A Fast Spectral Relaxation Approach to Matrix Completion via Kronecker Products

Hui Zhao[†] **Jiuqiang Han**[†] **Naiyan Wang**[‡] **Congfu Xu**[‡] **Zhihua Zhang**[‡]
[†]Department of Automation, Xi'an Jiaotong University,Xi'an,710049,China
[‡]Department of Computer Science and Technology, Zhejiang University
{zhaohui, jqhan}@mail.xjtu.edu.cn winsty@gmail.com xucongfu@zju.edu.cn zhzhang@gmail.com

Abstract

In the existing methods for solving matrix completion, such as singular value thresholding (SVT), soft-impute and fixed point continuation (FPCA) algorithms, it is typically required to repeatedly implement singular value decompositions (SVD) of matrices. When the size of the matrix in question is large, the computational complexity of finding a solution is costly. To reduce this expensive computational complexity, we apply Kronecker products to handle the matrix completion problem. In particular, we propose using Kronecker factorization, which approximates a matrix by the Kronecker product of several matrices of smaller sizes. We introduce Kronecker factorization into the soft-impute framework and devise an effective matrix completion algorithm. Especially when the factorized matrices have about the same sizes, the computational complexity of our algorithm is improved substantially.

Introduction

The matrix completion problem (Cai, Candès, and Shen 2010; Candès and Recht 2008; Keshavan, Montanari, and Oh 2009; Mazumder, Hastie, and Tibshirani 2010; Beck and Teboulle 2009) has become increasingly popular, because it occurs in many applications such as collaborative filtering, image inpainting, predicting missing data in sensor networks, etc. The problem is to complete a data matrix from a few observed entries. In a recommender system, for example, customers mark ratings on goods and vendors then collect the customer's preferences to form a customer-good matrix in which the known entries represent actual ratings. In order to make efficient recommendations, the vendors try to recover the missing entries to predict whether a certain customer would like a certain good.

A typical assumption in the matrix completion problem is that the data matrix in question is low rank or approximately low rank (Candès and Recht 2008). This assumption is reasonable in many instances such as recommender systems. On one hand, only a few factors usually contribute to an customer's taste. On the other hand, the low rank structure suggests that customers can be viewed as a small number of groups and that the customers within each group have similar taste.

Recently, it has been shown that matrix completion is not as ill-posed as originally thought. Srebro, Alon, and Jaakkola (2005) derived useful generalization error bounds for predicting missing entries. Several authors have shown that under certain assumptions on the proportion of the missing entries and locations, most low-rank matrices can be recovered exactly (Candès and Recht 2008; Candès and Tao 2009; Keshavan, Montanari, and Oh 2009).

The key idea of recovering a low-rank matrix is to solve a so-called matrix rank minimization problem. However, this problem is NP-hard. An efficient approach for solving the problem is to relax the matrix rank into the matrix nuclear norm. This relaxation technique yields a convex reconstruction minimization problem, which is tractably solved. In particular, Cai, Candès, and Shen (2010) devised a first-order singular value thresholding (SVT) algorithm for this minimization problem. Mazumder, Hastie, and Tibshirani (2010) then considered a more general convex optimization problem for reconstruction and developed a soft-impute algorithm for solving their problem. Other solutions to the convex relation problem include fixed point continuation (FPCA) and Bregman iterative methods (Ma, Goldfarb, and Chen 2009), the augmented Lagrange multiplier method (Lin et al. 2010; Candès et al. 2009), singular value projection (Jain, Meka, and Dhillon 2010), accelerated proximal gradient algorithm (Toh and Yun 2009), etc..

These methods require repeatedly computing singular value decompositions (SVD). When the size of the matrix in question is large, however, the computational burden is prohibitive. Implementations typically employ a numerical iterative approach to computing SVD such as the Lanczos method, but this does not solve the scaling problem.

In this paper we propose a fast convex relaxation for the matrix completion problem. In particular, we use a matrix approximation factorization via Kronecker products (Van Loan and Pitslanis 1993; Kolda and Bader 2009). Under the nuclear norm relaxation framework, we formulate a set of convex optimization subproblems, each of which is defined on a smaller-size matrix. Thus, the cost of computing SVDs can be mitigated. This leads us to an effective algorithm for handling the matrix completion problem. Compared with the algorithms which use numerical methods computing SVD, our algorithm is readily parallelized.

The paper is organized as follows. The next section re-

views recent developments on the matrix completion problem. Our completion method based on the Kronecker factorization of a matrix is then presented, followed by our experimental evaluations. Finally, we give some concluding remarks.

Problem Formulations

Consider an $n\times p$ real matrix $\mathbf{X} = [x_{ij}]$ with missing entries, let $\Omega \subset \{1,\ldots,n\}\times\{1,\ldots,p\}$ denote the indices of observation entries of $\mathbf{X}$, and let $\bar{\Omega} = \{1,\ldots,n\}\times\{1,\ldots,p\}\setminus\Omega$ be the indices of the missing entries. In order to complete the missing entries, a typical approach is to define an unknown low-rank matrix $\mathbf{Y} = [y_{ij}] \in \mathbb{R}^{n\times p}$ and to formulate the following optimization problem

$$\begin{array}{ll}\min_{\mathbf{Y}} & \mathrm{rank}(\mathbf{Y}) \\ \text{s.t.} & \sum_{(i,j)\in\Omega}(x_{ij}-y_{ij})^2 \le \delta, \end{array} \tag{1}$$

where $\mathrm{rank}(\mathbf{Y})$ represents the rank of the matrix $\mathbf{Y}$ and $\delta \ge 0$ is a parameter controlling the tolerance in training error.

