Robust Low-Rank Matrix Completion by Riemannian Optimization

Léopold Cambier[*], *P.-A. Absil*[†]

Abstract

Low-rank matrix completion is the problem where one tries to recover a low-rank matrix from noisy observations of a subset of its entries. In this paper, we propose RMC, a new method to deal with the problem of *robust* low-rank matrix completion, i.e., matrix completion where a fraction of the observed entries are corrupted by non-Gaussian noise, typically outliers. The method relies on the idea of smoothing the ℓ_1 norm and using Riemannian optimization to deal with the low-rank constraint. We first state the algorithms as the successive minimization of smooth approximations of the ℓ_1 norm and we analyze its convergence by showing the strict decrease of the objective function. We then perform numerical experiments on synthetic data and demonstrate the effectiveness on the proposed method on the NETFLIX dataset.

Keywords Low-Rank Matrix Completion, Riemannian optimization, outliers, smoothing techniques, ℓ_1 norm, non-smooth, fixed-rank manifold.

1 Introduction

The problem of low-rank matrix completion has drawn large interest in the past decade. It can be used as a building block for recommender systems, where one wants to predict users ratings based on partial ratings using collaborative filtering (Bennett & Lanning, 2007), in reconstructing 3D path of particles from only partial observation using a fixed camera (Kennedy *et al.*, 2014), in sensor network localization (Drineas *et al.*, 2006; So & Ye, 2007; Oh *et al.*, 2010) or image inpainting where low-rank completion is used as a way to reconstruct damaged image (Peng *et al.*, 2012).

Low-Rank Matrix Completion consists of recovering a low-rank matrix of size $m \times n$ from only a fraction (typically $\mathcal{O}(m+n)$ or $\mathcal{O}((m+n)\log(m+n))$) of its entries. Denoting

[*]ICTEAM Institute, Université catholique de Louvain, Louvain-la-Neuve, B-1348, Belgium. `leopoldcambier@gmail.com`

[†]ICTEAM Institute, Université catholique de Louvain, Louvain-la-Neuve, B-1348, Belgium. `pa.absil@uclouvain.be`

by Ω the set of observed entries, the problem can be stated as

$$\begin{aligned} \min_{X \in \mathbb{R}^{m \times n}} \quad & \operatorname{rank}(X) \\ \text{subject to} \quad & \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M) \end{aligned} \tag{1}$$

where $\mathcal{P}_\Omega : \mathbb{R}^{m \times n} \to \mathbb{R}^{m \times n}_\Omega : X \to \mathcal{P}_\Omega X$ is the orthogonal projector onto the space $\mathbb{R}^{m \times n}_\Omega$ of $m \times n$ matrices with zero-entries on $\bar{\Omega} = \{(i,j) | 1 \leq i \leq m, 1 \leq j \leq n\} \backslash \Omega$: $\mathcal{P}_\Omega(X)_{ij} = X_{ij}$ if $(i,j) \in \Omega$ and 0 otherwise. Finally, M is the matrix containing the known entries (with values known only on Ω). This problem, however, is now well known to be NP-hard (Chistov & Grigor'ev, 1984).

Candès & Recht (2009) stated the problem as

$$\begin{aligned} \min_{X \in \mathbb{R}^{m \times n}} \quad & \|X\|_* \\ \text{subject to} \quad & \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M) \end{aligned}$$

where $\| \cdot \|_*$ is the nuclear norm $\|X\|_* = \sum_{k=1}^{\min(m,n)} \sigma_k(X)$ with $\sigma_k(X)$ the k^{th} singular value of X. The authors proved that in the context of *exact* low-rank matrix completion, this formulation recover the original underlying matrix under some mild assumptions.

Another way to approach low-rank matrix completion of it is the following. Assume the rank r of the target matrix is known in advance (which does make sense in a lot of applications like in computer vision, where the rank is often related to the dimension of the space.). In this case, the problem can be stated as

$$\min_{X \in M_r} \|\mathcal{P}_\Omega(X - M)\|_{\ell_2}, \tag{2}$$

where

$$\mathcal{M}_r = \{X \in \mathbb{R}^{m \times n} : \operatorname{rank}(X) = r\}$$

and where $\| \cdot \|_{\ell_2} = \| \cdot \|_F$ is the ℓ_2 or Frobenius norm[1]. Intuitively, this problem seeks the matrix X of rank r that best fits the given data. The main advantage is that it is robust to Gaussian additive noise, in a sense that a small Gaussian additive noise still allows to recover the underlying low-rank matrix with an error proportional to the noise level (Keshavan *et al.*, 2009). This problem and other similar formulations have been addressed by different authors.

Vandereycken (2013) takes advantage of the fact that $\mathcal{M}_r$ is a smooth Riemannian manifold to apply recent optimization algorithms (Absil *et al.*, 2008) to efficiently solve the problem. Our method will use the exact same tools.

All formulations that rely on the Frobenius norm as in (2) suffer from one drawback: even though they are robust to additive Gaussian noise, they are not well suited to recover the underlying low-rank matrix when the noise becomes non-Gaussian. Here

[1] In this paper, we use the notation $\| \cdot \|_{\ell_2}$ for the Frobenius norm to emphasize the difference with the ℓ_1 norm, $\| \cdot \|_{\ell_1}$.

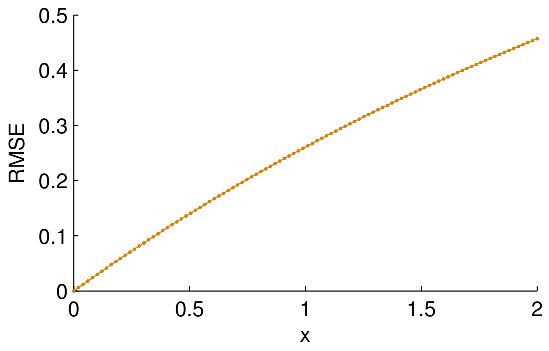


Fig. 1: The error between the recovered matrix and the original one when minimizing the ℓ_2 norm, as a function of x. We can clearly observe that when using an ℓ_2 loss function, even a small perturbation on a single entry of the matrix can lead to a large error with respect to the original matrix.

we focus on the situation where only a few of the observed entries, termed *outliers*, are perturbed; that is,

$$M = M_0 + S, \tag{3}$$

where M_0 is the unperturbed data matrix of rank r and S is a sparse matrix. For instance, consider recovering the best rank-1 approximation of the following matrix

$$M_x = \begin{pmatrix} 2 & -1+x \\ 4 & -2 \end{pmatrix}.$$

If $x = 0$, this matrix is rank-1 since, for instance,

$$M_0 = \begin{pmatrix} 2 & -1 \\ 4 & -2 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix} \cdot \begin{pmatrix} 2 & -1 \end{pmatrix}.$$

But when $x \neq 0$, this is not the case, and finding the rank-1 matrix that minimizes the ℓ_2 error leads to fundamentaly different solutions.

To observe that, we can simply compute, for each x, the rank-1 SVD of M_x (which is the solution an ℓ_2 method minimizing $\|M_x - X\|_{\ell_2}$ would return) and then compute the RMSE with respect to the original matrix M_0. This is depicted in figure 1, and we can see that the error starts to grow as soon as $x \neq 0$.

However, if the method was able to find out that only the top right entry of M_x is corrupted by noise, then it would be able to recover M_0 exactly by removing the top right entry from the mask Ω and performing low-rank matrix completion. More generally, in the problem of completing from its know entries Ω a matrix M generated as in equation (3), the ability of detecting the outliers in $\mathcal{P}_\Omega(M)$ (i.e., the entries affected by the sparse matrix $\mathcal{P}_\Omega(S)$) and removing those entries from the mask Ω would open the way for an exact recovery of the rank-r matrix M_0.

Matrix completion in the presence of outliers has been considered in several papers.

Chen *et al.* (2011) studied the problem of low-rank matrix completion where a large number of columns are arbitrarily corrupted. They showed that only a small fraction of the entries are needed in order to recover the low-rank matrix with high probability, without any assumptions on the location nor the amplitude of the corrupted entries.

Both Li (2013) and Chen *et al.* (2013) studied a harder problem, when a constant fraction of the entries (not the columns) of the matrix are outliers. They studied what conditions need to be imposed in order for the following convex optimization problem

$$\begin{array}{ll} \min & \gamma\|X\|_* + \|E\|_{\ell_1} \\ \text{subject to} & \mathcal{P}_\Omega(X+E) = \mathcal{P}_\Omega(M) \end{array}$$

to exactly recover the underlying low-rank matrix (with $\|\cdot\|_*$ the nuclear norm). Basically, they showed that there exist universal constants such that with overwhelming probability the solution of the problem is equal to M on the mask Ω. In the close context of low-rank PCA, Candès *et al.* (2011) was also able to solve the same problem. The advantage of such an algorithm is that it is convex and can then be analyzed thoroughly.

