ISIT'2013 1569717095

Recursive Sparse Recovery in Large but Structured Noise - Part 2

Chenlu Qiu and Namrata Vaswani
ECE dept, Iowa State University, Ames IA, Email: {chenlu,namrata}@iastate.edu

***Abstract*—We study the problem of recursively recovering a time sequence of sparse vectors, S_t, from measurements $M_t := S_t + L_t$ that are corrupted by structured noise L_t which is dense and can have large magnitude. The structure that we require is that L_t should lie in a low dimensional subspace that is either fixed or changes "slowly enough"; and the eigenvalues of its covariance matrix are "clustered". We do not assume any model on the sequence of sparse vectors. Their support sets and their nonzero element values may be either independent or correlated over time (usually in many applications they are correlated). The only thing required is that there be *some* support change every so often. We introduce a novel solution approach called Recursive Projected Compressive Sensing with cluster-PCA (ReProCS-cPCA) that addresses some of the limitations of earlier work. Under mild assumptions, we show that, with high probability, ReProCS-cPCA can exactly recover the support set of S_t at all times; and the reconstruction errors of both S_t and L_t are upper bounded by a time-invariant and small value.**

I. Introduction

We study the problem of recovering a time sequence of sparse vectors, S_t, from measurements $M_t := S_t + L_t$ that are corrupted by (potentially) large magnitude but dense and structured noise, L_t. The structure that we require is that L_t should lie in a low dimensional subspace that is either fixed or changes "slowly enough" in the sense defined in Sec II-A. As a by-product, at certain times, we are also able to recover a basis matrix for the subspace in which L_t lies. Thus, at these times, we also solve the recursive robust principal components' analysis (PCA) problem. For recursive robust PCA, L_t is the signal of interest while S_t is the outlier (large but sparse noise). A key application where this problem occurs is in video analysis where the goal is to separate a slowly changing background from moving foreground objects [1].

Related Work. Most existing works on sparse recovery in large but structured noise study the case of sparse recovery from large but sparse noise (outliers), e.g., [2], [3], [4]. However, here we focus on the case of large but low-dimensional noise. On the other hand, most older works on robust PCA cannot recover the outlier / sparse vector when its nonzero entries have magnitude much smaller than that of the low dimensional part. However, in this work the main goal is to study sparse recovery and hence we do not discuss these older works here. Some recent works on robust PCA such as [5], [6] assume that an entire measurement vector M_t is either an an inlier (S_t is a zero vector) or an outlier (all entries of S_t can be nonzero), and a certain number of M_t's are inliers. These works cannot be used when all S_t's are nonzero but sparse.

The works of [1], [7] pose batch robust PCA as a problem of separating a low rank matrix, $\mathcal{L}_t := [L_1, \dots, L_t]$ and a sparse matrix, $\mathcal{S}_t := [S_1, \dots, S_t]$, from the measurement matrix, $\mathcal{M}_t := [M_1, \dots, M_t] = \mathcal{L}_t + \mathcal{S}_t$. Thus, these works can be interpreted as batch solutions to sparse recovery in large but low dimensional noise. It was shown in [1] that one can recover $\mathcal{L}_t$ and $\mathcal{S}_t$ exactly by solving $\min_{\mathcal{L},\mathcal{S}} \|\mathcal{L}\|_* + \lambda\|\mathcal{S}\|_1$ subject to $\mathcal{L}+\mathcal{S} = \mathcal{M}_t$, provided that (a) $\mathcal{L}_t$ is dense, (b) any element of the matrix $\mathcal{S}_t$ is nonzero w.p. ϱ, and zero w.p. $1-\varrho$, independent of all others (in particular, this means that the support sets of the different S_t's are independent over time); Here $\|B\|_*$ is the nuclear norm of B (sum of singular values of B) while $\|B\|_1$ is the ℓ_1 norm of B seen as a long vector. In most applications, it is fair to assume that the low-dimensional part, L_t (background in case of video analysis) is dense. However, the assumption that the support of the sparse part (foreground in case of video) is independent over time is often not valid. Foreground objects typically move in a correlated fashion, and may even not move for a few frames. This results in $\mathcal{S}_t$ being sparse and low rank.

The question then is, what can we do if $\mathcal{L}_t$ is low rank and dense, but $\mathcal{S}_t$ is both sparse and low rank? Clearly in this case, without any extra information, in general, it is not possible to separate $\mathcal{S}_t$ and $\mathcal{L}_t$. In [8], [9], [10], we introduced the Recursive Projected Compressive Sensing (ReProCS) algorithm that provided a solution by using the extra piece of information that an initial short sequence of L_t's is available (which can be used to get an accurate estimate of the subspace in which the initial L_t's lie) and assuming slow subspace change. The key idea of ReProCS is as follows. At time t, assume that a $n \times r$ matrix with orthonormal columns, $\hat{P}_{(t-1)}$, is available with $\text{span}(\hat{P}_{(t-1)}) \approx \text{span}(\mathcal{L}_{t-1})$ ($\text{span}(P)$ refers to the span of the columns of P). We project M_t perpendicular to $\text{span}(\hat{P}_{(t-1)})$. Because of slow subspace change, this cancels out most of the contribution of L_t. Recovering S_t from these $n-r$ dimensional projected measurements then becomes a classical sparse recovery / compressive sensing (CS) problem in small noise [11]. Under a denseness assumption on $\text{span}(\mathcal{L}_{t-1})$, one can show that S_t can be accurately recovered via ℓ_1 minimization. Thus, $L_t = M_t - S_t$ can also be recovered accurately. Every α time instants, we use the estimates of L_t in a projection-PCA algorithm to update $\hat{P}_{(t)}$.

