FAST PRINCIPAL COMPONENT PURSUIT VIA ALTERNATING MINIMIZATION

Paul Rodríguez

Department of Electrical Engineering
Pontificia Universidad Católica del Perú
Lima, Peru

Brendt Wohlberg[*]

T-5 Applied Mathematics and Plasma Physics
Los Alamos National Laboratory
Los Alamos, NM 87545, USA

ABSTRACT

We propose a simple alternating minimization algorithm for solving a minor variation on the original Principal Component Pursuit (PCP) functional. In computational experiments in the video background modeling problem, the proposed algorithm is able to deliver a consistent sparse approximation even after the first outer loop, (taking approximately 12 seconds for a $640 \times 480 \times 400$ color test video) which is approximately an order of magnitude faster than Inexact ALM to construct a sparse component of the same quality.

Index Terms— Principal Component Pursuit, Video Background Modeling

1. INTRODUCTION

Principal Component pursuit (PCP), also known as Robust Principal Component Analysis (RPCA) has recently been proposed [1, 2] as a robust alternative to the well-known Principal Component Analysis (PCA). This method has found a variety of applications, including background modeling / foreground detection in video data [1, 2, 3, 4], text analysis [5], and image alignment [6].

The PCP problem was originally formulated [2] as

$$\arg\min_{L,S} \|L\|_* + \lambda\|S\|_1 \text{ s.t. } D = L + S \qquad (1)$$

where $D \in \Re^{m\times n}$ is the observed matrix, $\|L\|_*$ is the nuclear norm of matrix L (*i.e.* $\sum_k |\sigma_k(L)|$, the sum of the singular values of L) and $\|S\|_1$ is the ℓ^1 norm of matrix S.

Since PCP is computed via a computationally expensive optimization, and a real-time implementation capable of handling large data sets such as video is desirable, the development of fast algorithms for the PCP problem is an active area of research [7]. In this paper we propose a fast and computationally efficient algorithm to solve the PCP problem with a special focus on video background modeling / foreground detection, where each video frame is represented as a column of D.

[*]This research was supported by the NNSA's Laboratory Directed Research and Development Program.

2. NUMERICAL ALGORITHMS FOR PCP

Numerical algorithms for solving (1) are usually based on a splitting method [2, 5, 4, 3], such as the Augmented Lagrange Multiplier (ALM) method [8, 9] or its variants: (1) is solved via $\arg\min_{L,S,Y} \|L\|_* + \lambda \cdot \|S\|_1 + \langle Y, D - L - S\rangle + 0.5\mu \cdot \|D - L - S\|_F$, which includes a full or partial SVD (Singular Value Decomposition) depending on the ALM variant.

Originally in [2] the (exact) Augmented Lagrange Multiplier (ALM) method [8, 9] was used to solve (1); this approach is not computationally attractive since it performs a full SVD decomposition (follow by a thresholding on the resulting singular values) on the observed data, which has a negative impact on the computational performance, requiring, for example, about 43 minutes for a video sequence of 200 grayscale frames of size 144×176. In [4] the ALM algorithm was modified by partitioning the original problem into several sub-problems; while a full SVD decomposition is still required, the computational performance is reported to be about 6 times faster for the same video segmentation problem reported in [2]. In [3] it is proposed to use the power method to partially compute the SVD decomposition along with the ALM algorithm (this is somewhat similar to the inexact ALM where a partial SVD decomposition is computed via Lanczos); the computational performance is reported to be able to process a 640×480 color video at 12 frames per second in real-time (nevertheless, at the moment of writing this paper, the website mentioned in [3] does not have the code that implements this method). In [9] the inexact ALM method is proposed and thoroughly analyzed; furthermore, it is reported that this algorithm is consistently faster than previously reported state-of-the-art algorithms (which is verified in our computational simulations).

3. PROPOSED ALGORITHM

A number of variants of (1) can be constructed by changing constraints to penalties and vice-versa. We start with the functional $\arg\min_{L,S} \frac{1}{2}\|L + S - D\|_F + \lambda\|S\|_1$ s.t. $\|L\|_* \leq t$. Noting that we are usually interested in solutions for which the constraint $\|L\|_* \leq t$ is active, this becomes an equality constraint, and since in this case we can construct an algo-

rithm that constrains the rank itself rather than the nuclear norm relaxation, the functional becomes

$$\arg\min_{L,S} \frac{1}{2}\|L+S-D\|_F + \lambda\|S\|_1 \text{ s.t. } \text{rank}(L) = t \ . \quad (2)$$

While this modification does, unfortunately, discard some of the useful theoretical understanding of (1), including a principled choice of the parameter λ, in the video background modeling application component L typically has very low rank, and in practice we have not found it to be difficult to choose a suitable value of t.

