Summer 2019

Subspace tracking from missing and corrupted data using NORST and its heuristic extensions

Vahid Daneshpajooh

Subspace tracking from Missing and Corrupted Data Using NORST and its Heuristic Extensions

Vahid Daneshpajooh

July 2019

Abstract

We study the problem of subspace tracking (ST) in the presence of missing data (ST-miss). In recent work, we have studied the Robust ST (RST) problem. In this work, we show that a simple modification of our solution approach for RST also provably solves ST-miss under weaker assumptions. To our knowledge, our result is the first complete guarantee for ST-miss. This means we are able to show that, under assumptions on only the algorithm inputs (input data and/or initialization), the output subspace estimates are close to the true data subspaces at all times. Our guarantees hold under mild and easily interpretable assumptions and handle time-varying subspaces (unlike all previous work). We also show that our algorithm and its extensions are fast and have competitive experimental performance when compared with existing methods. Finally, our solution can be interpreted as a provably correct mini-batch and memory-efficient solution to low rank Matrix Completion (MC).

1 Introduction

Subspace tracking from missing data (ST-miss) is the problem of tracking the (fixed or time-varying) low-dimensional subspace in which a given data sequence approximately lies when some of the data entries are not observed. The assumption here is that consecutive subsets of the data are well-approximated as lying in a subspace that is significantly lower-dimensional than the ambient dimension. Time-varying subspaces is a more appropriate model for long data sequences (e.g. long surveillance videos). For such data, if a fixed subspace model is used, the required subspace dimension may be too large. As is common in time-series analysis, the simplest model for time-varying quantities is to assume that they are piecewise constant with time. We adopt this model here. If the goal is to provably track the subspaces to any desired accuracy, $\epsilon > 0$, then, this assumption is, in fact, necessary. Of course, experimentally, the proposed algorithm, and all existing ones, "work" (return good but not perfect estimates) even without this assumption, as long as the amount of change at each time is small enough. The reason is one can interpret subspace changes at each time as a "piecewise constant subspace" plus noise. The algorithms are actually tracking the "piecewise constant subspace" up to the noise level.

ST-miss can be interpreted as an easier special case of robust ST (ST in the presence of additive sparse outliers) [4]. We also study robust ST-miss which is a generalization of both ST-miss and robust ST. Finally, the solutions for ST-miss and robust ST-miss also provide novel mini-batch solutions for low-rank matrix completion (MC) and robust MC respectively.

Example applications where these problems occur include recommendation system design and video analytics. In video analytics, foreground occlusions are often the source of both missing and corrupted data: if the occlusion is easy to detect by simple means, e.g., color-based thresholding, then the occluding pixel can be labeled as "missing"; while if this cannot be detected easily, it is labeled as an outlier pixel. Missing data also occurs due to detectable video transmission errors (typically called "erasures"). In recommendation systems, data is missing because all users do not label all items. In this setting, time-varying subspaces model the fact that, as different types of users enter the system, the factors governing user preferences change.

Brief review of related work

ST has been extensively studied in both the controls' and the signal processing literature, see [5–8] for comprehensive overviews of both classical and modern approaches. Best known existing algorithms for ST and ST-miss include Projection Approximate Subspace Tracking (PAST) [9, 10], Parallel Estimation and Tracking by Recursive Least Squares (PETRELS) [11] and Grassmannian Rank-One Update Subspace Estimation (GROUSE) [12–15]. Of these, PETRELS is known to have the best experimental performance. There have been some attempts to obtain guarantees for GROUSE and PETRELS for ST-miss [13, 14, 16], however all of these results assume the statistically stationary setting of a *fixed unknown subspace* and all of them provide only *partial guarantees.* This means that the result does not tell us what assumptions the algorithm inputs (input data and/or initialization) need to satisfy in order to ensure that the algorithm output(s) are close to the true value(s) of the quantity of interest, either at all times or at least at certain times. The advantage of GROUSE and PETRELS is that they are streaming solutions (require a single-pass through the data). This may also be the reason that a complete guarantee is harder to obtain for these. Other related work includes streaming PCA with missing data [17, 18]. A provable algorithmic framework for robust ST is Recursive Projected Compressive Sensing (ReProCS) [4, 19–22]. Robust ST-miss has not received much attention in the literature.Provable MC has been extensively studied, e.g., [23–25].

1.1 Notation

We use the interval notation $[a, b]$ to refer to all integers between a and b, inclusive, and we use $[a, b) := [a, b-1]$. $\|.\|$ denotes the l_2 norm for vectors and induced l_2 norm for matrices unless specified otherwise, and $'$ denotes transpose. We use $\boldsymbol{M}_{\mathcal{T}}$ to denote a sub-matrix of $\boldsymbol{M}$ formed by its columns indexed by entries in the set $\mathcal{T}$. For a matrix $\boldsymbol{P}$ we use $\boldsymbol{P}^{(i)}$ to denote its i-th row.

A matrix $\boldsymbol{P}$ with mutually orthonormal columns is referred to as a *basis matrix* and is used to represent the subspace spanned by its columns. For basis matrices $\boldsymbol{P}_1, \boldsymbol{P}_2$, we use $\text{dist}(\boldsymbol{P}_1, \boldsymbol{P}_2) := \|(\boldsymbol{I} - \boldsymbol{P}_1\boldsymbol{P}_1{}')\boldsymbol{P}_2\|$ as a measure of Subspace Error (distance) between their respective subspaces. This is equal to the sine of the largest principal angle between the subspaces. If $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ are of the same dimension, $\text{dist}(\boldsymbol{P}_1, \boldsymbol{P}_2) = \text{dist}(\boldsymbol{P}_2, \boldsymbol{P}_1)$.

We use $\hat{\boldsymbol{L}}_{t;\alpha} := [\hat{\boldsymbol{\ell}}_{t-\alpha+1}, \cdots, \hat{\boldsymbol{\ell}}_t]$ to denote the matrix formed by $\hat{\boldsymbol{\ell}}_t$ and $(\alpha - 1)$ previous estimates. Also, r-SVD$[\boldsymbol{M}]$ refers to the matrix of top r left singular vectors of $\boldsymbol{M}$.

A set Ω that is randomly sampled from a larger set (universe), $\mathcal{U}$, is said be *"i.i.d. Bernoulli with parameter ρ"* if each entry of $\mathcal{U}$ has probability ρ of being selected to belong to Ω independent of all others.

We reuse C, c to denote different numerical constants in each use; C is for constants greater than one and c for those less than one.

1.2 Problem Statement

ST-miss is precisely defined as follows. At each time t, we observe a data vector $\boldsymbol{y}_t \in \mathbb{R}^n$ that satisfies

$$\boldsymbol{y}_t = \mathcal{P}_{\Omega_t}(\boldsymbol{\ell}_t) + \boldsymbol{\nu}_t, \text{ for } t = 1, 2, \ldots, d \tag{1}$$

where $\mathcal{P}_{\Omega_t}(\boldsymbol{z}_i) = \boldsymbol{z}_i$ if $i \in \Omega_t$ and 0 otherwise. Here $\boldsymbol{\nu}_t$ is small unstructured noise, Ω_t is the set of observed entries at time t, and $\boldsymbol{\ell}_t$ is the true data vector that lies in a fixed or changing low (r) dimensional subspace of $\mathbb{R}^n$, i.e., $\boldsymbol{\ell}_t = \boldsymbol{P}_{(t)}\boldsymbol{a}_t$ where $\boldsymbol{P}_{(t)}$ is an $n \times r$ basis matrix with $r \ll n$. The goal is to track $\text{span}(\boldsymbol{P}_{(t)})$ and $\boldsymbol{\ell}_t$ either immediately or within a short delay. Denoting the set of missing entries at time t as $\mathcal{T}_t$, (1) can also be written as

$$\boldsymbol{y}_t := \boldsymbol{\ell}_t - \boldsymbol{I}_{\mathcal{T}_t}\boldsymbol{I}_{\mathcal{T}_t}{}'\boldsymbol{\ell}_t + \boldsymbol{\nu}_t. \tag{2}$$

We use $\boldsymbol{z}_t := -\boldsymbol{I}_{\mathcal{T}_t}{}'\boldsymbol{\ell}_t$ to denote the missing entries. Clearly, $\mathcal{T}_t = (\Omega_t)^c$ (here ^{c} denotes the complement set w.r.t. $\{1, 2, \ldots, n\}$). Writing $\boldsymbol{y}_t$ as above allows us to tap into the solution framework from earlier work [4, 20]. This was developed originally for solving robust ST which involves tracking $\boldsymbol{\ell}_t$ and $\boldsymbol{P}_{(t)}$ from $\boldsymbol{y}_t := \boldsymbol{\ell}_t + \boldsymbol{\nu}_t + \boldsymbol{x}_t$ where $\boldsymbol{x}_t$ is a sparse vector with the outliers as its nonzero entries. ST-miss can

be interpreted as its (simpler) special case if we let $\boldsymbol{x}_t = -\boldsymbol{I}_{\mathcal{T}_t}\boldsymbol{I}_{\mathcal{T}_t}{}'\boldsymbol{\ell}_t$. It is simpler because the support of $\boldsymbol{x}_t$, $\mathcal{T}_t$, is known.