This robust formulation has been improved to deal with Gaussian noise (Hastie, 2012), leading to the following convex optimization problem

$$\begin{array}{ll} \min & \lambda\|X\|_* + \|E_1\|_{\ell_1} + \|E_2\|_{\ell_2} \\ \text{subject to} & \mathcal{P}_\Omega(X+E_1+E_2) = \mathcal{P}_\Omega(M) \end{array}.$$

He *et al.* (2011) developed a robust version of the GROUSE algorithm (Balzano *et al.*, 2010), named GRASTA, which aims at solving the problem of *robust* subspace tracking. Their algorithm can be casted to solve problems formulated as

$$\begin{array}{ll} \min & \|\mathcal{P}_\Omega(S)\|_{\ell_1} \\ \text{subject to} & \mathcal{P}_\Omega(UV+S) = \mathcal{P}_\Omega(M) \\ & U \in \mathrm{Gr}(m,r) \\ & V \in \mathbb{R}^{r\times n} \end{array}$$

where $\mathrm{Gr}(m,r)$ is the Grassman manifold, i.e., the set of linear r-dimensional subspaces of $\mathbb{R}^m$. GRASTA solves this problem by first building the augmented Lagrangian problem, and it then solves it by alternating between V, his dual variables and U and by performing steepest descent on the Grassman manifold. The advantage of their algorithm is that it is designed to tackle the problem of *online* subspace estimation from incomplete data, hence it can also be casted to solve online low-rank matrix completion where we observe one column of the matrix M at a time.

Nie *et al.* (2012a,b) solved a slightly more general problem where all norms become arbitrary p-norms

$$\min \lambda\|X\|_{S_p}^p + \|\mathcal{P}_\Omega(X-D)\|_{\ell_p}^p,$$

where $\|X\|_{S_p}^p = \sum_{i=1}^{\min(m,n)} \sigma_i^p(X)$ and $\|X\|_{\ell_p}^p = \sum_{i=1,j=1}^{m,n} |X_{ij}|^p$. The algorithm used to solve this non-convex program (when $p < 1$) is, again, an augmented Lagrangian method. We were unfortunately unable to obtain or write an efficient implementation of this algorithm since it requires the storage of the full $m \times n$ matrix, as well as SVD of full matrices of this size. This formulation, however, is efficient for moderate size problems.

Yan *et al.* (2013) solved ℓ_2 problems of the form

$$\min_{X \in \mathcal{M}_r} \|\mathcal{P}_\Omega(X - M)\|_{\ell_2}$$

where the mask Ω is adapted at each iteration to remove the suspected outliers. The idea is to first solve the problem with the original mask Ω, detect outliers, adapt the mask, and then solve the problem again until convergence. Intermediate problems are handled using RTRMC (Boumal & Absil, 2011).

Yang *et al.* (2014) studied the problem of robust low-rank matrix completion using a non-convex loss-function. They solve the following problem

$$\min_{X \in \mathbb{R}^{m \times n} : \operatorname{rank}(X) \leq r} \frac{\sigma^2}{2} \sum_{(i,j) \in \Omega} \left(1 - \exp\left(-(X_{ij} - B_{ij})^2 / \sigma^2\right)\right),$$

where the rank-constraint is relaxed using the now standard nuclear-norm heuristic.

Finally, Klopp *et al.* (2014) studied the optimal reconstruction error in the case of matrix completion, where the observations are noisy and column-wise or element-wise corrupted and where the only piece of information needed is a bound on the matrix entries. They provided a range of (optimal) estimators to solve such problems with guarantees.

In this paper, we consider low-rank matrix completion in the presence of outliers, assuming as in, e.g., Yan *et al.* (2013) that the rank r is known in advance, which naturally leads to the formulation

$$\min_{X \in \mathcal{M}_r} \|\mathcal{P}_\Omega(X - M)\|_{\ell_p}.$$

It remains to choose p. The choice $p = 0$, which consists in maximizing the number of exactly recovered entries over the mask Ω, seems natural when only a sparse noise is present, but this discontinuous objective function is unwieldy, and moreover it is inadequate in the presence of an additional dense (i.e., nonsparse) noise. The choice $p = 2$, as in (Vandereycken, 2013; Boumal & Absil, 2015) would be adequate for Gaussian additive noise, but its mean square nature makes it excessively sensitive to the outliers. We opt for the middle ground, namely $p = 1$. Its well-known sparsity-inducing property lets us expect exact recovery when the noise consists of just a few outliers. We will see in the numerical experiments that this is indeed the case.

The choice $p = 1$ leaves us with a nonsmooth objective function. We handle this difficulty by replacing successively the ℓ_1 norm by increasingly accurate smooth approximations thereof. As in (Yan *et al.*, 2013), the resulting minimization problems over the fixed-rank manifold $\mathcal{M}_r$ are tackled by Riemannian optimization techniques. However, while the

ℓ_2 formulation in (Yan *et al.*, 2013) enables a variable projection strategy that yields an optimization problem on the Grassmann manifold (Boumal & Absil, 2015), our smoothed ℓ_1 objective functions do not lend themselves to this approach. Following the way paved in (Vandereycken, 2013), we resort instead to a conjugate gradient scheme on the fixed-rank manifold $\mathcal{M}_r$ viewed as an embedded Riemannian submanifold of $\mathbb{R}^{m\times n}$. Numerical experiments confirm that the resulting algorithm is particularly efficient in the case where a few percents of the entries are largely corrupted by non-Gaussian noise. Our algorithm can also efficiently handle regularization, and in practice we solve

$$\min_{X\in\mathcal{M}_r} \|\mathcal{P}_\Omega(X-M)\|_{\ell_1} + \lambda\|\mathcal{P}_{\bar\Omega}(X)\|_{\ell_2}^2.$$

The regularization factor appears to be quite useful in applications to limit overfitting. This is justified by the work of Boumal & Absil (2015) where regularization is crucial to handle real datasets. The algorithm also scales well with the size of the problem and stays very efficient when the amplitude of the outliers increase. We finally implemented a simple MATLAB version of the algorithm, able to solve problems of size $50\,000 \times 50\,000$ of rank 10 in a matter of minutes on a regular desktop computer.

This paper is divided in the following way. In section 2 we remind the reader of the essential tools of optimization on manifolds. We introduce our algorithm in section 3. We study its convergence in section 4 while we perform numerical experiments, both synthetic ones and real-life applications, in section 5. Conclusions are drawn in section 6.

2 Optimization on Manifolds and the Low-Rank Matrices Manifold

The *low-rank matrix manifold*

$$\mathcal{M}_r = \{X \in \mathbb{R}^{m\times n} : \operatorname{rank}(X) = r\},$$

where $r \le \min(m,n)$, is known to be a smooth manifold embedded in $\mathbb{R}^{m\times n}$ of dimension $r(m+n-r)$ (Lee, 2003; Vandereycken *et al.*, 2009). Hence, optimization techniques presented in (Absil *et al.*, 2008) can be applied to solve smooth optimization problems where constraints are formulated using $\mathcal{M}_r$.

There are several ways of describing a matrix $X \in \mathcal{M}_r$ (Absil & Oseledets, 2014). In this document, we will use the very natural SVD-like representation

$$X = U\Sigma V^\top \tag{4}$$

where $U \in \mathbb{R}^{m\times r}$ is an orthogonal matrix ($U^\top U = I_r$), $V \in \mathbb{R}^{n\times r}$ is an orthogonal matrix and $\Sigma \in \mathbb{R}^{r\times r}$ is a diagonal full-rank matrix. This formulation requires $r(m+n+1) \approx r(m+n-r)$ storage capacity and has the advantage of having two orthogonal matrices.

Each vector $\dot X$ belonging to the tangent space $T_X\mathcal{M}_r$ of $\mathcal{M}_r$ at X has a unique representation $(\dot U, \dot\Sigma, \dot V)$ such that (Vandereycken, 2013)

$$\dot X = U\dot\Sigma V^\top + \dot U V^\top + U\dot V^\top, \tag{5}$$

$$U^\top \dot{U} = 0 \text{ and } V^\top \dot{V} = 0.$$

This formulation also requires $r(m+n+1) \approx r(m+n-r)$ storage capacity.

Given a vector Z in the ambient space $\mathbb{R}^{m\times n}$, its projection on the tangent space $T_X\mathcal{M}_r$ can be computed (Vandereycken, 2013) and is given by $P_{T_X\mathcal{M}_r}(Z)$, defined as

$$P_{T_X\mathcal{M}_r} : \mathbb{R}^{m\times n} \to T_X\mathcal{M}_r : Z \to P_U Z P_V + P_U^\perp Z P_V + P_U Z P_V^\perp,$$

with $P_U = UU^\top$ and $P_U^\perp = I - UU^\top$, P_V and $P_V^\perp$ being defined in the same way.

Using the tangent space representation, a basic identification yields the following representation for the (orthogonal) projection of an ambient vector Z onto $T_X\mathcal{M}_r$:

$$\dot{\Sigma} = U^\top Z V \qquad \dot{U} = (I - UU^\top) Z V \qquad \dot{V} = (I - VV^\top) Z^\top U. \tag{6}$$

The algorithm we will describe requires to be able to move along directions on the manifold. It is now well established (e.g. in Absil *et al.* 2008) that this can be cheaply achieved using a *retraction* instead of the expensive exponential map for instance, while keeping all convergence guarantees. We decided to use the projective retraction (Absil & Oseledets, 2014): given a vector $\dot{X} \in T_X\mathcal{M}_r$, it finds $Y \in \mathcal{M}_r$ such that

$$Y = R_X(\dot{X}) = \operatorname{argmin}_{Y\in\mathcal{M}_r} \|X + \dot{X} - Y\|_F.$$

The solution of this minimization problem is known to be the rank-r SVD of $X + \dot{X}$ (Eckart–Young theorem). Assuming X is given as (4) and $\dot{X}$ as (5), it is possible to compute it efficiently (Vandereycken, 2013; Absil & Oseledets, 2014).

