ALTERNATING MINIMIZATION TECHNIQUES FOR THE EFFICIENT RECOVERY OF A SPARSELY CORRUPTED LOW-RANK MATRIX

Silvia Gandy and Isao Yamada

Tokyo Institute of Technology
Department of Communications and Integrated Systems
Meguro-ku, Ookayama 2-12-1-S3-60
152-8550 Tokyo, Japan

ABSTRACT
We address the problem of recovering a low-rank matrix that has a small fraction of its entries arbitrarily corrupted. This problem is recently attracting attention as nontrivial extension of the classical PCA (principal component analysis) problem with applications in image processing and model/system identification. It was shown that the problem can be solved via a convex optimization formulation when certain conditions hold. Several algorithms were proposed in the sequel, including interior-point methods, iterative thresholding and accelerated proximal gradients. Based on algorithms from rank minimization and sparse vector recovery, we propose a computationally efficient greedy algorithm that scales better to large problem sizes than existing algorithms.

Index Terms— PCA, rank minimization, nuclear norm minimization, sparse error, greedy algorithms

1. INTRODUCTION
In high-dimensional data analysis, the data can often be modeled as lying in (or close to) a low-dimensional subspace. Under this assumption an approximate representation can be recovered via a principal component analysis (PCA). This problem of finding a low-rank approximation to the given data appears in a number of applications, including image processing, bioinformatic data, system identification, etc.

The classical PCA problem is modelled as follows:

\[
\min_{(X,E)} \|E\|_F, \text{ s.t. } \text{rank}(X) \leq r, \quad D = X + E, \quad (1)
\]

where \( \|\cdot\|_F \) denotes the Frobenius norm, and \( D, X, \text{and } E \in \mathbb{R}^{n \times n} \). (The restriction on square matrices is only chosen to simplify the notations.) By the Eckart-Young Theorem, the minimizer of this problem is given by truncating the singular value decomposition (SVD) of \( D \), retaining only the contribution of its \( r \) largest singular values. If \( D = X_0 + E_0 \) originates from a low-rank matrix \( X_0 \) (\( \text{rank}(X_0) \leq r \)) and \( E_0 \) contains low-power gaussian noise, then the truncated SVD will recover a matrix \( \hat{X} \approx X_0 \). However, if \( E_0 \) represents high-power noise, the SVD will return a matrix that is largely deviating from \( X_0 \), even if \( E_0 \) only affects a small fraction of the entries of \( X_0 \). In other words, if \( E_0 \) models a sparse error, representing a small number of largely corrupted entries of \( X_0 \), the SVD fails to recover \( X_0 \).

Recently, [1] and [2] showed independently from each other that under certain assumptions (on \( X_0 \) and \( E_0 \)) one can exactly recover the low-rank matrix \( X_0 \) from \( D = X_0 + E_0 \), where \( E_0 \) is a sparse matrix, by solving the following convex optimization problem:

\[
\min_{(X,E)} \|X\|_* + \lambda \|E\|_1, \quad \text{s.t. } D = X + E. \quad (2)
\]

We will call problem (2) the Robust Principal Component Analysis (RPCA) problem, as it was named in [1]. This problem is a convex relaxation of the problem:

\[
\min_{(X,E)} \text{rank}(X) + \lambda \|E\|_0, \quad \text{s.t. } D = X + E. \quad (3)
\]

(Notation: \( \|X\|_* = \sum_{i=1}^r \sigma_i(X) \) denotes the nuclear norm, i.e. the sum of the singular values of \( X \). The \( l_0 \)-norm of \( E \), \( \|E\|_0 \), counts the number of nonzero entries of \( E \).)

Note, that in a general setting, the minimization problem will not recover \( (X_0, E_0) \). Consider the matrix \( D = e_1 e_1^* \) that contains zeros in all entries except \( D_{1,1} = 1 \). As this matrix is both sparse and low-rank, there is no hope of recovering the “correct” pair \( (X_0, E_0) \).

The assumptions on \( (X_0, E_0) \) are given by the following two definitions, [1]:

We say that \( X_0 \in \mathbb{R}^{n \times n} \) originates from the random orthogonal model of rank \( r \) if its left and right singular spaces are chosen uniformly at random from all \( n \times r \) matrices with orthonormal columns. The singular values are arbitrary.
An error matrix $E_0$ is said to be drawn from the Bernoulli sign and support model with parameter $\rho_s$ if the entries of $\text{sign}(E_0)$ are independently distributed, each taking on value 0 with probability $1 - \rho_s$, and $\pm 1$ with probability $\frac{\rho_s}{2}$ each. In this model, the magnitude of the nonzero entries of $E_0$ can be arbitrary.