ReProCS is designed under the assumption that the subspace in which the most recent several L_t's lie can only grow over time. It assumes a model in which at every subspace change time, t_j, some new directions get added to this subspace. After

every subspace change, it uses projection-PCA to estimate the newly added subspace. As a result the rank of $\hat{P}_{(t)}$ keeps increasing with every subspace change. Therefore, the number of effective measurements available for the CS step, $(n-\text{rank}(\hat{P}_{(t-1)}))$, keeps reducing. To keep this number large enough at all times, ReProCS needs to assume a bound on the total number of subspace changes, J.

Contributions. In practice, usually, the dimension of the subspace in which the most recent several L_t's lie typically remains roughly constant. A simple way to model this is to assume that at every change time, t_j, some new directions can get added and some existing directions can get deleted from this subspace and to assume an upper bound on the difference between the total number of added and deleted directions (the earlier model in [10] is a special case of this). ReProCS still applies for this more general model as discussed in the extensions section of [10]. However, because it never deletes directions, the rank of $\hat{P}_{(t)}$ still keeps increasing with every subspace change time and so it still requires a bound on J.

In this work, we address the above limitation by introducing a novel approach called *cluster-PCA* that re-estimates the current subspace after the newly added directions have been accurately estimated. This re-estimation step ensures that the deleted directions have been "removed" from the new $\hat{P}_{(t)}$. We refer to the resulting algorithm as ***ReProCS-cPCA***. The design and analysis of cluster-PCA and ReProCS-cPCA is the focus of the current paper. Under the clustering assumption and some other mild assumptions, we show that, w.h.p, at all times, ReProCS-cPCA can exactly recover the support of S_t, and the reconstruction errors of both S_t and L_t are upper bounded by a time invariant and small value. Moreover, we show that the subspace recovery error decays roughly exponentially with every projection-PCA step. The proof techniques developed in this work are very different from those used to obtain performance guarantees in recent batch robust PCA works such as [1], [7], [12], [5], [6], [13], [14], [15], [16], [17], [18], [19]. Our proof utilizes sparse recovery results [11]; matrix perturbation theory (sin θ theorem [20] and Weyl's theorem [21]) and the matrix Hoeffding inequality [22].

A key difference of our work compared with most existing work analyzing finite sample PCA, e.g. [23], and references therein, is that in these works, the noise/error in the observed data is independent of the true (noise-free) data. However, in our case, because of how $\hat{L}_t$ is computed, the error $e_t = L_t - \hat{L}_t$ is correlated with L_t. As a result the tools developed in these earlier works cannot be used for our problem. This is the main reason we need to develop and analyze projection-PCA based approaches for both subspace addition and deletion.

Notation. For scalars t_1, t_2, $[t_1, t_2] := \{t_1, t_1 + 1, \dots t_2\}$. The notation $[\hat{L}_t; [t_1, t_2]]$ denotes the matrix $[\hat{L}_{t_1}, \hat{L}_{t_1+1}, \cdots, \hat{L}_{t_2}]$.

For a vector v, v_T denotes a vector consisting of the entries of v indexed by T. For a matrix B, B' denotes its transpose, and $B^\dagger$ its Moore-Penroe pseudo-inverse. Also, $\|B\|_2 := \max_{x \neq 0} \|Bx\|_2 / \|x\|_2$. For a Hermitian matrix, B, $B \overset{EVD}{=} U \Lambda U'$ to denotes the eigenvalue decomposition of B. Here U is an orthonormal matrix and Λ is a diagonal matrix with entries arranged in non-increasing order. We use I to denote an identity matrix. For an index set T and a matrix B, B_T is the sub-matrix of B containing columns with indices in the set T. Notice that $B_T = BI_T$. Also, $B \setminus B_T$ denotes B_{T^c} (here $T^c := \{i \in [1, n] : i \notin T\}$). Given matrices B and B_2, $[B\ B_2]$ constructs a new matrix by concatenating matrices B and B_2 in a horizontal direction. [.] denotes an empty matrix.

We refer to a matrix P as a *"basis matrix"* if $P'P = I$.

The *s-restricted isometry constant (RIC)* [24], δ_s, for an $m \times n$ matrix Ψ is the smallest real number satisfying $(1 - \delta_s)\|x\|_2^2 \leq \|\Psi_T x\|_2^2 \leq (1+\delta_s)\|x\|_2^2$ for all sets T with $|T| \leq s$ and all vectors x of length $|T|$.

II. Problem Formulation

The measurement vector at time t, M_t, is an n dimensional real vector which can be decomposed as $M_t = L_t + S_t$. Here S_t is a sparse vector with support set size at most s and minimum magnitude of nonzero values at least $S_{\min}$. L_t is a dense but low dimensional vector, i.e. $L_t = P_{(t)} a_t$ where $P_{(t)}$ is an $n \times r_t$ "basis matrix" with $r_t \ll n$, that changes every so often. $P_{(t)}$ and a_t change according to the model given below. We are given an accurate estimate of the subspace in which the initial t_{train} L_t's lie, i.e. we are given a "basis matrix" $\hat{P}_0$ so that $\|(I - \hat{P}_0 \hat{P}_0')P_0\|_2$ is small. Here P_0 is a "basis matrix" for $\text{span}(\mathcal{L}_{t_{\text{train}}})$, i.e. $\text{span}(P_0) = \text{span}(\mathcal{L}_{t_{\text{train}}})$. The goal is (1) to estimate both S_t and L_t at each time $t > t_{\text{train}}$, and (2) to estimate $\text{span}(P_{(t)})$ every-so-often, i.e., update $\hat{P}_{(t)}$ so that $\|(I - \hat{P}_{(t)} \hat{P}_{(t)}')P_{(t)}\|_2$ is small.