Because $\mathcal{M}_r$ is embedded in $\mathbb{R}^{m\times n}$, a suitable vector transport (i.e., a mapping from the tangent space at some point to the tangent space at another point) is simply the projection of the ambient version of the original vector in the tangent space at the new point (Vandereycken, 2013).

3 Riemannian Optimization and Smoothing Techniques

As explained earlier, the problem we aim to solve is the following

$$\min_{X\in\mathcal{M}_r} \|\mathcal{P}_\Omega(X - M)\|_{\ell_1} + \lambda\|\mathcal{P}_{\bar{\Omega}}(X)\|_{\ell_2}^2, \tag{7}$$

with the following interpretation: we ought to find a low-rank matrix X that fits the data M in the ℓ_1 sense on the mask Ω. On the remaining entries in $\bar{\Omega}$, we have a small confidence λ (typically between 0 and 10^{-5}-10^{-3}, even though this is application-dependent) that the value should be zero, hence we minimize the ℓ_2 error between X and 0 on $\bar{\Omega}$ [2]. This is motivated by previous studies of (Boumal & Absil, 2015) were regularization was especially useful to deal with real datasets. Note that the reason for

[2] Note that this assumes that the entries in the matrix have a mean equal to zero.

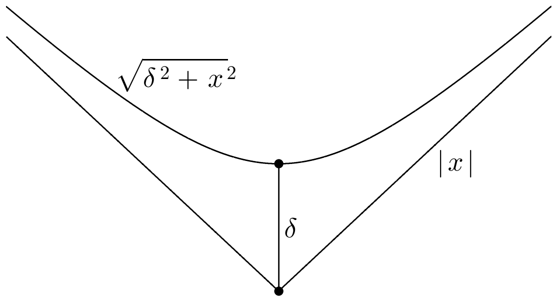


Fig. 2: Illustration of both $|x|$ and $\sqrt{\delta^2 + x^2}$.

the use of the ℓ_2 norm in the regularization term is twofold: first, the ℓ_2 norm will allow significant simplifications to be detailed in the forthcoming sections. Secondly, we can observe outliers on Ω, so the ℓ_1 norm makes sense there; but we obviously cannot observe outliers on $\bar{\Omega}$, so it does not seem necessary to use an ℓ_1 norm there, and an ℓ_2 norm should be more suitable.

The obvious main drawback of using (7) is that it is non differentiable. To remedy this problem, we decided to use smoothing techniques in order to make the objective differentiable. The idea is that, for a small $\delta > 0$, the following function

$$\sum_{(i,j)\in\Omega} \sqrt{\delta^2 + (X_{ij} - M_{ij})^2}$$

is a smooth approximation of

$$\|\mathcal{P}_\Omega(X - M)\|_{\ell_1},$$

as depicted on figure 2. The idea is thus to solve the following optimization problem

$$\min_{X\in\mathcal{M}_r} \sum_{(i,j)\in\Omega} \sqrt{\delta^2 + (X_{ij} - M_{ij})^2} + \lambda \sum_{(i,j)\in\bar{\Omega}} X_{ij}^2 \tag{8}$$

for decreasing values of δ.

To solve a problem in the form

$$\min_{x\in\mathcal{M}} f(x)$$

where $\mathcal{M}$ is a smooth Riemannian manifold and where f is smooth, there exist now many different techniques such as Riemannian conjugate gradient or trust-region algorithms (Absil *et al.*, 2008). We have opted for the conjugate gradient approach, as it appears to be more precise and efficient when δ becomes small.

3.1 The Objective Function and its Gradient

Using a first-order algorithm like conjugate gradient requires the computation of the cost function and the (Riemannian) gradient.

Taking a look at (8), one may think that just evaluating the cost would require $\mathcal{O}(mn)$ operations, since we need to evaluate X over *both* Ω and $\bar{\Omega}$. Actually, as pointed out in (Boumal & Absil, 2011) this is not the case, since

$$\|\mathcal{P}_{\bar{\Omega}}(X)\|_{\ell_2}^2 = \|X\|_{\ell_2}^2 - \|\mathcal{P}_{\Omega}(X)\|_{\ell_2}^2,$$

and, because we store X using the factorization $X = U\Sigma V^\top$, we can compute $\|X\|_{\ell_2}^2$ easily thanks to the invariance of the Frobenius norm to orthogonal changes of basis :

$$\|X\|_{\ell_2}^2 = \|\Sigma\|_{\ell_2}^2,$$

which requires $\mathcal{O}(r)$ operations. We can then rewrite the cost function of (8) as

$$f_\delta(X) = \sum_{(i,j)\in\Omega} \left(\sqrt{\delta^2 + (X_{ij} - M_{ij})^2} - \lambda X_{ij}^2 \right) + \lambda \|X\|_{\ell_2}^2. \tag{9}$$

Hence, computing the cost function requires $\mathcal{O}(|\Omega| + r)$ operations, after having evaluated the product $U\Sigma V^\top$ on the mask Ω.

We can clearly see here the advantage of having added an ℓ_2 regularization term: the fact that $\|X\|_{\ell_2}^2 = \|\Sigma\|_{\ell_2}^2$ is crucial to avoid an $\mathcal{O}(mn)$ complexity in the computation of the objective function.

To compute the gradient, because $\mathcal{M}_r$ is embedded in $\mathbb{R}^{m\times n}$, we first need to compute the Euclidian gradient of f at X and then project it onto $T_X\mathcal{M}_r$. The Euclidian gradient is

$$\nabla f_\delta(X) = S + 2\lambda X$$

where S is a *sparse matrix* defined as

$$S_{ij} = \begin{cases} \frac{X_{ij} - M_{ij}}{\sqrt{\delta^2 + (X_{ij} - M_{ij})^2}} - 2\lambda X_{ij} & \text{if } (i,j) \in \Omega, \\ 0 & \text{otherwise.} \end{cases}$$

The gradient is then the sum of a sparse component (S) and a low-rank one ($2\lambda X$). Then, projecting it onto $T_X M_r$ can be done efficiently thanks to equation (6) and to the factorization $X = U\Sigma V^\top$. Indeed, we have

$$\begin{aligned}
\dot{\Sigma} &= U^\top (S + 2\lambda X) V \\
&= U^\top S V + 2\lambda \Sigma, \\
\dot{U} &= (I - UU^\top)(S + 2\lambda X) V \\
&= SV + 2\lambda U\Sigma - UU^\top SV - 2\lambda U\Sigma \\
&= SV - UU^\top SV, \\
\dot{V} &= (I - VV^\top)(S + 2\lambda X)^\top U \\
&= S^\top U + 2\lambda V\Sigma - VV^\top S^\top U - 2\lambda V\Sigma \\
&= S^\top U - VV^\top S^\top U.
\end{aligned}$$

Given the fact that S is sparse, these three terms can be computed efficiently. Note that the addition of the regularization parameter is cheap, since it only requires to modify the S matrix, and to add a small $r \times r$ matrix to $\dot{\Sigma}$.

3.2 The Algorithm

The full algorithm is stated in algorithm 1; the name RMC stands for “Robust Matrix Completion”. Note that in the following, by outer and inner iteration we mean the δ and the conjugate-gradient (CG) loop, respectively.

We use a CG algorithm with a Hestenes-Stiefel modified rule (even though, after several experiments, we found that this choice does not really impact the algorithm) and an Armijo backtracking linesearch.

The starting point of the algorithm, $X^{(0)}$, can be chosen simply using the rank-r SVD of $\mathcal{P}_\Omega(M)$. Suitable values for $\delta^{(0)}$ (the initial value for the smoothing parameter δ) are application-dependent, but for data M with values around unity, we use $\delta^{(0)} = 1$. Note that this value can have a significant impact on the quality of the final solution. The smoothing parameter is then updated using a geometric rule $\delta^{(k+1)} = \theta \cdot \delta^{(k)}$. A quite “aggressive” value of $\theta = 0.05$ gives good results in our synthetic experiments. In real applications this parameter has to be tuned to find a suitable value. For all experiments, ϵ is set to 10^{-8} (but again, this is application-dependent). The stopping criterion of the conjugate-gradient algorithm is set to a maximum of 40 iterations or a gradient norm of 10^{-8}, whichever is reached first.