A natural approach to solving problem (2) is the alternating minimization

$$L_{k+1} = \arg\min_{L} \quad \|L+S_k-D\|_F \text{ s.t. } \text{rank}(L) = t \quad (3)$$

$$S_{k+1} = \arg\min_{S} \quad \|L_{k+1}+S-D\|_F + \lambda\|S\|_1 \quad (4)$$

Sub-problem (3) can be solved by computing a partial (with t components) SVD of $D - S_k$. This is the only computationally demanding part of the algorithm, and can be efficiently computed via the Lanczos (or the power) method when t is small; we make use of the "lansvd" routine from optimized PROPACK library [10] (which is also employed in the publicly available implementation of the Inexact ALM algorithm [11]). The solution to (4) is simply element-wise shrinkage (soft thresholding) $\text{shrink}(D - L_{k+1}, \lambda)$, where

$$\text{shrink}(x,\epsilon) = \text{sign}(x)\max\{0, |x| - \epsilon\} \ . \quad (5)$$

While we do not present results here due to space constraints, we note that we have also considered a different form of (2)

$$\arg\min_{L,S} \frac{1}{2}\|L+S-D\|_F + \lambda\|S\|_1 \text{ s.t. } \|L\|_* \leq t \quad (6)$$

for which the L sub-problem corresponds to a nuclear norm regularization (NNR) problem [12], for which a computationally efficient algorithm has recently been proposed [13, 14]; while a potential application to PCP was suggested [14, Section 4.5.1], it was not explored further. We have implemented this algorithm, but found that the performance was very similar to that of an optimized iterative SVD algorithm in solving (3), and therefore concentrate on the simpler approach outlined in (2) for the remainder of this paper.

Since the video background modeling application component L typically has very low rank, we propose a simple procedure to estimate an upper bound for t in (3), described in Algorithm 1: since the singular values of L_{k+1} are a by-product of solving (3), if at each outer loop we increase t by one, then we can estimate the contribution of the new singular vector; if such contribution is small enough then we stop increasing the value of t. In our experimental simulations the value of t is increased up to 3 by the rule described in Algorithm 1 (this is consistent with the rank estimation performed by inexact ALM).

We provide computational evidence that our proposed algorithm monotonically converges to the desired solution;

Initialize

$S_1 = 0$, $D =$ input video, $rank = 1$ (initial rank)

for $k = 1, 2, ..,$ outerLoops

solve (3) with $t = rank$ (save singular values to v)

if $\frac{v_{rank}}{\sum_{k=1}^{rank} v_k} > \tau$ then $++rank$

solve (4)

Algorithm 1: Proposed alternating Algorithm for PCP (for the video background modeling application); we include a simple procedure to estimate an upperbound for t in (3).

moreover our computational results also show that our proposed algorithm has a low memory footprint so that it could be applied in real time video applications, and that it is approximately twice as fast as the state-of-the-art implementations such the leading Augmented Lagrange Multiplier algorithm (ALM) for PCP, while delivering results of comparable quality.

4. RESULTS

We compared our proposed algorithm with the inexact ALM [9] (code downloaded from [11]) labeled as "inexact ALM", and with the non-smooth augmented Lagrangian algorithm [4], labeled as "NSA" (code downloaded from [15]) for the video background removal problem; as a test video, we use a 640×480 pixel, 400-frame traffic (color) video sequence of 26.66 seconds at 15 fps, from the Lankershim Boulevard dataset [16, camera3]; each frame was scaled by $\frac{1}{255}$ (so the dataset is forced to have values between 0 and 1). For our experiments (code available from [17]) we have considered two cases: a reduced size version of 320×240 pixel per frame as well as the original size (640×480 pixel). We present two kind of results: (i) computational performance measured as the time to complete a given number of outer loops and (ii) reconstruction quality at each outer loop measured by $\frac{\|S_{GT}-S_k\|_1}{N}$ where S_{GT} is the "ground truth" sparse video approximation (we used the inexact ALM solution after 20 outer loops), S_k is the sparse video approximation of a given algorithm at the k^{th} outer loop, and N is the number of pixel per frame (use as a normalization factor).

