PRACTICAL REPROCS FOR SEPARATING SPARSE AND LOW-DIMENSIONAL SIGNAL SEQUENCES FROM THEIR SUM – PART 1

Han Guo, Chenlu Qiu and Namrata Vaswani

ECE dept, Iowa State University, Ames, IA, USA
Email: \{hanguo,chenlu,namrata\}@iastate.edu

ABSTRACT
This paper designs and evaluates a practical algorithm, called Prac-ReProCS, for recovering a time sequence of sparse vectors \( S_t \) and a time sequence of dense vectors \( L_t \) from their sum, \( M_t := S_t + L_t \), when any subsequence of the \( L_t \)'s lies in a slowly changing low-dimensional subspace. A key application where this problem occurs is in video layering where the goal is to separate a video sequence into a slowly changing background sequence and a sparse foreground sequence that consists of one or more moving regions/objects. Prac-ReProCS is the practical analog of its theoretical counterpart that was studied in our recent work.

Index Terms— robust PCA, robust matrix completion, sparse recovery, compressed sensing

1. INTRODUCTION

The goal of this work is to recover a time sequence of sparse vectors \( S_t \) and a time sequence of dense vectors \( L_t \) from their sum, \( M_t := S_t + L_t \), when any subsequence of the \( L_t \)'s lies in a slowly changing low-dimensional subspace. The magnitude of the entries of \( L_t \) could be larger, roughly equal or smaller than that of the nonzero entries of \( S_t \).

The above problem can be interpreted as one of recursive sparse recovery from potentially large but structured noise. In this case, \( S_t \) is the quantity of interest and \( L_t \) is the potentially large but structured (low-dimensional) noise. Alternatively it can be posed as a recursive robust principal components analysis (PCA) problem. In this case \( L_t \), or in fact, the subspace in which the last several \( (d) \) \( L_t \)'s lie, \( \text{range}([L_{t-d+1}, \ldots, L_t]) \), is the quantity of interest while \( S_t \) is the outlier.

A key application where this problem occurs is in video layering where the goal is to separate a slowly changing background from moving foreground objects/regions (sparse image) [2, 3]. The foreground layer, e.g. moving people/objects, is of interest in applications such as automatic video surveillance, tracking moving objects, or video conferencing. The background sequence is of interest in applications such as background editing (video editing applications). In most static camera videos, the background images do not change much over time and hence the mean-subtracted background image sequence is well modeled as lying in a fixed or slowly-changing low-dimensional subspace of \( \mathbb{R}^n \) [3]. Moreover the changes are typically global, e.g. due to lighting variations, and hence modeling it as a dense image sequence is valid too. The foreground layer usually consists of one or more moving objects/persons/regions that move in a correlated fashion, i.e. it is a sparse image sequence that often changes in a correlated fashion over time. By letting \( M_t \) be the entire image, \( L_t \) be the background image and defining \( S_t \) as the the foreground-background intensity difference on the foreground support and zero everywhere else, video layering becomes a problem of separating \( S_t \) and \( L_t \) from \( M_t = S_t + L_t \).

Related Work. In the last few decades, there has been a large amount of work on robust PCA, e.g. [2, 4, 5, 6], and recursive robust PCA e.g. [7, 8, 9]. In most of these works, either the locations of the missing/corruped data points are assumed known [7] (not a practical assumption); or they first detect the corrupted data points and then replace their values using nearby values [8]; or weight each data point in proportion to its reliability (thus soft-detecting and down-weighting the likely outliers) [2, 9]; or just remove the entire outlier vector [5, 6].

In a series of recent works [3, 10], a new and provably correct solution to robust PCA called Principal Components Pursuit (PCP) has been proposed, that does not require a two step outlier location detection/correction process and also does not throw out the entire vector. It redefines batch robust PCA as a problem of separating a low rank matrix \( \mathcal{L} \) and a sparse matrix \( \mathcal{S} \) from their sum \( \mathcal{M} \). PCP finds \( \mathcal{S} \) and \( \mathcal{L} \) by solving \( \min_{\mathcal{S}, \mathcal{L}} \| \mathcal{S} \|_1 + \| \mathcal{L} \|_* \) s.t. \( \mathcal{M} = \mathcal{S} + \mathcal{L} \) where \( \| . \|_1 \) denotes the vector \( \ell_1 \) norm and \( \| . \|_* \) denotes the nuclear norm. It is shown that if the low-rank matrix is dense and if the sparse matrix has support set entries that are independently selected, then solving PCP will indeed return the correct sparse and low-rank matrices. Other recent works that also study batch algorithms for recovering a sparse matrix and a low-rank matrix from their sum, or from undersampled measurements of their sum, include [11, 12, 13, 14, 15, 16, 17, 18].

