Efficient Robust Matrix Factorization with Nonconvex Penalties

Quanming Yao

Abstract—Robust matrix factorization (RMF) is a fundamental tool with lots of applications. The state-of-art is robust matrix factorization by majorization and minimization (RMF-MM) algorithm. It iteratively constructs and minimizes a novel surrogate function. Besides, it is also the only RMF algorithm with convergence guarantee. However, it can only deal with the convex \( \ell_1 \)-loss and does not utilize sparsity when matrix is sparsely observed. In this paper, we proposed robust matrix factorization by nonconvex penalties (RMF-NP) algorithm addressing these two problems. RMF-NP enables nonconvex penalties as the loss, which makes it more robust to outliers. As the surrogate function from RMF-MM no longer applies, we construct a new one and solve it in its dual. This makes the runtime and memory cost of RMF-NP only depends on nonzero elements. Convergence analysis based on the new surrogate function is also established, which shows RMF-NP is guaranteed to produce a critical point. Finally, experiments on both synthetic and real-world data sets demonstrate the superiority of RMF-NP over existing algorithms in terms of recovery performance and runtime.

Index Terms—Nonconvex regularization, Proximal algorithm, Compressed sensing, Matrix completion

I. INTRODUCTION

MATRIX factorization is a fundamental machine learning tool with diverse applications. In computer vision, it has been applied in, for example, structure from motion [11], and recovery of videos/images [2, 3]. In computer vision, it is popularly used for community detection in social networks [4] and recommender systems [5]. The square loss is usually used, leading to a smooth but nonconvex optimization problem. Many algorithms have been developed to solve this problem. They are scalable and have convergence guarantees [6], [7], [8].

The square loss assumes that the noise is Gaussian, and is sensitive to outliers. To alleviate this problem, the \( \ell_1 \)-loss has been used instead, and has achieved much better empirical performance [9]. However, the resulting robust matrix factorization (RMF) problem is neither convex nor smooth, and becomes much harder. Several algorithms have been developed to address this challenge, such as \( \ell_1 \)-Wib [9], CWM [10], ARG-D [11], Reg-\( \ell_1 \) [12] UNuBi [13] and RMF-MM [14], which is the state-of-the-art and the only algorithm with convergence guarantee. In each RMF-MM iteration, a convex and nonsmooth surrogate function is constructed and solved by the alternating direction method of multipliers (ADMM) [15] algorithm.

However, the \( \ell_1 \)-loss is not robust enough for outliers. Its weights on outliers are still the same as on normal observations. A similar problem occurs for the \( \ell_1 \)-regularizer in sparse learning and low-rank matrix learning. Recently, nonconvex regularizers have been introduced to alleviate this problem. Examples include the Geman penalty (GP) [16], log-sum penalty (LSP) [17] and Laplace penalty [18]. They penalize less than the \( \ell_1 \)-regularizer on the important features or singular values, and achieve much better empirical performance [19], [20], [21], [22]. However, they have not been studied with matrix factorization.

Another deficiency with RMF-MM is that it cannot handle sparse data. In many applications such as structure from motion [11] and recommender systems [5], the data observations are very sparse. For example, the MovieLens-10M user-movie matrix has only 1.4% observed entries. However, the ADMM algorithm underlying RMF-MM cannot utilize data sparsity, and its runtime and memory cost grow linearly with the matrix size.

To address the above two problems, we propose in this paper the robust matrix factorization using nonconvex penalties (RMF-NP) algorithm. First, instead of the commonly used \( \ell_1 \) and \( \ell_2 \) losses, we adapt the nonconvex penalties as the loss. This allows matrix factorization to be more robust to outliers. The surrogate function in RMF-MM can no longer be applied, and a new one is constructed. Second, to efficiently handle sparse data, we transform the surrogate to its dual, and solve with the accelerate proximal gradient (APG) algorithm [23], [24]. The time complexity of the proposed algorithm depends only on the number of nonzero elements in data matrices, making it much faster on large-scale problems. Besides, the theoretical analysis in RMF-MM cannot be used on nonconvex penalties. We develop new proof techniques based on the Clarke subdifferential and show that the objective

Fig. 1. Example nonconvex penalties.
value is decreasing, and a critical point can be produced by RMF-NP. Experiments on both synthetic and real-world data sets demonstrate the superiority of RMF-NP over existing algorithms in terms of recovery performance and runtime.

**Notation:** For a matrix $X \in \mathbb{R}^{m \times n}$, $\|X\|_F = (\sum_{i,j} X_{ij}^2)^{1/2}$ is its Frobenius norm; $\|X\|_1 = \sum_{i,j} |X_{ij}|$ is its $\ell_1$-norm; $\|X\|_\infty = \max_{i,j} |X_{ij}|$ is its $\ell_\infty$ norm. For a square matrix $X \in \mathbb{R}^{m \times m}$, $\text{tr}(X) = \sum_{i} X_{ii}$. For two matrices $X, Y \in \mathbb{R}^{m \times n}$, $\odot$ denotes element-wise product, i.e., $[X \odot Y]_{ij} = X_{ij} Y_{ij}$. For a vector $x \in \mathbb{R}^m$, $\text{Diag}(x)$ construct a diagonal matrix $X \in \mathbb{R}^{m \times m}$ with $X_{ii} = x_i$.

## II. REVIEW: RMF-MM ALGORITHM

Robust matrix factorization (RMF) tries to factorize the data matrix $M$ into $UV^\top$, where $U \in \mathbb{R}^{m \times r}$ and $V \in \mathbb{R}^{n \times r}$, via the following optimization problem:

$$
\min_{U, V} H(U, V) \equiv \|W \odot (M - UV^\top)\|_1 + \frac{\lambda}{2} (\|U\|_F^2 + \|V\|_F^2). \quad (1)
$$

Here, $\lambda \geq 0$ is a regularization parameter, and $W \in \{0, 1\}^{m \times n}$ indexes the missing entries in $M$ (with $M_{ij} = 1$ if $M_{ij}$ is observed, and 0 otherwise).

Very recently, ([14] ([14]) proposed the RMF-MM algorithm, which solves the RMF problem using majorization minimization (also called optimization transfer ([25])). Its core idea is to replace the objective in (1) with a surrogate function, which is easier to optimize. Specifically, given the current iterate $(U_k, V_k)$, they first define some auxiliary functions as follows.

**Proposition II.1.** ([24]) Let $U = U_k + \hat{U}$ and $V = V_k + \hat{V}$.

Define $H_k(U, V) \equiv \|W \odot (M - (U_k + \hat{U})(V_k + \hat{V})^\top)\|_1 + \frac{\lambda}{2} \|U_k + \hat{U}\|_F^2 + \frac{\lambda}{2} \|V_k + \hat{V}\|_F^2$, and $J_k(U, V) \equiv \|W \odot (M - U_k V_k - UV_k^\top - U_k V_k^\top)\|_1 + \frac{\lambda}{2} \|U_k + \hat{U}\|_F^2 + \frac{\lambda}{2} \|V_k + \hat{V}\|_F^2$. Then,

$$
H_k(U, V) \leq J_k(U, V) + \frac{1}{2} \|\hat{U}\|_F^2 + \frac{1}{2} \|\hat{V}\|_F^2,
$$

where

$$
\hat{\Lambda} = \text{Diag}(\sqrt{n_{\text{nz}}(W_{(i,:)}), \ldots, \sqrt{n_{\text{nz}}(W_{(:n)})}}),
$$

and $n_{\text{nz}}(W_{(i,:)})$ (resp. $n_{\text{nz}}(W_{(:n)})$) is the number of nonzero elements in the $i$th row (resp. $j$th column) of $W$. Equality holds iff $(\hat{U}, \hat{V}) = (0, 0)$.