All simulations have been carried out using Matlab code (which heavily makes use of the PROPACK [10] library, which is also the case for the inexact ALM and NSA algorithms) on a 2.8Ghz Intel Xeon quad-core X5560 CPU (L2: 8192K, RAM: 64G). The value of λ was empirically chosen to be 0.025, and although the optimization problem is not jointly convex, we found that our algorithm converged reliably to a good solution without the need for multiple runs with different initial solutions. In Fig. 1 (and Table 1) we show the computational performance comparison between our proposed algorithm and the inexact ALM and NSA algorithms for the video background removal problem using the

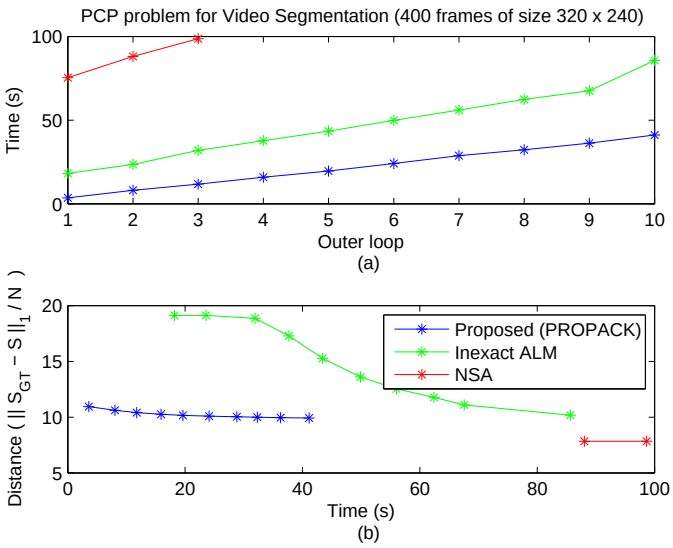

Fig. 1. Computational performance (a) and reconstruction quality (b) comparisons between our propose algorithm and the inexact ALM and NSA algorithms for a 400 frames video (each frame is 320×240 pixel). Note that in (b) we only show the NSA's reconstruction quality for the second and third outer loop since its first value is out of the chart.

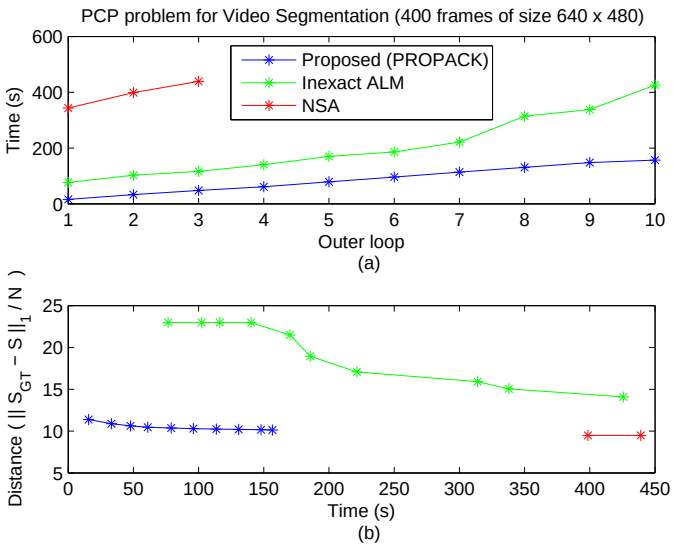

Fig. 2. Computational performance (a) and reconstruction quality (b) comparisons between our propose algorithm and the inexact ALM and NSA algorithms for a 400 frames video (each frame is 640×480 pixel). Note that in (b) we only show the NSA's reconstruction quality for the second and third outer loop since its first value is out of the chart.

previously described 400-frame (320×240 pixel each) traffic (color) video sequence. The observed data set requires 0.737 Gb of memory when represented using double precision values. In Fig. 1(a) we show the required time to complete a given number of outer loops (from one up to twenty, although in Table 1 we only report the performance for the first ten loops due to space constraints) for all the considered algorithms (the NSA algorithm was only run from one up to three outer loops because the improvement of the solution is only incremental after the third loop). Our proposed algorithm is faster than the inexact ALM (at least 1.8 times faster per outer loop) and significantly faster than the NSA algorithm (at least 8 times faster). More importantly, our proposed algorithm outperformed the inexact ALM and NSA on reducing the previously described reconstruction quality measure ($\frac{\|L_{GT}-L_k\|_1}{N}$), see Fig. 1(b) (as well as Table 2); this is basically due to the fact that the inexact ALM needs several outer loops to have a meaningful sparse approximation of the observed video, and that NSA needs a large amount of time to finished its second outer loop. Finally we mention that both our proposed algorithm and the inexact ALM can process the used color video sequence in real time.

