Greedy Bilateral Sketch, Completion & Smoothing

Tianyi Zhou
Centre for Quantum Comp. & Intelligent Syst.
University of Technology Sydney, Australia

Dacheng Tao
Centre for Quantum Comp. & Intelligent Syst.
University of Technology Sydney, Australia

Abstract

Recovering a large low-rank matrix from highly corrupted, incomplete or sparse outlier overwhelmed observations is the crux of various intriguing statistical problems. We explore the power of "greedy bilateral (GreB)" paradigm in reducing both time and sample complexities for solving these problems. GreB models a low-rank variable as a bilateral factorization, and updates the left and right factors in a mutually adaptive and greedy incremental manner. We detail how to model and solve low-rank approximation, matrix completion and robust PCA in GreB's paradigm. On their MATLAB implementations, approximating a noisy $10^4 \times 10^4$ matrix of rank 500 with SVD accuracy takes 6s; MovieLens10M matrix of size 69878×10677 can be completed in 10s from 30% of 10^7 ratings with RMSE 0.86 on the rest 70%; the low-rank background and sparse moving outliers in a 120×160 video of 500 frames are accurately separated in 1s. This brings 30 to 100 times acceleration in solving these popular statistical problems.

1 Introduction

Explosion of information introduces dramatic increasing data collected from Internet and digital sensors. These data are usually featured by their high-dimension, huge volume, serious incompleteness, dominating noise and complicated structure, which bring intriguing new challenges to compressive acquisition, signal processing and machine learning. Because traditional methods are limited by their time/sample complexities, storage and denoising capability. This fact has driven the recent exploitation of data intrinsic redundancy and structures. For a single signal or individual instance, the redundancy is usually exhibited by its sparsity, i.e., it is nonzero only on a small set of entries. Compressed sensing (Donoho, 2006)(Candès and Tao, 2006) recovers a signal from its highly compressed measurements, via solving a undetermined linear system, by leveraging the sparsity. For multiple instances, or more specifically a matrix, the redundancy is often identified by its low-rank structure, i.e., all the instances lie in a subspace spanned by a small number of bases. Low-rank structure can be analogized to sparsity due to its sparse spectrum. Equivalently saying, the matrix X can be written as the sum of a few rank-1 matrices such that $X = \sum_{i=1}^{r} U_i V_i$, wherein U_i is a column vector and V_i is a row vector.

Appearing in Proceedings of the 16th International Conference on Artificial Intelligence and Statistics (AISTATS) 2013, Scottsdale, AZ, USA. Volume 31 of JMLR: W&CP 31. Copyright 2013 by the authors.

Low-rank structure arises in a wide range of fundamental problems. In this paper, we mainly focus on three representatives catching substantial interests in several important applications: low-rank approximation (Halko et al., 2009), matrix completion (Candès and Recht, 2008) and robust PCA (Candès et al., 2009). In this paper, the low-rank matrix variable in these problems is modeled in a bilateral factorization form UV for the purpose of developing SVD-free algorithms.

We describe and analyze a general scheme called "greedy bilateral (GreB)" paradigm for solving the mainstream low-rank matrix recovery problems. GreB starts from U and V respectively containing a very few (e.g., one) columns and rows, and optimizes them alternately. Their updates are based on observation that the object value is determined by the product UV rather than individual U or V. Thus we can choose a different pair (U, V) producing the same UV but computed faster than the one derived by alternating least squares like in IRLS-M (Fornasier et al., 2011) and ALS (Zachariah et al., 2012). In GreB, the updates of U and V can be viewed as mutually adaptive update of the left and right sketches of the low-rank matrix. Such updates are repeated until the object convergence, then a few more columns (or rows) are concatenated to the obtained U (or V), and the alternating updates are restarted on a higher rank. Here, the added columns (or rows) are selected in a greedy manner. Specifically, they are composed of the rank-1 column (or row) directions on which the object decreases fastest. GreB incrementally increases the rank until when UV is adequately consistent with the observations.

GreB's greedy strategy avoids the failures brought by possible biased rank estimation. Moreover, greedy selecting optimization directions from 1 to r is faster than updating r directions in all iterates like in LMaFit (Wen et al., 2012) and (Zhou and Tao, 2011). In addition, the lower rank solution before each rank increment is invoked as the "warm start" of the next higher rank optimization and thus speed up convergence. Furthermore, its mutually adaptive updates of U and V yields a simple yet efficient SVD-free implementation. Under GreB paradigm, the overall time complexity of matrix completion is $\mathcal{O}(\max\{\|\Omega\|_0 r^2, (m+n)r^3\})$ (Ω-sampling set, $m \times n$-matrix size, r-rank), while the overall complexities of low-rank approximation and noisy robust PCA are $\mathcal{O}(mnr^2)$. An improvement on sample complexity can also be justified in our experiments.

2 Background and Problem Formulation

2.1 Low-rank Approximation

Real world data matrix is hardly to be exactly low-rank. However, approximating it by a low-rank one presents a good trade-off between accuracy and time/space costs, especially when its singular values decay fast. The low-rank approximation (Ye, 2005) can replace the original matrix in least square regression and matrix product, and also provides a low-dimensional representation which can boosts the classification and clustering performance. Although the low-rank approximation is provably optimal when constructed from SVD, the expensive time cost makes SVD prohibitive to large matrix. Thus many faster Monte Carlo algorithms (Drineas et al., 2006)(Clarkson and Woodruff, 2009)(Nguyen et al., 2009) have been proposed at the cost of producing sufficiently small error to SVD with high probability. Typically, they build the low-rank approximation from random selected columns or rows (Boutsidis et al., 2009), or linear projections (which are called "sketch") on certain random matrix (Halko et al., 2009). Some of them compute bilateral sketches (Fazel et al., 2008)(Zhou and Tao, 2012a) for both column space and row space in building the approximation. In this case, the matrix X is approximated as $X = USV$, wherein $U = XA_1$ and $V = A_2^T X$ are right and left sketches, A_1 and A_2 are random matrices.

