Rank-Constrained Solutions to Linear Matrix Equations using PowerFactorization

Authors:

Justin P. Haldar and Diego Hernando
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign

This is an author preprint of:

J. P. Haldar, D. Hernando. "Rank-Constrained Solutions to Linear Matrix Equations using PowerFactorization." *IEEE Signal Processing Letters* 16:584-587, 2009.

The published electronic version can be found at http://dx.doi.org/10.1109/LSP.2009.2018223

Abstract: Algorithms to construct/recover low-rank matrices satisfying a set of linear equality constraints have important applications in many signal processing contexts. Recently, theoretical guarantees for minimum-rank matrix recovery have been proven for *nuclear norm minimization* (NNM), which can be solved using standard convex optimization approaches. While nuclear norm minimization is effective, it can be computationally demanding. In this work, we explore the use of the PowerFactorization (PF) algorithm as a tool for rank-constrained matrix recovery. Empirical results indicate that incremented-rank PF is significantly more successful than NNM at recovering low-rank matrices, in addition to being faster.

I. Introduction

There has been significant recent interest in solving the *affine rank minimization problem* [1]–[3], defined as

$$\begin{aligned} &\text{minimize} && \operatorname{rank}(\mathbf{X}) \\ &\text{subject to} && \mathcal{A}(\mathbf{X}) = \mathbf{b}, \end{aligned} \tag{1}$$

where $\mathbf{X} \in \mathbb{C}^{m \times n}$ is the unknown matrix to be recovered, and the linear operator $\mathcal{A} : \mathbb{C}^{m \times n} \to \mathbb{C}^p$ and vector $\mathbf{b} \in \mathbb{C}^p$ are given. While this kind of optimization problem is NP-hard in general [1], a number of heuristic algorithms have been proposed in the literature [4]–[6]. Of particular note is the *nuclear norm heuristic* [4], which replaces Eq. (1) with

$$\begin{aligned} &\text{minimize} && \|\mathbf{X}\|_* \\ &\text{subject to} && \mathcal{A}(\mathbf{X}) = \mathbf{b}, \end{aligned} \tag{2}$$

where $\|\mathbf{X}\|_*$ is the *nuclear norm* of $\mathbf{X}$, and is defined as the sum of its singular values. The nuclear norm is convex, and its use as a surrogate for matrix rank is analogous to the way that the ℓ_1 norm is used as a surrogate for vector sparsity in the emerging field of *compressed sensing* [7], [8]. Similar to results in compressed sensing, theoretical conditions for the equivalence of Eqs. (1) and (2) have been derived [1]–[3], and rely on $\mathcal{A}$ having special *restricted isometry* or *incoherence* properties.

Nuclear norm minimization (NNM) can be cast as a semidefinite programming problem (SDP), and can be solved using off-the-shelf interior point solvers like SDPT3 [9] or SeDuMi [10]. However, for large problem sizes, these methods are limited by large computation and storage requirements. As a result, various fast algorithms for NNM have appeared [1], [11], [12].

In this work, we propose to use the PowerFactorization (PF) algorithm [13] to solve a rank-constrained analogue of Eq. (1). PF seeks to find a matrix that can be factored as $\mathbf{X} = \mathbf{UV}$, with $\mathbf{U} \in \mathbb{C}^{m \times r}$ and $\mathbf{V} \in \mathbb{C}^{r \times n}$ so that $\operatorname{rank}(\mathbf{X}) \leq r$. This type of low-rank parameterization has been used in previous NNM

algorithms [1], [11] to improve computational efficiency at the expense of introducing nonconvexity. In contrast to NNM methods, PF optimizes $\mathbf{U}$ and $\mathbf{V}$ in alternation to find a local solution to

$$\text{minimize} \quad \|\mathcal{A}(\mathbf{UV}) - \mathbf{b}\|_2 . \tag{3}$$

We find that PF solutions, with gradually incremented r, are often superior compared to solutions of Eq. (2) at recovering low-rank matrices from a small number p of measurements. This suggests that in many cases, the rank constraint alone is sufficient for robust matrix recovery, and that use of NNM can be suboptimal. This performance improvement is not surprising, in the sense that the known theoretical conditions for a unique rank-constrained data consistent solution to Eq. (1) (see Thm. 3.2 of [1]) are significantly less stringent than the known conditions for NNM-based recovery without rank constraints (see Thm. 3.3 of [1]). What is surprising is that the non-convexity introduced by the low-rank parameterization frequently does not confound the simple PF procedure.

