PCA and Robust PCA for Modern Datasets

Namrata Vaswani

Dept. of Electrical and Computer Engineering
Iowa State University
Web: `http://www.ece.iastate.edu/~namrata`

Introduction and Course Info

Course Plan I

Overall Plan:

- Math background: random matrix theory results
- PCA: basic idea, PCA for big data key points, PCA in non-isotropic and data-dependent noise
- Robust PCA and Dynamic Robust PCA (Robust Subspace Tracking)
- If time permits: Brief review of Low-rank Matrix Recovery

Above was the plan for a longer course, for a tutorial, we will change the order to Introduction, Robust and Dynamic Robust PCA, then PCA. Will switch over to Math preliminaries where needed.

Roughly 2 hours for RPCA, 1 hour for PCA, two 15-min breaks

Reading Material I

Reading Material:

1. Review the following probability and linear algebra notes ahead of time if you'd like
   - `http://www.ece.iastate.edu/~namrata/EE527_Spring12/322_recap.pdf` (very basic – should know already)
   - `http://www.ece.iastate.edu/~namrata/EE527_Spring14/Probability_recap_3.pdf`
   - `http://www.ece.iastate.edu/~namrata/EE527_Spring14/linearAlgebraNotes.pdf`
2. `https://arxiv.org/abs/1011.3027` R. Vershynin, "Introduction to the non-asymptotic analysis of random matrices" , also Davis-Kahan $\sin\theta$ theorem from 1970 paper by Davis and Kahan, "The Rotation of Eigenvectors by a Perturbation"
3. `https://arxiv.org/abs/1803.00651` Static and Dynamic Robust PCA and Matrix Completion: A Review, Proceedings of the IEEE, August 2018

Reading Material II

4. https://arxiv.org/abs/1712.06061 Nearly Optimal Robust Subspace Tracking, ICML 2018
5. https://arxiv.org/abs/1709.06255 (Finite Sample Guarantees for PCA in Non-Isotropic and Data-Dependent Noise), Allerton 2017

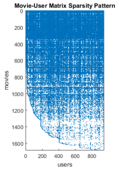
My Research: I am looking for graduate student(s)

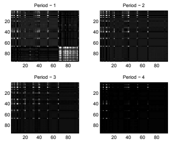
Dynamic Structured (Big) Data Recovery

1. Dynamic Compressive Sensing [KF-CS, ICIP'08], [Modified-CS, T-SP'10, T-IT'15], ...
   - clean data sequence is sparse w/ time-varying supports, measurements: undersampled linear projections of each clean data vector
   - our main message: use structure dynamics (slow support change) to reduce sample complexity without increasing algorithm complexity
2. Dynamic Robust PCA [ReProCS, Qiu et al., Allerton'10, T-IT'14], [AISTATS'16]
   - clean data sequence lies in a fixed or changing low-dimensional subspace, measurements: outlier-corrupted clean data vectors
   - our main message: use structure dynamics (slow subspace change) to improve outlier tolerance *and* reduce algorithm complexity
3. Low Rank Phase Retrieval - recently started work [LRPR, T-SP'17]
   - clean data sequence is low rank, measurements: magnitude-only linear projections of each column of the clean data matrix
   - useful in astronomy, sub-diffraction imaging, Fourier ptychography,...
   - current algorithm: only uses low rank to reduce sample complexity; ongoing work: exploit dynamics (slow subspace change)

Introduction

- In today's big data age, a lot of data is generated everywhere
  - e.g., tweets, video surveillance camera feeds, Netflix movie ratings' data, social network connectivity patterns across time, etc





video surveillance   Netflix movie ratings' data   Reality Mining dataset

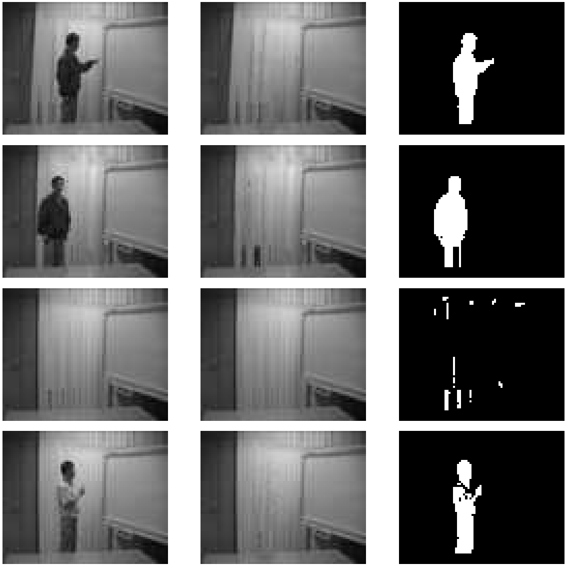
- A lot of it is streaming big data that is not stored or not for too long
  - and often needs to be analyzed on-the-fly.
- First step before processing most big (high-dimensional) datasets is dimension reduction and noise/outlier removal
  - focus of this tutorial
- Clean data usually has structure, e.g., sparsity or low-rank
  - in a long sequence, structure properties are dynamic (change with time)

Two classical approaches for dimension reduction

- Principal Component Analysis (PCA): estimate the low-dimensional subspace that best approximates a given dataset, project to it
  - first step before data classification, image or video retrieval, face recognition, Netflix problem (recommending new movies to users), exploratory data analysis, ...
  - works when data lies close to an (unknown) low-dimensional subspace
- Linear transform to sparsify, followed by zeroing out small entries
  - first step in lossy data compression, e.g., JPEG-2000 uses wavelet transform, and in denoising
  - works when data is sparsifiable in a known transform domain, e.g., wavelet

Many datasets satisfy neither property

- Many datasets are not one or the other, e.g.,
  - clean Netflix users' data lies close to a low-dimensional subspace (users' preferences governed by only a few factors),
    - ⋆ but is corrupted by data from lazy or malicious users (sparse outliers)
  - slow changing videos (e.g., video of moving waters) lie close to a low-dimensional subspace,
    - ⋆ but are often corrupted by foreground moving objects (occlusions)
  - a foreground image sequence is usually sparse,
    - ⋆ but the background image sequence may not be sparse or easily sparsifiable;
  - social media users' connectivity patterns often well-approximated by a low-dimensional tensor,
    - ⋆ but those of anomalous / outlier / suspicious users may not


original background foreground

Many above datasets = sparse + low-rank

- In all previous examples, data is well modeled as a sum of a low-dimensional subspace component and a sparse component,
  - i.e., data vector at time t,

$$y_t = x_t + \ell_t + w_t$$

    with x_t being sparse, ℓ_t lying in a low-dimensional subspace of $\mathbb{R}^n$, and w_t being small modeling error

- If the low-dimensional subspace is fixed or changing slowly,
  - the resulting matrix $L_t = [\ell_1, \ell_2, \ldots \ell_t]$ is low-rank

Mathematical Preliminaries

Cauchy-Schwarz for sums of matrices says the following.

Theorem

$$\left\| \frac{1}{\alpha} \sum_t \boldsymbol{X}_t \boldsymbol{Y}_t{}' \right\|^2 \leq \left\| \frac{1}{\alpha} \sum_t \boldsymbol{X}_t \boldsymbol{X}_t{}' \right\| \left\| \frac{1}{\alpha} \sum_t \boldsymbol{Y}_t \boldsymbol{Y}_t{}' \right\| \tag{1}$$

Theorem (sin θ theorem of Davis-Kahan 1970)

Let $\boldsymbol{D}_0$ be a Hermitian matrix whose span of top r eigenvectors equals span(P). *Let $\boldsymbol{D}$ be the Hermitian matrix with top r eigenvectors $\hat{P}$. The theorem states that*

$$\mathrm{SE}(\hat{P}, P) \leq \frac{\|(\boldsymbol{D} - \boldsymbol{D}_0)P\|_2}{\lambda_r(\boldsymbol{D}_0) - \lambda_{r+1}(\boldsymbol{D})} \leq \frac{\|(\boldsymbol{D} - \boldsymbol{D}_0)P\|_2}{\lambda_r(\boldsymbol{D}_0) - \lambda_{r+1}(\boldsymbol{D}_0) - \lambda_{\max}(\boldsymbol{D} - \boldsymbol{D}_0)} \tag{2}$$

as long as the denominator is positive. The second inequality follows from the first using Weyl's inequality.

Theorem (Hoeffding inequality for scalars)

Let $X_1, \ldots, X_n$ be independent zero-mean random variables. Suppose that $a_i \leq X_i \leq b_i$ almost surely, for all i. Also, let $M := \max_i (b_i - a_i)$. Then, for any $\epsilon > 0$,

$$\Pr\left(|\sum_{i=1}^{n} X_i| > \epsilon \right) \leq 2\exp\left(-\frac{\epsilon^2}{\sum_i (b_i - a_i)^2} \right) \leq 2\exp\left(-\frac{\epsilon^2}{nM^2} \right)$$

Theorem (Bernstein inequality for scalars)

Let $X_1, \ldots, X_n$ be independent zero-mean random variables. Suppose that $|X_i| \leq M$ almost surely, for all i. Then, for any $\epsilon > 0$,

$$\Pr\left(|\sum_{i=1}^{n} X_i| > \epsilon \right) \leq 2\exp\left(-\frac{\frac{1}{2}\epsilon^2}{\sum_j \mathbb{E}\left[X_j^2\right] + \frac{1}{3}M\epsilon} \right).$$

Theorem (matrix Bernstein, Theorem 1.6 of Tropp's User-friendly tail bounds)

For a finite sequence of $d_1 \times d_2$ zero mean independent matrices $\boldsymbol{Z}_k$ with

$$\|\boldsymbol{Z}_k\|_2 \le R, \ \textit{and} \max(\|\sum_k \mathbb{E}[\boldsymbol{Z}_k{}'\boldsymbol{Z}_k]\|_2, \|\sum_k \mathbb{E}[\boldsymbol{Z}_k\boldsymbol{Z}_k{}']\|_2) \le \sigma^2,$$

we have $\Pr(\|\sum_k \boldsymbol{Z}_k\|_2 \ge s) \le (d_1 + d_2) \exp\left(-\frac{s^2/2}{\sigma^2 + Rs/3}\right)$.

Matrix Bernstein conditioned on another r.v. X, says the following.

Theorem

Given an α-length sequence of $n_1 \times n_2$ dimensional random matrices and a r.v. X. Assume the following. For all $X \in \mathcal{C}$, (i) conditioned on X, the matrices $\boldsymbol{Z}_t$ are mutually independent, (i) $\mathbb{P}(\|\boldsymbol{Z}_t\| \leq R|X) = 1$, and (iii) $\max\left\{\left\|\frac{1}{\alpha}\sum_t \mathbb{E}[\boldsymbol{Z}_t{}'\boldsymbol{Z}_t|X]\right\|, \left\|\frac{1}{\alpha}\sum_t \mathbb{E}[\boldsymbol{Z}_t\boldsymbol{Z}_t{}'|X]\right\|\right\} \leq \sigma^2$*. Then, for an* $\epsilon > 0$,

$$\mathbb{P}\left(\left\|\frac{1}{\alpha}\sum_t \boldsymbol{Z}_t\right\| \leq \left\|\frac{1}{\alpha}\sum_t \mathbb{E}[\boldsymbol{Z}_t|X]\right\| + \epsilon \middle| X\right) \geq 1 - (n_1 + n_2)\exp\left(\frac{-\alpha\epsilon^2}{2\left(\sigma^2 + R\epsilon\right.}\right.$$

Vershynin's result for matrices with independent sub-Gaussian rows (Theorem 5.39 of Vershyin's tutorial), conditioned on another r.v. X, says the following.

Theorem

Given an N-length sequence of sub-Gaussian random vectors $\boldsymbol{w}_i$ in $\mathbb{R}^{n_w}$, a r.v X, and a set $\mathcal{C}$. Assume that for all $X \in \mathcal{C}$, (i) $\boldsymbol{w}_i$ are conditionally independent given X; (ii) the sub-Gaussian norm of $\boldsymbol{w}_i$ is bounded by K for all i. Let $\boldsymbol{W} := [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_N]'$. Then for an $0 < \epsilon < 1$ we have

$$\mathbb{P}\left(\left\| \frac{1}{N}\boldsymbol{W}'\boldsymbol{W} - \frac{1}{N}\mathbb{E}[\boldsymbol{W}'\boldsymbol{W}|X] \right\| \leq \epsilon \middle| X \right) \geq 1 - 2\exp\left(n_w \log 9 - \frac{c\epsilon^2 N}{4K^4} \right)$$

for all $X \in \mathcal{C}$.

