Greedy Learning of Generalized Low-Rank Models

Quanming Yao  James T. Kwok
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Hong Kong
{qyaoaa, jamesk}@cse.ust.hk

Abstract

Learning of low-rank matrices is fundamental to many machine learning applications. A state-of-the-art algorithm is the rank-one matrix pursuit (R1MP). However, it can only be used in matrix completion problems with the square loss. In this paper, we develop a more flexible greedy algorithm for generalized low-rank models whose optimization objective can be smooth or nonsmooth, general convex or strongly convex. The proposed algorithm has low per-iteration time complexity and fast convergence rate. Experimental results show that it is much faster than the state-of-the-art, with comparable or even better prediction performance.

1 Introduction

In many machine learning problems, the data samples can be naturally represented as low-rank matrices. For example, in recommender systems, the ratings matrix is low-rank as users (and items) tend to form groups. The prediction of unknown ratings is then a low-rank matrix completion problem [Candès and Recht, 2009]. In social network analysis, the network can be represented by a matrix with entries representing similarities between node pairs. Unknown links are treated as missing values and predicted as in matrix completion [Chiang et al., 2014]. Low-rank matrix learning also have applications in image and video processing [Candès et al., 2011], multitask learning [Argyriou et al., 2006], multilabel learning [Tai and Lin, 2012], and robust matrix factorization [Eriksson and van den Hengel, 2012].

The low-rank matrix optimization problem is NP-hard [Recht et al., 2010], and direct minimization is difficult. To alleviate this problem, one common approach is to factorize the target matrix \(X\) as a product of two low-rank matrices \(U\) and \(V\), where \(U \in \mathbb{R}^{m \times r}\) and \(V \in \mathbb{R}^{n \times r}\) with \(r \ll \min\{m, n\}\). Gradient descent and alternating minimization are often used for optimization [Srebro et al., 2004; Eriksson and van den Hengel, 2012; Wen et al., 2012]. However, as the objective is not jointly convex in \(U\) and \(V\), this approach can suffer from slow convergence [Hsieh and Olsen, 2014].

Another approach is to replace the matrix rank by the nuclear norm (i.e., sum of its singular values). It is known that the nuclear norm is the tightest convex envelope of the matrix rank [Candès and Recht, 2009]. The resulting optimization problem is convex, and popular convex optimization solvers such as the proximal gradient algorithm [Beck and Teboulle, 2009] and Frank-Wolfe algorithm [Jaggi, 2013] can be used. However, though convergence properties can be guaranteed, singular value decomposition (SVD) is required in each iteration to generate the next iterate. This can be prohibitively expensive when the target matrix is large. Moreover, nuclear norm regularization often leads to biased estimation. Compared to factorization approaches, the obtained rank can be much higher and the prediction performance is inferior [Mazumder et al., 2010].

Recently, greedy algorithms have been explored for low-rank optimization [Shalev-shwartz et al., 2011a; Wang et al., 2014]. The idea is similar to orthogonal matching pursuit (OMP) [Pati et al., 1993] in sparse coding. For example, the state-of-the-art in matrix completion is the rank-one matrix pursuit (R1MP) algorithm [Wang et al., 2014]. In each iteration, it performs an efficient rank-one SVD on a sparse matrix, and then greedily adds a rank-one matrix to the matrix estimate. Unlike other algorithms which typically require a lot of iterations, it only takes \(r\) iterations to obtain a rank-\(r\) solution. Its prediction performance is also comparable or even better than others.

However, R1MP is only designed for matrix completion with the square loss. As recently discussed in [Udell et al., 2015], different loss functions may be required in different learning scenarios. For example, in link prediction, the presence or absence of a link is naturally represented by a binary variable, and the logistic loss is thus more appropriate. In robust matrix learning applications, the \(\ell_1\) loss or Huber loss can be used to reduce sensitivity to outliers [Candès et al., 2011]. While computationally R1MP can be used with these loss functions, its convergence analysis is tightly based on OMP (and thus the square loss), and cannot be easily extended.