II. The PowerFactorization Algorithm

We begin the description of PF by introducing additional useful notation. In particular, we express the action of the linear operator $\mathcal{A}$ as

$$\begin{aligned} [\mathcal{A}(\mathbf{X})]_k &= \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ijk} [\mathbf{X}]_{(i,j)} \\ &= \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ijk} \sum_{\ell=1}^{r} [\mathbf{U}]_{(i,\ell)} [\mathbf{V}]_{(\ell,j)} \end{aligned} \tag{4}$$

for appropriate constants a_{ijk}, and for $k = 1, 2, \ldots, p$. As a consequence, we can write

$$\mathcal{A}(\mathbf{UV}) \equiv \mathbf{A}_{\mathbf{U}} \text{vec}(\mathbf{V}) \equiv \mathbf{A}_{\mathbf{V}} \text{vec}(\mathbf{U}) , \tag{5}$$

where $\text{vec}(\cdot)$ stacks the columns of its matrix argument into a single column vector. The matrices $\mathbf{A}_{\mathbf{U}} \in \mathbb{C}^{p \times rn}$ and $\mathbf{A}_{\mathbf{V}} \in \mathbb{C}^{p \times mr}$ are defined as

$$[\mathbf{A}_{\mathbf{U}}]_{(k,\ell+r(j-1))} = \sum_{i=1}^{m} a_{ijk} [\mathbf{U}]_{(i,\ell)} \tag{6}$$

and

$$[\mathbf{A}_{\mathbf{V}}]_{(k,i+m(\ell-1))} = \sum_{j=1}^{n} a_{ijk} [\mathbf{V}]_{(\ell,j)} , \tag{7}$$

respectively, for $k \in \{1, \ldots, p\}$, $\ell \in \{1, \ldots, r\}$, $j \in \{1, \ldots, n\}$, and $i \in \{1, \ldots, m\}$.

The PF algorithm iterates by alternatingly optimizing $\mathbf{U}$ and $\mathbf{V}$ using a linear-least squares (LLS) procedure. The algorithm runs as follows:

1) Initialize (arbitrarily) $(\mathbf{U} \in \mathbb{C}^{m \times r}, \mathbf{V} \in \mathbb{C}^{r \times n}) = \left(\mathbf{U}^{(0)}, \mathbf{V}^{(0)}\right)$. Set iteration number $q = 0$.
2) Holding $\mathbf{V}^{(q)}$ fixed, find $\mathbf{U}^{(q+1)}$ by solving

$$\mathbf{U}^{(q+1)} = \arg\min_{\mathbf{U}} \|\mathbf{A}_{\mathbf{V}^{(q)}} \mathrm{vec}\,(\mathbf{U}) - \mathbf{b}\|_2^2 . \tag{8}$$

3) Now fixing $\mathbf{U}^{(q+1)}$, find $\mathbf{V}^{(q+1)}$ by solving

$$\mathbf{V}^{(q+1)} = \arg\min_{\mathbf{V}} \|\mathbf{A}_{\mathbf{U}^{(q+1)}} \mathrm{vec}\,(\mathbf{V}) - \mathbf{b}\|_2^2 . \tag{9}$$

4) Increment q. If q exceeds a maximum number of iterations, if the iterations stagnate, or if the relative error $\left\|\mathcal{A}\left(\mathbf{U}^{(q)}\mathbf{V}^{(q)}\right) - \mathbf{b}\right\|_2 / \|\mathbf{b}\|_2$ is smaller than a desired threshold ε, then terminate the iterative procedure. Otherwise, repeat steps 2-4.

For matrix recovery, we have obtained best results by starting PF with $r = 1$, and gradually incrementing r until the desired rank constraint is achieved (or until the point where the relative error is smaller than ε in the case where the true rank is unknown). In this incremented-rank version of PF (IRPF), we initialize the new components of $\mathbf{U}$ and $\mathbf{V}$ using a rank-1 PF fit to the current residual.

The main computation in the PF procedure is solving the LLS problems in Eqs. (8) and (9). However, LLS problems are classical, and a number of efficient algorithms exist to compute solutions [14]. We do note that in some cases, the matrices $\mathbf{A}_{\mathbf{U}}$ and $\mathbf{A}_{\mathbf{V}}$ will not have full column rank, meaning that the LLS solution is non-unique; for example, if $\mathbf{V}$ is initialized to be identically zero, then $\mathbf{A}_{\mathbf{V}}$ is also identically zero. In these situations, it is beneficial to choose a LLS solution that is distinct from the minimum-norm least-squares solution; in our implementation, we randomly choose a vector from the linear variety of LLS solutions.