Defining the $n \times d$ matrix $\boldsymbol{L} := [\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \dots \boldsymbol{\ell}_d]$, the above is also a matrix completion (MC) problem; with the difference that for MC the estimates are needed only in the end (not on-the-fly). We use r_L to denote the rank of $\boldsymbol{L}$.

2 The NORST-miss algorithm and guarantees

The basic algorithm is discussed next. Then the guarantee for the noise-free $\boldsymbol{\nu}_t = 0$ case is mentioned in Sec. 2.2. Extensions of basic NORST-miss are given in Sec. 2.3.

2.1 NORST-miss algorithm

The complete psedo-code for the algorithm is provided in Algorithm 1. After initialization, the algorithm iterates between a *projected Least Squares (LS)* step and a *Subspace Update (including Change Detect)* step. Broadly, projected LS estimates the missing entries of $\boldsymbol{\ell}_t$ at each time t. Subspace update toggles between the "update" phase and the change "detect" phase. In the update phase, it improves the estimate of the current subspace using a short mini-batch of "filled in" versions of $\boldsymbol{\ell}_t$. In the detect phase, it uses these to detect subspace change.

Initialization: The algorithm starts in the "update" phase and with zero initialization: $\hat{\boldsymbol{P}}_0 \leftarrow \boldsymbol{0}_{n\times r}$. For the first α frames, the projected LS step (explained below) simply returns $\hat{\boldsymbol{\ell}}_t = \boldsymbol{y}_t$. Thus, a simpler way to understand the initialization is as follows: wait until $t = \alpha$ and then compute the first estimate of $\text{span}(\boldsymbol{P}_0)$ as the r-SVD (matrix of top r left singular vectors) of $[\boldsymbol{y}_1, \boldsymbol{y}_2, \dots \boldsymbol{y}_\alpha]$. This step is solving a PCA with missing data problem which, as explained in [26], can be interpreted as a problem of PCA in sparse data-dependent noise. Because we assume that the number of missing entries at any time t is small enough, and the set of missing entries changes sufficiently over time (Equivalently, we bound the maximum number of missing entries in any column and in any row of the data matrix), we can prove that this step gives a good first estimate of the subspace.

Projected LS: Recall that NORST-miss is a modification of NORST for robust ST from [4]. In robust ST, sudden subspace changes cannot be detected because these are confused for outliers. Its projected-LS step is thus designed using a slow (small) subspace change assumption. However, as we will explain later, for the current missing data setting, it also works in case of sudden changes. Suppose that the previous subspace estimate, $\hat{\boldsymbol{P}}_{(t-1)}$, is a "good enough" estimate of the previous subspace $\boldsymbol{P}_{(t-1)}$. Under slow subspace change, it is valid to assume that $\text{span}(\boldsymbol{P}_{(t-1)})$ is either equal to or close to $\text{span}(\boldsymbol{P}_{(t)})$. Thus, under this assumption, it is a good idea to project $\boldsymbol{y}_t$ onto the orthogonal complement of $\hat{\boldsymbol{P}}_{(t-1)}$ because this will nullify most of $\boldsymbol{\ell}_t$, i.e., the not-nullified part of $\boldsymbol{\ell}_t$, $\boldsymbol{b}_t := \boldsymbol{\Psi}\boldsymbol{\ell}_t$, will be small. Here $\boldsymbol{\Psi} := \boldsymbol{I} - \hat{\boldsymbol{P}}_{(t-1)}\hat{\boldsymbol{P}}_{(t-1)}{}'$. Using this idea, we compute $\tilde{\boldsymbol{y}}_t := \boldsymbol{\Psi}\boldsymbol{y}_t = \boldsymbol{\Psi}_{\mathcal{T}_t}\boldsymbol{z}_t + \boldsymbol{b}_t + \boldsymbol{\Psi}\boldsymbol{\nu}_t$. Estimating $\boldsymbol{z}_t$ can be interpreted as a LS problem $\min_{\boldsymbol{z}} \|\tilde{\boldsymbol{y}}_t - \boldsymbol{\Psi}_{\mathcal{T}_t}\boldsymbol{z}\|^2$. Solving this gives

$$\hat{\boldsymbol{z}}_t = \left(\boldsymbol{\Psi}_{\mathcal{T}_t}{}'\boldsymbol{\Psi}_{\mathcal{T}_t}\right)^{-1}\boldsymbol{\Psi}_{\mathcal{T}_t}{}'\tilde{\boldsymbol{y}}_t. \tag{3}$$

Next, we use this to compute $\hat{\boldsymbol{\ell}}_t = \boldsymbol{y}_t - \boldsymbol{I}_{\mathcal{T}_t}\hat{\boldsymbol{z}}_t$. Observe that the missing entries $\boldsymbol{z}_t$ are recoverable as long as $\boldsymbol{\Psi}_{\mathcal{T}_t}$ is well-conditioned. A necessary condition for this is $(n-r) > |\mathcal{T}_t|$. As we will see later, a sufficient condition is $|\mathcal{T}_t| < cn/r$ because this ensures that the restricted isometry constant (RIC) [27] of $\boldsymbol{\Psi}$ of level $|\mathcal{T}_t|$ is small.

In settings where $\text{span}(\boldsymbol{P}_{(t-1)})$ is *not* close to $\text{span}(\boldsymbol{P}_{(t)})$ (sudden subspace change), the above approach still works. Of course, in this case, it is not any better (or worse) than re-initialization to zero, because, in this case, $\|\boldsymbol{\Psi}\boldsymbol{\ell}_t\|$ is of the same order as $\|\boldsymbol{\ell}_t\|$. We can use the same arguments as those

used for the initialization step to argue that the first subspace update works even in this case.

Subspace Update: The $\hat{\boldsymbol{\ell}}_t$'s are used for subspace update. In its simplest (and provably correct) form, this is done once every α frames by r-SVD on the matrix formed by the previous α $\hat{\boldsymbol{\ell}}_t$'s. Let $\hat{t}_j$ be the time at which the j-th subspace change is detected (let $\hat{t}_0 := 0$). For each $k = 1, 2, \dots, K$, at $t = \hat{t}_j + k\alpha - 1$, we compute the r-SVD of $\hat{\boldsymbol{L}}_{t;\alpha}$ to get $\hat{\boldsymbol{P}}_{j,k}$ (k-th estimate of subspace $\boldsymbol{P}_j$). After K such updates, i.e., at $t = \hat{t}_j + K\alpha - 1 := \hat{t}_{j,fin}$ the update is complete and the algorithm enters the "detect" phase. Each update step is a PCA in sparse data-dependent noise problem. This allows us to use the result from [26] to show that, as long as the missing entries' set changes enough over time (max-miss-frac-row$_\alpha$ is bounded for each interval), each update step reduces the subspace recovery error to 0.3 times its previous value. Thus, by setting $K = C\log(1/\epsilon)$, one can show that, after K updates, the subspace is recovered to ϵ accuracy.

Subspace change detect: To simply understand the detection strategy, assume that the previous subspace $\boldsymbol{P}_{j-1}$ has been estimated to ϵ accuracy by $t = \hat{t}_{j-1,fin} = \hat{t}_{j-1} + K\alpha - 1$ and denote it by $\hat{\boldsymbol{P}}_{j-1} := \hat{\boldsymbol{P}}_{j-1,K}$. Also assume that $\boldsymbol{\nu}_t = 0$. At every $t = \hat{t}_{j-1,fin} + u\alpha - 1$, $u = 1, 2, \dots,$ we detect change by checking if the maximum singular value of the matrix $(\boldsymbol{I} - \hat{\boldsymbol{P}}_{j-1}\hat{\boldsymbol{P}}_{j-1}{}')\hat{\boldsymbol{L}}_{t;\alpha}$ is above a pre-set threshold, $\sqrt{\omega_{evals}\alpha}$, or not. This works because, if the subspace has not changed, this matrix will have all singular values of order $\epsilon\sqrt{\lambda^+}$. If it has changed, its largest singular value will be at least $\text{dist}(\boldsymbol{P}_{j-1}, \boldsymbol{P}_j)\sqrt{\lambda^-}$. By picking ϵ small enough, one can ensure that, whp, all changes are detected.

NORST-miss-smoothing for MC: The above is the tracking/online/filtering mode of NORST-miss. It outputs an estimate of $\boldsymbol{\ell}_t$ as soon as a new measurement vector $\boldsymbol{y}_t$ arrives and an estimate of the subspace every α frames. Notice that, order-wise, α is only a little more than r which is the minimum delay needed to compute the subspace even if perfect data $\boldsymbol{y}_t = \boldsymbol{\ell}_t$ were available. Once an ϵ-accurate estimate of the current subspace is available, one can improve all past estimates of $\boldsymbol{\ell}_t$ to ensure that all estimates are ϵ-accurate. This is called the smoothing mode of operation. To be precise, this is done as given in line 25 of Algorithm 1. This allows us to get a completed matrix $\hat{\boldsymbol{L}}$ with all columns being ϵ-accurate.

Memory Complexity: In online or filtering mode, NORST-miss needs $\alpha = O(r\log n)$ frames of storage. In smoothing mode, it needs $\mathcal{O}((K+2)\alpha) = \mathcal{O}(r\log n\log(1/\epsilon))$ frames of memory. Therefore its memory complexity, even in the smoothing mode, is just $\mathcal{O}(nr\log n\log(1/\epsilon))$. Thus, it provides a nearly memory-optimal mini-batch solution for MC.

