Nonsmooth Low-rank Matrix Recovery: Methodology, Theory and Algorithm

Wei Tu[1], Peng Liu[2], Yi Liu[3], Guodong Li[4], Bei Jiang[3], Linglong Kong*[3], Hengshuai Yao[5], and Shangling Jiu[6]

[1] Department of Public Health Sciences and Canadian Cancer Trials Group, Queen's University, Canada
[2] School of Mathematics, Statistics and Actuarial Science, University of Kent, United Kingdom
[3] Department of Mathematical and Statistical Sciences, University of Alberta, Canada
[4] Department of Statistics and Actuarial Science, University of Hong Kong, Hong Kong
[5] Huawei Hisilicon, Edmonton, Canada
[6] Huawei Hisilicon, Shanghai, China

Abstract. Many interesting problems in statistics and machine learning can be written as $\min_x F(x) = f(x) + g(x)$, where x is the model parameter, f is the loss and g is the regularizer. Examples include regularized regression in high-dimensional feature selection and low-rank matrix/tensor factorization. Sometimes the loss function and/or the regularizer is nonsmooth due to the nature of the problem, for example, $f(x)$ could be quantile loss to induce some robustness or to put more focus on different parts of the distribution other than the mean. In this paper we propose a general framework to deal with situations when you have nonsmooth loss or regularizer. Specifically we use low-rank matrix recovery as an example to demonstrate the main idea. The framework involves two main steps: the optimal smoothing of the loss function or regularizer and then a gradient based algorithm to solve the smoothed loss. The proposed smoothing pipeline is highly flexible, computationally efficient, easy to implement and well suited for problems with high-dimensional data. Strong theoretical convergence guarantee has also been established. In the numerical studies, we used L_1 loss as an example to illustrate the practicability of the proposed pipeline. Various state-of-the-art algorithms such as Adam, NAG and YellowFin all show promising results for the smoothed loss.

Keywords: Matrix factorization, Nonsmooth, Low-rank matrix, Nesterov's smoothing, Optimization

1 Introduction

Many problem in statistics and machine learning can be formulated as the following form:

$$\min_x F(x) = f(x) + g(x),$$

where x is the parameter, f is the loss and g is the regularizer. Examples includes penalized regression in high-dimensional feature selection [37] and low-rank matrix/tensor

* Corresponding author: lkong@ualberta.ca

recovery. Typically both $f(x)$ and $g(x)$ are proper convex functions such as using L_2 loss for $f(x)$. However, in some problems, due to the need of sparsity, robustness or other structural requirement of the parameter space, a nonsmooth or even nonconvex loss function or regularizer is often needed. In this paper we propose a general framework to deal with situations when you have nonsmooth loss or regularizer. Specifically we use low-rank matrix recovery as an example to illustrate the main idea.

In practice, many high dimensional matrices essentially have low-rank structure; see, for example, recommender systems [32], stochastic system realization [30] in systems and control, computer algebra [4], psychometrics [23] and even quantum state tomography [16] in physics. Meanwhile, these matrices usually are partially observed, and a lot of entries are left unobserved due to many different reasons. For example, we can only observe a few ratings from any particular recommender systems; or the quantum states have an exponentially large size so that it's not possible to obtain same scale observations. For these partially observed matrices with a high missing rate, it is of interest to ask "How to estimate the matrix with low-rank structure?" or "How to recover the low-rank matrix effectively?". This leads to an important problem of low-rank matrix recovery.

Since the Netflix prize competition in 2009, matrix factorization has been shown to outperform traditional nearest-neighbor based techniques in the sense that it allows the incorporation of additional information such as temporal effects, confidence levels and so on [22]. The basic idea of matrix factorization is to decompose the target matrix $M^* \in \mathbb{R}^{m\times n}$ into a bilinear form:

$$M^* = U^\top V$$

where $U \in \mathbb{R}^{r\times m}, V \in \mathbb{R}^{r\times n}$, and the rank of M^* is no more than r with $r \leq \min(m, n)$.

Due to the rapid improvement of computation power, matrix factorization has received more and more attention in various fields; see [20, 31, 6, 43] and among others. Most currently used methods are based on the L_2 loss, which is the optimal choice when the noise is Gaussian distributed. However, it is sensitive to outliers, and one possible solution is to consider a loss function other than the L_2 loss; see [5, 29, 20].

Meanwhile, as shown in [11], the nonsmooth optimization problem plays an important role in many areas such as image restoration, signal reconstruction, optimal control, and so on. In statistics, the least absolute deviation is well known to be robust to highly skewed and/or heavy-tailed data. The Manhattan distance in machine learning is actually based on L_1 loss. Quantile regression, which corresponds to the quantile loss, is another important estimating method in statistics, and is also commonly used to handle the highly skewed data. It is noteworthy to point out that both L_1 and quantile loss functions are nonsmooth. Other useful nonsmooth functions in statistics and machine learning include indicator function, step function, max function and so on [39, 40, 7].

Thus it is of importance to consider matrix factorization with nonsmooth loss function for the matrix recovery. Various algorithms, including the simplex, subgradient and quasi-monotone methods, have been proposed to tackle with the nonsmooth optimization, while few of them are efficient for recovering low-rank matrices with high dimensions. Smooth approximation recently has been studied for nonsmooth optimiza-

tion in many areas, such as complementarity problems, optimal control, eigenvalue optimization, etc., and it has been shown to be efficient even for the case with nonsmooth constraints; see, for example, [2], [10] and [12].

This paper considers the problem of low-rank matrix recovery from linear measurements. A general nonsmooth loss function is considered here, and Nesterov's smoothing method [26] is then applied to obtain an optimal smooth approximation. In practice, according to the specific nature of the problem and data, one can choose a suitable nonsmooth loss function, satisfying Nesterov's assumptions in [26], such that an efficient algorithm can be obtained. Due to the bilinear structure of matrix factorization, the alternating minimization method is thus employed to search for the solutions and, at each step, we compare the performance of various algorithms, which are based on gradient descent and momentum. Compared with previous work, this paper is more general in the following ways: 1) the transformation matrices we considered are more general; 2) a strong convergence guarantee is established for the proposed algorithm; 3) different state-of-the-art gradient based algorithms are used and compared. For example, vanilla gradient descent, Nesterov's momentum method [26], Adam [21] as well as YellowFin [42] algorithm. All the algorithms substantially improve the performance of original nonsmooth problem.

The rest of the paper is organized as follows. In section 2, we introduce the notation and formulate the problem mathematically. The proposed algorithms are presented in details in section 3, and the theoretical convergence analysis results can be found in section 4. In section 5, we illustrate the effectiveness of the proposed framework using the popularly used L_1 loss as a special example. Different gradient and momentum based algorithms are used to compare their performances. All the proofs for the theorems presented in the main paper can be found in the appendix.