In this paper, we omit S and consider optimizing U and V:

$$\begin{aligned}&\min_{U,V} \|X - UV\|_F^2\\ &\text{s.t. } rank(U) = rank(V) \le r.\end{aligned} \tag{1}$$

2.2 Matrix Completion

Data is usually obtained with many missing values. The goal of matrix completion (Candès and Recht, 2008)(Candès and Tao, 2009) is to recover the whole data matrix X from partial noisy entries or undersampled linear measurements $\mathcal{A}(X)$ ($\mathcal{A}$ is the sampling operator) by leveraging the connections between different instances. Such connections can be well captured by low-rank structure. Matrix completion has broad applications in various real problems of considerable importance, such as collaborative filtering in recommendation system, link prediction for social network, quantum state tomography and traffic distance completion. The matrix completion problem was firstly written as a rank minimization that is NP-hard both to solve and approximate. Thus trace norm (ℓ_1 norm of singular values) as the convex surrogate of rank has been minimized in many popular approaches (Cai et al., 2010)(Ji and Ye, 2009). For a matrix $X \in \mathbb{R}^{m\times n}$ of rank-r with SVD decomposition $X = B\Sigma D$, the obtained global solution is provable to exactly recover X from $\mathcal{O}(nr\log^6 n)$ uniformly sampled noisy entries if it fulfills incoherence property ($\mathcal{P}_B\cdot$ denotes the orthogonal projection onto B and e_i is standard basis)

$$\max_{1\le i\le m} \|\mathcal{P}_B e_i\|^2 \le \mu_0 r/m, \max_{1\le i\le n} \|\mathcal{P}_D e_i\|^2 \le \mu_0 r/n, \tag{2}$$

entries are upper bounded as $\max_{i,j}|X_{ij}| \le \mu_1\sqrt{r}/\sqrt{mn}$, and $\mathcal{A}$ obeys matrix restricted isometry property (RIP) with constant δ

$$(1-\delta)\|X\|_F^2 \le \|\mathcal{A}(X)\|_F^2 \le (1+\delta)\|X\|_F^2. \tag{3}$$

Another norm receiving much attention for encouraging low-rank solution is max-norm $\|X\|_{max} = \inf\{\|U\|_{2,\infty}\|V^T\|_{2,\infty} : X = UV\}$ (Srebro et al., 2005)(Lee et al., 2010) ($\|\cdot\|_{2,\infty}$ is the maximum ℓ_2 row norm of a matrix), whose theoretical recovery bound is established based on Rademacher complexity of its unit ball (Foygel and Srebro, 2011). The max-norm can be defined as the global solution to an non-convex optimization on left and right factors U and V.

Both trace norm minimization and max-norm minimization can be formulated as semidefinite programming (SDP) which has standard solvers. Various accelerated optimization methodologies (Ma et al., 2011)(Lee et al., 2010) also have been applied to them for practical purpose. However, most of them rely on costly computation of full SVD in each iterate, and thus do not scale well to large-scale problems. This fact induces a revisit of rank minimization/constraint based formulations. Forward greedy selection methods including ADMiRA (Lee and Bresler, 2010) and GECO (Shalev-Shwartz et al., 2011) are developed for rank minimization. GECO adopts an interesting greedy strategy: it increments the rank by 1 and adds the optimal rank-1 direction into the optimization per iterate. Incremental OptSpace (Keshavan and Oh, 2009) also has a similar scheme. We inherit a similar spirit but solve different optimization models in GreB. Error minimization with rank constraint is also considered in SVP (Jain et al., 2010). Some of them like OptSpace and GECO model the low-rank variable as factorization USV and optimize over (U, V) pair and S. Unfortunately, truncated SVD or large matrix multiplication is still required in these approaches. In addition, the update of S is to solve a large-scale overdetermined linear system and thus time consuming in practice. Furthermore, the rank is fixed within some algorithms, so in this case the recovery accuracy strongly relies on the quality of rank estimation. Online/stochastic gradient method (Balzano et al., 2010) and aggregated (divide-and-conquer) method (Mackey et al., 2011) have also been considered for pursuing approximated recovery. Since GreB can be straightforwardly transferred to or invoked as a subroutine by them to take their advantages, they will not be included in later comparison.

In this paper, we optimize the bilateral factorization form $X = UV$ of low-rank matrix X and address a rank constrained optimization:

$$\begin{array}{l} \min_{U,V,S} \|M - UV - S\|_F^2 \\ \text{s.t. } rank(U) = rank(V) \leq r, \mathcal{P}_\Omega S = 0, \end{array} \tag{4}$$

where $\mathcal{P}_\Omega\cdot$ denotes the projection of a matrix on an element subset $\Omega \subset [m] \times [n]$

$$\mathcal{P}_\Omega X = \begin{cases} X_{ij}, & (i,j) \in \Omega \\ 0, & (i,j) \in \Omega^C \end{cases} \tag{5}$$

and M consists of the observed entries and defined as $M = P_\Omega X$.

2.3 Robust PCA

For data with sparse outliers or partially contaminated by noise of overwhelming magnitude, sheer low-rank assumption cannot fully capture its complex structure. A more general assumption $X = L + S$ is applied in this case (Chandrasekaran et al., 2009), i.e., the data matrix X can be decomposed as the sum of a low-rank matrix L and a sparse matrix S. L explains the components that lie in a subspace and smoothly change across different instances, while S contains the spiky anomalies that are rarely shared by different instances. This model is called "robust PCA" (Candès et al., 2009) due to its robustness to sparse noise S when recovering principle components of L from X, and can be applied to video surveillance, graphical mode selection, image alignment, multi-label learning (Zhou and Tao, 2012b), etc.

PCP (Candès et al., 2009) recovers L and S from X by minimizing sum of the trace norm of L and the ℓ_1 norm of S. It can be proved that the solution to this convex relaxation is the exact recovery if $X = L + S$ indeed exists and L and S are sufficiently incoherent (Chandrasekaran et al., 2009)(Candès et al., 2009). That is, L obeys the incoherence property in (2) and thus is not sparse, while S has nonzero entries uniformly selected at random and thus is not low-rank. Popular optimization algorithms such as augmented Lagrangian multiplier, accelerated proximal gradient method and accelerated projected gradient method (Chen et al., 2010) have been applied. But full SVD as a costly subroutine is required to be repeatedly invoked in any of them.