Algorithm 1 RMC

```
procedure RMC($X^{(0)}, \delta^{(0)}, \theta, \epsilon, \lambda$)
    $f^{(0)} \leftarrow \infty$
    $k \leftarrow 0$
    $\delta^{(1)} \leftarrow \delta^{(0)}$
    $e \leftarrow \infty$
    while $e \geq \epsilon$ do
        Solve
        $$X^{(k+1)} = \text{argmin}_{X \in \mathcal{M}_r} \sum_{(i,j)\in\Omega} \sqrt{(\delta^{(k+1)})^2 + (X_{ij} - M_{ij})^2} + \lambda \|\mathcal{P}_{\bar{\Omega}}(X)\|_{\ell_2}^2 \qquad (10)$$
        using Riemannian Conjugate Gradient algorithm and $X^{(k)}$ as a starting point.
        $f^{(k+1)} \leftarrow f_{\delta^{(k+1)}}(X^{(k+1)})$
        $e \leftarrow f^{(k)} - f^{(k+1)}$
        $k \leftarrow k + 1$
        $\delta^{(k+1)} \leftarrow \delta^{(k)} \cdot \theta$
    end while
end procedure
```

4 Convergence Analysis

This section provides a basic convergence analysis of the RMC algorithm. Its goal is mostly to give sense to the stopping criterion of the outer loop, i.e., that the algorithm will terminate at some point (assuming exact arithmetic at least).

Theorem 1 (Strict decrease)**.** *If the sequence of iterates* $\{X^{(0)}, X^{(1)}, X^{(2)}, \dots\}$ *is produced by algorithm 1 with* $\theta < 1$*, defining* $f^{(k)} = f_{\delta^{(k)}}(X^{(k)})$*, we have*

$$f^{(0)} > f^{(1)} > \cdots > f^{(k)} > \cdots$$

Proof. At iteration k, the CG algorithm returns a feasible solution $X^{(k)} \in \mathcal{M}_r$, associated with $\delta^{(k)}$. It is easy to see that $X^{(k)}$ stays a feasible point for the next step (since it belongs to $\mathcal{M}_r$), and that

$$f_{\delta^{(k)}}(X^{(k)}) > f_{\delta^{(k+1)}}(X^{(k)}).$$

This follows from $\delta^{(k+1)} = \theta \cdot \delta^{(k)} < \delta^{(k)}$ and the expression (9) of $f_\delta(X)$. Then, because CG is a descent direction,

$$f_{\delta^{(k+1)}}(X^{(k)}) \geq f_{\delta^{(k+1)}}(X^{(k+1)}).$$

The claim follows from these two inequalities. □

Now, it is also easy to notice that, $\forall \delta \geq 0$ and $\forall X \in \mathcal{M}_r$,

$$f_\delta(X) \geq f(X).$$

Hence, defining

$$f^* = \inf_{X \in \mathcal{M}_r} f(X) \geq 0,$$

we observe that the sequence of iterates $\{f_k\}_{k=1}^{K}$ is monotonically decreasing and bounded below by f^*.

This conclusion gives sense to the stopping criterion of the algorithm saying that it stops after iteration k if the difference between $f^{(k)}$ and $f^{(k+1)}$ is below some threshold ϵ: the algorithm will terminate at some point. We emphasize the fact that this does not prove that the algorithm converges towards a global minimum.

5 Numerical Experiments

In this section, we first apply RMC on synthetic problems. We then perform experiments on the NETFLIX dataset to show how the algorithm behaves in this situation.

5.1 Synthetic Problems

On all experiments, synthetic data are created in the following way: we build $U \in \mathbb{R}^{m \times r}$ and $V \in \mathbb{R}^{r \times n}$ with i.i.d. Gaussian-entries such that their product $M = UV$ is filled with zero-mean and unit-variance non independent Gaussian entries. We then sample $k = fr(m + n - r)$ entries uniformly at random, where f is the oversampling factor.

In some cases, we will add some non-Gaussian noise on part the observed entries to create outliers. To do so, we add one realization of the following random variable

$$\mathcal{O} = \mathcal{S}_{\pm 1} \cdot \mathcal{N}(\mu, \sigma^2)$$

where $\mathcal{S}_{\pm 1}$ is a random variable with equal probability to be equal to $+1$ or -1, while $N(\mu, \sigma^2)$ is a Gaussian random variable of mean μ and variance σ^2. Unless stated otherwise, outliers are created uniformly at random with some probability, usually 5%.

Different factors affect the task of matrix completion. Obviously, the size of the problem matters, and we will try to tackle large enough problems to show that our algorithm scales well. The oversampling factor f should also be greater than 1, and in the following experiments it will be fixed to either 4 or 5.

Let us denote by M_0 the original low-rank matrix, without any outliers. We will monitor how the root mean square error (RMSE), defined as the error on *all* the entries between X and the *original* matrix M_0

$$RMSE(X, M_0) = \sqrt{\frac{\sum_{i=1,j=1}^{m,n} (X_{ij} - M_{0,ij})^2}{mn}}$$

decreases. Since we have access to the factorization of both X and M_0, this can be computed efficiently (Boumal & Absil, 2015). A decrease towards zero is what we expect from a robust matrix completion method, since our goal is to recover exactly (up to numerical errors) the original low-rank matrix, even in the presence of outliers.

We decided to compare RMC (algorithm 1) to both AOPMC (Yan *et al.*, 2013) and GRASTA (He *et al.*, 2011).

For RMC, the maximum number of CG iterations (the inner loop) is set to 40 with a gradient tolerance of 10^{-8}. We use $\delta_0 = 1$, as well as $\theta = 0.05$.

For AOPMC, we use the default settings with a maximum of 20 trust-region iterations at each outer iteration (i.e., when the mask Ω is fixed, with potentially some outliers removed) but a maximum of 20 iterations for the tCG algorithm (see (Boumal & Absil, 2011) for further information). Note that we use the code of the authors[3], but because we know the number of outliers, we decided to provide it to AOPMC. Otherwise, we would need to run the algorithm several times to guess the number of outliers. Also note that AOPMC automatically adjusts the number of iterations if it seems that the

[3] Available at `https://binary-matching-pursuit.googlecode.com/files/AOPMCv1.zip`.

convergence is too bad. We did not change this option, so this may explain why the algorithm sometimes does more than 20 iteration between each update of the mask Ω.

Regarding GRASTA, we also use the implementation provided by the authors[4], with the default settings, but we had some troubles to run the algorithm on large $50\,000 \times 50\,000$ problems, since it was not even able to perform two complete sweeps over the data in less than 10 minutes (while RMC terminates in about 5 minutes in this situation). It seems to be due to the fact that the algorithm operates one column at a time, and since each rank-1 update of the U matrix takes about 0.01-0.02 second (using a standard but efficient Matlab implementation), one loop over the 50 000 columns takes more than 10 minutes. Note that this can be easily understood, since the original goal of GRASTA is not to perform "batch" matrix completion, but rather *online* subspace tracking. For the 500×500 case, everything goes well and the algorithm converges in an amount of time comparable to RMC. The only modification made to the algorithm was to change the initial point that was set, like the other algorithms, to the left matrix of the rank-r SVD of the matrix $\mathcal{P}_\Omega(M)$ (instead of some complete random initialization). But it does not have a significant influence on the convergence of the algorithm.

On all figures, the large dots indicate a change in the outer-iteration: in RMC it indicates a decrease in δ, while it indicates an update of the mask Ω in AOPMC.

We point out that we tried, without success, to speed-up the RMC and AOPMC algorithms by terminating the inner loop sooner, when the decrease from iteration to iteration was small enough. It appears that solving the inner problems with a good enough quality (i.e., a small enough gradient norm) is crucial for the convergence of both algorithms towards the exact underlying low-rank matrix. For this reason, we left the gradient tolerance set to 10^{-8}.

Note that the experiments where run on quite large $50\,000 \times 50\,000$ matrices, in order to show that the algorithm presented here scales well on large matrices. Some experiments were also run on $500\,000 \times 500\,000$ matrices of rank 10: the behavior of RMC was very stable and the running time appeared to be linear in the size of the matrix, i.e., in $\mathcal{O}(m+n)$. They are omitted for simplicity. Results on smaller matrices are qualitatively the same, except of course for the time the algorithm takes to reach the optimal value.

All experiments where run on a 6-core Intel Xeon CPU E5-1650 v2 at 3.50GHz with 64 Go of RAN using Matlab R2014a on a 64-bit Linux machine. We used the Manopt toolbox (Boumal *et al.*, 2014) (version 1.0.7) to handle the optimization part of the RMC algorithm with the `fixedrankembeddedfactory` manifold factory and the default conjugate gradient method.

Perfect Low-Rank Matrix Completion As a sanity check, we test our algorithm on the very simple *perfect matrix completion* (matrix completion without any noise nor outliers) problem using a $50\,000 \times 50\,000$ matrix of rank 10. Results are depicted on

[4] Available at `https://sites.google.com/site/hejunzz/grasta`.

