An Alternating Direction Algorithm for Matrix Completion with Nonnegative Factors

Yangyang Xu[1], Wotao Yin[1], Zaiwen Wen[2], Yin Zhang[1]

[1]Department of Computational and Applied Mathematics, Rice University
[2]Department of Mathematics and Institute of Natural Sciences, Shanghai Jiaotong University

Abstract. This paper introduces a novel algorithm for the nonnegative matrix factorization and completion problem, which aims to find nonnegative matrices X and Y from a subset of entries of a nonnegative matrix M so that XY approximates M. This problem is closely related to the two existing problems: nonnegative matrix factorization and low-rank matrix completion, in the sense that it kills two birds with one stone. As it takes advantages of both nonnegativity and low rank, its results can be superior than those of the two problems alone. Our algorithm is applied to minimizing a non-convex constrained least-squares formulation and is based on the classic alternating direction augmented Lagrangian method. Preliminary convergence properties and numerical simulation results are presented. Compared to a recent algorithm for nonnegative random matrix factorization, the proposed algorithm yields comparable factorization through accessing only half of the matrix entries. On tasks of recovering incomplete grayscale and hyperspectral images, the results of the proposed algorithm have overall better qualities than those of two recent algorithms for matrix completion.

Keywords. nonnegative matrix factorization, matrix completion, alternating direction methd, hyperspectral unmixing

MSC. 15A83, 65F30, 90C26, 90C90, 94A08

1 Introduction

This paper introduces an algorithm for the following problem:

Definition 1 (Nonnegative matrix factorization / completion (NMFC))**.** *Given samples $M_{i,j}$, $(i,j) \in \Omega \subset \{1,\dots,m\} \times \{1,\dots,n\}$, of a* nonnegative rank-r *matrix $M \in \mathbb{R}^{m\times n}$, find* nonnegative *matrices $X \in \mathbb{R}^{m\times q}$ and $Y \in \mathbb{R}^{q\times n}$ such that $\|M - XY\|_F$ is minimized.*

Note that q is not necessarily set to equal r. Firstly, not all rank-r nonnegative matrices have nonnegative factors of size r. For some of them, the

available size of nonnegative factors is strictly greater than r. Secondly, when M is approximately low-rank, i.e. the singular values of M have a fast-decaying distribution, one often sets q to be the estimated rank or the number of significant singular values. This resulting problem can be called *approximate NMFC*. In general, depending on data and applications, q can be either equal, less than, or greater than r.

NMFC is a combination of nonnegative matrix factorization (NMF) — which finds nonnegative factors of a nonnegative matrix given all of its entries — and low-rank matrix completion (LRMC) — which recovers M from an incomplete set of its entries without assuming nonnegativity. Mathematically, given a matrix $M \in \mathbb{R}^{m\times n}$ and $q > 0$, we present the three problems with the following models

$$\text{NMFC:} \quad \min_{X,Y} \left\{ \|\mathscr{P}_\Omega(XY - M)\|_F^2 : \begin{array}{l} X \in \mathbb{R}^{m\times q}, Y \in \mathbb{R}^{q\times n}, \\ X_{ij} \geq 0, Y_{ij} \geq 0, \forall\ i, j \end{array} \right\}, \tag{1}$$

$$\text{NMF:} \quad \min_{X,Y} \left\{ \|XY - M\|_F^2 : \begin{array}{l} X \in \mathbb{R}^{m\times q}, Y \in \mathbb{R}^{q\times n}, \\ X_{ij} \geq 0, Y_{ij} \geq 0, \forall\ i, j \end{array} \right\}, \tag{2}$$

$$\text{LRMC:} \quad \min_{Z} \left\{ \text{rank}(Z) : \begin{array}{l} Z \in \mathbb{R}^{m\times n}, \\ \mathscr{P}_\Omega(Z - M) = 0 \end{array} \right\}, \tag{3}$$

where Ω indexes the known entries of M and $\mathscr{P}_\Omega(A)$ returns a copy of A that zeros out the entries not in Ω. Note that each of the three problems has other models. Examples include weighted least-squares for NMF and NMFC and nuclear-norm minimization for LRMC. While (1) and (2) return XY up to a fixed rank q, (3) seeks for a *least-rank* recovery Z. It is well known that models (1)–(3) are non-convex and generally difficult to solve. A recent advance for (3) is that if M is low-rank and the samples Ω satisfy the so-called incoherence property and are sufficiently large, then a convex problem based on nuclear norm minimization can exactly recover M (see the pioneering work [9], as well as recent results [25, 5, 30, 6]).

We are interested in NMFC since it complements NMF and LRMC. NMF has been widely used in data mining such as text mining, dimension reduction and clustering, as well as spectral data analysis. It started to appear in [23, 21, 22] and has become popular since the publication of [17] in 1999. More information on NMF can be found in the survey paper [1], as well as books [7, 8]. Unlike NMF, NMFC assumes that the underlying matrix is incompletely sampled; hence, it leads to saving of sampling time and storage (for data such as images) and has broader applicability. On the other hand, LRMC has recently found a large number of applications including *collaborative filtering*, which is used by Netflix to infer individual's preference from an incomplete set of user preferences [12], *global positioning*, which discovers the positions of nodes in a network from incomplete pair-wise distances [3], *system identification and order reduction*, which recovers or reduces the dimension of the state vectors of a linear time-invariant state-space model [19], as well as the *background subtraction* and *structure-from-motion* problems in computer vision. A rank-q matrix M can

be written as $M = XY$ for matrices X with q columns and Y with q rows. When X and Y are known to be nonnegative *a priori*, empirical evidence given in Section 3 shows that imposing nonnegativity on the factors improves the recovery quality. In particular, in certain applications such as hyperspectral unmixing, the factors are nonnegative due to their physical nature, so these applications will benefit from NMFC. To summarize, NMFC combines NMF and LRMC, and NMFC is useful when the underlying matrix has both low rank and nonnegative factors.

