CompAMa
www.compama.co.usb.ve

Bull. Comput. Appl. Math., Vol.2, No.2, 2014
ISSN 2244-8659

Matrix completion via a low rank factorization model and an Augmented Lagrangean Succesive Overrelaxation Algorithm

H. Lara [*], *H. Oviedo* [†], *J. Yuan* [‡]

CompAMa Vol.2, No.2, pp.21-46, 2014 - Accepted February 18, 2015

Abstract

The matrix completion problem (MC) has been approximated by using the nuclear norm relaxation. Some algorithms based on this strategy require the computationally expensive singular value decomposition (SVD) at each iteration. One way to avoid SVD calculations is to use alternating methods, which pursue the completion through matrix factorization with a low rank condition. In this work an augmented Lagrangean-type alternating algorithm is proposed. The new algorithm uses duality information to define the iterations, in contrast to the solely primal LMaFit algorithm, which employs a Successive Over Relaxation scheme. The convergence result is studied. Some numerical experiments are given to compare numerical performance of both proposals.

Keywords: Matrix completion, alternating minimization, nonlinear Gauss-Seidel method, nonlinear SOR method, Augmented Lagrange method.

[*]Department of Operational Research and Statistics, Universidad CentroOccidental Lisandro Alvarado, Núcleo Obelisco 3001, Barquisimeto, Venezuela, (hugol@ucla.edu.ve)

[†]Maestria en Optimización, Universidad CentroOccidental Lisandro Alvarado, Núcleo Obelisco, 3001, Barquisimeto, Venezuela, (harry_oviedo89@hotmail.com)

[‡]Department of Mathematics, Federal University of Parana, Centro Politecnico, Curitiba, CEP 81531-990, PR, Brazil, (yuanjy@gmail.com)

1 Introduction

We consider the problem of recovering a low rank matrix from a known subset of the entries. Such problem is called the Matrix Completion (MC) problem, which consists in solving

$$\min_{W\in\mathbb{R}^{m\times n}} rank(W), \quad s.t. \quad W_{ij} = M_{ij}, \forall (i,j) \in \Omega, \tag{1}$$

where $rank(W)$ denotes the rank of matrix W, M_{ij} is the $(i,j)-$th entry of M and $\Omega \subset \{(i,j) : 1 \leq i \leq m, 1 \leq j \leq n\}$. Although the above problem is NP-hard due to the combinatorial nature of the $rank(\cdot)$ function, it has been proved that a solution can be approximated by solving the following convex relaxation (see [1] and [2]):

$$\min_{W\in\mathbb{R}^{m\times n}} ||W||_* \quad s.t. \quad W_{ij} = M_{ij}, \ \forall (i,j) \in \Omega,$$

where $||W||_*$ denotes the nuclear norm of matrix W, and it is defined as the sum of the singular values of matrix W.

In [3], Wen, Yin and Zhang have proposed an alternative approach to calculate the low rank matrix, which avoids singular value calculations. By fixing an overestimation K of the real rank of the data matrix M, they have successfully proposed the following nonconvex relaxation of problem (1):

$$\min_{X,Y,Z} \frac{1}{2}||XY - Z||_F^2, \quad s.t. \quad Z_{ij} = M_{ij}, \forall (i,j) \in \Omega \tag{2}$$

where $X \in \mathbb{R}^{m\times K}$, $Y \in \mathbb{R}^{K\times n}$, $Z \in \mathbb{R}^{m\times n}$ and K is dynamically adjusted in order to deal with the rank minimization. The constraints in (2) can be written as $P_\Omega(Z - M) = 0$ where $P_\Omega(A)$ is the matrix with components equals to A_{ij} if $(i,j) \in \Omega$, and zero otherwise. The advantage is that the optimality conditions for problem (2) leads to a system of nonlinear equations with a special structure that allows us to efficiently solve them with cheap alternating schemes and numerical algebra tools.

Associated to this linear equality constrained optimization problem it is defined the Lagrangean function $\mathcal{L}$ by

$$\mathcal{L}(X,\ Y,\ Z,\ \Lambda) = \frac{1}{2}\|XY - Z\|_F^2 + \langle \Lambda, P_\Omega(Z - M)\rangle, \tag{3}$$

which leads to the first order optimality conditions for problem (2):

$$(XY - Z)Y^T = 0 \tag{4}$$
$$X^T(XY - Z) = 0 \tag{5}$$
$$Z - XY + P_\Omega(\Lambda) = 0 \tag{6}$$
$$P_\Omega(Z - M) = 0. \tag{7}$$

Note that $\Lambda = P_\Omega(\Lambda)$, that is the dual variables are defined componentwise only on the indices in Ω. Although this is a nonlinear system of equations, which can be difficult to solve, the special structure of problem (2) allows us to establish alternating schemes by decomposing the variables into tree groups X, Y and Z. By the decomposition, we can fix two of the groups of variables to compute the third one. The optimality problem designed to the third one is a quadratic optimization problem, which can be solved by the normal equations. For example, if variables Y and Z are given, then we can calculate X_+ by

$$X_+ := \text{argmin}_X \mathcal{L}(X, Y, Z, \Lambda).$$

For this convex problem the first order optimality condition is just (4), which provides the formula

$$X_+ = ZY^T(YY^T)^\dagger,$$

where $A^\dagger$ denotes a pseudoinverse matrix of A. Similarly we define Y_+ by fixing X_+ and Z and solving the underlying convex quadratic optimization problem, and finally fixing X_+ and Y_+ we obtain Z_+. The resulting alternating scheme, so called Nonlinear Gauss-Seidel Method provides the following updating formulae

$$\begin{aligned}
&X_+ := \text{argmin}_X\{\mathcal{L}(X, Y, Z, \Lambda)\} = ZY^T(YY^T)^\dagger,\\
&Y_+ := \text{argmin}_Y\{\mathcal{L}(X_+, Y, Z, \Lambda)\} = (X_+^TX_+)^\dagger X_+^TZ, \text{ and}\\
&Z_+ := \text{argmin}_Z\{\mathcal{L}(X_+, Y_+, Z, \Lambda)\} = X_+Y_+ + P_\Omega(M - X_+Y_+).
\end{aligned}$$

Note that for the structure of the problem, the dual variables Λ are vanished from the updating formulae. Alternatively to this scheme, they propose a Successive Over Relaxation scheme (SOR) to solve the (MC) problem. This procedure is usually employed to accelerate the Gauss Seidel scheme (See [4] and [5] for references on the Gauss-Seidel and SOR methods).

Augmented Lagrangean Methods are popular tools to deal with some optimization problems with equality constraints (see for example [6] or [7]). Here, we shall use an Augmented Lagrangean method to solve (2), which

is primal-dual algorithm in the sense of updating the original (primal variables) and the Lagrange multipliers (dual) alternatively in each iteration. In [8] Zhang has proposed an alternating direction algorithm for nonnegative Matrix Factorization. Lin, Chen, Wu and Ma [9] have dealt with an augmented Lagrange Multiplier Method for Exact Recovery Low-rank Matrices.

The remainder of the paper is organized as follows. In Section 2, we present the Augmented Lagrangean Methods to solve problem (2). Based on the theoretical Augmented Lagrangean algorithm, we develop an alternanting Augmented Lagrangean (ALM-GS) algorithm by using the Gauss-Seidel method on the first order optimality conditions for the minimization of the Augmented Lagrange function with fixed Lagrange Multipliers and penalty parameter. In Section 3, we propose an alternating Augmented Lagrangean algorithm accelerating Gauss-Seidel Method, by using the Successive Over Relaxation strategy (ALM-SOR). The convergence analysis of our algorithms is presented in Section 4. Computational experiments to illustrate the performance of the algorithms, and comparing uwith LMaFit [3] are given in Section 5. Finally we offer some concluding remarks in Section 6.