However, it is not tractable to reconstruct $\mathbf{Y}$ from the rank minimization problem in (1), because it is in general an NP-hard problem. Since the nuclear norm of $\mathbf{Y}$ is the best convex approximation of the rank function $\mathrm{rank}(\mathbf{Y})$ over the unit ball of matrices, a feasible approach relaxes the rank minimization into a nuclear norm minimization problem (Candès and Tao 2009; Recht, Fazel, and Parrilo 2007). Let $\|\mathbf{Y}\|_*$ denote the *nuclear norm* of $\mathbf{Y}$. We have

$$\|\mathbf{Y}\|_* = \sum_{i=1}^{r}\sigma_i(\mathbf{Y}),$$

where $r = \min\{n,p\}$ and the $\sigma_i(\mathbf{Y})$ are the singular values of $\mathbf{Y}$. The nuclear norm minimization problem is defined by

$$\begin{array}{ll}\min_{\mathbf{Z}} & \|\mathbf{Y}\|_* \\ \text{s.t.} & \sum_{(i,j)\in\Omega}(x_{ij}-y_{ij})^2 \le \delta. \end{array} \tag{2}$$

Clearly, when $\delta = 0$, Problem (2) reduces to

$$\begin{array}{ll}\min & ||\mathbf{Y}||_* \\ \text{s.t.} & y_{ij} = x_{ij}, \quad \forall (i,j)\in\Omega, \end{array} \tag{3}$$

which has been studied by Cai, Candès, and Shen (2010) and Candès and Tao (2009). However, Mazumder, Hastie, and Tibshirani (2010) argued that (3) is too rigid, possibly resulting in overfitting. In this paper we concentrate on the case in which $\delta > 0$.

The nuclear norm is an effective convex relaxation of the rank function. Moreover, off-the-shelf algorithms such as interior point methods can be used to solve problem (2). However, they are not efficient if the scale of the matrix in question is large. Equivalently, we can reformulate (2) in Lagrangian form as follows

$$\min_{\mathbf{Y}} \ \frac{1}{2}\sum_{(i,j)\in\Omega}(x_{ij}-y_{ij})^2 + \gamma\|\mathbf{Y}\|_*, \tag{4}$$

where $\gamma > 0$. Let $\mathcal{P}_\Omega(\mathbf{X})$ be an $n\times p$ matrix such that its (i,j)th entry is x_{ij} if $(i,j)\in\Omega$ and zero otherwise. We write the problem (4) in matrix form:

$$\min_{\mathbf{Z}} \ \frac{1}{2}\|\mathcal{P}_\Omega(\mathbf{X})-\mathcal{P}_\Omega(\mathbf{Y})\|_F^2 + \gamma\|\mathbf{Y}\|_*, \tag{5}$$

where $\|\mathbf{A}\|_F$ represents the Frobenius norm of $\mathbf{A} = [a_{ij}]$; that is,

$$\|\mathbf{A}\|_F^2 = \sum_{i,j} a_{ij}^2 = \mathrm{tr}(\mathbf{A}\mathbf{A}^T) = \sum_i \sigma_i^2(\mathbf{A}).$$

Recently, Cai, Candès, and Shen (2010) proposed a novel first-order singular value thresholding (SVT) algorithm for the problem (2). The SVT algorithm is based on a notion of *matrix shrinkage operator*.

Definition 1 *(Matrix Shrinkage Operator) Suppose that the matrix $\mathbf{A}$ is an $n\times p$ matrix of rank r. Let the condensed SVD of $\mathbf{A}$ be $\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$, where $\mathbf{U}$ $(n\times r)$ and $\mathbf{V}$ $(p\times r)$ satisfy $\mathbf{U}^T\mathbf{U} = \mathbf{I}_r$ and $\mathbf{V}^T\mathbf{V} = \mathbf{I}_r$, $\mathbf{\Sigma} = \mathsf{diag}(\sigma_1,\ldots,\sigma_r)$ is the $r\times r$ diagonal matrix with $\sigma_1 \ge \cdots \ge \sigma_r > 0$. For any $\tau > 0$, the matrix shrinkage operator $S_\tau(\cdot)$ is defined by $S_\tau(\mathbf{A}) := \mathbf{U}\mathbf{\Sigma}_\tau\mathbf{V}^T$ with*

$$\mathbf{\Sigma}_\tau = \mathsf{diag}([\sigma_1-\tau]_+,\ldots,[\sigma_r-\tau]_+).$$

Here $\mathbf{I}_r$ denotes the $r\times r$ identity matrix and $[t]_+ = \max(t,0)$.

Recently, using the matrix shrinkage operator, Mazumder, Hastie, and Tibshirani (2010) devised a so-called *soft-impute* algorithm for solving (5). The detailed procedure of the soft-impute algorithm is given in Algorithm 1, which computes a series of solutions to (5) with different step sizes (λ). As we see, the algorithm requires an SVD computation of an $n\times p$ matrix at every iteration. When both n and p are large, this is computationally prohibitive.