Notation and Basic PCA

Notation

- $\|.\|$: refers to l_2 norm
- $\boldsymbol{A}'$: transpose of $\boldsymbol{A}$; use other MATLAB notation as well.
- For a set $\mathcal{T}$, $\boldsymbol{I}_{\mathcal{T}}$ is a sub-matrix of $\boldsymbol{I}$ containing columns with indices in $\mathcal{T}$. For a matrix $\boldsymbol{A}$, $\boldsymbol{A}_{\mathcal{T}} := \boldsymbol{A}\boldsymbol{I}_{\mathcal{T}}$
- Basis matrix P: P is a tall $n \times r$ matrix with mutually orthonormal columns, i.e., $P'P = \boldsymbol{I}$
  - used to specify an r-dimensional subspace.
- Subspace error / distance between two subspaces $P, \hat{P}$:

$$\mathrm{SE}(\hat{P}, P) := \|(\boldsymbol{I} - \hat{P}\hat{P}')P\|$$

measures the largest principal angle between the corresponding subspaces.

Principal Component Analysis (PCA)

- Given data points $\boldsymbol{Y} := [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_\alpha]$, find the low-dimensional subspace that best approximates $\boldsymbol{Y}$, i.e., find an $n \times r$ basis matrix, $\hat{P}$, so that

$$\|\boldsymbol{Y} - \hat{P}\hat{P}'\boldsymbol{Y}\|$$

  is minimized.
- When looking for an r-dimensional subspace: r-PCA
- Solution [Eckart-Young]: SVD on $\boldsymbol{Y}$ or EVD on $\boldsymbol{Y}\boldsymbol{Y}'$
  - Let $\boldsymbol{Y} \overset{\mathrm{SVD}}{=} \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}'$; set $\hat{P} = \boldsymbol{U}_{[1:r]} := \boldsymbol{U}\boldsymbol{I}_{[1:r]}$
  - $\hat{P}$ is equivalently the matrix of top r eigenvectors of $\boldsymbol{Y}\boldsymbol{Y}'$.
- Practical issues: Estimate r automatically
  - look for largest eigen-gap or largest normalized eigen-gap
  - all eigenvalues above a pre-set threshold
  - percentage energy
- PCA is a one-line MATLAB command (svd) if data is clean and not very high-dimensional

Robust and Dynamic Robust PCA / Robust Subspace Tracking

Principal Components Analysis (PCA) and Robust PCA

- PCA: find the low-dimensional subspace that best approximates a given dataset
  - first step before many data analytics' tasks - video retrieval, face recognition, Netflix problem, exploratory data analysis, ...
  - PCA for clean data is easy: SVD on data matrix
- Robust PCA: problem of PCA in the presence of outliers; much harder problem; many heuristics exist for trying to solve it
  - best old solution: Robust Subspace Learning (RSL) [de la Torre, Black,'03]
- Recent work [Candes, Wright, Li, Ma, 2009] defined Robust PCA as the problem of decomposing a data matrix Y as

$$Y := L + X$$

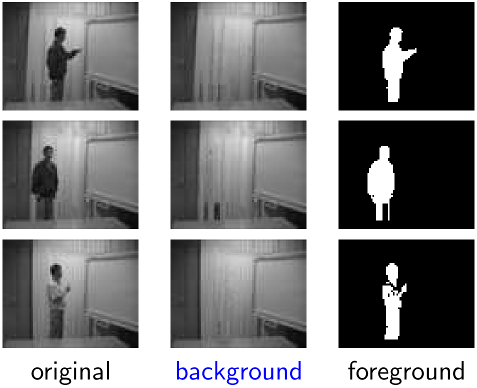
  where L is low rank and X is sparse
  - idea: outliers occur occasionally and usually on only a few data indices; their magnitude is typically large - model as sparse corruptions

Applications [Candes, Wright, Li, Ma, 2009]

- Video Layering (separate video into foreground and background video layers)

$$X = [x_1, x_2 \ldots, x_t, \ldots x_d], \; L = [\ell_1, \ell_2, \ldots \ell_t, \ldots \ell_d]$$

  - ℓ_t: background - usually slow changing,
  - x_t: foreground - sparse (technically x_t: (fg-bg) on fg support)
  - video layering can simplify many downstream computer vision and video analytics tasks, e.g., video retrieval, denoising, background editing, ...



Applications – 2

- Recommendation systems design (Netflix problem) [Candes et al'2009] (robust PCA with missing entries / robust matrix completion)
  - ℓ_t: ratings of movies by user t
  - the matrix L is low-rank: user preferences governed by only a few factors
  - x_t: some users may enter completely incorrect ratings due to laziness or malicious intent or just typos: outliers
  - goal: recover the matrix L in order to recommend movies
- Detecting anomalous connectivity patterns in social networks or in computer networks [Mateos et al.,2011]
  - ℓ_t: vector of n/w link "strengths" at time t when no anomalous behavior
  - x_t: outliers or anomalies on a few links
- Functional MRI based brain activity detection or other dynamic MRI based region-of-interest detection problems [Otazo, Candes, et al. 2014]
  - only a sparse brain region activated in response to stimuli, everything else: very slow changes

Practically useful and provably correct RPCA solutions

- [Candes,Wright,Li,Ma,2009], [Chandrasekharan et al,2009] intro. Principal Component Pursuit:

$$\min_{\tilde{X},\tilde{L}} \|\tilde{L}\|_* + \lambda \|\tilde{X}\|_1 \text{ s.t. } Y = \tilde{X} + \tilde{L}$$

  - first soln that “works” for real videos and has a provable guarantee
- Improved guarantee for PCP by [Hsu et al,2011]
- ReProCS (Recursive Projected Compressive Sensing): algorithm [Qiu,V., Allerton’10,’11], [Guo,Qiu,V., T-SP’14], & partial guarantee: [Qiu,V.,Lois,Hogben, ISIT’13,T-IT’14] for dynamic/online RPCA
- ORPCA (online solver for PCP) and partial guarantee [Feng, Xu, Yan, NIPS’13]
  - partial guarantee: makes assumptions on intermediate algorithm estimates
- AltProj: provable alt-min for RPCA [Netrapalli et al,NIPS’14]: faster than PCP
- First provably correct ReProCS: [Lois,V., ISIT’15], [Zhan,Lois,Guo,V., AISTATS’16]
- GD and NO-RMC: provable grad. descent for RPCA and robust matrix completion (RMC) [Yi et al., NIPS’16], [Jain et al., COLT’16]: faster than AltProj
  - NO-RMC is fastest but needs $d \approx n$
- **Best provable ReProCS: [Narayanamurthy,V.,ICML 2018]: this talk**

First provably correct solution to R-PCA: PCP

- [Candes et al,2009; Chandrasekharan et al,2009; Hsu et al,2011] introduced and studied a convex opt program called PCP:

$$\min_{\tilde{X},\tilde{L}} \|\tilde{L}\|_* + \lambda \|\tilde{X}\|_1 \text{ s.t. } Y = \tilde{X} + \tilde{L}$$

- If (a) left and right singular vectors of L are dense enough; (b) support of X is generated uniformly at random; (c) rank and sparsity are bounded, then PCP exactly recovers X and L from $Y := X + L$ w.h.p. [Candes et al,2011]
  - [Chandrasekharan et al,2011; Hsu et al,2011] (**??**): similar flavor; replace 'unif rand support' by upper bound on $\#$ of nonzeros in any row of X.
  - first set of guarantees for a practical robust PCA approach
- Implementation: Inexact Augmented Lagrangian Method (IALM)
- Web: `https://github.com/andrewssobral/lrslibrary`

Incoherence assumption I

- Introduced in Candes and Plan Matrix Completion work
- Let $L \overset{\mathrm{SVD}}{=} USV'$ be the reduced SVD of rank r matrix L of size $n_1 \times n_2$. Thus U is $n_1 \times r$.
- Incoherence with parameter μ means that

$$\|\boldsymbol{I}_i{}'\boldsymbol{U}\|^2 \leq \mu \frac{r}{n_1}, \ \|\boldsymbol{I}_i{}'\boldsymbol{V}\|^2 \leq \mu \frac{r}{n_2},$$

  row norm of $\boldsymbol{U}$ is not too much larger than r/n_1 (holds if no entry of U is too large); same for V

- Strong incoherence with parameter μ means that above holds *and*

$$\|UV'\|_{\max} \leq \sqrt{\frac{\mu r}{n_1 n_2}}$$

  inner product between any row of U and any row of V is below the RHS.

Guarantees for PCP I

- Let $r_L := \text{rank}(L)$
- [Candes et al, 2009] (**?**) If
  - incoherence and strong incoherence holds with parameter μ
  - support of X is generated uniformly at random
  - and is of size at most $c_2 n_1 n_2$ (thus max-outlier-frac $\leq c$)
  - and $r_L \leq \frac{c_1}{\mu} \frac{n}{(\log n)^2}$;

  then, PCP with $\lambda = 1/\sqrt{n}$ succeeds (returns $\hat{L} = \boldsymbol{L}$) w.p. at least $1 - cn^{-10}$.
- [Hsu et al (follow-up on Chandrasekharan et al)] (**?**): If
  - incoherence holds;
  - max-outlier-frac $\leq \frac{c}{\mu r_L}$
  - algorithm parameter set

  then, PCP succeeds (returns $\hat{L} = \boldsymbol{L}$) all the time.
- Second result removes uniform random support assumption and strong incoherence but needs much stronger bound on max-outlier-frac and r_L

Guarantees for PCP II

- Note: above results assume the PCP convex program is solved *exactly*. In practice, use an iterative solver, that is not possible. Can only solve it to a given small enough error in finite time.
- Time: for an $n \times \alpha$ matrix with $\alpha < n$, time is $O(n\alpha^2)$ per iteration and it needs $O(1/\epsilon)$ iterations to solve PCP to ϵ tolerance (for exact result see Inexact ALM technical report).

Non-convex RPCA solution: Alternating Minimization [Netrapalli et al,NIPS'14] I

- PCP is a convex opt solution - for an $n \times \alpha$ matrix with $\alpha < n$, time is $O(n\alpha^2)$ per iteration and it needs $O(1/\epsilon)$ iterations
- Alt-Min solution: much faster
  - time taken is $O(n\alpha r_L^2 \log(1/\epsilon))$
- AltProj Algorithm Netrapalli et al, NIPS 2014]: idea for rank-1 matrix L:
  - Initialize
    - ★ set $L^0 = 0$;
    - ★ threshold out large entries from M to get $S^0 = HT_{\beta\sigma_1}(M)$, where $\sigma_1 = \sigma_1(M)$
  - For Iterations t=1 to T do, set $T = c\log(1/\epsilon)$
    - ★ $L^t = \mathcal{P}_1(M - S^t)$: project $M - S^t$ onto space of rank-1 matrices
    - ★ $S^t = HT_{\beta(\sigma_2 + 0.5^t\sigma_1)}(M - L^t)$ where $\sigma_i = \sigma_i(M - S^t)$

    End For
- AltProj Algorithm: idea for rank r case:
  - proceed in r stages and do T iterations for each stage

Non-convex RPCA solution: Alternating Minimization [Netrapalli et al,NIPS'14] II

- at stage k: project onto rank k matrices; remove outliers of magnitude larger than $\beta(\sigma_{k+1} + 0.5^t \sigma_k)$

- Guarantee for AltProj (?): similar to PCP guarantee of Hsu et al. If β set carefully and if
  - incoherence condition holds,
  - max-outlier-frac $< \frac{1}{512\mu^2 r}$,
  - set parameter $\beta = \frac{4\mu^2 r}{\sqrt{n\alpha}}$; also need r

  Then $\|\hat{L} - L\|_F \leq \epsilon$ $\|S - \hat{S}\|_{max} \leq \epsilon/\sqrt{n\alpha}$ and $Supp(\hat{\boldsymbol{S}}) \subseteq Supp(S)$
- Details: `https://arxiv.org/pdf/1410.7660.pdf`

```
1: Input: Matrix $M \in R^{m \times n}$, convergence criterion $\epsilon$, target rank $r$,
   thresholding parameter $\beta$.
2: Set initial threshold $\zeta_0 \leftarrow \beta\sigma_1(M)$.
3: $L^{(0)} = 0, S^{(0)} = HT_{\zeta_0}(M - L^{(0)})$
4: for Stage $k = 1$ to $r$ do
5:     for Iteration $t = 0$ to $T = 10\log\left(n\beta\|M - S^{(0)}\|/\epsilon\right)$ do
6:         Set threshold $\zeta$ as
               $$\zeta \;=\; \beta\left(\sigma_{k+1}(M-S) + \left(\frac{1}{2}\right)^t \sigma_k(M-S)\right)$$
7:         $L^{(t+1)} = P_k(M - S^{(t)})$
8:         $S^{(t+1)} = HT_{\zeta}(M - L^{(t+1)})$
9:     end for
10:    if $\beta\sigma_{k+1}(L) < \frac{\epsilon}{2n}$ then
11:        Return: $\boldsymbol{L}^{(T)}, S^{(T)}$   /* Return rank-k estimate if remaining part
           has small norm */
12:    else
13:        $S^{(0)} = S^{(T)}$             /* Continue to the next stage */
14:    end if
15: end for
16: Return: $L^{(T)}, S^{(T)}$
```

Non-convex RPCA solution 2: Gradient Descent [Caramanis,?'16] I

- Motivation: reduce time complexity to almost as much as that for simple PCA (r-SVD for data with large eigen-gap); while nearly matching outlier support guarantees
- Key Ideas:
  - "Sparsifying Operator" $\mathcal{T}_B\left[()\right]$ zero out just enough entries so that each row and column of B has $b_{\text{max-outlier-frac}}$ or less nonzero entries; here $b_{\text{max-outlier-frac}}$ is the assumed bound on max-outlier-frac
  - Initialize:
    - estimate sparse matrix first: S_{init}
    - r-SVD (need not run to convergence) on $Y - S_{init}$.