1.1 Related Algorithms

There are two algorithms that have been widely used for NMF: the alternating least squares (ALS) in [23] and multiplicative updating (Mult) in [18]. The former algorithm alternatively updates factor matrices X and Y to reduce the least-squares cost $\|XY - M\|_F^2$. The closed-form updates are given as

$$\begin{aligned} X_{\text{new}} &\leftarrow \max\{0, MY^\top (YY^\top)^\dagger\}, \\ Y_{\text{new}} &\leftarrow \max\{0, (X^\top X)^\dagger X^\top M\}, \end{aligned}$$

where $\max\{\cdot,\cdot\}$ is applied component-wise and $\dagger$ denotes pseudo-inverse. The algorithm Mult has much cheaper multiplicative updates

$$\begin{aligned} (X_{\text{new}})_{ij} &\leftarrow X_{ij}(MY^\top)_{ij}/(XYY^\top + \epsilon)_{ij}, \ \forall\, i,j, \\ (Y_{\text{new}})_{ij} &\leftarrow Y_{ij}(X^\top M)_{ij}/(X^\top XY + \epsilon)_{ij}, \ \forall\, i,j, \end{aligned}$$

which do not involve matrix inversion. Starting from a nonnegative initial matrix Y, X and Y remain nonnegative during the iterations of Mult. The algorithm presented in this paper also applies to NMF if a complete sample set Ω is used. The resulting algorithm, which has been studied in paper [34], is simpler and compares favorably with ALS and Mult in terms of both speed and solution quality. In fact, the proposed algorithm in this paper extends the work in [34], and both algorithms are based on the algorithm of alternating direction method of multipliers (ADM) [11, 10, 2, 27, 32, 29]. Likewise, we can extend the algorithms ALS and Mult to solving NMFC. Extending ALS is as straightforward as adopting the least-square cost $\|\mathscr{P}_\Omega(XY - M)\|_F^2$ and deriving the corresponding updates. One simple approach to extend Mult is to replace M by $\tilde{M} \in \mathbb{R}^{m\times n}$, defined component-wise by $\tilde{M}_{ij} = M_{ij}\mathbf{1}_{(i,j)\in\Omega}$, i.e. $\tilde{M}$ is a copy of M with the unsampled entries set to 0. Drawing conclusions based on the comparative results in [34], we believe that ADM based methods deliver higher-quality solutions in shorter times.

There are also several algorithms for LRMC. Since LRMC can complete a matrix and return factors that happen to be (approximately) nonnegative, we shall briefly review a few well-known LRMC algorithms and compare them to the proposed algorithm. Singular value thresholding (SVT) [4] and fixed-point shrinkage (FPCA) [20] are two well-known algorithms. SVT applies the

linearized Bremgan iterations [33] to the unconstrained nuclear-norm model of LRMC:

$$\min \lambda\|Z\|_* + (1/2)\|\mathscr{P}_\Omega(Z-M)\|_F^2. \tag{4}$$

FPCA solves the same model using iterations based on an iterative shrinkage-thresholding algorithm [15]. Furthermore, classic alternating direction augmented Lagrangian methods have been applied to solving (4) or its variant with constraints $\mathscr{P}_\Omega(Z-M)=0$ in [13, 31]. The algorithm LMaFit [28] uses a different model:

$$\min_{X,Y,Z}\{\|XY-Z\|_F : \mathscr{P}_\Omega(Z-M)=0\}. \tag{5}$$

The model is solved by a nonlinear successive over-relaxation algorithm [14]. In section 3, we compare the proposed algorithm to FPCA and LMaFit and demonstrate the benefits of taking advantages of factor nonnegativity.

1.2 Organization

The rest of this paper is organized as follows. Section 2 reviews the ADM algorithm and presents an ADM-based algorithm for NMFC. A preliminary convergence result of this algorithm is given in Section 2.3. Section 3 presents the results of numerical simulations, which perform tasks such as decomposing nonnegative matrices, compressing grayscale images, as well as recovering three-dimensional hyperspectral cubes from incomplete samples. Finally, Section 4 concludes this paper.

2 Algorithm and Convergence

2.1 Background: the ADM approach

In a finite-dimensional setting, the classic alternating direction method (ADM) solves structured convex programs in the form of

$$\min_{x\in\mathscr{X},y\in\mathscr{Y}} f(x)+g(y), \text{s.t. } Ax+By=c, \tag{6}$$

where f and g are convex functions defined on closed subsets $\mathscr{X}$ and $\mathscr{Y}$ of a finite-dimensional space, respectively, and A, B and c are matrices and vector of appropriate sizes. The augmented Lagrangian of (6) is

$$\mathscr{L}_A(x,y,\lambda) = f(x)+g(y)+\lambda^T(Ax+By-c)+\frac{\beta}{2}\|Ax+By-c\|_2^2,$$

where λ is a Lagrangian multiplier vector and $\beta>0$ is a penalty parameter.

The classic alternating direction method is an extension of the augmented Lagrangian multiplier method [16, 24, 26]. It performs minimization with respect to x and y alternatively, followed by the update of λ; that is, at iteration

k,

$$x^{k+1} \leftarrow \arg\min_{x\in\mathscr{X}} \mathscr{L}_A(x, y^k, \lambda^k), \tag{7a}$$

$$y^{k+1} \leftarrow \arg\min_{y\in\mathscr{Y}} \mathscr{L}_A(x^{k+1}, y, \lambda^k), \tag{7b}$$

$$\lambda^{k+1} \leftarrow \lambda^k + \gamma\beta(Ax^{k+1} + By^{k+1} - c), \tag{7c}$$

where $\gamma \in (0, 1.618)$ is a step length. While (7a) only involves $f(x)$ in the objective and (7b) only involves $g(y)$, the classic augmented Lagrangian method requires a minimization of $\mathscr{L}_A(x, y, \lambda^k)$ with respect to x and y jointly, i.e., replacing (7a) and (7b) by

$$(x^{k+1}, y^{k+1}) \leftarrow \arg\min_{x\in\mathscr{X}, y\in\mathscr{Y}} \mathscr{L}_A(x, y, \lambda^k).$$

As the minimization couples $f(x)$ and $g(y)$, it can be much more difficult than (7a) and (7b).

2.2 Main Algorithm

To facilitate an efficient use of ADM, we consider an equivalent form of (1):

$$\begin{array}{ll} \min_{U,V,X,Y,Z} & \frac{1}{2}\|XY - Z\|_F^2 \\ \text{s.t.} & X = U, Y = V, \\ & U \geq 0, V \geq 0, \\ & \mathscr{P}_\Omega(Z - M) = 0, \end{array} \tag{8}$$

where $X, U \in \mathbb{R}^{m\times q}$ and $Y, V \in \mathbb{R}^{q\times n}$. The augmented Lagrangian of (8) is

$$\begin{aligned} \mathscr{L}_A(X, Y, Z, U, V, \Lambda, \Pi) &= \frac{1}{2}\|XY - Z\|_F^2 + \Lambda \bullet (X - U) \\ &\quad + \Pi \bullet (Y - V) + \frac{\alpha}{2}\|X - U\|_F^2 + \frac{\beta}{2}\|Y - V\|_F^2, \end{aligned}$$

where $\Lambda \in \mathbb{R}^{m\times q}$, $\Pi \in \mathbb{R}^{q\times n}$ are Lagrangian multipliers, $\alpha, \beta > 0$ are penalty parameters, and $A \bullet B := \sum_{i,j} a_{ij}b_{ij}$ for matrices A and B of the same size. We deliberately leave $\mathscr{P}_\Omega(Z - M) = 0$ in the constraints instead of relaxing them, so only those entries of Z not in Ω are free variables.