2 Methodology Framework

This paper considers the model,

$$b_i = \langle A_i, M^* \rangle + \epsilon_i, \quad i = 1, \ldots, p, \tag{1}$$

where M^* is the true value, we can observe $\{A_i, b_i\}, i = 1, \cdots, p$, and ϵ_i is the error term. Here $A_i \in \mathbb{R}^{m\times n}$ with $1 \leq i \leq p$ are given transformation matrices. Suppose that the rank of matrix $M^* \in \mathbb{R}^{m\times n}$ is no more than r with $r \ll \min(m, n, p)$. We then have $M^* = U^{*\top} V^*$, where $U^* \in \mathbb{R}^{r\times m}$ and $V^* \in \mathbb{R}^{r\times n}$, and the following optimization can be used to recover M^*:

$$\min_{U,V} \frac{1}{p} \sum_{i=1}^{p} f(b_i - \langle A_i, U^\top V \rangle), \tag{2}$$

where $f(\cdot)$ is a nonsmooth objective function. Let $\mathcal{A} : \mathbb{R}^{m\times n} \to \mathbb{R}^p$ be an affine transformation with the ith entry of $\mathcal{A}(M^*)$ being $\langle A_i, M^* \rangle$, and $\boldsymbol{f}(x) = p^{-1} \sum_{i=1}^{p} f(x_i)$ for a vector $x = (x_1, ..., x_p)^\top$. We then can rewrite (2) into a compact form,

$$\min_{U,V} \boldsymbol{f}(b - \mathcal{A}(U^\top V)). \tag{3}$$

Assume that $\boldsymbol{f}(\cdot)$ is differentiable almost everywhere, and has the following structure:

$$\boldsymbol{f}(b-\mathcal{A}(U^\top V)) = \hat{\boldsymbol{f}}(b-\mathcal{A}(U^\top V)) + \max_u \left\{ \langle B(b-\mathcal{A}(U^\top V), u)_2 - \hat{\phi}(u) \rangle \right\}, \quad (4)$$

where $\hat{\boldsymbol{f}}$ is a continuous and convex function; see [26]. We then can obtain the following optimal smooth approximation:

$$\begin{aligned} \boldsymbol{f}_\mu(b-\mathcal{A}(U^\top V)) = \hat{\boldsymbol{f}}(b-\mathcal{A}(U^\top V)) + \\ \max_u \left\{ \langle B(b-(U^\top V)), u \rangle_2 - \hat{\phi}(u) - \mu d_2(u) \right\}, \end{aligned} \quad (5)$$

where μ is a positive smoothness parameter.

The above smooth approximation is made to function $\boldsymbol{f}(\cdot)$ with respect to the vector $b-\mathcal{A}(U^\top V)$, rather than with respect to unknown matrix parameters U and V, since it can be handled more conveniently. Moreover, under the restricted isometry property assumption on $\mathcal{A}$, our theoretical results show that this does not affect the convergence rate.

In meanwhile, due to the scale problem, the direct solutions to (5) may not have a proper structure, and one commonly used correction is to introduce a penalty of $\lambda(\|U\|_F^2 + \|V\|_F^2)$; see, for example, [35], [44] and among others. However, such penalization can not preserve the intrinsic structure for U and V. This paper use the Procrustes flow penalty [38, 24] instead, and our optimization problem becomes

$$\min_{U,V} \boldsymbol{f}_\mu(b-\mathcal{A}(U^\top V)) + \lambda \|UU^\top - VV^\top\|_F^2, \quad (6)$$

where the ad hoc choice of λ is $1/16$, the objective function is smooth with respect to U and V, and can be denoted by $\boldsymbol{f}_\mu^\lambda(U,V)$. The original nonsmooth optimization problem corresponds to

$$\min_{U,V} \boldsymbol{f}(b-\mathcal{A}(U^\top V)) + \lambda \|UU^\top - VV^\top\|_F^2, \quad (7)$$

where the objective function can be denoted by $\boldsymbol{f}^\lambda(U,V)$.

Due to the bilinear structure of $\boldsymbol{f}_\mu^\lambda(U,V)$ with respect to U and V at (6), we adopt the alternating minimization method to search for the solutions. Specifically, in each iteration, we will keep one of U and V fixed, optimize over the other, and then switch in the next iteration until it converges. In the literature of matrix factorization, [20] showed its convergence for L_2 loss function, and this paper will further establish the convergence for a general nonsmooth loss function.

3 Algorithm

There are three steps in our algorithm, and the first one is the initialization; see Algorithm 1. When applying the alternating minimization at the second step, if the initial values of U and V are orthogonal to the true ones, then our algorithm may never converge. To avoid this situation, we adopt the singular value projection (SVP) to provide

Algorithm 1: Initialization by SVP algorithm

Input: $\mathcal{A}$, b, tolerance ϵ, step size ξ_t for $t = 0, 1, \cdots$, $M^0 = 0_{m\times n}$
Output: M^{t+1}

Repeat
$Y^{t+1} \leftarrow M^t - \xi_t \nabla_M \boldsymbol{f}_\mu(b - \mathcal{A}(M^t))$
Compute top r singular vectors of Y^{t+1}:U_r, Σ_r, V_r
$M^{t+1} \leftarrow U_r \Sigma_r V_r$
$t \leftarrow t + 1$
Until $\|M^{t+1} - M^t\|_F \leq \epsilon$

some starting values of U and V for the alternating minimization, and the SVP was proposed by [19] and later used by [15], [1], [38] and so on.

Algorithm 1 can be written into

$$M^{t+1} \leftarrow \mathcal{P}_r \left(M^t - \xi_t \nabla_M \boldsymbol{f}_\mu(b - \mathcal{A}(M^t)) \right),$$

where $\mathcal{P}_r$ denotes a projection onto the space of rank-r matrices. Actually Algorithm 1 can be directly used to recover the matrix X^* if it is iterated for sufficient times; see [19]. However, the singular value calculation here is time-consuming when the dimension of matrix X^* is large. Furthermore, as an initialization, we do not need a very small tolerance ϵ, i.e., a rough output is sufficient.

The second step is the alternating minimization; see Algorithm 2. Specifically, in each iteration, we will keep one of U and V fixed, optimize over the other, and then switch in the next iteration until it converges. We then take the final values of U and V, denoted by $\hat{U}$ and $\hat{V}$, as solutions.

The third step is to update the values of U and V, respectively, at each iteration of the alternating minimization at step 1.1 and 1.2. There are various methods for it in the literature. [20] and [38] used the vanilla gradient descent method, while it may be slow in our nonsmooth matrix factorization settings. Algorithm 3 introduces Nesterov's momentum method to update the value of U, while that of V is fixed. Similarly we can gives the algorithm to update the value of V.

In Algorithm 3, $\boldsymbol{\nu}_{(i)}^t$ stands for the momentum term, γ is the momentum parameter, η is the learning rate parameter. And usually γ is chosen to be around 0.9 [36, 42]. For the sake of comparison, we also consider two other momentum-based algorithms: Adam in [21] and Yellowfin in [42].

Algorithm 2: Alternating Minimization

Input: U^0, V^0
Output: $U^{n_{max}}$, $V^{n_{max}}$

Repeat
1.1.Update U^t with $U^{t+1} = NAG(U^t, V^t)$
1.2.Update V^t with $V^{t+1} = NAG(U^{t+1}, V^t)$
Until convergence

Algorithm 3: Nesterov's accelerate gradient (NAG) method for U^{t+1}

Input: U^t, V^t
Output: U^{t+1}
Repeat
$\boldsymbol{\nu}_{(i)}^t = \gamma \boldsymbol{\nu}_{(i-1)}^t + \eta \nabla_U \boldsymbol{f}_\mu^\lambda (U_{(i-1)}^t - \gamma \boldsymbol{\nu}_{(i-1)}^t, V^t)$
$U_{(i)}^t = U_{(i-1)}^t + \boldsymbol{\nu}_{(i)}^t$
Until convergence

4 Convergence analysis

Denote $\{\hat{U}^\pi, \hat{V}^\pi\} = \min_{U,V} \boldsymbol{f}_\mu^\lambda(U, V)$ and $\{\hat{U}, \hat{V}\} = \min_{U,V} \boldsymbol{f}^\lambda(U, V)$. The following theorem guarantees that the optimal solution of the smoothed objective function converges to that of the nonsmooth one.