In Fig. 2 (and Table 1) we show the computational performance of our algorithm on the same color video sequence but for frames of size 640×480. This data set requires 2.95 Gb of memory when represented using double precision values. We note that our proposed algorithm is also able to solve this problem on a more modest computer (3.4GHz Intel quadcore i7-2600 CPU, L2: 8192K, RAM: 16G), while the inexact ALM and NSA fail to do so. Regarding the results, in Fig. 2(a) we show the required time to complete a given number of outer loops (from one up to twenty) for all the considered algorithms (same restrictions apply to the NSA algorithm as for Fig.1(a)). As for the 320×240 case, our algorithm is consistently faster than the inexact ALM and the NSA algorithms; likewise, the reconstruction quality results (see Fig. 2(b) and Table 2) of our algorithm clearly outperforms those of the inexact ALM and the NSA algorithms.

We also notice that our algorithm could be use to solve the 640×480 color video background removal problem in real time since each additional outer loop needs about 16 seconds (average) and since even the results just after one outer loop of our algorithm has meaningful sparse approximation of the observed video (while the inexact ALM needs eleven outer loops to match the quality of our first approximation, see Fig. 2(b) and Table 2). Furthermore, in Fig. 3 we compared the sparse approximation (S component) for the initial and last frame of the 640×480 video sequence after one and twenty outer loops for our propose algorithm and after the twenty outer loops for the inexact ALM algorithm (since the first one is not meaningful), where it is clearly shown that our proposed algorithm has meaningful sparse approximation of the observed video even after one outer loop; the same results are observed for the 320×240 video sequence.

5. CONCLUSIONS

In the video background modeling application, our algorithm is approximately twice as fast as the leading Inexact ALM

(a) First frame of 640×480 pixel test video sequence

(b) Sparse approximation via our proposed algorithm after 1 outer loop (elapsed time: 14.26 sec).

(c) Sparse approximation via our proposed algorithm after 20 outer loops (elapsed time: 354.62 sec).

(d) Sparse approximation via inexact ALM after 20 outer loops (elapsed time: 1153.13 sec).

(e) Last frame of 640×480 pixel test video sequence

(f) Sparse approximation via our proposed algorithm after 1 outer loop (elapsed time: 14.26 sec).

(g) Sparse approximation via our proposed algorithm after 20 outer loops (elapsed time: 354.62 sec).

(h) Sparse approximation via inexact ALM after 20 outer loops (elapsed time: 1153.13 sec).

Fig. 3. Original initial (a) and final (d) frame of the 400 frames (640×480 pixel each) color video sequence and their respective sparse approximation after 1 outer loop ((b) and (f) respectively) and after 20 outer loops ((c), (d), (g) and (h) respectively) for our proposed Alternating Minimization algorithm and the inexact ALM algorithm. We notice that our proposed algorithm has a meaningful sparse approximation even after just one outer loop.

loop	320×240			640×480		
	Proposed	iALM	NSA	Proposed	iALM	NSA
1	3.26	16.54	75.44	14.26	70.42	338.55
2	7.48	22.51	89.46	31.35	106.65	400.80
3	11.44	29.10	96.10	49.46	122.88	429.58
4	15.55	34.09	–	72.33	149.49	–
5	21.12	39.94	–	90.73	173.05	–
6	28.07	46.18	–	110.53	193.85	–
7	31.79	52.14	–	126.38	230.40	–
8	35.92	57.29	–	138.62	307.17	–
9	40.99	62.13	–	155.14	325.92	–
10	43.21	78.73	–	174.11	432.06	–

Table 1. Computational performance (time in seconds needed by each algorithm to complete a given number of outer loops) comparison between our propose algorithm and the inexact ALM and NSA algorithms for a 400 frames video dataset.

loop	320×240			640×480		
	Proposed	iALM	NSA	Proposed	iALM	NSA
1	12.28	19.12	96.17	13.54	22.97	96.13
2	11.93	19.12	7.83	12.86	22.97	$9.47^{(*)}$
3	11.56	18.86	7.83	12.19	22.97	$9.47^{(*)}$
4	11.18	17.28	–	11.06	22.97	–
5	10.48	15.27	–	9.90	21.48	–
6	9.68	13.60	–	8.79	18.94	–
7	8.83	12.53	–	7.78	17.06	–
8	7.99	11.77	–	6.90	15.91	–
9	7.19	11.09	–	6.18	15.03	–
10	6.44	10.17	–	5.93	14.09	–