The alternating direction method for (8) is derived by successively minimizing $\mathscr{L}_A$ with respect to X, Y, Z, U, V, one at a time while fixing others at their most recent values, i.e.,

$$\begin{aligned} X_{k+1} &= \arg\min \mathscr{L}_A(X, Y_k, Z_k, U_k, V_k, \Lambda_k, \Pi_k), \\ Y_{k+1} &= \arg\min \mathscr{L}_A(X_{k+1}, Y, Z_k, U_k, V_k, \Lambda_k, \Pi_k), \\ Z_{k+1} &= \arg\min_{\mathscr{P}_\Omega(Z-M)=0} \mathscr{L}_A(X_{k+1}, Y_{k+1}, Z, U_k, V_k, \Lambda_k, \Pi_k), \\ U_{k+1} &= \arg\min_{U\geq 0} \mathscr{L}_A(X_{k+1}, Y_{k+1}, Z_{k+1}, U, V_k, \Lambda_k, \Pi_k), \\ V_{k+1} &= \arg\min_{V\geq 0} \mathscr{L}_A(X_{k+1}, Y_{k+1}, Z_{k+1}, U_{k+1}, V, \Lambda_k, \Pi_k), \end{aligned}$$

and then updating the multipliers Λ and Π. Specifically, these steps can be written in closed form as

$$X_{k+1} = (Z_k Y_k^T + \alpha U_k - \Lambda_k)(Y_k Y_k^T + \alpha I)^{-1}, \tag{9a}$$
$$Y_{k+1} = (X_{k+1}^T X_{k+1} + \beta I)^{-1}(X_{k+1}^T Z_k + \beta V_k - \Pi_k), \tag{9b}$$
$$Z_{k+1} = X_{k+1} Y_{k+1} + \mathscr{P}_\Omega(M - X_{k+1} Y_{k+1}), \tag{9c}$$
$$U_{k+1} = \mathscr{P}_+(X_{k+1} + \Lambda_k/\alpha), \tag{9d}$$
$$V_{k+1} = \mathscr{P}_+(Y_{k+1} + \Pi_k/\beta), \tag{9e}$$
$$\Lambda_{k+1} = \Lambda_k + \gamma\alpha(X_{k+1} - U_{k+1}), \tag{9f}$$
$$\Pi_{k+1} = \Pi_k + \gamma\beta(Y_{k+1} - V_{k+1}), \tag{9g}$$

where $\gamma \in (0, 1.618)$ and $(\mathscr{P}_+(A))_{ij} = \max\{a_{ij}, 0\}$. Since matrix inversions are applied to $q \times q$ matrices, they are relatively inexpensive for $q < \min\{m, n\}$.

2.3 Convergence

In this subsection, we provide a preliminary convergence property of the proposed ADM algorithm. To simplify notation, we consolidate all the variables as

$$W := (X, Y, Z, U, V, \Lambda, \Pi).$$

A point W is a KKT point of problem (8) if

$$(XY - Z)Y^\top + \Lambda = 0, \tag{10a}$$
$$X^\top(XY - Z) + \Pi = 0, \tag{10b}$$
$$\mathscr{P}_{\Omega^c}(XY - Z) = 0, \tag{10c}$$
$$\mathscr{P}_\Omega(Z - M) = 0, \tag{10d}$$
$$X - U = 0, \tag{10e}$$
$$Y - V = 0, \tag{10f}$$
$$\Lambda \le 0 \le U, \Lambda \odot U = 0, \tag{10g}$$
$$\Pi \le 0 \le V, \Pi \odot V = 0, \tag{10h}$$

where Ω^c is the complement of Ω, which indexes the unobserved entries of M, and $\odot$ denotes component-wise multiplication. We can obtain a result similar to Proposition 1 in [34].

Theorem 2.1. *Let $\{W_k\}$ be a sequence generated by the ADM algorithm (9) that satisfies the condition*

$$\lim_{k\to\infty}(W_{k+1} - W_k) = 0. \tag{11}$$

Then any accumulation point of $\{W_k\}$ is a KKT point of problem (8). Consequently, any accumulation point of $\{(X_k, Y_k)\}$ is a KKT point of problem (1).

Proof. Rearrange the ADM formulas in (9) into

$$\begin{aligned}
(X_{k+1}-X_k)(Y_kY_k^T+\alpha I) &= -((X_kY_k-Z_k)Y_k^T \\
&\quad +\alpha(X_k-U_k)+\Lambda_k), && \text{(12a)}\\
(X_{k+1}^TX_{k+1}+\beta I)(Y_{k+1}-Y_k) &= -(X_{k+1}^T(X_{k+1}Y_k-Z_k) \\
&\quad +\beta(Y_k-V_k)+\Pi_k), && \text{(12b)}\\
U_{k+1}-U_k &= \mathscr{P}_+(X_{k+1}+\Lambda_k/\alpha)-U_k, && \text{(12c)}\\
V_{k+1}-V_k &= \mathscr{P}_+(Y_{k+1}+\Pi_k/\beta)-V_k, && \text{(12d)}\\
\Lambda_{k+1}-\Lambda_k &= \gamma\alpha(X_{k+1}-U_{k+1}), && \text{(12e)}\\
\Pi_{k+1}-\Pi_k &= \gamma\beta(Y_{k+1}-V_{k+1}), && \text{(12f)}
\end{aligned}$$

and

$$Z_{k+1}=X_{k+1}Y_{k+1}+\mathscr{P}_\Omega(M-X_{k+1}Y_{k+1}). \tag{13}$$

Note $W_{k+1}-W_k\to 0$ implies that the left- and right-hand sides in (12) all go to zero, i.e.,

$$\begin{aligned}
(X_kY_k-Z_k)Y_k^T+\Lambda_k &\to 0, && \text{(14a)}\\
X_k^T(X_kY_k-Z_k)+\Pi_k) &\to 0, && \text{(14b)}\\
\mathscr{P}_+(X_k+\Lambda_k/\alpha)-U_k &\to 0, && \text{(14c)}\\
\mathscr{P}_+(Y_k+\Pi_k/\beta)-V_k &\to 0, && \text{(14d)}\\
X_k-U_k &\to 0, && \text{(14e)}\\
Y_k-V_k &\to 0, && \text{(14f)}
\end{aligned}$$

where the terms $\alpha(X_k-U_k)$ and $\beta(Y_k-V_k)$ have been eliminated in (14a) and (14b), respectively, by invoking (14e) and (14f). Since (13) exactly means

$$\mathscr{P}_\Omega(Z_k-M)=0,\ \text{and}\ \mathscr{P}_\Omega(X_kY_k-Z_k)=0,$$

then clearly, the first six equations in the KKT conditions (10) are satisfied at any limit point

$$\hat{W}=(\hat{X},\hat{Y},\hat{Z},\hat{U},\hat{V},\hat{\Lambda},\hat{\Pi}).$$

Then nonnegativity of $\hat{U}$ and $\hat{V}$ are guaranteed by the algorithm construction. Therefore, we only need to verify the non-positivity of $\hat{\Lambda}$ and $\hat{\Pi}$, and the complementarity between $\hat{U}$ and $\hat{\Lambda}$, and between $\hat{V}$ and $\hat{\Pi}$. Now we examine the following two equations derived from (14c) and (14d), respectively,

$$\begin{aligned}
\mathscr{P}_+(\hat{X}+\hat{\Lambda}/\alpha) &= \hat{U}, && \text{(15a)}\\
\mathscr{P}_+(\hat{Y}+\hat{\Pi}/\beta) &= \hat{V}. && \text{(15b)}
\end{aligned}$$