If $(X_0, E_0)$ is drawn from these models, then it was shown (Theorem 2.4 in [1]) that with very high probability the convex minimization problem (2) has $(X_0, E_0)$ as unique minimizer ($\lambda = \frac{1}{4}$).

Recently, a number of algorithms were proposed to solve the convex minimization problem (2) or to give near solutions to it. These include interior points methods [2], iterative thresholding [1] and proximal gradient algorithms [3].

Problem (2) can easily be reformulated as a semidefinite problem and then be solved with off-the-shelf solvers that use interior point methods. Interior point methods have superior convergence guarantees, however, they scale badly for large matrices, having a complexity of $O(n^6)$ to solve the associated Newton system. Thus, this approach is limited to matrices up to about $100 \times 100$. In [1], an iterative thresholding algorithm was introduced to solve a variant of (2), adding a term containing the norm of $X$ and $E$ to the cost function. This enabled them to solve the problem for matrices of size $800 \times 800$ in several hours on a normal PC. The most recent algorithms that were proposed for this setting are proximal gradient algorithms [3], where the ideas of FISTA (fast iterative soft thresholding algorithm) were applied to the RPCA problem, also slightly modifying the cost function. In [3], proximal gradient algorithms were shown to solve (2) for data sizes of $2000 \times 2000$ in some hours.

We were inspired by recent work on greedy algorithms for compressed sensing and low-rank matrix recovery to devise a greedy algorithm for the problem (2), as it contains both a $l_1$-minimization part and a nuclear norm minimization part. This algorithm will allow us to solve the problem for a $2000 \times 2000$ matrix in less then 2 minutes on a desktop PC.

In the setting of $l_1$-minimization, the greedy algorithm CoSaMP [4] solves the $s$-sparse vector approximation problem
\[
\min_{x \in \mathbb{R}^n} \|Ax - b\|_2, \quad \text{s.t.} \quad \|x\|_0 \leq s.
\]
Linear convergence to the sparsest vector lying close to the affine subspace defined by $Ax = b$ can be guaranteed if the linear map $A : \mathbb{R}^n \rightarrow \mathbb{R}^m$ has some special properties [4] ($A$ needs to fulfill the restricted isometry property with a small constant $\delta$).

ADMIRA (Atomic Decomposition for Minimum Rank Approximation, [5]) solves the rank-$r$ approximation problem:
\[
\min_{X \in \mathbb{R}^{n \times n}} \|A(X) - b\|_2, \quad \text{s.t.} \quad \text{rank}(X) \leq r.
\]
As in the vector case, properties of the linear map $A : \mathbb{R}^{n \times n} \rightarrow \mathbb{R}^m$ can ensure linear convergence of the algorithm to the best low-rank solution to $A(X) \approx b$ up to noise level. For example if the matrix representation of the linear map $A$ contains Gaussian distributed entries, then the required properties hold with high probability [6]. However, the special case when $A$ "samples" a subset of the entries of the matrix, as appearing in the matrix completion problem [7], is not covered in the analysis of the convergence rate [5].

\section{2. Algorithm}

The algorithms CoSaMP and ADMIRA find the sparsest/minimal rank solution by solving the problem of finding the best $s$-sparse/rank-$r$ approximation w.r.t. the error on the linear constraints.

We will use the same approach for problem (3). Given a data matrix $D$, we fix a target rank $r$ and a target sparsity $s$ and solve:
\[
\arg\min_{(X,E)} \|D - E - X\|_F \text{ s.t. rank}(X) \leq r, \|E\|_0 \leq s \quad (4)
\]

The idea was to alternatively use CoSaMP and ADMIRA to update $X^k$ and $E^k$ respectively. In the setting of problem (4), the task is to split a sum $D$ into its parts $(X_0, E_0)$, without any linear measurements being involved. Therefore, direct application of ADMIRA and CoSaMP simplifies to the following algorithm, which we name Atomic Decomposition Alternating Least Squares (AD_ALS):

\textbf{Algorithm (AD_ALS)}:

Initialization: $X^0 \leftarrow 0, E^0 \leftarrow 0, \Psi^0 \leftarrow \emptyset$, $k \leftarrow 1$

Iteration steps ($k$: iteration index)