Table 2. Reconstruction quality (measured by $\frac{\|S_{GT} - S_k\|_1}{N}$ at the k^{th} outer loop, where where S_{GT} and S_k are the "ground truth" and current sparse video approximation respectively) comparison between our proposed algorithm and the inexact ALM and NSA algorithms for a 400 frames video dataset. $^{(*)}$ precision is insufficient to show numerical difference.

algorithm for computing an approximate solution to the PCP problem. Subjectively, this solution is of comparable quality to the solution of the original PCP problem, and if an accurate solution to this specific problem is required, our algorithm can be used to provide a good initial solution for use by Inexact ALM or other algorithms. Furthermore, the proposed algorithm delivers a useful estimate of the of the sparse component after a single outer loop, taking approximately 12 seconds for a $640 \times 480 \times 400$ test color video, which is approximately an order of magnitude faster than Inexact ALM to construct a sparse component of the same quality.

Although currently we do not have a theoretical proof for the convergence of our proposed Alternating Minimization algorithm, computational simulations support our claim that it converges to the desired solution. Future work will focus on providing a theoretical analysis of our proposed algorithm.

6. REFERENCES

[1] J. Wright, A. Ganesh, S. Rao, Y. Peng, and Y. Ma, "Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization," in *Adv. in Neural Inf. Proc. Sys. (NIPS) 22*, 2009, pp. 2080–2088.

[2] E. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?," *Journal of the ACM*, vol. 58, no. 3, May 2011.

[3] G. Pope, M. Baumann, C. Studer, and G. Durisi, "Real-time principal component pursuit," in *Conference Record of the Forty Fifth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, nov. 2011, pp. 1433 –1437.

[4] N. Aybat, G. Goldfarb, and G. Iyengar, "Fast first-order methods for stable principal component pursuit," arXiv:1105.2126, 2011.

[5] K. Min, Z. Zhang, J. Wright, and Y. Ma, "Decomposing background topics from keywords by principal component pursuit," in *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM)*, New York, NY, USA, 2010, pp. 269–278, ACM.

[6] Y. Peng, A. Ganesh, J. Wright, W. Xu, and Y. Ma, "RASL: Robust alignment by sparse and low-rank decomposition for linearly correlated images," in *Proc. IEEE Conf. Comp. Vis. Patt. Recog. (CVPR)*, June 2010, pp. 763–770.

[7] C. Guyon, T. Bouwmans, and E. Zahzah, "Robust principal component analysis for background subtraction: Systematic evaluation and comparative analysis," in *Principal Component Analysis*, P. Sanguansat, Ed., chapter 12. InTech, 2012.

[8] G. Liu, Z. Lin, and Y. Yu, "Robust subspace segmentation by low-rank representation.," in *Intl. Conf. Mach. Learn. (ICML)*, 2010, pp. 663–670.

[9] Z. Lin, M. Chen, and Y. Ma, "The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices," arXiv:1009.5055v2, 2011.

[10] R. M. Larsen, "PROPACK," Functions for computing the singular value decomposition of large and sparse or structured matrices, available from `http://sun.stanford.edu/~rmunk/PROPACK`.

[11] "Low-rank matrix recovery and completion via convex optimization (sample code)," `http://perception.csl.illinois.edu/matrix-rank/sample_code.html`.

[12] H. Zheng, "First-order methods for nuclear norm minimization and its applications," M.S. thesis, Simon Fraser University, 2011.

[13] M. Jaggi and M. Sulovský, "A simple algorithm for nuclear norm regularized problems," in *Proc. International Conference on Machine Learning (ICML)*, 2010, pp. 471–478.

[14] M. Jaggi, *Sparse Convex Optimization Methods for Machine Learning*, Ph.D. thesis, ETH Zurich, Oct. 2011.

[15] N. Aybat, "Non-Smooth Augmented (NSA) Lagrangian algorithm," `http://www2.ie.psu.edu/aybat/codes.html`.

[16] "Lankershim Boulevard dataset," Jan. 2007, U.S. Department of Transportation Publication FHWA-HRT-07-029, data available from `http://ngsim-community.org/`.

[17] B. Wohlberg P. Rodriguez, "Fast PCP simulation," `http://sites.google.com/a/istec.net/prodrig/Home/en/pubs`.