Notation for S_t. Let $T_t := \{i : (S_t)_i \neq 0\}$ denote the support of S_t, $S_{\min} := \min_t \min_{i \in T_t} |(S_t)_i|$, and $s := \max_t |T_t|$.

Assumption 2.1 (Model on L_t): We assume that

1) $L_t = P_{(t)} a_t$ with $P_{(t)} = P_j$ for all $t_j \leq t < t_{j+1}$, $j = 0, 1, 2 \cdots J$, where P_j is an $n \times r_j$ "basis matrix" with $r_j \ll n$ and $r_j \ll (t_{j+1} - t_j)$. At the subspace change times, t_j, P_j changes as $P_j = [(P_{j-1} \setminus P_{j,\text{old}}),\ P_{j,\text{new}}]$. Here, $P_{j,\text{new}}$ is an $n \times c_{j,\text{new}}$ "basis matrix" with $P_{j,\text{new}}' P_{j-1} = 0$ and $P_{j,\text{old}}$ contains $c_{j,\text{old}}$ columns of P_{j-1}. Thus $r_j = r_{j-1} + c_{j,\text{new}} - c_{j,\text{old}}$. We let t_{J+1} equal the sequence length which can be infinite.
2) There exists a constant $c_{\max}$ such that $0 \leq c_{j,\text{new}} \leq c_{\max}$ and $\sum_{i=1}^{j} (c_{i,\text{new}} - c_{i,\text{old}}) \leq c_{\max}$ for all j. Thus, $r_j = r_0 + \sum_{i=1}^{j} (c_{i,\text{new}} - c_{i,\text{old}}) \leq r_0 + c_{\max}$, i.e., the rank of P_j is upper bounded. Let $r_{\max} := r_0 + c_{\max}$.
3) The coefficients' vector, $a_t = {P_{(t)}}' L_t$, is a random variable with the following properties. (a) a_t's are mutually independent over t. (b) It is a zero mean bounded r.v., i.e. $\mathbf{E}(a_t) = 0$ and there exists a constant γ_* s.t. $\|a_t\|_\infty \leq \gamma_*$ for all t. (c) Its covariance matrix $\Lambda_t := \text{Cov}(a_t) = \mathbf{E}(a_t a_t')$ is diagonal with $\lambda^- := \min_t \lambda_{\min}(\Lambda_t) > 0$ and $\lambda^+ := \max_t \lambda_{\max}(\Lambda_t) < \infty$. Thus the condition number of Λ_t is bounded by $f := \frac{\lambda^+}{\lambda^-}$.

A. Slow subspace change

Slow subspace change means the following. First, the delay between subspace change times, $t_{j+1} - t_j$, is large enough.

Second, the projection of L_t along the newly added directions, $a_{t,\text{new}}$, is initially small, i.e. $\max_{t_j \le t < t_j+\alpha} \|a_{t,\text{new}}\|_\infty \le \gamma_{\text{new}}$, with $\gamma_{\text{new}} \ll \gamma_*$ and $\gamma_{\text{new}} \ll S_{\min}$, but can increase gradually. We model this as follows. Split $[t_j, t_{j+1}-1]$ into α length periods. Then

$$\max_j \max_{t \in [t_j+(k-1)\alpha, t_j+k\alpha-1]} \|a_{t,\text{new}}\|_\infty \le \min(v^{k-1}\gamma_{\text{new}}, \gamma_*)$$

for a $v > 1$ but not too large. This assumption is verified for real video data in [10, Sec X-B].

Third, the number of newly added directions is small, i.e. $c_{j,\text{new}} \le c_{\max} \ll r_0$. This is also verified in [10, Sec X-B].

B. Measuring denseness of a matrix and its relation with RIC

For a tall $n \times r$ matrix, B, or for a $n \times 1$ vector, B, we define the the denseness coefficient as follows [10]:

$$\kappa_s(B) := \max_{|T| \le s} \frac{\|I_T{}' B\|_2}{\|B\|_2}.$$

where $\|.\|_2$ is the matrix or vector 2-norm respectively. κ_s measures the denseness (non-compressibility) of a vector B or of the columns of a matrix B. As explained in [10], [25], $\kappa_s(B)$ is related to the denseness assumptions required by PCP [1].

The lemma below relates κ_s of a "basis matrix" P to the RIC of $I - PP'$. The proof is in [10, Appendix].

Lemma 2.2: For a "basis matrix" P, $\delta_s(I-PP') = \kappa_s^2(P)$.