Theorem 1. *(Convergence of optimal solution of smoothed objective function) As $\pi \to 0^+$, we have $\hat{U}^{\pi\top}\hat{V}^\pi \to \hat{U}^\top \hat{V}$.*

Many existing literatures on smoothing technique usually only focus on analyzing the theoretical properties while ignore the relationship between smooth objective function and original nonsmooth objective function, for example, [3]. However, theorem 1 shows that if we only focus on optimizing smooth objective function, we can still obtain the optimal solution for nonsmooth objective function, the benefit is that we can thus have lots of choices with respect to the algorithms based on smooth objective function, then we can simple choose a fast one to obtain the solution.

Suppose that U and V are a pair of solutions, i.e. $M = U^\top V$. It then holds that, for an orthonormal matrix R satisfying $R^\top R = I_r$, $U^\dagger = RU$ and $V^\dagger = RV$ are another pair of solutions. To evaluate the performance of the proposed algorithm, we first define a distance between two matrices,

$$\text{dist}(U, U^\dagger) = \min_{R \in \mathbb{R}^{r\times r}: R^\top R = I_r} \|U - RU^\dagger\|_F,$$

where $U, U^\dagger \in \mathbb{R}^{r\times m}$ with $m \geq r$; see [38].

Theorem 2. *Let $M \in \mathbb{R}^{m\times n}$ be a rank r matrix, with singular values $\sigma_1(M) \geq \sigma_2(M) \geq \cdots \geq \sigma_r(M) > 0$ and condition number $\kappa = \sigma_1(M)/\sigma_r(M)$, let $M = A^\top \Sigma B$ be the SVD decomposition. Define $U = A^\top \Sigma^{1/2} \in \mathbb{R}^{m\times r}$, $V = B^\top \Sigma^{1/2} \in \mathbb{R}^{n\times r}$. Assume $\mathcal{A}$ satisfies a rank-6r RIP condition with RIP constant $\sigma_{6r} < \frac{1}{25}$, $\xi_t = \frac{1}{p}$. Then use $T_0 \geq 3\log(\sqrt{r}\kappa) + 5$ iterations in SVP initialization yields a solution U_0, V_0 obeying*

$$dist\left(\begin{bmatrix} U_0 \\ V_0 \end{bmatrix}, \begin{bmatrix} U \\ V \end{bmatrix}\right) \leq \frac{1}{4}\sigma_r(U). \tag{8}$$

Furthermore, starting from any initial solution obeying (11), the t-th iterate of algorithm 2 satisfies

$$dist\left(\begin{bmatrix} U_t \\ V_t \end{bmatrix}, \begin{bmatrix} U \\ V \end{bmatrix}\right) \leq \frac{1}{4}(1-\tilde{\tau}_1)^t \frac{\tilde{\mu}}{\tilde{\xi}} \frac{1+\delta_r}{1-\delta_r} \sigma_r(U) \tag{9}$$

under Nesterov's momentum method.

We make several contributions in Theorem 2, the first one is that we extend [19]'s SVP algorithm from least square matrix factorization to nonsmooth matrix factorization, in addition, we provide theoretical convergence guarantees for alternating minimization with Nesterov's momentum method for general objective function, this generalize [38]'s linear convergence guarantees for alternating minimization with gradient descent for least square objective function. We also would like to mention that though [41] also provide a smoothing approximation using Nesterov's smoothing technique, however, they are dealing with nonnegative matrix factorization case, which is much simpler than ours and they also did not provide rigorous theoretical analysis.

5 Numerical Studies

In the simulation study, we use L_1 loss as an example to illustrate the practical applicability of our theoretical results and provide insights about the choice of the algorithms. The corresponding smoothed approximation of L_1 loss is the popular Huber loss function (Huber 1981) in robust statistics. The Huber loss function is defined as

$$L_\mu(a) = \begin{cases} \frac{1}{2\mu}|a|^2 & \text{for} \quad |a| \leq \mu \\ |a| - \frac{\mu}{2} & \text{otherwise} \end{cases}, \tag{10}$$

where μ is the predetermined smoothness parameter, controlling the tradeoff between smoothness and precision. When $\mu \to 0^+$, the Huber function converge to absolute loss uniformly. On the other hand, when $\mu \to +\infty$, the Huber function resembles the L_2 loss. Figure 2 further illustrates the differences of these loss functions.

5.1 Synthetic data

The dataset is generated in the following manner: all entries in $U \in \mathbb{R}^{r\times m}$, $V \in \mathbb{R}^{r\times n}$ and $A_i \in \mathbb{R}^{m\times n}$ are independently sampled from Gaussian distribution $N(0,1)$. The ground truth M^* to recover is then calculated as $M^* = U^\top V$, and the rank of M^* is r. The observations b_i are generated following the assumed data generating process indicated in (1).

To evaluate the performance of the algorithms, the following two metrics are used: 1) the value of the loss function; 2) relative recovery error: $\|\hat{U}^\top \hat{V} - M\|_F / \|M\|_F$, where $\hat{U}$ and $\hat{V}$ are the estimated U and V using the proposed smoothing method. For both metrics, a smaller value is desired.

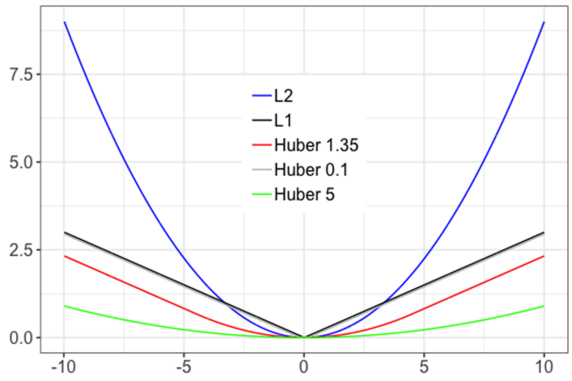


Fig. 1: The comparison of different loss functions. Here Huber 1.35 in the legend denotes Huber loss with $\mu = 1.35$.

Smooth parameter μ: The smoothing parameter μ in (10) controls the tradeoff between the smoothness and precision. In this simulation we aim to investigate the relationship between the choice of μ and the relative recovery error. Without loss of generality, $m = n = 32$, $p = 512$ and $r = 10$ is used here. To enhance the difference between L_1 loss and L_2 loss, all oberservations b_i have been contaminated by a Cauchy noise e_i, and Nesterov's momentum method is used for this simulation.

Figure 3 shows the trend between the smoothing parameter and the relative recovery error, and the x axis is the natural logrithm transformed μ. We observe three interesting patterns from this plot: 1) the performance of the algorithm is not sensitive to the choice of μ as we can see a reasonable small choice of μ will result in a small recovery error; 2) as μ approaches 0, the huber loss becomes very close to the L_1 loss, and recovery error increases slightly. This might due to the non-smoothness of L_1 loss; 3) as μ approaches $+\infty$, the huber loss becomes very close to the L_2 loss, and L_2 loss does not work well here due to the Cauchy noise added to the data. The empirical results here also further verifies our theoretical result in Theorem 1: as the smoothing paramter approaches to 0, the optimal solution of the smoothed objective function approaches to that of the nonsmooth one (L_1 loss here).

Algorithm comparisons: The third step of the algorithm of updating U or V can be solved by many different algorithms. Four methods are implemented here: the vanilla gradient descent method (GD), the Nesterov's momentum method (NAG) [25], the Adam [21] method and the YellowFin [42] algorithm that features step size auto-tuning capacity.

For each algorithm, we start with an unreasonably high step size, $\eta = 1$. Under such a high step size, most algorithms are expected to diverge or suffer from numerical instability. Then, we repetitively decrease the step size via multiplying η by $\frac{1}{\sqrt{10}}$ for each iteration until the algorithm starts to converge properly.