Despite the strong theoretical guarantee of robust PCA, the exact decomposition $X = L + S$ does not always exist for real data matrix X. Thus a more adaptive model $X = L + S + G$ is preferred, where $L + S$ approximates X and G is the dense noise. Such noisy robust PCA becomes the central interests of many recent works including stable PCP (Zhou et al., 2010), GoDec (Zhou and Tao, 2011) and DRMF (Xiong et al., 2010). Direct rank constraint for L and cardinality constraint for S have been employed in order to replace the full SVD with truncated SVD or faster low-rank approximation. They also face the rank estimation problem when determining the rank constraint r.

In this paper, we formulate the noisy robust PCA by replacing L with its bilateral factorization $L = UV$ and regularizing the ℓ_1 norm of S's entries:

$$\begin{array}{l} \min_{U,V,S} \|X - UV - S\|_F^2 + \lambda\|\text{vec}(S)\|_1 \\ \text{s.t. } rank(U) = rank(V) \leq r. \end{array} \tag{6}$$

The ℓ_1 regularization induces soft-thresholding in updating S, which is faster than sorting caused by cardinality constraint in GoDec and DRMF.

3 Greedy Bilateral (GreB) Paradigm

In this section, we sequentially detail how to solve low-rank approximation, matrix completion and noisy robust PCA in GreB's paradigm. The resulting three algorithms are named as "greedy bilateral sketch (GreBske)", "greedy bilateral completion (GreBcom)" and "greedy bilateral smoothing (GreBsmo)" respectively according to their normal usages in practical applications. We summarize all the three algorithms in GreB's paradigm given in Algorithm 1.

3.1 Greedy Bilateral Sketch

Alternately optimizing U and V in (1) immediately yields the following updating rules, note subscript in $\cdot_k$ denotes the variable in the k^{th} iterate and $(\cdot)^\dagger$ stands for the Moore-Penrose pseudo-inverse:

$$\begin{cases} U_k = XV_{k-1}^T \left(V_{k-1}V_{k-1}^T\right)^\dagger, \\ V_k = \left(U_k^T U_k\right)^\dagger U_k^T X. \end{cases} \tag{7}$$

It can be observed that the object value in (1) is merely determined by the matrix product UV rather than individual U or V, and different (U, V) pair can produce the same UV. It is then of interest to find a pair of (U, V) that have the same product as (U_k, V_k) in (7) but can be computed in

Algorithm 1: Greedy Bilateral (GreB) Paradigm

```
Input: Object function $f$; rank step size $\Delta r$; power $K$;
       tolerance $\tau$; observations of data matrix $X$
Output: low-rank matrix $UV$ (and sparse $S$)
Initialize $V \in \mathbb{R}^{r_0 \times n}$ (and $S$);
while residual error $\leq \tau$ do
    for $k \leftarrow 1$ to $K$ do
        Greedy Bilateral Sketch: sequentially compute (9);
        Greedy Bilateral Completion: sequentially
        compute (13);
        Greedy Bilateral Smoothing: sequentially
        compute (17);
    end
    Calculate the top $\Delta r$ right singular vectors $v$ (or
    $\Delta r$-dimensional random projections) of $\partial f / \partial V$
    (given in (10), (14) and (18) for different problems);
    Set $V := [V; v]$;
end
```

less time than U_k and V_k. For this purpose, we investigate the product $U_k V_k$

$$U_k V_k = U_k \left(U_k^T U_k\right)^{\dagger} U_k^T X = \mathcal{P}_{U_k} X. \tag{8}$$

This implies that the product $U_k V_k$ equals to the orthogonal projection of X onto the column space of U_k. According to (7), the column space of U_k can be represented by arbitrary orthonormal basis for the columns of XV_{k-1}^T. For example, we can compute it as Q via fast QR decomposition $XV_{k-1}^T = QR$. In this case, the product $U_k V_k$ can be equivalently computed as $U_k V_k = \mathcal{P}_Q X = QQ^T X$. Therefore, U_k and V_k in (7) can be replaced by Q and $Q^T X$ respectively, while the product $U_k V_k$ and the corresponding object value are kept the same. This gives a faster updating procedure

$$\begin{cases} U_k = Q, \mathrm{QR}\left(XV_{k-1}^T\right) = QR, \\ V_k = Q^T X. \end{cases} \tag{9}$$

This alternating update can be viewed as mutually adaptive optimization of right sketch XV^T and left sketch $Q^T X$ for X, where the right projection matrix V^T is the former left sketch, and the left projection matrix Q is the orthonormal basis of the former right sketch. This mutually adaptive optimization of left and right factors will appear in GreBcom and GreBsmo as well.

Since the right and left sketches respectively describe the column and row spaces, which largely decide the approximation precision, we can temporarily ignore the QR decomposition in order to see how the column/row space is tracked within this scheme. Specifically, we start from a random matrix V and repeat $U_k = XV_{k-1}, V_k = U_k^T X$ for K times, the resulting $V_K = V(X^T X)^K$ are exactly the randomized SVD under power scheme (Roweis, 1998) given in (Halko et al., 2009). The K-order matrix exponential accelerates the decaying of singular values and thus improves the approximation precision. Similar theoretical analysis as (Halko et al., 2009) supports that (7) achieves the same accuracy as power scheme randomized SVD.

Different from power scheme whose speed would be quickly limited with the increasing of power K, GreBske invokes the updates in (7) with a greedy incremental rank r for both U and V. In particular, GreBske starts from a $V \in \mathbb{R}^{n \times r_0}$ with a small integer r_0, iterates (7) for K times, and then augment the rank of V to $r_1 = r_0 + \Delta r$ by adding Δr extra rows to V, where Δr is the rank step size. In GreB, the Δr rows are selected greedily as the top Δr row basis on which the object decreases fastest. Accordingly, they maximizes the magnitude of the partial derivative of the object w.r.t. UV, which is

$$\frac{\partial \|X - UV\|_F^2}{\partial UV} = X - UV. \tag{10}$$

Hence the Δr rows are the top Δr right singular vectors of the fat matrix $U^T(X - UV)$, which can be quickly obtained by a small SVD or its faster approximation like random projections $A^T U^T (X - UV)$, wherein A is a $r_i \times \Delta r$ random matrix. The rank r stops augmenting when reaching certain error tolerance.