C. Clustering assumption

Let $\tilde{t}_j := t_j + K\alpha$. Consider the case where $\tilde{t}_j < t_{j+1}$. We assume that (1) by $t = \tilde{t}_j$, the variances along the newly added directions as well as those along the existing directions (i.e. all diagonal entries of Λ_t) have stabilized and do not change much, so that we can cluster them into a few groups that remain the same for all $t \in [\tilde{t}_j, t_{j+1}-1]$. Moreover, (2) the distance between consecutive clusters is large; (3) the distance between the smallest and largest element of each cluster is small and (4) the number of clusters is small. Mathematically,

Assumption 2.3: we assume the following.

1) The index set $\{1, 2, \dots r_j\}$ can be partitioned into ϑ_j groups $\mathcal{G}_{j,(1)}, \mathcal{G}_{j,(2)}, \cdots, \mathcal{G}_{j,(\vartheta_j)}$ such that $\min_{i \in \mathcal{G}_{j,(k)}} \min_{t \in [\tilde{t}_j, t_{j+1}-1]} \lambda_i(\Lambda_t) > \max_{i \in \mathcal{G}_{j,(k+1)}} \max_{t \in [\tilde{t}_j, t_{j+1}-1]} \lambda_i(\Lambda_t)$, i.e. each group is a cluster, the first group contains the largest eigenvalues, the second one the next smallest set and so on. Let
   a) $G_{j,k} := (P_j)_{\mathcal{G}_{j,(k)}}$,
   b) $\tilde{c}_{j,k} := |\mathcal{G}_{j,(k)}|$ be the number of elements in $\mathcal{G}_{j,(k)}$,
   c) ${\lambda_{j,k}}^- := \min_{i \in \mathcal{G}_{j,(k)}} \min_{t \in [\tilde{t}_j, t_{j+1}-1]} \lambda_i(\Lambda_t)$, ${\lambda_{j,k}}^+ := \max_{i \in \mathcal{G}_{j,(k)}} \max_{t \in [\tilde{t}_j, t_{j+1}-1]} \lambda_i(\Lambda_t)$ and ${\lambda_{j,\vartheta_j+1}}^+ := 0$,
   d) $\tilde{g}_{j,k} := {\lambda_{j,k}}^+ / {\lambda_{j,k}}^-$,
   e) $\tilde{h}_{j,k} := {\lambda_{j,k+1}}^+ / {\lambda_{j,k}}^-$,
   f) $\vartheta_{\max} := \max_j \vartheta_j$.
2) $\tilde{h}_{\max} := \max_j \max_{k=1,2,\cdots,\vartheta_j} \tilde{h}_{j,k}$ is small enough,
3) $\tilde{g}_{\max} := \max_j \max_{k=1,2,\cdots,\vartheta_j} \tilde{g}_{j,k}$ is small enough,
4) $\tilde{c}_{\min} := \min_j \min_{k=1,2,\cdots,\vartheta_j} \tilde{c}_{j,k}$ is large enough.

III. ReProCS with cluster-PCA (ReProCS-cPCA)

We first briefly explain the main idea of projection-PCA (proj-PCA) [10]. The ReProCS with cluster-PCA (ReProCS-cPCA) algorithm is then explained.

Definition 3.1: Let $\tilde{t}_j := t_j + K\alpha$. Define
1) $\mathcal{I}_{j,k} := [t_j + (k-1)\alpha, t_j + k\alpha - 1]$ for $k = 1, 2, \cdots, K$.
2) $\tilde{\mathcal{I}}_{j,k} := [\tilde{t}_j + (k-1)\tilde{\alpha}, \tilde{t}_j + k\tilde{\alpha} - 1]$ for $k = 1, 2, \cdots, \vartheta_j$.
3) $\tilde{\mathcal{I}}_{j,\vartheta_j+1} := [\tilde{t}_j + \vartheta_j\tilde{\alpha}, t_{j+1} - 1]$.

A. The Projection-PCA algorithm

Given a data matrix $\mathcal{D}$, a basis matrix P and an integer r, projection-PCA (proj-PCA) applies PCA on $\mathcal{D}_{\text{proj}} := (I - PP')\mathcal{D}$, i.e., it computes the top r eigenvectors (the eigenvectors with the largest r eigenvalues) of $\frac{1}{\alpha_\mathcal{D}} \mathcal{D}_{\text{proj}} {\mathcal{D}_{\text{proj}}}'$. Here $\alpha_\mathcal{D}$ is the number of column vectors in $\mathcal{D}$. This is summarized in Algorithm 1. If $P = [.]$, then projection-PCA reduces to standard PCA, i.e. it computes the top r eigenvectors of $\frac{1}{\alpha_\mathcal{D}} \mathcal{D}\mathcal{D}'$.

We should mention that the idea of projecting perpendicular to a partly estimated subspace has been used in different contexts in past work, e.g. see [5] and references therein.

Algorithm 1 projection-PCA: $Q \leftarrow$ proj-PCA$(\mathcal{D}, P, r)$

1) Projection: compute $\mathcal{D}_{\text{proj}} \leftarrow (I - PP')\mathcal{D}$
2) PCA: compute $\frac{1}{\alpha_\mathcal{D}} \mathcal{D}_{\text{proj}} {\mathcal{D}_{\text{proj}}}' \overset{EVD}{=} \begin{bmatrix} Q & Q_\perp \end{bmatrix} \begin{bmatrix} \Lambda & 0 \\ 0 & \Lambda_\perp \end{bmatrix} \begin{bmatrix} Q' \\ {Q_\perp}' \end{bmatrix}$ where Q is an $n \times r$ basis matrix and $\alpha_\mathcal{D}$ is the number of columns in $\mathcal{D}$.