Non-convex RPCA solution 2: Gradient Descent [Caramanis,?'16] II

- project left and right singular vectors onto $\mathcal{U}, \mathcal{V}$ respectively:

$$\begin{aligned}\mathcal{U} &:= \left\{A \in \mathbb{R}^{d_1 \times r} \mid \|A\|_{2,\infty} \leq \sqrt{\frac{2\mu r}{d_1}} \|U_0\|_{\text{op}}\right\}, \\ \mathcal{V} &:= \left\{A \in \mathbb{R}^{d_2 \times r} \mid \|A\|_{2,\infty} \leq \sqrt{\frac{2\mu r}{d_2}} \|V_0\|_{\text{op}}\right\}.\end{aligned} \qquad (3)$$

  (if row norm of some row of U_t is larger than bound, rescale all entries of that row to so it equals the bound)

- Algorithm:
  - update sparse matrix by applying "sparsifying" operator to $Y - U_t V_t'$
  - gradient descent for cost function $\mathcal{L}(U, V; S) + 0.125\|U'U - V'V\|_F^2$ where $\mathcal{L}(U, V; S) := \|Y - S - UV'\|_F^2$
  - project left and right singular vectors onto $\mathcal{U}, \mathcal{V}$ respectively:

- Guarantee for RPCA-GD [Caramanis,?'16] If
  - incoherence condition holds,
  - max-outlier-frac $< \max(\frac{1}{\mu\kappa^2 r}, \frac{1}{\mu\sqrt{\kappa r^3}})$,

Non-convex RPCA solution 2: Gradient Descent [Caramanis,?'16] III

- set algorithm parameters: need to know r, $\sigma_1(L)$ (to set the step size), max-outlier-frac (for sparse estimation) and κ (to set total number of iterations $T = O(\kappa \log(1/\epsilon))$

Then $\mathrm{SE}(U^t, U) \leq (1 - \frac{c}{\kappa})^t \sqrt{\kappa} \mu r \sqrt{r} \sqrt{\lambda_1}$ (subspace error at iteration t)

- Time complexity for GD: $O(n\alpha r \cdot \kappa(-\log \epsilon))$

RPCA via GD algorithm (?)

Algorithm 1 Fast RPCA

```
1:  Input: Observed matrix $Y$ with rank $r$ and corruption fraction $\alpha$; parameters $\gamma, \eta$; number of iterations $T$.
2:  $S_{\text{init}} \leftarrow \mathcal{T}_{\alpha}[Y]$
3:  $[L, \Sigma, R] \leftarrow \text{SVD}_r[Y - S_{\text{init}}]$ 1
4:  $U_0 \leftarrow L\Sigma^{1/2}$, $V_0 \leftarrow R\Sigma^{1/2}$. Let $\mathcal{U}, \mathcal{V}$ be defined according to (3).
5:  $U_0 \leftarrow \Pi_{\mathcal{U}}(U_0)$, $V_0 \leftarrow \Pi_{\mathcal{V}}(V_0)$
6:  for $t = 0, 1, \ldots, T-1$ do
7:      $S_t \leftarrow \mathcal{T}_{\gamma\alpha}\left[Y - U_t V_t^{\top}\right]$
8:      $U_{t+1} \leftarrow \Pi_{\mathcal{U}}\left(U_t - \eta \nabla_U \mathcal{L}(U_t, V_t; S_t) - \frac{1}{2}\eta U_t (U_t^{\top} U_t - V_t^{\top} V_t)\right)$
9:      $V_{t+1} \leftarrow \Pi_{\mathcal{V}}\left(V_t - \eta \nabla_V \mathcal{L}(U_t, V_t; S_t) - \frac{1}{2}\eta V_t (V_t^{\top} V_t - U_t^{\top} U_t)\right)$
10: end for
11: Output: $(U_T, V_T)$
```

Performance Guarantees summary: PCP, AltProj, GD

- $n \times d$ data matrix $Y := L + X$: L has rank r_L, X is sparse
- PCP or AltProj recovers L and X with error at most ϵ if
  - incoherence (denseness) holds for left and right singular vectors of $\boldsymbol{L}$
    - ★ (ensures L is not sparse)
  - and max-outlier-frac $\in O(1/r_L)$ (max-outlier-frac: max fraction of nonzeros in any row or col of X)
    - ★ (ensures X is not low rank)
- Storage complexity: $O(nd)$
- Best Time complexity: $O(ndr_L^2 \log(1/\epsilon))$ for AltProj
  - or $O(ndr_L \log(1/\epsilon))$ for GD but it needs max-outlier-frac $\in O(1/r_L^{1.5})$

Dynamic Robust PCA / Robust Subspace Tracking

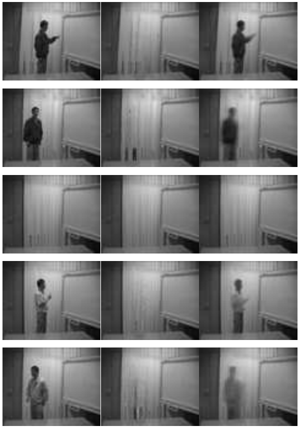
Limitations of existing RPCA solutions

1. Need outliers to be uniformly randomly generated (impractical) or need tight bounds on outlier fractions: need max-outlier-frac $\in O(1/r_L)$
   - these bounds are often violated in practice, e.g.,
     - ★ in video analytics: often have occasionally static or slow moving foreground (fg) objects: large outlier fractions per row
     - ★ can also have large-sized fg objects: large outlier fractions per column
     - ★ in network anomaly detection: anomalous behavior continues on most of the same edges for a period of time after begins
2. Need to store the entire data matrix – memory inefficient
3. Are usually slow

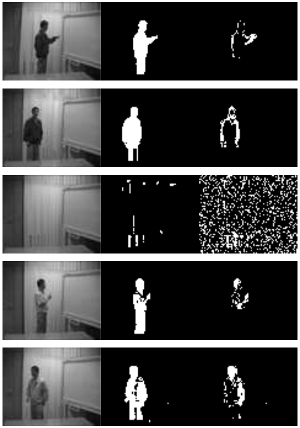
Our work: by exploiting the dynamics (slow subspace change) and a lower bound on most outlier magnitudes, can

- significantly improve outlier tolerance – allow max-outlier-frac-row $\in O(1)$,
- *and* get an online, fast, and memory-efficient algorithm

A Preview



(a) **Background recovery**



(b) **Foreground recovery**

Figure: Slow moving person $\Rightarrow$ sparse matrix X is also low rank $\Rightarrow$ PCP confuses person for background. Proposed method (ReProCS) works; and needs only 16.5ms and 50 frames of memory; PCP needs 44ms and 1200 frames of memory

Problem formulation: some definitions

- P' denotes transpose of matrix P
- P is a "basis matrix" $\Leftrightarrow$ P is a tall matrix with mutually orthonormal columns
  - estimate P $\Leftrightarrow$ estimate span(P) (subspace spanned by col's of P)
- Subspace error / distance: for two r dimensional subspaces $\hat{P}$, P,

$$\mathrm{SE}(\hat{P}, P) = \|(I - \hat{P}\hat{P}')P\|_2$$

measures the largest principal angle between the two subspaces.

Dynamic Robust PCA or Robust Subspace Tracking [Guo,Qiu,Vaswani,T-SP'14][2], [Qiu,Vaswani,Allerton'10,'11][3]

- Given length-n data vectors

$$\boldsymbol{y}_t := \boldsymbol{\ell}_t + \boldsymbol{x}_t, \quad t = 1, 2, \ldots, d$$

  - ℓ_t lies in a fixed or slowly-changing low (r) dimensional subspace of $\mathbb{R}^n$
    - ⋆ $\ell_t = P_t \boldsymbol{a}_t$, P_t: $n \times r$ matrix with $r \ll n$,
    - ⋆ P_t changes only a "little" every so often (slow subspace change)
    - ⋆ columns of P_t are dense vectors (left incoherence)
    - ⋆ $\boldsymbol{a}_t$'s i.i.d. & element-wise bounded (similar to right incoherence)
  - x_t: sparse outlier vector with support set $\mathcal{T}_t$
    - ⋆ the outlier support size is bounded (bound on max-outlier-frac-col)
    - ⋆ outlier supp changes enough over time (bound on max-outlier-frac-row)
- Recursively estimate ℓ_t, span(P_t), starting with initial estimate of span(P_0)
  - initial estimate: either assume outlier-free data available; or few iterations of AltProj on $Y_{init} := [y_1, y_2, \ldots, y_{t_{\mathrm{train}}}]$

[2] H. Guo, C. Qiu, N. Vaswani, An Online Algorithm for Separating Sparse and Low-Dimensional Signal Sequences From Their Sum", IEEE Trans.SP, Aug 2014

[3] C. Qiu and N. Vaswani, Real-time Robust Principal Components' Pursuit, Allerton, 2010

Recursive Projected Compressive Sensing (ReProCS)

[Qiu,Vaswani,Allerton'10,Allerton'11],[Guo,Qiu,Vaswani,T-SP'14]

Recall: $y_t := x_t + \ell_t$, $\ell_t = P_t a_t$, P_t: tall $n \times r$ basis matrix, x_t: sparse

Given $\hat{P}_0$ (obtain using PCP or AltProj on init data). For all later times t, do

1. Projection: compute $\tilde{y}_t := \mathbf{\Phi} y_t$, where $\mathbf{\Phi} := I - \hat{P}_{t-1}\hat{P}_{t-1}{}'$
   - then $\tilde{y}_t = \mathbf{\Phi} x_t + \boldsymbol{\beta}_t$, $\boldsymbol{\beta}_t := \mathbf{\Phi}\ell_t$ is small "noise" because of slow subspace change
2. Noisy Compressive Sensing (CS): l_1 min + support estimate + LS: get $\hat{\boldsymbol{x}}_t$
   - denseness of columns of $P_t \Rightarrow$ sparse x_t recoverable from $\tilde{y}_t$
3. Recover ℓ_t by subtraction: compute $\hat{\ell}_t = y_t - \hat{\boldsymbol{x}}_t$
4. Subspace update: use $\hat{\boldsymbol{\ell}}_t$'s to update $\hat{P}_t$ every α frames
   - next slide...