In GreBske, the top r_i row basis are successfully obtained when optimizing V of $r_i + \Delta r$ rows. The essential task of the updates is to optimize the added Δr rows, while the first r_i rows take part in the update merely for keep the incoherence between rows. So it converges faster than simultaneously optimizing the whole r rows. In addition, the newly added Δr rows are initialized as the fastest decreasing directions, so its distance to the optimal solution is shortened. The computation in GreBske is dominated by the two matrix multiplications that take $2mnr_i$ flops. It can be further speeded up if designing sparse updates of U and V, which will be studied in future works.

3.2 Greedy Bilateral Completion

Following a similar methodology as GreBske, alternately optimizing U, V and S in (4) immediately yields the following updating rules:

$$\begin{cases} U_k = (M - S_{k-1}) V_{k-1}^T \left(V_{k-1} V_{k-1}^T\right)^{\dagger}, \\ V_k = \left(U_k^T U_k\right)^{\dagger} U_k^T (M - S_{k-1}), \\ S_k = \mathcal{P}_{\Omega}^C (M - U_k V_k), \end{cases} \tag{11}$$

Note the the object value in (4) is solely determined by the matrix product UV and S rather than individual U or V, we use the same trick of replacing the (U, V) pair with a faster computed one as we did in GreBske. This yields

$$\begin{cases} U_k = Q, \mathrm{QR}\left((M - S_{k-1}) V_{k-1}^T\right) = QR, \\ V_k = Q^T (M - S_{k-1}), \\ S_k = -\mathcal{P}_{\Omega}^C (U_k V_k). \end{cases} \tag{12}$$

Since $M - S_{k-1} = M - \mathcal{P}_\Omega(U_{k-1}V_{k-1}) + U_{k-1}V_{k-1}$, define the residual $E = M - \mathcal{P}_\Omega(U_{k-1}V_{k-1})$, the above procedure (12) can be further accelerated as

$$\begin{cases} U_k = Q, \mathrm{QR}\left(E_{k-1}V_{k-1}^T + U_{k-1}\left(V_{k-1}V_{k-1}^T\right)\right) = QR, \\ V_k = Q^T E_{k-1} + \left(Q^T U_{k-1}\right) V_{k-1}, \\ E_k = M - \mathcal{P}_\Omega\left(U_k V_k\right). \end{cases} \tag{13}$$

It is not hard to verify that computation of the above procedure requires $3|\Omega|_0 r_i + (3m+2n)r_i^2$ flops for $U \in \mathbb{R}^{m \times r_i}$ and $V \in \mathbb{R}^{r_i \times n}$.

Similar to GreBske, GreBcom adopts a greedy strategy incrementally changing r_i to $r_i + \Delta r$ by concatenating new rows to V after executing K times of updates in (13). The number of iterating (13) can also be determined by the convergence of object, which implies a successful tracking of the first r_i basis within the row space of low-rank matrix X. The greedy selection of the Δr new rows is based on the partial derivative of the object w.r.t. UV

$$\frac{\partial \|M - UV - S\|_F^2}{\partial UV} = M - UV - S = E. \tag{14}$$

The Δr new rows are chosen as the Δr right singular vectors of $U^T E$ associated with the Δr largest singular values, which can be approximated by the random projections $A^T U^T E$ wherein A is a $r_i \times \Delta r$ random matrix (Halko et al., 2009). This greedy selection enables the object decreasing fastest along the newly added Δr basis in the next round of updates (13). GreBcom starts from a small r_0 and increases the rank until reaching certain tolerance for the residual, when the final rank r of X is determined automatically. As well as GreBske, such greedy incremental scheme brings evident improvement in speed.

We report the phase diagram of GreBcom in Figure 3.2. The white region denotes where GreBcom can successfully recover the incomplete low-rank matrix with probability of 1, while the black region is where the algorithm fails. It can be seen that GreBcom exhibits evident phase transition property, which is a significant phenomenon verified in many matrix completion algorithms. Compared to the published phase transition plots of them, GreBcom possesses a larger region for successful recovery on the ρ-r plane.

3.3 Greedy Bilateral Smoothing

Alternately optimizing U, V and S in (6) immediately yields the following updating rules:

$$\begin{cases} U_k = \left(X - S_{k-1}\right) V_{k-1}^T \left(V_{k-1}V_{k-1}^T\right)^\dagger, \\ V_k = \left(U_k^T U_k\right)^\dagger U_k^T \left(X - S_{k-1}\right), \\ S_k = \mathcal{S}_\lambda\left(X - U_k V_k\right), \end{cases} \tag{15}$$

where $\mathcal{S}_\lambda$ is an element-wise soft thresholding operator with threshold λ such that

$$\mathcal{S}_\lambda X = \left\{\mathrm{sgn}\left(X_{ij}\right)\max\left(|X_{ij}| - \lambda, 0\right) : (i,j) \in [m] \times [n]\right\}. \tag{16}$$

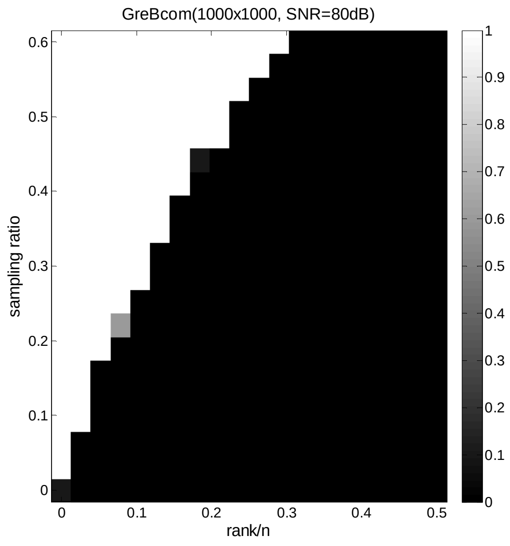

Figure 1: Phase diagram for GreBcom on 1000×1000 matrices. On the 20×20 grid of sampling ratio-rank/n plane, 10 trials are performed for each (ρ, r) pair. A matrix is said to be successfully recovered if rel. err.$\leq 10^{-3}$. The phase diagram shows the successful recovery rate for each (ρ, r) pair.