B. The ReProCS-cPCA algorithm

ReProCS-cPCA is summarized in Algorithm 2. It proceeds as follows. Steps 1, 2, 3a and 3b are explained in detail in [10]. Step 1 projects M_t perpendicular to $\hat{P}_{(t-1)}$, solves the ℓ_1 minimization problem, followed by support recovery and finally computes a least squares (LS) estimate of S_t on its estimated support. This final estimate $\hat{S}_t$ is used to estimate L_t as $\hat{L}_t = M_t - \hat{S}_t$ in step 2. The sparse recovery error, $e_t := \hat{S}_t - S_t$. Since $\hat{L}_t = M_t - \hat{S}_t$, e_t also satisfies $e_t = L_t - \hat{L}_t$. Thus, a small e_t (accurate recovery of S_t) means that L_t is also recovered accurately. Step 3a is used at times when no subspace update is done. In step 3b, the estimated $\hat{L}_t$'s are used to obtain improved estimates of span$(P_{j,\text{new}})$ every α frames for a total of $K\alpha$ frames using the proj-PCA procedure given in Algorithm 1. As explained in [10], within K proj-PCA updates (K chosen as given in Theorem 4.1), it can be shown that both $\|e_t\|_2$ and the subspace error, $\text{SE}_{(t)} := \|(I - \hat{P}_{(t)}\hat{P}_{(t)}')P_{(t)}\|_2$, drop down to a constant times ζ. In particular, if at $t = t_j - 1$, $\text{SE}_{(t)} \le r\zeta$, then at $t = \tilde{t}_j := t_j + K\alpha$, we can show that $\text{SE}_{(t)} \le (r + c_{\max})\zeta$. Here $r := r_{\max} = r_0 + c_{\max}$.

To bring $\text{SE}_{(t)}$ down to $r\zeta$ before t_{j+1}, we need a step so that by $t = t_{j+1} - 1$ we have an estimate of only span(P_j), i.e. we have "deleted" span$(P_{j,\text{old}})$. One simple way to do this is by standard PCA: at $t = \tilde{t}_j + \tilde{\alpha} - 1$, compute $\hat{P}_j \leftarrow$ proj-PCA$([\hat{L}_t; \tilde{\mathcal{I}}_{j,1}], [.], r_j)$ and let $\hat{P}_{(t)} \leftarrow \hat{P}_j$. Using the $\sin\theta$

Algorithm 2 Recursive Projected CS with cluster-PCA

Parameters: algorithm parameters: ξ, ω, α, $\tilde{\alpha}$, K, model parameters: t_j, r_0, $c_{j,\text{new}}$, ϑ_j and $\tilde{c}_{j,i}$

Initialization: Let $\hat{P}_{(t_{\text{train}})} \leftarrow \hat{P}_0$. Let $j \leftarrow 1$, $k \leftarrow 1$. For $t > t_{\text{train}}$, do the following:

1) **Estimate T_t and S_t via Projected CS:**
   a) Nullify most of L_t: compute $\Phi_{(t)} \leftarrow I - \hat{P}_{(t-1)}\hat{P}'_{(t-1)}$, $y_t \leftarrow \Phi_{(t)} M_t$
   b) Sparse Recovery: compute $\hat{S}_{t,\text{cs}}$ as the solution of $\min_x \|x\|_1 \ s.t. \ \|y_t - \Phi_{(t)}x\|_2 \le \xi$
   c) Support Estimate: $\hat{T}_t = \{i : \ |(\hat{S}_{t,\text{cs}})_i| > \omega\}$
   d) LS Estimate: $(\hat{S}_t)_{\hat{T}_t} = ((\Phi_t)_{\hat{T}_t})^\dagger y_t, \ (\hat{S}_t)_{\hat{T}_t^c} = 0$
2) **Estimate L_t.** $\hat{L}_t = M_t - \hat{S}_t$.
3) **Update $\hat{P}_{(t)}$:**
   a) If $t \neq t_j + q\alpha - 1$ for any $q = 1, 2, \dots K$ and $t \neq t_j + K\alpha + \vartheta_j\tilde{\alpha} - 1$, set $\hat{P}_{(t)} \leftarrow \hat{P}_{(t-1)}$
   b) **Addition: Estimate span($P_{j,\text{new}}$) iteratively using proj-PCA:** If $t = t_j + k\alpha - 1$
      i) $\hat{P}_{j,\text{new},k} \leftarrow$ proj-PCA($[\hat{L}_t; \mathcal{I}_{j,k}], \ \hat{P}_{j-1}, c_{j,\text{new}}$)
      ii) set $\hat{P}_{(t)} \leftarrow [\hat{P}_{j-1} \ \hat{P}_{j,\text{new},k}]$.
      iii) If $k = K$, reset $k \leftarrow 1$; else increment $k \leftarrow k+1$.
   c) **Deletion: Estimate span(P_j) by cluster-PCA:** If $t = t_j + K\alpha + \vartheta_j\tilde{\alpha} - 1$,
      i) For $i = 1, 2, \cdots, \vartheta_j$,
      $\hat{G}_{j,i} \leftarrow$ proj-PCA($[\hat{L}_t; \tilde{\mathcal{I}}_{j,k}], [\hat{G}_{j,1}, \hat{G}_{j,2}, \dots \hat{G}_{j,i-1}], \tilde{c}_{j,i}$)
      End for
      ii) set $\hat{P}_j \leftarrow [\hat{G}_{j,1}, \cdots, \hat{G}_{j,\vartheta_j}]$ and set $\hat{P}_{(t)} \leftarrow \hat{P}_j$.
      iii) increment $j \leftarrow j+1$.