Note we have $\hat{X}=\hat{U}\geq 0$. If $\hat{U}_{ij}=\hat{X}_{ij}=0$, then (15a) reduces $\mathscr{P}_+(\hat{\Lambda}/\alpha)_{ij}=0$, which implies $\hat{\Lambda}_{ij}\leq 0$. On the other hand, if $\hat{U}_{ij}=\hat{X}_{ij}>0$, then (15a) implies $\hat{\Lambda}_{ij}=0$. This proves the non-positivity of $\hat{\Lambda}$ and the complementarity between

$\hat{U}$ and $\hat{\Lambda}$. The same argument can be applied to (15b), due to the identical structure, to prove the non-positivity of $\hat{\Pi}$ and the complementarity between $\hat{V}$ and $\hat{\Pi}$.

We have verified the statement concerning the sequence $\{W_k\}$ and problem (8). The statement concerning the sequence $\{(X_k, Y_k)\}$ and problem (1) follows directly from the equivalence between the two problems. This completes the proof. □

Immediately, we can get the following corollary.

Corollary 2.1. *Whenever $\{W_k\}$ converges, it converges to a KKT point.*

Through a tedious derivation, we can relax the assumption from the convergence of $\{W_k\}$ to the convergence of $\{\Lambda_k\}$ and $\{\Pi_k\}$ only. However, we have not yet been able to further remove the relaxed assumption.

3 Numerical Results

3.1 Implementation and Parameters

A pseudo code for the proposed algorithm is given in Algorithm 1 below.

Algorithm 1 ADM-based algorithm for NMFC

Input $A = \mathscr{P}_\Omega(M) \in \mathbb{R}^{m\times n}$, integer $q > 0$, maxiter > 0, and tol > 0.
Set $\alpha, \beta, \gamma > 0$. Set Y as a nonnegative random matrix, $Z = A$, and U, V, Λ, Π as zero matrices of appropriate sizes.
for $k = 1, \ldots, maxiter$ **do**
  Update $(X_k, Y_k, Z_k, U_k, V_k, \Lambda_k, \Pi_k)$ by the formulas (9);
  if a stopping criterion is met **then**
    exit and output (X_k, Y_k)
  end if
end for

The most important algorithmic parameters are α, β and γ. In our implementation, we set $\gamma = 1.618$, and $\alpha = m\beta/n$. By running a range of numerical experiments, we heuristically scale A so that $\|A\|_F = 2.5 \times 10^5$, and select $\alpha = 1.91 \times 10^{-4}\|A\|_F \max(m, n)/k$. The selection seems to have worked well for our tested matrices. A stopping criterion is met as long as one of the following two conditions is satisfied:

$$\frac{|f_{k+1} - f_k|}{\max(1, |f_k|)} \le tol, \tag{16a}$$

$$f_k \le tol, \tag{16b}$$

where $f_k = \|\mathscr{P}_\Omega(X_kY_k - A)\|_F/\|A\|_F$. All tests were performed on a Lenovo T410 laptop with an i7-620m CPU and 3 gigabytes of memory and running 32-bit Windows 7 and MATLAB 2010b.

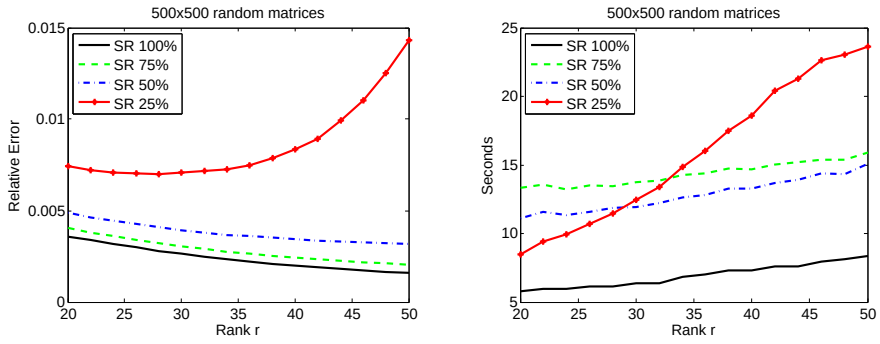


Figure 1: Matrix completion with different sample rates (SRs). Left: relative error in Frobenious norm; Right: cpu time in seconds. The algorithm in [34] was used for SR=100%. Algorithm 1 was used for SR=70%, 50%, 25%. All tests used the same parameters and stopping tolerances, and results are the averages over 50 independent trials

3.2 Random Nonnegative Matrices Factorization

We compared the proposed algorithm 1 to the algorithm proposed in [34], which takes complete samples of M and performs similar ADM-based iterations, on random matrices with varying number of sampled entries. Algorithm 1 was given with matrices subject to three different sample rates: 75%, 50%, and 25% while the compared algorithm took matrices with 100% entries. While other reported tests in this paper use parameters and stopping rules given above, this test set used non-optimal yet consistent parameters for both algorithms in order to accurately reveal their performance difference and compare NMF to NMFC: $\alpha = \beta = 1000$ and $tol = 10^{-6}$. We generated each rank-r nonnegative matrix $M \in \mathbb{R}^{m\times n}$ in the form of $M = LDR$, where $L \in \mathbb{R}^{m\times r}$ and $R \in \mathbb{R}^{r\times n}$ were generated by calling Matlab's command `rand` and D is an $r \times r$ diagonal matrix with diagonal elements $1, 2, \ldots, r$. Such scaling makes M slightly ill-conditioned. We tested different combinations of n and m and obtained roughly consistent results. Figure 1 depicts the recovery qualities and speeds corresponding to $m = n = 500$ and varying $k = r = 20$ through 50. The results are the averages of 50 independent trials.

The quality of recovery is similar for SR = 100%, 75%, and 50% for the set of tested matrices. They are all faithful recoveries with relative errors around 0.4%. The relative errors for SR = 75%, and 50% are just slightly worse. The low SR = 25% makes the recovery more difficult. When the ranks r are between 20 and 30, the four error curves are roughly parallel though the red curve (SR = 25%) is worse at relative errors around 0.6%. When $r > 30$, 25% of entries are no longer enough for faithful recovery and consequently, the red curve (SR = 25%) begins to deviate from the others as r increases, and it exhibits a steep upward trend. The difficulty with SR = 25% samples for large r is also shown in terms of cpu seconds. The times for SR = 75% and 50% are about three times

as long as those for SR = 100%. Since the times are the averages of merely 20 trials, the curves are not as smooth as they would be if the trials were much more.