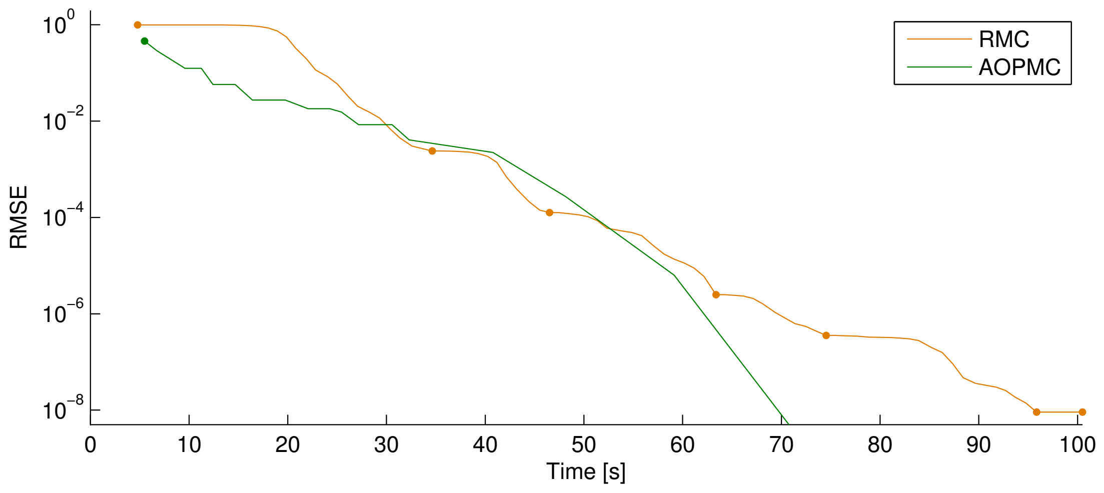


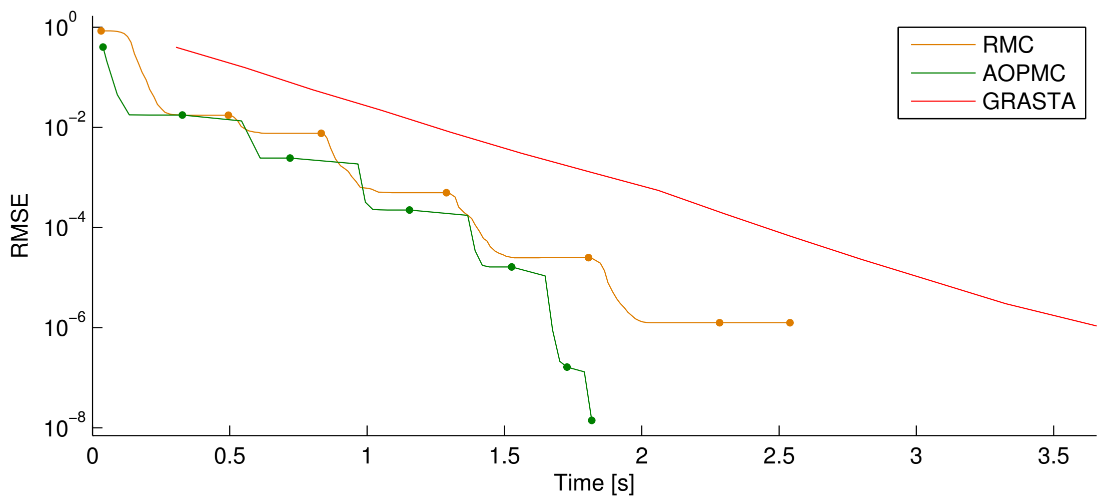
Fig. 3: Perfect low-rank matrix completion: low-rank matrix completion of a rank-10 $50\,000 \times 50\,000$ matrix observed with an oversampling of 4. The decrease in the objective function from iteration to iteration is clear. In this example, RMC struggles to significantly decrease the RMSE at the end since the function becomes less and less differentiable near the "kinks" of the absolute values, where the solution is located. AOPMC does not have this problem, and converges well towards 0.

figure 3. Both RMC and AOPMC successfully recover the original matrix: the RMSE is driven towards zero, as expected. GRASTA is too slow, and does not make one sweep over the data before 10 minutes.

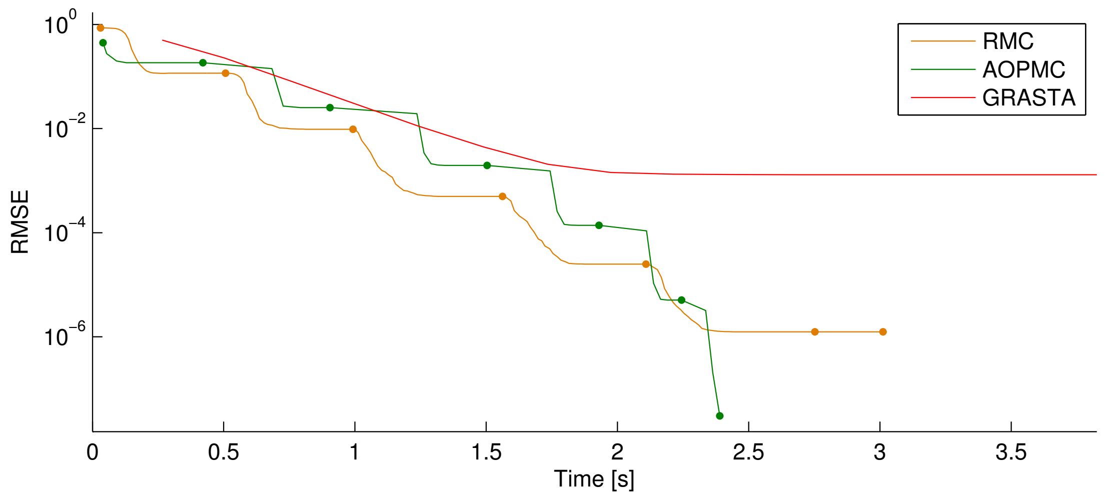
Low-Rank Matrix Completion with Outliers Given a 500×500 matrix for which we observe the entries uniformly at random with an oversampling of 4, we perturbed 5% of the observed entries by adding to them some non-Gaussian noise to create outliers. This problem would be cumbersome to solve with an ℓ_2 method because of the high weights the outliers would have in the objective function.

When running our algorithms, we obtain the results depicted on figure 4(a) for outliers created using $\mu = \sigma = 0.1$ and on figure 4(b) using $\mu = \sigma = 1$.

All the algorithms successfully solve the first problem, but we can see that GRASTA starts to have difficulties solving the second one. We suspect that this is due to the fact that GRASTA still uses an ℓ_2 norm in the algorithm (because of the augmented Lagrangian method), so the strength of the outliers still have some impact on the convergence. AOPMC seems to handle better the outliers, perhaps because the adaptive mask allows it to completely remove their impact from the objective function at the end.



(a) $\mu = \sigma = 0.1$



(b) $\mu = \sigma = 1$

Fig. 4: Low-rank matrix completion with outliers: robust low-rank matrix completion of rank-10 500×500 matrices observed with an oversampling of 4 and with 5% outliers in the observed entries.

We then run the same experiment on larger $50\,000 \times 50\,000$ matrices, with still 5% of outliers. Figure 5(a) and 5(b) illustrate the results of these experiments, with $\mu = \sigma = 1$ and $\mu = \sigma = 5$ respectively, using an oversampling of 5.

We can see that both AOPMC and RMC solve the first problem well. GRASTA, on the other hand, does not converge in a decent amount of time. We also observe that RMC stays very robust when the strength of the outliers increases, while AOPMC starts to have important difficulties in the second experiment. The robustness of RMC may be due to the asymptotic linear behavior of the cost function, even in the first iterations.

Note that this is a quite extreme experiment, in a sense that the outliers have a mean (absolute) amplitude of 5, while the entries have mean (absolute) value of 1. Yet, it demonstrates the robustness of RMC. Also note that the oversampling has a significant importance in this experiment, as an oversampling of 4 seems to make things harder for RMC: in this case, the RMSE stagnates around $10^{-4} - 10^{-5}$.

Noisy Low-Rank Matrix Completion with Outliers In this experiment, we try to tackle the important problem of matrix completion in the presence of *both* (dense) noise and (sparse) outliers. Outliers are defined as previously, while noise is the addition, at each observed entry, of a zero-mean Gaussian random variable with variance σ_N^2.