theorem and the Hoeffding corollaries, it can be shown that, as long as f is small enough, doing this is guaranteed to give an accurate estimate of span(P_j). However f being small is not compatible with the slow subspace change assumption. Notice from Sec II that $\lambda^- \le \gamma_{\text{new}}$ and $\mathbf{E}[||L_t||_2^2] \le r\lambda^+$. Slow subspace change implies that γ_{new} is small. Thus, λ^- is small. However, to allow L_t to have large magnitude, λ^+ needs to be large. Thus, $f = \lambda^+/\lambda^-$ cannot be small unless we require that L_t has small magnitude for all times t.

In step 3c, we introduce a generalization of the above strategy called cluster-PCA, that removes the bound on f, but instead only requires that the eigenvalues of Cov(L_t) be sufficiently clustered as explained in Sec II-C. The main idea is to recover one cluster of entries of P_j at a time. In the k^{th} iteration, we apply proj-PCA on $[\hat{L}_t; \tilde{I}_{j,k}]$ with $P \leftarrow [\hat{G}_{j,1}, \hat{G}_{j,2}, \dots \hat{G}_{j,k-1}])$ to estimate span($G_{j,k}$). The first iteration uses $P \leftarrow [.]$, i.e. it computes standard PCA to estimate span($G_{j,1}$). By modifying the idea of [10], we can show that since $\tilde{g}_{j,k}$ and $\tilde{h}_{j,k}$ are small enough (by Assumption 2.3), span($G_{j,k}$) will be accurately recovered, i.e. $\|(I - \sum_{i=1}^k \hat{G}_{j,i}\hat{G}'_{j,i})G_{j,k}\|_2 \le \tilde{c}_{j,k}\zeta$. We do this ϑ_j times and finally we set $\hat{P}_j \leftarrow [\hat{G}_{j,1}, \hat{G}_{j,2} \dots \hat{G}_{j,\vartheta_j}]$ and $\hat{P}_{(t)} \leftarrow \hat{P}_j$. All of this is done at $t = \tilde{t}_j + \vartheta_j\tilde{\alpha} - 1$. Thus, at this time, $\text{SE}_{(t)} = \|(I - \hat{P}_j\hat{P}'_j)P_j\|_2 \le \sum_{k=1}^{\vartheta_j} \|(I - \sum_{i=1}^k \hat{G}_{j,i}\hat{G}'_{j,i})G_{j,k}\|_2 \le \sum_{k=1}^{\vartheta_j} \tilde{c}_{j,k}\zeta = r_j\zeta \le r\zeta$. Under the assumption that $t_{j+1} - t_j \ge K\alpha + \vartheta_{\max}\tilde{\alpha}$, this means that before t_{j+1} $\text{SE}_{(t)} \le r\zeta$.

IV. Performance Guarantees

We state the main result here first and then discuss it in the next section. For the proof outline and proof, see [25].

Theorem 4.1: Consider Algorithm 2. Let $c := c_{\max}$ and $r := r_{\max} = r_0 + c$. Assume that L_t obeys Assumption 2.1. Also, assume that $\|(I - \hat{P}_0\hat{P}'_0)P_0\| \le r_0\zeta$, for a ζ that satisfies

$$\zeta \le \min\left(\frac{10^{-4}}{(r+c)^2}, \frac{1.5\times 10^{-4}}{(r+c)^2 f}, \frac{1}{(r+c)^3\gamma_*^2}\right) \text{ where } f := \frac{\lambda^+}{\lambda^-}$$

Let $\xi_0(\zeta), \rho, \alpha_{\text{add}}(\zeta), \alpha_{\text{del}}(\zeta)$, $g_{j,k}$ be as defined in Definition 5.2 of [25]. and let $K(\zeta) := \left\lceil \frac{\log(0.6c\zeta)}{\log 0.6} \right\rceil$. If

1) *(algorithm parameters)* $\xi = \xi_0(\zeta), \ 7\rho\xi \le \omega \le S_{\min} - 7\rho\xi, \ K = K(\zeta), \ \alpha \ge \alpha_{\text{add}}(\zeta), \ \tilde{\alpha} \ge \alpha_{\text{del}}(\zeta)$,
2) *(denseness)*

$$\max_j \kappa_{2s}(P_{j-1}) \le 0.3, \ \max_j \kappa_{2s}(P_{j,\text{new}}) \le 0.15,$$
$$\max_j \max_{0\le k\le K} \kappa_{2s}(D_{j,\text{new},k}) \le 0.15,$$
$$\max_j \max_{0\le k\le K} \kappa_{2s}((I - P_{j,\text{new}}P_{j,\text{new}}')\hat{P}_{j,\text{new},k}) \le 0.15,$$
$$\max_j \kappa_s(R_j) \le \kappa_{s,e}^+$$

where $R_j := (I - \hat{P}_{j-1}\hat{P}'_{j-1} - \hat{P}_{j,\text{new},K}\hat{P}'_{j,\text{new},K})P_j$ and $D_{j,\text{new},k} := (I - \hat{P}_{j-1}\hat{P}'_{j-1} - \hat{P}_{j,\text{new},k}\hat{P}'_{j,\text{new},k})P_{j,\text{new}}$,