Algorithm parameters: The algorithm has 4 parameters: r, K, α, and ω_{evals}. Theoretically these are set as follows: assume that r, λ^+, λ^- are known and pick a desired recovery error ϵ. Set $\alpha = C_1 f^2 r\log n$ with $f = \lambda^+/\lambda^-$, $K = C_2\log(1/\epsilon)$ and $\omega_{evals} = c\lambda^-$ with c a small constant. We explain practical approaches in Sec 4.

2.2 Main Result: noise-free ST-miss and MC

First, for simplicity, consider the noise-free case, i.e., assume $\boldsymbol{\nu}_t = 0$. Let $\Delta_j := \text{dist}(\boldsymbol{P}_{j-1}, \boldsymbol{P}_j)$.

Theorem 2.1 (NORST-miss, $\boldsymbol{\nu}_t = 0$ case)**.** *Consider Algorithm 1. Let* $\alpha := Cf^2 r\log n$, $\boldsymbol{\Lambda} := \mathbb{E}[\boldsymbol{a}_1\boldsymbol{a}_1{}']$, $\lambda^+ := \lambda_{\max}(\boldsymbol{\Lambda})$, $\lambda^- := \lambda_{\min}(\boldsymbol{\Lambda})$, $f := \lambda^+/\lambda^-$.

Pick an $\epsilon \le \min(0.01, 0.03\min_j \text{dist}(\boldsymbol{P}_{j-1}, \boldsymbol{P}_j)^2/f)$. *Let* $K := C\log(1/\epsilon)$. *If*

Algorithm 1 NORST-miss.

Input: $\boldsymbol{y}_t$, $\mathcal{T}_t$ **Output**: $\hat{\boldsymbol{\ell}}_t$, $\hat{\boldsymbol{P}}_{(t)}$ **Parameters:** r, $K = C\log(1/\epsilon)$, $\alpha = Cf^2 r \log n$, $\omega_{evals} = 2\epsilon^2\lambda^+$
$\hat{\boldsymbol{P}}_0 \leftarrow \boldsymbol{0}_{n\times r}$, $j \leftarrow 1$, $k \leftarrow 1$
phase $\leftarrow$ update; $\hat{t}_0 \leftarrow 0$; $\hat{t}_{-1,fin} = 0$
for $t > 0$ **do**
  $\boldsymbol{\Psi} \leftarrow \boldsymbol{I} - \hat{\boldsymbol{P}}_{(t-1)}\hat{\boldsymbol{P}}_{(t-1)}{}'$; $\tilde{\boldsymbol{y}}_t \leftarrow \boldsymbol{\Psi}\boldsymbol{y}_t$;
  $\hat{\boldsymbol{\ell}}_t \leftarrow \boldsymbol{y}_t - \boldsymbol{I}_{\mathcal{T}_t}(\boldsymbol{\Psi}_{\mathcal{T}_t}{}'\boldsymbol{\Psi}_{\mathcal{T}_t})^{-1}\boldsymbol{\Psi}_{\mathcal{T}_t}{}'\tilde{\boldsymbol{y}}_t$.
  if phase = update **then**
    if $t = \hat{t}_j + u\alpha - 1$ for $u = 1,\ 2,\ \cdots$, **then**
      $\hat{\boldsymbol{P}}_{j,k} \leftarrow r$-SVD$[\hat{\boldsymbol{L}}_{t;\alpha}]$, $\hat{\boldsymbol{P}}_{(t)} \leftarrow \hat{\boldsymbol{P}}_{j,k}$, $k \leftarrow k+1$.
    else
      $\hat{\boldsymbol{P}}_{(t)} \leftarrow \hat{\boldsymbol{P}}_{(t-1)}$
    end if
    if $t = \hat{t}_j + K\alpha - 1$ **then**
      $\hat{t}_{j,fin} \leftarrow t$, $\hat{\boldsymbol{P}}_j \leftarrow \hat{\boldsymbol{P}}_{(t)}$
      $k \leftarrow 1$, $j \leftarrow j+1$, phase $\leftarrow$ detect.
    end if
  end if
  if phase = detect and $t = \hat{t}_{j-1,fin} + u\alpha$ **then**
    $\boldsymbol{\Phi} \leftarrow (\boldsymbol{I} - \hat{\boldsymbol{P}}_{j-1}\hat{\boldsymbol{P}}_{j-1}{}')$, $\boldsymbol{B} \leftarrow \boldsymbol{\Phi}\hat{\boldsymbol{L}}_{t,\alpha}$
    if $\lambda_{\max}(\boldsymbol{B}\boldsymbol{B}') \geq \alpha\omega_{evals}$ **then**
      phase $\leftarrow$ update, $\hat{t}_j \leftarrow t$,
    end if
  end if
end for
Smoothing mode: At $t = \hat{t}_j + K\alpha$ **for** $t \in [\hat{t}_{j-1} + K\alpha, \hat{t}_j + K\alpha - 1]$
$\hat{\boldsymbol{P}}^{\text{smooth}}_{(t)} \leftarrow \text{basis}([\hat{\boldsymbol{P}}_{j-1}, \hat{\boldsymbol{P}}_j])$
$\boldsymbol{\Psi} \leftarrow \boldsymbol{I} - \hat{\boldsymbol{P}}^{\text{smooth}}_{(t)}\hat{\boldsymbol{P}}^{\text{smooth}\prime}_{(t)}$
$\hat{\boldsymbol{\ell}}^{\text{smooth}}_t \leftarrow \boldsymbol{y}_t - \boldsymbol{I}_{\mathcal{T}_t}(\boldsymbol{\Psi}_{\mathcal{T}_t}{}'\boldsymbol{\Psi}_{\mathcal{T}_t})^{-1}\boldsymbol{\Psi}_{\mathcal{T}_t}{}'\boldsymbol{y}_t$

1. *left and statistical right incoherence: $\boldsymbol{P}_j$'s are μ-incoherent and $\boldsymbol{a}_t$'s satisfy statistical right incoherence;*
2. *max-miss-frac-col $\leq \frac{c_1}{\mu r}$, max-miss-frac-row$_\alpha \leq \frac{c_2}{f^2}$;*
3. *subspace change: assume $t_{j+1} - t_j > Cr\log n\log(1/\epsilon)$;*
4. *$\boldsymbol{a}_t$'s are independent of the set of missing entries $\mathcal{T}_t$;*

then, with probability (w.p.) at least $1 - 10dn^{-10}$,

1. *subspace change is detected quickly: $t_j \leq \hat{t}_j \leq t_j + 2\alpha$,*
2. *the subspace recovery error satisfies*

$$\text{dist}(\hat{\boldsymbol{P}}_{(t)}, \boldsymbol{P}_{(t)}) \leq \begin{cases} (\epsilon + \Delta_j) & \text{if } t \in \mathcal{J}_1, \\ (0.3)^{k-1}(\epsilon + \Delta_j) & \text{if } t \in \mathcal{J}_k, \\ \epsilon & \text{if } t \in \mathcal{J}_{K+1}. \end{cases}$$

3. *and $\|\hat{\boldsymbol{\ell}}_t - \boldsymbol{\ell}_t\| \leq 1.2(\text{dist}(\hat{\boldsymbol{P}}_{(t)}, \boldsymbol{P}_{(t)}) + \epsilon)\|\boldsymbol{\ell}_t\|$.*

Here $\mathcal{J}_0 := [t_j, \hat{t}_j + \alpha)$, $\mathcal{J}_k := [\hat{t}_j + k\alpha, \hat{t}_j + (k+1)\alpha)$ and $\mathcal{J}_{K+1} := [\hat{t}_j + (K+1)\alpha, t_{j+1})$ and $\Delta_j := \text{dist}(\boldsymbol{P}_{j-1}, \boldsymbol{P}_j)$.

The memory complexity is $\mathcal{O}(nr\log n\log(1/\epsilon))$ and the time complexity is $\mathcal{O}(ndr\log(1/\epsilon))$.

Corollary 2.2 (NORST-miss for MC)**.** *Under the assumptions of Theorem 2.1, NORST-miss-smoothing (line 25 of Algorithm 1) satisfies* $\|\hat{\boldsymbol{\ell}}_t - \boldsymbol{\ell}_t\| \le \epsilon\|\boldsymbol{\ell}_t\|$ *for all* t*. Thus,* $\|\hat{\boldsymbol{L}} - \boldsymbol{L}\|_F \le \epsilon\|\boldsymbol{L}\|_F$*.*

The proof can be found in [4].