1. Update $E^k$:
\[
E^k \leftarrow \arg\min_{E} \{ \|D - X^{k-1} - E\|_F : |\text{supp}(E)| \leq s \}
\]

2. Update $X^k$:
\[
\Psi^k \leftarrow \bigcup_{\Psi \subset \mathcal{O}} \arg\max_{\Psi \subset \mathcal{O}} \{ \|P_{\Psi} (D - E^k - X^{k-1})\|_F : |\Psi| \leq 2r \}
\]
\[
X^k \leftarrow \arg\max_{X} \{ \|D - E^k - X\|_F : X \in \text{span}(\Psi') \}
\]
\[
\Psi^k \leftarrow \bigcup_{\Psi' \subset \mathcal{O}} \{ \|P_{\Psi'} X\|_F : |\Psi'| \leq r \}
\]

The update of $E^k$ simply computes the best $s$-sparse approximation to the residual $D - X^{k-1}$. This is all that remains from the CoSaMP algorithm. The step to compute $X^k$ is a little bit more involved. We use the same notation as in [5]. $\Psi$ stands for a set of rank-one matrices, the so-called atoms. $\mathcal{O}$ denotes the set of atoms of $\mathbb{R}^{n \times n}$ and $P_{\Psi}(Z)$ denotes the orthogonal projection of a matrix $Z$ onto the space spanned by the elements of the set $\Psi$.

The first step in the update of $X^k$ is the calculation of a set of $2r$ (rank-one) matrices that represent the singular spaces of the best rank-$2r$-approximation of the new residual $D - X^{k-1} - E^k$. Then, the best approximation to $D - E^k$ within the space spanned by the elements in $\Psi'$ is calculated. Thus, $X'$ is a linear combination of the $2r$ atoms computed before and the $r$ rank-one matrices which are derived from the singular vectors of $X^{k-1}$. (5): $\Psi^k$ contains the $r$ matrices $\psi_i^k$.
which are the products of the right and left singular vectors $u_i$ and $v_i$ of $X^k$: $\psi^k_i = u_i v_i^T$, $\Psi^k = \{\psi^k_i \}$.

**Algorithm (ALS):**
We also introduce a further simplified version of ADALS, the alternating least squares (ALS) algorithm. ALS updates $E^k$ and $X^k$ via alternately minimizing the residual. Therefore, step 1 (update of $E^k$) stays as stated above. Step 2 is changed to:

2'. Update $X^k$:

$$X^k = \text{argmin}_X \left\{ ||D - E^k - X||_F : \text{rank}(X) \leq r \right\},$$

whose minimizer can be computed via a partial SVD of the matrix $D - E^k$.

3. ANALYSIS OF THE ALGORITHMS

We compared ALS and ADALS on several runs, deducing that the number of necessary iterations is slightly higher forALS, but the overall behaviour of the ADMIRA-inspired algorithm and alternating least squares is similar, as expected. ALS is computationally less expensive, as it avoids computing the projection of $D - E^k$ onto the 3$r$-dimensional subspace, which is an additional least squares problem.

The algorithms ALS and ADALS depend strongly on a good choice of the target rank $r$ and the target sparsity $s$. If no target rank $r$ and target sparsity $s$ is given, then it is reasonable to run the algorithm for increasing pairs $(r,s)$ and to choose the minimal pair $(\bar{X}(r,s), \bar{E}(r,s))$ that achieves a small enough residual $||D - \bar{X} - \bar{E}||_F$.

The algorithms are empirically stable w.r.t. additional small gaussian noise affecting $D$, as the problem formulation (4) is robust to additional noise contained within $D$. We conducted experiments using a perturbed data matrix $D$,

$$D = X_0 + E_0 + Z,$$

where $Z$ is a low-power gaussian matrix. The algorithms recovered $(X_0, E_0)$ up to noise level as long as the pair $(X_0, E_0)$ was low-rank and sparse enough.

It is not an easy task to give any theoretical convergence guarantees. The algorithms CoSaMP and ADMIRA deduce their convergence rates from properties of the linear map $A$ (resp. $A$) when applied to sparse vectors/low-rank matrices. In the current setting, no such tool is available.

The error is guaranteed to decrease in each step, but the algorithm can become stationary if a new iterate of the sequence $X^k$ decreases the residual such that $E^{k+1} = E^k$. Then, the algorithm stays stationary with this pair $(X^k, E^k)$. Yet, we did not observe this in the numerical experiments.

4. NUMERICAL EXPERIMENTS

We randomly generated an input pair $(X_0, E_0)$ as follows: $X_0 := X_L X_R^T$, where $X_L, X_R \in \mathbb{R}^{n \times r}$ with the entries of $X_L, X_R$ being i.i.d. gaussian variable with mean 0 and variance 1. We chose the support set of $E_0$ uniformly at random from all support sets of size $\rho n^2$. The nonzero entries are independently drawn from a uniform distribution on $[-500, 500]$.