In each iteration, RMF-MM then minimizes the following surrogate $F_k$:

$$
\min_{\hat{U}, \hat{V}} F_k(\hat{U}, \hat{V}) \equiv J_k(\hat{U}, \hat{V}) + \frac{1}{2} \|\hat{U}\|_F^2 + \frac{1}{2} \|\hat{V}\|_F^2, \quad (2)
$$

which is convex but not smooth. ([14] ([14]) transformed (2) as

$$
\min_{\hat{U}, \hat{V}, E} \|W \odot E\|_1 + \frac{\lambda}{2} \|U_k + \hat{U}\|_F^2 + \frac{\lambda}{2} \|V_k + \hat{V}\|_F^2 + \frac{1}{2} \|\hat{U}\|_F^2 + \frac{1}{2} \|\hat{V}\|_F^2,
$$

s.t. $E = M - U_k V_k^\top - UV_k^\top - U_k V_k^\top$.

This is then solved with a more efficient ADMM variant called LADMP2SAP ([26]).

The RMF-MM algorithm is summarized in Algorithm 1. It has fast empirical performance. Moreover, it is guaranteed to generate a critical point of (1). Assume that ADMM takes $I$ iterations to converge. The time complexity of Algorithm 1 is then $O(mnrIK)$, which is linear in the matrix size. Besides, its space complexity is $O(mn)$.

**Algorithm 1** Robust matrix factorization by majorization minimization (RMF-MM) ([14]).

- 1. Initialize $U_1 \in \mathbb{R}^{m \times r}$ and $V_1 \in \mathbb{R}^{n \times r}$
- 2. construct $\hat{\Lambda}$ and $\hat{\Lambda}$ as defined in Proposition II.1
- 3. for $k = 1, 2, \ldots, K$ do
- 4. compute $[U_{k+1}, V_{k+1}]$ from (3) using ADMM;
- 5. $U_{k+1} \leftarrow U_{k+1}$;
- 6. end for
- 7. return $U_{K+1}$ and $V_{K+1}$;

## III. PROPOSED ALGORITHM

Section III-A presents our RMF formulation. A new surrogate function is proposed in Section III-B which is then transformed to its dual and optimized in Section III-C. The complete algorithm is presented in Section III-D. Finally, Section III-E provides convergence analysis.

### A. Nonconvex Loss

In the following, we extend the optimization problem in (1) to non-binary weight and nonconvex loss:

$$
\min_{U, V} H(U, V) \equiv \sum_{i=1}^{m} \sum_{j=1}^{n} W_{ij} \phi(|M_{ij} - [UV^\top]_{ij}|) + \frac{\lambda}{2} \|U\|_F^2 + \|V\|_F^2, \quad (4)
$$

where $W \in \mathbb{R}^{m \times n}$ with $W_{ij} \geq 0$, and $\phi$ is the nonconvex penalty underlying the nonconvex regularizer. We make the following assumption on $\phi$, which is satisfied by the Geman penalty (GP) ([16]), log-sum-penalty (LSP) ([17]), and Laplace penalty ([13]) in Table 1.

**Assumption 1.** $\phi(\alpha)$ is a concave, smooth and strictly increasing function on $\alpha \geq 0$.

Obviously, the $\ell_1$ function also satisfies Assumption 1. Some nonconvex regularizers (such as the capped-$\ell_1$ penalty ([27]), minimax concave penalty (MCP) ([28]) and smooth-capped-absolute-deviation (SCAD) penalty ([29]) can be slightly modified so that Assumption 1 holds. Details can be found in Appendix A.

We also make the following assumption on the weights.

**Assumption 2.** $W$ has no zero rows or columns.

This is a standard assumption when missing values are present ([30]), and is also used in RMF-MM. When the $i$th row of $W$ is zero, the $i$th row of $U$ obtained from (3) must be...
zero because of the $\|U\|_F^2$ regularizer. Similarly, when the $i$th column of $W$ is zero, the corresponding column in $V$ is also zero. Thus, Assumption 2 is used to avoid trivial solutions.

However, problem (4) is difficult to solve. Algorithms for RMF [9], [12], [13], [14] cannot be used as they rely on the nonconvex penalties. However, they all use these penalties as regularization term and require the loss function be smooth. Thus, they all use these penalties as regularization term and require the loss function be smooth.

**B. Constructing the Surrogate**

Optimization transfer, which is a general technique to make difficult optimization problems easier [25]. The idea is to replace the original objective $h$, which is hard to optimize, by a surrogate $f$ which is easier. Given a current iterate $Y_k$, a good surrogate should have the following properties:

(a) $h(Y_k + Y) \leq f(Y)$;
(b) $0 = \arg\min_Y f(Y) - h(Y_k + Y)$;
(c) $f$ is easy to optimize (usually convex).

Next iterate is then obtained as $Y_{k+1} = Y_k + \arg\min_Y f(Y)$. RMF-MM is exactly based on optimization transfer. However, its surrogate as constructed in Proposition III.1 cannot be used on (4), because of the nonconvex $\phi$ and non-binary $W$. In this section, we construct a new surrogate which meets all the above requirements. Besides, while these requirements only ensure $h(Y_{k+1}) \leq h(Y_k)$, we will show in Section III.2 that the proposed surrogate ensures a stronger sufficient decrease condition.

First, we introduce the following Proposition, which helps to handle the nonconvexity of $\phi$. An illustration of the upper bound using the LSP penalty is shown in Figure 2.

**Proposition III.1.** For any $\alpha, \beta \in \mathbb{R}$, we have $\phi'(|\beta|) \leq \phi'(|\alpha|) (|\beta| - |\alpha|) + \phi(\alpha)$. Equality holds iff $\beta = \pm \alpha$.

Let $X_k = U_k V_k^\top$ and $[A_k]_{ij} = \phi'((X_k)_{ij})$. Using Proposition III.1 for any $(U_k, V_k)$, the $\sum_{i=1}^m \sum_{j=1}^n W_{ij} \phi([M_{ij} - [U_k V_k^\top]_{ij}])$ term in (4) can be upper-bounded as

$$b_k + \|\hat{W}_k \circ (M - (U_k + \hat{V})(V_k + \hat{V})^\top)\|_1,$$

where $b_k = \sum_{i=1}^m \sum_{j=1}^n W_{ij} \phi'((X_k)_{ij}) - \|X_k\|_1 [A_k]_{ij}$ and $\hat{W}_k = A_k \odot W$. However, (5) still involves a product between $\hat{U}$ and $\hat{V}$, making it nonconvex and no easier to optimize than the original problem (4). The next Lemma provides further relaxation.

**Lemma III.2.** Let $s_r(W, i) = \sum_{j=1}^n W_{ij}$ and $s_c(W, j) = \sum_{i=1}^m W_{ij}$. For the nonnegative $W$, we have

$$\|W \odot (UV^\top)\|_1 \leq \frac{1}{2}\|\hat{A} U\|_F + \frac{1}{2}\|\hat{A} V\|_F^2,$$

where $\hat{A} = \text{Diag}([s_r(W, 1)\hat{a}_1, \ldots, s_r(W, m)\hat{a}_n])$ and $\hat{A} = \text{Diag}([s_c(W, 1)\hat{a}_1, \ldots, s_c(W, n)\hat{a}_2])$. Equality holds iff $(U, V) = (0, 0)$.

Using Proposition III.1 and Lemma III.2 the following Proposition constructs a convex upper bound for (4).

