Accelerated Outlier Detection in Low-Rank and Structured Data: Robust PCA and Extensions

HanQin Cai

Department of Mathematics
University of California at Los Angeles

Seminar on Applied Math and Data Science
March 25^{st}, 2020

Outline

Principal Component Analysis (PCA)

- Principal Component Analysis (PCA), a.k.a. the best low rank approximation:

$$\begin{aligned} \operatorname*{minimize}_{\boldsymbol{L}'} & \|\boldsymbol{D} - \boldsymbol{L}'\|_F \\ \text{subject to } & \operatorname{rank}(\boldsymbol{L}') \leq r \end{aligned}$$

  - Fundamental tool for dimension reduction

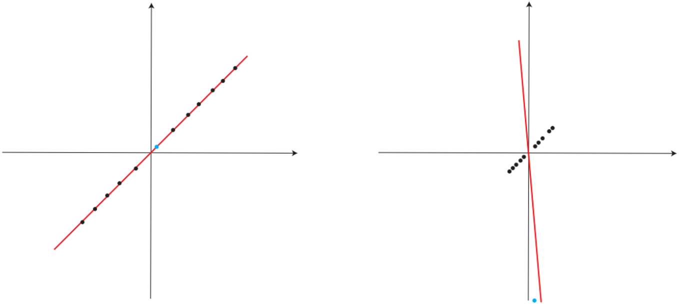
- Can be efficiently solved via truncated singular-value decomposition (SVD) $\mathcal{H}_r$:

$$\mathcal{H}_r(\boldsymbol{D}) = \mathcal{H}_r(\boldsymbol{U}\Sigma\boldsymbol{V}^T) = \boldsymbol{U}\Sigma_r\boldsymbol{V}^T$$

  where $\boldsymbol{\Sigma}_r$ keeps only the first r singular values.

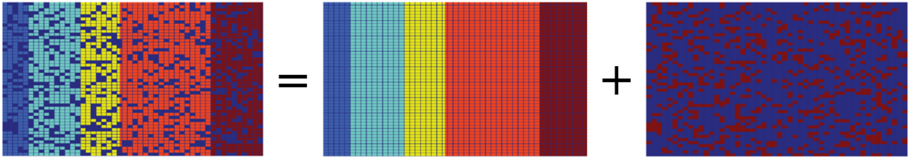
Principal Component Analysis (PCA)

- PCA, a.k.a. the best low rank approximation.
- Solved efficiently via truncated SVD $\mathcal{H}_r$.
- Tolerates small (zero-mean) noise, but over-sensitive to outliers.



Robust PCA

- Given $\boldsymbol{D} = \boldsymbol{L} + \boldsymbol{S} \in \mathbb{R}^{m \times n}$, where $\boldsymbol{L}$ is low rank and $\boldsymbol{S}$ is sparse.



Robust PCA

- Given $\boldsymbol{D} = \boldsymbol{L} + \boldsymbol{S} \in \mathbb{R}^{m \times n}$, where $\boldsymbol{L}$ is low rank and $\boldsymbol{S}$ is sparse.

- Split/recover $\boldsymbol{L}$, $\boldsymbol{S}$ from $\boldsymbol{D}$:

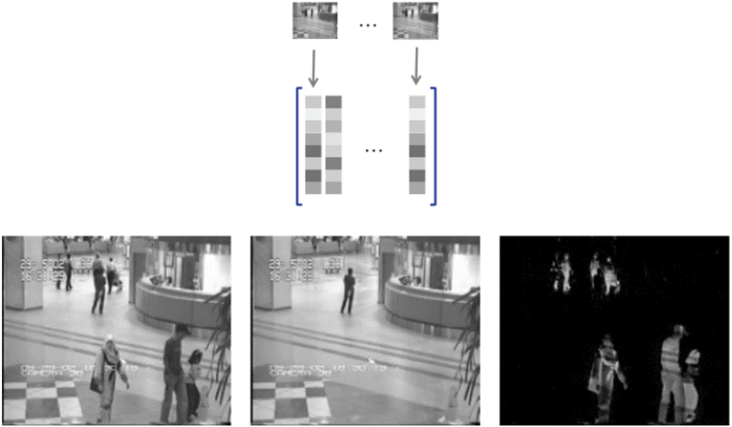
$$\begin{aligned} &\underset{\boldsymbol{L}', \boldsymbol{S}'}{\text{minimize}} \, \|\boldsymbol{D} - \boldsymbol{L}' - \boldsymbol{S}'\|_F \\ &\text{subject to } \operatorname{rank}(\boldsymbol{L}') \le r, \|\boldsymbol{S}'\|_0 \le \alpha mn \end{aligned} \tag{1}$$

  where α is the sparsity level of $\boldsymbol{S}$, compared to its size.
  - Non-convex problem

- Tolerates outliers better.

