NJ: Wiley. [244]

Liu, J., and Ye, J. (2009), "Efficient Euclidean Projection in Linear Time," in *Proceedings of the 26th Annual International Conference on Machine Learning*, 2009, pp. 657–664. [245]

Park, S., Pennock, D., Madani, O., Good, N., and DeCoste, D. (2006), "Naive Filterbots for Robust Cold-Start Recommendations," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 699–705. [241]

Mairal, J., Bach, F., Ponce, J., and Sapiro, G. (2009), "Online Dictionary Learning for Sparse Coding," in *Proceedings of the 26th Annual International Conference on Machine Learning*, pp. 689–696. [243]

Mazumder, R., Hastie, T., and Tibshirani, R. (2010), "Spectral Regularization Algorithms for Learning Large Incomplete Matrices," *The Journal of Machine Learning Research*, 11, 2287–2322. [248,249]

McCullagh, P., and Nelder, J. A. (1990), *Generalized Linear Model* (2nd ed.), London: Chapman and Hall. [242,244]

Nguyen, J., and Zhu, M. (2012), "Content-Boosted Matrix Factorization Techniques for Recommender Systems," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 6, 286–301. [241]

Shen, X., Pan, W., and Zhu, Y. (2012), "Likelihood-Based Selection and Sharp Parameter Estimation," *Journal of the American Statistical Association*, 107, 223–232. [244,246]

Srebro, N., Alon, N., and Jaakkola, T. (2005), "Generalization Error Bounds for Collaborative Prediction With Low-Rank Matrices," in *Advances in Neural Information Processing Systems*, (Vol. 17), Cambridge, MA: MIT Press, pp. 5–27. [247,248]

Srebro, N., Rennie, J., and Jaakkola, T. (2005), "Maximum-Margin Matrix Factorization," in *Advances in Neural Information Processing Systems*, (Vol. 17), Cambridge, MA: MIT Press, pp. 1329–1336. [241,246]

Takács, G., Pilászy, I., Németh, B., and Tikk, D. (2009), "Scalable Collaborative Filtering Approaches for Large Recommender Systems," *Journal of Machine Learning Research*, 10, 623–656. [246]

Tipping, M. E., and Bishop, C. M. (1999), "Probabilistic Principal Component Analysis," *Journal of the Royal Statistical Society*, Series B, 61, 611–622. [243]

Van Meteren, R., and Van Someren, M. (2000), "Using Content-Based Filtering for Recommendation," in *Proceedings Machine Learning in New Information Age MLnet–ECML2000 Workshop*, pp. 312–321. [241]

Zhou, Y., Wilkinson, D., Schreiber, R., and Pan, R. (2008), "Large-Scale Collaborative Filtering for the Netflix Prize," in *Proceedings of the 4th international conference on Algorithmic Aspects in Information and Management*, pp. 337–348. [246]