**Proposition III.3.** Let $U = U_k + \hat{U}$ and $V = V_k + \hat{V}$. Define

$$\hat{H}_k(\hat{U}, \hat{V}) \equiv \sum_{i=1}^m \sum_{j=1}^n W_{ij} \phi([M_{ij} - [UV^\top]_{ij}])$$

$$+ \frac{\lambda}{2}\|U + \hat{U}\|_F^2 + \frac{\lambda}{2}\|V + \hat{V}\|_F^2,$$

$$\hat{J}_k(\hat{U}, \hat{V}) \equiv \|\hat{W}_k \circ (M - (U_k + \hat{V})(V_k + \hat{V})^\top)\|_1$$

$$+ \frac{\lambda}{2}\|U + \hat{U}\|_F^2 + \frac{\lambda}{2}\|V + \hat{V}\|_F^2 + b_k.$$

Then,

$$\hat{H}_k(\hat{U}, \hat{V}) \leq \hat{J}_k(\hat{U}, \hat{V}) + \frac{1}{2}\|\hat{A}_k U\|_F^2 + \frac{1}{2}\|\hat{A}_k V\|_F^2,$$

where $\hat{A}_k = \text{Diag}([s_r(W, 1)\hat{a}_1, \ldots, s_r(W, m)\hat{a}_n])$ and $\hat{A}_k = \text{Diag}([s_c(W, 1)\hat{a}_1, \ldots, s_c(W, n)\hat{a}_2])$. Equality holds iff $(\hat{U}, \hat{V}) = (0, 0)$.

We then minimize the following surrogate of (4):

$$\min_{\hat{U}, \hat{V}} \hat{F}_k(\hat{U}, \hat{V}) \equiv \hat{J}_k(\hat{U}, \hat{V}) + \frac{1}{2}\|\hat{A}_k U\|_F^2 + \frac{1}{2}\|\hat{A}_k V\|_F^2. \quad (6)$$

Note that (6) is convex but nonsmooth. Using Proposition III.3 it is easy to see that $\hat{F}_k$ meets all the three requirements of a good surrogate. In the special case where the $\ell_1$-loss and binary weights are used, Proposition III.3 reduces to Proposition II.1 and (6) reduces to (2). Unlike RMF-MM, the weights $\hat{W}_k, \hat{A}_k$ and $\hat{A}_k$ change from iteration to iteration.

**C. Optimizing the Surrogate**

As discussed in [14], the difficulty of solving (2) (and similarly (3)) is on the matrix products $UV^\top$ and $U_k V_k^\top$ inside the $\ell_1$-norm. ADMM, which has been used in RMF-MM, can also be used to solve (6). However, ADMM introduces an extra variable $E$ in (3), which is neither sparse nor low-rank. Thus, it fails to utilize possible sparsity of $W$, which may then lead to large time and space complexities. For example,
the MovieLens-10 data set in Section IV-B only has 1.4% observed elements. Another problem is that the convergence rate of ADMM is only $O(1/T)$ [26], where $T$ is the number of iterations.

Instead of using the approach in RMF-MM to solve (6), we first transform the surrogate in (5) to its dual as follows.

Proposition III.4. The dual problem of (6) is

$$
\min_X D_k(X) \equiv \frac{1}{2} \text{tr}(\hat{X} V_k - \lambda U_k)^\top \hat{A}_k (\hat{X} V_k - \lambda U_k) + \frac{1}{2} \text{tr}((\hat{X}^\top U_k - \lambda V_k) \tilde{A}_k (\hat{X}^\top U_k - \lambda V_k)) - \text{tr}(\hat{X}^\top M) \\
\text{s.t. } \|\hat{X}\|_\infty \leq 1, X_{ij} = 0 \text{ when } (\hat{W}_k)_{ij} = 0,
$$

(7) where $\hat{X} = \hat{W}_k \odot X$, $\hat{A}_k = (\lambda I + \tilde{A}_k^2)^{-1}$ and $\tilde{A}_k = (\lambda I + \hat{A}_k^2)^{-1}$. Moreover,

$$
\hat{U} = \hat{A}_k (\hat{X} V_k - \lambda U_k), \quad \hat{V} = \hat{A}_k (\hat{X}^\top U_k - \lambda V_k).
$$

(8)

From Proposition III.4 we have below remark, which shows $X$ is as sparse as $W_k$.

Remark III.1. The constraint (8) requires the nonzero positions of $X$ and $W_k$ be the same.

Problem (7) can be solved with the accelerated proximal gradient (APG) algorithm, which has a fast $O(1/T^2)$ convergence rate [23, 24]. In each iteration, one needs to compute the gradient $\nabla D_k$ and the proximal step. The following Proposition shows that the proximal step for (7) has a closed-form solution.

Proposition III.5. Let $Z^* = \arg \min_X \frac{1}{2}\|X - Z\|_F^2$ s.t. $\|\hat{W}_k \odot X\|_\infty \leq 1$, and $X_{ij} = 0$ when $(\hat{W}_k)_{ij} = 0$. Then,

$$
Z^*_{ij} = \begin{cases} 
\text{sign}(Z_{ij}) \min(|Z_{ij}|, 1/(\hat{W}_k)_{ij}) & (\hat{W}_k)_{ij} > 0 \\
0 & \text{otherwise}
\end{cases}
$$

Then, the computation of the gradient is

$$
\nabla D_k(X) = \hat{W}_k \odot [\hat{A}_k (\hat{X} V_k - \lambda U_k)V_k^\top] + \hat{W}_k \odot [U_k (\hat{X}^\top \hat{X} - \lambda V_k^\top) \tilde{A}_k] - \hat{W}_k \odot M.
$$

(10)

A direct implementation takes $O(mnr)$ time and $O(mnr)$ space. In the following, we reduce the computational complexity by exploiting sparsity. Consider the first term in (10). Let $Q_k = \hat{A}_k (\hat{X} V_k - \lambda U_k)$ and $S_k = \hat{W}_k \odot (Q_k V_k^\top)$. Note that $\hat{A}_k$ is a diagonal matrix, $Q_k$ can be computed by

$$
Q_k = \hat{A}_k \left( (\hat{W}_k \odot X) V_k \right) - \lambda \hat{A}_k U_k
$$

in $O(\text{nnz}(W)r + mnr)$ time where $r$ is the number of columns in $U_k$ and $V_k$. $S_k$ is a sparse matrix with the same nonzero positions as $W$. If $W_{ij} \neq 0$, $S_{ij}$ can be obtained from the inner product between the $i$th row of $Q_k$ and the $j$th row of $V_k$ [27, 28]. Thus, computing $S_k$ takes $O(\text{nnz}(W)r)$ time, and computing the whole the first term in (10) takes $O(\text{nnz}(W)r + mnr)$ time. Similarly, computing the second and last terms in (10) take $O(\text{nnz}(W)r + nr)$ and $O(\text{nnz}(W))$ time, respectively. In total, computing the gradient in (10) takes $O(\text{nnz}(W)r + (m + n)r)$ time. Using Proposition III.5 as we only need to consider the nonzero positions of $W$, this proximal step can be computed in $O(\text{nnz}(W))$ time. Hence, each APG iteration takes $O((m + n)r + \text{nnz}(W)r)$ time, which is much cheaper than the $O(mnr)$ time each ADMM iteration in RMF-MM takes.

As for space, one only needs to store the sparse matrix $S_k$ and $(U_k, V_k)$. Thus, APG takes $O((m + n)r + \text{nnz}(W))$ space. In contrast, ADMM in RMF-MM takes $O(mn)$ space as $E$ in (3) is dense. A summary is shown in Table III.

D. Complete Algorithm

The complete procedure is shown in Algorithm 2. The surrogate is optimized via its dual form in step 4. The primal solution $(U_k, V_k)$ is recovered in step 5, and $U_k, V_k$ updated in step 6. There are two main differences when compared with RMF-MM (Algorithm 1). First, as mentioned in Section III-B the weight matrix in the new surrogate is neither binary nor fixed. Instead, it changes with $k$, which results from Proposition III.3. Second, APG is used to optimize the dual of the surrogate, which can utilize data sparsity and make both iteration time complexity and memory cost much cheaper.