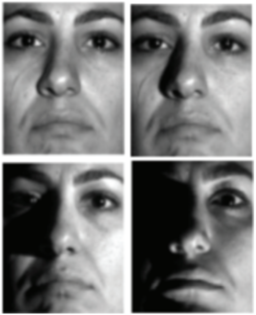
Applications

- Video background subtraction with static background



Applications

- Video background subtraction with static background
- Face recognition

Applications

- Video background subtraction with static background
- Face recognition
- System identification
- Fault isolation
- Netflix problem
- and more ...

RPCA is Ill-Posed without Assumptions

Cannot have a unique decomposition if $\boldsymbol{D}$ is both low rank and sparse.

$$\begin{aligned}
\begin{bmatrix} 0 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 0 \end{bmatrix} &= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\
&= \cdots
\end{aligned}$$

Assumption A1 for Uniqueness of Solution

- $\boldsymbol{L} \in \mathbb{R}^{m \times n}$ is rank-r with μ-incoherence, i.e.,

$$\max_i \|\boldsymbol{e}_i^T \boldsymbol{U}\|_2 \leq \sqrt{\frac{\mu r}{m}}$$
$$\max_j \|\boldsymbol{e}_j^T \boldsymbol{V}\|_2 \leq \sqrt{\frac{\mu r}{n}}$$

  where $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ is the SVD of $\boldsymbol{L}$.

- $\mu \in [1, \max\{m, n\}/r]$
- Smaller μ is better, which implies the entries of $\boldsymbol{L}$ are not too spiky/sparse. - Energy is evenly distributed among the entries of $\boldsymbol{L}$.

Assumption A2 for Uniqueness of Solution

- $\boldsymbol{S} \in \mathbb{R}^{m\times n}$ is α-sparse, i.e., $\boldsymbol{S}$ has at most αn non-zero entries in each row, and at most αm non-zero entries in each column.
- Non-zero entries are not locally dense.
- Satisfied with high probability if the support of $\boldsymbol{S}$ is drawn by some stochastic methods.
  - Bernoulli process
  - Uniform sampling

Outline

Outline

Convex Relaxation

- Consider convex Principal Component Pursuit (PCP):

$$\begin{aligned} &\underset{\boldsymbol{L}',\boldsymbol{S}'}{\text{minimize}}\, \|\boldsymbol{L}'\|_* + \lambda \|\boldsymbol{S}'\|_1 \\ &\text{subject to } \boldsymbol{L}' + \boldsymbol{S}' = \boldsymbol{D} \end{aligned} \tag{2}$$

- Nuclear norm term for low rank
  - $\|\boldsymbol{L}\|_* = \sum_{i=1}^r \sigma_i(\boldsymbol{L})$
- Vector ℓ_1-norm term for sparsity, where $\boldsymbol{S}$ is viewed as a long vector
  - $\|\boldsymbol{S}\|_1 = \sum_{ij} |S_{ij}|$

Recovery Guarantee of PCP

If $\lambda = \sqrt{1/\max\{m, n\}}$ is chosen, then, with high probability, the solution of convex PCP

$$\begin{aligned} &\underset{\boldsymbol{L}',\boldsymbol{S}'}{\text{minimize}}\, \|\boldsymbol{L}'\|_* + \lambda \|\boldsymbol{S}'\|_1 \\ &\text{subject to } \boldsymbol{L}' + \boldsymbol{S}' = \boldsymbol{D} \end{aligned}$$

is exact the solution of original non-convex Robust PCA problem

$$\begin{aligned} &\underset{\boldsymbol{L}',\boldsymbol{S}'}{\text{minimize}}\, \|\boldsymbol{D} - \boldsymbol{L}' - \boldsymbol{S}'\|_F \\ &\text{subject to } \operatorname{rank}(\boldsymbol{L}') \leq r, \|\boldsymbol{S}'\|_0 \leq \alpha mn \end{aligned}$$

under some natural conditions. [Candès, Li, Ma, Wright, 2009]