3.3 Overview of Algorithm LMaFit and FPCA

Before more simulation results are presented, let us overview LMaFit and FPCA, which were compared to Algorithm 1 in next two simulations. LMaFit solves (5) based on a nonlinear successive over-relaxation (SOR) method. From its first-order optimality conditions

$$\begin{aligned}(XY - Z)Y^\top &= 0,\\ X^\top(XY - Z) &= 0,\\ \mathscr{P}_{\Omega^c}(Z - XY) &= 0,\\ \mathscr{P}_\Omega(Z - M) &= 0.\end{aligned}$$

the nonlinear SOR scheme is derived as

$$\begin{aligned}X_{k+1} &= Z_k Y_k^\top (Y_k Y_k^T)^\dagger,\\ X_{k+1}(\omega) &= \omega X_{k+1} + (1-\omega)X_k,\\ Y_{k+1} &= (X_{k+1}(\omega)^\top X_{k+1}(\omega))^\dagger (X_{k+1}(\omega)^\top Z_k),\\ Y_{k+1}(\omega) &= \omega Y_{k+1} + (1-\omega)Y_k,\\ Z_{k+1}(\omega) &= X_{k+1}(\omega)Y_{k+1}(\omega) + \mathscr{P}_\Omega(M - X_{k+1}(\omega)Y_{k+1}(\omega)),\end{aligned}$$

where the weight $\omega \geq 1$. One of its stopping criterions is the same as (16a). In our tests described below, we set tol $= 10^{-5}$ for Alg 1 and LMaFit and chose different maximum numbers of iterations based on the size of recovered matrix, which will be specified below. We applied the rank-estimation technique coming with LMaFit (hence, we did not fix q for LMaFit).

FPCA solves convex problems in the form of

$$\min \mu\|X\|_* + \frac{1}{2}\|\mathscr{A}(X) - b\|_2^2,$$

which includes (4) as a special case by setting the linear operator $\mathscr{A}$ to $\mathscr{P}_\Omega$. Introducing $h(X) = \mathscr{A}^*(\mathscr{A}(X) - b)$, where $\mathscr{A}^*$ is the adjoint of $\mathscr{A}$, we can write the iteration of FPCA as

$$\left\{\begin{array}{rcl} Y_k &\leftarrow& X_k - \tau h(X_k),\\ X_{k+1} &\leftarrow& S_{\tau\mu}(Y_k),\end{array}\right.$$

where $S_\nu(\cdot)$ is a matrix singular-value shrinkage operator. In our tests described below, the parameters for FPCA were set to their default values: specifically, tol $= 10^{-6}$ and maxiter $= 10^5$. For the default values of other parameters such as τ and μ, we refer the reader to [20].

3.4 Hyperspectral Data Recovery

In this subsection, we compare Algorithm 1 with LMaFit [28] and FPCA [20] on recovering three-dimensional hyperspectral images from their incomplete observations. Hyperspectral (or multispectral) imaging is widely used in applications from environmental studies and biomedical imaging to military surveillance. A hyperspectral image is a three-dimensional datacube that records the electromagnetic reflectance of a scene at varying wavelengths, from which different materials in the scene can be identified by exploiting their electromagnetic scattering patterns. We let each hyperspectral datacube be represented by a three-dimensional array whose first two dimensions are spatial and third dimension is wavelength. A hyperspectral datacube can have several hundreds of wavelengths (along the third dimension) but no more than a dozen dominant materials. As a consequence, the spectral vector at every spatial location can be (approximately) linearly expressed by a small set of common vectors, called endmembers or spectral signatures of materials. The number of these basic vectors is much smaller than the number of wavelengths. Since endmembers are naturally nonnegative, a hyperspectral datacube is a set of nonnegative mixtures of a few endmembers, which are also nonnegative. This property makes it possible to recover the endmembers and mixture coefficients from a hyperspectral datacube, and it is called *unmixing*. Although unmixing is not as simple as NMF, the results of NMF can be used as an initial guess. Compared to NMF, NMFC not only performs initial unmixing but also recovers the datacube from an incomplete set of observed voxels. This advantage will translate to shorter sampling times and perhaps simpler designs of hyperspectral imaging devices.

In our simulation, the hyperspectral datacube has 163 wavelengths or slices, and the size of each slice is 80×80. Three selected slices are shown in figure 2. They depict an urban area at three different wavelengthes. Roads, roofs, plants, as well as other objects exhibit different intensities. Our simulation begin with reshaping the $80 \times 80 \times 163$ hyperspectral datacube to a 6400×163 matrix M, each slice becoming one column of M. While M is full rank, its singular values are fast decaying. We chose the estimate rank $k = 30$, and set tol=10^{-5} and maxiter = 1000 for Algorithm 1, and tol=10^{-5} and maxiter = 1000, est_rank=2, rk_inc =3 for LMaFit. The parameters for FPCA were set to their default values.

The three algorithms were compared on recovering M from incomplete observations of SR = 30%, 40%, 50%, and their results were compared in terms of peak signal-to-noise ratio (PSNR), mean squared error (MSE), as well as relative nonnegativity feasibility (FA). Specifically, given a recovered matrix $\hat{M}$

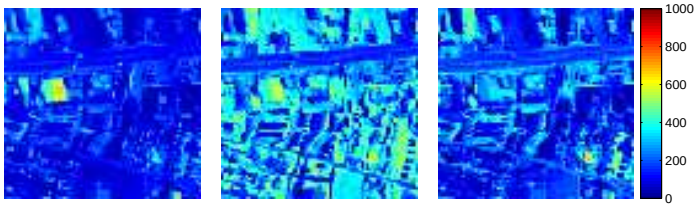
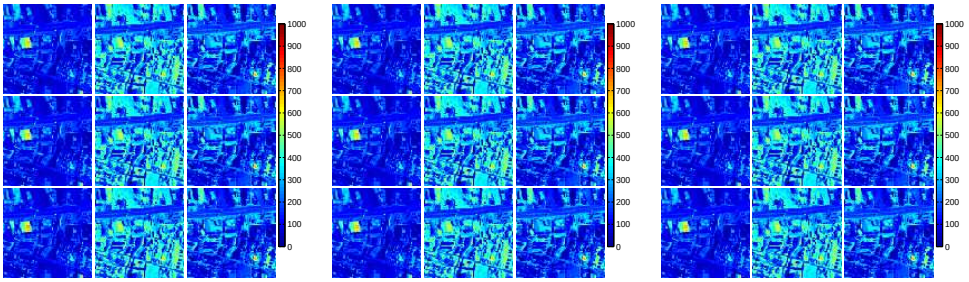