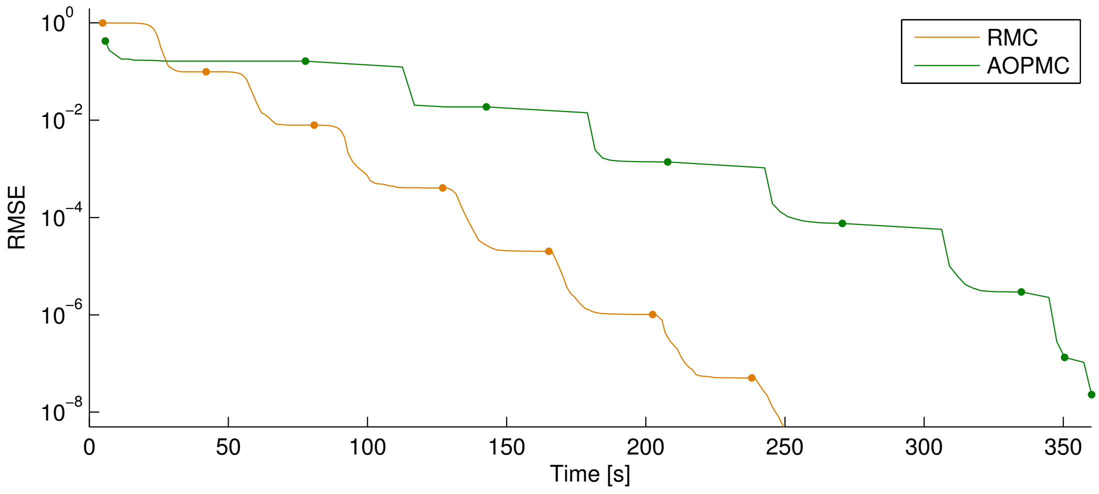
To have a point of comparison, we compare the results using RMC to the performances an oracle knowing the row and column space of M_0, and returning the best matrix using this information, would give. If the entries are perturbed by Gaussian noise (without outliers) with variance σ_N^2, the best RMSE is equal (in expectation) to (Candes & Plan, 2010)

$$\mathrm{RMSE}_{\mathrm{Oracle}} = \sigma_N \sqrt{\frac{2nr - r^2}{|\Omega|}}$$

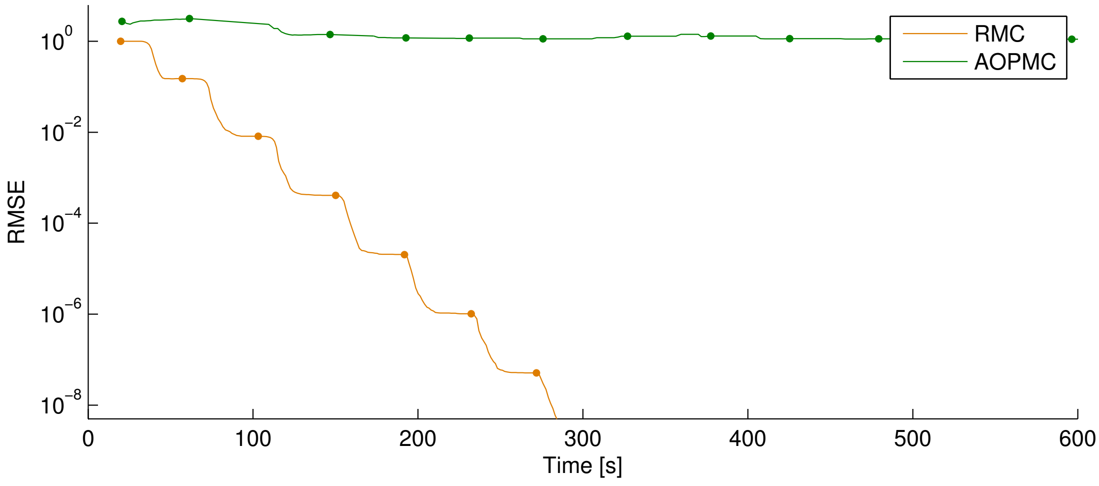
for the low-rank completion of an $n \times n$ matrix *with an ℓ_2 method.*

Figure 6 depicts the RMSE at termination of RMC with respect to the signal-to-noise (SNR) ratio. Results are the average of 3 successive experiments. Entries in the matrix M are such that the matrix has unit-variance Gaussian entries. We thus have $\mathrm{SNR} = \frac{1}{\sigma_N^2}$.

We clearly see that—even in the presence of 5% of outliers—the algorithm successfully recovers the original low-rank matrix with an error proportional to the noise level. As long are the noise level is not too high, we have performances very similar to those of the oracle bound. For a high level of noise ($\mathrm{SNR} < 1$), we see that the algorithm is slightly better than the oracle bound. This can be due to the ℓ_1 objective function which helps reduce the effect of the high variance. For a low level of noise, with an SNR greater than 10^{14}, the algorithm begins to have numerical difficulties to drive the RMSE towards zero because the objective function becomes less and less differentiable near the "kinks" of the absolute values. This is the same effect as in the previous experiments where the RMSE begins to stagnate around $10^{-8} - 10^{-6}$ due to the non-differentiability of the objective function near the solution, which happens to be exactly at the non-differentiable part of



(a) $\mu = \sigma = 1$



(b) $\mu = \sigma = 5$

Fig. 5: Low-rank matrix completion with outliers: robust low-rank matrix completion on $50\,000 \times 50\,000$ matrices of rank 10 with an oversampling of 5 and 5% outliers in the observed entries.

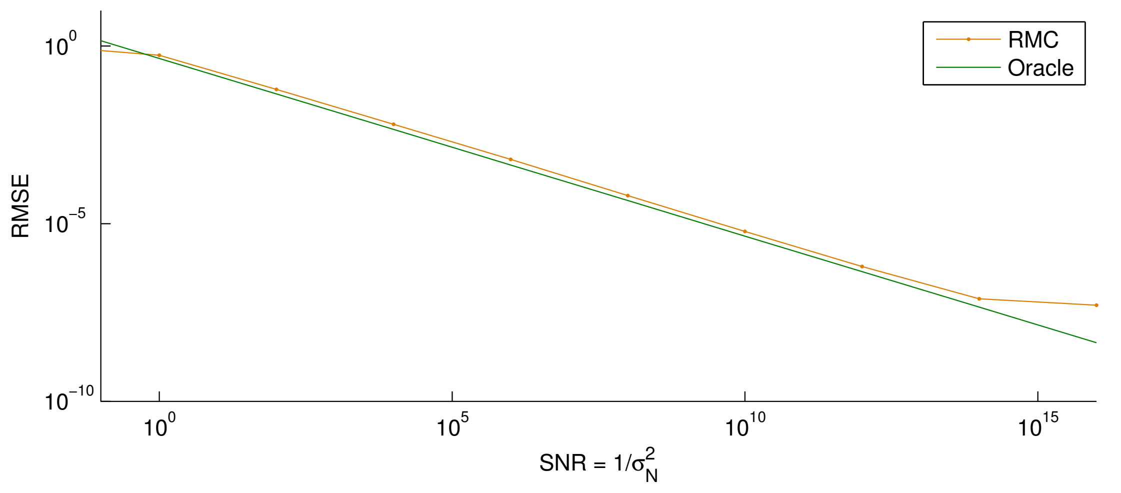


Fig. 6: Noisy low-rank matrix completion with outliers: Evolution of the RMSE with respect to the signal-to-noise ratio SNR $= \frac{1}{\sigma_N^2}$ on a $5\,000 \times 5\,000$ matrix of rank 10 with an oversampling of 5 and 5% outliers created using $\mu = 1$ and $\sigma = 1$. Noise is the addition of i.i.d. Gaussian variables $\mathcal{N}(0, \sigma_N^2)$.

the objective function. A moderate and high level of noise has the effect of "smoothing" the objective function since the solution starts to deviate from the kinks of the ℓ_1 norm.

Evolution with the Number of Outliers It is now interesting to ask the following: is all of this true for all percentages of outliers ? Does our algorithm recover the original matrix for all levels of outliers ? Of course this is not the case but figure 7 shows how the RMSE at termination evolves when either the percentage of outliers added or the outliers strength increases. This figure is obtained after the average (at each point on the plot) of 3 experiments, aiming at the completion of a $5\,000 \times 5\,000$ rank-10 matrix observed with an oversampling of 4 (note the triple-log scale).

Three things are worth noting on this figure. First, the abrupt change in the RMSE around 0.5% of outliers is purely numeric. The reason is that the slight increase of the percent of outliers allows the method to converge better: the algorithm succeeds in decreasing $\delta^{(k)}$ one more time (without the conjugate gradient stalling) and can then decrease the RMSE even more.

Secondly, we notice a strong increase in the RMSE around 6-8% of outliers. This is quite low but can be explained by the small oversampling ratio used: a low percentage of outliers can reduce the number of healthy entries below the minimum required number. Still, this is interesting, as it shows that as long as the number of outliers is small enough, the recovered matrix stays almost exactly the same as the original unperturbed matrix.