2 Algorithm

In this section we shall propose Alternating algorithm based on the Augmented Lagrange Method (ALM) for (2). The Augmented Lagrangean method to solve optimization problems with equality constraints of the form

$$\min_{\omega} f(\omega) \text{ s.t. } h(\omega) = 0$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $h : \mathbb{R}^n \to \mathbb{R}^m$, is defined as follows: First, the Augmented Lagrangean Function

$$\mathcal{L}_\mu(\omega, \lambda) = f(\omega) + \langle \lambda, h(\omega) \rangle + \frac{\mu}{2} ||h(\omega)||_F^2$$

is defined, where μ is a positive scalar parameter, called penalty parameter. The problem can be solved by the following algorithm:

Algorithm: Generic ALM

INPUT: $\rho > 1$, $k = 1$

1. while not convergent, do

2. solve $\omega_{k+1} := \text{argmin}_{\omega}\mathcal{L}_{\mu_k}(\omega, \lambda_k)$
3. $\lambda_{k+1} := \lambda_k + \mu_k h(\omega_{k+1})$;
4. $\mu_{k+1} := \rho\mu_k$;
5. k=k+1;
6. End (While)

OUTPUT: ω_k, λ_k

A variant of the algorithm, in which μ_k is increased only if there is not sufficient progress in the feasibility, produces a sequence $\{\mu_k\}$ that can be bounded. It has been proved in [6] that under general conditions, the Lagrange Multipliers λ_k produced by the above algorithm converge $Q-$linearly to an optimal solution for continuously differentiable f and g, when $\{\mu_k\}$ is bounded and super $Q-$ linearly when $\{\mu_k\}$ is unbounded. This convergence property makes ALM very attractive. Another merit of ALM is that the optimal step size to update λ_k was shown to be the chosen penalty parameter μ_k, making easy the parameter tuning. A third merit of ALM is that the algorithm converges to the exact optimal solution, even without requiring μ_k to approach infinity (see [6] or [7]).

2.1 The Augmented Lagrangean Algorithm

We first establish the classic Augmented Lagrangean method to solve problem (2). Let us consider the Augmented Lagrangean function

$$\mathcal{L}_{\mu}(X, Y, Z, \Lambda) = \frac{1}{2}||XY - Z||_F^2 + \langle \Lambda, \mathcal{P}_{\Omega}(Z - M)\rangle + \frac{\mu}{2}||\mathcal{P}_{\Omega}(Z - M)||_F^2. \quad (8)$$

The first order optimality conditions for minimizing $\mathcal{L}_{\mu}$ are equations (4), (5) and

$$Z - XY + P_{\Omega}(\Lambda + \mu(Z - M)) = 0. \quad (9)$$

We first establish the theoretical ALM for the problem (2) as follows.

ALM for (2)

INPUT: M

1. $\Lambda_0^* = 0$; $\mu_0 = 1$; $\rho > 1$; $k = 0$.

2. While not convergent, do
3. solve $(X^*_{k+1}, Y^*_{k+1}, Z^*_{k+1}) := argmin_{X,Y,Z}\mathcal{L}_{\mu_k}(X, Y, Z, \Lambda^*_k)$;
4. $\Lambda^*_{k+1} := \Lambda^*_k + \mu_k\mathcal{P}_\Omega(Z^*_{k+1} - M)$;
5. $\mu_{k+1} := \rho\mu_k$; $k := k + 1$;
6. End (While)

OUTPUT: $(X^*_k, Y^*_k, Z^*_k, \Lambda^*_k)$

Note that the ALM algorithm deals with the augmented Lagrangean function minimization in an alternating scheme by fixing the multiplier Λ to minimize the Lagrangean function in one step, and then updating Λ in another step until convergence. The underlying optimization problem in step 3 is a nonlinear optimization problem which could be treated by a nonlinear optimization algorithm.

Instead of solving the Augmented Lagrangean minimization for all the variables at the same time, like Wen Yin and Zhang in [3], we approximate separately the iterates through an alternating scheme, consistent with a nonlinear Gauss Seidel algorithm applied to the optimality conditions (4), (5) and (9), for fixed Λ and μ. The procedure can be seen as an internal loop which approximately calculates the primal iterates, while the multipliers are fixed. In order to do it, we first fix initial Y and Z to obtain $X_+ = \text{argmin}_X\mathcal{L}_\mu(Z, Y, Z, \Lambda)$, in the second step fix X_+ and Z to calculate $Y_+ = \text{argmin}_Y\mathcal{L}_\mu(X_+, Y, Z, \Lambda)$, and finally determine Z_+ by $Z_+ = \text{argmin}_Z\mathcal{L}_\mu(X_+, Y_+, Z, \Lambda)$. By manipulating the optimality conditions we establish the following ALM-GS updating scheme:

ALM-GS updating formulae

$$\begin{aligned} X_+ &:= ZY^\dagger \equiv ZY^\top(YY^\top)^\dagger, \\ Y_+ &:= (X_+)^\dagger Z \equiv (X_+^\top X_+)^\dagger(X_+^\top Z), \\ Z_+ &:= X_+Y_+ + \frac{\mu}{1+\mu}\mathcal{P}_\Omega(M - X_+Y_+ - \frac{1}{\mu}\Lambda). \end{aligned} \tag{10}$$

Now, by using some tools coming from linear algebra, we establish alternative ways to deal with the above calculations, in a similar manner that Wen, Yin and Zhang [3]. Since our goal is to obtain the product matrix XY, and not each of the factor matrices individually, then there are different

ways to define X and Y obtaining the same product. By (10) we note that $\mathcal{R}(X_+) = \mathcal{R}(ZY^T)$. This fact leads us to use

$$X_+ = ZY^T$$

to update X_+ instead of (10). A second variant of ALM-GS is obtained by taking advantage of the $QR-$factorization: For a $m \times n$ matrix A, $Q := orth(A)$ denotes an orthonormal basis for $\mathcal{R}(A)$. Let $V = orth(ZY^T)$ be an orthonormal base of subspace $\mathcal{R}(ZY^T)$. Then, we can use the following updating formulas:

$$X_+ := V, \tag{11}$$

$$Y_+ := V^\top Z. \tag{12}$$

Note that $X_+Y_+ = VV^TZ$. By using these last formulas we can use the fast QR factorization to deal with the pseudoinverses immerse in the ALM-GS alternating scheme.

We are ready to propose the Augmented Lagrangean Gauss Seidel method for solving MC. By using the ALM-GS scheme we build the following algorithm:

Alternating ALM for MC (ALM-GS)

INPUT: M

1. $\Lambda_0 = 0$; $Y_0 = I$; $Z_0 = \mathcal{P}_\Omega(M)$; $\mu_0 = 1$; $\rho > 1$; $k = 0$.
2. While not convergent, do
3. **lines 4-9 approximately solve $(X_{k+1}, Y_{k+1}, Z_{k+1}) := argmin_{X,Y,Z}\mathcal{L}_{\mu_k}(X, Y, Z, \Lambda_k)$;
4. While not convergent, do
5. $X_k^{l+1} := Orth\{Z_k^l(Y_k^l)^T\}$
6. $Y_k^{l+1} := (X_k^{l+1})^T Z_k^l$
7. $Z_k^{l+1} := X_k^{l+1}Y_k^{l+1} + \frac{\mu_k}{1+\mu_k}\mathcal{P}_\Omega(M - X_k^{l+1}Y_k^{l+1} - \frac{1}{\mu_k}\Lambda_k)$;
8. $l := l + 1$;
9. End (While).
10. $(X_{k+1}, Y_{k+1}, Z_{k+1}) := (X_k^l, Y_k^l, Z_k^l)$;
11. $\Lambda_{k+1} := \Lambda_k + \mu_k\mathcal{P}_\Omega(Z_{k+1} - M)$;

12. $\mu_{k+1} := \rho\mu_k$
13. $k := k + 1$;
14. End (While).