Algorithm 2 Robust matrix factorization using nonconvex penalties (RMF-NP) algorithm.

1: Initialize $U_1 \in \mathbb{R}^{m \times r}$ and $V_1 \in \mathbb{R}^{n \times r}$;
2: for $k = 1, 2, \ldots, K$ do
3: construct $\hat{A}_k$, $\hat{W}_k$, $\hat{A}_k$ and $\tilde{A}_k$ as defined in Proposition III.3;
4: compute $X_k^* = \arg \min_X D_k(X)$ in Proposition III.4 using APG;
5: compute $U_k$ and $V_k$ using (9);
6: $U_{k+1} = \left[ \begin{array}{c} U_k \\ V_k \end{array} \right] + \left[ \begin{array}{c} \hat{U}_k \\ \hat{V}_k \end{array} \right];$
7: return $U_{K+1}$ and $V_{K+1}$;
8: end for

E. Convergence Analysis

Here, we prove the convergence of the RMF-NP (Algorithm 2). As $\phi$ is not a convex function, analysis for RMF-MM fails here. Specifically, Proposition 1 in [14] cannot be extended to $H_k$ defined at Proposition III.3. Instead, our new proof is based on the new surrogate function and a close examination of Clarke subdifferential.

RMF-MM is guaranteed to achieve sufficient decrease (Definition III.1) in each iteration [14]. Here, Theorem III.6 below indicates the proposed RMF-NP (Algorithm 2) can also achieve this goal. It also shows the sequence $\{U_k, V_k\}$ generated from Algorithm 2 is bounded, which has at least one limit point.

Definition III.1. A function $f$ is said to have sufficient decrease on the sequence $\{X_k\}$ if there exists a constant $\alpha > 0$ such that $f(X_k) - f(X_{k+1}) \geq \alpha \|X_k - X_{k+1}\|_F^2, \forall k$.

Proposition III.6. For Algorithm 2 we have

(a) $\min_{k=1}^{K} \hat{A}_k \geq \alpha$ and $\min_{k=1}^{K} \bar{A}_k \geq \alpha$ for all $k$, where $\alpha$ is a positive constant;
(b). $\tilde{H}$ has sufficient decrease on the sequence $\{(U_k, V_k)\}$

$$
\tilde{H}(U_k, V_k) - \tilde{H}(U_{k+1}, V_{k+1}) \geq \frac{1}{2}\|\tilde{A}_k(U_{k+1} - U_k)\|_F^2 + \frac{1}{2}\|\tilde{A}_k(V_{k+1} - V_k)\|_F^2;
$$

(c). $\lim_{k \to \infty} (U_{k+1} - U_k) = 0$ and $\lim_{k \to \infty} (V_{k+1} - V_k) = 0$;

(d). The sequence $\{(U_k, V_k)\}$ is bounded.

However, to prove convergence to some critical points, the method based on the directional gradient used in [14] no longer applies. As $\phi$ is continuous but not convex, we make use of Clarke subdifferential (Definition III.2).

**Definition III.2.** [31] Let $f : \mathbb{R}^{m \times n} \to \mathbb{R}$ be a locally Lipschitz function. Its Clarke generalized directional derivative at $X$ in direction of $V$ is given by

$$
f^\circ(X, V) = \limsup_{Y \to X, \lambda \to 0} \frac{1}{\lambda} [f(Y + \lambda V) - f(Y)].
$$

Then, the Clarke subdifferential of $f$ at $X$ is given by

$$
\partial f^\circ(X) = \{ \xi : f^\circ(X, V) \geq \text{tr}(\xi^\top V), \forall V \in \mathbb{R}^{m \times n} \}.
$$

**Definition III.3.** [31] A point $X$ is a critical of $f$ if its satisfies

$$
0 \in \partial f^\circ(X).
$$

For our problem [4], using the above definitions, the critical points are given in the following Lemma III.7.

**Lemma III.7.** Let $C = M - UV^\top$, if $(U, V)$ satisfies

$$
0 \in (W \circ S)V + \lambda U \text{ and } 0 \in (W \circ S)^\top U + \lambda V,
$$

where $S_{ij} = \text{sign}(C_{ij}) \cdot \phi'(|C_{ij}|)$ if $C_{ij} \neq 0$, and $S_{ij} \in [-\phi'(0), \phi'(0)]$ otherwise. Then, it is a critical point of [4].

The following Proposition III.8 connects the subgradient of surrogate function $\tilde{F}_k$ to Clarke subdifferential of $\tilde{H}_k$.

**Proposition III.8.** (a). $\partial \tilde{F}_k(0, 0) = \partial \tilde{H}_k(0, 0)$; (b). If $0 \in \partial \tilde{H}_k(0, 0)$, then $(U_k, V_k)$ is a critical point of [4].

Finally, we are ready to show convergence result to critical points in Theorem III.9.

**Theorem III.9.** Let $\{(U_k, V_k)\}$ be the sequence generated by Algorithm [2] then any of its accumulating points is also a critical point of [4].

As [4] is neither convex nor smooth, convergence to critical points is the best result one can obtain [32]. Recall that RMF-MM is the only RMF algorithm with convergence guarantee to critical points. While ours problem [4] is more difficult than [1] (due to nonconvex loss), the proposed RMF-NP algorithm still share the same guarantee as RMF-MM. Besides, in Section IV-X we can see RMF-NP also empirically converges as fast as RMF-MM.

### IV. Experiments

In this section, we compare the proposed algorithms with the state-of-the-art on both synthetic and real-world data sets. Experiments are performed on a PC with Intel i7 CPU and 32GB RAM. All the codes are in Matlab, except for operations handling sparse matrices are written in C++.

#### A. Synthetic Data

The ground-truth is generated following $X = UV^\top$ where $U \in \mathbb{R}^{m \times 5}$ and $V \in \mathbb{R}^{n \times 5}$, and elements of $U$ and $V$ are sampled i.i.d from Gaussian distribution $\mathcal{N}(0, 1)$. The corrupted observed matrix is constructed as $M = X + S + N$, where $S$ is a sparse matrix of which 5% of its elements are nonzero and uniformly sampled from $\{\pm 5\}$, and $N$ is Gaussian noise sampled from $\mathcal{N}(0, 0.1)$. We randomly draw $\frac{15}{86}$ log($m$)% elements from $M$ as noisy observations. We use 50% of them for training and the rest for validation, the remained unobserved elements are used for testing. The matrix size is set as $m = 250, 500$ and $1000$.

For performance evaluation, two measurements are used: (i). normalized mean square error: $\text{NMSE} = \|\hat{W} \circ (X - \hat{UV}^\top)\|_F^2 / \|\hat{W} \circ X\|_F^2$ where testing elements are indicated by the binary matrix $\hat{W} \in \{0, 1\}^{m \times n}$, $\hat{U}$ and $\hat{V}$ are from algorithms; (ii). CPU time. Such measurements are also used in [14]. Following algorithms are compared

- RMF-MM [14]: Algorithm [1] which uses the convex $\ell_1$-loss;
- proposed Algorithm [2] Two variants are compared. (i). The new surrogate function [6] is solved with ADMM same as [14] (denoted as “RMF-NP(ADMM)”); and (ii). solved by APG in the dual problem [7] (denoted as “RMF-NP(APG)”).
- Specifically, the log-sum-penalty (LSP) [17] is used as loss function;
- Alternative gradient descent (AltGrad) [33] is used as the baseline. It is a gradient descent algorithm which targets at matrix factorization with square loss, i.e., $\min_{U, V} \frac{1}{2}\|\hat{W} \circ (M - \hat{UV}^\top)\|_F^2 + \frac{1}{2}\|U\|_F^2 + \frac{1}{2}\|V\|_F^2$.