This motivates us to develop more general greedy algorithms that can be used in a wider range of low-rank matrix learning scenarios. In particular, we consider low-rank matrix optimization problems of the form

\[
\min_X f(X) : \text{rank}(X) \leq r, \tag{1}
\]

where \(r\) is the target rank, and the objective \(f\) can be smooth
or nonsmooth, (general) convex or strongly convex. The proposed algorithm is an extension of R1MP, and can be reduced to R1MP when \( f \) is the square loss. In general, when \( f \) is convex and Lipschitz-smooth, convergence is guaranteed with a rate of \( O(1/T) \). When \( f \) is strongly convex, this is improved to a linear rate. When \( f \) is nonsmooth, we obtain a \( O(1/\sqrt{T}) \) rate for (general) convex objectives and \( O(\log(T)/T) \) for strongly convex objectives. Experiments on large-scale data sets demonstrate that the proposed algorithms are much faster than the state-of-the-art, while achieving comparable or even better prediction performance.

**Notation:** The transpose of vector / matrix is denoted by the superscript \(^T\). For matrix \( A = [A_{ij}] \in \mathbb{R}^{m \times n} \) (without loss of generality, we assume that \( m \leq n \)), its Frobenius norm is \( \| A \|_F = \sqrt{\sum_{i,j} A_{ij}^2} \), \( \ell_1 \) norm is \( \| X \|_1 = \sum_{i,j} | X_{ij} | \), nuclear norm is \( \| A \|_* = \sum_{i,j} \sigma_i (A) \), where \( \sigma_i (A) \)'s are the singular values, and \( \sigma_{\max} (A) \) is its largest singular value. For two vectors \( x, y \), the inner product \( \langle x, y \rangle = \sum_i x_i y_i \); whereas for two matrices \( A \) and \( B \), \( \langle A, B \rangle = \sum_{i,j} A_{ij} B_{ij} \). For a smooth function \( f \), \( \nabla f \) denotes its gradient. When \( f \) is convex but nonsmooth, \( g \in \{ u \mid f(y) \geq f(x) + \langle u, x - y \rangle \} \) is its subgradient at \( x \). Moreover, given \( \Omega \in \{ 0, 1 \}^{m \times n} \), \( [P_\Omega (A)]_{ij} = A_{ij} \) if \( \Omega_{ij} = 1 \), and 0 otherwise.

## 2 Review: Rank-One Matrix Pursuit

The rank-one matrix pursuit (R1MP) algorithm [Wang et al., 2014] is designed for matrix completion [Candes and Recht, 2009]. Given a partially observed \( m \times n \) matrix \( O = [O_{ij}] \), indices of the observed entries in the matrix \( \Omega \in \{ 0, 1 \}^{m \times n} \), where \( \Omega_{ij} = 1 \) if \( O_{ij} \) is observed and 0 otherwise. The goal is to find a low-rank matrix that is most similar to \( O \) at the observed entries. Mathematically, this is formulated as the following optimization problem:

\[
\min_X \sum_{(i,j) \in \Omega} (X_{ij} - O_{ij})^2 : \text{rank}(X) \leq r, \tag{2}
\]

where \( r \) is the target rank. Note that the square loss has to be used in R1MP.

The key observation is that if \( X \) has rank \( r \), it can be written as the sum of \( r \) rank-one matrices, i.e., \( X = \sum_{i=1}^r \theta_i u_i v_i^T \), where \( \theta_i \in \mathbb{R} \) and \( \| u_i \|_2 = \| v_i \|_2 = 1 \). To solve (2), R1MP starts with an empty estimate. At the \( t \)-th iteration, \( (u_t, v_t) \) pair that is most correlated with the current residual \( R_t = P_\Omega (O - X_{t-1}) \) is greedily added. It can be easily shown that this \( (u_t, v_t) \) pair are the leading left and right singular vectors of \( R_t \), and can be efficiently obtained from the rank-one SVD of \( R_t \). After adding this new \( u_t v_t^T \) basis matrix, all coefficients of the current basis can be updated as

\[
(\theta_1, \ldots, \theta_t) \leftarrow \arg \min_{\delta_1, \ldots, \delta_t} \left\| P_\Omega \left( \sum_{i=1}^t \theta_i u_i v_i^T - O \right) \right\|_F^2. \tag{3}
\]

Because of the use of the square loss, this is a simple least-squares regression problem with closed-form solution.