Subspace update step – simplest [Narayanamurthy, V., ICML 2018]

- Suppose $P_t := P_{t_j}$ for all $t \in [t_j, t_{j+1})$, $j = 0, 1, \ldots, J$
  - (in practice, algorithm also works without this assumption)
- To understand the main idea, suppose that the t_j's are known
  - at $t = t_j + \alpha$: compute first estimate of $P_j := P_{t_j}$

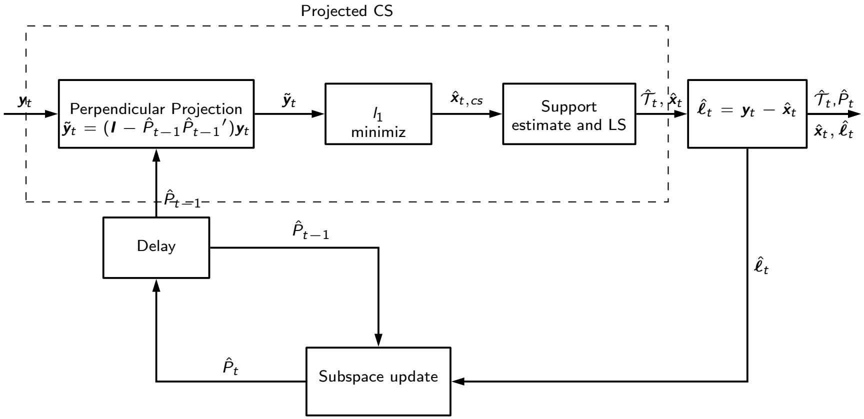
    $$\hat{P}_t \leftarrow \hat{P}_{j,1} \leftarrow \text{top } r \text{ left singular vectors of } [\hat{\boldsymbol{\ell}}_{t_j}, \hat{\boldsymbol{\ell}}_{t_j+1}, \ldots, \hat{\boldsymbol{\ell}}_{t_j+\alpha-1}]$$

  - at $t = t_j + 2\alpha$: compute second estimate of $P_j := P_{t_j}$

    $$\hat{P}_t \leftarrow \hat{P}_{j,2} \leftarrow \text{top } r \text{ left singular vectors of } [\hat{\boldsymbol{\ell}}_{t_j+\alpha}, \hat{\boldsymbol{\ell}}_{t_j+\alpha+1}, \ldots, \hat{\boldsymbol{\ell}}_{t_j+2\alpha-1}]$$

  - repeat K times: get final estimate $\hat{P}_j := \hat{P}_{j,K}$

Recursive Projected Compressive Sensing (ReProCS) framework



Projected CS
$\boldsymbol{y}_t$
Perpendicular Projection
$\tilde{\boldsymbol{y}}_t = (\boldsymbol{I} - \hat{P}_{t-1}\hat{P}_{t-1}{}')\boldsymbol{y}_t$
$\tilde{\boldsymbol{y}}_t$
l_1 minimiz
$\hat{\boldsymbol{x}}_{t,cs}$
Support estimate and LS
$\hat{\mathcal{T}}_t, \hat{\boldsymbol{x}}_t$
$\hat{\boldsymbol{\ell}}_t = \boldsymbol{y}_t - \hat{\boldsymbol{x}}_t$
$\hat{\mathcal{T}}_t, \hat{P}_t$
$\hat{\boldsymbol{x}}_t, \hat{\boldsymbol{\ell}}_t$
$\hat{P}_{t-1}$
Delay
$\hat{P}_{t-1}$
$\hat{\boldsymbol{\ell}}_t$
$\hat{P}_t$
Subspace update

Figure: The ReProCS framework: ℓ_1 min can be replaced by any other method; can also use structured sparse recovery methods

Why ReProCS works - intuition [Qiu,Vaswani,Lois,Hogben,T-IT,2014] [4]

- Slow subspace change $\Rightarrow$ noise $\boldsymbol{\beta}_t := \boldsymbol{\Phi}\ell_t$ seen by CS step small
- Denseness of columns of P_t and slow subspace change $\Rightarrow$ RIP constant of $\boldsymbol{\Phi} := I - \hat{P}_{t-1}\hat{P}_{t-1}{}'$ small. Reason: max-outlier-frac-col $\leq c/\mu r$ and

$$\delta_{2s}(I - PP') = \max_{|T| \leq 2s} \|I_T{}'P\|^2 \leq C\ \mu\ (\text{max-outlier-frac-col})\ r$$

- Above two facts + a guarantee for l_1 min + carefully set support recovery threshold and lower bound on $x_{\min} \Rightarrow x_t$ is accurately recovered; and hence $\ell_t = y_t - x_t$ is accurately recovered
- Most of the work: show accurate subspace recovery $\hat{P}_{(t)} \approx P_{(t)}$
  - standard PCA results not applicable: $\boldsymbol{e}_t := \hat{\boldsymbol{\ell}}_t - \boldsymbol{\ell}_t$ correlated with $\boldsymbol{\ell}_t$
    - ★ reason: $\boldsymbol{e}_t = \boldsymbol{x}_t - \hat{\boldsymbol{x}}_t$ and this depends on $\boldsymbol{\beta}_t := \boldsymbol{\Phi}\ell_t$
  - all existing guarantees for PCA assumed data, noise uncorrelated
    - ★ above problem inspired our work on correlated-PCA [Vaswani,Guo,NIPS'16] , [Vaswani,Narayanamurthy,PCA in Data-Dependent Noise, Allerton'17]

[4] C. Qiu, N. Vaswani, B. Lois and L. Hogben, Recursive Robust PCA or Recursive Sparse Recovery in Large but Structured Noise, IEEE Trans. IT, 2014

Simulation experiments: plot of $\frac{\|\boldsymbol{\ell}_t - \hat{\boldsymbol{\ell}}_t\|_2}{\|\boldsymbol{\ell}_t\|_2}$ v/s time t

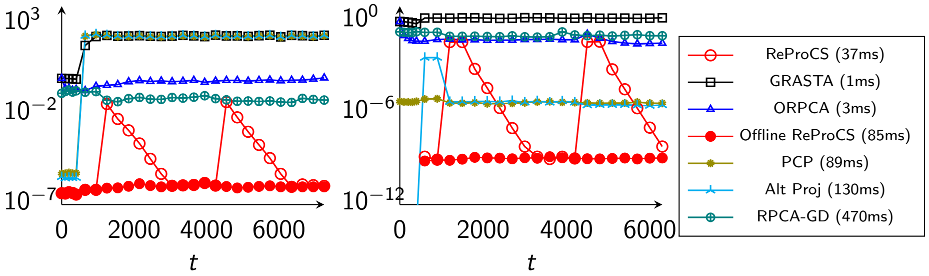


Figure: **Normalized error of ℓ_t.** Left: larger max-outlier-frac-row and non-random outliers (simulated slow moving objects), max-outlier-frac-row $= 0.4$ (after init), Right: random outliers (Bernoulli) and max-outlier-frac-row $= 0.2$ (after init).

- With each SVD step, the error decreases exponentially
  - better estimate of $P_t \Rightarrow$ smaller noise β_t seen by CS step in next α-frame interval $\Rightarrow$ smaller CS step error $\boldsymbol{e}_t := \boldsymbol{x}_t - \hat{\boldsymbol{x}}_t = \hat{\boldsymbol{\ell}}_t - \boldsymbol{\ell}_t \Rightarrow$ smaller perturbation seen at next SVD step $\Rightarrow$ even better next estimate of P_t
  - reason that first estimate of P_t is good and each next estimate is better:
    - ⋆ $\boldsymbol{e}_t$ is sparse with support $\mathcal{T}_t$ that changes enough (bound on max-outlier-frac-row holds)

ReProCS algorithm parameters: practice

- Use the following parameters for all videos:
  - support threshold: $\omega_{supp,t} = \sqrt{\|\boldsymbol{y}_t\|^2/n}$ (RMS of image pixel intensities)
  - noise bound for l_1 min step: $\xi_t = \|\boldsymbol{\Phi}\hat{\boldsymbol{\ell}}_{t-1}\|$
  - eigenvalue threshold to detect subspace change: $\omega_{evals} = 0.01\lambda^-$ (λ^-: r-th eigval. of empirical cov of $\hat{\boldsymbol{L}}_{[1,t_{\mathrm{train}}]}$ with $r = 40$)
  - $\alpha = 60$,
  - $K = 3$,
  - $t_{\mathrm{train}} = 400$, AltProj applied to first t_{train} frames
- Code: `http://www.ece.iastate.edu/~hanguo/PracReProCS.html`
- Show switch-light video

ReProCS algorithm parameters: theory

- Needs knowledge of 4 model parameters: r, λ^-, λ^+, $x_{\min}$ (min nonzero magnitude of any $\boldsymbol{x}_t$)
- Set the algorithm parameters as
  - support threshold: $\omega_{supp} = x_{\min}/2$
  - noise bound for l_1 min step: $\xi = x_{\min}/15$
  - eigenvalue threshold to detect subspace change: $\omega_{evals} = 0.01\lambda^-$
  - $\alpha = Cf^2 r \log n$ where $f = \lambda^+/\lambda^-$
  - $K = C \log(1/\epsilon)$,
  - $t_{\mathrm{train}} = Cf^2 r \log n$, AltProj applied to the first t_{train} frame dataset

ReProCS - Main Result I

We have $\boldsymbol{y}_t = \boldsymbol{\ell}_t + \boldsymbol{x}_t + \boldsymbol{v}_t$, $\boldsymbol{\ell}_t = P_j \boldsymbol{a}_t$ for $t \in [t_j, t_{j+1})$

- Let $\boldsymbol{\Lambda} := \mathbb{E}[\boldsymbol{a}_1 \boldsymbol{a}_1{}']$, $\lambda^+ := \lambda_{\max}(\boldsymbol{\Lambda})$, $\lambda^- := \lambda_{\min}(\boldsymbol{\Lambda})$,
- Let $x_{\min} := \min_t \min_{i \in \mathcal{T}_t} (\boldsymbol{x}_t)_i$ denote minimum outlier magnitude
- Pick ϵ (desired accuracy) s.t., $\epsilon \le \min(0.01, 0.03(\min_j \mathrm{SE}(P_{j-1}, P_j)^2/f)$
- Let $K := C \log(1/\epsilon)$, and $\alpha := Cr \log n$.
- *We assume the condition number $f := \lambda^+/\lambda^-$ is a constant*

See: P. Narayanamurthy and N. Vaswani, "Nearly Optimal Robust Subspace Tracking", ICML 2018.

ReProCS - Main Result II

Theorem

Each subspace is recovered to ϵ error w/ delay only $O(r \log n \log(1/\epsilon))$ w.h.p. if

1. **left & right incoherence:** *P_j's are μ-incoherent; and $\boldsymbol{a}_t$'s are zero mean, i.i.d., and element-wise bounded* $(\max_t \max_i (\boldsymbol{a}_t)_i^2 \leq C\lambda_i(\boldsymbol{\Lambda}))$;
2. *$\boldsymbol{v}_t$'s zero mean, mutually independent, indep. of $\boldsymbol{x}_t, \ell_t$, & $\|\boldsymbol{v}_t\|^2 \leq cr\epsilon^2\lambda^-$,*
3. **outlier frac:** *max-outlier-frac-col $\in O(1/r)$, max-outlier-frac-row$^\alpha \in O(1)$;*
4. **slow subspace change & outlier magnitudes lower bounded:**
    1. $t_{j+1} - t_j > Cr \log n \log(1/\epsilon)$,
    2. $\Delta := \max_j \mathrm{SE}(P_{j-1}, P_j)$ *satisfies* $C_1\sqrt{r\lambda^+}(\Delta + 2\epsilon) < \min_t \min_{i \in \mathcal{T}_t} (\boldsymbol{x}_t)_i$
5. **initializ:** $\mathrm{SE}(\hat{P}_0, P_0) \leq 0.25$, *and* $C_1\sqrt{r\lambda^+}\mathrm{SE}(\hat{P}_0, P_0) < \min_t \min_{i \in \mathcal{T}_t} (\boldsymbol{x}_t)_i$

Can relax outlier mag lower bound to: most outlier magnitudes are lower bounded, while the rest are small enough s.t. their squared sum is upper bounded

Main Result: Other conclusions

With probability at least $1 - 10dn^{-11}$ all the following also hold:

- at all times t, $\hat{\mathcal{T}}_t = \mathcal{T}_t$: exact outlier support recovery
- $t_j \le \hat{t}_j \le t_j + 2\alpha$ (recall $\alpha = \mathcal{O}(r \log n)$): subspace change detected quickly
- subspace error decays exponentially with each new update:

$$\mathrm{SE}(\hat{P}_{(t)}, P_{(t)}) \le \begin{cases} \epsilon + \Delta & t \in [t_j, \hat{t}_j + \alpha) \\ (0.3)^{k-1}(\epsilon + \Delta) & t \in [\hat{t}_j + (k-1)\alpha, \hat{t}_j + k\alpha) \\ \epsilon & t \in [\hat{t}_j + K\alpha, t_{j+1}) \end{cases}$$

- $\|\hat{\boldsymbol{x}}_t - \boldsymbol{x}_t\| = \|\hat{\boldsymbol{\ell}}_t - \boldsymbol{\ell}_t\| \le 1.2(\mathrm{SE}(\hat{P}_{(t)}, P_{(t)}) + \epsilon)\|\boldsymbol{\ell}_t\|$: error stable and small at all times
- Offline: $\mathrm{SE}(\hat{P}^{off}_{(t)}, P_{(t)}) \le \epsilon$, $\|\hat{\boldsymbol{x}}^{off}_t - \boldsymbol{x}_t\| = \|\hat{\boldsymbol{\ell}}^{off}_t - \boldsymbol{\ell}_t\| \le \epsilon\|\boldsymbol{\ell}_t\|$ at all t.
- Memory complexity is $O(nr \log n \log(1/\epsilon))$: nearly-optimal
- Time complexity is $O(ndr \log(1/\epsilon))$: comparable to complexity for simple (non-robust) PCA via SVD