3) *(slow subspace change)*

$$\min_j (t_{j+1} - t_j) > K\alpha + \vartheta_{\max}\tilde{\alpha},$$
$$\max_j \max_{t\in\mathcal{I}_{j,k}} \|a_{t,\text{new}}\|_\infty \le \min(1.2^{k-1}\gamma_{\text{new}}, \gamma_*),$$
$$14\rho\xi_0(\zeta) \le S_{\min},$$

4) *(small condition number of new directions)* $g_{j,k} \le \sqrt{2}$
5) *(clustered eigenvalues)* Assumption 2.3 holds with $\tilde{g}_{\max}, \tilde{h}_{\max}$ small enough and $\tilde{c}_{\min}$ large enough s.t. $F(\tilde{g}_{\max}, \tilde{h}_{\max}, \tilde{c}_{\min}) > 0$ where $F(.)$ is defined in [25, Theorem 4.1],

then, with probability at least $1 - 2n^{-10}$, at all times, t,

1) $\hat{T}_t = T_t$ and $\|e_t\|_2 = \|L_t - \hat{L}_t\|_2 = \|\hat{S}_t - S_t\|_2 \le 0.18\sqrt{c}\gamma_{\text{new}} + 1.24\sqrt{\zeta}$.
2) the subspace error, $\text{SE}_{(t)} := \|(I - \hat{P}_{(t)}\hat{P}'_{(t)})P_{(t)}\|_2 \le$

$$\begin{cases} 0.6^{k-1} + r\zeta + 0.4c\zeta & \text{if } t \in [t_j, \tilde{t}_j - 1] \\ (r+c)\zeta & \text{if } t \in [\tilde{t}_j, \tilde{t}_j + \vartheta_j\tilde{\alpha} - 1] \\ r\zeta & \text{if } t \in [\tilde{t}_j + \vartheta_j\tilde{\alpha}, t_{j+1} - 1] \end{cases}$$
$$\le \begin{cases} 0.6^{k-1} + 10^{-2}\sqrt{\zeta} & \text{if } t \in [t_j, \tilde{t}_j - 1] \\ 10^{-2}\sqrt{\zeta} & \text{if } t \in [\tilde{t}_j, t_{j+1} - 1] \end{cases}$$

3) e_t follows a trend similar to that of $\text{SE}_{(t)}$ at various times (the bounds are available in [25, Theorem 4.1].

In words, the above result says the following. Assume that the initial subspace error is small enough. If (a) the

algorithm parameters are set appropriately; (b) the matrices whose columns span the previous subspace, the newly added subspace, and the currently unestimated parts of the previous and newly added subspaces are dense enough; (c) the subspace change is slow enough; (d) the condition number of the average covariance matrix of $a_{t,\text{new}}$ is small enough, and (e) the eigenvalues of $\text{Cov}(L_t)$ are clustered enough, then, w.h.p., we will get exact support recovery at all times. Moreover, the sparse recovery error (and the error in recovering L_t) will always be bounded by $0.18\sqrt{c}\gamma_{\text{new}}$ plus a constant times $\sqrt{\zeta}$. Since ζ is very small, and $\gamma_{\text{new}} \ll S_{\min}$, and c is also small, the normalized reconstruction error for S_t will be small at all times, thus making this a meaningful result.

V. Discussion

From Definition 5.2 of [25],

$$\alpha_{\text{add}}(\zeta) := \lceil (\log 6KJ + 11 \log n) \frac{8 \cdot 24^2}{(\zeta \lambda^-)^2} b \rceil$$

where $b = \max(\min(1.2^{4K}\gamma_{\text{new}}^4, \gamma_*^4), \frac{16}{c^2}, 4(0.186\gamma_{\text{new}}^2 + 0.0034\gamma_{\text{new}} + 2.3)^2)$ and $\alpha_{\text{del}}(\zeta)$ has a similar form.

Let us compare the above result with that for ReProCS for the current signal model [10, Corollary 43]. First, ReProCS requires $\kappa_{2s}([P_0, P_{1,\text{new}}, \dots P_{J,\text{new}}]) \le 0.3$ whereas ReProCS-cPCA only requires $\max_j \kappa_{2s}(P_j) \le 0.3$ which is significantly weaker. Moreover, ReProCS requires ζ to satisfy $\zeta \le \min(\frac{10^{-4}}{(r_0+(J-1)c)^2}, \frac{1.5\times 10^{-4}}{(r_0+(J-1)c)^2 f}, \frac{1}{(r_0+(J-1)c)^3\gamma_*^2})$ whereas in case of ReProCS-cPCA the denominators in the bound on ζ only contain $r + c = r_0 + 2c$ (instead of $r_0 + (J-1)c$). Because of the above, in Theorem 4.1 for ReProCS-cPCA, the only place where J (the number of subspace change times) appears is in the definitions of α_{add} and α_{del}. These, in turn, govern the delay between subspace change times, $t_{j+1} - t_j$. Thus, with ReProCS-cPCA, J can keep increasing, as long as α_{add} and α_{del}, and hence $t_{j+1} - t_j$, also increase accordingly. Moreover, the dependence of α_{add} and α_{del} on J is only logarithmic and thus $t_{j+1} - t_j$ needs to only increase in proportion to $\log J$. On the other hand, for ReProCS, J appears in the denseness assumption, in the bound on ζ and in the definition of α_{add}. Because of this, ReProCS requires a tight bound on J irrespective of how large $t_{j+1} - t_j$ is. The main extra assumption that ReProCS-cPCA needs is the clustering assumption. As explained in [25], this is practically valid.