To save computation, R1MP also has an economic variant. This only updates the combination coefficients of the current estimate and the rank-one update matrix as:

\[
(\mu, \rho) \leftarrow \arg \min_{\mu, \rho} \left\| P_\Omega \left( \mu \sum_{i=1}^{t-1} \theta_i u_i v_i^T + \rho u_t v_t^T - O \right) \right\|_F^2. \tag{4}
\]

The whole procedure is shown in Algorithm 1.

### Algorithm 1 R1MP [Wang et al., 2014].

1. **Initialize:** \( X_0 = 0 \);
2. **for** \( t = 1, \ldots, T \) **do**
   3. \( R_t = P_\Omega (O - X_{t-1}) \);
   4. \((u_t, v_t) = \text{rank1SVD}(R_t)\);
   5. update coefficients using (3) (standard version), or (4) (economic version);
   6. \( X_t = \sum_{i=1}^t \theta_i u_i v_i^T \);
7. **end for**
8. **return** \( X_T \).

Note that each R1MP iteration is computationally inexpensive. Moreover, as the matrix’s rank is increased by one in each iteration, only \( r \) iterations are needed in order to obtain a rank-\( r \) solution. It can also be shown that the residual’s norm decreases at a linear rate, i.e., \( \| R_t \|_F^2 \leq \gamma^{t-1} \| P_\Omega (O) \|_F^2 \) for some \( \gamma \in (0, 1) \).

## 3 Low-Rank Matrix Learning with Smooth Objectives

Though R1MP is scalable, it can only be used for matrix completion with the square loss. In this Section, we extend R1MP to problems with more general, smooth objectives. Specifically, we only assume that the objective \( f \) is convex and \( L \)-Lipschitz smooth. This will be further extended to nonsmooth objectives in Section 4.

**Definition 1.** \( f \) is \( L \)-Lipschitz smooth if \( f(X) \leq f(Y) + \langle X - Y, \nabla f(Y) \rangle + \frac{L}{2} \| X - Y \|_F^2 \) for any \( X, Y \).  

### 3.1 Proposed Algorithm

Let the matrix iterate at the \( t \)-th iteration be \( X_{t-1} \). We follow the gradient direction \( \nabla f(X_{t-1}) \) of the objective \( f \), and find the rank-one matrix \( M \) that is most correlated with \( \nabla f(X_{t-1}) \):

\[
\max_M \langle M, \nabla f(X_{t-1}) \rangle : \text{rank}(M) = 1, \| M \|_F = 1. \tag{5}
\]

Similar to [Wang et al., 2014], its optimal solution is given by \( u_t v_t^T \), where \((u_t, v_t)\) are the leading left and right singular vectors of \( \nabla f(X_{t-1}) \). We then set the coefficient \( \theta_t \) for this new rank-one update matrix to \( -s_t/L \), where \( s_t \) is the singular value corresponding to \((u_t, v_t)\). Optionnally, all the coefficients \( \theta_1, \ldots, \theta_t \) can be refined as

\[
(\theta_1, \ldots, \theta_t) \leftarrow \arg \min_{\theta_1, \ldots, \theta_t} f \left( \sum_{i=1}^t \theta_i u_i v_i^T \right). \tag{6}
\]

As in R1MP, an economic variant is to update the coefficients as \( [\mu \theta_1, \ldots, \mu \theta_{t-1}, \rho] \), where \( \mu \) and \( \rho \) are obtained as

\[
(\mu, \rho) \leftarrow \arg \min_{\mu, \rho} f \left( \mu \sum_{i=1}^{t-1} \theta_i u_i v_i^T + \rho u_t v_t^T \right). \tag{7}
\]
Theorem 2. \( h \) is obtained.

Proposition 1. The proofs will be omitted. Because of the lack of space, all creasing in each iteration. Because of the lack of space, all square loss. In contrast, our analysis for Algorithm 2 here the analysis of R1MP is based on orthogonal matching 3.2 Convergence

Algorithm 2 GLRL for low-rank matrix learning with smooth convex objective \( f \).

1: Initialize: \( X_0 = 0 \);
2: for \( t = 1, \ldots, T \) do
3: \[ [u_t, s_t, v_t] = \text{rank1SVD}(\nabla f(X_{t-1})) \];
4: \[ X_t = X_{t-1} - \frac{s_t}{d_t} u_t v_t^T \];
5: (optional:) refine coefficients using (6) (standard version) or (7) (economic version);
6: end for
7: return \( X_T \).