All algorithms use $r = 5$ as it is the rank of the ground-truth $X$, other parameters are tuned using the validation set. RMF-MM and RMF-NP need to solve a surrogate in each iteration, we stop the inner loop when the relative change on the objective is smaller than $10^{-4}$.

1) **Recovery Performance:** The experiment is repeated five times. Recovery performance is at Table III. We can see that AltGrad, which is based on the square loss, cannot deal with large errors and gets much worse performance than RMF-MM and RMF-NP. RMF-NP achieves the lowest NMSE as LSP is more robust than the $\ell_1$-loss used by RMF-MM. On runtime,

| Algorithm | Iteration complexity | Space complexity |
|-----------|----------------------|-----------------|
| RMF-MM    | $O(1/T)$             | $O((m + n)r + \text{nnz}(W)r)$ |
| APG (proposed) | $O(1/T^2)$ | $O((m + n)r + \text{nnz}(W))$ |
TABLE III
Recovery performance on the synthetic dataset. NMSE is scaled by 10^{-2} and CPU time is in seconds. The best and comparable results according to pairwise 95% significant test are high-lightened.

|          | 250 (nnz: 11.04%) |          | 500 (nnz: 6.21%) |          | 1000 (nnz: 1.45%) |
|----------|-------------------|----------|------------------|----------|-------------------|
|          | NMSE              | CPU time | NMSE             | CPU time | NMSE             | CPU time |
| RMF-MM   | 9.2 ± 1.1         | 87.0 ± 13.0 | 8.5 ± 1.2         | 275.2 ± 57.8 | 7.7 ± 0.4         | 4135.0 ± 293.5 |
| RMF-NP   |                  |          |                  |          |                  |          |
| ADMM     | 5.0 ± 0.3         | 73.2 ± 29.7 | 4.5 ± 0.1         | 430.8 ± 121.6 | 4.0 ± 0.1         | 2147.7 ± 380.9 |
| APG      | 5.0 ± 0.3         | 14.0 ± 5.2 | 4.5 ± 0.1         | 13.1 ± 2.6  | 4.0 ± 0.1         | 18.9 ± 2.8   |
| AllGrad  | 47.1 ± 3.5        | 1.2 ± 0.3  | 40.7 ± 1.3        | 1.4 ± 0.2   | 35.7 ± 0.5        | 2.0 ± 0.4    |

TABLE IV
Recovery performance on the MovieLens data sets. The CPU time is in seconds. The best and comparable results according to pairwise 95% significant test are high-lightened. RMF-MM is too slow to run on the MovieLens-10M.

|          | 100K |          |          | 1M     |          |          | 10M     |          |
|----------|------|----------|----------|--------|----------|----------|---------|----------|
|          | MABS | CPU time | MABS     | CPU time | MABS     | CPU time | MABS    | CPU time |
| RMF-MM   | 0.707±0.012 | 1698.1±309.2 | 0.675±0.002 | 51164.9±6903.2 | 0.616±0.002 | 15517.0±844.5 |
| RMF-NP   | 0.669±0.004 | 14.7±6.6  | 0.630±0.001 | 286.2±76.2 | 0.616±0.002 | 15517.0±844.5 |
| APG      | 0.919±0.028 | 1.1±0.3   | 0.913±0.004 | 153.7±53.0 | 0.923±0.001 | 6314.9±1101.7 |

A detailed comparison on convergence of objective value is at Figure 3(b) We can see RMF-NP have comparable convergence speed as RMF-MM (Figure 3(a)). However, as weight matrix changes from iteration from iteration, more iterations are needed for the surrogate of RMF-NP(ADMM) than RMF-MM, which makes RMF-NP(ADMM) slightly slower (Figure 3(b)). Finally, when sparsity is utilized, RMF-NP(APG) is significantly faster than both RMF-NP(ADMM) and RMF-MM.

2) Solving the Surrogate Function: Finally, we compare the algorithms on solving the new surrogate function (6). There methods are considered: (i). ADMM solver by [14]; (ii). the proposed APG solver without using sparsity (denoted “APG(Dense)”; and with utilizing sparsity (denoted “APG(Sparse)”). The comparison for a typical run is in Figure 4. When measured by iterations (Figure 4(b)), APG(Dense) and APG(Sparse) have exactly the same convergence behavior which are all much faster than ADMM. This is explained by the faster $O(1/T^2)$ rate than $O(1/T)$ of ADMM. Then, on CPU time (Figure 4(b), APG(Sparse) is the fastest as it has both faster convergence and low iteration complexity.

B. Recommender Systems

In this section, we perform experiments on the MovieLens data sets (Table V), which contain ratings of different users on movies. We follow the setup in [34], and use 50% of the observed ratings for training, 25% for validation and the rest for testing. Besides, we randomly pick up 5% of ratings for training and corrupt them by adding ±5 with equal possibility.

Same as Section IV-A, we again consider RMF-MM, RMF-NP(APG) and AltGrad. We do not include RMF-NP(ADMM) as it solves the same problem as RMF-NP(APG) but is much slower. Note that previous algorithms for matrix factorization with $\ell_1$-loss, such as $\ell_1$-Wib [9], CWM [10], ARG-D [11], Reg-$\ell_1$ [12] and UNuBi [13] are not compared. As reported in [14], they are too slow (i.e., $\ell_1$-Wib), do not have convergence guarantee (i.e., Reg-$\ell_1$ and UNuBi), or inferior to RMF-MM

---

---

---

---

---
on recovery performance (i.e., CWM and ARG-D).

Let $\bar{U}$ and $\bar{V}$ be output from the algorithms. For performance evaluation, we also use the (i) mean absolute error on the test set $\bar{W}$: $MABS = \|\bar{W} \odot (\bar{U}\bar{V}^T - X)\|_1/\|\bar{W}\|_1$, where the testing positions are indicating by the binary matrix $\bar{W}$ and the ratings are given by $X$; (ii) CPU time. The rank $r$ is fixed the same for all algorithms, and it is 2 for MovieLens-100K, 5 for 1M and 9 for 10M data set, as suggested in [34]. The experiment is repeated five times and the performance obtained by different algorithms is compared in Table IV. On testing MABS, again, we can see the square loss cannot deal with outliers, thus AltGrad has the worst recovery performance. RMF-NP consistently achieves lower MABS than RMF-MM. This again results from the usage of nonconvex penalty (RMF-NP) algorithm. It addresses two problems of the state-of-the-art RMF-MM [14] algorithm.

Finally, in Figure 5, we show the impact of $\lambda$ on MovieLens-100K data set when RMF-MM and RMF-NP using the same rank. We can see RMF-NP consistently achieves lower MABS than RMF-MM. This again results from the usage of nonconvex penalty.

V. CONCLUSION

In this paper, we propose robust matrix factorization using nonconvex penalty (RMF-NP) algorithm. It addresses two problems of the state-of-the-art RMF-MM [14] algorithm. First, nonconvex penalties is used by our RMF-NP, which makes our algorithm more robust to outliers than RMF-MM. Besides, unlike RMF-MM, the proposed algorithm can utilize sparsity in data matrix, which makes it both fast and memory saving. Theoretical analysis is also provided, which shows RMF-NP is guaranteed to generate a decreasing sequence and produce a critical point. Experiments on both synthetic and real-world data sets are also performed, which verifies the superiority of the proposed algorithm over RMF-MM.