Algorithm 1 The Soft-Impute Algorithm

1: Initialize $\mathbf{Y}^{(old)} = \mathbf{0}$, $\tau > 0$, tolerance ϵ and create an ordering set $\Lambda = \{\lambda_1,\ldots,\lambda_k\}$ in which $\lambda_i \ge \lambda_{i+1}$ for any i.
2: **for** every fixed $\lambda \in \Lambda$ **do**
3: Compute $\mathbf{M} \leftarrow S_\tau(\mathbf{Y}^{(old)})$.
4: Compute $\mathbf{Y}^{(new)} \leftarrow \mathcal{P}_\Omega(\mathbf{X}) + \mathcal{P}_{\bar{\Omega}}(\mathbf{M})$.
5: **if** $\frac{\|\mathbf{Y}^{(new)}-\mathbf{Y}^{(old)}\|_F}{\|\mathbf{Y}^{(old)}\|_F} < \epsilon$ **then**
6: Assign $\mathbf{Y}_\lambda \leftarrow \mathbf{Y}^{(new)}, \mathbf{Y}^{(old)} \leftarrow \mathbf{Y}^{(new)}$ and go to step 2.
7: **else**
8: Assign $\mathbf{Y}^{(old)} \leftarrow \mathbf{Y}^{(new)}$ and go to step 3.
9: **end if**
10: **end for**
11: Output the sequence of solutions $\mathbf{Y}_{\lambda_1},\ldots,\mathbf{Y}_{\lambda_k}$.

Methodology

Before we present our approach, we give some notation and definitions. For an $n\times m$ matrix $\mathbf{A} = [a_{ij}]$, let $\mathsf{vec}(\mathbf{A}) = (a_{11},\ldots,a_{n1},a_{12},\ldots,a_{nm})^T$ be the $nm\times 1$ vector. In addition, $\mathbf{A}\otimes\mathbf{B} = [a_{ij}\mathbf{B}]$ represents the Kronecker product of $\mathbf{A}$ and $\mathbf{B}$, and $\mathbf{K}_{nm}$ $(nm\times nm)$ is the commutation matrix which transforms $\mathsf{vec}(\mathbf{A})$ into $\mathsf{vec}(\mathbf{A}^T)$ (i.e. $\mathbf{K}_{nm}\mathsf{vec}(\mathbf{A}) = \mathsf{vec}(\mathbf{A}^T)$). Please refer to (Lütkepohl 1996; Magnus and Neudecker 1999) for some properties related to the Kronecker product and the commutation matrix.

Assume that $\mathbf{Y}_i$ for $i = 1, \ldots,$ are $n_i \times p_i$ matrices where $n = \prod_{i=1}^{s} n_i$ and $p = \prod_{i=1}^{s} p_i$. We consider the following relaxation:

$$\min_{\mathbf{Y}_i \in \mathbb{R}^{n_i \times p_i}} \frac{1}{2}\|\mathcal{P}_\Omega(\mathbf{X}) - \mathcal{P}_\Omega(\mathbf{Y})\|_F^2 + \sum_{i=1}^{s} \gamma_i \|\mathbf{Y}_i\|_*, \quad (6)$$

where $\mathbf{Y} = \mathbf{Y}_1 \otimes \mathbf{Y}_2 \otimes \cdots \otimes \mathbf{Y}_s$. For $i = 1, \ldots, s$, let

$$\mathbf{Y}_{-i} = \mathbf{Y}_1 \otimes \cdots \mathbf{Y}_{i-1} \otimes \mathbf{Y}_{i+1} \otimes \cdots \otimes \mathbf{Y}_s,$$

which is $(n/n_i) \times (p/p_i)$. We now treat $\mathbf{Y}_j$ for $j \neq i$ fixed.

Lemma 1 *Assume that $\mathbf{X}$ is fully observed. If the $\mathbf{Y}_j$ for $j \neq i$ are fixed and nonzero, then*

$$h(\mathbf{Y}_i) = \frac{1}{2}\|\mathbf{X} - \mathbf{Y}\|_F^2 + \gamma_i \|\mathbf{Y}_i\|_*$$

is strictly convex in $\mathbf{Y}_i$.

Theorem 1 *Assume that $\mathbf{X}$ is fully observed. If the $\mathbf{Y}_j$ for $j \neq i$ are fixed and nonzero, then the minimum of $h(\mathbf{Y}_i)$ is obtained when*

$$\mathbf{Y}_i = S_{\gamma_i}(\mathbf{Z}_i),$$

where

$$\mathbf{Z}_i = \big(\mathsf{vec}(\mathbf{Y}_{-i}^T)^T \otimes \mathbf{I}_{n_i}\big)\big(\mathbf{I}_{\frac{n}{n_i}} \otimes \mathbf{K}_{n_i, \frac{p}{p_i}}^T\big)\mathbf{X}_i$$

is an $n_i \times p_i$ matrix. Here $\mathbf{X}_i$ is the $(np/p_i) \times p_i$ matrix defined by

$$\mathsf{vec}(\mathbf{X}_i') = \mathsf{vec}(\mathbf{R}_i \mathbf{X}^T \mathbf{Q}_i^T),$$

where $\mathbf{Q}_i = \mathbf{I}_{\alpha_{i-1}} \otimes \mathbf{K}_{\beta_{s-i}, n_i}$ ($n \times n$) with

$$\alpha_0 = \beta_0 = 1,\ \alpha_i = \prod_{j=1}^{i} n_j, \ \text{and}\ \beta_i = \prod_{j=1}^{i} n_{s+1-j}$$

and $\mathbf{R}_i = \mathbf{I}_{\zeta_{i-1}} \otimes \mathbf{K}_{\eta_{s-i}, p_i}$ ($p \times p$) with

$$\zeta_0 = \eta_0 = 1,\ \zeta_i = \prod_{j=1}^{i} p_j, \ \text{and}\ \eta_i = \prod_{j=1}^{i} p_{s+1-j},$$

for $i = 1, \ldots, s$.