Note that (6), (7) are smooth minimization problems (with \( r \) and 2 variables, respectively). As the target matrix is low-rank, \( r \) should be small and thus (6), (7) can be solved inexpenislly. In the experiments, we use the popular limited-memory BFGS (L-BFGS) solver [Nocedal and Wright, 2006]. Empirically, fewer than five L-BFGS iterations are needed. Preliminary experiments show that using more iterations does not improve performance.

Unlike R1MP, note that the coefficient refinement at step 5 is optional. Convergence results in Theorems 2 and 3 below still hold even when this step is not performed. However, as will be illustrated in Section 5.1, coefficient refinement is always beneficial in practice. It results in a larger reduction of the objective in each iteration, and thus a better rank-\( r \) model after running for \( r \) iterations.

3.2 Convergence

The analysis of R1MP is based on orthogonal matching pursuit [Pati et al., 1993]. This requires the condition \( \frac{1}{2} \| \nabla f(X) \|_F^2 = f(X) \), which only holds when \( f \) is the square loss. In contrast, our analysis for Algorithm 2 here is novel and can be used for any Lipschitz-smooth \( f \).

The following Proposition shows that the objective is decreasing in each iteration. Because of the lack of space, all the proofs will be omitted.

Proposition 1. If \( f \) is L-Lipschitz smooth,

\[
f(X_t) \leq f(X_{t-1}) - \frac{\gamma_{t-1}^2}{2L} \| \nabla f(X_{t-1}) \|_F^2,
\]

where

\[
\gamma_{t-1} = \frac{\sigma_{\text{max}}(\nabla f(X_{t-1}))}{\| \nabla f(X_{t-1}) \|_F} \in \left[ \frac{1}{\sqrt{m}}, 1 \right]. \tag{8}
\]

If \( f \) is strongly convex, a linear convergence rate can be obtained.

Definition 2. \( f \) is \( \mu \)-strongly convex if \( f(X) \geq f(Y) + \langle X-Y, \nabla f(Y) \rangle + \frac{\mu}{2} \| X-Y \|_F^2 \) for any \( X, Y \).

Theorem 2. Let \( X^\ast \) be the optimal solution of (1). If \( f \) is \( \mu \)-strongly convex and L-Lipschitz smooth, then

\[
f(X_T) - f(X^\ast) \leq \left( 1 - \frac{d_t^2 \mu}{L} \right)^T [f(X_0) - f(X^\ast)],
\]

where \( d_t = \min_{i=1}^T \gamma_t \).

If \( f \) is only (general) convex, the following shows that Algorithm 2 converges at a slower \( O(1/T) \) rate.

Theorem 3. If \( f \) is (general) convex and L-Lipschitz smooth, then

\[
f(X_T) - f(X^\ast) \leq \frac{2d_2^2 L [f(X_0) - f(X^\ast)]}{d_2^2 T [f(X_0) - f(X^\ast)] + 2d_2^2 L},
\]

where \( d_2 = \max_{t=1}^T \| X_t - X^\ast \|_F \).

The square loss in (2) is only general convex and 1-Lipschitz smooth. From Theorem 3, one would expect GLRL to only have a sublinear convergence rate of \( O(1/T) \) on matrix completion problems. However, our analysis can be refined in this special case. The following Theorem shows that a linear rate can indeed be obtained, which also agrees with Theorem 3.1 of [Wang et al., 2014].

Theorem 4. When \( f \) is the square loss, \( f(X_T) - f(X^\ast) \leq (1 - d_2^2)^T [f(X_0) - f(X^\ast)] \).

3.3 Per-Iteration Time Complexity

The per-iteration time complexity of Algorithm 2 is low. Here, we take the link prediction problem in Section 5.1 as an example. With \( f \) defined only on the observed entries of the link matrix, \( \nabla f \) is sparse, and computation of \( \nabla f(X_{t-1}) \) in step 3 takes \( O(\| \Omega \|_1) \) time. The rank-one SVD on \( \nabla f(X_{t-1}) \) can be obtained by the power method [Halko et al., 2011] in \( O(\| \Omega \|_1) \) time. Refining coefficients using (6) takes \( O(t \| \Omega \|_1) \) time for the \( t \)th iteration. Thus, the total per-iteration time complexity of GLRL is \( O(t \| \Omega \|_1) \). Similarly, the per-iteration time complexity of GEGLRL is \( O(t \| \Omega \|_1) \). In comparison, the state-of-the-art AIS-Impute algorithm [Yao and Kwok, 2015] (with a convergence rate of \( O(1/T^2) \)) takes \( O(r^2 \| \Omega \|_1) \) time in each iteration, whereas the alternating minimization approach in [Chiang et al., 2014] (whose convergence rate is unknown) takes \( O(r \| \Omega \|_1) \) time per iteration.