REFERENCES

[1] R. Basri, D. Jacobs, and I. Kemelmacher, “Photometric stereo with general, unknown lighting,” International Journal of Computer Vision, vol. 72, no. 3, pp. 239–257, 2007.
[2] H. Ji, C. Liu, Z. Shen, and Y. Xu, “Robust video denoising using low rank matrix completion,” in Computer Vision and Pattern Recognition, 2010, pp. 1791–1798.
[3] S. Gu, L. Zhang, W. Zuo, and X. Feng, “Weighted nuclear norm minimization with application to image denoising,” in Computer Vision and Pattern Recognition, 2014, pp. 2862–2869.
[4] J. Yang and J. Leskovec, “Overlapping community detection at scale: a nonnegative matrix factorization approach,” in International Conference on Web Search and Data Mining, 2013, pp. 587–596.
[5] Y. Koren, R. Bell, and C. Volinsky, “Matrix factorization techniques for recommender systems,” Computer, vol. 42, no. 8, 2009.
[6] A. Mnih and R. Salakhutdinov, “Probabilistic matrix factorization,” in Advances in Neural Information Processing Systems, 2008, pp. 1257–1264.
[7] R. Gemulla, E. Nijkamp, P. Haas, and Y. Sismanis, “Large-scale matrix factorization with distributed stochastic gradient descent,” in ACM SIGKDD Conference of Knowledge Discovery and Data Mining, 2011, pp. 69–77.
[8] H. Yu, C. Hsieh, S. Si, and I. Dhillon, “Scalable coordinate descent approaches to parallel matrix factorization for recommender systems,” in International Conference on Data Mining, 2012, pp. 765–774.
[9] A. Eriksson and A. Van Den Hengel, “Efficient computation of robust low-rank matrix approximations for large-scale problems using the $\ell_1$-norm,” in Computer Vision and Pattern Recognition, 2010, pp. 771–778.
[10] D. Meng, Z. Xu, L. Zhang, and J. Zhao, “A cyclic weighted median method for $\ell_1$ low-rank matrix factorization with missing entries,” in AAAI Conference on Artificial Intelligence, 2013, pp. 704–710.
[11] E. Kim, M. Lee, C. Choi, N. Kwak, and S. Oh, “Efficient $\ell_1$-norm-based low-rank matrix approximations for large-scale problems using alternating rectified gradient method,” IEEE Transactions on Neural Networks and Learning Systems, vol. 26, no. 2, pp. 237–251, 2015.
[12] Y. Zheng, G. Liu, S. Sugimoto, S. Yan, and M. Okutomi, “Practical low-rank matrix approximation under robust $\ell_2$-norm,” in Computer Vision and Pattern Recognition, 2012, pp. 1410–1417.
[13] R. Cabral, F. De la Torre, J. Costeira, and A. Bernardino, “Unifying nuclear norm and bilinear factorization approaches for low-rank matrix decomposition,” in International Conference on Computer Vision, 2013, pp. 2488–2495.
[14] Z. Lin, C. Xu, and H. Zha, “Robust matrix factorization by majorization minimization,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, no. 99, 2017.

[15] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Foundations and Trends in Machine Learning*, vol. 3, no. 1, pp. 1–122, 2011.

[16] D. Geman and C. Yang, “Nonlinear image recovery with half-quadratic regularization,” *IEEE Transactions on Image Processing*, vol. 4, no. 7, pp. 932–946, 1995.

[17] E. Candès, M. Wakin, and S. Boyd, “Enhancing sparsity by reweighted l1 minimization,” *Journal of Fourier Analysis and Applications*, vol. 14, no. 5-6, pp. 877–905, 2008.

[18] J. Trzasko and A. Manduca, “Highly undersampled magnetic resonance image reconstruction via homotopic-minimization,” *IEEE Transactions on Medical Imaging*, vol. 28, no. 1, pp. 106–121, 2009.

[19] P. Gong, C. Zhang, Z. Lu, J. Huang, and J. Ye, “A general iterative shrinkage and thresholding algorithm for non-convex regularized optimization problems,” in *International Conference on Machine Learning*, 2013, pp. 37–45.

[20] W. Zuo, D. Meng, L. Zhang, X. Feng, and D. Zhang, “A generalized iterated shrinkage algorithm for non-convex sparse coding,” in *International Conference on Computer Vision*, 2013, pp. 217–224.

[21] P. Ochs, A. Dosovitskiy, T. Brox, and T. Pock, “On iteratively reweighted algorithms for nonsmooth nonconvex optimization in computer vision,” *SIAM Journal on Imaging Sciences*, vol. 8, no. 1, pp. 331–372, 2015.

[22] Q. Yao and J. Kwok, “Efficient learning with a family of nonconvex regularizers by redistributing nonconvexity,” in *International Conference on Machine Learning*, 2016, pp. 2645–2654.

[23] M. Beck, A. and Teboulle, “A fast iterative shrinkage-thresholding algorithm for linear inverse problems,” *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[24] Y. Nesterov, “Gradient methods for minimizing composite functions,” *Mathematical Programming*, vol. 140, no. 1, pp. 125–161, 2013.

[25] K. Lange, R. Hunter, and I. Yang, “Optimization transfer using surrogate objective functions,” *Journal of Computational and Graphical Statistics*, vol. 9, no. 1, pp. 1–20, 2000.

[26] Z. Lin, R. Liu, and H. Li, “Linearized alternating direction method with parallel splitting and adaptive penalty for separable convex programs in machine learning,” *Machine Learning*, vol. 2, no. 99, pp. 287–325, 2015.

[27] T. Zhang, “Analysis of multi-stage convex relaxation for sparse regularization,” *Journal of Machine Learning Research*, vol. 11, pp. 1081–1107, 2010.

[28] C. Zhang, “Nearly unbiased variable selection under minimax concave penalty,” *The Annals of Statistics*, vol. 38, no. 2, pp. 894–942, 2010.

[29] J. Fan and R. Li, “Variable selection via nonconcave penalized likelihood and its oracle properties,” *Journal of the American Statistical Association*, vol. 96, no. 456, pp. 1348–1360, 2001.

[30] E. Candès and B. Recht, “Exact matrix completion via convex optimization,” *Foundations of Computational Mathematics*, vol. 9, no. 6, pp. 717–772, 2009.

[31] J. Dutta, “Generalized derivatives and nonsmooth optimization, a finite dimensional tour,” *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, vol. 13, no. 2, pp. 185–279, 2005.

[32] Y. Nesterov, * Introductory lectures on convex optimization: A basic course*. Springer, 1998.

[33] T. Zhao, Z. Wang, and H. Liu, “A nonconvex optimization framework for low rank matrix estimation,” in *Advances in Neural Information Processing Systems*, 2015, pp. 559–567.

[34] Q. Yao, J. T. Kwok, and W. Zhong, “Fast low-rank matrix learning with nonconvex regularization,” in *International Conference on Data Mining*, 2015, pp. 539–548.

[35] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[36] P. Gong and J. Ye, “HONOR: Hybrid Optimization for NOnc-convex regularized problems,” in *Advances in Neural Information Processing Systems*, 2015, pp. 415–423.
APPENDIX A
DISCUSSION ON OTHER PENALTIES

A. MCP
For the MCP [28] and SCAD [29] penalty, the problem is that \( \phi' \) will become zero one \( |\alpha| \) is larger than a threshold. Using the MCP penalty as an example, it is defined as
\[
\phi(|\alpha|) = \begin{cases} 
\gamma |\alpha| - \frac{\alpha^2}{2\theta} & |\alpha| \leq \gamma \theta \\
\frac{1}{2} \theta |\alpha|^2 & |\alpha| > \gamma \theta
\end{cases}
\]
To avoid such problem, we can modify MCP as
\[
\phi_M(|\alpha|) = \phi(|\alpha|) + \delta |\alpha|
\]
where \( \delta > 0 \) is a small positive constant. The new function \( \phi_M \) will not have zero derivation. The same can be done to the SCAD penalty.