The same trick of replacing the (U, V) pair with a faster computed one is applied and produce

$$\begin{cases} U_k = Q, \mathrm{QR}\left(\left(X - S_{k-1}\right) V_{k-1}^T\right) = QR, \\ V_k = Q^T \left(X - S_{k-1}\right), \\ S_k = \mathcal{S}_\lambda\left(X - U_k V_k\right), \end{cases} \tag{17}$$

The above procedure can be performed in $3mnr_i + mr_i^2$ flops for $U \in \mathbb{R}^{m \times r_i}$ and $V \in \mathbb{R}^{r_i \times n}$.

In GreBsmo, (17) is iterated as a subroutine of GreB's greedy incremental paradigm. In particular, the updates in (17) are iterated for K times or until the object converging, then Δr rows are added into V as the new directions for decreasing the object value. In order to achieve the fastest decreasing directions, we greedily select the added Δr rows as the top Δr right singular vectors of the partial derivative

$$\frac{\partial \|X - UV - S\|_F^2}{\partial V} = X - UV - S. \tag{18}$$

We also allow to approximate row space of the singular vectors via random projections (Halko et al., 2009). The selected Δr rows maximize the magnitude of the above partial derivative and thus lead to the most rapid decreasing of the object value, a.k.a., the decomposition error. GreBsmo repeatedly increases the rank until a sufficiently small decomposition error is achieved. So the rank of the low-rank component is adaptively estimated in GreBsmo and does not relies on initial estimation.

We report the phase diagram of GreBsmo in Figure 3.3 from results on randomly generated matrix that is the sum of a low-rank part and a sparse part. The low-rank part is generated as the product of two Gaussian matrices and the sparse part has a Bernoulli model generated support set on which ± 1 values are randomly assigned. The phase transition phenomenon is in consistency with existing low-rank and sparse decomposition algorithms. It also shows that GreBsmo is able to gain accurate separation of L even if its rank is close to $0.4n$, given the sparse part has an adequately sparse support set. This is competitive to published result (Candès et al., 2009). Interestingly, the phase transition curve has a regular shape and implies a theoretical analysis to its behavior is highly possible in future studies.

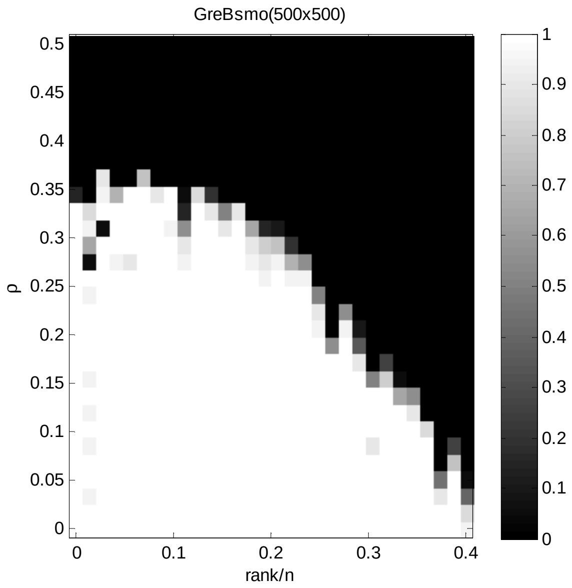

Figure 2: Phase diagram for GreBsmo on 500×500 matrices. Low-rank component is generated as $L = UV$, where entries of U and V are sampled from $\mathcal{N}(0, 1/n)$. Entries of sparse component S are sampled as 1 or -1 with probability $\rho/2$ and 0 with probability $1-\rho$. On the 30×30 grid of sparsity-rank/n plane, 20 trials are performed for each (ρ, r) pair. L is said to be successfully recovered if its rel. err.$\leq 10^{-2}$. The phase diagram shows the successful recovery rate for each (ρ, r) pair.

4 Analysis

It is not direct to analyze the theoretical guarantee of GreB due to its combination of alternating minimization and greedy forward selection. Hence, we consider analyzing its convergence behavior by leveraging the results from GECO (Shalev-Shwartz et al., 2011) analysis. This is reasonable because they share the same objective function yet different optimization variables. In particular, the risk function in GECO is $R(A) = R(A(\lambda)) = f(\lambda)$, where $A = \sum_i \lambda_i U_i V_i$. It can be seen that the variable A in GECO is able to be written as $A = UV$ without any loss of generality. Therefore, for the same selection of $R(A)$, we can compare the objective value of GECO and GreB at arbitrary step of their algorithm. This results in the following theorem.

Theorem 1. *Assume $R(A)$ is a β-smooth function according to GECO (Shalev-Shwartz et al., 2011) and $\epsilon > 0$, and $F(U,V) = R(UV)$ is the objective function of GreB. Given a rank constraint r to A and a tolerance parameter $\tau \in [\,0, 1\,)$. Let $A^* = U^*V^*$ is the solution of GreB. Then for all matrices $A = UV$ with*

$$\|UV\|_{tr}^2 \leq \frac{\epsilon(r+1)(1-\tau)^2}{2\beta} \tag{19}$$

we have $F(U^*, V^*) \leq F(U,V) + \epsilon$.