3.4 Discussion

To learn the generalized low-rank model, Udell et al. (2015) followed the common approach of factorizing the target matrix as a product of two low-rank matrices and then performing alternating minimization [Srebro et al., 2004; Eriksson and van den Hengel, 2012; Wen et al., 2012; Chiang et al., 2014; Yu et al., 2014]. However, this may not be very efficient, and is much slower than R1MP on matrix completion problems [Wang et al., 2014]. More empirical comparisons will be demonstrated in Section 5.1.

Similar to R1MP, the greedy efficient component optimization (GECO) [Shalev-shwartz et al., 2011a] is also based on greedy approximation but can be used with any smooth objective. However, GECO is even slower than R1MP [Wang et al., 2014]. Moreover, it does not have convergence guarantee.

4 Low-Rank Matrix Learning with Nonsmooth Objectives

Depending on the application, different (convex) nonsmooth loss functions may be used in generalized low-rank matrix
models [Udell et al., 2015]. For example, the $\ell_1$ loss is useful in robust matrix factorization [Candès et al., 2011], the scalding loss in quantile regression [Koenker, 2005], and the hinge loss in multilabel learning [Yu et al., 2014]. In this Section, we extend the GLRL algorithm, with simple modifications, to nonsmooth objectives.

4.1 Proposed Algorithm

As the objective is nonsmooth, one has to use the subgradient $g_t$ of $f(X_{t-1})$ at the $t$th iteration instead of the gradient in Section 3. Moreover, refining the coefficients as in (6) or (7) will now involve nonsmooth optimization, which is much harder. Hence, we do not optimize the coefficients. To ensure convergence, a sufficient reduction in the objective in each iteration is still required. To achieve this, instead of just adding a rank-one matrix, we add a rank-$k$ matrix (where $k$ may be greater than 1). This matrix should be most similar to $g_t$, which can be easily obtained as:

$$M^* = \arg \min_{M: \text{rank}(M) = k} \|M - g_t\|_F^2 = \sum_{i=1}^k s_i u_i v_i^T,$$

where $\{(u_1, v_1), \ldots, (u_k, v_k)\}$ are the $k$ leading left and right singular vectors of $g_t$, and $\{s_1, \ldots, s_k\}$ are corresponding singular values. The proposed procedure is shown in Algorithm 3. The stepsize in step 3 is given by:

$$\eta_t = \begin{cases} c_1/t & \text{if } f \text{ is } \mu\text{-strongly convex} \\ c_2/\sqrt{t} & \text{if } f \text{ is (general) convex} \end{cases},$$

where $c_1 \geq 1/\mu$ and $c_2 > 0$.

Algorithm 3 GLRL for low-rank matrix learning with nonsmooth objective $f$.

1: Initialize: $X_0 = 0$ and choose $\nu \in (0, 1)$;
2: for $t = 1, \ldots, T$ do
3: set $\nu_t$ as in (10);
4: compute subgradient $g_t$ of $f(X_{t-1})$, $h_t = 0$;
5: for $i = 1, \ldots, m$ do
6: $u_i, s_i, v_i = \text{rank1SVD}(g_t - h_t)$;
7: $h_t = h_t + s_i u_i v_i^T$;
8: if $\|g_t - h_t\|_F^2 \leq \nu \|g_{t-1} - h_{t-1}\|_F^2$ then break;
9: end if
10: end for
11: $X_t = X_{t-1} - \eta_t h_t$;
12: end for
13: return $X_T$.

4.2 Convergence

The following Theorem shows that when $f$ is nonsmooth and strongly convex, Algorithm 3 has a convergence rate of $O(\log T/T)$.