B. SCAD
The SCAD [29] penalty can be modified using the same method as MCP penalty.

C. Capped-\( \ell_1 \)
Besides vanishing derivative, capped-\( \ell_1 \) penalty also has an extra nonsmooth point other than the zero point. As a result, Assumption [11] excluded such penalty. However, as discussed in [28], [17], [22], for those nonconvex penalties, the most important things are (i) concave on \( \mathbb{R}^+ \) and (ii) nonsmooth at zero point. Thus, the extra nonsmooth point inside the capped-\( \ell_1 \) penalty is not important, and we can use MCP or SCAD (with above modification) instead.

APPENDIX B
PROOFS

A. Proposition [11]
Proof. Note that \( \phi(x) \) is concave on \( x \geq 0 \), then for any \( y \geq 0 \), we have
\[
\phi(y) \leq \phi(x) + (y-x)\phi'(x).
\]
Let \( y = |\beta| \) and \( x = |\alpha| \), and then we obtain
\[
\phi(|\beta|) \leq \phi(|\alpha|) + (|\beta| - |\alpha|)\phi'(|\alpha|).
\]
Finally, as \( \phi \) is concave and strictly increasing on \( \mathbb{R}^+ \), the equality holds only when \( \beta = \pm \alpha \).

B. Lemma [11.2]
Proof. First, we have
\[
\|W \odot (UV^T)\|_1 = \|W \odot \begin{bmatrix} u_{11}^T \cdot \cdot \cdot u_{1m}^T \\ \cdot \cdot \cdot \\ u_{n1}^T \cdot \cdot \cdot u_{nm}^T \end{bmatrix}\|_1 = \sum_{i=1}^{m} \sum_{j=1}^{n} W_{ij} |u_{ij}^T v_{ij}|.
\]
where \( u_{ij} \) is \( ij \)th row in \( U \) (similar, for \( v_{ij} \) in \( V \)). Then, from Cauchy inequality, we have
\[
|u_{ij}^T v_{ij}| \leq \|u_{ij}\|_2 \|v_{ij}\|_2 \leq \frac{1}{2} \left( \|u_{ij}\|^2_2 + \|v_{ij}\|^2_2 \right).
\]
Together with (11), we have
\[
\|W \odot (UV^T)\|_1 \leq \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{n} W_{ij} \left( \|u_{ij}\|^2_2 + \|v_{ij}\|^2_2 \right)
\]
and the equality holds only when \( (U, V) = (0, 0) \).

C. Proposition [11.3]
Proof. From Cauchy inequality, we have
\[
\|W_k \odot (M - (U_k + \bar{U})(V_k + \bar{V})^T)\|_1 \leq \|W_k \odot (M - U_k V_k^T - \bar{U}V_k^T - U_k \bar{V}^T)\|_1 + \frac{1}{2} \left( \|\bar{U}^2 \|^2_F + \|\bar{V}^2 \|^2_F \right) + b_k.
\]
For the last term, using Lemma [11.2], we have
\[
\|W_k \odot (\bar{U} \bar{V}^T)\|_1 \leq \frac{1}{2} \left( \|\bar{U}^2 \|^2_F + \|\bar{V}^2 \|^2_F \right).
\]
Combine (3), (12) and (13), we have
\[
\sum_{i=1}^{m} \sum_{j=1}^{n} W_{ij} \phi \left( |M_{ij} - [UV^T]_{ij}\right) \leq \|W_k \odot \left( M - (U_k + \bar{U})(V_k + \bar{V})^T \right)\|_1 + \frac{1}{2} \left( \|\bar{U}^2 \|^2_F + \|\bar{V}^2 \|^2_F \right) + b_k.
\]
Finally, adding
\[
\frac{\lambda}{2} \|U_k + \bar{U}\|^2_F + \frac{\lambda}{2} \|V_k + \bar{V}\|^2_F
\]
to both side of (14), we then obtain the proposition.

D. Proposition [11.4]
Proof. Using the fact that \( \|X\|_1 = \max_{y \leq 1} \text{tr}(X^T Y) \) [35], and note that the \( \ell_1 \)-norm in \( J_k \) only need to count values at nonzero elements indicating by \( \hat{W}_k \). Then, (6) can be rewritten as
\[
\max_{\hat{X}, \hat{U}, \hat{V}} \mathcal{P}(X, \hat{U}, \hat{V}) \quad \text{s.t. } \|\hat{X}\|_{\infty} \leq 1 \text{ and } X_{ij} = 0 \text{ for } (W_k)_{ij} = 0.
\]
where
\[
\mathcal{P}(X, \hat{U}, \hat{V}) = \text{tr}(\hat{X}^T (M - \bar{U}V_k^T - U_k \bar{V}^T)) + \frac{\lambda}{2} \|U_k + \bar{U}\|^2_F + \frac{\lambda}{2} \|V_k + \bar{V}\|^2_F + \frac{1}{2} \|\bar{U}\|_F^2.
\]
As (15) is an unconstrained, smooth and convex problem on \( \hat{U} \), the optimal is obtained on zero gradient point, i.e., \( \nabla_{\hat{U}} \mathcal{P}(X, \hat{U}, \hat{V}) = 0 \). Then,
\[
\hat{U} = \hat{A}_k (\hat{X} V_k - \lambda U_k).
\]
Similarly, we can obtain
\[
\hat{V} = \hat{A}_k (\hat{X}^T U_k - \lambda V_k).
\]
Substituting (16) and (17) back into (15), we then obtain \( D_k(X) \) in the proposition. 

\[\Box\]
E. Lemma III.5
Proof.

Note that
\[\|X - Z\|_F^2 = \sum_{i=1}^{m} \sum_{j=1}^{n} (X_{ij} - Z_{ij})^2\]

Then, we only need to consider two cases
- \(W_{ij} = 0\): Obviously, \(Z_{ij} = 0\).
- \(W_{ij} \neq 0\): We have \(|X_{ij}| \leq 1/W_{ij}\), and
  \[Z_{ij} = \text{sign}(Z_{ij}) \min(|Z_{ij}|, 1/W_{ij}).\]

Combining above two cases, we then obtain the Lemma.

F. Proposition III.6
Proof. Conclusion (a). First note that,
\[
\inf_{U, V} H(U, V) \geq 0, \quad \lim_{\|U\|_F \to \infty} \lim_{\|V\|_F \to \infty} H(U, V) = \infty, \quad (18)
\]

Thus, the sequence \(\{U_k\}\) and \(\{V_k\}\) is bounded, and there exists a positive constant \(c\) such that
\[c_1 \geq \|(U_kV_k^T)_{ij}\|, \quad \forall i, j, k.\]

From Assumption 1 \(\phi\) is a strictly increasing function, thus \(\phi' > 0\). Then, there exist a positive constant \(c_2\) such that
\[\phi'(\|(U_kV_k^T)_{ij}\|) \geq c_2 \equiv \phi'(c_1).\]

From Assumption 2 each row and column in \(W\) has at least one nonzero element, then by the definition of \(\Lambda_k\) at Proposition III.3 its diagonal elements is given by
\[\left[\Lambda_k\right]_{ii} \geq \sqrt{\sum_{j=1}^{n} W_{ij}c_2}.\]

Same for \(\tilde{\Lambda}_k\). Thus, there exists a constant \(\alpha > 0\), such that all diagonal elements in \(\Lambda_k\) and \(\tilde{\Lambda}_k\) are not smaller than it.