2.3 Extensions of basic NORST-miss

Sample-Efficient-NORST-miss

This is a simple modification of NORST-miss that will reduce its sample complexity. The reason that NORST-miss needs many more observed entries is because of the projected LS step which solves for the missing entries vector, $\boldsymbol{z}_t$, after projecting $\boldsymbol{y}_t$ orthogonal to $\hat{\boldsymbol{P}}_{(t-1)}$. This step is computing the pseudo-inverse of $(\boldsymbol{I} - \hat{\boldsymbol{P}}_{(t-1)}\hat{\boldsymbol{P}}_{(t-1)}{}')_{\mathcal{T}_t}$. Our bound on max-miss-frac-col helps ensure that this matrix is well conditioned for any set $\mathcal{T}_t$ of size at most max-miss-frac-col $\cdot\, n$. Notice however that we prove that NORST-miss recovers $\boldsymbol{P}_j$ to ϵ accuracy with a delay of just $(K+2)\alpha = Cr\log n\log(1/\epsilon)$. Once the subspace has been recovered to ϵ accuracy, there is no need to use projected LS to recover $\boldsymbol{z}_t$. One just needs to recover $\boldsymbol{a}_t$ given a nearly perfect subspace estimate and the observed entries. This can be done more easily as follows (borrows PETRELS idea): let $\hat{\boldsymbol{P}}_{(t)} \leftarrow \hat{\boldsymbol{P}}_{(t-1)}$, solve for $\boldsymbol{a}_t$ as $\hat{\boldsymbol{a}}_t := (\boldsymbol{I}_{\Omega_t}{}'\hat{\boldsymbol{P}}_{(t)})^\dagger \boldsymbol{I}_{\Omega_t}{}'\boldsymbol{y}_t$, and set $\hat{\boldsymbol{\ell}}_t \leftarrow \hat{\boldsymbol{P}}_{(t)}\hat{\boldsymbol{a}}_t$. Recall here that $\Omega_t = \mathcal{T}_t^c$. If the set of observed or missing entries was i.i.d. Bernoulli for just the later time instants, this approach will only need $\Omega(r\log r\log^2 n)$ samples at each time t, whp.

NORST-sliding-window

In the basic NORST approach we use a different set of estimates $\hat{\boldsymbol{\ell}}_t$ for each subspace update step. So, for example, the first subspace estimate is computed at $\hat{t}_j + \alpha - 1$ using $\hat{\boldsymbol{L}}_{\hat{t}_j+\alpha-1;\alpha}$; the second is computed at $\hat{t}_j + 2\alpha - 1$ using $\hat{\boldsymbol{L}}_{\hat{t}_j+2\alpha-1;\alpha}$; and so on. This is done primarily to ensure mutual independence of the set of $\boldsymbol{\ell}_t$'s in each interval because this is what makes the proof easier (allows use of matrix Bernstein for example). However, in practice, we can get faster convergence to an ϵ-accurate estimate of $\boldsymbol{P}_j$, by removing this restriction. This approach is of course motivated by the sliding window idea that is ubiquitous in signal processing. For any sliding-window method, there is the window length which we keep as α and the hop-length which we denote by β.

Thus, NORST-sliding-window (β) is Algorithm 1 with the following change: compute $\hat{\boldsymbol{P}}_{j,1}$ using $\hat{\boldsymbol{L}}_{\hat{t}_j+\alpha-1;\alpha}$; compute $\hat{\boldsymbol{P}}_{j,2}$ using $\hat{\boldsymbol{L}}_{\hat{t}_j+\alpha+\beta-1;\alpha}$; compute $\hat{\boldsymbol{P}}_{j,3}$ using $\hat{\boldsymbol{L}}_{\hat{t}_j+\alpha+2\beta-1;\alpha}$; and so on. Clearly $\beta < \alpha$ and $\beta = \alpha$ returns the basic NORST-miss.

NORST-buffer

Another question if we worry only about practical performance is whether re-using the same α data samples $\boldsymbol{y}_t$ in the following way helps: At $t = \hat{t}_j + k\alpha - 1$, the k-th estimate is improved R times as follows. First we obtain $\hat{\boldsymbol{L}}_{t;\alpha} := [\hat{\boldsymbol{\ell}}_{t-\alpha+1}, \hat{\boldsymbol{\ell}}_{t-\alpha+2}, \dots \hat{\boldsymbol{\ell}}_t]$ which are used to compute $\hat{\boldsymbol{P}}_{j,k}$ via r-SVD. Let us denote this by $\hat{\boldsymbol{P}}^0_{j,k}$. Now, we use this estimate to obtain a second, and slightly more refined estimate of the same $\boldsymbol{L}_{t;\alpha}$. We denote these as $\hat{\boldsymbol{L}}^{(1)}_{t;\alpha}$ and use this estimate to get $\hat{\boldsymbol{P}}^{(1)}_{j,k}$. This process is repeated for a total of $R+1$ (reuse) times. We noticed that using $R = 4$ suffices in most synthetic data experiments and for real data, $R = 0$ (which reduces to the basic NORST algorithm) suffices. This variant has the same memory requirement as NORST-original. The time complexity, however, increases by a factor of $R+1$.

Hybrid: Buffer + Sliding Window

We can combine the two previous ideas and come up with hybrid variants that exploit the advantages of both extensions. We provide detailed experimental validation of the different extensions in Sec 4.

3 Robust ST with missing entries

Robust ST with missing entries (RST-miss) is a generalization of robust ST and of ST-miss. In this case, we observe n-dimensional data vectors that satisfy

$$\boldsymbol{y}_t = \mathcal{P}_{\Omega_t}(\boldsymbol{\ell}_t + \boldsymbol{g}_t) + \boldsymbol{\nu}_t, \text{ for } t = 1, 2, \dots, d. \tag{4}$$

where $\boldsymbol{g}_t$'s are the sparse outliers. Let $\boldsymbol{x}_t := \mathcal{P}_{\Omega_t}(\boldsymbol{g}_t)$. We use $\mathcal{T}_{\text{sparse},t}$ to denote the support of $\boldsymbol{x}_t$. This is the part of the outliers that actually corrupt our measurements, thus in the sequel we will only work with $\boldsymbol{x}_t$. With $\boldsymbol{x}_t$ defined as above, $\boldsymbol{y}_t$ can be expressed as

$$\boldsymbol{y}_t = \mathcal{P}_{\Omega_t}(\boldsymbol{\ell}_t) + \boldsymbol{x}_t + \boldsymbol{\nu}_t \tag{5}$$

Observe that, by definition, $\boldsymbol{x}_t$ is supported outside of $\mathcal{T}_t$ and hence $\mathcal{T}_t$ and $\mathcal{T}_{\text{sparse},t}$ are disjoint. Defining the $n \times d$ matrix $\boldsymbol{L} := [\boldsymbol{\ell}_1, \boldsymbol{\ell}_2, \dots \boldsymbol{\ell}_d]$, the above is a robust MC problem.

The main modification needed in this case is outlier support recovery. The original NORST for robust ST [4] used l_1 minimization followed by thresholding based support recovery for this purpose. In this case, the combined sparse vector is $\tilde{\boldsymbol{x}}_t := \boldsymbol{x}_t - \boldsymbol{I}_{\mathcal{T}_t}\boldsymbol{I}_{\mathcal{T}_t}{}'\boldsymbol{\ell}_t$. Support recovery in this case is thus a problem of sparse recovery with partial support knowledge $\mathcal{T}_t$. In this case, we can still use l_1 minimization followed by thresholding. However a better approach is to use noisy modified-CS [28,29] which was introduced to exactly solve this problem. We use the latter. The second modification needed is that, just like in case of robust ST, we need an accurate subspace initialization. To get this, we can use the approach used in robust ST [4]: for the initial $Cr \log n \log(1/\epsilon)$ samples, use the AltProj algorithm for robust PCA (while ignoring the knowledge of $\mathcal{T}_t$ for this initial period). We summarize the approach in Algorithm 2.

We have the following guarantee for NORST-miss-robust. Let max-out-frac-row$_\alpha$ be the maximum fraction of outliers per row of any sub-matrix of $\boldsymbol{X}$ with α consecutive columns; max-out-frac-col be the maximum fraction of outlier per column of $\boldsymbol{X}$. Also define $x_{\min} := \min_t \min_{i \in \mathcal{T}_{\text{sparse},t}} |(\boldsymbol{x}_t)_i|$ to denote the minimum outlier magnitude and let $\Delta := \max_j \Delta_j = \max_j \text{dist}(\boldsymbol{P}_{j-1}, \boldsymbol{P}_j)$.

Corollary 3.3. *Consider Algorithm 2. Assume all conditions of Theorem 2.1 hold and*

1. *max-miss-frac-col* $+ 2 \cdot$ *max-out-frac-col* $\le \frac{c_1}{\mu r}$*; and max-miss-frac-row*$_\alpha$ $+$ *max-out-frac-row*$_\alpha \le \frac{c_2}{f^2}$*;*
2. *subspace change:*
   - *(a)* $t_{j+1} - t_j > (K+2)\alpha$*, and*
   - *(b)* $\Delta \le 0.8$ *and* $C_1\sqrt{r\lambda^+}(\Delta + 2\epsilon) \le x_{\min}$
3. *initialization satisfies* $\text{dist}(\hat{\boldsymbol{P}}_0, \boldsymbol{P}_0) \le 0.25$ *and* $C_1\sqrt{r\lambda^+}\text{dist}(\hat{\boldsymbol{P}}_0, \boldsymbol{P}_0) \le x_{\min}$*;*

then, all guarantees of Theorem 2.1 and Corollary 2.2 hold.