Proof outline: assume $\boldsymbol{v}_t = 0$

- Slow subspace change, denseness (incoherence) of columns of $P_{(t)}$, and bound on max-outlier-frac-col $\Rightarrow$ RIP constant of $\boldsymbol{\Phi} = (\boldsymbol{I} - \hat{P}_{t-1}\hat{P}_{t-1}{}')$ is small.
- This + CS guarantee + lower bnd on outlier magnitudes $\Rightarrow$ $\boldsymbol{e}_t := \boldsymbol{x}_t - \hat{\boldsymbol{x}}_t$ satisfies

$$\boldsymbol{e}_t = \boldsymbol{I}_{\mathcal{T}_t}(\boldsymbol{I}_{\mathcal{T}_t}{}'\boldsymbol{\Phi}\boldsymbol{I}_{\mathcal{T}_t})^{-1}\boldsymbol{I}_{\mathcal{T}_t}{}'\boldsymbol{\Phi}\boldsymbol{\ell}_t$$

  notice $\boldsymbol{e}_t$ is data-dependent, sparse, with changing support
- Since $\hat{\boldsymbol{\ell}}_t = \boldsymbol{y}_t - \hat{\boldsymbol{x}}_t$ and $\boldsymbol{y}_t = \boldsymbol{\ell}_t + \boldsymbol{x}_t$, $\hat{\boldsymbol{\ell}}_t = \boldsymbol{\ell}_t + \boldsymbol{e}_t$; using $\hat{\boldsymbol{\ell}}_t$'s for subspace update is thus a PCA in Data-Dependent Noise problem:
  - made the analysis harder – almost no results for data-dependent noise
  - use results from [Vaswani, Narayanamurthy, ISIT'18, Allerton'17], [Vaswani, Guo, NIPS'16]

Overall idea: show error reduction after first subspace update $\Rightarrow$ further reduced error seen by the CS step in next interval $\Rightarrow$ further reduced $\boldsymbol{e}_t = \boldsymbol{x}_t - \hat{\boldsymbol{x}}_t = \hat{\boldsymbol{\ell}}_t - \boldsymbol{\ell}_t$ $\Rightarrow$ use to show further error reduction in next subspace update

- Key idea in showing error reduction after subspace update: $\boldsymbol{e}_t$ sparse with changing support (max-outlier-frac-row$^{\alpha} \leq c$)
- thus norm of time-average of $\mathbb{E}[\boldsymbol{e}_t\boldsymbol{e}_t{}']$ is $\sqrt{c}$ times $\|\mathbb{E}[\boldsymbol{e}_t\boldsymbol{e}_t{}']\|$

Proof technique I

Theorem (Davis-Kahan $\sin\theta$ theorem - simplified)

Let $\boldsymbol{D}_0$ be a Hermitian matrix whose span of top r eigenvectors equals span(P)*. Let $\boldsymbol{D}$ be the Hermitian matrix with top r eigenvectors $\hat{P}$. Then,*

$$\mathrm{SE}(\hat{P}, P) \leq \frac{\|\boldsymbol{D} - \boldsymbol{D}_0\|_2}{\lambda_r(\boldsymbol{D}_0) - \lambda_{r+1}(\boldsymbol{D}_0) - \|\boldsymbol{D} - \boldsymbol{D}_0\|_2} \tag{4}$$

as long as the denominator is positive.

- Apply above w/ $\boldsymbol{D}_0 = \frac{1}{\alpha}\sum_t \boldsymbol{\ell}_t\boldsymbol{\ell}_t' = P(\frac{1}{\alpha}\sum_t \boldsymbol{a}_t\boldsymbol{a}_t')P'$ and $\boldsymbol{D} = \frac{1}{\alpha}\sum_t \hat{\boldsymbol{\ell}}_t\hat{\boldsymbol{\ell}}_t{}'$
- Use matrix Bernstein and Vershynin's sub-Gaussian result to conclude that, w.h.p.

$$\mathrm{SE}(\hat{P}_{j,k}, P_j) \lessapprox \frac{2\left\|\frac{1}{\alpha}\sum_t \mathbb{E}[\boldsymbol{\ell}_t\boldsymbol{e}_t{}']\right\| + \left\|\frac{1}{\alpha}\sum_t \mathbb{E}[\boldsymbol{e}_t\boldsymbol{e}_t{}']\right\| + 0.3\epsilon\lambda^+}{\lambda^- - 0.1\epsilon\lambda^+ - \text{numerator}}$$

Proof technique II

- Use Cauchy-Schwarz for sums of products of matrices to bound above terms
- In k-th subspace update step: start with $\mathrm{SE}(\hat{P}_{j,k-1}, P_j) \leq 0.3^{k-1}(\Delta + \epsilon)$,

$$\left\| \frac{1}{\alpha} \sum_t \mathbb{E}[\boldsymbol{e}_t \boldsymbol{e}_t{}'] \right\| = \left\| \frac{1}{\alpha} \sum_t \boldsymbol{I}_{\mathcal{T}_t} \boldsymbol{B}_t \boldsymbol{\Phi} P_j \boldsymbol{\Lambda} P_j' \boldsymbol{\Phi} \boldsymbol{B}_t{}' \boldsymbol{I}_{\mathcal{T}_t}{}' \right\|$$
$$\leq \sqrt{\left\| \frac{1}{\alpha} \sum_t \boldsymbol{I}_{\mathcal{T}_t} \boldsymbol{I}_{\mathcal{T}_t}{}' \right\| (\max_t (\|\boldsymbol{B}_t\|^2 \mathrm{SE}(\hat{P}_{j,k-1}, P_j))^2 \lambda^+)^2}$$
$$\leq \sqrt{\text{max-outlier-frac-row}^{\alpha}} (1.2)^2 (0.3^{k-1}(\Delta + \epsilon))^2 \lambda^+$$

- Similarly can show that

$$\left\| \frac{1}{\alpha} \sum_t \mathbb{E}[\boldsymbol{\ell}_t \boldsymbol{e}_t{}'] \right\| \leq \sqrt{\text{max-outlier-frac-row}^{\alpha}} (1.2)^2 0.3^{k-1} (\Delta + \epsilon) \lambda^+$$

- Use above to show that $\mathrm{SE}(\hat{P}_{j,k}, P_j) \leq 0.3^k(\Delta + \epsilon)$

Comparisons: assumptions, time, memory

	PCP	AltProj	GD	ReProCS
max-outlier-frac-row	$O(1/r_L)$	$O(1/r_L)$	$O(1/\sqrt{r_L^3})$	$O(1)$
max-outlier-frac-col	$O(1/r_L)$	$O(1/r_L)$	$O(1/\sqrt{r_L^3})$	$O(1/r)$
slow subspace change	No	No	No	Yes
lower bound on outlier mag.	No	No	No	Yes
initial data Y_{init}				assumptions of AltProj: max-outlier-frac $\leq c/r$
# of algo parameters	1	2	5	4
time	$O(nd^2 \frac{1}{\epsilon})$	$O(ndr_L^2 \log \frac{1}{\epsilon})$	$O(ndr_L \log \frac{1}{\epsilon})$	$O(ndr \log \frac{1}{\epsilon})$
memory	$O(nd)$	$O(nd)$	$O(nd)$	$O(nr \log n \log \frac{1}{\epsilon})$

Table: An $n \times d$ data matrix $Y := L + X + V$; rank of L is $r_L = rJ$: r is subspace dimension at any time, J is total number of subspace changes.

Other work:

PCP (Candes et al): needs uniform random outlier support and strong incoherence but allows allows max-outlier-frac $\in O(1)$ and $r_L \in O(n/(\log^2 n))$

Pros and Cons of ReProCS – Video context

Pros

1. Allows video objects that move every so often or move very slowly
   - tolerates max-outlier-frac-row $\in O(1)$; others need $O(1/r_L)$
2. Typically, also allows larger-sized foreground objects than other methods
   - tolerates max-outlier-frac-col $\in O(1/r)$; others need $O(1/r_L)$
     - ★ e.g., if $r = O(\log n)$, but $J = O(d/(r \log n))$, then $r_L = rJ$ is almost $O(d)$: ReProCS still works, all others fail
3. ReProCS is the fastest; and has nearly optimal memory complexity

Cons: needs

1. Slowly changing subspace of video backgrounds
   - (usually valid for static camera videos)
2. Foreground pixels are either different enough from background pixels or very similar
   - outlier magnitudes are either large or very small
   - mild assumption: follows from definition of outlier as a large magnitude corruption

Discussion

- First set of complete guarantees for any online / dynamic / streaming RPCA solution:
  - first algorithm for static or dynamic RPCA that tolerates a constant fraction of outliers per row without assumptions on outlier support
  - first method with near-optimal memory complexity
- Earlier work:
  - partial guarantees (required assumptions on intermediate algo. estimates):
    - ReProCS [Qiu,Vaswani,Lois,Hogben,ISIT'13,T-IT'14],
    - ORPCA [Feng et a;.,NIPS'13]
  - complete guarantee for ReProCS but with more assumptions: ReProCS [Lois,V.,ICASSP'15,ISIT'15], [Zhan,Lois,Guo,V.,AISTATS'16]
- New proof techniques needed to be developed
  - useful for various other problems, e.g., correlated-PCA [Vaswani,Guo,NIPS'16]

Methods compared

Online:

- ReProCS – [Qiu,V., Allerton'10], [Qiu,V.,Lois,Hogben,T-IT'14], [Zhan,Lois,Guo,V.,AISTATS'16]
- Modified-PCP (Robust PCA with Partial Subspace Knowledge) – [Zhan, V.,T-SP'15]
- GRASTA – [He, Balzano, et al, CVPR 2012]
- pROST – [Seidel et al. arXiV 2013]
- ORPCA – [Feng, Xu, et al, NIPS 2013], online algorithm to solve PCP

Batch:

- PCP (IALM) – batch algo. for static RPCA - convex opt.
- AltProj – batch algo. for static RPCA - Alt-Min
- RPCA-GD – batch algo. for static RPCA - Grad. Desc.
- NO-RMC – batch algo. for static robust matrix completion
- GoDec – [Zhou and Tao, ICML'11]
- 2PRPCA – [Gao et al, PAMI'14] Block-sparse RPCA for salient motion detection
- 3TD – [Oreifej et al, PAMI'13] Simultaneous video stabilization and moving object detection in turbulence
- PRMF – [Wang et al, ECCV'12] A Probabilistic approach to Robust Matrix Factorization

Evaluating basic ReProCS for background recovery: visual comparisons I

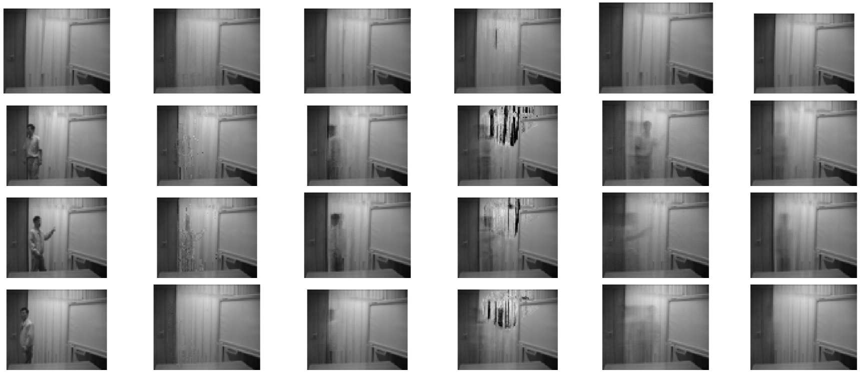


Figure: Background recovery comparison. ReProCS works best; is the fastest among provable methods; and needs only 60 frames of memory instead of all 1200.

Evaluating basic ReProCS for background recovery: visual comparisons II

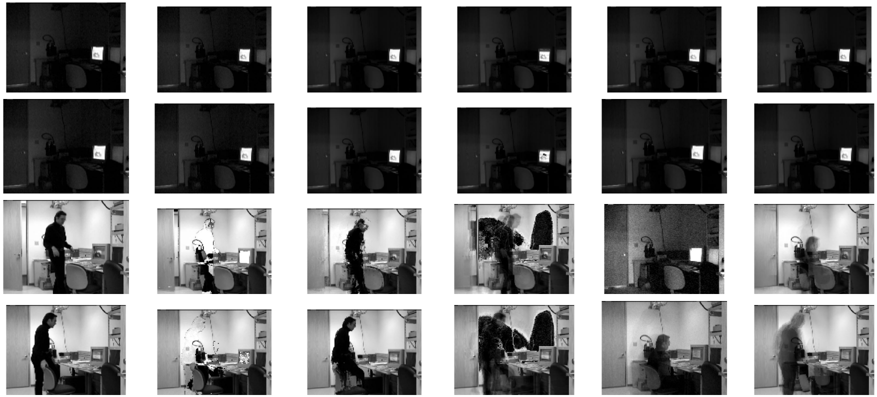


Figure: Background recovery comparison for frames $t = 400 + 60, 200, 688, 999$.