Algorithms for PCP

- Off-the-shelf solver
  - CVX package for Matlab
    - Based on semidefinite programming
    - $O(n^6)$ - expensive
    - Cannot handle problem larger than 200×200 dimension

Algorithms for PCP

- Off-the-shelf solver
  - CVX package for Matlab
    - Based on semidefinite programming
    - $O(n^6)$ - expensive
    - Cannot handle problem larger than 200×200 dimension

- Alternating direction method of multipliers (ADMM)
  - Consider the augmented Lagrangian

$$\ell(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{Y}) = \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_1 + \langle \boldsymbol{Y}, \boldsymbol{D} - \boldsymbol{L} - \boldsymbol{S} \rangle + \frac{\eta}{2}\|\boldsymbol{D} - \boldsymbol{L} - \boldsymbol{S}\|_F^2$$

  - Updating $\boldsymbol{L}$, $\boldsymbol{S}$ and dual variable $\boldsymbol{Y}$ alternatively

Algorithms for PCP

- Off-the-shelf solver
  - CVX package for Matlab
    - Based on semidefinite programming
    - $O(n^6)$ - expensive
    - Cannot handle problem larger than 200×200 dimension

- Alternating direction method of multipliers (ADMM)
  - Consider the augmented Lagrangian

$$\ell(\boldsymbol{L}, \boldsymbol{S}, \boldsymbol{Y}) = \|\boldsymbol{L}\|_* + \lambda\|\boldsymbol{S}\|_1 + \langle \boldsymbol{Y}, \boldsymbol{D} - \boldsymbol{L} - \boldsymbol{S}\rangle + \frac{\eta}{2}\|\boldsymbol{D} - \boldsymbol{L} - \boldsymbol{S}\|_F^2$$

  - Updating $\boldsymbol{L}$, $\boldsymbol{S}$ and dual variable $\boldsymbol{Y}$ alternatively
  - Convergence of ADMM has been well studied
  - Good per-iteration complexity, but sublinear convergence rate

Outline

Gradient Descent (GD)

- Updating $\boldsymbol{L}$:
  - Consider $\boldsymbol{L} = \boldsymbol{U}\Sigma\boldsymbol{V}^T = (\boldsymbol{U}\Sigma^{1/2})(\Sigma^{1/2}\boldsymbol{V}^T) := \boldsymbol{P}\boldsymbol{Q}^T$
  - Gradient descent on $\boldsymbol{P}$, $\boldsymbol{Q}$ separately, based on objective function:

$$\underbrace{\frac{1}{2}\|\boldsymbol{P}\boldsymbol{Q}^T + \boldsymbol{S} - \boldsymbol{D}\|_F^2}_{\text{loss function}} + \underbrace{\frac{1}{8}\|\boldsymbol{P}^T\boldsymbol{P} - \boldsymbol{Q}^T\boldsymbol{Q}\|_F^2}_{\text{keep scale being close}} \tag{3}$$

  - Enforce incoherence on $\boldsymbol{P}$, $\boldsymbol{Q}$ after gradient descent

- Updating $\boldsymbol{S}$:
  - Sparsification operator $\mathcal{F}_\alpha$
    - Keeps only the largest α-fraction elements per row and column
    - Partial sorting on each row and column - expensive with larger α

[Yi, Park, Chen, Caramanis, 2016]

Recovery Guarantee of GD

- The output $(\boldsymbol{P}_k, \boldsymbol{Q}_k)$ of GD satisfies

$$\|\boldsymbol{P}_k \boldsymbol{Q}_k^T - \boldsymbol{L}\|_F \leq \varepsilon$$

  in $O(\kappa \log(1/\varepsilon))$ iterations if followings are satisfied:
  - $\boldsymbol{L}$ is μ-incoherent
  - $\boldsymbol{S}$ is α-sparse with $\alpha \leq O\left(\min\left\{\frac{1}{\mu r^{1.5}\kappa^{1.5}}, \frac{1}{\mu r \kappa^2}\right\}\right)$
  - $\gamma = 2$: parameter for sparsification
  - $\eta \leq 1/36\sigma_1^L$: step size for gradient descent

- Linear convergence with rate of $1 - O(1/\kappa)$.