This is the same simulation setting as in [3], in order to be able to roughly compare the results. The numerical experiments in [3] showed that the proximal gradient algorithms outperform iterative thresholding. Therefore we will limit our comparison to the proximal gradient algorithm APG, [3].

The tests were run on an Intel (R) Core(TM) 2 Quad CPU (four cores) with 3.0 GHz. In [3] a Mac Pro computer having eight cores and 2.8 GHz was used. Thus, the machine specifications are rather close, so we expect comparable computation times.

Figure 1 shows the recovery performance of ADALS. For small values of the rank $r$ of $X_0$ and a very sparse matrix $E_0$, the figure shows perfect recovery in all tries. This result shows the same type of transition from recovery in all tries to non-recovery as is to be expected for the direct minimization of the convex problem formulation (2). In [1], a similar figure was given for the iterative thresholding algorithm.

Figure 2 was obtained from the same setting as Figure 1. We additionally introduced a mismatch between the correct rank/sparsity of $(X_0, E_0)$ and the target rank/sparsity of the algorithm, to see the dependency on the parameter values.

Next, we compared the computation time of our algorithm (ADALS / ALS) and APG (accelerated proximal gradient, [3]) in Table 1. ADALS is outperforming APG already in the case of small problem sizes ($n = 100, \ldots, 800$). The times for ALS are not shown, but ALS converges even faster. As stopping criterion we used that the relative error falls below $10^{-5}$. Table 2 shows the comparison of the computation time for a large-scale problem ($n = 2000$). These experiments demonstrate the effectiveness of the proposed methods.

Due to a non-optimized ad-hoc implementation that we used to test ADALS, we ran into memory problems for problem sizes of $n = 2000$. Therefore, we only compare APG and ALS. ALS is by far faster than APG, recovering the input pair $(X_0, E_0)$ in less than 2 minutes (compared to some hours for APG).

Implementation: We used Lanczos iterations to calculate the partial SVDs (calculating only the largest singular values and their singular vectors), in order to avoid the computational burden of a complete singular value decomposition.

5. DISCUSSION

We presented two computationally efficient greedy algorithms (ADALS and ALS) for the robust principal component analysis problem.

The main point of interest of this work is to point out that ADALS is the analog of CoSaMP and ADMIRA for the RPCA problem. ADALS updates $E^k$ and $X^k$ alternatively.
Table 1. Comparison of the runtime for different matrix sizes $n$. APG (Accelerated Projected Gradient) is significantly slower than AD_ALS.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$\frac{|X-X_0|_F}{|X_0|_F}$</th>
<th>time (s)</th>
<th>$\frac{|X-X_0|_F}{|X_0|_F}$</th>
<th>time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>$3.4 \times 10^{-5}$</td>
<td>3.1</td>
<td>$9.9 \times 10^{-6}$</td>
<td>0.35</td>
</tr>
<tr>
<td>200</td>
<td>$2.2 \times 10^{-5}$</td>
<td>16</td>
<td>$4.9 \times 10^{-6}$</td>
<td>1.1</td>
</tr>
<tr>
<td>400</td>
<td>$1.6 \times 10^{-5}$</td>
<td>111</td>
<td>$7.1 \times 10^{-6}$</td>
<td>8.2</td>
</tr>
<tr>
<td>800</td>
<td>$1.1 \times 10^{-5}$</td>
<td>766</td>
<td>$4.5 \times 10^{-6}$</td>
<td>68.9</td>
</tr>
</tbody>
</table>

Table 2. Large-size example: Alternating least squares recovers $(X_0, E_0)$ of sizes $2000 \times 2000$ matrix in less than 2 minutes, even if the rank is $0.1 \times 2000 = 200$ and $400000$ entries are corrupted. The accelerated proximal gradient algorithm needs some hours in the same setting.

<table>
<thead>
<tr>
<th>rank$(A_0)/n$</th>
<th>$|E_0|_0/n^2$</th>
<th>$\frac{|X-X_0|_F}{|X_0|_F}$</th>
<th>APG [3] time (s)</th>
<th>ALS time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.05</td>
<td>0.05</td>
<td>14 300 s</td>
<td>29.9 s</td>
<td></td>
</tr>
<tr>
<td>0.05</td>
<td>0.1</td>
<td>14 700 s</td>
<td>67.8 s</td>
<td></td>
</tr>
<tr>
<td>0.1</td>
<td>0.1</td>
<td>14 300 s</td>
<td>95.5 s</td>
<td></td>
</tr>
</tbody>
</table>

6. REFERENCES