The proofs of Lemma 1 and Theorem 1 are given in Appendix . The theorem motivates us to employ an iterative approach to solving the problem in (6).

In the remainder of the paper we consider the special case of $s = 2$ for simplicity. Let $\mathbf{Y}_1 = \mathbf{A} = [a_{ij}]$ ($n_1 \times p_1$) and $\mathbf{Y}_2 = \mathbf{B} = [b_{ij}]$ ($n_2 \times p_2$). Moreover, we partition $\mathbf{X}$ into

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_{11} & \mathbf{X}_{12} & \cdots & \mathbf{X}_{1,p_1} \\ \mathbf{X}_{21} & \mathbf{X}_{22} & \cdots & \mathbf{X}_{2,p_1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{X}_{n_1,1} & \mathbf{X}_{n_1,2} & \cdots & \mathbf{X}_{n_1,p_1} \end{bmatrix}, \quad \mathbf{X}_{ij} \in \mathbb{R}^{n_2 \times p_2}.$$

Using Matlab colon notation, we can write $\mathbf{X}_{ij} = \mathbf{X}((i-1)n_2 + 1 : in_2, (j-1)p_2 + 1 : jp_2)$ for $i = 1, \ldots, n_1; j = 1, \ldots, p_1$. Thus,

$$\phi = \frac{1}{2}\|\mathbf{X} - \mathbf{A} \otimes \mathbf{B}\|_F^2 = \frac{1}{2}\sum_{i=1}^{n_1}\sum_{j=1}^{p_1}\|\mathbf{X}_{ij} - a_{ij}\mathbf{B}\|_F^2$$

$$= \frac{1}{2}\sum_{i=1}^{n_1}\sum_{j=1}^{p_1}\mathrm{tr}[(\mathbf{X}_{ij} - a_{ij}\mathbf{B})(\mathbf{X}_{ij} - a_{ij}\mathbf{B})^T].$$

In order to use the matrix shrinkage operator, we now have that the (i,j)th element of $\mathbf{Z}_1$ is $\mathrm{tr}(\mathbf{X}_{ij}^T\mathbf{B})$, i.e., $[\mathbf{Z}_1]_{ij} = \mathrm{tr}(\mathbf{X}_{ij}^T\mathbf{B})$, and $\mathbf{Z}_2 = \sum_{i=1}^{n_1}\sum_{j=1}^{p_1} a_{ij}\mathbf{X}_{ij}$. We defer the derivations of $\mathbf{Z}_1$ and $\mathbf{Z}_2$ to the appendix.

According to Theorem 1 and Algorithm 1, we immediately have an algorithm for solving the problem in (6), which is summarized in Algorithm 2. We call it the *Kronecker Product-Singular Value Thresholding* (KP-SVT). We now see that at every iteration the algorithm implements the SVD of two matrices with sizes $n_1 \times p_1$ and $n_2 \times p_2$, respectively. Compared with Algorithm 1, the current algorithm is computationally effective. In particular, if $n_1 \approx n_2 \approx \sqrt{n}$ and $p_1 \approx p_2 \approx \sqrt{p}$, our algorithm becomes quite effective. Moreover, we can further speed up the algorithm by setting $s > 2$, letting $\mathbf{Y} = \mathbf{Y}_1 \otimes \mathbf{Y}_2 \otimes \cdots \otimes \mathbf{Y}_s$ for $s > 2$.

Algorithm 2 Kronecker Product-Singular Value Thresholding (KP-SVT)

Initialize $\mathbf{Y} = \mathcal{P}_\Omega(\mathbf{X})$, $\mathbf{A} = \mathbf{1}_{n_1}\mathbf{1}_{p_1}^T$, $\mathbf{B} = \mathbf{1}_{n_2}\mathbf{1}_{p_2}^T$, $\tau > 0$, tolerance ϵ, maxStep $= k$, step $= 1$.
for step $<$ maxStep **do**
  step $=$ step $+ 1$.
  Compute $[\mathbf{A}^{(old)}]_{ij} \leftarrow \mathrm{tr}(\mathbf{Y}_{ij}^T\mathbf{B}^{(old)})$ and $\mathbf{A}^{(new)} \leftarrow S_{\frac{\tau_1}{||\mathbf{B}^{old}||_F^2}}(\frac{\mathbf{A}^{(old)}}{||\mathbf{B}^{(old)}||_F^2})$.
  Compute $\mathbf{B}^{(old)} \leftarrow \sum_{i=1}^{n_1}\sum_{j=1}^{p_1}[\mathbf{A}^{(new)}]_{ij}\mathbf{Y}_{ij}$ and $\mathbf{B}^{(new)} \leftarrow S_{\frac{\tau_2}{||\mathbf{A}^{(new)}||_F^2}}(\frac{\mathbf{B}^{(old)}}{||\mathbf{A}^{new}||_F^2})$.
  Compute $\mathbf{Y}^{(new)} \leftarrow \mathcal{P}_\Omega(\mathbf{X}) + \mathcal{P}_{\bar{\Omega}}(\mathbf{A}^{(new)} \otimes \mathbf{B}^{(new)})$.
  if $\frac{\|\mathbf{Y}^{(new)} - \mathbf{Y}^{(old)}\|_F}{\|\mathbf{Y}^{(old)}\|_F} < \epsilon$ **then**
    $\hat{\mathbf{Y}} = \mathbf{Y}^{(new)}$.
    break.
  end if
end for
Output the solutions $\hat{\mathbf{Y}}$.