The proof is similar to that given in [4]. Please see the Appendix of [3] for an explanation of the differences. The advantage of using modified-CS to replace l_1 min when recovering the outlier support is that it weakens the required upper bound on max-miss-frac-col by a factor of two. If we used l_1 min, we would need $2 \cdot$ (max-miss-frac-col + max-out-frac-col) to satisfy the upper bound given in the first condition.

Algorithm 2 NORST-miss-robust. Obtain $\hat{\boldsymbol{P}}_0$ by $C \log r$ iterations of AltProj applied to $\boldsymbol{Y}_{[1;t_{\text{train}}]}$ with $t_{\text{train}} = Cr$ and with setting $(\boldsymbol{y}_t)_{\mathcal{T}_t} = 10$ (or any large nonzero value) for all $t = 1, 2, \ldots, t_{\text{train}}$.

Input: $\boldsymbol{y}_t$, $\mathcal{T}_t$ **Output**: $\hat{\boldsymbol{\ell}}_t$, $\hat{\boldsymbol{P}}_{(t)}$
Extra Parameters: $\omega_{supp} \leftarrow x_{\min}/2$, $\xi \leftarrow x_{\min}/15$
$\hat{\boldsymbol{P}}_0 \leftarrow$ obtain as given in the caption;
$j \leftarrow 1$, $k \leftarrow 1$, phase $\leftarrow$ update; $\hat{t}_0 \leftarrow t_{\text{train}}$;
for $t > t_{\text{train}}$ **do**
  $\boldsymbol{\Psi} \leftarrow \boldsymbol{I} - \hat{\boldsymbol{P}}_{(t-1)}\hat{\boldsymbol{P}}_{(t-1)}'$; $\tilde{\boldsymbol{y}}_t \leftarrow \boldsymbol{\Psi}\boldsymbol{y}_t$;
  $\hat{\boldsymbol{x}}_{t,cs} \leftarrow \arg\min_{\boldsymbol{x}} \|(\boldsymbol{x})_{\mathcal{T}_t^c}\|_1$ s.t $\|\tilde{\boldsymbol{y}}_t - \boldsymbol{\Psi}\boldsymbol{x}\| \leq \xi$.
  $\hat{\mathcal{T}}_t \leftarrow \mathcal{T}_t \cup \leftarrow \{i : |\hat{\boldsymbol{x}}_{t,cs}| > \omega_{supp}\}$
  $\hat{\boldsymbol{\ell}}_t \leftarrow \boldsymbol{y}_t - \boldsymbol{I}_{\hat{\mathcal{T}}_t}(\boldsymbol{\Psi}_{\hat{\mathcal{T}}_t}'\boldsymbol{\Psi}_{\hat{\mathcal{T}}_t})^{-1}\boldsymbol{\Psi}_{\hat{\mathcal{T}}_t}'\tilde{\boldsymbol{y}}_t$
  Lines $9 - 27$ of Algorithm 1
end for
Offline (RMC solution): line 25 of Algorithm 1.

4 Experimental Comparisons

We present the results of numerical experiments on synthetic and real data[1]. All the codes for our experiments are available at https://github.com/vdaneshpajooh/NORST-rmc. *In this section, we refer to NORST-miss as just NORST.* All time comparisons are performed on a Desktop Computer with Intel Xeon E3-1200 CPU, and 8GB RAM.

4.1 Parameter Setting for NORST

The algorithm parameters required for NORST are r, K, α and ω_{evals}. For our theory, we assume r, λ^+, λ^-, are known, and we pick a desired accuracy, ϵ. We set $K = C\log(1/\epsilon)$, $\alpha = Cf^2 r \log n$, and $\omega_{evals} = 2\epsilon^2\lambda^-$ with C being a numerical constant more than one. Experimentally, the value of r needs to be set from model knowledge, however, overestimating it by a little does not significantly affect the results. In most of our experiments, we set $\alpha = 2r$ (ideally it should grow as $r \log n$ but since $\log n$ is very small for practical values of n it can be ignored). α should be a larger multiple of r when either the data is quite noisy or when few entries are observed. We set K based on how accurately we would like to estimate the subspace. The parameter ω_{evals} needs to be set as a small fraction of the minimum signal space eigenvalue. In all synthetic data experiments, we set $\omega_{evals} = 0.0008\lambda^-$. Another way to set ω_{evals} is as follows. After $K\alpha$ frames, we can estimate $\hat{\lambda}^-$ as the r-th eigenvalue of $\sum_{\tau=t-\alpha+1}^{t} \hat{\boldsymbol{\ell}}_\tau\hat{\boldsymbol{\ell}}_\tau'/\alpha$ and set $\omega_{evals} = c\hat{\lambda}^-$ as mentioned before. We use the Conjugate Gradient Least Squares (CGLS) method [30] for the LS step with tolerance as 10^{-16}, and maximum iterations as 20.

For the video experiments, we estimated r using training data from a few videos and fixed it as $r = 30$. We let λ^- be the r-th eigenvalue of the training dataset. We used $\omega_{evals} = 1.6 \times 10^{-6}\lambda^- = 0.002$, $\alpha = 2r$ and $K = 3$ for the video data. The reason that we use a smaller fraction of λ^- as ω_{evals} is because videos are only approximately low-rank.

4.2 Fixed Subspace, Noise-free data

We generated the data according to (1) and set $\boldsymbol{\nu}_t = 0$. We assume a fixed subspace i.e. $J = 1$. We generate the subspace basis matrix $\boldsymbol{P} \in \mathbb{R}^{n\times r}$ by ortho-normalizing the columns of a random Gaussian matrix with $n = 1000$ and $r = 30$. The $\boldsymbol{a}_t$'s (for $t = 1, \cdots, d$ and $d = 4000$) are generated independently as $(\boldsymbol{a}_t)_i \overset{\text{i.i.d}}{\sim} \text{unif}[-q_i, q_i]$ where $q_i = \sqrt{f} - \sqrt{f}(i-1)/2r \quad \text{for} \quad i = 1, 2, \cdots, r-1$ and $q_r = 1$. Thus, the condition number of $\boldsymbol{\Lambda}$ is f and we set $f = 100$.

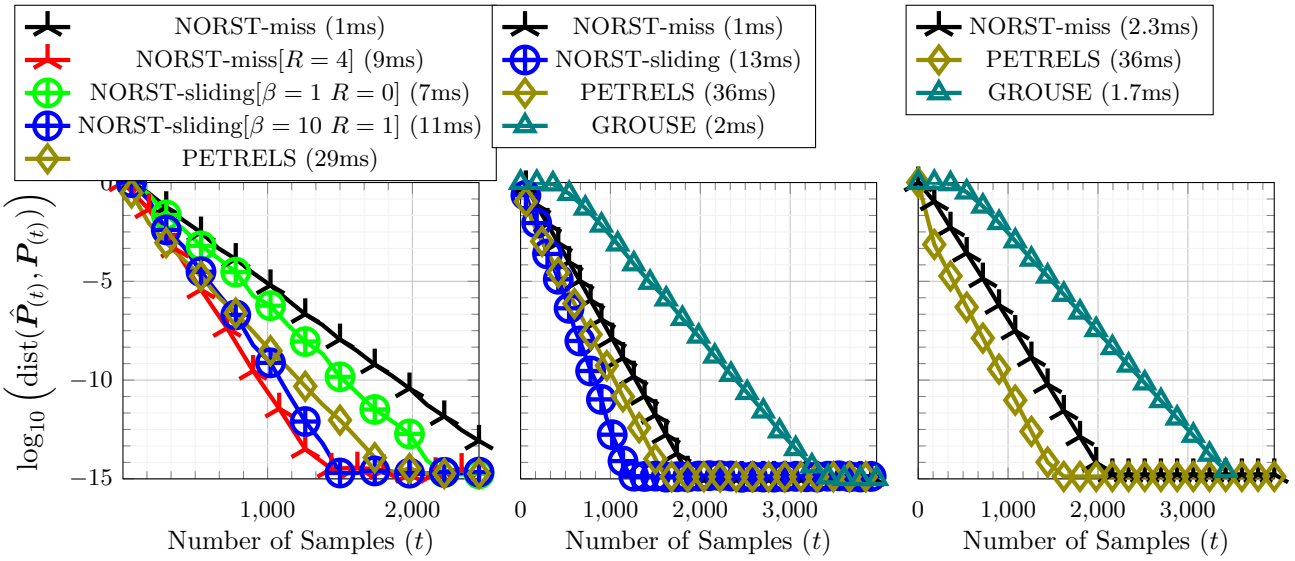
For our first experiment, the observed entries' set was i.i.d. Bernoulli with fraction of observed entries $\rho = 0.7$. We compared all NORST extensions and PETRELS. We set the algorithm parameters for NORST and extensions as mentioned before and used $K = 33$ to see how low the NORST error can

[1]We downloaded the PETRELS' and GROUSE code from the authors' website and all other algorithms from https://github.com/andrewssobral/lrslibrary.

Table 1: (top) Number of samples (frames) required by NORST and its heuristic extensions, and PETRELS to attain $\approx 10^{-16}$ accuracy. The observed entries are drawn from a i.i.d. Bernoulli model with $\rho = 0.7$ fraction of observed entries. Notice that NORST-buffer(4) and NORST-sliding-window ($\beta = 10, R = 1$) converges at the same rate as PETRELS and the time is also comparable. The other variants require more samples to obtain the same error but are faster compared to PETRELS. (bottom) Evaluation of Sample Efficient NORST with $\rho_1 = 0.9$ and $\rho_2 = 0.15$.