**Theorem 5.** Assume that $f$ is $\mu$-strongly convex, and $\|g_t\|_F \leq b_1$ for some $b_1$ ($t = 1, \ldots, T$), then

$$\min_{t=0,\ldots,T} f(X_t) - f(X_*) \leq \frac{1}{T} \left( 1 + \log T \right) \frac{c_1 b_1^2}{2} + b_2,$$

where $c_1$ is as defined in (10), and $b_2$ is a constant (depending on $X_0$, $\mu$, $\nu$ and $c_1$).

When $f$ is only (general) convex, the following Theorem shows that the rate is reduced to $O(1/\sqrt{T})$.

**Theorem 6.** Assume that $f$ is (general) convex, $\|g_t\|_F \leq b_1$ for some $b_1$ and $\|X_t - X_*\|_F \leq b_3$ for some $b_3$ ($t = 1, \ldots, T$), then

$$\min_{t=0,\ldots,T} f(X_t) - f(X_*) \leq \frac{c_2 (b_1^2 + b_3^2)}{2\sqrt{T}} - \frac{2b_1 b_3}{(1 - \sqrt{\nu})T},$$

where $c_2$ is as defined in (10).

For other convex nonsmooth optimization problems, the same $O(\log(T)/T)$ rate for strongly convex objectives and $O(1/\sqrt{T})$ rate for general convex objectives have also been observed [Shalev-Shwartz et al., 2011b]. However, their analysis is for different problems, and cannot be readily applied to our low-rank matrix learning problem here.

4.3 Per-Iteration Time Complexity

To study the per-iteration time complexity, we take the robust matrix factorization problem in Section 5.2 as an example. The main computations are on steps 4 and 6. In step 4, since the subgradient $g_t$ is sparse (nonzero only at the observed entries), computing $g_t$ takes $O(\|\Omega\|_1)$ time. At outer iteration $t$ and inner iteration $i$, $g_t$ in step 6 is sparse and has low rank (equal to $i - 1$). Thus, $g_t - h_t$ admits the so-called “sparse plus low-rank” structure [Mazumder et al., 2010; Yao et al., 2015]. This allows matrix-vector multiplications and subsequently rank-one SVD to be performed much more efficiently. Specifically, for any $v \in \mathbb{R}^n$, the multiplication $(g_t - h_t)v$ takes only $O(\|\Omega\|_1 + (i - 1)n)$ time (and similarly for the multiplication $u^T(g_t - h_t)$ with any $u \in \mathbb{R}^m$, and rank-one SVD using the power method takes $O(\|\Omega\|_1 + (i - 1)n)$ time. Assuming that $i_t$ inner iterations are run at (outer) iteration $t$, it takes a total of $O(i_t\|\Omega\|_1 + (i_t - 1)n^2)$ time. Typically, $i_t$ is small (empirically, usually 1 or 2).

In comparison, though the ADMM algorithm in [Lin et al., 2010] has a faster $O(1/T)$ convergence rate, it needs SVD and takes $O(m^2n)$ time in each iteration. As for the Wirig algorithm [Eriksson and van den Hengel, 2012], its convergence rate is unknown and a linear program with $n$ variables needs to be solved in each iteration. As will be seen in Section 5.2, this is much slower than GLRL.

5 Experiments

In this section, we compare the proposed algorithms with the state-of-the-art on link prediction and robust matrix factorization. Experiments are performed on a PC with Intel i7 CPU and 32GB RAM. All the codes are in Matlab.

5.1 Social Network Analysis

Given a graph with $m$ nodes and an incomplete adjacency matrix $O \in \{\pm 1\}^{m \times m}$, link prediction aims to recover a low-rank matrix $X \in \mathbb{R}^{m \times m}$ such that the signs of $X_{ij}$’s and $O_{ij}$’s agree on most of the observed entries. This can be
formulated as the following optimization problem [Chiang et al., 2014]:

$$\min_X \sum_{(i,j)\in \Omega} \log(1 + \exp(-X_{ij}O_{ij})) : \text{rank}(X) \leq r, \quad (11)$$

where $\Omega$ contains indices of the observed entries. Note that (11) uses the logistic loss, which is more appropriate as $O_{ij}$'s are binary.