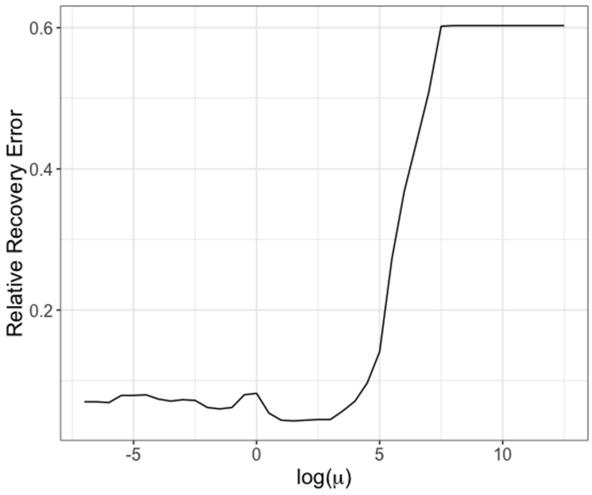


Fig. 2: Relative recovery error under different choices of smooth parameter μ. The horizontal axis is the μ in logrithm scale, and the vertical axis is the relative recovery error.

To limit our simulation in reasonable amount of time, m, n, r and p are chosen as 64, 64, 8 and 2048. Here we choose $\mu = 1.35$ as suggested by [18], and later on used by [27] and [13], among others. For each algorithm, the experiments are repeated 100 times. In each experiment, the step size tuning procedure yield incidental result for GD, NAG and ADAM. Step size are all decided as $10^{-2.5}$. The exception is the YellowFin algorithm, even extreme choices like 10 or 10^{-9} won't significantly alter the outcome of the optimization because of the fact that unlike other algorithms, the internal auto tuning process of YellowFin will override the preassigned step size before the end of first iteration.

One of the motivation to use nonsmooth objective function such as L_1 loss is the good robust performance of it. We have experimented different choices of error distributions for observations b_i. Due to the limited space here, we select two different scenerios have to present the findings: no contamination and adding a chi squared noise to the observations b_i. The findings of each scenerio has been summarized below:

Table 1: Number of iterations needed to reach a relative recovery error smaller than 20%, 10%, 5%, 1% for each algorithm under no contamination setting

	20%	10%	5%	1%
Adam	775	962	1105	1251
GD	3799	4788	> 5000	> 5000
NAG	709	881	1103	1148
YellowFin	1341	1509	1668	1764

No contamination: Figure 4 and Figure 5 show the behaviors of the loss and relative recovery error of 4 different algorithms when the observed b_i contains no error. We observe 4 interesting findings: 1) all algorithms converge eventually and the final relative recovery errors are all very close to 0; 2) as expected, the vanilla gradient descent converges the slowest in terms of both loss and relative recovery error; 3) Adam and Nestrov's momentum method have very similar behaviors, and Adam slightly outperforms Nestrov's momentum method at the later stage of optimization. Both of them are considerably more desirable than the vanilla gradient descent method and do not differ significantly in practical use. 4) The YellowFin algorithm have different behavior. It takes extra iterations for the algorithm to figure out optimal learning rate before the loss function starts to monotonously decrease.

Table 1 presents the number of iterations needed to reach a relative recovery error smaller than a certain threshold for each algorithms. We can see that all algorithms expect than gradient descent reaches 20% error in a relatively fast speed. Towards the end, from 5% to 1%, NAG has only needs less than 50 steps, while Adam takes aout 150 steps.

Table 2: Number of iterations needed to reach a relative recovery error smaller than 50%, 30%, 25%, 20% for each algorithm under chi-squared noise setting, and here NaN means the algorithm can not reach an error smaller than this value

	50%	30%	25%	20%
Adam	351	813	1108	NaN
GD	1289	4464	>5000	NaN
NAG	257	852	1229	NaN
YellowFin	444	1334	1606	NaN

Chi squared noise: Figure 6 and Figure 7 show the behaviors of the loss and relative recovery error of 4 different algorithms when each of the observed b_i is contaminated by a chi squared noise. Specifically, each b_i is replaced by $b_i + 10 * e_i$, where e_i follows a chi squared distribution with 3 degree of freedom. Compare to the no contamination setting, we have noticed that even though all algorithms converge eventually, the final loss and relative recovery error are not as close as to 0 as before. This is expected as the Cauchy error brought unnegligible noise to the observations. For the comparisons between different algorithms, the patterns are similar as the no contamination setting.

Table 2 presents the number of iterations needed to reach a relative recovery error smaller than a certain threshold for each algorithms. We can see that no algorithm can reach an error smaller than 20%, which shows that contaminated observations can bring serious trouble in the recovery of the original matrix. Adam and NAG have similar performances here, while Adam needs slightly smaller iterations to reach 25% error.

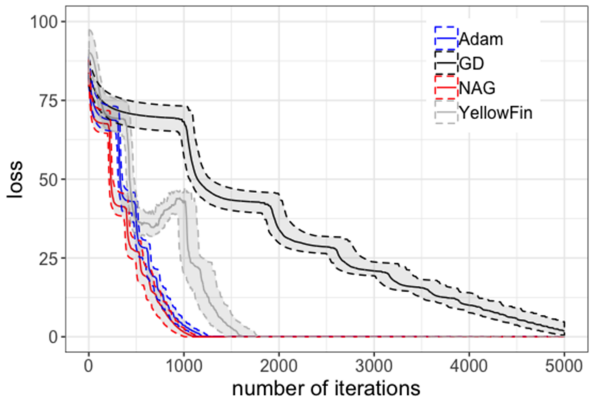


Fig. 3: Loss function curves under no contamination setting. The horizontal axis is training steps, and the vertical axis is the value of loss function. Each curve represents median over 100 runs, and the area between 0.25 and 0.75 quantile are plotted as shadow.

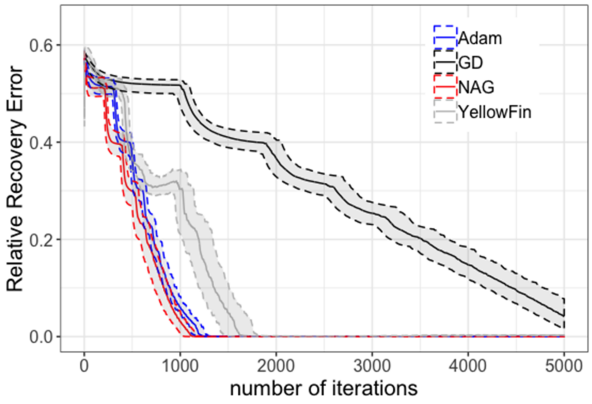


Fig. 4: Relative matrix recovery error curves under no contamination setting. The horizontal axis is training steps, and the vertical axis is the recovery error. Each curve represents median over 100 runs, and the area between 0.25 and 0.75 quantile are plotted as shadow.

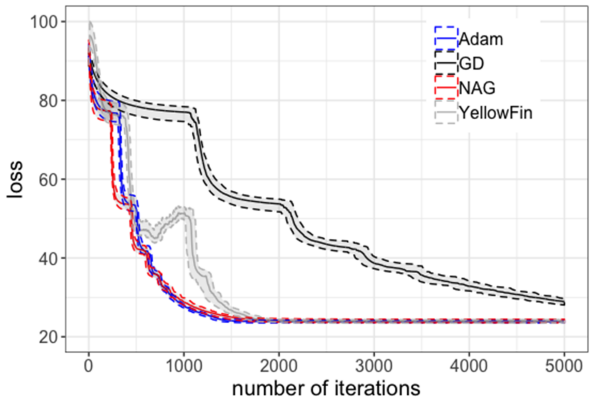


Fig. 5: Loss function curves under Cauchy noise setting. The horizontal axis is training steps, and the vertical axis is the value of loss function. Each curve represents median over 100 runs, and the area between 0.25 and 0.75 quantile are plotted as shadow.