Algorithm	NORST	NORST-buffer				NORST-sliding-window and buffer		PETRELS
Parameter R, β		$R = 1$	$R = 2$	$R = 3$	$R = 4$	$\beta = 1$, $R = 0$	$\beta = 10$, $R = 1$	
Time taken (ms)	1.9	10.8	18.6	27.5	34.5	16	35	33
Number of samples	3540	2580	2100	2050	1950	2400	1740	1740

Algorithm	NORST-miss (6)	NORST-samp-eff (1)	PETRELS (15)	GROUSE (2)
Average Error	0.04	0.04	0.02	0.13

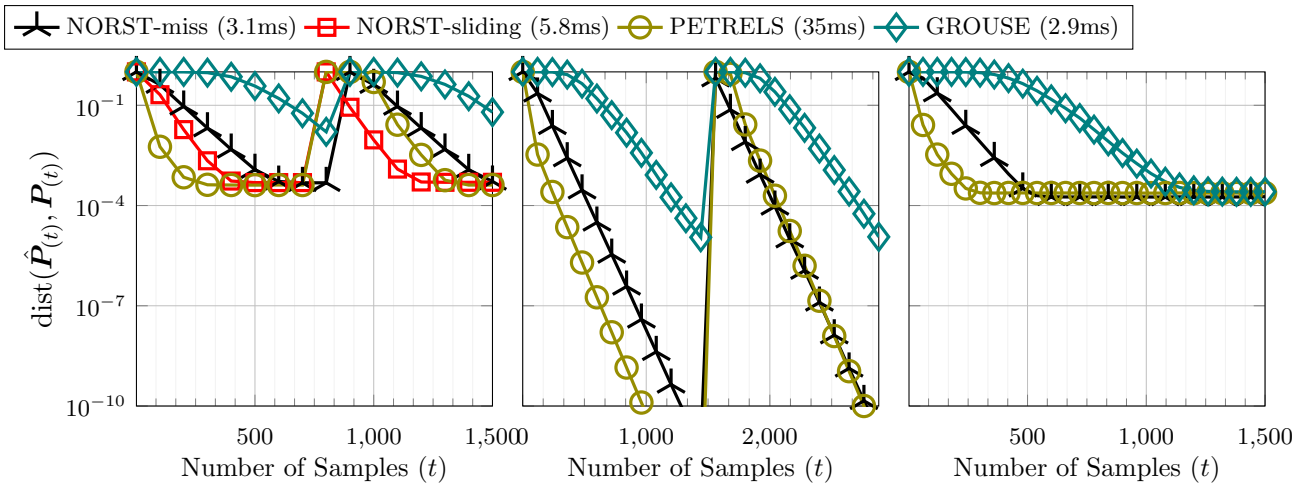


(a) Moving Object Model ($\rho = 0.8$) (b) Bernoulli Model ($\rho = 0.9$) (c) Bernoulli(0.9) and $\boldsymbol{\Lambda}_t$ time-varying

Figure 1: We compare NORST-miss and its extensions with PETRELS and GROUSE. We plot the logarithm of the subspace error between the true subspace $\boldsymbol{P}_{(t)}$ and the algorithm estimates, $\hat{\boldsymbol{P}}_{(t)}$ on the y-axis and the number of samples (t) on the x-axis. As can be seen, in the first two cases, NORST-buffer and NORST-sliding have the best performance (while also being faster than PETRELS), followed by PETRELS, basic NORST and then GROUSE. PETRELS performs best in the scenario of time varying $\boldsymbol{\Lambda}_t$. The computational time per sample (in milliseconds) for each algorithm is mentioned in the legend.

go. For PETRELS we set `max_cycles` $= 1$, forgetting parameter $\lambda = 0.98$ as specified in the paper. We display the results in Table 1 (top). Notice that NORST-miss and its extensions are significantly faster than PETRELS. Also, the $\beta = 10, R = 1$ is the best of all the NORST extensions and is as good as PETRELS.

In our second set of experiments, we compared NORST (and a few extensions) with PETRELS and GROUSE for three settings of missing data. For GROUSE, we set maximum cycles as 1 as specified in the documentation and set the step size, $\eta = 0.1$ and the step-size is udpated according to [14]. The first was for missing generated from the Moving Object model [22, Model 6.19] with $s = 200$, and $b_0 = 0.05$. This translates to $\rho = 0.8$ fraction of observed entries. This is an example of a deterministic model on missing entries. We plot the subspace recovery error versus time for this case in Fig. 1(a) As can be seen, NORST-buffer (R=4) and NORST-sliding-window ($\beta = 10, R = 4$) have the best performance, followed by PETRELS, basic NORST, and then GROUSE. PETRELS is the slowest in terms of time taken. In Fig. 1(b), we plot the results for Bernoulli observed entries' set



(a) Piecewise Constant (Noisy) (b) Piecewise Constant (Noise-Free) (c) Subspace change at each time

Figure 2: Subspace error versus time plot for changing subspaces. We plot the dist$(\hat{\boldsymbol{P}}_{(t)}, \boldsymbol{P}_{(t)})$ on the y-axis and the number of samples (t) on the x-axis. The entries are observed under Bernoulli model with $\rho = 0.9$. The computational time taken per sample (in milliseconds) is provided in the legend parenthesis. **(a) Piecewise constant subspace change and noise-sensitivity:** Observe that after the first subspace change, NORST-sliding adapts to subspace change using the least number of samples and is also $\approx$ 6x faster than PETRELS whereas GROUSE requires more samples than our approach and thus is unable to converge to the noise-level ($\approx 10^{-4}$); **(b) Piecewise Constant and noise-free:** All algorithms perform significantly better since the data is noise-free. We clip the y-axis at 10^{-10} for the sake of presentation but NORST and PETRELS attain a recovery error of 10^{-14}. **(c) Subspace changes a little at each time:** All algorithms are able to track the span of top-r singular vectors of $[\boldsymbol{P}_{(t-\alpha+1)}, \cdots, \boldsymbol{P}_{(t)}]$ to an accuracy of 10^{-4}. As explained, the subspace change at each time can be thought of as noise. GROUSE needs almost 2x number of samples to obtain the same accuracy as NORST while PETRELS is approximately 10x slower than both NORST and GROUSE.

with $\rho = 0.9$. Here again, NORST-sliding has the best performance. Basic NORST is only slightly worse than PETRELS. As can be seen from the time taken (displayed in the legend), NORST and its extensions are much faster than PETRELS.

In Fig. 1(c), as suggested by an anonymous reviewer, we evaluate the same case but with the covariance matrix of $\boldsymbol{\ell}_t$ being time-varying. We generate the $\boldsymbol{a}_t$'s as described earlier but with $q_{t,i} = \sqrt{f} - \sqrt{f}(i-1)/2r - \lambda^-/2$ for $t = 2, 4, 6, \cdots$ and $q_{t,i} = \sqrt{f} - \sqrt{f}(i-1)/2r + \lambda^-/2$ for $t = 1, 3, 5, \cdots$ and $q_{t,r} = 1$. As can be seen all approaches still work in this case. PETRELS converges with the fewest samples but is almost $18x$ slower.

4.3 Changing Subspaces, Noisy and Noise-free Measurements

Piecewise constant subspace change, noisy and noise-free: We generate the changing subspaces using $\boldsymbol{P}_j = e^{\gamma_j \boldsymbol{B}_j} \boldsymbol{P}_{j-1}$ as done in [6] where γ_j controls the amount subspace change and $\boldsymbol{B}_j$'s are skew-symmetric matrices. We used the following parameters: $n = 1000$, $d = 10000$, $J = 6$, and the subspace changes after every 800 frames. The other parameters are $r = 30$, $\gamma_j = 100$ and the matrices $\boldsymbol{B}_i$ are generated as $\boldsymbol{B}_i = (\tilde{\boldsymbol{B}}_i - \tilde{\boldsymbol{B}}_i{}')$ where the entries of $\tilde{\boldsymbol{B}}_i$ are generated independently from a standard normal distribution and $\boldsymbol{a}_t$'s are generated as in the fixed subspace case. For the missing entries supports, we consider the Bernoulli Model with $\rho = 0.9$. The noise $\boldsymbol{\nu}_t$'s are generated as i.i.d. Gaussian r.v.'s with $\sqrt{\lambda_v^+} = 3 \times 10^{-3}\sqrt{\lambda^-}$. The results are summarized in Fig. 2(a). For NORST we set $\alpha = 100$ and $K = 7$. We observe that all algorithms except GROUSE are able to attain final accuracy approximately equal to the noise-level, 10^{-3} within a short delay of the subspace change. We also observe that NORST-sliding-window adapts to subspace change using the fewest samples possible. Moreoever it is much faster than PETRELS.

In Fig. 2(b), we plot results for the above setting but with noise $\nu_t = 0$. In this case, the underlying subspace is recovered to accuracy lower than 10^{-12} by NORST and PETRELS but GROUSE only tracks to error 10^{-7}.