OUTPUT: $(X_k, Y_k, Z_k), \Lambda_k$.

Note that the Augmented Lagrangean minimization in the algorithm is solved between lines 4 and 9 by using nonlinear Gauss-Seidel method on the optimality conditions (4), (5) and (9), with relation (11) - (12). This internal procedure is used in LMafit algorithm (see [3]). In practice, we only use $O(1)$ internal iterations for each external one. In our experiments we keep counting of the internal iterations, in which the main tool is solving a QR factorization.

3 An ALM-SOR algorithm for the MC problem

According to our numerical experiments, the alternating ALM-GS algorithm for MC exhibits good performance in quality of the solutions, and in some cases is faster than the LMaFit algorithm. However, our procedure is not sufficiently faster in number of iterations, and the running time, in large scale problems (see tables 1 and 3). In order to overcome this inconvenience, we propose to accelerate the algorithm by an Augmented Lagrangean-Successive Over Relaxation (SOR) algorithm, in a similar manner that in [3]. The SOR scheme is a procedure designed to accelerate the Gauss Seidel algorithm, by extrapolating the iterates (see [4] and [5]). In [3] Wen, Yin and Zhang propose a nonlinear SOR to deal with the MC problem applied to its optimality conditions for the minimization of the regular Lagrangean function (3). Such conditions are similar to (4) to (7) with the difference that equation (3) does not consider the regularization (augmented) term. Their SOR algorithm is a solely primal method, in the sense that only the primal X, Y, Z variables are updated in each iteration. In our formulation we build a primal-dual method in which both groups the primal and dual variables are alternatively updated. The proposed augmented Lagrangean type method for solving MC is via factorization, merging a Gauss Seidel-SOR alternating scheme to solve the Augmented Lagrangean function minimization.

ALM-SOR Updating Formulae

$$X_+ := ZY^\dagger \equiv ZY^T(YY^T)^\dagger, \tag{13}$$

$$X_+(\omega) := \omega X_+ + (1-\omega)X, \tag{14}$$

$$Y_+ := (X_+(\omega))^\dagger Z \equiv (X_+(\omega)^\top X_+(\omega))^\dagger (X_+(\omega)^\top Z), \tag{15}$$

$$Y_+(\omega) := \omega Y_+ + (1-\omega)Y, \tag{16}$$

$$Z_+ := X_+(\omega)Y_+(\omega) + \frac{\mu}{1+\mu}\mathcal{P}_\Omega(M - X_+(\omega)Y_+(\omega) - \frac{1}{\mu}\Lambda). \tag{17}$$

where $\omega \geq 1$. Obviously if $\omega = 1$ we recover the nonlinear Gauss-Seidel method for our problem. Let us denote the residual by:

$$S_\mu := \frac{\mu}{1+\mu}P_\Omega(M - XY - \frac{1}{\mu}\Lambda)$$

which will be used to measure optimality. Note that after each iteration the variable Z can be expressed as:

$$Z = XY + S_\mu.$$

In order to obtain a simplification of the above formulae ((13) to (17)), we define Z_ω by:

$$Z_\omega := XY + \omega S_\mu,$$

which is

$$\begin{aligned} Z_\omega = XY + \omega S_\mu &= XY + \omega(Z - XY) \\ &= \omega Z + (1-\omega)XY. \end{aligned}$$

We can easily prove that $X_+(\omega) = Z_\omega Y^\dagger$, like the first Gauss-Seidel formula but using Z_ω instead of Z. In fact, $Z_\omega Y^\dagger = \omega ZY^\dagger + (1-\omega)XYY^\dagger = \omega X_+ + (1-\omega)X$. The other two correspondent relations are not directly achieved. For instance, we would like to show that $Y_+(\omega) = X_+(\omega)^\dagger Z_\omega$, but we only accomplish this equivalence assimtoticaly. To see it,

$$\begin{aligned} X_+(\omega)^\dagger Z_\omega &= X_+(\omega)^\dagger(\omega Z + (1-\omega)XY) \\ &= \omega Y_+ + (1-\omega)\tilde{Y}, \end{aligned}$$

where $\tilde{Y} = X_+(\omega)^\dagger XY$. Since X and $X_+(\omega)$ are iterates to close each other, the limit $\tilde{Y}$ approaches Y.

This relation allows us to establish the modified SOR scheme, which takes advantages of numerical linear algebra.

Simplified ALM-SOR Updating formulae:

$$X_+(\omega) := Z_\omega Y^\dagger$$
$$Y_+(\omega) := (X_+(\omega))^\dagger Z_\omega$$
$$Z_+(\omega) := X_+(\omega)Y_+(\omega) + \frac{\mu}{1+\mu}\mathcal{P}_\Omega(M - X_+(\omega)Y_+(\omega) - \frac{1}{\mu}\Lambda).$$

Now we need a strategy for choosing weight values ω, adjusting it dynamically according to the change of the objective values. After a point $(X_+(\omega), Y_+(\omega), Z_+(\omega))$ is computed, we calculate the residual ratio

$$\gamma(\omega) = \frac{\|S_{\mu+}(\omega)\|_F}{\|S_\mu\|_F}, \tag{18}$$

where $S_{\mu+}(\omega) := \frac{\mu}{1+\mu}P_\Omega(M - X_+(\omega)Y_+(\omega) - \frac{1}{\mu}\Lambda)$. If $\gamma(\omega) < 1$, this new point is accepted and the step is called "successful"; otherwise the step is "unsuccessful", ω is reset to 1 and the procedure is repeated. Note that small $\gamma(\omega)$ indicates that the current weight value ω works well so far. Hence, ω is increased only if the calculated point is acceptable but the residual ratio is considered "too large"; that is, $\gamma(\omega) \in [\gamma_1, 1)$ for some $\gamma_1 \in (0, 1)$. From the above considerations we arrive at the following algorithm.

ALM-SOR

INPUT: M

1. $\Lambda_0 = 0$; $Y_0 = I$; $Z_0 = \mathcal{P}_\Omega(M)$; $\mu_0 = 1$; $\rho > 1$; $k = 0$; $\omega = 1$; $\widetilde{\omega} > 1$; $\delta > 0$; $\gamma_1 \in (0, 1)$.
2. While not convergent, do
3. **lines 4-12 approximately solve $(X_{k+1}, Y_{k+1}, Z_{k+1}) := argmin_{X,Y,Z}\mathcal{L}_{\mu_k}(X, Y, Z, \Lambda_k)$;
4. While not convergent, do
5. $X_k^{l+1}(\omega) := orth((Z_k^l(\omega))_\omega (Y_k^l)^T)$;
6. $Y_k^{l+1}(\omega) := ((X_k^{l+1}(\omega))^\top (Z_k^l(\omega))_\omega$;
7. $Z_k^{l+1}(\omega) := X_k^{l+1}(\omega)Y_k^{l+1}(\omega) + \frac{\mu_k}{1+\mu_k}\mathcal{P}_\Omega(M - X_k^{l+1}(\omega)Y_k^{l+1}(\omega) - \frac{1}{\mu_k}\Lambda_k)$;
8. calculate $\gamma(\omega)$ according to (18).

9. if $\gamma(\omega) \geq 1$ then do $\omega = 1$ and go back to step 5.
10. $l := l + 1$;
11. if $\gamma(\omega) \geq \gamma_1$ then do $\delta = \max(\delta, 0.25(\omega - 1))$ and $\omega = \min(\omega + \delta, \widetilde{\omega})$
12. End (While).
13. $(X_{k+1}, Y_{k+1}, Z_{k+1}) := (X_k^l(\omega), Y_k^l(\omega), Z_k^l(\omega))$;
14. $\Lambda_{k+1} := \Lambda_k + \mu_k \mathcal{P}_\Omega(Z_{k+1} - M)$;
15. $\mu_{k+1} := \rho \mu_k$
16. $k := k + 1$;
17. End (while).