Notice that most applications where video layering is required, such as video surveillance, require an online solu-
tion. A batch solution would require a long delay; and would also be much slower than a recursive solution. Moreover, the assumption that the foreground support is independent over time is not usually valid. To address these issues, in [19] we introduced a novel recursive solution approach which we later called Recursive Projected Compressive Sensing (ReProCS) [20]. In recent work [21, 22], we have tried to obtain performance guarantees for ReProCS. Under mild assumptions and an assumption on an algorithm estimate (that holds in simulations as long as there is some support change every few frames), we showed that, with high probability, ReProCS can exactly recover the support set of $S_t$.

We assume that the subspace spanned by the last several $t$’s, and gradual support change of $S_t$’s. We show via extensive simulation experiments that ReProCS is more robust to correlated support change of $S_t$’s, and slow subspace change of $L_t$’s, and gradual support change of $S_t$’s. We show via extensive simulation experiments that ReProCS is more robust to correlated support change of $S_t$ than PCP and other existing work. Also, it is also able to recover small magnitude sparse vectors better than other existing works. The simulation experiments are shown in this paper; the model verification and real video experiments are shown in longer version of this paper [1].

Some later work of this topic includes [23]. Its key idea is similar to that of the original ReProCS algorithm [19, 20].

Notation. For a set $T \subseteq \{1, 2, \cdots n\}$, we use $|T|$ to denote its cardinality, i.e., the number of elements in $T$. The symbols $\cup, \cap, \setminus$ denote set union set intersection and set difference respectively. The notation $[.]$ denotes an empty matrix. We use the notation $B^{SVD} \equiv U^T \Sigma V$ to denote the singular value decomposition (SVD) of $B$, and range($B$) denotes the subspace spanned by the columns of $B$.

A matrix $P$ is a basis matrix if $P^T P = I$.

The notation $Q = \text{basis}(\text{range}(M))$, or $Q = \text{basis}(M)$ for short, means that $Q$ is a basis matrix for range($M$) i.e. $Q$ satisfies $Q^T Q = I$ and range($Q$) = range($M$).

The $b\%$ left singular values’ set of a matrix $M$ is the smallest set of indices of its singular values that contains at least $b\%$ of the total singular values’ energy. The corresponding matrix of left singular vectors, $U_T$, is referred to as the $b\%$ left singular vectors’ matrix.

Definition 1.1 The notation $Q = \text{approx-basis}(M,b\%)$ means that $Q$ is the $b\%$ left singular vectors’ matrix for $M$. The notation $Q = \text{approx-basis}(M,r)$ means that $Q$ contains the left singular vectors of $M$ corresponding to its $r$ largest singular values.

2. PROBLEM DEFINITION AND ASSUMPTIONS

The measurement vector at time $t$, $M_t$, is an $n$ dimensional vector which can be decomposed as

$$M_t = S_t + L_t$$  \hspace{0.5cm} (1)

where $S_t$ is a sparse vector and $L_t$ is a dense but low-dimensional vector. We use $T_t$ to denote the support set of $S_t$.

Suppose that an initial training sequence which does not contain the sparse components is available, i.e. we are given $M_{\text{train}} = [M_t; 1 \leq t \leq t_{\text{train}}]$ with $M_t = L_t$. This is used to get an initial estimate of the subspace in which the $L_t$’s lie $^1$. At each $t > t_{\text{train}}$, the goal is to recursively estimate $S_t$ and $L_t$ and the subspace in which the last several $L_t$’s lie. By "recursively" we mean: use $S_{t-1}, L_{t-1}$ and the previous subspace estimate to estimate $S_t$ and $L_t$.