By construction, PF monotonically decreases the cost function in Eq. (3), and thus the value of the cost function is guaranteed to converge since it is bounded below by 0. In general, the iterates themselves are not guaranteed to converge, particularly in the case when there is sustained rank-deficiency in the LLS problems. Our empirical results show this is not generally an issue when the number of measurements p is large enough.

TABLE I
EXECUTION TIMES FOR IRPF AND NNM

Unknown $\mathbf{X}$			Time (s)	
size $(n \times n)$	$\frac{p}{(2n-r)r}$	p/n^2	IRPF	NNM
30×30	5	0.95	1.6	123.5
	4	0.76	1.4	141.1
	3	0.57	1.5	89.9
40×40	5	0.72	3.8	768.5
	4	0.58	3.5	555.1
	3	0.43	3.6	413.2
50×50	5	0.58	7.1	2331.1
	4	0.47	6.8	1616.7
	3	0.35	7.2	1223.6

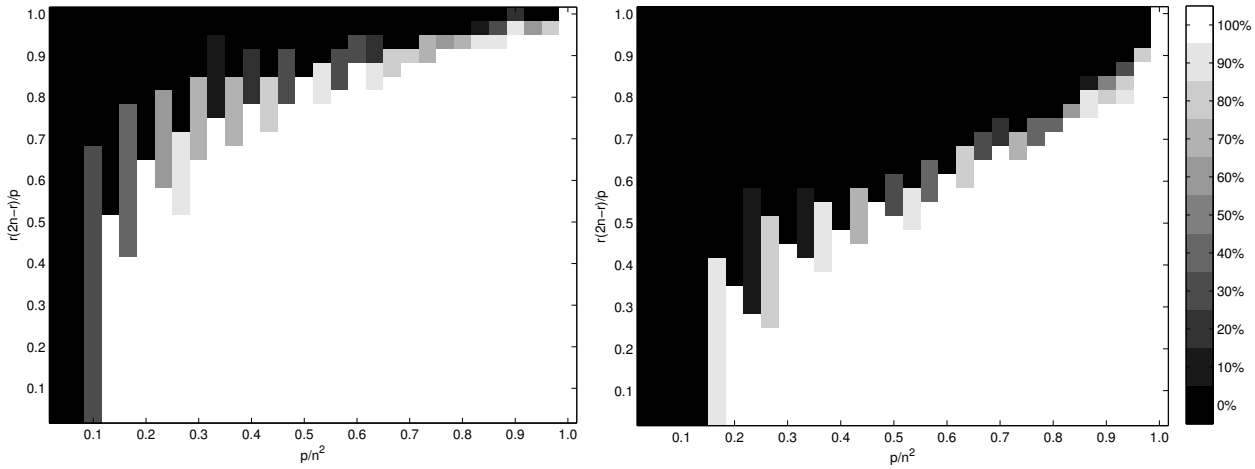
III. EMPIRICAL RESULTS

The IRPF algorithm was implemented in MATLAB and compared in several ways with an SDP implementation of NNM. All SDPs were solved using SDPT3 [9]. Our experiments are similar to those of [1]–[3], [12].

A. Speed Comparison

To test the speed of IRPF versus NNM, we generated a test set of random $\mathcal{A}$ operators and matrices $\mathbf{X}$ for a series of different problem sizes. The a_{ijk} defining the $\mathcal{A}$ operators were sampled from an i.i.d. Gaussian distribution. The $\mathbf{X}$ matrices were generated by sampling two matrices $\mathbf{M}_L \in \mathbf{C}^{m\times 3}$ and $\mathbf{M}_R \in \mathbf{C}^{3\times n}$, each containing i.i.d. Gaussian entries, and setting $\mathbf{X} = \mathbf{M}_L\mathbf{M}_R$. All experiments used $m = n$. The IRPF algorithm was stopped at $r = 3$ (i.e., the true rank), and was iterated until it had a relative error of $\varepsilon = 10^{-10}$. The matrices were successfully recovered in all cases for both algorithms, where recovery was declared if the estimated matrix $\hat{\mathbf{X}}$ satisfied $\|\hat{\mathbf{X}} - \mathbf{X}\|_F / \|\mathbf{X}\|_F < 10^{-3}$, where $\|\cdot\|_F$ is the Frobenius norm.

Table I shows the result of this speed comparison, and reports the execution time in seconds (on a 3.16 GHz CPU) for each of the two algorithms, averaged over 5 realizations. There is a clear order-of-magnitude difference in the speed of IRPF versus NNM with SDP.