Table 2: Comparison of $\|\boldsymbol{L} - \hat{\boldsymbol{L}}\|_F / \|\boldsymbol{L}\|_F$ for MC. We report the time taken per sample in milliseconds in parenthesis. Thus the table format is Error (computational time per sample). The first three rows are for the fixed subspace model. The fourth row contains results for time-varying subspace and with noise of standard deviation $0.003\sqrt{\lambda^-}$ added. The last row reports Background Video Recovery results (for the curtain video shown in Fig. 3 when missing entries are Bernoulli with $\rho = 0.9$.

Subspace model	NORST-smoothing	nuclear norm min (NNM) solvers		projected-GD
		IALM	SVT	
Fixed (Bern, $\rho = 0.9$)	1.26×10^{-15} (10)	1.43×10^{-12} (150)	7.32×10^{-7} (164)	0.98 (1)
Fixed (Bern, $\rho = 0.3$)	3.5×10^{-6} (11)	5.89×10^{-13} (72)	–	0.98 (9)
Noisy, Changing (Bern, $\rho = 0.9$)	3.1×10^{-4} (3.5)	3.47×10^{-4} (717)	2.7×10^{-3} (256)	0.97 (2)
Video Data	0.0074 (83.7)	0.0891 (57.5)	0.0034 (6177)	–

Subspace change at each time: Here we generate the data using the approach of [12]: $\boldsymbol{P}_{(1)}$ is generated by ortho-normalizing the columns of a i.i.d. Gaussian matrix and let $\boldsymbol{P}_{(t)} = e^{\gamma \boldsymbol{B}} \boldsymbol{P}_{(t-1)}$. We set $\gamma = 10^{-7}$. No extra noise $\boldsymbol{\nu}_t$ was added, i.e., $\boldsymbol{\nu}_t = 0$, in this experiment. We plot $\text{dist}(\hat{\boldsymbol{P}}_{(t)}, \boldsymbol{P}_{(t)})$ in Fig. 2(c). Notice that, even without added noise $\boldsymbol{\nu}_t$, all algorithms are only able to track the subspaces to accuracy at most 10^{-3} in this case. The reason is, as explained earlier in Sec. **??**, subspace change at each time can be interpreted as r dimensional piecewise constant subspace change plus noise.

4.4 Matrix Completion

In Table 2, we compare NORST-smoothing with existing MC solutions (for which code is available). This table displays the Monte-Carlo mean of the normalized Frobenius norm error along with time-taken per column displayed in parentheses. We compare two solvers for nuclear norm min (NNM) – (i) Singular Value Thresholding (SVT) with maximum iterations as 500, tolerance as 10^{-8}, $\delta = 1.2/\rho$, and $\tau = 5\sqrt{nd}$ and (ii) Inexact Augmented Lagrangian Multiplier (IALM) [31] with maximum iterations 500 and tolerance 10^{-16}. We also evaluate the projected Gradient Descent (projected-GD) algorithm of [25], this is a non-convex and hence fast approach, with the best sample complexity among non-convex approaches. This seems to be the only provable non-convex MC approach for which code is available. NORST-smoothing used $K = 33$ and $\alpha = 2r$.

The matrix $\boldsymbol{L}$ was generated as described in Sec. 4.2 for the "fixed" subspace rows and as in Sec. 4.3 (piecewise constant subspace change) for the "Noisy, Changing" subspace row. The observed entries set followed the Bernoulli model with different values of ρ in the different rows. The table demonstrates our discussion from Sec. 2.2. (1) In all cases, NORST-smoothing is much faster than both the solvers for convex MC (NNM), but is slower than the best non-convex MC approach (projected-GD). (2) NORST-smoothing is always better than projected-GD (implemented using default code, it is not easy to change the code parameters). It is nearly as good as IALM (one of the two solvers for NNM) when ρ is large, but is worse than IALM when ρ is small.

4.5 Real Video Data

Here we consider the task of Background Recovery for missing data. We use the `Meeting Room` video which is a benchmark dataset in Background Recovery. It contains 1755 images of size 64x80 in which a curtain is moving in the wind. Subsequently, there are 1209 frames in which a person walks into the room, writes on a blackboard, and exits the room. The first 1755 frames are used for ST-miss while the subsequent frames are used for RST-miss (since we can model the person as a sparse outlier [**?**]).

We generate the set of observed entries using the Bernoulli model with $\rho = 0.9$. In all experiments, we use the estimate of rank as $r = 30$. The parameters of NORST-miss are $\alpha = 60$, $K = 3$, and $\omega_{evals} = 2 \times 10^{-3}$. We noticed that PETRELS failed to retrieve the background with default parameters so we increased `max_cycles`= 10 and refer to this as PETRELS(10) in the sequel. Furthermore, we also ensured that the input data matrix has more columns than rows by transposing the matrix when necessary. All other algorithms are implemented as done in the previous experiments. We observed that NORST-miss and SVT provide a good estimate of the background and NORST is $\approx$ 150x faster.

Figure 3: Background Recovery under Moving Object Model missing entries ($\rho = 0.98$). We show the original, observed, and recovered frames at $t = \{980, 1000, 1020\}$. NORST and SVT are the only algorithms that work although NORST is almost 3 orders of magnitude faster than SVT. PETRELS(10) exhibits artifacts, while IALM and GROUSE do not capture the movements in the curtain. The time taken per sample for each algorithm is shown in parenthesis.

The relative Frobenius error is provided in the last row of Table. 2. Notice that, in this case, SVT outperforms IALM and NORST, but NORST is the fastest one. These results are averaged over 10 independent trials.

Moving Object Missing Entries: In our second video experiment, we generated the set of missing entries using the moving object model with $\rho = 0.98$. All algorithms are implemented as in the previous experiment. Interestingly, even though we observe 98% of the entries, the performance of all algorithms degrade compared to the Bern(0.9). This is possibly because the support sets are highly correlated over time and thus the assumptions of other algorithms break down. The results are shown in Fig. 3. Observe that NORST-miss and SVT provide the best visual comparison and NORST-miss is faster than SVT by ≈ 400x. PETRELS(10) contains significant artifacts in the recovered background and IALM provides a *static* output in which the movements of the curtain are not discernible.

4.6 RST-miss and RMC

In this experiment, we consider the RST-miss problem, i.e., we generate data according to (4). We generate the low rank matrix, $\boldsymbol{L}$, as done in experiment 1 (single subspace). We generate the sparse matrix, $\boldsymbol{X}$ as follows: we use the Moving Object Model to generate the support sets such that $s/n = 0.05$ and $b_0 = 0.05$ which translates to $\rho_{\text{sparse}} = 0.05$ *fraction of sparse outliers*. The non-zero magnitudes of $\boldsymbol{X}$ are generated uniformly at random between $[x_{\min}, x_{\max}]$ with $x_{\min} = 10$ and $x_{\max} = 25$. We generated the support of observed entries using Bernoulli Model with probability $\rho_{\text{obs}} = 0.9$.

For initialization step of NORST-miss-robust (Algorithm 2), for the first $t_{\text{train}} = 400$ data samples, we set $(\boldsymbol{y}_t)_i = 10$ for all $i \in \mathcal{T}_t$. We do this to allow us to use AltProj [**?**], which is an RPCA solution, for obtaining the initial subspace estimate. The parameters for this step are set as 500 maximum iterations of AltProj, and tolerance 10^{-3}. The other algorithm parameters for NORST-miss-robust are $\alpha = 60$, $K = 33$, $\omega_{evals} = 7.8 \times 10^{-4}$, $\xi = x_{\min}/15$, and $\omega_{supp} = \boldsymbol{x}_{\min}/2 = 5$. We compare[2] GRASTA-RMC [32] and projected-GD [25]. For GRASTA-RMC we used the tolerance 10^{-8}, and `max_cycles`= 1. For projected-GD, we use the default tolerance 10^{-1} and max. iterations 70. The results are given in Table. 3. Observe that NORST-miss-robust obtains the best estimate among the RMC algorithms.

Real video data: In this experiment, we consider Background recovery applied on the second part of the dataset (last 1209 frames). In addition to the person who enters the room and writes on the board (sparse component), we generate missing entries from the Bernoulli model with $\rho = 0.9$.

[2]we do not compare it with NNM based methods for which code is not available online

Table 3: Comparing recovery error for Robust MC methods. Missing entries were Bernoulli with $\rho = 0.9$, and the outliers were sparse Moving Objects with $\rho_{\text{sparse}} = 0.95$. The time taken per sample is shown in parentheses.