κ: condition number of $\boldsymbol{L}$

Alternating Projections (AltProj)

Consider two sets:

- $\mathcal{M}_r = \{\boldsymbol{L} \in \mathbb{R}^{m\times n} \mid \mathrm{rank}(\boldsymbol{L}) \le r\}$
- $\mathcal{S}_\alpha = \{\boldsymbol{S} \in \mathbb{R}^{m\times n} \mid \boldsymbol{S} \text{ is } \alpha\text{-sparse}\}$

Alternating Projections (AltProj)

Consider two sets:

- $\mathcal{M}_r = \{\boldsymbol{L} \in \mathbb{R}^{m \times n} \mid \mathrm{rank}(\boldsymbol{L}) \leq r\}$
- $\mathcal{S}_\alpha = \{\boldsymbol{S} \in \mathbb{R}^{m \times n} \mid \boldsymbol{S} \text{ is } \alpha\text{-sparse}\}$
- Find $\{(\boldsymbol{L}, \boldsymbol{S}) \mid \boldsymbol{L} \in \mathcal{M}_r, \boldsymbol{S} \in \mathcal{S}_\alpha \text{ and } \boldsymbol{L} + \boldsymbol{S} = \boldsymbol{D}\}$

Alternating Projections (AltProj)

Consider two sets:

- $\mathcal{M}_r = \{\boldsymbol{L} \in \mathbb{R}^{m \times n} \mid \mathrm{rank}(\boldsymbol{L}) \leq r\}$
- $\mathcal{S}_\alpha = \{\boldsymbol{S} \in \mathbb{R}^{m \times n} \mid \boldsymbol{S} \text{ is } \alpha\text{-sparse}\}$
- Find $\{(\boldsymbol{L}, \boldsymbol{S}) \mid \boldsymbol{L} \in \mathcal{M}_r, \boldsymbol{S} \in \mathcal{S}_\alpha \text{ and } \boldsymbol{L} + \boldsymbol{S} = \boldsymbol{D}\}$

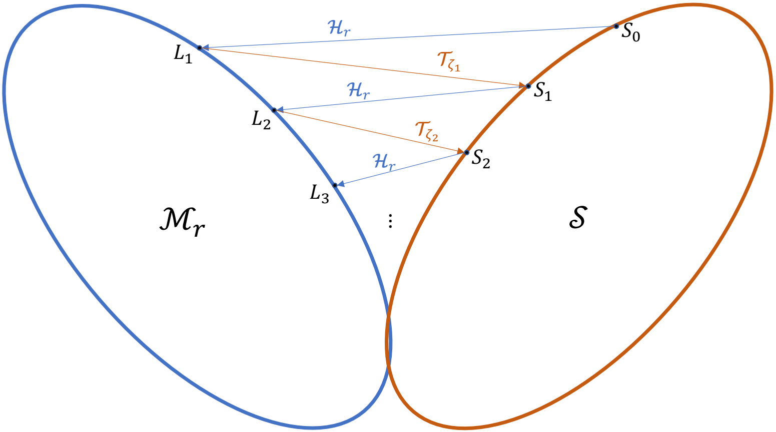
Updating:

- $\boldsymbol{L}_{new} = \mathcal{H}_r(\boldsymbol{D} - \boldsymbol{S}_{old})$.
  - $\mathcal{H}_r$: truncated SVD - expensive

- $\boldsymbol{S}_{new} = \mathcal{T}_\zeta(\boldsymbol{D} - \boldsymbol{L}_{new})$.
  - $\mathcal{T}_\zeta(x) = x \cdot \mathbb{1}_{\{|x| > \zeta\}}$: hard-thresholding
  - Choose ζ such that $\mathrm{supp}(\boldsymbol{S}_{new}) \subset \mathrm{supp}(\boldsymbol{S})$

[Netrapalli, Niranjan, Sanghavi, Anandkumar, Jain, 2014]