Experiments

In this section we first demonstrate the convergence of the KP-SVT algorithm through experiments and discuss how to choose the proper sizes of $\mathbf{A}$ and $\mathbf{B}$. Next, we compare our KP-SVT with the soft-impute algorithm (Mazumder, Hastie, and Tibshirani 2010) and the conventional SVT algorithm (Cai, Candès, and Shen 2010) both in terms of accuracy and efficiency. All experiments are implemented in MATLAB and all the reported results are obtained on a desktop computer with a 2.57 GHz CPU and 4 GB of memory.

Both toy data and real world datasets are used. In our simulations, we generate $n \times p$ matrices $\mathbf{X}$ of rank q by taking $\mathbf{X} = \mathbf{UV} + \text{noise}$, where $\mathbf{U}$ ($n \times q$) and $\mathbf{V}$ ($q \times p$) are independent random matrices with i.i.d. uniform distribution between 0 and 1, and the noise term is the zero-mean Gaussian white noise. The set of observed entries, Ω, is uniformly sampled at random over the indices of the matrix. The eval-

uation criterion is based on the test error as follows:

$$\text{test error} = \frac{||\mathcal{P}_{\tilde{\Omega}}(\mathbf{X} - \hat{\mathbf{Y}})||_F}{||\mathcal{P}_{\tilde{\Omega}}(\mathbf{X})||_F}.$$

We also adopt the Movielens datasets [1] and the Jester joke dataset [2] to evaluate the performance of the algorithm. The root mean squared error (RMSE) over the probe set is used as the evaluation criteria on these datasets. Five datasets are used in this section, Movielens 100K contains 943 users and 1690 movies with 100,000 ratings (1-5), and Movielens 1M contains 6,040 users and 4,000 movies with 1 million ratings. Jester joke contains over 1.7 million continuous ratings (-10.00 to +10.00) of 150 jokes from 63,974 users. Toy_1 is a 1000×1000 matrix with 20% known entries whose $\text{rank} = 10$, and Toy_2 is a 5000×5000 matrix with 10% known entries whose $\text{rank} = 20$. For real world datasets, 80% ratings are used for training set and 20% ratings are used for test set.

Convergence Analysis

We conduct the KP-SVT algorithm on toy datasets and recommendation datasets. Figure 1 depicts the iterative processes of the algorithm on these datasets. From Figure 1, we see that the algorithm converges after a relatively small number of iterations.

The Choice of the Sizes of A and B

The main idea of our KP-SVT algorithm is using $\mathbf{A} \otimes \mathbf{B}$ instead of $\mathbf{Y}$ to approximate the original matrix $\mathbf{X}$. Following the property of Kronecker Products, we have $n = n_1 \times n_2$ and $p = p_1 \times p_2$. If we set $n_1 = n$ and $p_1 = p$, our model degenerates to the conventional low-rank approximation problem. Thus, it is very important how to choose the appropriate values of n_1 and p_1.

The main calculation bottleneck of SVT-based low-rank approximation algorithms lies in the calculation of the SVD decompositions of $n \times p$ matrices, while in KP-SVT we need only to calculate the SVD decompositions of two sets of smaller-size matrices. In order to choose an appropriate size, we make use of the following criterion,

$$\begin{aligned} \min \quad & |n_1 - n_2| + |p_1 - p_2|, \\ \text{s.t.} \quad & n = n_1 \times n_2 \quad \text{and} \quad p = p_1 \times p_2. \end{aligned}$$

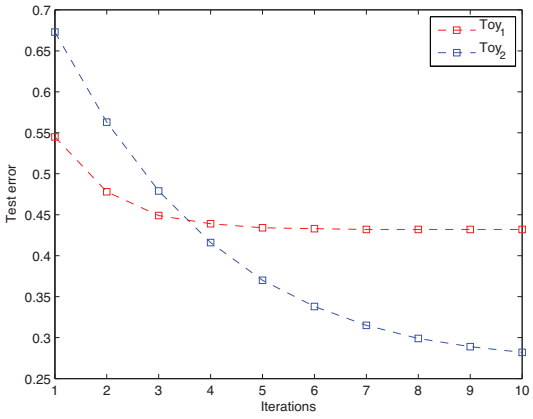
From Table 1 we can see that if we set (n_1, p_1) and (n_2, p_2) neither too large nor too small, the algorithm runs faster (sometimes twice as fast) while it at same time obtains better recovery accuracy.

Performance Comparison

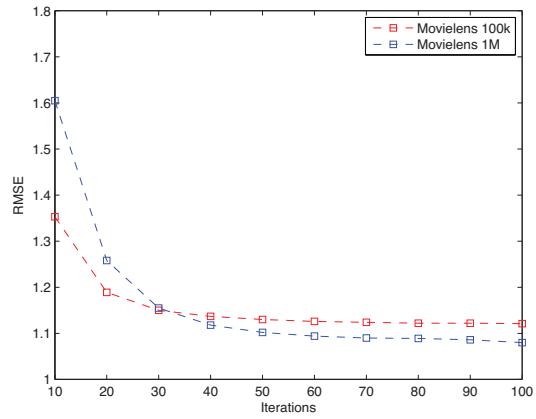
In this section, we conduct a set of experiments on five different datasets to compare the soft-impute (Mazumder, Hastie, and Tibshirani 2010) algorithm and the conventional SVT (Cai, Candès, and Shen 2010) algorithm. Especially for Movielens 1M dataset, we reconfigure the ratio between training and test set to prove the robustness of KP-SVT.