NORST-miss-rob	GRASTA-RMC	projected-GD
0.0832 (3)	0.1431 (2.9)	0.5699 (2)

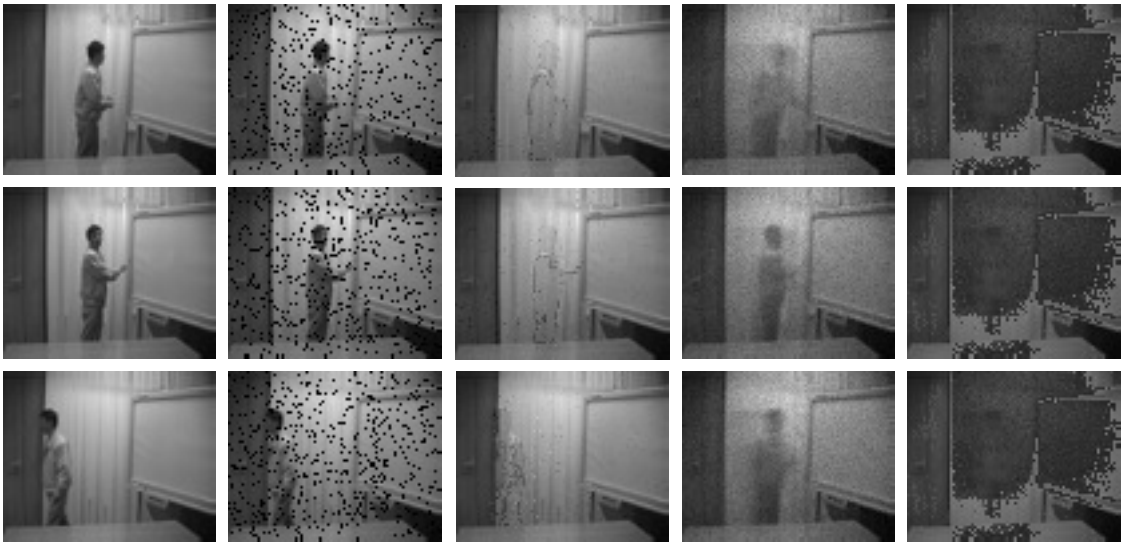


Figure 4: Background Recovery with foreground layer, and Bernoulli missing entries ($\rho = 0.9$). We show the original, observed and recovered frames at $t = 1755 + \{1059, 1078, 1157\}$. NORST-miss-rob exhibits artifacts, but is able to capture most of the background information, whereas, GRASTA-RMC and projected-GD fail to obtain meaningful estimates. The time taken per sample for each algorithm is shown in parenthesis.

We initialize using AltProj with tolerance 10^{-2} and 100 iterations. We set $\omega_{supp,t} = 0.9\|\boldsymbol{y}_t\|/\sqrt{n}$ using the approach of [4]. The comparison results are provided in Fig. 4. Notice that both GRASTA-RMC and projected-GD fail to accurately recover the background. Although NORST-miss-robust exhibits certain artifacts around the edges of the sparse object, it is able to capture most of the information in the background.

5 Conclusions

This work studied the related problems of subspace tracking in missing data (ST-miss) and its robust version. We show that the proposed approaches are provably accurate under simple assumptions on only the observed data (in case of ST-miss), and on the observed data and initialization (in case of robust ST-miss). Thus, in both cases, the required assumptions are only on the algorithm inputs, making both results *complete guarantees*. Moreover, the guarantees show that the algorithms need near-optimal memory; are as fast as vanilla PCA; and can detect and track subspace changes quickly.

We also show that NORST-miss and NORST-miss-robust have good experimental performance as long as the fraction of missing entries is not too large.

While these approaches have near-optimal memory complexity, they are not streaming. This is because they use SVD and hence need multiple passes over short batches of stored data. A key open question is whether a fully streaming provably correct solution can be developed without assuming the i.i.d. Bernoulli model on the set of missing entries? Also, another important open question is: can the required number of observed entries be reduced (the limiting bound here is the bound on missing fractions per column)?

6 Acknowledgement

This report is based on [3]-which has been published in *IEEE Transactions on Signal Processing*- and [1, 2]. The main algorithm (NORST) and its guarantees were introduced by *P. Narayanamurthy* and *N. Vaswani*. I contributed to this work by implementing the experimental comparisons and finding ways to improve performance of the NORST algorithm in terms of speed (modifying the `MATLAB` code) and accuracy (heuristic extensions of basic NORST).

References

[1] P. Narayanamurthy, V. Daneshpajooh, and N. Vaswani, "Provable memory-efficient online robust matrix completion," in *IEEE Int. Conf. Acoust., Speech and Sig. Proc. (ICASSP). IEEE*, 2019, pp. 7918–7922.

[2] P. Narayanamurthy, V. Daneshpajooh, and N. Vaswani, "Provable subspace tracking with missing entries," *ISIT*, 2019.

[3] P. Narayanamurthy, V. Daneshpajooh, and N. Vaswani, "Provable Subspace Tracking from Missing Data and Matrix Completion," *IEEE Trans. Sig. Proc.*, 2019.

[4] P. Narayanamurthy and N. Vaswani, "Nearly optimal robust subspace tracking," in *International Conference on Machine Learning*, 2018, pp. 3701–3709.

[5] P. Comon and G. H. Golub, "Tracking a few extreme singular values and vectors in signal processing," *Proceedings of the IEEE*, vol. 78, no. 8, pp. 1327–1343, 1990.

[6] L. Balzano, Y. Chi, and Y. M. Lu, "Streaming pca and subspace tracking: The missing data case," *Proceedings of IEEE*, 2018.

[7] A. Gonen, D. Rosenbaum, Y. C. Eldar, and S. Shalev-Shwartz, "Subspace learning with partial information," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1821–1841, 2016.

[8] N. Vaswani and P. Narayanamurthy, "Static and dynamic robust pca and matrix completion: A review," *Proceedings of the IEEE*, vol. 106, no. 8, pp. 1359–1379, 2018.

[9] B. Yang, "Projection approximation subspace tracking," *IEEE Trans. Sig. Proc.*, pp. 95–107, 1995.

[10] B. Yang, "Asymptotic convergence analysis of the projection approximation subspace tracking algorithms," *Signal Processing*, vol. 50, pp. 123–136, 1996.

[11] Y. Chi, Y. C. Eldar, and R. Calderbank, "Petrels: Parallel subspace estimation and tracking by recursive least squares from partial observations," *IEEE Trans. Sig. Proc.*, December 2013.

[12] L. Balzano, B. Recht, and R. Nowak, "Online Identification and Tracking of Subspaces from Highly Incomplete Information," in *Allerton Conf. Comm., Control, Comput.*, 2010.

[13] L. Balzano and S. Wright, "Local convergence of an algorithm for subspace identification from partial data," *Found. Comput. Math.*, vol. 15, no. 5, 2015.

[14] D. Zhang and L. Balzano, "Global convergence of a grassmannian gradient descent algorithm for subspace estimation," in *AISTATS*, 2016.

[15] G. Ongie, D. Hong, D. Zhang, and L. Balzano, "Enhanced online subspace estimation via adaptive sensing," in *Asilomar*, 2018.

[16] C. Wang, Y. C. Eldar, and Y. M. Lu, "Subspace estimation from incomplete observations: A high-dimensional analysis," *JSTSP*, 2018.

[17] I. Mitliagkas, C. Caramanis, and P. Jain, "Streaming pca with many missing entries," *Preprint*, 2014.

[18] A. Gonen, D. Rosenbaum, Y. C. Eldar, and S. Shalev-Shwartz, "Subspace learning with partial information," *Journal of Machine Learning Research*, vol. 17, no. 52, pp. 1–21, 2016.

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

[20] C. Qiu, N. Vaswani, B. Lois, and L. Hogben, "Recursive robust pca or recursive sparse recovery in large but structured noise," *IEEE Trans. Info. Th.*, pp. 5007–5039, August 2014.

[21] J. Zhan, B. Lois, H. Guo, and N. Vaswani, "Online (and Offline) Robust PCA: Novel Algorithms and Performance Guarantees," in *Intnl. Conf. Artif. Intell. Stat.* (AISTATS), 2016.

[22] P. Narayanamurthy and N. Vaswani, "Provable dynamic robust pca or robust subspace tracking," *EEE Transactions on Information Theory*, vol. 65, no. 3, pp. 1547–1577, 2019.

[23] E. J. Candes and B. Recht, "Exact matrix completion via convex optimization," *Found. of Comput. Math*, , no. 9, pp. 717–772, 2008.

[24] P. Netrapalli, P. Jain, and S. Sanghavi, "Low-rank matrix completion using alternating minimization," in *STOC*, 2013.

[25] Y. Cherapanamjeri, K. Gupta, and P. Jain, "Nearly-optimal robust matrix completion," *ICML*, 2016.

[26] N. Vaswani and P. Narayanamurthy, "Pca in sparse data-dependent noise," in *ISIT*, 2018, pp. 641–645.

[27] E. Candes, "The restricted isometry property and its implications for compressed sensing," *C. R. Math. Acad. Sci. Paris Serie I*, 2008.

[28] N. Vaswani and W. Lu, "Modified-CS: Modifying compressive sensing for problems with partially known support," *IEEE Trans. Signal Processing*, September 2010.

[29] J. Zhan and N. Vaswani, "Time invariant error bounds for modified-CS based sparse signal sequence recovery," *IEEE Trans. Info. Th.*, vol. 61, no. 3, pp. 1389–1409, 2015.

[30] C. C. Paige and M. A. Saunders, "Lsqr: An algorithm for sparse linear equations and sparse least squares," *ACM Transactions on Mathematical Software (TOMS)*, vol. 8, no. 1, pp. 43–71, 1982.

[31] Z. Lin, M. Chen, and Y. Ma, "The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices," *arXiv* preprint arXiv:1009.5055, 2010.

[32] J. He, L. Balzano, and A. Szlam, "Incremental gradient on the grassmannian for online foreground and background separation in subsampled video," in *IEEE Conf. on Comp. Vis. Pat. Rec. (CVPR)*, 2012.