Illustration of Alternating Projections



Alternating Projections (AltProj)

Consider two sets:

- $\mathcal{M}_r = \{\boldsymbol{L} \in \mathbb{R}^{m \times n} \mid \mathrm{rank}(\boldsymbol{L}) \leq r\}$
- $\mathcal{S}_\alpha = \{\boldsymbol{S} \in \mathbb{R}^{m \times n} \mid \boldsymbol{S} \text{ is } \alpha\text{-sparse}\}$
- Find $\{(\boldsymbol{L}, \boldsymbol{S}) \mid \boldsymbol{L} \in \mathcal{M}_r, \boldsymbol{S} \in \mathcal{S}_\alpha \text{ and } \boldsymbol{L} + \boldsymbol{S} = \boldsymbol{D}\}$

Updating:

- $\boldsymbol{L}_{new} = \mathcal{H}_r(\boldsymbol{D} - \boldsymbol{S}_{old}); \qquad \boldsymbol{S}_{new} = \mathcal{T}_\zeta(\boldsymbol{D} - \boldsymbol{L}_{new}).$

Break algorithm into r stages:

- At the t^{th} stage, use $\mathcal{H}_t$ instead of $\mathcal{H}_r$
- Overcome the case of bad condition number of $\boldsymbol{L}$

[Netrapalli, Niranjan, Sanghavi, Anandkumar, Jain, 2014]

Recovery Guarantee of AltProj

- The output $(\boldsymbol{L}_{t,k}, \boldsymbol{S}_{t,k})$ of AltProj satisfies

$$\|\boldsymbol{L}_{t,k} - \boldsymbol{L}\|_F \leq \varepsilon, \quad \|\boldsymbol{S}_{t,k} - \boldsymbol{S}\|_\infty \leq \varepsilon/\sqrt{mn},$$
$$\text{and } \mathrm{supp}(\boldsymbol{S}_{t,k}) \subset \mathrm{supp}(\boldsymbol{S})$$

  in $O(t \log_{\frac{1}{2}} \varepsilon)$ iterations if followings are satisfied:
  - $\boldsymbol{L}$ is μ-incoherent
  - $\boldsymbol{S}$ is α-sparse with $\alpha \leq \frac{1}{512\mu r}$
  - $\beta = 4\mu r/\sqrt{mn}$: parameter for thresholding

- Linear convergence with rate of $1/2$, at t^{th} stage.

Outline

Outline

Key Idea

- AltProj is fast enough when updating $\boldsymbol{S}$ with hard thresholding, but using truncated SVD for $\boldsymbol{L}$ updating can still be very expensive when the problem size is larger.

- Accelerating by first projecting $\boldsymbol{D} - \boldsymbol{S}_{old}$ onto some local low dimensional subspace before obtaining a new estimate of $\boldsymbol{L}$ via truncated SVD.

The Low Dimensional Subspace

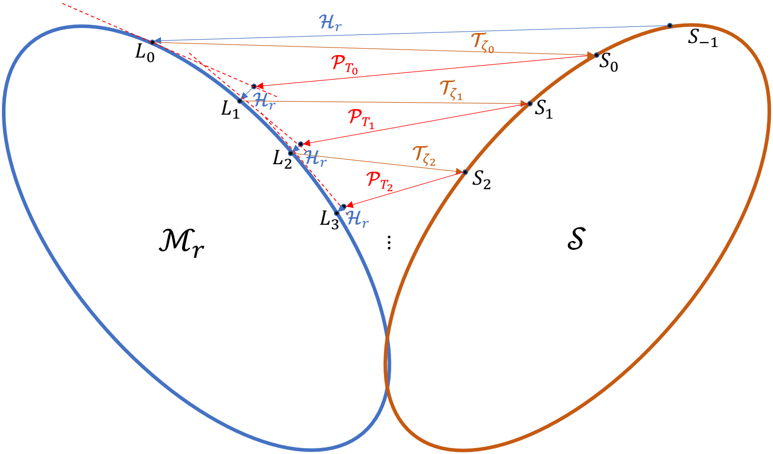
- $\mathcal{M}_r = \{\boldsymbol{L} \in \mathbb{R}^{m\times n} \mid \mathrm{rank}(\boldsymbol{L}) \le r\}$ is a Riemannian manifold.
- Tangent space of $\mathcal{M}_r$ at $\boldsymbol{L}$