[1] Download from http://www.grouplens.org/node/73

[2] Download from http://eigentaste.berkeley.edu/dataset/



(a) On toy datasets



(b) On recommendation datasets

Figure 1: Convergence analysis of KP-SVT.

The size of the auxiliary matrices $\mathbf{A}$ and $\mathbf{B}$ is described in the left of Table 2. First, let's examine the recovery performance of these three algorithms. For toy datasets, SVT obtains a slightly better result while the performance of our KP-SVT is also acceptable. For real recommendation datasets, unfortunately, SVT does not converge. Note that here we use the MATLAB implementation of the conventional SVT doanloaded from the second author's webpage (Cai, Candès, and Shen 2010). Note also that KP-SVT yields better performance than the soft-impute algorithm on most datasets. Turning to the computational times, KP-SVT is generally 5 to 10 times faster than the competitors.

Concluding Remarks

In this paper we have proposed a fast spectral relaxation approach to the matrix completion problem by using Kronecker products. In particular, we have devised a Kronecker-product-based algorithm under the soft-impute framework (Mazumder, Hastie, and Tibshirani 2010). Our empirical studies have shown that KP-SVT can substantially

Table 1: The choice of the sizes of $\mathbf{A}$ and $\mathbf{B}$.

M 100k (943×1690)				M 1M (6040×4000)			
(n_1, p_1)	(n_2, p_2)	RMSE	Time	(n_1, p_1)	(n_2, p_2)	RMSE	Time
(23, 65)	(41, 26)	**1.128**	**13.8**	(151, 80)	(40, 50)	1.101	**169.9**
(41, 65)	(23, 26)	1.130	14.3	(151, 10)	(40, 400)	1.108	171.7
(41, 10)	(23, 169)	1.138	14.9	(151, 400)	(40, 10)	**1.100**	199.1
(23, 10)	(41, 169)	1.152	16.0	(604, 400)	(10, 10)	1.132	385.3
(41, 169)	(23, 10)	1.130	16.4	(10, 10)	(604, 400)	1.167	388.1

Table 2: Experimental results of the three algorithms correspond to different training sizes: $err-$ the test error; $time-$ the corresponding computational time (s), $NC-$ No convergence.

Datasets	Sizes		SVT		Soft-impute		KP-SVT	
	(n_1, p_1)	(n_2, p_2)	err	$time$	err	$time$	err	$time$
Toy_1	(40, 40)	(25, 25)	**0.375**	21.1	0.381	74.6	0.433	**4.7**
Toy_2	(100, 100)	(50, 50)	**0.206**	237.7	0.522	1051.5	0.281	**47.0**
M 100K	(23, 65)	(41, 26)	NC	NC	1.229	156.0	**1.128**	**13.8**
M 1M 70%	(151, 80)	(40, 50)	NC	NC	1.513	305.5	**1.114**	**169.4**
M 1M 80%	(151, 80)	(40, 50)	NC	NC	1.141	387.9	**1.101**	**169.9**
M 1M 90%	(151, 80)	(40, 50)	NC	NC	1.382	314.5	**1.096**	**171.8**
Jester	(1103, 15)	(58, 10)	NC	NC	**4.150**	271.0	5.094	**13.9**

reduce computational cost while maintaining high recovery accuracy. The approach can also be applied to speed up other matrix completion methods which are based on SVD, such as FPCA (Ma, Goldfarb, and Chen 2009).

Although we have illustrated especially two-matrix products in this paper, Kronecker products can be applied recursively when the size of the matrix in question is large. Another possible proposal is to consider the following optimization problem

$$\min_{\mathbf{A}_i, \mathbf{B}_i} \frac{1}{2}\|\mathcal{P}_\Omega(\mathbf{X}) - \mathcal{P}_\Omega(\mathbf{Y})\|_F^2 + \sum_{i=1}^{s} \gamma_{1i}\|\mathbf{A}_i\|_* + \sum_{i=1}^{s} \gamma_{2i}\|\mathbf{B}_i\|_*,$$

where the $\mathbf{A}_i$ and $\mathbf{B}_i$ have appropriate sizes and $\mathbf{Y} = \sum_{i=1}^{s} \mathbf{A}_i \otimes \mathbf{B}_i$. We will address matrix completion problems using this proposal in future work.

The Proofs of Lemma 1 and Theorem 1

Consider that $\mathbf{R}_i\mathbf{X}^T\mathbf{Q}_i^T$ is $p \times n$. We are always able to define an $(np/p_i) \times p_i$ matrix $\mathbf{X}_i$ such that

$$\mathsf{vec}(\mathbf{X}_i^T) = \mathsf{vec}(\mathbf{R}_i\mathbf{X}^T\mathbf{Q}_i^T).$$