Proof. According to Lemma 3 in GECO (Shalev-Shwartz et al., 2011), let $\epsilon_i = f(\lambda^{(i)}) - f(\bar{\lambda})$, where $\lambda^{(i)}$ is the value of λ at the beginning of iteration i and $\bar{\lambda}$ fulfills $f(\lambda) > f(\bar{\lambda})$, we have

$$f(\lambda^{(i)}) - \min_{\eta} f(\lambda^{(i)} + \eta e^{u,v}) \geq \frac{\epsilon_i^2 (1-\tau)^2}{2\beta \|A\|_{tr}^2}. \tag{20}$$

At the end of iteration i, the objective value of GreB equals $R(UV)$, while GECO optimizes λ over the support of $\mathrm{span}(U) \times \mathrm{span}(V)$ (i.e., optimizes S when fixing U and V). We use the same notation $\cdot^{(i)}$ to denote the variable in iteration i. This yields

$$\begin{aligned} F(U^{(i)}, V^{(i)}) = \quad & R(U^{(i)}V^{(i)}) \geq \\ & \min_S R(U^{(i)} S V^{(i)}) = f(\lambda^{(i)}). \end{aligned} \tag{21}$$

At the beginning of iteration $i+1$, both GECO and GreB computes the direction (u, v) along which the object declines fastest. However, GECO adds both u and v to the ranges of U and V, while GreB only adds v to V and then optimizes U when fixing V. Because the range of U in GreB is optimized rather than previously fixed, we have

$$\begin{aligned} F(U^{(i+1)}, V^{(i+1)}) = \quad & \min_U F(U, [V^{(i+1)}; v]) \leq \\ & \min_\eta f(\lambda^{(i)} + \eta e^{u,v}). \end{aligned} \tag{22}$$

Plug (21) and (22) into (20), we gain a similar result:

$$F(U,V) - \min_U F(U, [V; v]) \geq \frac{\epsilon_i^2 (1-\tau)^2}{2\beta \|A\|_{tr}^2}. \tag{23}$$

Following the analysis after Lemma 3 in GECO (Shalev-Shwartz et al., 2011), we can immediately obtain the results of the theorem. □

The theorem states that GreB solution is at least close to optimum as GECO. Note when sparse S is alternatively optimized with UV in GreB scheme, such as GreBcom, the theorem can still holds. This is because after optimizing S in each iteration of GreBcom, we have $\mathcal{P}_{\Omega^C}(S + UV) = 0$, which enforces the objective function $\|M - UV - S\|_F^2$ degenerates to that of GECO, which is $\|P_\Omega(M - UV)\|_F^2$.

Table 1: Relative error and time cost of OptSpace, SVP, ADMiRA and GreBcom in matrix completion tasks of different matrix size and rank. Notations: $m(n)$-square matrix size, r-rank, ρ-sampling ratio $|\Omega|_0/mn$, rel. err.-relative error, time-CPU seconds, "-"-does not apply due to speed or divergence.

$m(n)$	r	ρ	OptSpace		SVP		ADMiRA		GreBcom	
			rel. err.	time	rel. err.	time	rel. err.	time	rel. err.	time
5000	10	0.01	2.29×10^{-2}	304	8.51×10^{-1}	86	5.13×10^{-1}	77	2.01×10^{-2}	0.73
	50	0.04	3.41×10^{-2}	1582	6.95×10^{-1}	1362	5.36×10^{-1}	358	3.06×10^{-2}	16
	100	0.08	5.64×10^{-2}	3944	7.01×10^{-1}	24519	4.84×10^{-1}	36266	2.38×10^{-3}	116
10000	10	0.01	1.34×10^{-2}	516	4.54×10^{-1}	1322	1.22×10^{-1}	442	1.55×10^{-3}	2.17
	50	0.04	1.19×10^{-2}	2192	2.35×10^{-1}	5961	2.58×10^{-2}	186591	1.40×10^{-3}	49
	100	0.08	2.64×10^{-3}	15910	-	-	9.66×10^{-2}	755082	1.20×10^{-3}	153
20000	10	0.006	7.06×10^{-2}	1928	-	-	3.04×10^{-1}	181	1.20×10^{-3}	4.06
	50	0.025	7.66×10^{-3}	11397	-	-	4.33×10^{-2}	346651	1.20×10^{-3}	113
30000	10	0.006	8.29×10^{-2}	6121	2.43×10^{-1}	2235	4.19×10^{-1}	71	1.20×10^{-3}	18

Table 2: $RMSE_{test}$/CPU seconds of OptSpace, SVP and GreBcom in matrix completion tasks on recommendation system data with different training set ratio (for MovieLens) or different number of test ratings per user (for Jester), "-"-does not apply due to speed or divergence. Size and rank information (m/n/r) of datasets: 100k(943/1682/3), 1M(6040/3952/10), 10M(69878/10677/10), J1(24983/100/10), J2(23500/100/10), J3(24938/100/10).

	OptSpace			SVP			GreBcom		
Movie	10%	30%	50%	10%	30%	50%	10%	30%	50%
100k	3.13/334s	2.82/394s	1.02/117s	1.23/5.01s	1.06/2.19s	0.97/2.35s	1.01/0.04s	0.98/0.04s	0.97/0.04s
1M	1.35/2241s	0.93/2550s	0.89/3383s	1.12/38s	0.98/41s	0.92/32s	0.96/1.15s	0.90/1.22s	0.89/1.89s
10M	0.92/9021s	-	-	0.96/588s	0.87/464s	0.84/694s	0.88/9.61s	0.86/10.13s	0.82/25.65s
Jester	2	5	10	2	5	10	2	5	10
J1	4.10/756s	4.10/732s	4.13/669s	5.31/41.26s	5.09/40.12s	4.24/32.54s	4.01/7.29s	4.06/7.88s	4.08/6.28s
J2	4.14/1058s	4.14/640s	4.16/840s	4.24/320s	4.23/22.14s	4.31/66.74s	4.12/10.23s	4.18/8.80s	4.09/10.06s
J3	4.57/340s	4.64/261s	4.51/99.31s	7.75/14.71s	7.28/13.27s	6.13/6.85s	4.91/3.15s	4.43/3.02s	4.38/1.20s

5 Applications

In this section, we show how to use the above three algorithms developed from GreB to solve low-rank approximation, matrix completion and robust PCA by justifying their performance on both artificial generated and real datasets. For simplicity, we fix the times of rank increment in GreB as 5, which implies $\Delta r = \max\{1, \lfloor rank/5 \rfloor\}$. All the experiments are performed on MATLAB.