A quantitative comparison with the PCP result of [1] is not possible since the proof techniques used are very different; we solve a recursive version of the problem whereas PCP solves a batch one; and the conclusions are different too. PCP provides guarantees for exact recovery of S_t's and L_t's. We obtain guarantees for exact support recovery of the S_t's and only bounded error recovery of their nonzero values and of L_t's. Also, ReProCS-cPCA requires knowledge of model parameters for subspace change of L_t, but PCP does not. Of course, in [25], we explain how to set the ReProCS-cPCA parameters in practice when the model is not known.

We can compare the two results qualitatively. The PCP result assumes independence of the support sets of S_t's but assumes nothing about L_t's whereas our result assumes a model on subspace change of the L_t's but nothing about the S_t's. Denseness assumptions are required by both, with those for PCP being stronger. These are compared in [25].

Simulation comparisons with PCP are given in [25].

References

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

[2] J. Wright and Y. Ma, "Dense error correction via l1-minimization," *IEEE Trans. on Info. Th.*, vol. 56, no. 7, pp. 3540–3560, 2010.

[3] J. Laska, M. Davenport, and R. Baraniuk, "Exact signal recovery from sparsely corrupted measurements through the pursuit of justice," in *Asilomar Conf. on Sig. Sys. Comp.*, Nov 2009, pp. 1556 –1560.

[4] N. H. Nguyen and T. D. Tran, "Robust lasso with missing and grossly corrupted observations," *To appear in IEEE Transaction on Information Theory*, 2012.

[5] M. McCoy and J. Tropp, "Two proposals for robust pca using semidefinite programming," *arXiv:1012.1086v3*, 2010.

[6] H. Xu, C. Caramanis, and S. Sanghavi, "Robust pca via outlier pursuit," *IEEE Tran. on Information Theorey*, vol. 58, no. 5, 2012.

[7] V. Chandrasekaran, S. Sanghavi, P. A. Parrilo, and A. S. Willsky, "Rank-sparsity incoherence for matrix decomposition," *SIAM Journal on Optimization*, vol. 21, 2011.

[8] C. Qiu and N. Vaswani, "Real-time robust principal components' pursuit," in *Allerton Conference on Communication, Control, and Computing*, 2010.

[9] ——, "Support-predicted modified-cs for principal components' pursuit," in *IEEE Intl. Symp. on Information Theory (ISIT)*, 2011.

[10] C. Qiu, N. Vaswani, and L. Hogben, "Recursive robust pca or recursive sparse recovery in large but structured noise," in *IEEE Intl. Conf. Acoustics, Speech, Sig. Proc. (ICASSP)*, 2013, longer version in arXiv: 1211.3754 [cs.IT].

[11] E. Candes, "The restricted isometry property and its implications for compressed sensing," *Compte Rendus de l'Academie des Sciences, Paris, Serie I*, pp. 589–592, 2008.

[12] T. Zhang and G. Lerman, "A novel m-estimator for robust pca," *arXiv:1112.4863v1*, 2011.

[13] M. B. McCoy and J. A. Tropp, "Sharp recovery bounds for convex deconvolution, with applications," *arXiv:1205.1580*.

[14] V. Chandrasekaran, B. Recht, P. A. Parrilo, and A. S. Willsky, "The convex geometry of linear inverse problems," *Foundations of Computational Mathematics*, no. 6, 2012.

[15] A. Ganesh, K. Min, J. Wright, and Y. Ma, "Principal component pursuit with reduced linear measurements," *arXiv:1202.6445*.

[16] M. Mardani, G. Mateos, and G. B. Giannakis, "Recovery of low-rank plus compressed sparse matrices with application to unveiling traffic anomalies," *arXiv:1204.6537*.

[17] D. Hsu, S. M. Kakade, and T. Zhang, "Robust matrix decomposition with outliers," *arXiv:1011.1518*.

[18] J. Wright, A. Ganesh, K. Min, and Y. Ma, "Compressive principal component pursuit," *arXiv:1202.4596*.

[19] M. Tao and X. Yuan, "Recovering low-rank and sparse components of matrices from incomplete and noisy observations," *SIAM Journal on Optimization*, vol. 21, no. 1, pp. 57–81, 2011.

[20] C. Davis and W. M. Kahan, "The rotation of eigenvectors by a perturbation. iii," *SIAM Journal on Numerical Analysis*, Mar. 1970.

[21] R. Horn and C. Johnson, *Matrix Analysis*. Cambridge University Press, 1985.

[22] J. A. Tropp, "User-friendly tail bounds for sums of random matrices," *Foundations of Computational Mathematics*, vol. 12, no. 4, 2012.

[23] B. Nadler, "Finite sample approximation results for principal component analysis: A matrix perturbation approach," *The Annals of Statistics*, vol. 36, no. 6, 2008.

[24] E. Candes and T. Tao, "Decoding by linear programming," *IEEE Trans. Info. Th.*, vol. 51(12), pp. 4203 – 4215, Dec. 2005.

[25] C. Qiu and N. Vaswani, "Recursive sparse recovery in large but structured noise - part 2," *arXiv: 1211.3754 [cs.IT]*, 2013.