Improved Practical ReProCS

- Also exploited slow support change of the foreground object(s) when possible
- Applications
  - Video layering:
    - Foreground recovery for video surveillance, and Background recovery and subspace tracking - useful for simulating video textures
  - Video Denoising and Enhancement
  - Work of Selin Aviyente et al.:
    - Tensor ReProCS for detecting anomalous connectivity patterns in social networks data on-the-fly

Video surveillance application - foreground recovery

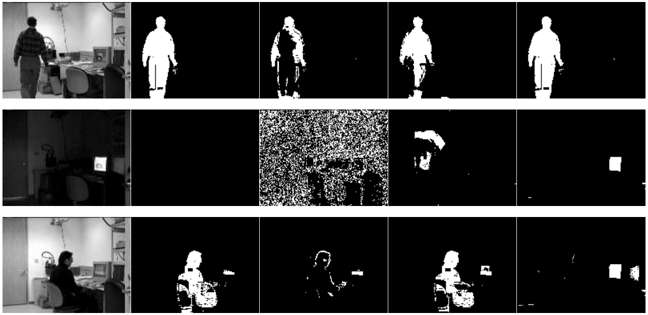


Figure: Foreground recovery ($t = t_{\text{train}} + 35, 500, 1300$)

Background recovery and subspace tracking - useful for simulating video textures

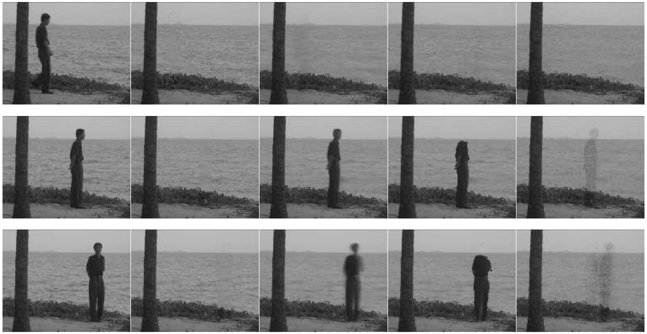


Figure: Background recovery for modeling ($t = t_{\text{train}} + 30, 80, 140$).

Quantitative comparisons for video layering (fg-bg separation) I

- Comparisons on 15 challenging sequences from the CDnet video dataset:
  - two sequences from 'Baseline' category, three from Dynamic Background (DB) category, two from Camera Jitter category, three from Intermittent Object Motion (IOM), category, three from 'Shadows' category, and two from 'Thermal' category.
- For all these videos, foreground pixels have been manually labeled. These serve as ground truth.
- Use F measure (standard metric for evaluating information retrieval methods): $F = 2\frac{\mathrm{Recall}\cdot\mathrm{Precision}}{\mathrm{Recall}+\mathrm{Precision}}$
  - Recall is number of pixels correctly labeled as foreground (fg) as a fraction of total number of pixels labeled by the algorithm as fg.
  - $\mathrm{Precision}$ is number of pixels correctly labeled as fg as a fraction of total number of true fg pixels.
- Comparisons done by Sajid Javed and Thierry Bouwmans.

Quantitative comparisons for video layering (fg-bg separation) II

Table: Comparing F scores (higher is better) and time for various categories of videos from CDnet dataset. DB: Dynamic Backgrounds, IOM: Intermittent Object Motion.

Provable Methods	Baseline	DB	Camera Jitter	Shadow	Thermal	IOM	Average	Time (secs/frame)
PCP (batch)	0.75	0.69	0.62	0.73	0.65	0.48	0.65	4.19
AltProj (batch)	**0.78**	**0.71**	0.60	**0.76**	0.69	0.58	**0.68**	2.38
NO-RMC (batch)	0.71	0.64	0.64	0.66	**0.71**	0.50	0.64	2.85
RPCA-GD (batch)	0.74	0.62	0.68	**0.75**	0.66	0.49	0.65	2.46
ReProCS-provable (online)	**0.77**	**0.77**	**0.69**	0.71	**0.74**	**0.70**	**0.73**	0.74
Modified-PCP (online)	0.75	0.64	**0.70**	0.65	0.69	**0.70**	**0.68**	**0.44**
Heuristics Methods	Baseline	DB	Camera Jitter	Shadow	Thermal	IOM	Average	Time
ReProCS (online)	0.80	0.76	0.72	0.75	0.77	**0.69**	**0.74**	**0.61**
GRASTA (online)	0.66	0.35	0.43	0.52	0.42	0.35	0.45	1.16
3TD (batch)	**0.88**	0.75	0.72	0.68	**0.78**	0.55	0.72	2.17
2PRPCA (batch)	**0.92**	**0.79**	**0.81**	**0.80**	0.76	**0.65**	**0.78**	1.63
GoDec (batch)	0.77	0.58	0.48	0.51	0.62	0.38	0.55	1.56
OR-PCA (online)	0.86	0.75	0.70	0.74	0.76	0.56	0.72	**0.22**
pROST (online)	0.79	0.59	0.79	0.70	0.58	0.48	0.65	2.03
PRMF	**0.92**	**0.77**	**0.85**	**0.88**	**0.83**	0.48	**0.78**	2.40

- For Intermittent Object Motion (IOM) and Dynamic Background (DB), ReProCS has *the best* or *second best* performance compared with *all* methods.
  - IOM implies large max-outlier-frac-row, DB implies large r_L.
- Compared with the provable methods (simple methods), ReProCS has best average performance while also being very fast.

Video denoising of very noisy videos

- Idea: large variance noise can always be split as frequently occurring small noise and occasionally occurring large outliers.
- Approach:
  - use ReProCS to get $\hat{\boldsymbol{x}}_t$ and $\hat{\boldsymbol{\ell}}_t$ for each frame t
  - apply a state-of-art denoiser, VBM-3D, to each layer separately
  - use denoised $\hat{\boldsymbol{\ell}}_t$ in most cases; sometimes use denoised image (add up denoised layers)
- Waterfall video: `http://www.ece.iastate.edu/~hanguo/denoise.html`, `https://youtu.be/pycgXFQAC9Y`

Peak Signal-to-Noise-Ratio (PSNR) comparisons for Video Denoising

σ	ReProCS-LD	PCP-LD	AltProj-LD	GRASTA-LD	VBM3D	MLP
25	32.78 (73.54)	**32.84** (198.87)	31.98 (101.78)	28.11 (59.43)	32.02 (24.83)	28.26 (477.22)
30	**32.68** (73.33)	32.60 (185.47)	31.56 (106.30)	26.89 (58.76)	30.96 (23.96)	26.96 (474.26)
50	**32.27** (73.14)	31.65 (195.77)	30.09 (128.35)	23.97 (58.23)	27.99 (24.14)	18.87 (477.60)
70	**31.79** (69.77)	30.67 (197.94)	29.63 (133.53)	21.81 (55.45)	24.42 (21.01)	15.03 (478.73)

Table: PSNR measures denoised image quality, so larger PSNR value is better.

- The video is made noisy by adding zero mean Gaussian noise with standard deviation σ varied from 25 to 70.
- Images were of size $n = 108 \times 192$ and video length $d = 650$.
- ReProCS-LD (ReLD): ReProCS based layering denoising: use ReProCS to get low-rank and sparse layers, then use VBM3D on each layer.
- Similarly PCP-LD, AltProj-LD and GRASTA-LD.

Low-light Video Enhancement: "seeing in the dark"

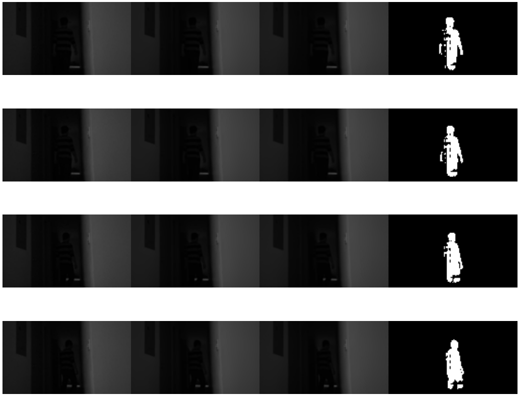


Figure: Original, V-BM-3D, K-SVD, ReProCS. In the video, a person is walking through a hallway. ReProCS successfully "sees" the person.

Summary and Open Questions

Pros and Cons of ReProCS – Video context

Pros

1. Allows video objects that move every so often or move very slowly
   - tolerates max-outlier-frac-row $\leq c$; others need $\leq c/r_L$ ($r_L = \text{rank}(L)$)
2. Typically, also allows larger-sized foreground objects than other methods
   - tolerates max-outlier-frac-col $\leq c/r$; others need c/r_L
     - ★ e.g., if $r = O(\log n)$, but $J = O(n)$, then $r_L = r + J = O(n)$: ReProCS works, others fail
3. ReProCS is the fastest; has nearly optimal storage complexity; and is online

Cons: needs

1. Slowly changing subspace of video backgrounds
   - (usually valid for static camera videos)
2. Most foreground pixels are different enough from background pixels
   - needs a lower bound on most outlier magnitudes
   - mild assumption: follows from definition of outlier as large magnitude corruption

Open Questions

1. Extensions to dynamic robust matrix completion, undersampled RPCA
2. Remove lower bound on outlier magnitudes and still get a guarantee (even if with a tighter outlier fraction bound) – replace CS step by thresholding
3. Moving camera; sudden scene changes; algorithm speed-up w/o significant loss in performance; streaming ReProCS
4. Applications:
    - use video layering to simplify computer vision and video analytics' tasks
    - functional MRI based brain activity pattern tracking;
    - tracking user preferences over time
5. Open: use in related problems: dynamic subspace clustering, phaseless robust PCA (preliminary ongoing work on low rank phase retrieval)

Main references discussed in detail

Solutions for original RPCA

1. PCP: *Robust Principal Components Analysis?*, J. ACM, 2009.
2. AltProj: *Non-Convex Robust PCA*, NIPS, 2014.
3. RPCA-GD: *Fast Algorithms for Robust PCA via Gradient Descent*, NIPS 2016

Our work on ReProCS and ReProCS-NORST:

1. Nearly Optimal Robust Subspace Tracking, ICML 2018
2. Online (and Offline) Robust PCA: Novel Algorithms and Performance Guarantees, AISTATS 2016
3. Recursive Robust PCA or Recursive Sparse Recovery in Large but Structured Noise, IEEE Trans. IT, 2014
4. An Online Algorithm for Separating Sparse and Low-dimensional Signal Sequences from their Sum, IEEE Trans. SP, 2014
5. Real-time Robust Principal Components' Pursuit, Allerton, 2010

PCA in data-dependent (correlated) noise:

- Correlated-PCA: Principal Component Analysis when Data and Noise are Correlated, NIPS 2016
- Finite Sample Guarantees for PCA in Non-Isotropic and Data-Dependent Noise, Allerton 2017.

PCA for big-data

PCA for big-data I

- When data is outlier-corrupted (robust PCA):
  - much harder problem: SVD fails; focus of most of this talk
- For high-dimensional data: full SVD is expensive: $O(nd^2)$ for $n \times d$ matrix
  - partial (r-SVD), e.g., svds, is fast if data well-approximated as lying in a low-dimensional subspace (has large eigen-gap)
  - but is much slower (needs many more iterations) when gap is small
  - r-SVD complexity is $O(ndrq)$: q is the number of iterations.
  - Need: $q = O\left(\max\left(\frac{\log(d/\epsilon)}{\sqrt{\epsilon}}, \frac{\log(d/\epsilon)}{\sqrt{\mathrm{gap}}}\right)\right)$ where $\mathrm{gap} := (\sigma_r - \sigma_{r+1})/\sigma_r$
    [Musco,Musco,NIPS'15], [Chinmay Hegde talk]
  - Constant gap: r-SVD cost is $O(ndr)$ within log factors;
    Small gap: r-SVD cost is $O(ndr/\sqrt{\epsilon})$ within log factors
  - In the above discussion,

PCA for big-data II

- ⋆ the guarantee is that the partial SVD of a given matrix is computed to ϵ error in the stated amount of time; no conditions on size of matrix, result holds for all matrices.
- ⋆ it says nothing about how "close" the estimated singular vectors or their subspace is to the top r singular vectors of the true (noise-free) data – in fact here there is no notion of true data and noise
- ⋆ just a result about computing top r singular vectors of a given matrix – fits in the field called randomized numerical LA (randNLA); dominated by computer scientists
- ⋆ still an active research area
- ⋆ papers of Woodruff et al, Tropp et al, Musco and Musco 2015, much later work in NIPS 2016 and later

- So far we have tried to answer the question: given a matrix M that is large sized (but small enough that it can be "stored" in memory), how to quickly can we find its left singular vectors?

PCA for big-data III

- Now consider the statistical (signal processing) problem: observed data = true data + noise. To accurately recover the subspace of true data, we need to have enough samples d.
  If noise is Gaussian, then we need at least n samples, i.e., need $d \geq Cn$
  But then, if n is large, there may not be enough memory to store the resulting $n \times d$ matrix.
  This is where streaming PCA algorithms become important.
  Claims: if number of samples d is large enough, then the streaming-PCA-algorithm will approximate the true data's subspace.