Our algorithm is based on three assumptions that we explain next. These assumptions are verified for real video data in [1].

2.1. Low-dimensionality and slow subspace change

One way to quantitify this assumption is as follows. We let $L_t = P(t) \alpha_t$ where $P(t)$ is a tall matrix that is piecewise constant with time, i.e. $P_{t} = P_{t'}$ for all $t \in [t_j, t_j+1)$ where $P_j$ is an $n \times r_j$ basis matrix with $r_j \ll \min((t_{j+1} - t_{j}), n)$. A very simple model for slow subspace change is to let $P_j$ change as

$$P_j = [(P_{j-1} R_j \setminus P_{j,\text{old}}), P_{j,\text{new}}]$$

where $P_{j,\text{new}}$ and $P_{j,\text{old}}$ are basis matrices of size $n \times c_{j,\text{new}}$ and $n \times c_{j,\text{old}}$ respectively with $P_{j,\text{new}} P_{j-1} = 0$ and $R_j$ is a rotation matrix. Moreover, the projection of $L_t$ along $P_{j,\text{new}}$ is small initially for the first $\alpha$ frames, i.e.

$$\| (I - P_{j-1} P_{j-1}^T) L_t \|_2 \ll \min (\| L_t \|_2, \| S_t \|_2) \text{ if } t \in [t_j, t_j+\alpha)$$

and can increase gradually after $t_j + \alpha$.

2.2. Denseness

We assume that the subspace spanned by the $L_t$’s is dense, i.e.

$$\kappa_{2s}(P_j) = \kappa_{2s}(\{L_{t_j}, \ldots, L_{t_{j+1}}\}) \leq \kappa_s$$

for a $\kappa_s$ significantly smaller than one. Here

$$\kappa_s(B) = \kappa_s(\text{range}(B)) := \max_{|T| \leq s} \| I_T \text{basis}(B) \|_2$$

is the denseness coefficient for any vector or matrix $B$ [21, 22]. Moreover, a similar assumption holds for $P_{j,\text{new}}$ with a

---

$^1$If an initial sequence without $S_t$’s is not available, one can use a batch robust PCA algorithm to get the initial subspace estimate as long as the initial sequence satisfies its required assumptions.
tighter bound: \( \kappa_{2s}(P_{j,\text{new}}) \leq \kappa_{\text{new}} < \kappa_s \). By [21, Lemma 2], a small \( \kappa_{2s}(P_j) \) means that the restricted isometry constant (RIC) [24] of the matrix \((I - P_jP_j')\) is small. Using any of the RIC based sparse recovery results, e.g. [25], this ensures that for \( t \in [t_j, t_{j+1}) \), \( s \)-sparse vectors \( S_t \) are recoverable from \((I - P_jP_j')M_t = (I - P_jP_j)S_t \) by \( \ell_1 \) minimization.

2.3. Support size, support change of \( S_t \)

We assume two things. First, we assume that either the support size is small or the support changes are slow or both. At the same time, we also assume that there is some support change during any set of \( \alpha \) frames. Practically, this is needed to ensure that at least some of the foreground behind the foreground is visible so that the changes to the background subspace can be estimated. In the performance guarantees derived in [21], this ensures that the currently unestimated subspace of range(\( P_{j,\text{new}} \)) is dense.

3. PRACTICAL REPROCS

The complete practical Recursive Projected Compressive Sensing (ReProCS) algorithm is summarized in Algorithm 1. We explain its steps below. We use \( \hat{S}_t, T_t, L_t \) to denote estimates of \( S_t \), its support, \( T_t \), and \( L_t \) respectively; and we use \( \hat{P}_t \) to denote the basis matrix of the estimated subspace of the last several \( L_t \)’s (sometimes we just refer to \( \hat{P}_t \) as the subspace estimate). Also, let

\[
\beta_t := \Phi_t(t) L_t, \quad \text{where } \Phi_t(t) := (I - \hat{P}_{t-1}\hat{P}_{t-1}')(3)
\]

Given the initial training sequence which does not contain the sparse components, \( M_{\text{train}} = [L_1, L_2, \ldots, L_{\text{train}}] \) we compute \( \hat{P}_0 \) as an approximate basis for \( M_{\text{train}} \), i.e. \( \hat{P}_0 = \text{approx-basis}(M_{\text{train}}, 0\%) \) with \( 0\% \approx 95\% \). Also let \( \hat{r} = \text{rank}(\hat{P}_0) \). We need to compute an approximate basis because for real data, the \( L_t \)'s are only approximately low-dimensional. After this, at each time \( t \), ReProCS involves 4 steps that we explain next.