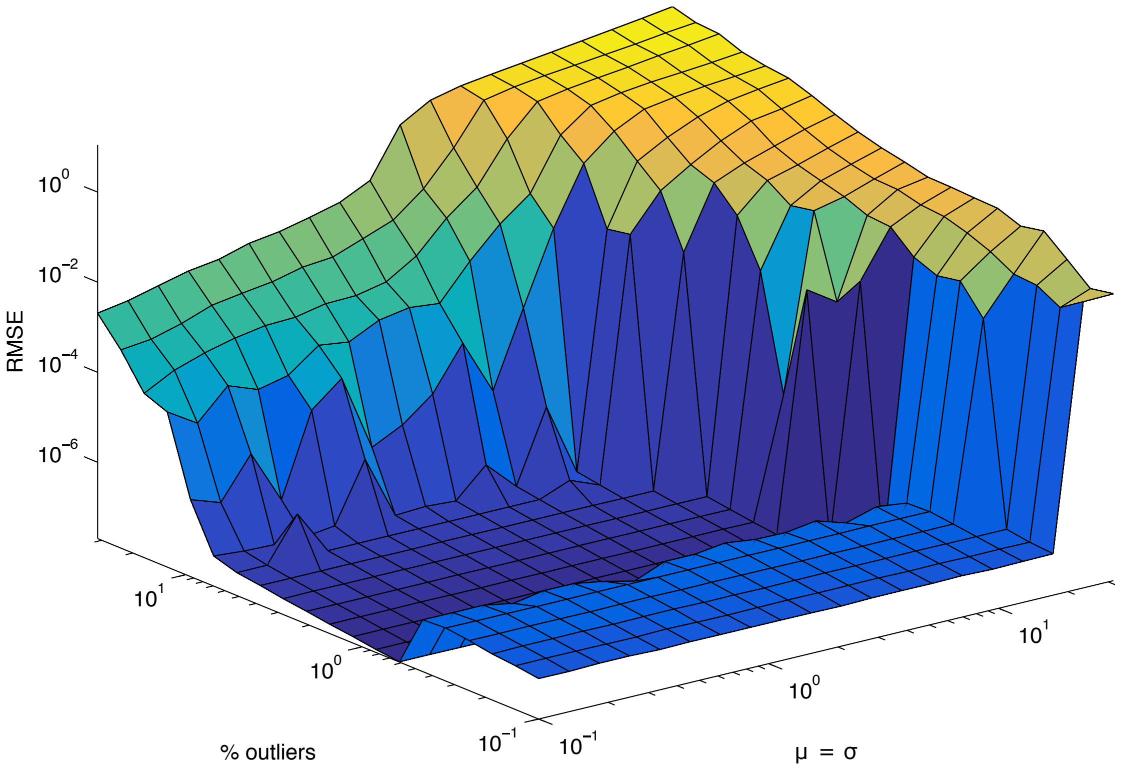


Fig. 7: Evolution with the Number of Outliers: evolution of the RMSE with respect to the percentage of outliers and their strength on a $5\,000 \times 5\,000$ matrix of rank 10, observed with an oversampling of 4 and using $\mu = \sigma$.

Thirdly, it is interesting to note that the strength of the outliers, from $\mu = \sigma = 0.1$ to $\mu = \sigma \approx 10$, has a negligible impact on the quality of the final solution (remember entries have values around unity) as long as it remains low enough. This is in complete opposition with the previous experiment on noisy matrix completion, where the RMSE is clearly proportional to the level of the dense Gaussian noise

5.2 Application: The Netflix Dataset

One of the most emblematic uses of Low-Rank Matrix Completion is in recommender systems and in particular in the NETFLIX Prize (Bennett & Lanning, 2007). In this section, we propose to test our algorithm RMC on this real-world dataset.

The problem is the following: NETFLIX, a movie rental company, wants to recommend movies to its users. To do so, they have a limited database of known ratings provided by some users themselves. In practice, this translates into the completion of a large $480\,189 \times 17\,770$ matrix, where columns correspond to movies, rows to user, and each entry is the rating of the movie represented by an integer from 1 to 5. To train the algorithm, we have 99 072 112 entries revealed; that is, approximately 1% of the entries are known. We then test our model on another set of 1 408 395 entries. Note that all values are shifted towards zero by subtracting the mean of the revealed entries (3.604), since our (regularized) model assumes a mean value of 0.

To assess the results, we use the root mean square criterion, i.e.,

$$\text{RMSE}_{\text{test}} = \sqrt{\frac{\sum_{(i,j)\in\text{TestSet}}(X_{ij} - M_{ij})^2}{|\text{TestSet}|}},$$

on the *test set* (i.e., the unrevealed entries). Returning the mean ratings as a prediction leads to a test RMSE of 1.13, while the winner of the NETFLIX prize, the BellKor's Pragmatic Chaos algorithm (Netflix, 2009), reached an RMSE of 0.8567, using a combination of many different techniques. Boumal & Absil (2015) performed extensive comparisons between different low-rank matrix completion algorithms (all minimizing the ℓ_2 norm on the training set). We can observe that the best result was obtained using LMaFit (Balzano *et al.*, 2010), reaching a test RMSE of 0.955.

Two parameters have a significant influence on the results. The rank "dictates" the complexity of the solution; the regularization parameter λ is used to limit overfitting, i.e., to avoid fitting too well the known entries while deviating too much from the mean on the unknown entries. To study the impact of both parameters, we picked a reference point with $r = 10$ and $\lambda = 8 \cdot 10^{-4}$ and we then changed the two parameters around these two values, one at the time. Note that the rank was chosen arbitrarily, while the value of λ is the one that gives the best results for a rank of 10. Also note that, after a few trials, we found that in this context, iteratively decreasing the δ parameter was not particularly useful. We then decided to fix it to 1. This value might seems high, but the smooth version is already a quite good approximation of the ℓ_1 norm. The number of

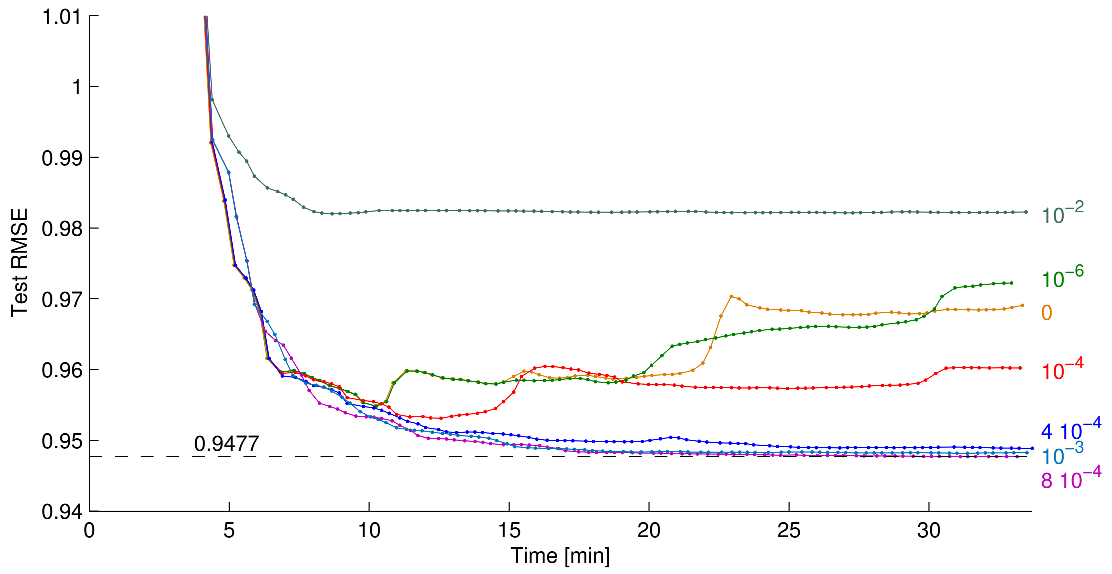


Fig. 8: Evolution of the test RMSE on the NETFLIX dataset for different values of λ (displayed on the right) using a rank $r = 10$. It is clear from this experiment that the regularization λ play an important role. By tuning it to the right value, we obtain a quite good test RMSE of 0.9477

iterations was limited to 100, and the gradient tolerance was set to 10^{-8}. In practice, most of the runs did not reach a gradient tolerance of 10^{-8} and were interrupted after 100 iterations.

Figure 8 depicts the evolution of the test RMSE for different values of λ. We can easily see that the non-regularized algorithm reaches a reasonable test RMSE but then tends to overfit. Increasing the regularization parameter λ around 10^{-3}-10^{-4} allows to find a good compromise between training error and overfitting. By increasing it even more, the test RMSE eventually stagnates, i.e., the regularization is so strong that it does not really fit anything except the mean value (see the $\lambda = 10^{-2}$ curve for instance).

Figure 9 illustrates how the rank plays a significant role in the solution quality. From this plot, it seems that the choice $r = 10$ was the right one, since other values for r give higher results. As underlined in (Boumal & Absil, 2015), choosing a high rank from the beginning does not seem to be the best choice and rank increasing methods may be worth investigating.

We can conclude from these experiments that our algorithm performs well on the NETFLIX dataset since it slightly outperforms the low-rank matrix completion algorithms that use the ℓ_2 norm. This may be due to the fact that our method is robust to outliers: this can help to reduce overfitting (even with no regularization), hence leading to a better overall model that better fits the test set.