Figure 2: Real data: original slices

Figure 3: Real data: the first, second and third row of each subfigure are recovered images by Algorithm 1, LMaFit, and FPCA, respectively



(a) r=30, SR = 30% (b) r=30, SR = 40% (c) r=30, SR = 50%

from incomplete samples of $M \in \mathbb{R}^{m\times n}$, we let

$$\begin{aligned} MSE &:= \frac{1}{mn}\|M_{app} - M\|_F^2, \\ PSNR &:= 20\log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right), \\ FA &:= \frac{\|\min(M_{app}, 0)\|_F}{\|M\|_F}, \end{aligned}$$

where MAX_I is the maximum pixel intensity, which is 1023 in this subsection for the tested hyperspectral data and 1 in subsection 3.5 for two grayscale images. The results are listed in table 1, and the three slices of the recovered datacube that correspond to those in figure 2 are depicted in figure 3. The results show that Algorithm 1 performs better than FPCA in both CPU time and recovery quality. LMaFit is generally faster but less accurate than algorithm 1. We believe that the use of nonnegativity is a major factor for the superiority of the results of algorithm 1.

3.5 Tests on images

Despite that natural image recovery from incomplete random samples is not a typical image processing task, we picked it to test algorithm 1, LMaFit, and

Table 1: real data: recovered slices by Algorithm 1, LMaFit, and FPCA. The rank estimate for Algorithm 1 and LMaFit is 30.

problem	Alg 1				LMaFit				FPCA			
seed	CPU	PSNR	MSE	FA	CPU	PSNR	MSE	FA	CPU	PSNR	MSE	FA
SR: 30%												
3445	35.04	47.52	1.85e+1	0	40.39	42.10	6.45e+1	8.57e-3	38.49	44.53	3.69e+1	1.12e-3
31710	34.77	47.32	1.94e+1	0	23.43	43.46	4.71e+1	5.08e-3	38.40	44.42	3.79e+1	1.24e-3
43875	34.31	47.42	1.89e+1	0	38.27	42.54	5.83e+1	7.70e-3	38.64	44.71	3.54e+1	1.21e-3
69483	34.19	47.36	1.92e+1	0	34.53	42.74	5.57e+1	6.83e-3	38.75	44.62	3.61e+1	1.04e-3
95023	33.69	47.42	1.90e+1	0	27.88	42.98	5.27e+1	6.19e-3	38.78	44.51	3.70e+1	1.19e-3
SR: 40%												
3445	36.84	48.83	1.37e+1	0	35.05	43.90	4.26e+1	7.55e-3	42.76	44.72	3.53e+1	1.09e-3
31710	36.45	48.66	1.42e+1	0	24.45	44.71	3.54e+1	5.48e-3	42.57	44.53	3.69e+1	1.06e-3
43875	36.61	48.92	1.34e+1	0	17.67	46.07	2.59e+1	4.65e-3	42.85	44.66	3.58e+1	1.31e-3
69483	38.37	48.68	1.42e+1	0	19.13	45.59	2.89e+1	5.52e-3	42.88	44.52	3.70e+1	1.06e-3
95023	38.65	48.50	1.48e+1	0	22.16	45.30	3.09e+1	5.02e-3	43.22	44.56	3.66e+1	1.09e-3
SR: 50%												
3445	39.87	49.74	1.11e+1	0	34.31	44.72	3.53e+1	9.62e-3	47.12	44.24	3.94e+1	8.55e-4
31710	39.77	49.75	1.11e+1	0	29.90	45.23	3.14e+1	4.78e-3	46.64	45.25	3.12e+1	1.12e-3
43875	37.78	49.78	1.10e+1	0	25.53	45.92	2.68e+1	6.12e-3	46.63	44.60	3.63e+1	1.44e-3
69483	38.24	49.65	1.14e+1	0	31.14	44.92	3.37e+1	7.77e-3	47.07	44.39	3.81e+1	1.08e-3
95023	40.09	49.64	1.14e+1	0	29.92	45.42	3.00e+1	5.64e-3	47.90	43.93	4.24e+1	1.16e-3

FPCA since it is easy to visualize their solution qualities. This simulation used two grayscale images, the 768×1024 Kittens and the 1200×1600 Panda, shown in figure 4.

Figure 4: Original images: Kittens (left) and Panda (right)

We applied relatively small (thus, challenging) sample rates of SR=10%, 20%, 30% for Kittens and SR=10%, 15%, 20% for Panda. We set tol=10^{-5}, and maxiter=2000 for Algorithm 1 and LMaFit and est_rank=2, rk_inc =3 for LMaFit. The parameters for FPCA were set to their default values. The results are given in tables 2 and 3 and the recovered images in figures 5 and 6.

Tables 2 and 3 indicate that FPCA performs slightly better than algorithm

Table 2: Recover Kittens by Algorithm 1, LMaFit, and FPCA. The rank estimate for Algorithm 1 and LMaFit is 40.

problem	Alg 1				LMaFit				FPCA			
seed	CPU	PSNR	MSE	FA	CPU	PSNR	MSE	FA	CPU	PSNR	MSE	FA
SR: 10%												
3445	33.86	18.23	1.50e-2	0	36.60	9.32	1.17e-1	1.79e-1	23.66	20.09	9.80e-3	9.33e-4
31710	30.23	18.14	1.54e-2	0	31.53	9.37	1.15e-1	1.71e-1	24.30	20.12	9.72e-3	1.62e-3
43875	32.10	18.05	1.56e-2	0	47.21	9.17	1.21e-1	1.88e-1	23.25	20.18	9.59e-3	1.13e-3
69483	31.31	18.09	1.55e-2	0	44.50	9.05	1.24e-1	1.90e-1	23.29	20.07	9.84e-3	1.04e-3
95023	32.19	18.09	1.55e-2	0	40.70	9.19	1.21e-1	1.81e-1	23.16	20.06	9.87e-3	7.84e-4
SR: 20%												
3445	17.69	23.26	4.72e-3	0	21.65	19.71	1.07e-2	3.04e-2	34.54	22.38	5.78e-3	9.45e-4
31710	14.34	23.14	4.85e-3	0	14.28	19.81	1.05e-2	3.19e-2	34.29	22.25	5.95e-3	7.17e-4
43875	14.66	23.19	4.80e-3	0	14.40	19.57	1.10e-2	3.51e-2	34.24	22.26	5.94e-3	8.53e-4
69483	15.17	23.14	4.86e-3	0	20.10	19.45	1.13e-2	3.81e-2	34.37	22.38	5.78e-3	1.01e-3
95023	14.36	23.16	4.83e-3	0	10.93	20.05	9.89e-3	2.74e-2	34.22	22.31	5.87e-3	5.67e-4
SR: 30%												
3445	13.49	24.53	3.52e-3	0	6.08	24.05	3.94e-3	2.37e-3	55.82	23.44	4.53e-3	6.46e-4
31710	11.15	24.48	3.56e-3	0	2.53	24.05	3.94e-3	4.26e-3	55.84	23.30	4.68e-3	7.95e-4
43875	14.67	24.48	3.56e-3	0	3.29	24.02	3.96e-3	2.74e-3	55.96	23.47	4.50e-3	9.63e-4
69483	11.06	24.45	3.59e-3	0	5.19	24.02	3.97e-3	2.93e-3	55.74	23.40	4.57e-3	5.73e-4
95023	11.52	24.46	3.58e-3	0	3.43	24.05	3.93e-3	2.58e-3	55.02	23.33	4.65e-3	2.67e-4