Note that $\mathbf{Q}_i\mathbf{Q}_i^T = \mathbf{I}_n$, $\mathbf{R}_i\mathbf{R}_i = \mathbf{I}_p$ and $\mathbf{Q}_i(\mathbf{Y}_1 \otimes \cdots \otimes \mathbf{Y}_s)\mathbf{R}_i^T = \mathbf{Y}_{-i} \otimes \mathbf{Y}_i$. Let $\phi = \frac{1}{2}\|\mathbf{X} - \mathbf{Y}\|_F^2$ We have

$$\begin{aligned}
d\phi &= -\mathrm{tr}\big((\mathbf{X}-\mathbf{Y})(\mathbf{Y}_1^T \otimes \cdots \otimes d\mathbf{Y}_i^T \otimes \cdots \otimes \mathbf{Y}_s^T)\big) \\
&= -\mathrm{tr}\big((\mathbf{X}-\mathbf{Y})\mathbf{R}_i^T(\mathbf{Y}_{-i}^T \otimes \mathbf{Y}_i^T)\mathbf{Q}_i\big) \\
&= -\mathrm{tr}(\mathbf{Q}_i\mathbf{X}\mathbf{R}_i^T(\mathbf{Y}_{-i}^T \otimes d\mathbf{Y}_i^T)) \\
&\quad + \mathrm{tr}(\mathbf{Q}_i\mathbf{Y}\mathbf{R}_i^T(\mathbf{Y}_{-i}^T \otimes d\mathbf{Y}_i^T)) \\
&= -\mathrm{tr}(\mathbf{Q}_i\mathbf{X}\mathbf{R}_i^T(\mathbf{Y}_{-i}^T \otimes d\mathbf{Y}_i^T)) \\
&\quad + \mathrm{tr}((\mathbf{Y}_{-i}^T\mathbf{Y}_{-i}^T) \otimes (\mathbf{Y}_i d\mathbf{Y}_i^T))
\end{aligned}$$

Using some matric algebraic calculations, we further have

$$\begin{aligned}
d\phi &= -\mathsf{vec}(\mathbf{R}_i\mathbf{X}^T\mathbf{Q}_i^T)^T\mathsf{vec}(\mathbf{Y}_{-i}^T \otimes d\mathbf{Y}_i^T) \\
&\quad + (\prod_{j\neq i}\|\mathbf{Y}_j\|_F^2)\mathrm{tr}(\mathbf{Y}_i d\mathbf{Y}_i^T) \\
&= -\mathsf{vec}(\mathbf{X}_i^T)^T[(\mathbf{I}_{\frac{n}{n_i}} \otimes \mathbf{K}_{n_i,\frac{p}{p_i}})(\mathsf{vec}(\mathbf{Y}_{-i}^T) \otimes \mathbf{I}_{n_i}) \otimes \mathbf{I}_{p_i}]\mathsf{vec}(\mathbf{d}\mathbf{Y}_i^T) \\
&\quad + (\prod_{j\neq i}\|\mathbf{Y}_j\|_F^2)\mathrm{tr}(\mathbf{Y}_i d\mathbf{Y}_i^T) \\
&= -\mathrm{tr}\big[(\mathsf{vec}(\mathbf{Y}_{-i}^T)^T \otimes \mathbf{I}_{n_i})(\mathbf{I}_{\frac{n}{n_i}} \otimes \mathbf{K}_{n_i,\frac{p}{p_i}}^T)\mathbf{X}_i\mathbf{d}\mathbf{Y}_i^T\big] \\
&\quad + (\prod_{j\neq i}\|\mathbf{Y}_j\|_F^2)\mathrm{tr}(\mathbf{Y}_i d\mathbf{Y}_i^T) \\
&= -\mathrm{tr}(\mathbf{Z}_i\mathbf{d}\mathbf{Y}_i^T) + (\prod_{j\neq i}\|\mathbf{Y}_j\|_F^2)\mathrm{tr}(\mathbf{Y}_i d\mathbf{Y}_i^T).
\end{aligned}$$

Here we use the fact that if $\mathbf{A}$ and $\mathbf{B}$ are $m \times n$ and $p \times q$ matrices, then

$$\mathsf{vec}(\mathbf{A} \otimes \mathbf{B}) = [((\mathbf{I}_n \otimes \mathbf{K}_{qm})(\mathsf{vec}(\mathbf{A}) \otimes \mathbf{I}_q)) \otimes \mathbf{I}_p]\mathsf{vec}(\mathbf{B}).$$

Accordingly, we conclude that $\frac{\partial^2\phi}{\partial\mathsf{vec}(\mathbf{Y}^T)\partial\mathsf{vec}(\mathbf{Y}^T)} = (\prod_{j\neq i}\|\mathbf{Y}_j\|_F^2)\mathbf{I}_{n_1p_i}$. Thus, $h(\mathbf{Y}_i)$ is strictly convex in $\mathbf{Y}_i$.

We now obtain that $\hat{\mathbf{Y}}_i$ minimizes h if and only if $\mathbf{0}$ is a subgradient of the function h at the point $\hat{\mathbf{Y}}_i$; that is,

$$\mathbf{0} \in \hat{\mathbf{Y}}_i - \frac{1}{\prod_{j\neq i}\|\mathbf{Y}_j\|_F^2}\mathbf{Z}_i + \frac{\gamma_i}{\prod_{j\neq i}\|\mathbf{Y}_j\|_F^2}\partial\|\hat{\mathbf{Y}}_i\|_*,$$

where $\partial\|\hat{\mathbf{Y}}_i\|_*$ is the set of subgradients of the nuclear norm.