**Perpendicular Projection.** At time \( t \), we project the measurement vector, \( M_t \), into the space orthogonal to \( \hat{P}_{t-1} \) to get \( y_t := \Phi_t(t)M_t \). As we explain in the Subspace Update step, \( \hat{P}_t \) is updated every \( \alpha \) frames.

**Sparse Recovery (Recover \( T_t \) and \( S_t \).** With the above projection, \( y_t \) can be rewritten as

\[
y_t = \Phi_t(t)S_t + \beta_t
\]

where \( \beta_t \) is defined in (3). As explained in [1, 21], \( \|\beta_t\|_2 \) is small. Briefly, if the current subspace is accurately estimated, then this is because projecting orthogonal to range(\( P_{t-1} \)) nullifies most of the contribution of \( L_t \); on the other hand, if range(\( P_{j,\text{new}} \)) has so far not been estimated, then this is still true because of the slow subspace change assumption. As a result the problem of recovering \( S_t \) from \( y_t \) becomes a traditional sparse recovery / CS problem in small noise, \( \beta_t \). Notice that, since the \( n \times n \) projection matrix, \( \Phi(t) \), has rank \( (n - \text{rank}(\hat{P}_{t-1})) \), therefore \( y_t \) has only this many “effective” measurements, even though its length is \( n \).

To recover \( S_t \) from \( y_t \), we solve

\[
\min_x \|x\|_1 \quad \text{s.t.} \quad y_t - \Phi(t) x_2 \leq \xi \tag{4}
\]

and denote its solution by \( \hat{S}_{t,c} \). By the denseness assumption, the basis matrix \( P_{t-1} \) is dense. Since \( P_{t-1} \) approximates it, this is true for \( \hat{P}_{t-1} \) as well [21, Lemma 8.3]. Thus, by [21, Lemma 2], the restricted isometry constant (RIC) of \( \Phi(t) \) is small. Using [25, Theorem 1], this and the fact that \( \beta_t \) is small ensures that \( \hat{S}_t \) can be accurately recovered from \( y_t \). By thresholding on \( \hat{S}_{t,c} \) to get an estimate of its support followed by computing a least squares (LS) estimate of \( S_t \) on the estimated support and setting it to zero everywhere else, we can get a more accurate estimate, \( \hat{S}_t \). The thresholding and LS help to reduce the bias and total reconstruction error in the solution.

The constraint \( \xi \) used in the minimization should equal \( \|\beta_t\|_2 \) or its upper bound. Since \( \beta_t \) is unknown we can replace \( \|\beta_t\|_2 \) by \( \hat{\beta}_t \) where \( \hat{\beta}_t := \Phi(t)\hat{L}_{t-1} \). This will usually be smaller than the upper bound on \( \|\beta_t\|_2 \). However that only means that the solution of (4) may have some extra nonzero elements. With an appropriate thresholding step, most of these should not be detected into the support.

**Recall \( L_t \).** The estimate \( \hat{S}_t \) is used to estimate \( L_t \) as

\[
\hat{L}_t = M_t - \hat{S}_t.
\]

Thus, if \( S_t \) is recovered accurately, so will \( L_t \).