1 in terms of CPU time and recovery quality when SR is as small as 10% while at this SR, LMaFit almost fails. With larger SRs such as 20% and 30% for Kittens and SR=15% and 20% for Panda, algorithm 1 is both faster and returns better images than FPCA. With SR=20%, algorithm 1 is better than LMaFit on Kittens in terms of both CPU time and recovery quality. However, with SR=30%, LMaFit becomes faster than algorithm 1, yet LMaFit's recovery quality remains worse. As SR further increases, the three algorithms will return images with almost the same quality while LMaFit is the best in speed.

4 Conclusions

Among the extensive applications of nonnegative matrix factorization and those of low-rank matrix completion, there is a rich subset of problems with both low-rank matrices and nonnegativity factors, and the input matrices are incomplete. These features and challenges altogether lend us the opportunity to improve the solutions by adopting a formulation of nonnegative matrix factorization and completion. This paper presents our first attempt for solving this non-convex formulation, using an iteration based on the classic alternating direction augmented Lagrangian method. Both the per-iteration and total computation complexities are satisfactory especially when the estimate rank q is small. We hope that the numerical results in this paper convince the reader that the underlying formulation is useful and the alternating direction algorithm can be very powerful. As global convergence and recovery guarantee results are largely unknown, we hope that the results of this paper will also motivate

Figure 5: Recovered 768×1024 Kittens with estimate rank $k = 40$ by Algorithm 1 (left), LMaFit (middle), and FPCA (right)

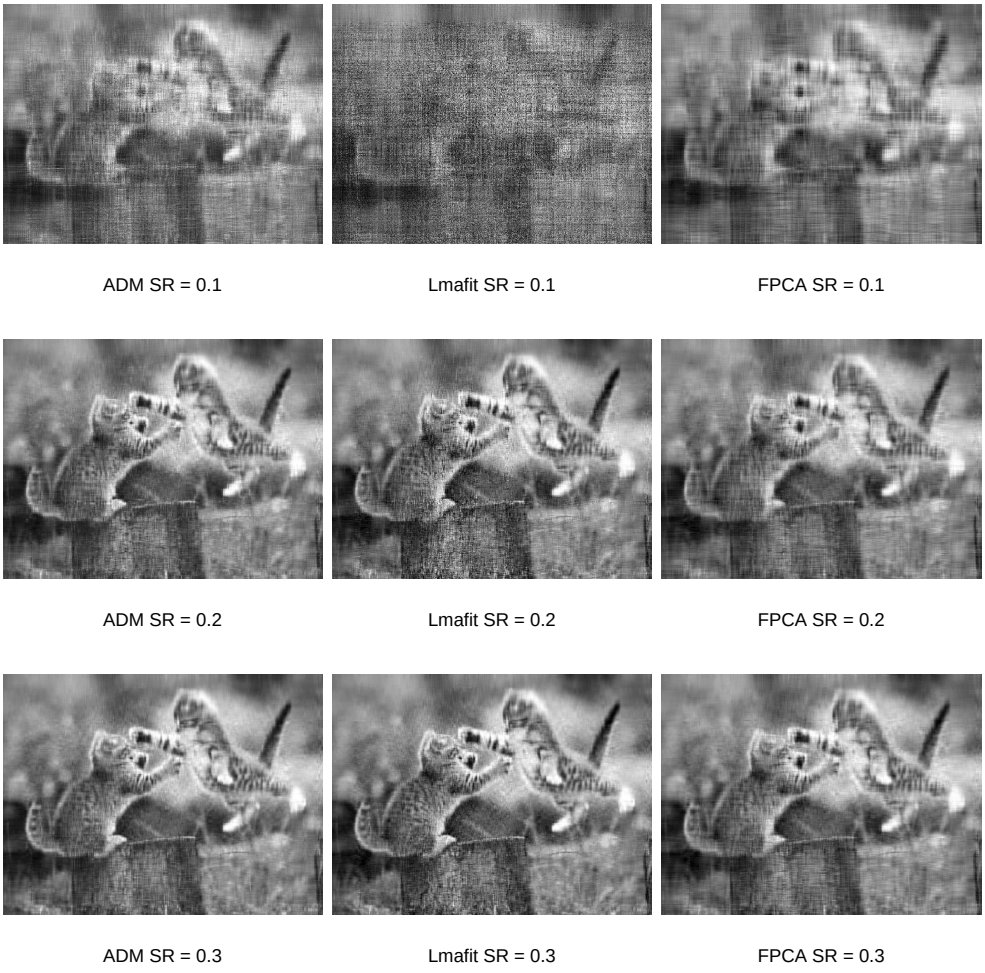


Figure 6: Recovered 1200×1600 Panda for estimate rank $k = 40$ by Algorithm 1 (left), LMaFit (middle), and FPCA (right)

Table 3: Panda: recovered images by Algorithm 1, LMaFit, and FPCA. The rank estimate for Algorithm 1 and LMaFit is 40.

problem	Alg 1				LMaFit				FPCA			
seed	CPU	PSNR	MSE	FA	CPU	PSNR	MSE	FA	CPU	PSNR	MSE	FA
SR: 10%												
3445	65.81	23.58	4.39e-3	0	82.75	14.47	3.57e-2	2.28e-1	59.15	23.51	4.46e-3	1.96e-2
31710	71.95	23.32	4.66e-3	0	7.72	16.56	2.21e-2	1.18e-1	58.44	23.66	4.30e-3	1.78e-2
43875	64.98	23.56	4.41e-3	0	26.56	16.27	2.36e-2	1.53e-1	58.01	23.67	4.29e-3	1.72e-2
69483	72.74	23.75	4.22e-3	0	59.49	15.03	3.14e-2	2.02e-1	57.75	23.63	4.34e-3	1.86e-2
SR: 15%												
3445	43.08	25.82	2.62e-3	0	30.36	23.89	4.09e-3	3.78e-2	72.95	25.26	2.98e-3	1.48e-2
31710	40.20	25.75	2.66e-3	0	15.36	24.19	3.81e-3	3.25e-2	71.82	25.15	3.06e-3	1.53e-2
43875	45.65	25.84	2.60e-3	0	22.66	23.86	4.12e-3	4.15e-2	70.57	25.14	3.06e-3	1.49e-2
69483	38.19	25.80	2.63e-3	0	18.58	24.09	3.90e-3	3.48e-2	71.08	25.13	3.07e-3	1.56e-2
SR: 20%												
3445	37.45	26.55	2.21e-3	0	10.96	26.68	2.15e-3	1.70e-2	94.53	25.81	2.62e-3	1.31e-2
31710	33.10	26.53	2.22e-3	0	6.42	26.73	2.12e-3	1.81e-2	89.36	25.87	2.59e-3	1.34e-2
43875	39.87	26.56	2.21e-3	0	8.27	26.70	2.14e-3	1.79e-2	89.83	25.73	2.67e-3	1.34e-2
69483	29.68	26.56	2.21e-3	0	11.60	26.67	2.15e-3	1.72e-2	89.14	25.85	2.60e-3	1.31e-2

future theoretical studies, as well as more efficient algorithms.