OUTPUT: $(X_k, Y_k, Z_k), \Lambda_k$.

Notice that in the above algorithm, the Augmented Lagrangean function minimization is solved between lines 4 and 12, by using nonlinear SOR.

4 Convergence analysis

In this section we shall prove convergence results of our algorithms. We begin by studying the ALM-GS. Our first lemma shows that the internal loop provides a KKT point for the problem $\min_{X,Y,Z} \mathcal{L}_{\mu_k}(X, Y, Z, \Lambda_k)$.

Lemma 1. *Let* $\{(X_k^l, Y_k^l, Z_k^l)\}$ *be the inner sequence generated by the ALM-GS algorithm. If* $\lim_{l\to\infty}(X_k^{l+1}, Y_k^{l+1}, Z_k^{l+1}) - (X_k^l, Y_k^l, Z_k^l) = 0$, *then any accumulation point* (X_k, Y_k, Z_k) *of* $\{(X_k^l, Y_k^l, Z_k^l)\}$ *is a KKT point of the problem* $\min_{X,Y,Z} \mathcal{L}_{\mu_k}(X, Y, Z, \Lambda_k)$.

Proof. Since X_k^{l+1} solves $\min_X \mathcal{L}_{\mu_k}(X, Y_k^l, Z_k^l, \Lambda_k)$, it satisfies $(X_k^{l+1}Y_k^l - Z_k^l){Y_k^l}^T = 0$. Now by merging $X_k^{l+1} = X_k^l + (X_k^{l+1} - X_k^l)$ in the last equation we obtain

$$(X_k^l Y_k^l - Z_k^l){Y_k^l}^T = -(X_k^{l+1} - X_k^l)Y_k^l{Y_k^l}^T,$$

which leads to $(X_k Y_k - Z_k)Y_k^T = 0$ when $l \to \infty$. Similarly, $Y_k^{l+1} = \text{argmin}_Y \mathcal{L}_{\mu_k}(X_k^{l+1}, Y, Z_k^l, \Lambda_k)$ implies $(X_k^{l+1})^T(X_k^{l+1}Y_k^{l+1} - Z_k^l) = 0$, which in turn provides

$$(X_k^{l+1})^T(X_k^{l+1}Y_k^{l+1} - Z_k^{l+1}) = -(X_k^{l+1})^T(Z_k^{l+1} - Z_k^l),$$

obtaining $(X_k)^T(X_kY_k - Z_k) = 0$ in the limit. The fact that $Z_k^{l+1} = \text{argmin}_Z \mathcal{L}_{\mu_k}(X_k^{l+1}, Y_k^{l+1}, Z, \Lambda_k)$ implies $Z_k^{l+1} - X_k^{l+1}Y_k^{l+1} + P_\Omega(\Lambda_k + \mu_k(Z_k^{l+1} - M)) = 0$ leading to the third condition $Z_k - X_kY_k + P_\Omega(\Lambda_k + \mu_k(Z_k - M)) = 0$ in the limit. □

The next result establishes that whenever the ALM-GS algorithm converges, every accumulation point is a KKT point for problem (2).

Theorem 1. *Let* $\{(X_k, Y_k, Z_k, \Lambda_k)\}$ *be the outer sequence generated byALM-GS algorithm. Suppose that* $\lim_{k\to\infty}(X_{k+1}, Y_{k+1}, Z_{k+1}, \Lambda_{k+1}) - (X_k, Y_k, Z_k, \Lambda_k) = 0$. *Then every accumulation point* $(X^*, Y^*, Z^*, \Lambda^*)$ *of the sequence* $\{(X_k, Y_k, Z_k, -\Lambda_k)\}$ *is a KKT point for problem (2).*

Proof. The lemma above provides us the first two KKT conditions (4) and (5). The optimality of Z_{k+1} together with the updating formula for Λ_{k+1} implies that

$$P_\Omega(Z_{k+1} - X_{k+1}Y_{k+1} + \Lambda_{k+1}) = 0,$$
$$P_{\Omega^C}(Z_{k+1} - X_{k+1}Y_{k+1}) = 0,$$

which leads to $P_\Omega(Z^* - X^*Y^* + \Lambda^*) = 0$ and $P_{\Omega^C}(Z^* - X^*Y^*) = 0$. The remaining condition comes easily from $\Lambda_{k+1} - \Lambda_k = \mu_k P_\Omega(Z_{k+1} - M)$ which goes to zero when k grows, obtaining (7) in the limit. □

Now we shall study the ALM-SOR algorithm, which exhibits better convergence properties. We first prove that the sequence of Lagrange multipliers is bounded:

Proposition 1. *The sequence* $\{\Lambda_k\}$ *generated by ALM-SOR algorithm is bounded.*

Proof. Let us consider an inner iterate (X_k^l, Y_k^l, Z_k^l) of the algorithm. The third updating formula results in $Z_k^l - X_k^lY_k^l = \frac{\mu_k}{1+\mu_k}P_\Omega(M - X_k^lY_k^l - \frac{1}{\mu_k}\Lambda_k)$. Denote by r_k^l the size of the residual, that is $r_k^l = \|\frac{\mu_k}{1+\mu_k}P_\Omega(M - X_k^lY_k^l - \frac{1}{\mu_k}\Lambda_k)\|_F$, which in turn leads to $r_k^l = \|Z_k^l - X_k^lY_k^l\|_F$. Since at the $l-$th iteration of the algorithm it satisfies $\gamma_k^{l+1} = \frac{r_k^{l+1}}{r_k^l} \leq 1$, there is

$$r_k^{l+1} \leq r_k^l \leq \cdots \leq r_k^0.$$

Now, denoting $r_{k+1} = \|Z_{k+1} - X_{k+1}Y_{k+1}\|_F = \lim_{l\to\infty}\|Z_k^l - X_k^lY_k^l\|_F = \lim_{l\to\infty} r_k^l$. Noting that $r_k = \|Z_k - X_kY_k\|_F = \|Z_k^0 - X_k^0Y_k^0\|_F$, we can obtain

$$r_{k+1} \leq r_k \leq \cdots \leq r_0,$$

showing the boundedness of $\{P_\Omega(Z_k - X_kY_k)\}$. Now, since Z_{k+1} satisfies

$$Z_{k+1} - X_{k+1}Y_{k+1} + \Lambda_k + \mu_k P_\Omega(Z_{k+1} - M) = 0,$$

we obtain $Z_{k+1} - X_{k+1}Y_{k+1} + \Lambda_{k+1} = 0$ by using the Lagrange Multipliers updating formula, obtaining our result. □

The next result ensures the boundedness of other sequences generated by the ALM-SOR algorithm.