We now consider the special case that $s = 2$. In this case, we let $\mathbf{Y}_1 = \mathbf{A} = [a_{ij}]$ $(n_1 \times p_1)$ and $\mathbf{Y}_2 = \mathbf{B} = [b_{ij}]$ $(n_2 \times p_2)$. Moreover, we partition $\mathbf{X}$ into

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_{11} & \mathbf{X}_{12} & \cdots & \mathbf{X}_{1,p_1} \\ \mathbf{X}_{21} & \mathbf{X}_{22} & \cdots & \mathbf{X}_{2,p_1} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{X}_{n_1,1} & \mathbf{X}_{n_1,2} & \cdots & \mathbf{X}_{n_1,p_1} \end{bmatrix}, \quad \mathbf{X}_{ij} \in \mathbb{R}^{n_2 \times p_2}.$$

Using MATLAB colon notation, we can write $\mathbf{X}_{ij} = \mathbf{X}((i-1)n_2+1 : in_2, (j-1)p_2+1 : jp_2)$ for $i = 1,\ldots,n_1; j = 1,\ldots,p_1$. Thus,

$$\phi = \frac{1}{2}\|\mathbf{X} - \mathbf{A}\otimes\mathbf{B}\|_F^2 = \frac{1}{2}\sum_{i=1}^{n_1}\sum_{j=1}^{p_1}\|\mathbf{X}_{ij} - a_{ij}\mathbf{B}\|_F^2$$

$$= \frac{1}{2}\sum_{i=1}^{n_1}\sum_{j=1}^{p_1}\mathrm{tr}[(\mathbf{X}_{ij} - a_{ij}\mathbf{B})(\mathbf{X}_{ij} - a_{ij}\mathbf{B})^T].$$

We have

$$d\phi = -\sum_{i=1}^{n_1}\sum_{j=1}^{p_1}\mathrm{tr}[\mathbf{B}(\mathbf{X}_{ij} - a_{ij}\mathbf{B})^T]da_{ij}$$

$$-\sum_{i=1}^{n_1}\sum_{j=1}^{p_1}a_{ij}\mathrm{tr}[(\mathbf{X}_{ij} - a_{ij}\mathbf{B})d\mathbf{B}^T].$$

It then follows that

$$\frac{\partial\phi}{\partial\mathbf{A}} = \Big[\frac{\partial\phi}{\partial a_{ij}}\Big] = -\mathbf{Z}_1 + \mathrm{tr}(\mathbf{B}\mathbf{B}^T)\mathbf{A},$$

where the $(i,j)$the element of $\mathbf{Z}_1$ is $\mathrm{tr}(\mathbf{X}_{ij}^T\mathbf{B})$, i.e., $[\mathbf{Z}_1]_{ij} = \mathrm{tr}(\mathbf{X}_{ij}^T\mathbf{B})$, and that

$$\frac{\partial\phi}{\partial\mathbf{A}} = -\mathbf{Z}_2 + \mathrm{tr}(\mathbf{A}\mathbf{A}^T)\mathbf{B}$$

where $\mathbf{Z}_2 = \sum_{i=1}^{n_1}\sum_{j=1}^{p_1}a_{ij}\mathbf{X}_{ij}$.

References

Beck, A., and Teboulle, M. 2009. A fast iterative shrinkage-thresholding alghrithm for linear inverse problems. *SIAM Journal on Imaging Sciences* 183–202.

Cai, J.; Candès, E. J.; and Shen, Z. 2010. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization* 20:1956–1982.

Candès, E. J., and Recht, B. 2008. Exact matrix completion via convex optimization. *Found. of Comput. Math.* 9:717–772.

Candès, E. J., and Tao, T. 2009. The power of convex relaxation: Near-optimal matrix completion. *IEEE Trans. Inform. Theory (to appear)*.

Candès, E. J.; Li, X.; Ma, Y.; and Wright, J. 2009. Robust principal component analysis. *Microsoft Research Asia, Beijing, China*.

Jain, P.; Meka, R.; and Dhillon, I. 2010. Guaranteed rank minimization via singular value projection. In *Neural Information Processing Systems(NIPS)24*.

Keshavan, R.; Montanari, A.; and Oh, S. 2009. Matrix completion from a few entries. In *Proceedings of International Symposium on Information Theory (ISIT 2009)*.

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

Lin, Z.; Chen, M.; Wu, L.; and Ma, Y. 2010. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. Technical report, Electrical & Computer Engineering Department, University of Illinois at Urbana-Champaign, USA.

Lütkepohl, H. 1996. *Handbook of Matrices*. New York: John Wiley & Sons.

Ma, S.; Goldfarb, D.; and Chen, L. 2009. Fixed point and Bregman iterative methods for matrix rank minimization. *Mathematical Programming Series A*.

Magnus, J. R., and Neudecker, H. 1999. *Matrix Calculus with Applications in Statistics and Econometric*. New York: John Wiley & Sons, revised edition edition.

Mazumder, R.; Hastie, T.; and Tibshirani, R. 2010. Spectral regularization algorithms for learning large incomplete matrices. *Journal of machine learning research* 11(2):2287–2322.

Recht, B.; Fazel, M.; and Parrilo, P. A. 2007. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review* 52(3):471–501.

Srebro, N.; Alon, N.; and Jaakkola, T. 2005. Generalization error bounds for collaborative prediction with low-rank matrices. *Advances in Neural Information Processing Systems*.

Toh, K., and Yun, S. 2009. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems.

Van Loan, C. F., and Pitslanis, N. 1993. Approximation with kronecker products. In Moonen, M. S.; Golub, G. H.; and de Moor, B. L. R., eds., *Linear Algebra for Large Scale and Real-Time Applications*. Dordrecht: Kluwer Academic Publisher. 293–314.