- Memory issues for high-dimensional and noisy data: resolved by streaming solutions, but need a larger lower bound on the number of columns d (sample complexity) than what guarantees for batch PCA need
  - stochastic power method and variants
  - use optimal memory – $O(nr)$ – but need more samples d to accurately get the true principal components (as compared to simple SVD used for batch-PCA)

PCA for big-data IV

- for the batch PCA solution, $d = O(n/\epsilon^2)$ suffices to get an estimate that is within ϵ error of the true subspace;
- for block-stochastic power method, need d to be at least $C\ (nr \log n/\epsilon^2)$ to get within ϵ error of the true subspace (under certain data models)
- also still an active research area
- papers of Mitliakgas et al, NIPS'13, Hardt and Price, NIPS'14 (Noisy power method), Jain et al, ?'16

Power method and Orthogonal iteration: simplest algorithm for partial SVD

- Power method: for 1-PCA
  - let $A = \frac{1}{\alpha} \sum_t \boldsymbol{y}_t \boldsymbol{y}_t{}'$.
  - generate h with entries iid $\mathcal{N}(0, 1)$
  - set $q_0 = h / \|h\|$.
  - for $\tau = 1, 2, \dots, T$, do
    - ★ compute $s_{\tau+1} = A q_\tau$,
    - ★ $q_{\tau+1} = s_{\tau+1} / \|s_{\tau+1}\|$
  - Output q_T as estimate of top singular/eigen vector of A.
- Extension for r-PCA (Orthogonal Iteration)
  - let $A = \frac{1}{\alpha} \sum_t \boldsymbol{y}_t \boldsymbol{y}_t{}'$.
  - generate an $n \times r$ matrix H with entries iid $\mathcal{N}(0, 1)$
  - get Q_0 by QR decomposition: $H = Q_0 R_0$
  - For $\tau > 0$, do
    - ★ compute $S_{\tau+1} = A Q_\tau$,
    - ★ get $Q_{\tau+1}$: QR decomp on $S_{\tau+1} = Q_{\tau+1} R_{\tau+1}$
  - Output Q_T

Stochastic Gradient Descent (GD) I

- Basic GD for $\min_x \sum_{t=1}^{\alpha} F_t(y_t, x)$ (y_t is observed data).

$$\hat{\boldsymbol{x}}^{\tau+1} = \hat{\boldsymbol{x}}^{\tau} - \eta_\tau \sum_{t=1}^{\alpha} \nabla F_t(y_t, \hat{\boldsymbol{x}}^{\tau})$$

- Stochastic GD:

$$\hat{\boldsymbol{x}}^{\tau+1} = \hat{\boldsymbol{x}}_{\tau} - \eta_\tau \nabla F_\tau(y_\tau, \hat{\boldsymbol{x}}_\tau)$$

  advantage: (a) do not need to store y_t's; (b) faster: computing one gradient at each time τ instead of α gradients; (c) if x actually changes with time, this can track the changes.

- Adaptive filters do exactly the above.
- 1-PCA: top eigenvector is a solution to $\min_{v \neq 0} -\frac{v'(\sum_t \boldsymbol{y}_t \boldsymbol{y}_t')v)}{\|v\|^2}$; thus, $F_t(\boldsymbol{y}_t, v) = -\frac{v' \boldsymbol{y}_t \boldsymbol{y}_t' v}{\|v\|^2}$

Stochastic power method variants: solve both speed and memory issues I

- PCA for big-data (both speed and memory issues) solved by "stochastic power method" based algorithms
- Krasulina's algorithm for 1-PCA: stochastic GD [Krasulina'69]
  - initialize w_0
  - for $t = 1, 2, \ldots, T$, do
    - ★ $w_t \leftarrow w_{t-1} + \eta_t(\boldsymbol{y}_t \boldsymbol{y}_t{}' - \frac{w_{t-1}{}' \boldsymbol{y}_t \boldsymbol{y}_t{}' w_{t-1}}{\|w_{t-1}\|^2} \boldsymbol{I}) w_{t-1}$
- Block-Stochastic Power Method
  - Generate H with entries iid $\mathcal{N}(0,1)$;
  - QR decomposition on H: get Q_0
  - For $\tau = 1, 2, \ldots, T$, do
    - ★ $S_{\tau+1} \leftarrow 0$,
    - ★ For $t = B\tau + 1, \ldots, (B+1)\tau$:
      $S_{\tau+1} \leftarrow S_{\tau+1} + \frac{1}{B} \boldsymbol{y}_t(\boldsymbol{y}_t' Q_\tau)$,
      End For.
    - ★ QR decomp on $S_{\tau+1}$: get $Q_{\tau+1}$
      (computes $Q_{\tau+1}$ as QR dec of $S_{\tau+1} := (\frac{1}{B} \sum_{t=B\tau+1}^{t=(B+1)\tau} \boldsymbol{y}_t \boldsymbol{y}_t{}') Q_\tau)$

Stochastic power method variants: solve both speed and memory issues II

```
    End For
```
- Output Q_T

- This takes $O(nr)$ memory and $O(nr)$ time
- Sample complexity: needs $\alpha = O(nr^2 \log n/\epsilon^2)$ samples for spiked covariance model (for fixed eigenvalues and final error) instead of $\alpha = O(n/\epsilon^2)$

Basic PCA (svd) – guarantees for non-isotropic and/or data-dependent noise settings

This section is based on

- N. Vaswani and P. Narayanamurthy, “PCA in Sparse Data-Dependent Noise”, ISIT 2018
- N. Vaswani and P. Narayanamurthy, “Finite Sample Guarantees for PCA in Non-Isotropic and Data-Dependent Noise”, Allerton 2017
- N. Vaswani and H. Guo, “Correlated-PCA: PCA when Data and Noise are Correlated”, NIPS 2016

Principal Components Analysis (PCA)

- Most commonly used technique for dimension reduction; first step in exploratory data analysis, classification, clustering, etc.
- Given data vectors $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_\alpha$ that lie in $\Re^n$, find their best r-dimensional dimensional subspace approximation,
  - i.e., find an $n \times r$ basis matrix (matrix with orthonormal entries), $\hat{P}$, so that,
  $$\|\boldsymbol{Y} - \hat{P}\hat{P}'\boldsymbol{Y}\|_2$$
  is minimized. Here $\boldsymbol{Y} := [\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_\alpha]$ is $n \times \alpha$
- Solution [Eckart-Young]: SVD
  - $\hat{P}$: top r singular vectors of $\boldsymbol{Y}$
- **Question: if intepret $\boldsymbol{y}_t$ as true data (signal) plus noise, how close is $\hat{P}$ to the signal subspace?**

Quantifying subspace recovery error:

- For two subspaces (defined by their basis matrices), $\hat{P}$, P, the subspace error (SE) betwen their column spans is quantified by

$$\mathrm{SE}(\hat{P}, P) := \|(\boldsymbol{I} - \hat{P}\hat{P}')P\|_2$$

  This measures the sine of the largest principal angle between the subspaces ($'$ denotes transpose)

Existing results

- Almost all existing work that studies the SVD solution
  - assumes the spiked covariance model:
    - data and noise uncorrelated, noise is isotropic (white)
  - most work provides asymptotic guarantees
- Finite sample guarantees: Nadler'08
  - assumed the spiked covariance model;
  - studied $r = 1$ dimensional PCA;
  - assumed Gaussian data and noise

Our Work

- Noise can be non-isotropic (colored) and data-dependent;
- Study PCA for a general dimension $r \geq 1$;
- Data and noise either bounded or sub-Gaussian; in the bounded case: can achieve near-optimal sample complexity in certain regimes
  - most sensors power-limited: bounded-ness is a more practical assumption than Gaussianity
- Application: helps obtain the first simple guarantee for dynamic robust PCA

Problem setting [Vaswani,Guo,NIPS'16, Correlated-PCA], [Vaswani,Narayanamurthy,ISIT'18]

- For $t = 1, 2, \ldots, \alpha$, we are given n-length data vectors,

$$\boldsymbol{y}_t := \boldsymbol{\ell}_t + \boldsymbol{w}_t + \boldsymbol{v}_t, \text{ where } \boldsymbol{\ell}_t = P\boldsymbol{a}_t, \ \boldsymbol{w}_t = \boldsymbol{M}_t\boldsymbol{\ell}_t, \ \mathbb{E}[\boldsymbol{\ell}_t\boldsymbol{v}_t{}'] = 0$$

  where
  - P: subspace basis: $n \times r$ matrix with orthonormal columns and $r \ll n$
  - $\boldsymbol{\ell}_t$: true data ("signal") vector
  - $\boldsymbol{w}_t$: data-dependent noise
  - $\boldsymbol{v}_t$: uncorrelated noise, with covariance matrix $\boldsymbol{\Sigma}_v$
  - the matrices $\boldsymbol{M}_t$ are *unknown* and typically such that $\mathbb{E}[\boldsymbol{\ell}_t\boldsymbol{w}_t'] \neq 0$: so $\boldsymbol{w}_t$ is correlated with $\boldsymbol{\ell}_t$

- Observe: in general, $\boldsymbol{w}_t$'s do not lie in a lower dim subspace of $\Re^n$.

- Applications: PCA in sparse data-dependent noise; two special cases:
  (a) PCA in missing data:

$$\boldsymbol{y}_t = \boldsymbol{\ell}_t - \boldsymbol{I}_{\mathcal{T}_t}\boldsymbol{I}_{\mathcal{T}_t}{}'\boldsymbol{\ell}_t, \ \mathcal{T}_t: \text{ missing entries' set at } t$$

  (b) subspace update step in ReProCS for dynamic robust PCA:

$$\hat{\boldsymbol{\ell}}_t = \boldsymbol{\ell}_t - \boldsymbol{I}_{\mathcal{T}_t}\boldsymbol{B}_t(\boldsymbol{I} - \hat{P}_{t-1}\hat{P}_{t-1}{}')(\boldsymbol{\ell}_t + \boldsymbol{v}_t)$$

Statistical assumptions: bounded or sub-Gaussian

- Recall: for $t = 1, 2, \ldots, \alpha$, we are given n-length data vectors,

$$\boldsymbol{y}_t := \boldsymbol{\ell}_t + \boldsymbol{w}_t + \boldsymbol{v}_t, \text{ where } \boldsymbol{\ell}_t = P\boldsymbol{a}_t, \ \boldsymbol{w}_t = \boldsymbol{M}_t\boldsymbol{\ell}_t, \ \mathbb{E}[\boldsymbol{\ell}_t\boldsymbol{v}_t'] = 0$$

- Bounded data and noise
  - $\boldsymbol{a}_t$'s zero mean, i.i.d., element-wise bounded, with diagonal cov $\boldsymbol{\Lambda}$
  - $\boldsymbol{v}_t$'s zero mean, i.id., bounded, with cov $\boldsymbol{\Sigma}_v$
  - $\boldsymbol{w}_t = \boldsymbol{M}_t\boldsymbol{\ell}_t = \boldsymbol{M}_tP\boldsymbol{a}_t$ (model on $\boldsymbol{a}_t$ implies model on $\boldsymbol{w}_t$)
  - let $\lambda^- := \lambda_{\min}(\boldsymbol{\Lambda})$, $\lambda^+ := \lambda_{\max}(\boldsymbol{\Lambda})$, $f := \frac{\lambda^+}{\lambda^-}$, $\lambda_v^+ := \lambda_{\max}(\boldsymbol{\Sigma}_v)$
  - $|(\boldsymbol{a}_t)_i|^2 \le \eta\lambda_i$ and $\|\boldsymbol{v}_t\|_2^2 \le r_v\lambda_v^+$; r_v: effective noise dimension of $\boldsymbol{v}_t$
- Sub-Gaussian data and noise:
  - above, but $\boldsymbol{a}_t$'s and $\boldsymbol{v}_t$'s are sub-Gaussian with sub-Gaussian norms bounded by $c\sqrt{\lambda^+}$ and $c\sqrt{\lambda_v^+}$

Only uncorrelated noise case: $\boldsymbol{y}_t = \boldsymbol{\ell}_t + \boldsymbol{v}_t$, $\mathbb{E}[\boldsymbol{v}_t \boldsymbol{v}_t{}'] = \boldsymbol{\Sigma}_v$

$\hat{P}$: matrix of top r eigenvectors of $\boldsymbol{D} := \frac{1}{\alpha}\sum_{t=1}^{\alpha} \boldsymbol{y}_t \boldsymbol{y}_t{}'$

Corollary ($\boldsymbol{\Sigma}_v = \lambda_v^+ \boldsymbol{I}$ - spiked covariance model)

To ensure $\mathrm{SE}(\hat{P}, P) \le \epsilon$ *w.p. at least* $1 - 10n^{-10}$, *need*