Proposition 2. *Let $\{X_k\}, \{Y_k\}$ and $\{Z_k\}$ be the sequences generated by Algorithm ALM-SOR. Suppose that the sequence $\{P_{\Omega^C}(X_kY_k)\}$ is bounded. Then, the sequences$\{X_k\}, \{Y_k\}$, $\{Z_k\}$ and $\{\mathcal{L}_{\mu_k}(X_k, Y_k, Z_k, \Lambda_k)\}$ are all bounded.*

Proof. By the construction of the inner iterations we have

$$\begin{aligned}\mathcal{L}_{\mu_k}(X_k^{l+1}, Y_k^{l+1}, Z_k^{l+1}, \Lambda_k) &\leq \mathcal{L}_{\mu_k}(X_k^{l+1}, Y_k^{l+1}, Z_k^{l}, \Lambda_k)\\ &\leq \mathcal{L}_{\mu_k}(X_k^{l+1}, Y_k^{l}, Z_k^{l}, \Lambda_k)\\ &\leq \mathcal{L}_{\mu_k}(X_k^{l}, Y_k^{l}, Z_k^{l}, \Lambda_k).\end{aligned}$$

In consequence,

$$\begin{aligned}\mathcal{L}_{\mu_k}(X_{k+1}, Y_{k+1}, Z_{k+1}, \Lambda_k) &\leq \mathcal{L}_{\mu_k}(X_k, Y_k, Z_k, \Lambda_k)\\ &= \tfrac{1}{2}\|X_kY_k - Z_k\|_F^2 + \langle \Lambda_k, P_\Omega(Z_k - M)\rangle + \tfrac{\mu_k}{2}\|P_\Omega(Z_k - M)\|_F^2.\end{aligned}$$

Since $\Lambda_k = \Lambda_{k-1} + \mu_{k-1}P_\Omega(Z_k - M)$ we obtain

$$\mathcal{L}_{\mu_k}(X_{k+1}, Y_{k+1}, Z_{k+1}, \Lambda_k) \leq \mathcal{L}_{\mu_{k-1}}(X_k, Y_k, Z_k, \Lambda_{k-1}) + \frac{\mu_k + \mu_{k-1}}{2\mu_k^2}\|\Lambda_k - \Lambda_{k-1}\|_F^2.$$

By the updating formula for μ_k, and the fact that $\rho > 1$ we have

$$\sum_{k=1}^{\infty} \frac{\mu_k + \mu_{k-1}}{2\mu_k^2} \leq 2\sum_{k=1}^{\infty} \frac{\mu_{k-1}}{2\mu_k^2} = \frac{1}{\mu_0\rho}\sum_{k=1}^{\infty}\frac{1}{\rho^k} < \infty. \tag{19}$$

Then, the sequence $\{\mathcal{L}_{\mu_k}(X_{k+1}, Y_{k+1}, Z_{k+1}, \Lambda_k)\}$ is bounded due to the boundedness of $\{\Lambda_k\}$ and (19).

On the other hand, we have

$$\begin{aligned}\Lambda_k = P_\Omega(X_kY_k - Z_k) &= P_\Omega(X_kY_k - M) + P_\Omega(M - Z_k)\\ &= P_\Omega(X_kY_k - M) - \tfrac{1}{\mu_k}(\Lambda_k - \Lambda_{k-1}).\end{aligned}$$

The boundedness of the Lagrange multipliers and the assumption on $\{P_{\Omega^C}(X_kY_k)\}$ imply that $\{(X_kY_k)\}$ is bounded. Now, it follows from $X_k^{l+1} = orth((Z_k^l)_\omega {Y_k^l}^T)$ that the sequence $\{X_k\}$ is bounded, which implies that $\{Y_k\}$ and $\{Z_k\}$ are also bounded. □

Theorem 2. *Every accumulation point* (X_*, Y_*, Z_*) *of the sequence* $\{(X_k, Y_k, Z_k, \Lambda_k)\}$ *generated by ALM-SOR Algorithm is a KKT point for problem (2); and the convergence rate is at least* $O(\mu_k^{-1})$ *in the sense that* $\mathcal{L}_{\mu_{k-1}}(X_k, Y_k, Z_k, \Lambda_{k-1}) - \frac{1}{2}\|X_kY_k - Z_k\|_F^2 = O(\mu_k^{-1})$.

Proof. In a similar manner as in Lemma 1 we can demonstrate that the iterate (X_k, Y_k, Z_k) satisfies (4) and (5). Hence, in the limit these conditions are also satisfied. In addition,

$$P_\Omega(Z_{k+1} - X_{k+1}Y_{k+1}) + \Lambda_k + \mu_k P_\Omega(Z_{k+1} - M) = 0,$$
$$P_{\Omega^C}(Z_{k+1} - X_{k+1}Y_{k+1}) = 0.$$

The first equation, and the updating formula for Λ_k provide $P_\Omega(Z_{k+1} - X_{k+1}Y_{k+1} + \Lambda_{k+1}) = 0$, which gives us (6) in the limit. Condition (7) is obtained from the boundedness of $\{\Lambda_k\}$ and $\Lambda_{k+1} - \Lambda_k = \mu_k P_\Omega(Z_{k+1} - M)$. On the other hand, the relation

$$\frac{1}{2}\|X_kY_k - Z_k\|_F^2 - \mathcal{L}_{\mu_{k-1}}(X_k, Y_k, Z_k, \Lambda_{k-1}) = -\frac{1}{2\mu_{k-1}}(\|\Lambda_k\|_F^2 - \|\Lambda_{k-1}\|_F^2)$$

and the boundedness of $\{\Lambda_k\}$ imply the convergence rate result. □

5 Computational experiments

5.1 Randomly generated matrices

In this subsection we build rank r matrices $M \in \mathbb{R}^{m\times n}$ as follows: First we generate two random matrices $M_L \in \mathbb{R}^{m\times r}$ and $M_R \in \mathbb{R}^{n\times r}$ from a Gaussian distribution, and then we assemble the matrix $M = M_L M_R^T$. After that, we define the set Ω with p random entries chosen from an uniform distribution. The ratio $\frac{p}{(mn)}$ between the number of measurements and the total number of the matrix entries is denoted by "**sr**" (Sampling ratio). The ratio $\frac{r(m+n-r)}{p}$ between the degree of freedom in a rank r matrix and the number of measurements is denoted by "fr".

Our first experiment has the aim of comparing the **LMaFit**, **ALM-GS** and **ALM-SOR** algorithms in a diverse collection of problems. From very small problems where the size of the matrix is $m = n = 10$, passing through middle sized matrices ($m = n = 100, 500$) and the larger ones ($m = n = 1000, 2000, 5000$). In all the tests we randomly generate the problems and choose the sampling ratio from the set $\{0.04, 0.8, 0.15, 0.20, 0.25,$-

$0.30, 0.35, 0.40, 0.50, 0.55\}$. The smaller the **sr** value the harder the problem. Besides, for the three algorithms, we solve problems with initial values for the rank estimate K of $K = [1.25r]$ and $K = [1.5r]$. At the tables we show results about the following quantities: "Est.rk" (representing the rank of the matrix obtained for each algorithm), "time" (cpu time), Rel.err (relative error) and "iter" (number of iterations). The number of iterations for all the algorithms are calculated with respect to the internal iterations, in which a QR factorization is performed.

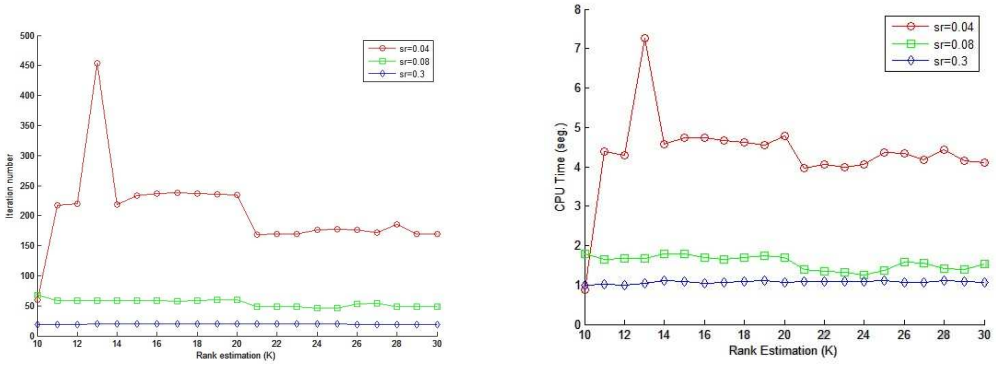
In the implementation we use a rank estimation heuristic firstly employed by LMaFit in [3]: first we start from large K ($K > r$) and decrease it until a dramatic change in the estimated rank of the variable X is detected, based on its QR factorization, which usually occurs after a few iterations (see [3] for more details).