**Subspace Update (Update \( P_t \)).** Within a short delay after every subspace change time, one needs to update the subspace estimate, \( \hat{P}_t \). In practice, since the subspace model used in this paper is known, the subspace update needs to be done at regular short enough intervals. This is needed to ensure that the subspace gets updated quickly enough so that the projected noise \( \beta_t \) seen by the sparse recovery step never becomes too large. At the same time, to get an accurate subspace estimate using simple PCA, one needs to use \( d \) frames for a \( d \) that is large enough compared to \( r_j \). To satisfy both requirements, we use overlapping periods for subspace estimation: every \( \alpha \) frames, we do a subspace update using the previous \( d \) estimates \( \hat{L}_t \) with a \( d \gg \alpha \). To be precise at every \( t = t_{\alpha} + k\alpha, k = 1, 2, \ldots \), we compute \( \hat{P}(t) = \text{approx-basis}(\hat{L}_{t-d+1}, \ldots, \hat{L}_t, \hat{r}) \) where \( \hat{r} = \text{rank}(\hat{P}_0) \). The choice of \( \alpha \) is governed by computational complexity. In the experiments shown, we used \( d = 10^2 \) and \( \alpha = 50 \).

The subspace update step can be made recursive as explained in [1]. Alternatively, one can use projection PCA introduced in [21] (practical version explained in [1]). Experiments using these are shown in [1].

**Improved Sparse Recovery.** Whenever slow support change holds, one can replace \( \ell_1 \) minimization by modified-CS [26] or its generalization called weighted \( \ell_1 \) [27, 28].
For Definition 3.1

In the algorithm, estimation can be improved by using an approach similar to enough estimate of the current support, \( \hat{T}_{i-1} \), when the previous support estimate, \( \hat{T}_i \), These require fewer measurements for exact/accurate recovering (see [1]). The moving object was simulated as explained in [1]. We generated 50 realizations of the video sequence and compared all the algorithms to estimate \( S_t \) small or of the same order as \( L_t \) (making it a difficult sequence). We controlled the foreground intensity so that \( ||S_t||_2 \) was roughly equal or smaller than \( ||L_t||_2 \) making it a difficult sequence. Moreover it provides ground truth data so that the recovery performance can be quantitatively compared.

The background was a video of moving waters in a lake (see [1]). The moving object was simulated as explained in [1]. We generated 50 realizations of the video sequence and compared all the algorithms to estimate \( S_t, L_t \) and then the foreground and the background sequences. We show comparisons of the normalized mean squared error (NMSE) in recovering \( S_t \) in Fig. 1. As can be seen, the ReProCS error is the smallest and stable. PCP gives very large error for this sequence since the object moves in a highly correlated fashion and overlaid a simulated foreground sequence consisting of a moving rectangular object on it. The use of a real background sequence allows us to evaluate performance for data that only approximately satisfies the low-dimensional and slow subspace change assumptions. The use of the simulated foreground allows us to control its intensity so that the resulting \( S_t \) is small or of the same order as \( L_t \) (making it a difficult sequence). We controlled the foreground intensity so that \( ||S_t||_2 \) was roughly equal or smaller than \( ||L_t||_2 \) making it a difficult sequence.

We used a real slowly changing background sequence and overlaid a simulated foreground sequence consisting of a moving rectangular object on it. The use of a real background sequence allows us to evaluate performance for data that only approximately satisfies the low-dimensional and slow subspace change assumptions. The use of the simulated foreground allows us to control its intensity so that the resulting \( S_t \) is small or of the same order as \( L_t \) (making it a difficult sequence). We controlled the foreground intensity so that \( ||S_t||_2 \) was roughly equal or smaller than \( ||L_t||_2 \) making it a difficult sequence.

Moreover it provides ground truth data so that the recovery performance can be quantitatively compared.

The background was a video of moving waters in a lake (see [1]). The moving object was simulated as explained in [1]. We generated 50 realizations of the video sequence and compared all the algorithms to estimate \( S_t, L_t \) and then the foreground and the background sequences. We show comparisons of the normalized mean squared error (NMSE) in recovering \( S_t \) in Fig. 1. As can be seen, the ReProCS error is the smallest and stable. PCP gives very large error for this sequence since the object moves in a highly correlated fashion and occupies a large part of the image. GRASTA [23] also does not work and we think it is because the GRASTA code does not update the background subspace. Robust Subspace Learning (RSL) [2] is able to recover a large part of the object correctly, however it also recovers many more extras than ReProCS. The reason is that the magnitude of the nonzero entries of \( S_t \) is equal or small compared to those of \( L_t \).

Real video experiments are available in [1] and http://www.ece.iastate.edu/~hanguo/PracReProCS.html.
5. REFERENCES