$$T = \{\boldsymbol{U}\boldsymbol{A}^T + \boldsymbol{B}\boldsymbol{V}^T \mid \boldsymbol{A} \in \mathbb{R}^{n\times r}, \boldsymbol{B} \in \mathbb{R}^{m\times r}\}$$

  where $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ is the SVD of $\boldsymbol{L}$.
  - The low dimensional subspace we want, but don't have in practice.
  - Instead, use tangent space of $\mathcal{M}_r$ at $\boldsymbol{L}_{old}$.
- Project an arbitrary matrix onto the tangent space T:

$$\mathcal{P}_T\boldsymbol{Z} = \boldsymbol{U}\boldsymbol{U}^T\boldsymbol{Z} + \boldsymbol{Z}\boldsymbol{V}\boldsymbol{V}^T - \boldsymbol{U}\boldsymbol{U}^T\boldsymbol{Z}\boldsymbol{V}\boldsymbol{V}^T \tag{4}$$

Illustration of Our Approach



Algorithm: AccAltProj

RPCA by Accelerated Alternating Projections (AccAltProj)

Initialization

$k = 0$

while $\|\boldsymbol{D} - \boldsymbol{L}_k - \boldsymbol{S}_k\|_F / \|\boldsymbol{D}\|_F \geq \varepsilon$ **do**

$\widetilde{\boldsymbol{L}}_k = \mathrm{Trim}\left(\boldsymbol{L}_k, \mu\right)$ // Enforce Incoherence of Tangent Space

$\boldsymbol{L}_{k+1} = \mathcal{H}_r\left(\mathcal{P}_{\widetilde{T}_k}(\boldsymbol{D} - \boldsymbol{S}_k)\right)$

$\zeta_{k+1} = \beta\left(\sigma_{r+1}\left(\mathcal{P}_{\widetilde{T}_k}(\boldsymbol{D} - \boldsymbol{S}_k)\right) + \gamma^{k+1}\sigma_1\left(\mathcal{P}_{\widetilde{T}_k}(\boldsymbol{D} - \boldsymbol{S}_k)\right)\right)$

$\boldsymbol{S}_{k+1} = \mathcal{T}_{\zeta_{k+1}}(\boldsymbol{D} - \boldsymbol{L}_{k+1})$

$k = k + 1$

end while

Accelerating Truncated SVD

Denote $\boldsymbol{Z} := \boldsymbol{D} - \boldsymbol{S}_k$. Computing $\mathcal{H}_r(\mathcal{P}_{\widetilde{T}_k}\boldsymbol{Z})$ efficiently:

- $\mathcal{P}_{\widetilde{T}_k}\boldsymbol{Z} = \underbrace{\begin{bmatrix} \widetilde{\boldsymbol{U}}_k & \boldsymbol{Q}_1 \end{bmatrix}}_{\text{orthogonal}} \underbrace{\begin{bmatrix} \widetilde{\boldsymbol{U}}_k^T \boldsymbol{Z} \widetilde{\boldsymbol{V}}_k & \boldsymbol{R}_2^T \\ \boldsymbol{R}_1 & 0 \end{bmatrix}}_{2r\times 2r} \underbrace{\begin{bmatrix} \widetilde{\boldsymbol{V}}_k & \boldsymbol{Q}_2 \end{bmatrix}^T}_{\text{orthogonal}}$
  - $\boldsymbol{Q}_1\boldsymbol{R}_1 = (\boldsymbol{I} - \widetilde{\boldsymbol{U}}_k\widetilde{\boldsymbol{U}}_k^T)\boldsymbol{Z}\widetilde{\boldsymbol{V}}_k \in \mathbb{R}^{m\times r}$
  - $\boldsymbol{Q}_2\boldsymbol{R}_2 = (\boldsymbol{I} - \widetilde{\boldsymbol{V}}_k\widetilde{\boldsymbol{V}}_k^T)\boldsymbol{Z}\widetilde{\boldsymbol{U}}_k \in \mathbb{R}^{n\times r}$