5.1 Low-rank Approximation

The approximation accuracy and time cost of GreBske is evaluated on the task of approximating a randomly generated $10^4 \times 10^4$ matrix X, with thorough comparison to the results obtained by Lanczos algorithm for SVD and randomized SVD (Halko et al., 2009), which are two popular approximation algorithms of SVD. Each entry of the matrix is sampled from an i.i.d. standard Gaussian distribution $\mathcal{N}(0, 1)$. We uniformly select 20 values from 1 to 500 as the rank parameters. Then the associated 20 low-rank approximations are computed by the three different algorithms. Randomized SVD and GreBske of different power parameters are tested. We show the approximation error $\|\hat{X} - X\|_F / \|X\|_F$ and CPU seconds for each approximation $\hat{X}$ in the two right plots of Figure 5.1. It can been verified that Lanczos method achieves the smallest error yet along with expensive computations, while randomized SVD has the fastest speed yet largest error. GreBske has error very close to that of Lanczos method, but its computational time is comparable with that of randomized SVD. Thus it provides a good trade-off between speed and accuracy, which is highly preferred in real applications.

5.2 Matrix Completion

The performance of GreBcom in matrix completion tasks are evaluated and compared with other approaches on both artificial generated data and real data from movie (MovieLens) and joke (Jester) recommendation systems.

Numerical results on artificial data. We generate low-rank matrix $X \in \mathbb{R}^{n \times n}$ by $X = UV + Z$, wherein $U \in \mathbb{R}^{n \times r}$ and $V \in \mathbb{R}^{r \times n}$ are random matrices whose entries are sampled from i.i.d. normal distribution $\mathcal{N}(0, 1)$ and each entry of noise Z is sampled from $\mathcal{N}(0, 10^{-10})$. The observed entries are uniformly selected with probability ρ. OptSpace, SVP, ADMiRA and GreBcom are performed on large-scale matrices of $n \in [5000, 30000]$ and $r \in [10, 100]$. Their completion accuracy is measured by relative error $\|\hat{X} - X\|_F / \|X\|_F$ wherein $\hat{X}$ is the low-

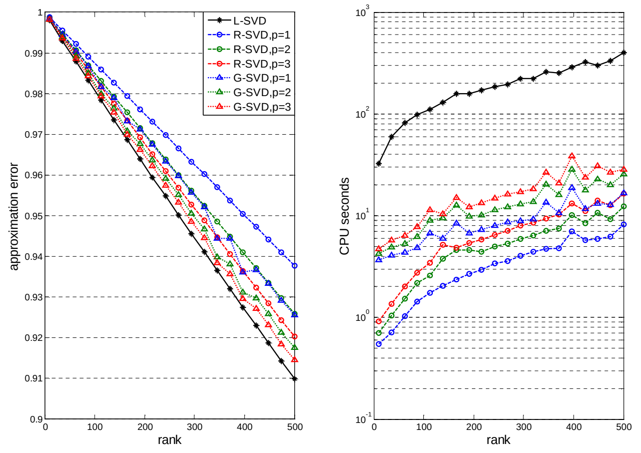

Figure 3: Low-rank approximation performed by Lanczos method (L-SVD), randomized SVD (R-SVD) and GreBske (G-SVD) on $10^4 \times 10^4$ matrix whose entries are sampled from i.i.d. normal distribution, p (K in G-SVD) is the power parameter.

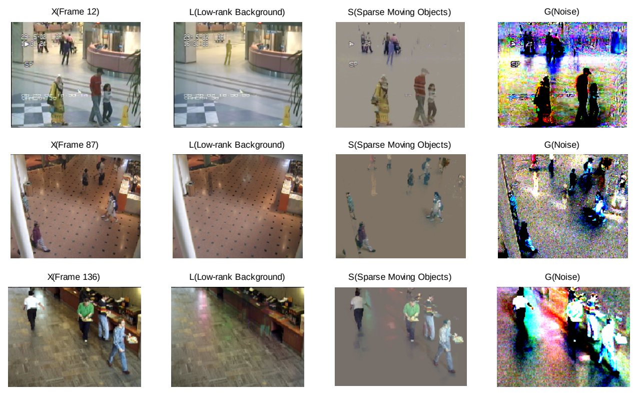

Figure 4: Background modeling of GreBsmo on three video sequences, top row: Hall, 144×176 pixels, 500 frames; middle row: ShoppingMall, 256×320 pixels, 253 frames; bottom row: Boostrap, 120×160 pixels, 500 frames.

rank recovery, while the time cost is measured by CPU seconds. The results are given in Table 1. It shows that GreBcom achieves the lowest relative error and brings substantial acceleration. The time cost of GreBcom slowly augmented with the increasing of matrix size, which indicates its promising scalability for solving large-scale problems.

Collaborative filtering. We now apply different matrix completion algorithms to real datasets MovieLens [1] and Jester [2] respectively collected from movie and joke recommendation systems. Each of the two datasets contains 3 matrices, whose rows denote users, columns denote items and the entry values are associated ratings. Ratings in MovieLens are integers between 1 and 5, while ratings in Jester have values in $[-10, 10]$. For MovieLens, we randomly select $\{10\%, 30\%, 50\%\}$ of the given observations as the training set observable to the matrix completion algorithms, while the rest are left for test set. For Jester, we randomly select $\{2, 5, 10\}$ ratings from each user as test set and treat the others as training instances. The completion accuracy is evaluated by comparing the difference between the given matrix and the completed one on the test set. In Table 2, the difference is measured by root mean square error (RMSE) of the test set such that $RMSE_{test} = \sqrt{\text{mean}(\hat{X}_{ij} - X_{ij})}, (i,j) \in$ test set, while the CPU seconds tells the required computation time. In these experiments, GreBcom exhibits evident priority in both completion accuracy and time cost over other methods. In addition, less observations are required in GreBcom to reach a sufficiently small RMSE, which support the effectiveness of GreBcom in real applications.