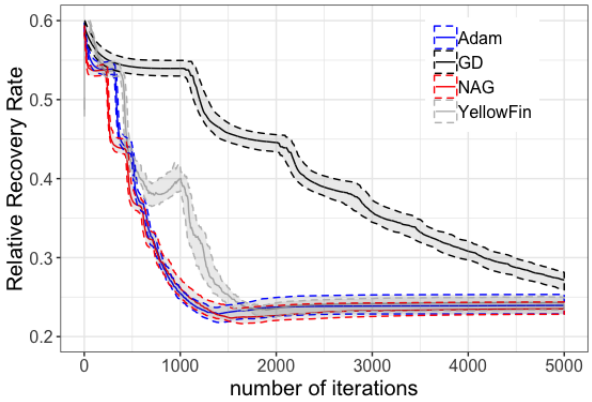


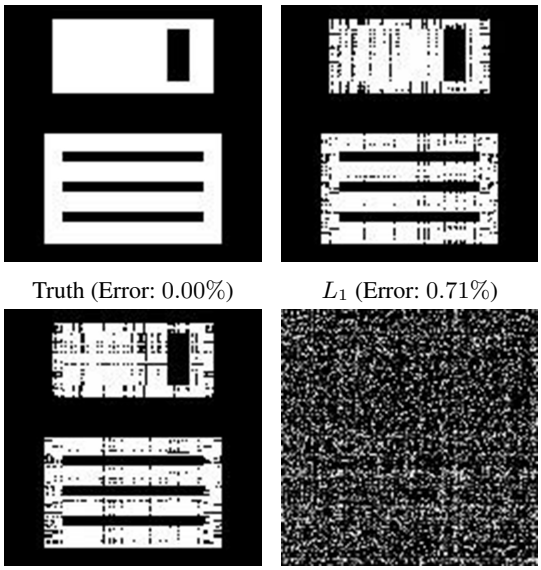
Fig. 6: Relative matrix recovery error curves under Cauchy noise setting. The horizontal axis is training steps, and the vertical axis is the recovery error. Each curve represents median over 100 runs, and the area between 0.25 and 0.75 quantile are plotted as shadow.

5.2 Real world data experiment

In this section, we demonstrate the efficiency of our method via a real world example. Not all data are normally distributed as is in our synthesized data set, furthermore, noise is ubiquitously unavoidable in real world practices.

The real world data we use in this experiment is an old-school gray-scale saving icon with dimension of $m = n = 128$ and the rank of this picture is $r = 6$. 8000 normal distributed A_i are generated in the setting of matrix sensing and b_i are calculated accordingly. To show the robustness trait of L_1 loss is well preserved by our smoothing method, a noise with independent *Cauchy* distribution is additionally applied to all b_i.

The results are shown in Figure 8. L_2 loss can not recover the image and it turns out totally blurred. Both L_1 and Huber case can recover recognizable picture benefit from our loss. However, L_1 optimization based on nonsmooth subgradient method takes over 20x more time to reach converge than the Huber design and the ratio will increase even more as the scale of the problem exaggerates.



Truth (Error: 0.00%) L_1 (Error: 0.71%)

Huber (Error: 1.01%) L_2 (Error: $> 100\%$)

Fig. 8: Final recovery result for different loss function

6 Conclusion

This paper considers the matrix factorization for low-rank matrix recovery, and a general nonsmooth loss function is assumed. It includes the commonly used L_1 and quantile loss functions as special cases, and this gives us much flexibility by choosing a suitable form according to our knowledge and observations.

In the proposed algorithm, we first suggest an optimal smooth approximation of the nonsmooth objective function [26], then a lot of algorithms based on gradient can be applied to the problem, we use the vanilla gradient descent, Nesterov's momentum algorithm, adam as well as yellowfin as examples and compare their performance. Though smoothing changed the problem's structure, however, the benefit is that it brings us much more flexibility to choose different algorithms.

Bibliography

[1] D. Achlioptas and F. McSherry. Fast computation of low-rank matrix approximations. *Journal of the ACM (JACM)*, 54(2):9, 2007.

[2] G. Alefeld and X. Chen. A regularized projection method for complementarity problems with non-lipschitzian functions. *Mathematics of Computation*, 77(261):379–395, 2008.

[3] A. Y. Aravkin, A. Kambadur, A. C. Lozano, and R. Luss. Sparse quantile huber regression for efficient and robust estimation. *arXiv preprint arXiv:1402.4624*, 2014.

[4] S. Barnett. Greatest common divisor of two polynomials. *Linear Algebra and its Applications*, 3(1):7–9, 1970.

[5] S. Bhojanapalli, A. Kyrillidis, and S. Sanghavi. Dropping convexity for faster semi-definite optimization. In *Conference on Learning Theory*, pages 530–582, 2016.

[6] D. Bokde, S. Girase, and D. Mukhopadhyay. Matrix factorization model in collaborative filtering algorithms: A survey. *Procedia Computer Science*, 49:136–146, 2015.

[7] L. Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[8] E. J. Candes. The restricted isometry property and its implications for compressed sensing. *Comptes rendus mathematique*, 346(9-10):589–592, 2008.

[9] E. J. Candes, X. Li, and M. Soltanolkotabi. Phase retrieval via wirtinger flow: Theory and algorithms. *IEEE Transactions on Information Theory*, 61(4):1985–2007, 2015.

[10] X. Chen. First order conditions for nonsmooth discretized constrained optimal control problems. *SIAM journal on control and optimization*, 42(6):2004–2015, 2004.

[11] X. Chen. Smoothing methods for nonsmooth, nonconvex minimization. *Mathematical programming*, 134(1):71–99, 2012.

[12] X. Chen, R. S. Womersley, and J. J. Ye. Minimizing the condition number of a gram matrix. *SIAM Journal on optimization*, 21(1):127–148, 2011.

[13] J. Fan. *Local polynomial modelling and its applications: monographs on statistics and applied probability 66*. Routledge, 2018.

[14] O. Fercoq and P. Richtárik. Smooth minimization of nonsmooth functions with parallel coordinate descent methods. *arXiv preprint arXiv:1309.5885*, 2013.

[15] R. Garg and R. Khandekar. Gradient descent with sparsification: an iterative algorithm for sparse recovery with restricted isometry property. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 337–344. ACM, 2009.

[16] D. Gross, Y.-K. Liu, S. T. Flammia, S. Becker, and J. Eisert. Quantum state tomography via compressed sensing. *Physical review letters*, 105(15):150401, 2010.

[17] N. Guan, D. Tao, Z. Luo, and J. Shawe-Taylor. Mahnmf: Manhattan non-negative matrix factorization. *arXiv preprint arXiv:1207.3438*, 2012.

[18] P. J. Huber. *Robust statistics*. John Wiley and Sons, 1981.

[19] P. Jain, R. Meka, and I. S. Dhillon. Guaranteed rank minimization via singular value projection. In *Advances in Neural Information Processing Systems*, pages 937–945, 2010.

[20] P. Jain, P. Netrapalli, and S. Sanghavi. Low-rank matrix completion using alternating minimization. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 665–674. ACM, 2013.

[21] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[22] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.

[23] I. Markovsky and K. Usevich. *Low rank approximation*. Springer, 2012.

[24] J. Mount. Approximation by orthogonal transform. winvector.gitbuh.io/xDrift/orthApprox.pdf, 2014. [Online; accessed 05-Sep-2018].

[25] Y. Nesterov. A method of solving a convex programming problem with convergence rate o(1/sqr(k)). *Soviet Mathematics Doklady*, 27:372–376, 1983.