Accelerating Truncated SVD

Denote $\boldsymbol{Z} := \boldsymbol{D} - \boldsymbol{S}_k$. Computing $\mathcal{H}_r(\mathcal{P}_{\widetilde{T}_k}\boldsymbol{Z})$ efficiently:

- $\mathcal{P}_{\widetilde{T}_k}\boldsymbol{Z} = \underbrace{\begin{bmatrix} \widetilde{\boldsymbol{U}}_k & \boldsymbol{Q}_1 \end{bmatrix}}_{\text{orthogonal}} \underbrace{\begin{bmatrix} \widetilde{\boldsymbol{U}}_k^T \boldsymbol{Z} \widetilde{\boldsymbol{V}}_k & \boldsymbol{R}_2^T \\ \boldsymbol{R}_1 & 0 \end{bmatrix}}_{2r\times 2r} \underbrace{\begin{bmatrix} \widetilde{\boldsymbol{V}}_k & \boldsymbol{Q}_2 \end{bmatrix}^T}_{\text{orthogonal}}$
  - $\boldsymbol{Q}_1\boldsymbol{R}_1 = (\boldsymbol{I} - \widetilde{\boldsymbol{U}}_k\widetilde{\boldsymbol{U}}_k^T)\boldsymbol{Z}\widetilde{\boldsymbol{V}}_k \in \mathbb{R}^{m\times r}$
  - $\boldsymbol{Q}_2\boldsymbol{R}_2 = (\boldsymbol{I} - \widetilde{\boldsymbol{V}}_k\widetilde{\boldsymbol{V}}_k^T)\boldsymbol{Z}\widetilde{\boldsymbol{U}}_k \in \mathbb{R}^{n\times r}$
- $\mathcal{H}_r(\mathcal{P}_{\widetilde{T}_k}\boldsymbol{Z}) = \begin{bmatrix} \widetilde{\boldsymbol{U}}_k & \boldsymbol{Q}_1 \end{bmatrix} \mathcal{H}_r\left(\begin{bmatrix} \widetilde{\boldsymbol{U}}_k^T \boldsymbol{Z} \widetilde{\boldsymbol{V}}_k & \boldsymbol{R}_2^T \\ \boldsymbol{R}_1 & 0 \end{bmatrix}\right) \begin{bmatrix} \widetilde{\boldsymbol{V}}_k & \boldsymbol{Q}_2 \end{bmatrix}^T$
- Only need a SVD of $2r \times 2r$ matrix + two QR-decompositions

Accelerating Truncated SVD

Denote $\boldsymbol{Z} := \boldsymbol{D} - \boldsymbol{S}_k$. Computing $\mathcal{H}_r(\mathcal{P}_{\widetilde{T}_k}\boldsymbol{Z})$ efficiently:

- $$\mathcal{P}_{\widetilde{T}_k}\boldsymbol{Z} = \underbrace{\begin{bmatrix} \widetilde{\boldsymbol{U}}_k & \boldsymbol{Q}_1 \end{bmatrix}}_{\text{orthogonal}} \underbrace{\begin{bmatrix} \widetilde{\boldsymbol{U}}_k^T\boldsymbol{Z}\widetilde{\boldsymbol{V}}_k & \boldsymbol{R}_2^T \\ \boldsymbol{R}_1 & 0 \end{bmatrix}}_{2r\times 2r} \underbrace{\begin{bmatrix} \widetilde{\boldsymbol{V}}_k & \boldsymbol{Q}_2 \end{bmatrix}^T}_{\text{orthogonal}}$$
  - $\boldsymbol{Q}_1\boldsymbol{R}_1 = (\boldsymbol{I} - \widetilde{\boldsymbol{U}}_k\widetilde{\boldsymbol{U}}_k^T)\boldsymbol{Z}\widetilde{\boldsymbol{V}}_k \in \mathbb{R}^{m\times r}$
  - $\boldsymbol{Q}_2\boldsymbol{R}_2 = (\boldsymbol{I} - \widetilde{\boldsymbol{V}}_k\widetilde{\boldsymbol{V}}_k^T)\boldsymbol{Z}\widetilde{\boldsymbol{U}}_k \in \mathbb{R}^{n\times r}$
- $$\mathcal{H}_r(\mathcal{P}_{\widetilde{T}_k}\boldsymbol{Z}) = \begin{bmatrix} \widetilde{\boldsymbol{U}}_k & \boldsymbol{Q}_1 \end{bmatrix} \mathcal{H}_r\left(\begin{bmatrix} \widetilde{\boldsymbol{U}}_k^T\boldsymbol{Z}\widetilde{\boldsymbol{V}}_k & \boldsymbol{R}_2^T \\ \boldsymbol{R}_1 & 0 \end{bmatrix}\right) \begin{bmatrix} \widetilde{\boldsymbol{V}}_k & \boldsymbol{Q}_2 \end{bmatrix}^T$$
- Only need a SVD of $2r \times 2r$ matrix + two QR-decompositions
- Complexities: $4n^2r + n^2 + O(nr^2 + r^3)$ $[\mathcal{H}_r\mathcal{P}_{\widetilde{T}_k}]$