Acknowledgements

The work of W. Yin was supported in part by US NSF CAREER Award DMS-07-48839, ONR Grant N00014-08-1-1101, ARL and ARO grant W911NF-09-1-0383, and an Alfred P. Sloan Research Fellowship. The work of Z. Wen was supported in part by NSF DMS-0439872 through UCLA IPAM. The work of Y. Zhang was supported in part by NSF DMS-0811188 and ONR grant N00014-08-1-1101.

References

[1] M.W. Berry, M. Browne, A.N. Langville, V.P. Pauca, and R.J. Plemmons. Algorithms and applications for approximate nonnegative matrix factorization. *Computational Statistics & Data Analysis*, 52(1):155–173, 2007.

[2] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and distributed computation: numerical methods*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[3] P. Biswas, T.C. Lian, T.C. Wang, and Y. Ye. Semidefinite programming based algorithms for sensor network localization. *ACM Transactions on Sensor Networks (TOSN)*, 2(2):188–220, 2006.

[4] J.F. Cai, E.J. Candes, and Z. Shen. A singular value thresholding algorithm for matrix completion export. *SIAM J. Optim.*, 20:1956–1982, 2010.

[5] E.J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

[6] E.J. Candes and T. Tao. The power of convex relaxation: Near-optimal matrix completion. *Information Theory, IEEE Transactions on*, 56(5):2053–2080, 2010.

[7] A. Cichocki, M. Morup, P. Smaragdis, W. Wang, and R. Zdunek. Advances in nonnegative matrix and tensor factorization. *Computational intelligence and neuroscience*, 2008.

[8] A. Cichocki, R. Zdunek, A. H. Phan, and S. Amari. Nonnegative Matrix and Tensor Factorizations - Applications to Exploratory Multiway Data Analysis and Blind Source Separation. 2009.

[9] M. Fazel. *Matrix Rank Minimization with Applications*. PhD thesis, Stanford University, 2002.

[10] D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers and Mathematics with Applications*, 2(1):17–40, 1976.

[11] R. Glowinski and A. Marrocco. Sur lapproximation par elements finis dordre un, et la resolution par penalisation-dualite dune classe de problemes de Dirichlet nonlineaires. *Rev. Francaise dAut. Inf. Rech. Oper.*, pages 41–76, 1975.

[12] D. Goldberg, D. Nichols, B.M. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.

[13] D. Goldfarb, S. Ma, and Z. Wen. Solving low-rank matrix completion problems efficiently. *In proceedings of 47th Annual Allerton Conference on Communication, Control, and Computing, Monticello, Illinois*, 2009.

[14] L Grippo and M Sciandrone. On the convergence of the block nonlinear gauss-seidel method under convex constraints* 1. *Operations Research Letters*, 26(3):127–136, 2000.

[15] E. T. Hale, W. Yin, and Y. Zhang. Fixed-point continuation for ℓ_1-minimization: methodology and convergence. *SIAM Journal on Optimization*, 19(3):1107–1130, 2008.

[16] M.R. Hestenes. Multiplier and gradient methods. *Journal of optimization theory and applications*, 4(5):303–320, 1969.

[17] D.D. Lee and H.S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

[18] D.D. Lee and H.S. Seung. Algorithms for Non-Negative Matrix Factorization. *Advances in Neural Information Processing Systems*, 13:556–562, 2001.

[19] Z. Liu and L. Vandenberghe. Interior-point method for nuclear norm approximation with application to system identification. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1235–1256, 2009.

[20] S. Ma, D. Goldfarb, and L. Chen. Fixed point and Bregman iterative methods for matrix rank minimization. *Mathematical Programming*, pages 1–33, 2009.

[21] P. Paatero. Least squares formulation of robust non-negative factor analysis. *Chemometrics and Intelligent Laboratory Systems*, 37(1):23–35, 1997.

[22] P. Paatero. The multilinear engine: A table-driven, least squares program for solving multilinear problems, including the n-way parallel factor analysis model. *Journal of Computational and Graphical Statistics*, 8(4):854–888, 1999.

[23] P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994.

[24] M. J. D. Powell. A method for nonlinear constraints in minimization problems, in optimization. pages 283–298, 1969.

[25] 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.

[26] R.T. Rockafellar. The multiplier method of Hestenes and Powell applied to convex programming. *Journal of Optimization Theory and Applications*, 12(6):555–562, 1973.

[27] Y. Wang, J. Yang, W. Yin, and Y. Zhang. A new alternating minimization algorithm for total variation image reconstruction. *SIAM J. Imaging Sci.*, 1(3):248–272, 2008.

[28] Z. Wen, W. Yin, and Y. Zhang. Solving a low-rank factorization model for matrix completion by a non-linear successive over-relaxation algorithm. *Rice University CAAM Technical Report TR10-07. Submitted*, 2010.

[29] Zaiwen Wen, Donald Goldfarb, and Wotao Yin. Alternating direction augmented lagrangian methods for semidefinite programming. *Mathematical Programming Computation*, 2(3-4):203–230, 2010.

[30] J. Wright, A. Ganesh, S. Rao, and Y. Ma. Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. *submitted to Journal of the ACM*, 2009.

[31] J. Yang and X. Yuan. An inexact alternating direction method for trace norm regularized least squares problem. *online at optimization online*, 2010.

[32] J. Yang, Y. Zhang, and W. Yin. An efficient tvl1 algorithm for deblurring multichannel images corrupted by impulsive noise. *SIAM Journal on Scientific Computing*, 31:2842–2865, 2008.

[33] W. Yin, S. Osher, D. Goldfarb, and J. Darbon. Bregman iterative algorithms for ℓ_1-minimization with applications to compressed sensing. *SIAM Journal on Imaging Sciences*, 1(1):143–168, 2008.

[34] Y. Zhang. An alternating direction algorithm for nonnegative matrix factorization. *Rice Technical Report*, 2010.