We first examine the sensitivity of our **ALM-SOR** algorithm with respect to the initial rank estimate K. In this test we use matrices with size $m = n = 1000$ and rank $r = 10$. We generate three tests for three different sr values, that is for $sr = 0.04$, $sr = 0.08$ and $sr = 0.3$ respectively. In each case, we ran our AML-SOR algorithm for each $K = 10, 11, 12, ..., 30$. In the figure 1 we show the number of iterations and the CPU time for each of the K values. We use the parameters $rho = 2$, $miu = 10$ and $reschgxy < 1 * tol$. We observe stability on the number of iterations and the CPU time in changes on the initial estimate for the rank. Only in the very sparse problem the graphic exhibits a peak. We can say that there is no sensitivity on the initial rank estimate K.

5.2 ALM-GS vs ALM-SOR

Now we compare the performance of the ALM-GS vs ALM-SOR in randomly generated problems. The results are shown in Table 1. We can observe that the quality of the solutions in terms of the relative error, and the rank estimation are similar, but ALM-SOR is consistently faster in CPU time and number of iterations.

Tables 2 and 3 show a comparison between LMaFit and **ALM-SOR** for problems of size $m = n = 100$ and 500 respectively. In these tables we see that the quality of the solutions are similar (in terms of relative error and rank estimation). For example, when the estimated rank is well calculated, both of the algorithms provided the same estimation, and when it fails, both of them do (see table 2, row 5). Comparison about performance comes in favor



(a) Number of iterations (b) CPU time

Fig. 1: Number of iterations and CPU time

Problem			**ALM-GS**				**ALM-SOR**			
m	r	sr	est.rk	Time	Rel.err	Iter	est.rk	Time	Rel.Err	Iter
10	2	0.15	2	0.0241	0.1438	57	2	0.0285	0.1438	79
10	2	0.25	2	0.0705	6.0471e-4	500	2	0.0918	1.1849e-4	323
10	2	0.35	2	0.0748	0.0031	500	3	0.0414	1.1687e-4	41
10	2	0.55	1	0.0253	6.2132e-5	110	1	0.0140	0.3553	111
100	10	0.15	13	0.2839	1.2494e-4	363	4	0.0887	0.3739	104
100	10	0.25	1	0.0629	0.8540	87	1	0.0572	0.8533	63
100	10	0.35	10	0.2711	1.2312e-4	307	10	0.1503	1.2201e-4	153
100	10	0.55	10	0.1456	1.2326e-4	78	10	0.0316	9.9925e-5	28
1000	30	0.15	30	12.243	1.2212e-4	205	30	3.0601	1.1327e-4	52
1000	30	0.25	30	4.4785	1.2392e-4	65	30	1.5223	9.4369e-5	23
1000	30	0.35	30	2.8779	1.0765e-4	36	30	1.4004	6.2161e-5	18
1000	30	0.55	30	1.7515	8.5597e-5	17	30	1.3276	6.8091e-5	12
1000	15	0.04	12	11.629	0.2414	500	15	13.750	4.5372e-4	500
1000	15	0.1	15	8.3452	1.2467e-4	237	15	1.4792	1.0250e-4	42
2000	15	0.1	15	34.805	1.2193e-4	188	15	6.9849	9.8830e-5	38
2000	15	0.15	15	23.293	1.2193e-4	102	15	4.0600	1.2460e-4	21

Tab. 1: ALM-GS vs ALM-SOR

of our **ALM-SOR**, consistently, in terms of number of iterations. In Table 4 we provide results for larger problems, comparing the three algorithms. Although the quality of the solutions are acceptable for **ALM-GS**, it is consistently slower when compared to the other two methods. In general, our **ALM-SOR** performs better than the other two.

5.3 Real data problems

5.3.1 Jester jokes and Movie Lens.

In this subsection we consider low rank problems coming from real data: The *Jester joke* [10] and *MovieLens* [11] libraries. In both data sets , only certain quantity of the entries are available. Both of the problems, *Jester Joke* and *MovieLens* are example of problems in the area of systems of recommendation. The first one is a list of jokes presented to a group of people who should assign a qualification to each one. Since the list of jokes is large, each user can qualify only a subset of the jokes. With the qualifications a data matrix is built, where the rows represents users, the columns represents jokes and (i, j) entry is the qualification that user i give to joke j. The second problem (MovieLens), deals with a set of movies (columns of the data matrix) and users which rate the movies (rows of the data matrix), but users typically rate very few movies so that there are very few scattered observed entries of this data matrix. The objective is to complete the matrix so that the vendor might recommend titles that any particular user is likely to be willing to order.

The *Jester Joke* data base consists of four problems "jester-1", "jester-2", "jester-3" and "jester-all", where the last one is a combination of the first three. *MovieLens* contains the problems "movie-100k", "movie-1M" and "movie-10M". This data is available in **www.ieor.berkeley.edu/∼Egoldberg/jester-data** and **www.grouplens.org** respectively. For the algorithms **LMaFit**, and **ALM-SOR** the set of parameter are $tol = 10^{-3}$, $est\_rank = 2$, $K = 1$ y $rk\_ink = 2$. Besides, we choose 60 as a maximum guess for the rank estimator for *Jester joke* and 100 for the *MovieLens* for both of the solvers. Since the entries of the matrix are known only on Ω, to measure accuracy we calculate the normalized absolute error (NMAE) by:

$$NMAE = \frac{1}{(r_{max} - r_{min})|\Omega|} \sum_{(i,j)\in\Omega} |W_{ij} - M_{ij}|$$

Problem				LMaFit								ALM-SOR							
miu	rho	r	sr	$K=\lceil 1.25r\rceil$				$K=\lceil 1.5r\rceil$				$K=\lceil 1.25r\rceil$				$K=\lceil 1.5r\rceil$			
				Est.rk	Time	Rel.Err	Iter	Est.rk	Time	Rel.Err	Iter	Est.rk	Time	Rel.Err	Iter	Est.rk	Time	Rel.Err	Iter
12	2	2	0.04	1	0.025	0.3839	106	1	0.064	0.3961	153	1	0.0434	0.3796	96	1	0.0917	0.3946	130
12	2	2	0.08	1	0.037	0.5839	102	2	0.171	1.487e-4	500	1	0.0385	0.5857	52	2	0.1667	1.2459e-4	339
100	4	2	0.15	1	0.0141	0.6217	32	2	0.050	1.219e-4	67	1	0.0314	0.6221	24	2	0.0590	1.1729e-4	67
100	10	2	0.30	1	0.026	0.5892	13	2	0.017	1.189e-4	26	1	0.0297	0.5892	13	2	0.0338	9.8973e-5	25
100	10	10	0.20	7	0.159	0.1942	253	1	0.083	0.8483	65	7	0.1146	0.2027	140	1	0.0610	0.8483	65
1000	1.5	10	0.25	10	0.274	2.719e-4	500	9	0.140	0.1199	197	10	0.2853	1.2268e-4	410	10	0.2533	1.2246e-4	282
1000	1.5	10	0.30	10	0.227	1.249e-4	324	10	0.247	1.244e-4	395	10	1.388	1.2241e-4	149	10	0.1795	1.1834e-4	190
1000	1.5	10	0.40	10	0.100	1.155e-4	98	10	0.102	1.245e-4	113	10	0.0888	1.1193e-4	76	10	0.0995	1.235e-4	78
100	10	20	0.35	2	0.074	0.8363	64	3	0.075	0.7632	66	2	0.0958	0.8363	65	3	0.0932	0.7632	66
100	1.5	20	0.40	6	0.083	0.5763	72	29	0.241	1.242e-4	255	21	0.6006	0.0185	500	29	0.2463	1.2247e-4	139
100	1.5	20	0.50	20	0.160	1.187e-4	158	19	0.124	0.0594	105	20	0.1281	1.1886e-4	89	20	0.1573	1.0840e-4	99
500	1.5	20	0.55	20	0.123	1.193e-4	105	20	0.123	1.139e-4	117	20	0.1124	1.1420e-4	74	20	0.1373	1.1126e-4	84