Table 3: Comparison of time costs in CPU seconds of PCP, GoDec and GreBsmo in low-rank and sparse matrix decomposition task on background modeling datasets.

	PCP	GoDec	GreBsmo
Hall	$87s$	$56s$	$1.13s$
ShoppingMall	$351s$	$266s$	$3.29s$
Bootstrap	$71s$	$49s$	$0.98s$

5.3 Robust PCA

For real data, three robust PCA algorithms, i.e., inexact augmented Lagrangian multiplier method for PCP, GoDec and GreBsmo are applied to separate the low-rank background and sparse moving objects in 3 video sequences [3]. Pixel values of each frame in the video are vectorized as a row vector in a matrix X, and the background modeling can be modeled as performing robust PCA on the matrix. We show the robust PCA decomposition results of one frame for each video sequence obtained by GreBsmo in the left plot of Figure 5.2. The decomposition includes a low-rank background, a sparse component containing moving objects, and a dense noise part. The time costs for all the three methods are listed in Table 3. It shows GreBsmo considerably speed up the decomposition and performs 30-100 times faster than most existing algorithms.

Acknowledgements

We would like to thank all the anonymous reviewers for their constructive comments on improving this paper. This work is supported by Australian Research Council Discovery Project with number ARC DP-120103730.

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

[2]http://www.ieor.berkeley.edu/~goldberg/jester-data/

[3]http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html

References

L. Balzano, R. Nowak, and B. Recht. Online identification and tracking of subspaces from highly incomplete information. `arXiv:1006.4046v1`, 2010.

C. Boutsidis, M. W. Mahoney, and P. Drineas. An improved approximation algorithm for the column subset selection problem. In *SODA*, pages 968–977, 2009.

J. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9:717–772, 2008.

E. J. Candès and T. Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Trans on Information Theory*, 52(12):5406–5425, 2006.

E. J. Candès and T. Tao. The power of convex relaxation: Near-optimal matrix completion. `arXiv: 0903.1476`, 2009.

E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM*, 2009.

V. Chandrasekaran, S. Sanghavi, P. A Parrilo, and A. S. Willsky. Rank-sparsity incoherence for matrix decomposition. `arXiv:0906.2220`, 2009.

J. Chen, J. Liu, and J. Ye. Learning incoherent sparse and low-rank patterns from multiple tasks. In *ACM SIGKDD*, 2010.

K. L. Clarkson and D. P. Woodruff. Numerical linear algebra in the streaming model. In *STOC*, 2009.

D. L. Donoho. Compressed sensing. *IEEE Trans on Information Theory*, 52(4):1289–1306, 2006.

P. Drineas, R. Kannan, and M. W. Mahoney. Fast monte carlo algorithms for matrices iii: Computing a compressed approximate matrix decomposition. *SIAM Journal on Computing*, 36(1):184–206, 2006.

M. Fazel, E. J. Candès, B. Recht, and P. Parrilo. Compressed sensing and robust recovery of low rank matrices. In *Asilomar*, 2008.

M. Fornasier, H. Rauhut, and R. Ward. Low-rank matrix recovery via iteratively reweighted least squares minimization. *SIAM Journal on Optimization*, 21(4):1614–1640, 2011.

R. Foygel and N. Srebro. Concentration-based guarantees for low-rank matrix reconstruction. In *COLT*, 2011.

N. Halko, P. G. Martinsson, and J. A. Tropp. Finding structure with randomness: Stochastic algorithms for constructing approximate matrix decompositions. `arXiv: 0909.4061`, 2009.

P. Jain, R. Meka, and I. S. Dhillon. Guaranteed rank minimization via singular value projection. In *NIPS*, 2010.

S. Ji and J. Ye. An accelerated gradient method for trace norm minimization. In *ICML*, 2009.

R. Keshavan and S. Oh. Optspace: A gradient descent algorithm on grassman manifold for matrix completion. *Submitted to IEEE Trans on Signal Processing*, 2009.

J. Lee, B. Recht, R. Salakhutdinov, and N. Srebro. Practical large-scale optimization for max-norm regularization. In *NIPS*, 2010.

K. Lee and Y. Bresler. Admira: atomic decomposition for minimum rank approximation. *IEEE Trans on Information Theory*, 56(9):4402–4416, 2010.

S. Ma, D. Goldfarb, and L. Chen. Fixed point and bregman iterative methods for matrix rank minimization. *Mathematical Programming*, 128(1-2):321–353, 2011.

L. Mackey, A. Talwalkar, and M. I. Jordan. Divide-and-conquer matrix factorization. In *NIPS*, 2011.

Nam H. Nguyen, Thong T. Do, and Trac D. Tran. A fast and efficient algorithm for low-rank approximation of a matrix. In *STOC*, pages 215–224, 2009.

S. Roweis. Em algorithms for pca and spca. In *NIPS*, 1998.

S. Shalev-Shwartz, A. Gonen, and O. Shamir. Large-scale convex minimization with a low-rank constraint. In *ICML*, 2011.

N. Srebro, J. Rennie, and T. Jaakkola. Maximum-margin matrix factorization. In *NIPS*, 2005.

Z. Wen, W. Yin, and Y. Zhang. Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm. *Mathematical Programming Computation*, 4(4):333–361, 2012.

L. Xiong, X. Chen, and J. Schneider. Direct robust matrix factorization for anomaly detection. In *ICDM*, 2010.

J. Ye. Generalized low rank approximations of matrices. *Machine Learning*, 61(1):167–191, 2005.

D. Zachariah, M. Sundin, M. Jansson, and S. Chatterjee. Alternating least-squares for low-rank matrix reconstruction. *IEEE Signal Processing Letters*, 19(4):231–234, 2012.

T. Zhou and D. Tao. Godec: Randomized low-rank & sparse matrix decomposition in noisy case. In *ICML*, 2011.

T. Zhou and D. Tao. Bilateral random projections. In *ISIT*, 2012a.

T. Zhou and D. Tao. Multi-label subspace ensemble. In *AISTATS*, 2012b.

Z. Zhou, X. Li, J. Wright, E. J. Candès, and Y. Ma. Stable principal component pursuit. In *ISIT*, 2010.