[26] Y. Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005.

[27] A. B. Owen. A robust hybrid of lasso and ridge regression. *Contemporary Mathematics*, 443(7):59–72, 2007.

[28] S. Oymak, B. Recht, and M. Soltanolkotabi. Sharp time–data tradeoffs for linear inverse problems. *IEEE Transactions on Information Theory*, 64(6):4129–4158, 2018.

[29] D. Park, A. Kyrillidis, C. Caramanis, and S. Sanghavi. Finding low-rank solutions via non-convex matrix factorization, efficiently and provably. *arXiv preprint arXiv:1606.03168*, 2016.

[30] G. Picci. Stochastic realization theory. In *Mathematical System Theory*, pages 213–229. Springer, 1991.

[31] B. Recht, M. Fazel, and P. A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, 52(3):471–501, 2010.

[32] P. Resnick and H. R. Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[33] B. Riley. Convergence analysis of deterministic and stochastic methods for convex optimization. Master's thesis, University of Waterloo, 2017.

[34] W. Su, S. Boyd, and E. Candes. A differential equation for modeling nesterov's accelerated gradient method: Theory and insights. In *Advances in Neural Information Processing Systems*, pages 2510–2518, 2014.

[35] R. Sun and Z.-Q. Luo. Guaranteed matrix completion via non-convex factorization. *IEEE Transactions on Information Theory*, 62(11):6535–6579, 2016.

[36] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

[37] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[38] S. Tu, R. Boczar, M. Simchowitz, M. Soltanolkotabi, and B. Recht. Low-rank solutions of linear matrix equations via procrustes flow. In *International Conference on Machine Learning*, pages 964–973, 2016.
[39] C. J. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.
[40] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.
[41] Z. Yang, Y. Zhang, W. Yan, Y. Xiang, and S. Xie. A fast non-smooth nonnegative matrix factorization for learning sparse representation. *IEEE access*, 4:5161–5168, 2016.
[42] J. Zhang and I. Mitliagkas. Yellowfin and the art of momentum tuning. *arXiv preprint arXiv:1706.03471*, 2017.
[43] T. Zhao, Z. Wang, and H. Liu. Nonconvex low rank matrix factorization via inexact first order oracle. *Advances in Neural Information Processing Systems*, 2015.
[44] R. Zhu, D. Niu, and Z. Li. Robust web service recommendation via quantile matrix factorization. In *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*, pages 1–9. IEEE, 2017.

A Appendix A: Proof Sketch of Theorem 2

In the prove of Theorem 2, we will only provide the proof road map, the key idea is first to prove the convergence of SVP initialization, then the alternating minimization have an linear convergence rate combined with Nesterov's momentum algorithm. Previously [38] has already prove that alternating minimization have an linear convergence rate combined with gradient descent under least square matrix factorization setting by using the results from [9], here we prove that alternating minimization also have an linear convergence rate combined with Nesterov's momentum algorithm under nonsmooth matrix factorization, given the condition that the nonsmooth function is L smooth and strong convex, here we employ another result from [33] which analyze the linear convergence rate for Nesterov's momentum algorithm by using Lyapunov function.

Below only provide results for $\boldsymbol{f}_\mu$, in Theorem 1 we will provide another proof that actually the solution of $\boldsymbol{f}_\mu^\lambda$ will converge to $\boldsymbol{f}^\lambda$. Assumptions for function $\boldsymbol{f}_\mu$ are as follows:

1 This assumption is ξ-strongly convex assumption [29]: $\boldsymbol{f}_\mu(\mathcal{A}(N)-b)-\boldsymbol{f}_\mu(\mathcal{A}(M)-b) \geq \langle \nabla \boldsymbol{f}_\mu(\mathcal{A}(M)-b), N-M\rangle + \xi\|\mathcal{A}(M)-\mathcal{A}(N)\|_2^2$.
2 This assumption basically means that $\boldsymbol{f}_\mu(\mathcal{A}(N)-b)-\boldsymbol{f}_\mu(\mathcal{A}(M)-b) \leq \langle \nabla \boldsymbol{f}_\mu(\mathcal{A}(M)-b), N-M\rangle + \eta\|\mathcal{A}(M)-\mathcal{A}(N)\|_2^2$.
3 Without loss of generality, assume M^* is the optimal matrix, then $\boldsymbol{f}_\mu(\mathcal{A}(M^*)-b)=0$, and $\boldsymbol{f}_\mu(M)\geq 0$.
4 Assume that $\boldsymbol{f}_\mu$ also defines a matrix norm when act on a matrix M, and $c_1\|X\|_2^2 \leq \boldsymbol{f}_\mu(M) \leq c_2\|M\|_2^2$.

Remark: Assumption 1-2 defines the condition number of function $\boldsymbol{f}_\mu(\cdot)$, similar assumptions also appears in [34]. Usually $\kappa = \eta/\xi$ is called the condition number of a function [5].

Lemma 1. *(Restricted Isometry Property (RIP)) A linear map $\mathcal{A}$ satisfies the r-RIP with constant δ_r, if*

$$(1-\delta_r)\|M\|_F^2 \leq \|\mathcal{A}(M)\|_2^2 \leq (1+\delta_r)\|M\|_F^2$$

is satisfied for all matrices $M \in \mathbb{R}^{m\times n}$ of rank at most r.

Lemma 2. *[8] Let $\mathcal{A}$ satisfy 2r-RIP with constant δ_{2r}. Then for all matrices M,N of rank at most r, we have*

$$|\langle \mathcal{A}(M), \mathcal{A}(N)\rangle - \langle M, N\rangle| \leq \delta_{2r}\|M\|_F\|N\|_F.$$

The next lemma characterizes the convergence rate of initialization procedure:

Lemma 3. *[28] Let $M \in \mathbb{R}^{m\times n}$ be an arbitrary matrix of rank r. Also let $b = \mathcal{A}(M) \in \mathbb{R}^p$ be p linear measurements. Consider the iterative updates*

$$M^{t+1} \leftarrow \mathcal{P}_r\left(M^t - \xi_t \nabla_M \boldsymbol{f}_\mu(\mathcal{A}(M^t) - b)\right).$$

where M^i are $m \times n$ matrices. Then

$$\|M^t - M\|_F \leq \psi(\mathcal{A})^t \|M^0 - M\|_F.$$

holds. Here $\psi(\mathcal{A})$ is defined as

$$\psi(\mathcal{A}) = 2sup_{\|M\|_F=\|N\|_F=1, rank(M)\leq 2r, rank(N)\leq 2r} |\langle \mathcal{A}(M), \mathcal{A}(N)\rangle - \langle M, N\rangle|.$$

We can prove this lemma by using the results of Theorem 3.

First we will prove that the initialization procedure is indeed converge to the true value of M.