Tab. 2: Problems with $m = n = 100$

Problem				LMaFit								ALM-SOR							
miu	rho	r	sr	$K = \lceil 1.25r \rceil$				$K = \lceil 1.5r \rceil$				$K = \lceil 1.25r \rceil$				$K = \lceil 1.5r \rceil$			
				est.rk	Time	Rel.err	Iter	est.rk	Time	Rel.err	Iter	est.rk	Time	Rel.err	Iter	est.rk	Time	Rel.err	Iter
12	1.01	10	0.04	4	0.619	0.4954	117	5	0.733	0.3891	163	4	0.4511	0.5086	77	11	3.2904	0.0513	500
5	1.2	10	0.08	10	1.176	1.225e-4	200	10	1.081	1.209e-4	208	10	0.8381	1.245e-4	97	10	0.9786	1.1635e-4	125
5	1.2	10	0.15	10	0.286	1.059e-4	38	10	0.325	1.136e-4	42	10	0.3054	1.165e-4	33	10	0.5351	8.6751e-5	48
5	2.7	10	0.3	10	0.177	1.231e-4	20	10	0.212	1.005e-4	21	10	0.2253	6.858e-5	18	10	0.2251	5.095e-5	18
10	1.1	50	0.20	61	6.959	0.0014	500	7	1.070	0.7964	74	61	3.4787	1.1623e-4	165	7	1.3305	0.7968	74
15	1.2	50	0.25	50	4.448	1.223e-4	340	48	2.027	0.0691	151	50	2.4482	1.1714e-4	118	48	2.1932	0.0679	93
10	1.4	50	0.30	50	1.723	1.242e-4	119	50	2.292	1.188e-4	158	50	1.2962	1.1743e-4	59	50	2.4598	1.0150e-4	85
10	2	50	0.40	50	0.738	1.184e-4	48	50	0.960	1.104e-4	57	50	0.7228	1.1566e-4	28	50	1.1937	1.0773e-4	40
100	1.1	100	0.35	13	1.639	0.7841	62	15	1.780	0.7546	65	117	7.4423	1.2497e-4	165	146	22.366	0.0317	500
300	3	100	0.40	67	3.242	0.2261	142	59	2.959	0.2905	135	99	8.8866	0.0219	257	59	6.8179	0.2905	239
300	1.5	100	0.50	100	3.445	1.198e-4	135	100	3.910	1.233e-4	147	100	3.6655	1.1294e-4	97	100	4.9047	1.1209e-4	105
300	1.1	100	0.55	100	1.886	1.189e-4	66	100	3.340	1.209e-4	103	100	2.2733	1.0122e-4	55	100	4.5079	1.1851e-4	96

Tab. 3: Middle sized problems $m = 500$

Problema					LMaFit								ALM-GS								ALM-SOR							
m	r	sr	miu	rho	$K = \lceil 1.25r \rceil$				$K = \lceil 1.5r \rceil$				$K = \lceil 1.25r \rceil$				$K = \lceil 1.5r \rceil$				$K = \lceil 1.25r \rceil$				$K = \lceil 1.5r \rceil$			
					est.rk	Time	Rel.err	Iter	est.rk	Time	Rel.err	Iter	est.rk	Time	Rel.err	Iter	est.rk	Time	Rel.err	Iter	est.rk	Time	Rel.err	Iter	est.rk	Time	Rel.err	Iter
1000	25	0.15	100	1.5	25	1.47	1.18e-4	43	25	1.46	1.18e-4	42	25	4.16	1.15e-4	79	25	4.25	1.12e-4	80	25	1.90	1.19e-4	35	25	1.93	1.01e-4	36
1000	25	0.25	100	1.5	25	0.97	1.11e-4	23	25	0.98	1.07e-4	23	25	3.22	9.36e-5	57	25	2.94	1.09e-4	48	25	1.24	2.50e-5	22	25	1.35	9.99e-5	22
1000	25	0.35	100	1.5	25	0.99	9.22e-5	21	25	1.02	1.03e-4	21	25	2.32	4.39e-5	35	25	2.41	4.45e-5	35	25	1.29	4.15e-5	19	25	1.35	4.59e-5	19
1000	25	0.55	100	1.5	25	0.71	1.21e-4	12	25	0.85	9.29e-5	14	25	1.63	2.54e-5	18	25	1.61	2.75e-5	18	25	1.18	3.18e-5	12	25	1.14	4.94e-5	12
2000	30	0.15	100	1.5	30	3.17	1.07e-4	27	30	3.88	9.53e-5	27	30	34.26	1.22e-4	137	30	19.27	1.09e-4	109	30	5.73	1.05e-4	25	30	4.87	5.93e-5	27
2000	30	0.25	100	1.5	30	3.26	1.13e-4	21	30	3.59	9.03e-5	22	30	10.46	1.01e-4	45	30	10.54	1.01e-4	45	30	4.45	4.13e-5	12	30	4.09	4.13e-5	18
2000	30	0.35	100	1.5	30	2.97	1.24e-4	16	30	3.30	8.31e-5	17	30	6.81	8.70e-5	26	30	7.03	8.75e-5	26	30	4.00	6.20e-5	15	30	4.27	7.31e-5	15
2000	30	0.55	1000	10	30	2.34	8.34e-5	10	30	3.06	6.09e-5	13	30	6.06	9.60e-5	16	30	6.28	1.05e-4	16	30	3.69	6.68e-5	10	30	5.35	7.11e-5	14
5000	50	0.15	1000	10	50	27.40	1.06e-4	25	50	28.36	9.29e-5	25	50	217.08	1.20e-4	104	50	241.34	1.19e-4	104	50	35.96	9.51e-5	18	50	41.88	9.24e-5	19
5000	50	0.25	1000	1.2	50	24.90	1.15e-4	22	50	27.68	8.54e-5	23	50	63.76	1.11e-4	41	50	73.36	7.58e-5	44	50	29.00	6.67e-5	19	50	43.15	1.12e-4	25
5000	50	0.35	1000	1.2	50	18.26	1.09e-4	14	50	20.18	6.92e-5	15	50	54.32	7.79e-5	29	50	55.26	7.89e-5	29	50	23.83	5.62e-5	13	50	24.62	6.62e-5	13
5000	50	0.55	1000	1.2	50	14.80	8.55e-5	9	50	18.86	7.63e-5	11	50	42.21	6.36e-5	17	50	43.10	6.83e-5	17	50	21.22	4.75e-5	9	50	33.36	2.91e-5	13

Tab. 4: Large sized problems

where r_{min} and r_{max} are the lower and upper bounds for the rates. Specifically, $r_{min} = -10$, $r_{max} = 10$ for *Jester Joke* problems, and $r_{min} = 1$, $r_{max} = 5$ for the *MovieLens* ones. The results are shown in Table 5.