(a) Incremented-Rank PowerFactorization (b) Nuclear norm minimization

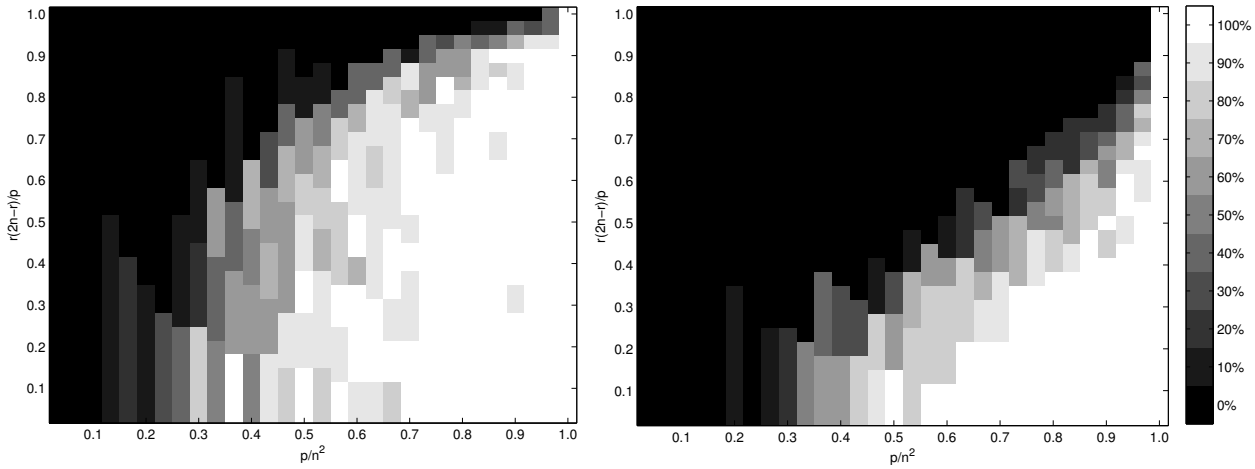
Fig. 1. Matrix recovery results for (a) IRPF and (b) NNM with Gaussian observations. The color of each cell corresponds to the empirical recovery rate, with white denoting perfect recovery and black denoting failure in all 10 experiments. The vertical axis is $r(2n-r)/p$, which is the ratio of the number of degrees of freedom for an $n \times n$ rank-r matrix to the number of measurements p.

B. Recovery Comparison

The matrix recovery capabilities of IRPF were compared to those of NNM using two sets of experiments. In the first set of recovery experiments, linear operators $\mathcal{A}$ and 30×30 matrices $\mathbf{X}$ were generated at random from the Gaussian distribution, as in the previous speed comparison experiment. Test cases were generated for many different combinations of the number of measurements p and $\operatorname{rank}(\mathbf{X})$ (assumed to be known), and 10 realizations were computed for each (p, r) pair. The second set of recovery experiments was identical to the first set, except that the linear operators $\mathcal{A}$ were chosen to directly observe p entries (selected uniformly at random) from $\mathbf{X}$. This corresponds to the so-called *matrix completion problem* [2], [11]–[13]. Theoretical properties of NNM for Gaussian observations and matrix completion are discussed in [1] and [2], respectively.

Figure 1 shows the results of the experiment with Gaussian observations. While NNM is able to successfully recover a large fraction of the low-rank matrices, IRPF is able to recover a significant additional fraction that NNM is unable to recover. As in NNM [1]–[3], there appears to be phase-transition behavior for IRPF with Gaussian measurements, though the boundary of this phase transition appears in a different location.

Figure 2 shows the results of the matrix completion experiment. Again, there is a significant fraction of matrices that is successfully recovered by IRPF, but that is not recovered by NNM. However, in



(a) Incremented-Rank PowerFactorization (b) Nuclear norm minimization

Fig. 2. Matrix completion results for (a) IRPF and (b) NNM. The color of each cell corresponds to the empirical recovery rate, as in Fig. 1.

this experiment, we observe a small number of cases when NNM succeeds while IRPF fails because it becomes trapped at a stationary point of the cost function. These few cases are easily identified without knowing the true $\mathbf{X}$, due to a large residual data error. For moderate-size problems, this can be efficiently overcome by performing IRPF several times with randomly selected initializations.

These results were obtained from relatively small matrices. Preliminary experiments indicate that an advantage of IRPF over NNM is maintained for larger matrices, although the asymptotic behavior is unknown.