- *bounded-ness and* $\alpha \ge C \frac{f \frac{\lambda_v^+}{\lambda^-}}{\epsilon^2} \max(r_v, r) \log n$
- or *sub-Gaussianity and* $\alpha \ge C \frac{f \frac{\lambda_v^+}{\lambda^-}}{\epsilon^2} n$

Corollary ($\boldsymbol{\Sigma}_v \ne \lambda_v^+ \boldsymbol{I}$ - non-isotropic uncorrelated noise)

To ensure $\mathrm{SE}(\hat{P}, P) \le \epsilon$ *w.p. at least* $1 - 10n^{-10}$, *need*

- *sample complexity lower bound given above* and
- $\lambda_{\max}(\boldsymbol{\Sigma}_v - PP'\boldsymbol{\Sigma}_v PP') - \lambda_{\min}(P'\boldsymbol{\Sigma}_v P) < 0.5\lambda^-$, $\|P_\perp{}'\boldsymbol{\Sigma}_v P\|_2 < 0.45\epsilon\lambda^-$

Near optimal sample complexity if $r_v \in O(r)$, $\lambda_v^+ \le C\epsilon^2\lambda^-$, bounded data, noise

Reasoning

Recall: $\hat{P}$ is top r eigenvectors of $\boldsymbol{D} := \frac{1}{\alpha} \sum_{t=1}^{\alpha} \boldsymbol{y}_t \boldsymbol{y}_t{}'$

- When $\boldsymbol{\Sigma}_v = \lambda_v^+ \boldsymbol{I}$,

$$\mathbb{E}[\boldsymbol{D}] = P \boldsymbol{\Lambda} P' + \boldsymbol{\Sigma}_v = P(\boldsymbol{\Lambda} + \lambda_v^+ \boldsymbol{I}) P' + \lambda_v^+ P_\perp P_\perp{}'$$

  span of top r eigenvectors of $\mathbb{E}[\boldsymbol{D}]$ equals span(P); if α large enough w.h.p. this will approximately be true for $\hat{P}$ too

- But for a general $\boldsymbol{\Sigma}_v$, above argument does not work
  - to see a simple example, if $\boldsymbol{\Sigma}_v = (1.2\lambda^-)(P_\perp)_1 (P_\perp)_1{}'$, then can show that when α large enough, w.h.p. $\mathrm{SE}(\hat{P}, P) \geq 1 - c\epsilon$
  - need assumptions to ensure that noise power outside signal subspace is small

Proof technique

- Use the Davis-Kahan $\sin\theta$ theorem

Theorem (Davis-Kahan $\sin\theta$ theorem)

Let $\boldsymbol{D}_0$ be a Hermitian matrix whose span of top r eigenvectors equals span(P)*. Let $\boldsymbol{D}$ be the Hermitian matrix with top r eigenvectors $\hat{P}$. Then,*

$$\begin{aligned} \mathrm{SE}(\hat{P}, P) &\leq \frac{\|(\boldsymbol{D} - \boldsymbol{D}_0)P\|_2}{\lambda_r(\boldsymbol{D}_0) - \lambda_{r+1}(\boldsymbol{D})} \\ &\leq \frac{\|(\boldsymbol{D} - \boldsymbol{D}_0)P\|_2}{\lambda_r(\boldsymbol{D}_0) - \lambda_{r+1}(\boldsymbol{D}_0) - \lambda_{\max}(\boldsymbol{D} - \boldsymbol{D}_0)} \end{aligned} \tag{5}$$

as long as the denominator is positive. The second inequality follows from the first using Weyl's inequality.

- Apply it with $\boldsymbol{D}_0 = P(P'\boldsymbol{\Sigma}_v P)P'$ and $\boldsymbol{D} = \frac{1}{\alpha}\sum_t \boldsymbol{y}_t \boldsymbol{y}_t{}'$ and simplify
- Use matrix concentration inequalities: matrix Bernstein and Vershynin's sub-Gaussian result

Only data-dependent noise: special case of sparse data-dependent noise

$\boldsymbol{y}_t = \boldsymbol{\ell}_t + \boldsymbol{w}_t$,
$\boldsymbol{\ell}_t = P\boldsymbol{a}_t$ is true data (lies in r-dimensional subspace)
$\boldsymbol{w}_t = \boldsymbol{I}_{\mathcal{T}_t}\boldsymbol{M}_{s,t}\boldsymbol{\ell}_t$ is the sparse data-dependent noise with support $\mathcal{T}_t$

Corollary (bounded case)

$\hat{P}$ (top-r singular vectors of $\frac{1}{\alpha}\sum_t \boldsymbol{y}_t\boldsymbol{y}_t{}'$) satisfies $\mathrm{SE}(\hat{P}, P) \le \epsilon$ with probability at least $1 - 10n^{-10}$ if, for a $q < 1$ and $b < 1$,

1. *the fraction of nonzeros in any row of $[\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_\alpha]$ is at most b,*
2. $\max_t \|\boldsymbol{M}_{s,t}P\|_2 \le q$ *and b, q satisfy $3\sqrt{b}qf < 0.4\epsilon$,*
3. *and $\alpha \ge \alpha_0 := \frac{Cq^2 f}{\epsilon^2} r \log n$ (in the bounded $\boldsymbol{a}_t$ case)*

Condition 1 holds if the sparse noise support $\mathcal{T}_t$ changes "enough" over time

Observe: near-optimal sample complexity α

Understanding the Result - I

- $\|\boldsymbol{M}_{s,t} P\|_2 \leq q < 1$ implies that
  1. noise power $\|\mathbb{E}[\boldsymbol{w}_t \boldsymbol{w}_t{}']\|_2 \leq q^2 \lambda^+$; thus q^2 bounds noise-to-signal ratio
- There are at most $b\alpha$ non-zeros in any row of $[\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_\alpha]$ implies

$$\left\| \frac{1}{\alpha} \sum_{t=1}^{\alpha} \boldsymbol{I}_{\mathcal{T}_t} \boldsymbol{I}_{\mathcal{T}_t}{}' \right\|_2 \leq b.$$

  1. this helps reduce time-averaged noise power $\sqrt{b}$ times: by Cauchy-Schwarz, time-averaged noise power,

$$\| \frac{1}{\alpha} \sum_{t=1}^{\alpha} \mathbb{E}[\boldsymbol{w}_t \boldsymbol{w}_t{}'] \|_2 \leq \sqrt{b} q^2 \lambda^+$$

  2. it also helps reduce time-averaged signal-noise correlation: $\|\mathbb{E}[\boldsymbol{\ell}_t \boldsymbol{w}_t{}']\|_2 \leq q\lambda^+$ but

$$\| \frac{1}{\alpha} \sum_{t=1}^{\alpha} \mathbb{E}[\boldsymbol{\ell}_t \boldsymbol{w}_t{}'] \|_2 \leq \sqrt{b} q \lambda^+$$

Understanding the Result - II

Recall we need $3\sqrt{b}qf < 0.4\epsilon$

- to achieve error level $\epsilon = q/2$, need $\sqrt{b}f < 0.1$,
- notice the need for the $\boldsymbol{I}_{\mathcal{T}_t}$ assumption: without it, $b = 1$: impossible to achieve $\epsilon = q/2$ (or any frac of q)

Sample complexity to achieve $\epsilon = q/2$ is $\alpha \geq Cr \log n$: near-optimal

General data-dependent noise case: $\boldsymbol{y}_t = \boldsymbol{\ell}_t + \boldsymbol{w}_t$, $\boldsymbol{w}_t = \boldsymbol{M}_t \boldsymbol{\ell}_t$

Theorem (data-dependent noise)

Assume bounded-ness. Assume the following data-noise correlation assumption: $\boldsymbol{M}_t$ can be decomposed as $\boldsymbol{M}_t = \boldsymbol{M}_{2,t} \boldsymbol{M}_{1,t}$ with $\|\boldsymbol{M}_{2,t}\| = 1$,

$$\|\boldsymbol{M}_{1,t} P\|_2 \leq q < 1, \text{ and} \tag{6}$$

$$\left\| \frac{1}{\alpha} \sum_{t=1}^{\alpha} \boldsymbol{M}_{2,t} {\boldsymbol{M}_{2,t}}' \right\|_2 \leq b_0 \ll 1. \tag{7}$$

With probability at least $1 - 10n^{-10}$, $\mathrm{SE}(\hat{P}, P) \leq \epsilon$ if

- $\alpha \geq C \frac{q^2 f^2}{\epsilon^2} (r \log n)$ *and*
- $\sqrt{b_0}(2q + q^2) f < 0.45\epsilon$.

Under sub-Gaussianity, lower bound on α changes to $\alpha \geq c \frac{q^2 f^2}{\epsilon^2} n$.

Application: a simple guarantee for dynamic robust PCA

- Given outlier-corrupted data vectors, $\boldsymbol{y}_t := \boldsymbol{\ell}_t + \boldsymbol{x}_t$, $t = 1, 2, \ldots, d$: $\boldsymbol{\ell}_t = P_t \boldsymbol{a}_t$, $\boldsymbol{x}_t$ are sparse outliers, track span(P_t) and $\boldsymbol{\ell}_t$ over time
- Use ReProCS [Qiu,Vaswani,Lois,Hogben,T-IT'14], [Narayanamurthy,Vaswani'17]

Guarantee: If

- subspace P_t is piecewise constant with time and changes "slowly":
  - $P_t = P_{t_j}$ for all $t \in [t_j, t_{j+1})$,
  - $\mathrm{SE}(P_{j-1}, P_j) \leq \Delta$ with $\Delta\sqrt{r\lambda^+} \leq (x_{\min} - 15b_v)/15$, and
  - $(t_{j+1} - t_j) \geq K\alpha$ where $K := C \log \frac{\Delta}{\epsilon}$ and $\alpha = Cf^2(r \log n)$
- columns of P_t are dense, $\boldsymbol{a}_t$ satisfies simple statistical assumptions (bounded, mutually indep.), and outlier fractions bounded:
  - max-outlier-frac-col $\leq \frac{c}{\mu r}$ and max-outlier-frac-row $\leq \frac{0.01}{f^2}$
- initial subspace estimate satisfies $\mathrm{SE}(\hat{P}_0, P_0) \leq \Delta$,
  - use any RPCA method, e.g., PCP, for initial short batch of data

then, can track the subspace change with a delay at most $O(r \log n \log \frac{1}{\epsilon})$

Weakens standard RPCA assumptions by exploiting slow subspace change (see

Low Rank Matrix Recovery: Matrix Completion, Matrix Sennsing, Robust MC

Matrix Completion (MC) and Matrix Sensing I

- MC: Recover a low rank matrix $\boldsymbol{L}$ from a subset of its entries

$$\boldsymbol{y} := \mathcal{P}_{\Omega}(\boldsymbol{L})$$

  here Ω is the set of indices (i,j) of the matrix that are observed; $\mathcal{P}_{\Omega}$: means observe the entries of $\boldsymbol{L}$ which belong to the set Ω

- Matrix Sensing: Recover a low rank matrix $\boldsymbol{L}$ from linear projections of the matrix ("like Compressed Sensing for low-rank matrices")

$$\boldsymbol{y} := \mathcal{A}(\boldsymbol{L})$$

  here $\mathcal{A}(.)$: computes m linear projections of $\boldsymbol{L}$, i.e.,

$$\boldsymbol{y}_i =< \boldsymbol{A}_i, \boldsymbol{L} >:= \mathrm{Trace}(\boldsymbol{A}_i{}'\boldsymbol{L})$$

- Solutions
  - nuclear norm minimization

Matrix Completion (MC) and Matrix Sensing II

- alternating minimization with spectral init
- projected GD – IHT-like method with zero init
- ReProCS-MC – also solves dynamic MC : can develop ReProCS for MC by eliminating the support recovery step (see Lois and Vaswani, "Online Matrix Completion and Online Robust PCA", ISIT 2015)

- Can define low-rank-matrix-RIP: satisfied by certain kinds of matrix sensing problems; not satisfied by MC problem.
- Guarantees for MC
  - If the set Ω is generated Bernoulli(p), left and right singular vectors of $\boldsymbol{L}$ satisfy incoherence (denseness) with parameter μ, and p upper bounded (see paper)
    then whp, $\|\boldsymbol{L} - \hat{L}\|_F \leq \epsilon$

Matrix Completion (MC) and Matrix Sensing III

- Robust MC problem: Recover low rank matrix $\boldsymbol{L}$

$$\boldsymbol{y} := \mathcal{P}_\Omega(\boldsymbol{L} + X)$$

  here X is the sparse outlier matrix – this is also the “RPCA with missing entries” problem.

- solutions:
  - nuc norm
  - projected GD : Prateek Jain et al, Nearly Optimal Robust Matrix Completion
  - can also develop ReProCS-RMC : replace ell-1 min step by Modified-CS with support knowledge T being the set of missing entries at time t