5.4 Matrix completion in images

In this subsection we apply **LMaFit**, **ALM-GS** and **ALM-SOR** to improve quality of images. The task here is to fill the missing pixel values of an image at given pixel positions that have been determined to contain impulsive noise. This procedure is known as "inpainting" (see [12] and [13]), specifically when the lost pixel positions are randomly distributed. In our experiment we use the Chess original photograph in grayscale of size 266×192 (see figure 2(a), obtained freely from worldwidephotos.org). The image in Figure 2(b) was built from the original by truncating the rank to 40. At the image in Figure 2(c) 50% of the pixel are randomly removed from the original chess. Figure 2(d) represents the reconstruction of image 2(c) by using **LMaFit**; while Figures 2(e) and 2(f) are obtained by **ALM-GS** and **ALM-SOR** respectively. On the other hand we show in Figure 3(a) the image obtained by extracting 50% of the pixels to the 40 rank image (figure 2(b)), and Figures 3(b), 3(c) and 3(d) show its respective reconstructions.

In the above images we can appreciate that the approximations generated from image 2(c) are not so good in quality of image when compared with the original 2(a). This is because the original matrix is not a low rank matrix. However, if we appreciate, images 3(b), 3(c) and 3(d), they are very good approximations of the image in 2(c). This can be explained because the image in 2(c) is built with low rank (equal 40). This last comment sugest that **LMaFit** and **ALM-SOR** are specially good algorithms when the matrices to be reconstructed present originally low rank.

6 Concluding remarks

We deal with the low-rank matrix completion problem. Even though this NP-hard problem has been approximated by the convex nuclear norm relaxation, we settle it through matrix factorization, by solving the nonconvex relaxation (2). Despite the nonconvexity of this model, its special structure allows us to pursue KKT solutions in an alternating scheme in which three convex quadratic subproblems are solved, in each iteration. Each one

Problema			LMaFit					ALM-GS								ALM-SOR							
name	m	n	Time	Iter	est.rk	Rel.err	NMAE	Time	Iter	est.rk	Rel.err	miu	rho	reschagxy	NMAE	Time	Iter	est.rk	Rel.err	miu	rho	reschagxy	NMAE
Jester-1	24983	100	22.43	117	80	0.186	0.024	43.95	111	80	0.174	100	3	10xtol	0.023	40.19	109	80	0.263	100	3	10xtol	0.043
Jester-2	23500	100	20.85	118	80	0.186	0.025	37.75	114	80	0.176	1000	10	1xtol	0.023	34.48	111	80	0.188	1000	10	1xtol	0.026
Jester-3	24938	100	24.59	235	55	9.312e-4	4.054e-5	77.71	392	39	9.927e-4	100	10	0.1xtol	3.410e-5	43.39	236	55	9.273e-4	100	10	0.1xtol	4.137e-5
Jester-all	73421	100	55.38	114	80	0.164	0.020	95.67	117	80	0.165	100	10	0.1xtol	0.020	77.51	101	80	0.179	20	2	0.6xtol	0.023
Movie-100k	943	1682	20.43	507	100	2.1e-3	9.952e-4	50.147	507	100	2.6e-3	20	5	0.03xtol	1.4e-3	21.279	279	100	6.1e-3	20	5	0.03xtol	3.5e-3
Movie-1M	6040	3706	46.715	170	100	0.1022	7.10e-2	189.193	206	100	0.1040	20	3	0.7xtol	7.45e.2	101.649	141	100	0.1114	20	3	0.7xtol	7.96e-2

Tab. 5: Jester Jokes and Movie Lens

of this convex quadratic optimization problems leads us to closed formulas. We prove that the sequence of Lagrange Multipliers generated by ALM-SOR algorithm is bounded, and that the accumulation points verify the KKT conditions. Our numerical experiments show that ALM-SOR is competitive to deal with simulated data problems, and also with real data.

Acknowledgements

We would like to thank CDCHT-UCLA-Venezuela for their support, and CNPq-Brasil, though Projeto Prosul.

References

[1] Candes E.J. Cai J-F. and Shen Z. A singular value thresholding algorithm for matrix completion. Technical Report arXiv:0810.3286, Cornell University Library, 2008. http://arxiv.org/pdf/0810.3286.pdf.

[2] Candés E. and Recht B. Exact matrix completion via convex optimization. *Foundation of Computational Mathematics*, 9:717–772, 2009.

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

[4] Golub G.H. and Van Loan C.F. *Matrix Computations*. The Johns Hopkins University Press, 3 edition, 1996.

[5] Grippo L. and Sciandrone M. On the convergence of the block nonlinear gauss-seidel method under convex constraints. *Oper. Res. Lett.*, 26:127–136, 2000.

[6] Bertsekas D. *Constrained Optimization and Lagrange Multiplier Method*. Academic Press, 1982.

[7] Birgin E.G. and Martínez J.M. *Practical Augmented Lagrangian Methods for Constrained Optimization*. Society for Industrial and Applied Mathematics, Philadelphia, USA, 2014.

[8] Zhang Y. An alternating direction algorithm for nonnegative matrix factorization. Technical report, Dept. of Computational and Applied Mathematics, Rice University, 2010. Houston, TX 7705.

[9] Lin Z., Chen M., Wu L., and Ma Y. The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. To appear, 2015.

[10] Goldberg K., Roeder T., Gupta D., and Perkins C. Eigentaste: A constant time collaborative filtering algorithm. *Inf. Retr.*, 4:133–155, 2001.

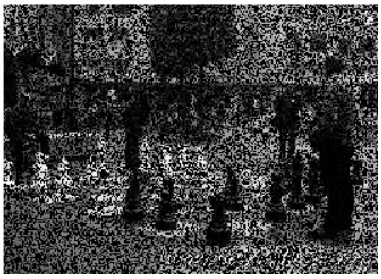
[11] Herlocker J.L., Konstan J.A., Borchers A., and Riedi J. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 230–237, New York, USA, 1999. ACM.

[12] Morita T. and Kanade T. A sequential factorization method for recovering shape and motion from image streams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:33–45, 1997.

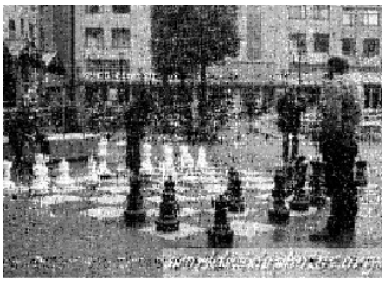
[13] Tomasi C. and Kanade T. Shape and motion from image streams under orthography: a factorization method. *Int. J. Comput. Vision*, 9:137–154, 1992.

(a) Original image

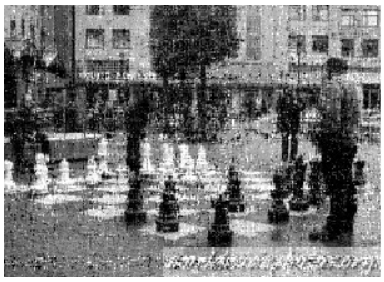
(b) Rank 40% of original image



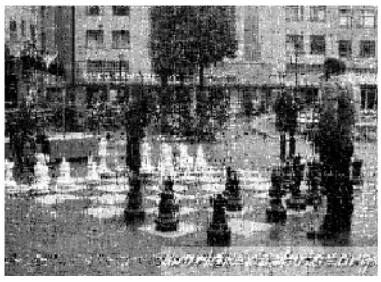
(c) Corrupted image at 50% from original



(d) Solution from LMaFit

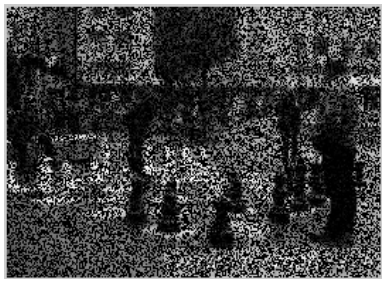


(e) Approach from ALM-GS



(f) Approach from ALM-SOR

Fig. 2: Chess images



(a) Corrupted image at 50% from image at rank 40

(b) Approach from LMaFit

(c) Approach from ALM-GS

(d) Approach from ALM-SOR

Fig. 3: Chess images 2