A.1 proof of the initialization results, equation (1) in Theorem 2

Lemma 4 (Lemma for initialization). *Assume that $\xi\frac{1+\delta_{2k}}{1-\delta_{2k}} - \eta > 0$. Denote $\tilde{\boldsymbol{f}}_\mu(M) = \boldsymbol{f}_\mu(\mathcal{A}(M) - b)$. Let M^* be an optimal solution and let M^t be the iterate obtained by the SVP algorithm at t-th iteration. Then*

$$\tilde{\boldsymbol{f}}_\mu(M^{t+1}) \leq \tilde{\boldsymbol{f}}_\mu(M^t) + \left(\xi\frac{1+\delta_{2k}}{1-\delta_{2k}} - \eta\right)\|\mathcal{A}(M^* - M^t)\|_2^2.$$

Proof. From assumption, we have

$$\begin{aligned}&\tilde{\boldsymbol{f}}_\mu(M^{t+1}) - \tilde{\boldsymbol{f}}_\mu(M^t) \leq \langle \nabla\tilde{\boldsymbol{f}}_\mu(M^t), M^{t+1} - M^t\rangle + \xi\|\mathcal{A}(M^{t+1}) - \mathcal{A}(M^t)\|_2^2\\ &\leq \langle \nabla\tilde{\boldsymbol{f}}_\mu(M^t), M^{t+1} - M^t\rangle + \xi(1+\delta_{2k})\|M^{t+1} - M^t\|_F^2\end{aligned}$$

where the last inequality comes from RIP. Let $N^{t+1} = M^t - \frac{1}{2\xi(1+\delta_{2k})}\nabla\tilde{\boldsymbol{f}}_\mu(M^t)$, and

$$f_t(M) = \langle \nabla\tilde{\boldsymbol{f}}_\mu(M^t), M - M^t\rangle + \xi(1+\delta_{2k})\|M - M^t\|_F^2.$$

Then

$$f_t(M) = \xi(1+\delta_{2k})\left[\|M - N^{t+1}\|_F^2 - \frac{1}{4\xi^2(1+\delta_{2k})^2}\|\nabla \tilde{\boldsymbol{f}}_\mu(M^t)\|_F^2\right]$$

By definition, $\mathcal{P}_k(N^{t+1}) = M^{t+1}$, then $f_t(M^{t+1}) \le f_t(M^*)$. Thus

$$\begin{aligned}
&\tilde{\boldsymbol{f}}_\mu(M^{t+1}) - \tilde{\boldsymbol{f}}_\mu(M^t) \le f_t(M^{t+1}) \le f_t(M^*)\\
&= \langle \nabla \tilde{\boldsymbol{f}}_\mu(M^t), M^* - M^t\rangle + \xi(1+\delta_{2k})\|M^* - M^t\|_F^2\\
&\le \nabla \tilde{\boldsymbol{f}}_\mu(M^t), M^* - M^t\rangle + \xi\frac{1+\delta_{2k}}{1-\delta_{2k}}\|\mathcal{A}(M^*) - \mathcal{A}(M^t)\|_F^2\\
&\le \nabla \tilde{\boldsymbol{f}}_\mu(M^t), M^* - M^t\rangle + \eta\|\mathcal{A}(M^*) - \mathcal{A}(M^t)\|_F^2 + \left(\xi\frac{1+\delta_{2k}}{1-\delta_{2k}} - \eta\right)\|\mathcal{A}(M^*) - \mathcal{A}(M^t)\|_F^2\\
&\le \tilde{\boldsymbol{f}}_\mu(M^*) - \tilde{\boldsymbol{f}}_\mu(M^t) + \left(\xi\frac{1+\delta_{2k}}{1-\delta_{2k}} - \eta\right)\|\mathcal{A}(M^*) - \mathcal{A}(M^t)\|_F^2
\end{aligned}$$

Theorem 3. *Let $b = \mathcal{A}(M^*) + e$ for rank k matrix M^* and an error vector $e \in \mathbb{R}^p$. Then, under the assumption that $D < 1$, where $D = \frac{1}{C^2} + \left(\xi\frac{1+\delta_{2k}}{1-\delta_{2k}} - \eta\right)\left(\frac{2}{C^2} + \sqrt{\frac{2}{c_1}}\frac{1}{C} + \frac{1}{c_1}\right)$, the SVP algorithm with step size $\eta_t = \frac{1}{2\xi(1+\delta_{2k})}$ outputs a matrix M of rank at most k such that $\tilde{\boldsymbol{f}}_\mu(\mathcal{A}(M) - b) \le (C^2+\varepsilon)\frac{\|e\|^2}{2}$, $\varepsilon \ge 0$, in at most $\left\lceil \frac{1}{\log D}\log\frac{(C^2+\varepsilon)\|e\|^2}{2c_2\|b\|_2^2}\right\rceil$ iterations.*

Proof. Let the current solution M^t satisfy $\boldsymbol{f}_\mu(M^t) \ge \frac{C^2\|e\|^2}{2}$, by lemma 4 and $b - \mathcal{A}(M^*) = e$, we have

$$\begin{aligned}
\boldsymbol{f}_\mu(M^{t+1}) &\le \frac{\|e\|^2}{2} + \left(\xi\frac{1+\delta_{2k}}{1-\delta_{2k}} - \eta\right)\|b - \mathcal{A}(M^t) - e\|^2\\
&\le \frac{\|e\|^2}{2} + \left(\xi\frac{1+\delta_{2k}}{1-\delta_{2k}} - \eta\right)\left(\|b - \mathcal{A}(M^t)\|^2 - 2e^\top(b - \mathcal{A}(M^t)) + \|e\|^2\right)\\
&\le \frac{\boldsymbol{f}_\mu(M^t)}{C^2} + \left(\xi\frac{1+\delta_{2k}}{1-\delta_{2k}} - \eta\right)\left(\frac{2\boldsymbol{f}_\mu(M^t)}{C^2} + \frac{\boldsymbol{f}_\mu(M^t)}{c_1} + \frac{\sqrt{2}\boldsymbol{f}_\mu(M^t)}{C\sqrt{c_1}}\right)\\
&= D\boldsymbol{f}_\mu(M^t).
\end{aligned}$$

Since $D < 1$, combine the fact that $\boldsymbol{f}_\mu(M^0) \le c_2\|b\|^2$, by taking $t = \left\lceil \frac{1}{\log D}\log\frac{(C^2+\varepsilon)\|e\|^2}{2c_2\|b\|_2^2}\right\rceil$, we complete the proof.

The following lemma was adapted from [38]:

Lemma 5. *Let $M_1, M_2 \in \mathbb{R}^{m\times n}$ be two rank r matrices with SVD decomposition $M_1 = U_1^\top\Sigma_1 V_1$, $M_2 = U_2^\top\Sigma_2 V_2$, for $l = 1, 2$, define $M_l = U_l^\top\Sigma_l^{1/2} \in \mathbb{R}^{m\times r}$, $N_l = V_l^\top\Sigma_l^{1/2} \in \mathbb{R}^{n\times r}$. Assume M_1, M_2 obey $\|M_2 - M_1\| \le \frac{1}{2}\sigma_r(M_1)$. Then:*

$$dist^2\left(\begin{bmatrix}M_2\\N_2\end{bmatrix}, \begin{bmatrix}M_1\\N_1\end{bmatrix}\right) \le \frac{2}{\sqrt{2}-1}\frac{\|M_2 - M_1\|_F^2}{\sigma_r(X_1)}.$$

Combine lemma 1, 2 and 5, follow the similar route of [38], we can prove that using more than $3\log(\sqrt{r}\kappa)+5$ iterations of SVP initialization algorithm, we can obtain

$$\text{dist}\left(\begin{bmatrix}U_0\\V_0\end{bmatrix},\begin{bmatrix}U\\V\end{bmatrix}\right)\leq\frac{1}{4}\sigma_r(U). \tag{11}$$

Thus we finished the proof of convergence in the initialization procedure, next we will study the linear decay rate for each iteration for penalized object function.