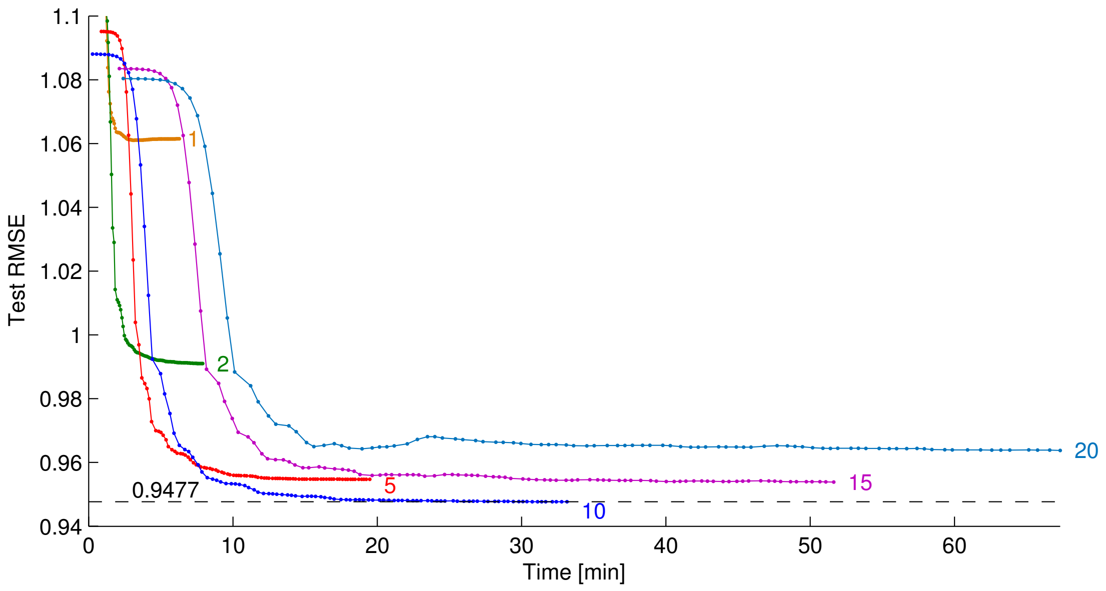


Fig. 9: Evolution of the test RMSE on the NETFLIX dataset for different values of the rank r using $\lambda = 8 \cdot 10^{-4}$. It seems that the choice $r = 10$ is the right one, and that increasing the rank leads to overfitting.

Still, it is known that to reach better test RMSE, it is useful to combine this idea of low-rank matrix completion with other techniques. For instance, temporal effects have a large impact, as explained in (Bennett & Lanning, 2007): movies become more or less popular over time, user tastes and ratings can change over time, and so on. Neighborhood models, where users and movies are aggregated into groups of similar profiles, also help in decreasing the test RMSE (Bennett & Lanning, 2007). Yet, our algorithm proved itself to be quite efficient and may certainly be used as a good building block for a more complex algorithm. It should also be possible to better fine tune each parameter of the algorithm to find an even better configuration.

6 Conclusion

In this paper, we have developed a *robust* low-rank matrix completion algorithm designed to solve the matrix-completion problem where a small part of the entries are outliers. Our method can also handle the case were the all the entries are corrupted by a small Gaussian noise.

We tackled this problem using Riemannian optimization as well as smoothing techniques to handle the non-smooth objective function. The resulting algorithm (Algorithm 1) clearly outperforms state-of-the-art methods when outliers have a large amplitude. On the Netflix data sets, it provides a 0.8% improvement on the test RMSE compared to the most accurate (according to Boumal & Absil (2015)) ℓ_2 low-rank matrix completion method, namely LMaFit.

The code of RMC can be downloaded from `www.baemerick.be/lcambier`.

Acknowledgments

The authors thank Ming Yan for providing the code of AOPMC as well as Nicolas Boumal and Yurii Nesterov for their insightful comments.

References

Absil, P-A, & Oseledets, Ivan V. 2014. Low-rank retractions: a survey and new results. *Computational Optimization and Applications*, 1–25.

Absil, P-A, Mahony, Robert, & Sepulchre, Rodolphe. 2008. *Optimization algorithms on matrix manifolds.* Princeton University Press.

Balzano, Laura, Nowak, Robert, & Recht, Benjamin. 2010. Online identification and tracking of subspaces from highly incomplete information. *Pages 704–711 of: Communication, Control, and Computing (Allerton), 2010 48th Annual Allerton Conference on.* IEEE.

Bennett, James, & Lanning, Stan. 2007. The netflix prize. *Page 35 of: Proceedings of KDD cup and workshop*, vol. 2007.

Boumal, Nicolas, & Absil, P.-A. 2015. Low-rank matrix completion via preconditioned optimization on the Grassmann manifold. *Linear Algebra and its Applications*, **475**(0), 200 – 239.

Boumal, Nicolas, & Absil, Pierre-antoine. 2011. RTRMC: A Riemannian trust-region method for low-rank matrix completion. *Pages 406–414 of: Advances in neural information processing systems.*

Boumal, Nicolas, Mishra, Bamdev, Absil, P.-A., & Sepulchre, Rodolphe. 2014. Manopt, a Matlab Toolbox for Optimization on Manifolds. *Journal of Machine Learning Research*, **15**, 1455–1459.

Candes, Emmanuel J, & Plan, Yaniv. 2010. Matrix completion with noise. *Proceedings of the IEEE*, **98**(6), 925–936.

Candès, Emmanuel J, & Recht, Benjamin. 2009. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, **9**(6), 717–772.

Candès, Emmanuel J, Li, Xiaodong, Ma, Yi, & Wright, John. 2011. Robust principal component analysis? *Journal of the ACM (JACM)*, **58**(3), 11.

Chen, Yudong, Xu, Huan, Caramanis, Constantine, & Sanghavi, Sujay. 2011. Robust matrix completion with corrupted columns. *arXiv preprint arXiv:1102.2254.*

Chen, Yudong, Jalali, Ali, Sanghavi, Sujay, & Caramanis, Constantine. 2013. Low-rank matrix recovery from errors and erasures. *Information Theory, IEEE Transactions on*, **59**(7), 4324–4337.

Chistov, Alexander L, & Grigor'ev, D Yu. 1984. Complexity of quantifier elimination in the theory of algebraically closed fields. *Pages 17–31 of: Mathematical Foundations of Computer Science 1984*. Springer.

Drineas, Petros, Javed, Asif, Magdon-Ismail, Malik, Pandurangan, Gopal, Virrankoski, Reino, & Savvides, Andreas. 2006. Distance matrix reconstruction from incomplete distance information for sensor network localization. *Pages 536–544 of: Sensor and Ad Hoc Communications and Networks, 2006. SECON'06. 2006 3rd Annual IEEE Communications Society on*, vol. 2. IEEE.

Hastie, Trevor. 2012. Matrix Completion and Large-scale SVD Computations.

He, Jun, Balzano, Laura, & Lui, John. 2011. Online robust subspace tracking from partial information. *arXiv preprint arXiv:1109.3827*.

Kennedy, Ryan, Balzano, Laura, Wright, Stephen J, & Taylor, Camillo J. 2014. Online algorithms for factorization-based structure from motion. *Pages 37–44 of: Applications of Computer Vision (WACV), 2014 IEEE Winter Conference on*. IEEE.

Keshavan, Raghunandan H, Montanari, Andrea, & Oh, Sewoong. 2009. Low-rank matrix completion with noisy observations: a quantitative comparison. *Pages 1216–1222 of: Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*. IEEE.

Klopp, Olga, Lounici, Karim, & Tsybakov, Alexandre B. 2014. Robust Matrix Completion. *arXiv preprint arXiv:1412.8132*.

Lee, John. 2003. *Introduction to smooth manifolds*. Vol. 218. Springer Gradate Texts in Mathematics.

Li, Xiaodong. 2013. Compressed sensing and matrix completion with constant proportion of corruptions. *Constructive Approximation*, **37**(1), 73–99.

Netflix. 2009. *Netflix Prize Leaderboard*. Available at http://www.netflixprize.com/leaderboard.

Nie, Feiping, Huang, Heng, & Ding, Chris. 2012a. Low-Rank Matrix Recovery via Efficient Schatten p-Norm Minimization. *Pages 655–661 of: Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*. AAAI.

Nie, Feiping, Wang, Hua, Cai, Xiao, Huang, Heng, & Ding, Chris. 2012b. Robust matrix completion via joint schatten p-norm and lp-norm minimization. *Pages 566–574 of: Data Mining (ICDM), 2012 IEEE 12th International Conference on*. IEEE.

Oh, Sewoong, Montanari, Andrea, & Karbasi, Amin. 2010. Sensor network localization from local connectivity: Performance analysis for the mds-map algorithm. *Pages 1–5 of: Information Theory Workshop (ITW), 2010 IEEE*. IEEE.

Peng, Yigang, Ganesh, Arvind, Wright, John, Xu, Wenli, & Ma, Yi. 2012. RASL: Robust alignment by sparse and low-rank decomposition for linearly correlated images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, **34**(11), 2233–2246.

So, Anthony Man-Cho, & Ye, Yinyu. 2007. Theory of semidefinite programming for sensor network localization. *Mathematical Programming*, **109**(2-3), 367–384.

Vandereycken, Bart. 2013. Low-rank matrix completion by Riemannian optimization. *SIAM Journal on Optimization*, **23**(2), 1214–1236.

Vandereycken, Bart, Absil, Pierre-Antoine, Vandewalle, Stefan, *et al.* . 2009. Embedded geometry of the set of symmetric positive semidefinite matrices of fixed rank. *Pages 389–392 of: Proceedings of the IEEE 15th Workshop on Statistical Signal Processing.*

Yan, Ming, Yang, Yi, & Osher, Stanley. 2013. Exact low-rank matrix completion from sparsely corrupted entries via adaptive outlier pursuit. *Journal of Scientific Computing*, **56**(3), 433–449.

Yang, Yuning, Fend, Yunlong, & Suykens, Johan A.K. 2014. *A Nonconvex Approach to Robust Matrix Completion.* Available at ftp://ftp.esat.kuleuven.be/stadius/yyang/yfs2014rmc.pdf.