The objective in (11) is convex and smooth. Hence, we compare the proposed GLRL (Algorithm 2 with coefficient update step (6)) and its economic variant EGLRL (using coefficient update step (7)) with the following:

1. AIS-Impute $^1$ [Yao and Kwok, 2015]: This is an accelerated proximal gradient algorithm with further speedup based on approximate SVD and the special "sparse plus low-rank" matrix structure in matrix completion;

2. Alternating minimization ("AltMin") [Chiang et al., 2014]: This factorizes $X$ as a product of two low-rank matrices, and then uses alternating gradient descent for optimization.

As a further baseline, we also compare with the GLRL variant that does not perform coefficient update. We do not compare with greedy efficient component optimization (GECO) [Shalev-shwartz et al., 2011a], matrix norm boosting [Zhang et al., 2012] and active subspace selection [Hsieh and Olsen, 2014], as they have been shown to be slower than AIS-Impute and AltMin [Yao and Kwok, 2015; Wang et al., 2014].

Experiments are performed on the Epinions and Slashdot data sets$^2$ [Chiang et al., 2014] (Table 1). Each row/column of the matrix $O$ corresponds to a user (users with fewer than two observations are removed). For Epinions, $O_{ij} = 1$ if user $i$ trusts user $j$, and $-1$ otherwise. Similarly for Slashdot, $O_{ij} = 1$ if user $i$ tags user $j$ as friend, and $-1$ otherwise.

Table 1: Signed network data sets used.

<table>
<thead>
<tr>
<th></th>
<th>#rows</th>
<th>#columns</th>
<th>#observations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Epinions</td>
<td>42,470</td>
<td>40,700</td>
<td>$7.5 \times 10^5$</td>
</tr>
<tr>
<td>Slashdot</td>
<td>30,670</td>
<td>39,196</td>
<td>$5.0 \times 10^5$</td>
</tr>
</tbody>
</table>

As in [Wang et al., 2014], we fix the number of power method iterations to 30. Following [Chiang et al., 2014], we use 10-fold cross-validation and fix the rank $r$ to 40. Note that AIS-Impute uses the nuclear norm regularizer and does not explicitly constrain the rank. We select its regularization parameter so that its output rank is 40. To obtain a rank-$r$ solution, GLRL is simply run for $r$ iterations. For AIS-Impute and AltMin, they are stopped when the relative change in the objective is smaller than $10^{-4}$. The output predictions are binarized by thresholding at zero. As in [Chiang et al., 2014], the sign prediction accuracy is used as performance measure.

Table 2 shows the sign prediction accuracy on the test set. All methods, except the GLRL variant that does not perform coefficient update, have comparable prediction performance. However, as shown in Figure 1, AltMin and AIS-Impute are much slower (as discussed in Section 3.3). EGLRL has the lowest per-iteration cost, and is also faster than GLRL.

Table 2: Testing sign prediction accuracy (%) on link prediction. The best and comparable results (according to the pairwise t-test with 95% confidence) are highlighted.

<table>
<thead>
<tr>
<th></th>
<th>Epinions</th>
<th>Slashdot</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIS-Impute</td>
<td>93.3±0.1</td>
<td>84.2±0.1</td>
</tr>
<tr>
<td>AltMin</td>
<td>93.5±0.1</td>
<td>84.9±0.1</td>
</tr>
<tr>
<td>GLRL w/o coef upd</td>
<td>92.4±0.1</td>
<td>82.6±0.3</td>
</tr>
<tr>
<td>GLRL</td>
<td>93.6±0.1</td>
<td>84.1±0.4</td>
</tr>
<tr>
<td>EGLRL</td>
<td>93.6±0.1</td>
<td>84.4±0.3</td>
</tr>
</tbody>
</table>

5.2 Robust Matrix Factorization

Instead of using the square loss, robust matrix factorization uses the $\ell_1$ loss to reduce sensitivities to outliers [Candès et al., 2011]. This can be formulated as the optimization problem [Lin et al., 2010]:

$$\min_X \sum_{(i,j)\in \Omega} |X_{ij} - O_{ij}| \text{ s.t. rank}(X) \leq r.$$  

Note that the objective is only general convex, and its sub-gradient is bounded ($\leq \|\Omega\|_1$). Since there is no smooth component in the objective, AIS-Impute and AltMin cannot be used. Instead, we compare GLRL in Algorithm 3 (with $\nu = 0.99$ and $c_2 = 0.05$) with the following:

1. Alternating direction method of multipliers (ADMM)$^3$ [Lin et al., 2010]: The rank constraint is replaced by

---

$^1$https://github.com/quanmingyao/AIS-impute

$^2$https://snap.stanford.edu/data/

$^3$http://perception.csl.illinois.edu/matrix-rank/Files/inexact alm rpca.zip

Figure 1: Training and testing accuracies (%) vs CPU time (seconds) on the Epinions (left) and Slashdot (right) data sets.
the nuclear norm regularizer, and ADMM [Boyd et al., 2011] is then used to solve the equivalent problem:
\[
\min_{X,Y} \sum_{(i,j) \in \Omega} |X_{ij} - O_{ij}| + \lambda \|Y\|_1, \text{ s.t. } Y = X.
\]

2. The Wiberg algorithm [Eriksson and van den Hengel, 2012]: It factorizes \( X \) into \( UV^T \) and optimizes them by linear programming. Here, we use the linear programming solver in Matlab.

Experiments are performed on the MovieLens data sets\(^4\) (Table 3), which have been commonly used for evaluating recommender systems [Wang et al., 2014]. They contain ratings \( \{1, 2, \ldots, 5\} \) assigned by various users on movies. The setup is the same as in [Wang et al., 2014]. 50% of the ratings are randomly sampled for training while the rest for testing. The ranks used for the 100K, 1M, 10M data sets are 10, 10, and 20, respectively. For performance evaluation, we use the mean absolute error on the unobserved entries \( \Omega^2 \):
\[
\text{MABS} = \frac{1}{|\Omega^2|} \sum_{(i,j) \in \Omega^2} |\hat{X}_{ij} - O_{ij}|,
\]

where \( \hat{X} \) is the predicted matrix [Eriksson and van den Hengel, 2012]. Experiments are repeated five times with random training/testing splits.

Table 3: MovieLens data sets used in the experiments.

<table>
<thead>
<tr>
<th>#users</th>
<th>#movies</th>
<th>#ratings</th>
</tr>
</thead>
<tbody>
<tr>
<td>100K</td>
<td>943</td>
<td>1,682</td>
</tr>
<tr>
<td>1M</td>
<td>6,040</td>
<td>3,449</td>
</tr>
<tr>
<td>10M</td>
<td>69,878</td>
<td>10,677</td>
</tr>
</tbody>
</table>

Table 4: Testing MABS on the MovieLens data sets. The best results (according to the pairwise t-test with 95% confidence) are highlighted. ADMM cannot converge in 1000 seconds on the 1M and 10M data sets, and thus is not shown.

<table>
<thead>
<tr>
<th></th>
<th>100K</th>
<th>1M</th>
<th>10M</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADMM</td>
<td>0.717(\pm)0.004</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>Wiberg</td>
<td>0.726(\pm)0.001</td>
<td>0.728(\pm)0.006</td>
<td>0.715(\pm)0.005</td>
</tr>
<tr>
<td>GLRL</td>
<td>0.724(\pm)0.004</td>
<td>0.694(\pm)0.001</td>
<td>0.683(\pm)0.001</td>
</tr>
</tbody>
</table>

6 Conclusion

In this paper, we propose an efficient greedy algorithm for the learning of generalized low-rank models. Our algorithm is based on the state-of-art R1MP algorithm, but allows the optimization objective to be smooth or nonsmooth, general convex or strongly convex. Convergence analysis shows that the proposed algorithm has fast convergence rates, and is compatible with those obtained on other (convex) smooth/nonsmooth optimization problems. Specifically, on smooth problems, it converges with a rate of \(O(1/T)\) on general convex problems and a linear rate on strongly convex problems. On nonsmooth problems, it converges with a rate of \(O(\log(T)/T)\) on general convex problems and \(O(\log(T)/T)\) rate on strongly convex problems. Experimental results on link prediction and robust matrix factorization show that the proposed algorithm achieves comparable or better prediction performance as the state-of-the-art, but is much faster.

Acknowledgments

This research was supported in part by the Research Grants Council of the Hong Kong Special Administrative Region.

\[^{4}\text{http://grouplens.org/datasets/movielens/}\]
References