Before study the theoretical properties, we first rearrange object function (7) by using the uplifting technique so that it's easy to simultaneously consider $\boldsymbol{f}_\mu$ and the regularization term, to see this, consider rank r matrix $M\in\mathbb{R}^{m\times n}$ with SVD decomposition $M=U^\top\Sigma V$, define Sym : $\mathbb{R}^{m\times n}\to\mathbb{R}^{m\times n}$ as

$$\text{Sym}(X)=\begin{bmatrix}0_{m\times m} & X\\ X^\top & 0_{n\times n}\end{bmatrix}.$$

Given the block matrix $\mathbf{A}=\begin{bmatrix}\mathbf{A}_{11}\ \mathbf{A}_{12}\\ \mathbf{A}_{21}\ \mathbf{A}_{22}\end{bmatrix}$ with $\mathbf{A}_{11}\in\mathbb{R}^{m\times m}$, $\mathbf{A}_{12}\in\mathbb{R}^{m\times n}$, $\mathbf{A}_{21}\in\mathbb{R}^{n\times m}$, $\mathbf{A}_{22}\in\mathbb{R}^{n\times n}$. Define $\mathcal{P}_{\text{diag}}(\mathbf{A})=\begin{bmatrix}\mathbf{A}_{11} & 0_{m\times n}\\ 0_{n\times m} & \mathbf{A}_{22}\end{bmatrix}$, $\mathcal{P}_{\text{off}}(\mathbf{A})=\begin{bmatrix}0_{m\times m} & \mathbf{A}_{12}\\ \mathbf{A}_{21} & 0_{n\times n}\end{bmatrix}$, define $\mathcal{B}:\mathbb{R}^{(m+n)\times(m+n)}\to\mathbb{R}^{(m+n)\times(m+n)}$ as the uplift version of $\mathcal{A}$ operator:

$$\mathcal{B}(M)_k=\langle B_k,M\rangle,\ \text{where } B_k=\text{Sym}(M).$$

Define $W=[U^\top;V^\top]$, Then as a result, we can rewrite object function (7) as:

$$\begin{aligned}&g(W)\coloneqq g(U,V)\\&=\boldsymbol{f}_\mu(b-\mathcal{A}(U^\top V))+\lambda\|UU^\top-VV^\top\|_F^2\\&=\frac{1}{2}\boldsymbol{f}_\mu\left(\mathcal{B}(\text{Sym}(U^\top V))-\text{Sym}(X)\right)+\frac{1}{2}\lambda\|\text{Sym}(U^\top V)-\text{Sym}(X)\|_F^2.\end{aligned}$$

From equation (12) we can see that actually the non-penalize part and penalize part have similar structure.

As a result, although we made the assumption 1,2,4 on function $\mathcal{M}$, we can see from equation (12) that after adding the penalization term, the penalized object function still retains similar property, as for assumption 3, we can also use some location transform techniques to make the penalized object function satisfies this assumption, as a result, it does not make so much difference whether we deal with penalized object function or un-penalized object function.

Then similar to [38], the alternating minimization incorporate with Nesterov's momentum algorithm with respect to U and V (sub vector of W), respectively, are actually can be written as the NAG algorithm applied to $g(W)$ with respect to W.

For the convergence analysis of Nesterov's momentum algorithm, we employ follow lemma, which is a theorem in [33]:

Lemma 6. *For minimization problem* $\min_{x\in\mathcal{X}}f(x)$*, where* x *is a vector, using the Lyapunov function*

$$\tilde{V}_k=f(y_k)+\xi\|z_k-x^*\|^2$$

it can be shown

$$\tilde{V}_{k+1} - \tilde{V}_k = -\tau_k \tilde{V}_k + \varepsilon_{k+1} \tag{12}$$

where the error is expressed as

$$\varepsilon_{k+1} = \left(\frac{\tau_k^2}{4\xi}\right) \|\nabla f(x_k)\|^2 + \left(\tau_k \eta - \frac{\xi}{\tau_k}\right) \|x_k - y_k\|^2,$$

τ_k is the step size in Nesterov's momentum algorithm, usually equals $1/\sqrt{\kappa}$, $y_{k+1} = x_k - \frac{1}{2\eta}\nabla f(x_k)$, $x_{k+1} = \frac{1}{1+\tau_k} y_k + \frac{\tau_k}{1+\tau_k} z_k$, $z_{k+1} = z_k + \tau_k \left(x_{k+1} - z_k - \frac{1}{2\xi}\nabla f(x+k+1)\right)$.

Assume that $\tau_0 = 0$, $\tau_1 = \tau_2 = \cdots = \tilde{\tau}$, and $\varepsilon_1, \cdots, \varepsilon_{k+1}$ has a common upper bound $\tilde{\varepsilon}$, then (12) implies:

$$|\tilde{V}_{k+1}| = |(1-\tilde{\tau})^{k+1}\tilde{V}_0 + \sum_{i=1}^{k+1}(1-\tilde{\tau})^{i-1}\varepsilon_{k+2-i}| \leq (1-\tilde{\tau})^{k+1}|\tilde{V}_0| + \frac{\tilde{\varepsilon} - \tilde{\varepsilon}(1-\tilde{\tau})^k}{\tilde{\tau}}$$

Substitute x_k with $W_{k+1} = [U^{k+1}, V^{k+1}]^\top$, f with g, if we want to deal with the convergence analysis with respect to $W_t - W^*$ and $W_0 - W^*$, we need to handle two parts, the first part is the error part with respect to $\tilde{\varepsilon}$, this can be solved by choosing initial estimate close to true value, as a result $\|\nabla f(x_1)\|$ can be arbitrary close to 0. For the sake of notation simplicity, assume that $\frac{\tilde{\varepsilon} - \tilde{\varepsilon}(1-\tilde{\tau})^k}{\tilde{\tau}} \leq \varepsilon^\dagger$. Since $\tilde{V}_k$ still satisfies assumption 1 and 2, without loss of generality, assume the corresponding parameter are $\tilde{\xi}$ and $\tilde{\eta}$.

In the next, we want to seek the relation of $W_t - W^*$ with $\tilde{V}_t$. This will involve the assumption 1 and 2 as well as lemma 1. With a rough handle of the gradient part in assumption 1 and 2, we can obtain

$$\tilde{\xi}\|\mathcal{A}(W_t) - \mathcal{A}(W^*)\|_2^2 \leq (1-\tilde{\tau})^t \tilde{\mu}\|\mathcal{A}(W_0) - \mathcal{A}(W^*)\|_2^2 + \tilde{\varepsilon}$$

Notice that $\tilde{\varepsilon}$ can be made arbitrary small so that

$$\tilde{\xi}\|\mathcal{A}(W_t) - \mathcal{A}(W^*)\|_2^2 \leq (1-\tilde{\tau}_1)^t \tilde{\mu}\|\mathcal{A}(W_0) - \mathcal{A}(W^*)\|_2^2$$

and $1 - \tilde{\tau}_1$ still larger than 0 smaller than 1. Employ the Restricted Isometry Property,

$$\tilde{\xi}(1-\delta_r)\|W_t - W^*\|_2^2 \leq (1-\tilde{\tau}_1)^t \tilde{\mu}(1+\delta_r)\|W_0 - W^*\|_2^2$$

Thus

$$\begin{aligned}\text{dist}\left(\begin{bmatrix} U_t \\ V_t \end{bmatrix}, \begin{bmatrix} U \\ V \end{bmatrix}\right) &\leq (1-\tilde{\tau}_1)^t \frac{\tilde{\mu}}{\tilde{\xi}} \frac{1+\delta_r}{1-\delta_r} \text{dist}\left(\begin{bmatrix} U_0 \\ V_0 \end{bmatrix}, \begin{bmatrix} U \\ V \end{bmatrix}\right) \\ &\leq \frac{1}{4}(1-\tilde{\tau}_1)^t \frac{\tilde{\mu}}{\tilde{\xi}} \frac{1+\delta_r}{1-\delta_r} \sigma_r(U)\end{aligned}$$

Theorem 2 is proved.

Remark on equation (12): From (12), we provide a guideline with respect to the selection of λ compared with [38], by combine (12) and assumption 4.