C. Recovery using PF versus IRPF

While IRPF is more successful at matrix recovery and can converge faster than classical PF, PF alone can also perform surprisingly well given sufficient measurements and appropriately-chosen r. To illustrate this, we again generated Gaussian observation operators $\mathcal{A}$ for various p values, and random 40×40 matrices $\mathbf{X}$ of rank 8. We tested PF with this dataset, allowing r to range from 1 up to 20. The results of this experiment, averaged over 10 realizations, are shown in Fig. 3.

IV. Conclusion

This work investigated the IRPF algorithm as an alternative for NNM in the context of low-rank matrix recovery. IRPF is significantly faster than SDP-based NNM algorithms, and empirically has better

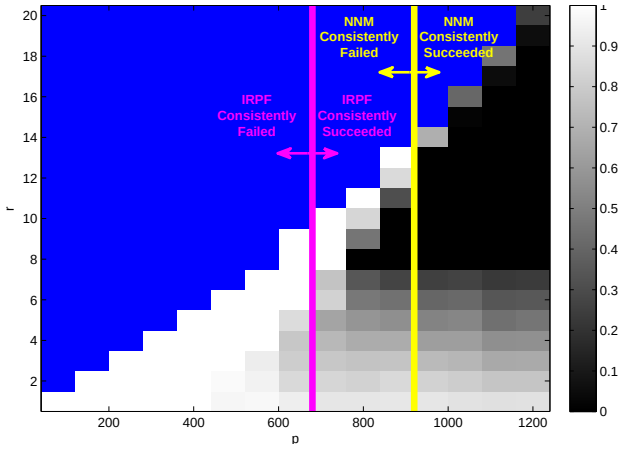


Fig. 3. Mean relative reconstruction error using PF for various r values. The true rank is 8. Blue indicates untested cases (the number of degrees-of-freedom exceeded p). The success/failure regimes for NNM and IRPF are indicated with yellow and pink lines, respectively.

recovery properties than NNM. While its theoretical properties are not yet fully established, IRPF has promising potential for practical matrix-recovery problems.

V. Acknowledgements

We would like to thank Yi Ma and John Wright for useful discussions.

References

[1] B. Recht, M. Fazel, and P. A. Parrilo, "Guaranteed minimum rank solutions to linear matrix equations via nuclear norm minimization," *Preprint*, 2008, http://arxiv.org/abs/0706.4138.

[2] E. Candes and B. Recht, "Exact matrix completion via convex optimization," *Preprint*, 2008, http://arxiv.org/abs/0805.4471.

[3] B. Recht, W. Xu, and B. Hassibi, "Necessary and sufficient conditions for success of the nuclear norm heuristic for rank minimization," *Preprint*, 2008, http://arxiv.org/abs/0809.1260.

[4] M. Fazel, H. Hindi, and S. Boyd, "A rank minimization heuristic with application to minimum order system approximation," in *Proc American Control Conf*, Arlington, 2001, pp. 4734–4739.

[5] ——, "Log-det heuristic for matrix rank minimization with applications to Hankel and Euclidean distance matrices," in *Proc American Control Conf*, Denver, 2003, pp. 2156–2162.

[6] K. Grigoriadis and E. Beran, "Alternating projection algorithms for linear matrix inequalities problems with rank constraints," in *Advances in Linear Matrix Inequality Methods in Control*, L. E. Ghaoui and S.-I. Niculescu, Eds. SIAM, 2000, pp. 251–267.

[7] E. Candes, J. Romberg, and T. Tao, "Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information," *IEEE Trans Info Theory*, vol. 52, pp. 489–509, 2006.

[8] D. Donoho, "Compressed sensing," *IEEE Trans Info Theory*, vol. 52, pp. 1289–1306, 2006.

[9] K. C. Toh, M. J. Todd, and R. H. Tütüncü, *SDPT3 - a MATLAB software package for semidefinite-quadratic-linear programming*, http://www.math.nus.edu.sg/~mattohkc/sdpt3.html.

[10] J. F. Sturm, "Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones," *Optimization Methods and Software*, vol. 11–12, pp. 625–653, 1999.

[11] J. D. M. Rennie and N. Srebro, "Fast maximum margin matrix factorization for collaborative prediction," in *Proc ICML*, 2005.

[12] J.-F. Cai, E. Candes, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *Preprint*, 2008, http://arxiv.org/abs/0810.3286.

[13] R. Vidal, R. Tron, and R. Hartley, "Multiframe motion segmentation with missing data using PowerFactorization and GPCA," *Int J Comput Vis*, vol. 79, pp. 85–105, 2008.

[14] G. Golub and C. van Loan, *Matrix Computations*, 3rd ed. London: The Johns Hopkins University Press, 1996.