Conclusion (b). As \((\tilde{U}_k, \tilde{V}_k)\) is the optimal solution to min \(\tilde{F}_k\), then
\[(0, 0) \in \partial \tilde{F}_k(\tilde{U}_k, \tilde{V}_k).\]

Recall the definition of \(\tilde{F}_k\) at (3), we have
\[(G\tilde{U}_k, G\tilde{V}_k) \in \partial \tilde{J}_k(\tilde{U}_k, \tilde{V}_k)\]

such that
\[(0, 0) = (G\tilde{U}_k, G\tilde{V}_k) + \left(\tilde{\Lambda}_k^2 \tilde{U}, \tilde{\Lambda}_k^2 \tilde{V}\right).\]

Then, multiplying \((\tilde{U}_k, \tilde{V}_k)\) on both side of (19), we have
\[0 = \text{tr}(G\tilde{U}_k^T \tilde{U} + G\tilde{V}_k^T \tilde{V} + \|\tilde{\Lambda}_k\|_F^2 + \|\tilde{\Lambda}_k^2 \tilde{V}\|_F^2).\]

As \(\tilde{J}_k\) is a convex function, by the definition of the subgradient, we have
\[\tilde{J}_k(0, 0) \geq \tilde{J}_k(\tilde{U}_k, \tilde{V}_k) - \text{tr}(G\tilde{U}_k^T \tilde{U} + G\tilde{V}_k^T \tilde{V}).\]

Combining (20) and (21), we obtain
\[\tilde{J}_k(0, 0) \geq \tilde{J}_k(\tilde{U}_k, \tilde{V}_k) + \|\tilde{\Lambda}_k\|_F^2 + \|\tilde{\Lambda}_k^2 \tilde{V}\|_F^2 \geq H_k(\tilde{U}_k, \tilde{V}_k) + \frac{1}{2}\|\tilde{\Lambda}_k\|_F^2 + \frac{1}{2}\|\tilde{\Lambda}_k^2 \tilde{V}\|_F^2.\]

Note that
\[\tilde{J}_k(0, 0) = H(U_k, V_k), \quad H_k(\tilde{U}_k, \tilde{V}_k) = H(U_{k+1}, V_{k+1}),\]

and using (22), we have
\[
H(U_k, V_k) - H(U_{k+1}, V_{k+1}) \geq \frac{1}{2}\|\tilde{\Lambda}_k\|_F^2 + \frac{1}{2}\|\tilde{\Lambda}_k^2 \tilde{V}\|_F^2. \quad (23)
\]

Together with conclusion (a) and Definition III.1 we obtain the first conclusion in Theorem III.4

Conclusion (c). Summing all inequalities in (23) from \(k = 1\) to \(K\), we have
\[
H(U_1, V_1) - H(U_{K+1}, V_{K+1}) \geq \sum_{k=1}^{K} \alpha \|U_k\|_F^2 + \frac{\alpha}{2}\|U_k\|_F^2.
\]

From (18), we must have
\[
\sum_{k=1}^{\infty} \|U_k\|_F^2 < \infty, \quad \sum_{k=1}^{\infty} \|V_k\|_F^2 < \infty.
\]

which indicates
\[
\lim_{k \to \infty} \|U_k\|_F^2 = \lim_{k \to \infty} \|V_k\|_F^2 = 0.
\]

Then, we have the Conclusion 2.

Conclusion (d). This conclusion results from (18).

G. Lemma III.7
Proof. For nonconvex penalties function satisfying Assumption 1 according to (26) their Clark subdifferential is given by
\[
\begin{align*}
\partial^\phi \phi(|\alpha|) &= \text{sign}(\alpha) \cdot \phi'(|\alpha|) & &\text{if } \alpha \neq 0, \\
\partial^\phi \phi(|\alpha|) &= \phi'((0), \phi'(0)) & &\text{otherwise}.
\end{align*}
\]

By Definition III.3 if \((U, V)\) is a critical point of (4), then it need to satisfies
\[(0, 0) \in \partial^\phi \hat{H}(U, V).\]

Combining (25) and (26), we then get the Lemma.

H. Proposition III.8
Proof. Conclusion (a). We prove this by checking Clark subdifferential of \(\hat{H}_k\) and subgradient of \(\tilde{F}_k\).

- Clark subdifferential of \(\hat{H}_k\): Let \(C^H = M - UV^T\). Be the definition of Clark differential we have
\[
\partial^\phi \hat{H}_k(U, V) = (W \circ S^H)(V_k + \hat{U}) + \lambda(U_k + \hat{U}), \quad (27)
\]
\[
\partial^\phi \hat{H}_k(U, V) = (W \circ S^H)^T(U_k + \hat{U}) + \lambda(V_k + \hat{V}). \quad (28)
\]

where \(S^H_{ij} = \text{sign}(C^H_{ij}) \cdot \phi'(|C^H_{ij}|)\) if \(C^H_{ij} \neq 0\), and \(S^H_{ij} \in [-\phi'(0), \phi'(0)]\) otherwise.
Subgradient of $\dot{F}_k$: Let $C^F = M - U_k V_k^T - \bar{U}_k V_k^T - U_k \bar{V}_k^T$. Then, for $\dot{F}_k$, we have
\begin{align*}
\partial_U \dot{F}_k(U, \bar{V}_k) &= (\dot{W}_k \odot S^F)(V_k + \bar{V}_k) + \lambda (U_k + \bar{U}_k) \\
&\quad + \Lambda_k^U \dot{U} \\
\partial_V \dot{F}_k(U, \bar{V}_k) &= (\dot{W}_k \odot S^F)^T (U_k + \bar{U}_k) + \lambda (V_k + \bar{V}_k) \\
&\quad + \Lambda_k^V \dot{V}
\end{align*}
where $S^F_{ij} = \text{sign}(C^F_{ij})$ if $C^F_{ij} \neq 0$, and $S^F_{ij} \in [-1, 1]$ otherwise.

Note that when $\bar{U} = 0$ and $\bar{V} = 0$, we have $C^H = C^F$. Then, by the definition of $\dot{W}_k = A_k \odot W$, we also have $W \odot S^H = \dot{W}_k \odot S^F$. Finally, the last term in (29) also vanishes to zero as $\bar{U} = 0$. Thus, (27) is exactly the same as (29). Similarly for (28) and (30), they are also the same. As a result, we have $\partial^o \dot{F}_k(0, 0) = \partial^o \dot{H}_k(0, 0)$.

Conclusion (b). From the definition of $\dot{H}$ at (4) and $\dot{H}_k$ in Proposition III.3, we have $\dot{H}_k(U_0, \bar{V}_k) = \dot{H}(U_k + \bar{U}_k, V_k + \bar{V}_k)$. Thus, if $(0, 0) \in \partial^o \dot{H}_k(0, 0)$, we then have $(0, 0) \in \partial^o \dot{H}(U_k, V_k)$, which shows $(U_k, V_k)$ is a critical point.

I. Theorem III.9

Proof. From Theorem III.6, we know there is at least one limit point for the sequence $\{(U_k, V_k)\}$. Let $\{(U_{k_j}, V_{k_j})\}$ be one of its subsequence, and
\begin{align*}
U_* &= \lim_{k_j \to \infty} U_{k_j}, \\
V_* &= \lim_{k_j \to \infty} V_{k_j}
\end{align*}
where $(U_*, V_*)$ is a limit point. Then, using Proposition III.8 we have
\begin{align*}
\lim_{k_j \to \infty} \partial^o \dot{F}_{k_j}(U_{k_j}, \bar{V}_{k_j}) &= \lim_{k_j \to \infty} \partial^o \dot{F}_{k_j}(0, 0) \\
&= \lim_{k_j \to \infty} \partial^o \dot{H}_{k_j}(0, 0) \\
&= \partial^o \dot{H}(U_*, V_*)
\end{align*}
Thus, $(0, 0) \in \partial^o \dot{H}(U_*, V_*)$, which shows $(U_*, V_*)$ is a critical point.