  $O(n^2r)$ with large hidden constant $[\mathcal{H}_r]$

Efficient S Updating with Hard Thresholding

- Updating $\boldsymbol{S}_{k+1}$ with $\mathcal{T}_{\zeta_{k+1}}(\boldsymbol{D}-\boldsymbol{L}_{k+1})$, where

$$\zeta_{k+1}=\beta\left(\sigma_{r+1}\left(\mathcal{P}_{\widetilde{T}_k}(\boldsymbol{D}-\boldsymbol{S}_k)\right)+\gamma^{k+1}\sigma_1\left(\mathcal{P}_{\widetilde{T}_k}(\boldsymbol{D}-\boldsymbol{S}_k)\right)\right),$$

so that we have $\mathrm{supp}(\boldsymbol{S}_{k+1})\subset\mathrm{supp}(\boldsymbol{S})$ in theory.

Efficient $\boldsymbol{S}$ Updating with Hard Thresholding

- Updating $\boldsymbol{S}_{k+1}$ with $\mathcal{T}_{\zeta_{k+1}}(\boldsymbol{D}-\boldsymbol{L}_{k+1})$, where

$$\zeta_{k+1}=\beta\left(\sigma_{r+1}\left(\mathcal{P}_{\widetilde{T}_k}(\boldsymbol{D}-\boldsymbol{S}_k)\right)+\gamma^{k+1}\sigma_1\left(\mathcal{P}_{\widetilde{T}_k}(\boldsymbol{D}-\boldsymbol{S}_k)\right)\right),$$

  so that we have $\operatorname{supp}(\boldsymbol{S}_{k+1})\subset\operatorname{supp}(\boldsymbol{S})$ in theory.

- The singular values of $\mathcal{P}_{\widetilde{T}_k}(\boldsymbol{D}-\boldsymbol{S}_k)$ were already computed in $\boldsymbol{L}$ updating.
  - Cost $O(1)$ for computing ζ_{k+1}

Efficient S Updating with Hard Thresholding

- Updating $\boldsymbol{S}_{k+1}$ with $\mathcal{T}_{\zeta_{k+1}}(\boldsymbol{D} - \boldsymbol{L}_{k+1})$, where

$$\zeta_{k+1} = \beta \left(\sigma_{r+1} \left(\mathcal{P}_{\widetilde{T}_k}(\boldsymbol{D} - \boldsymbol{S}_k) \right) + \gamma^{k+1} \sigma_1 \left(\mathcal{P}_{\widetilde{T}_k}(\boldsymbol{D} - \boldsymbol{S}_k) \right) \right),$$

  so that we have $\mathrm{supp}(\boldsymbol{S}_{k+1}) \subset \mathrm{supp}(\boldsymbol{S})$ in theory.

- The singular values of $\mathcal{P}_{\widetilde{T}_k}(\boldsymbol{D} - \boldsymbol{S}_k)$ were already computed in $\boldsymbol{L}$ updating.
  - Cost $O(1)$ for computing ζ_{k+1}

- Complexity: $2n^2 + O(1)$
  - No expensive partial sorting needed.

Initialization

- If choose $\boldsymbol{L}_0 = 0$, then $\boldsymbol{U}_0 = \boldsymbol{V}_0 = 0$.
  - We don't want this happen.

Initialization

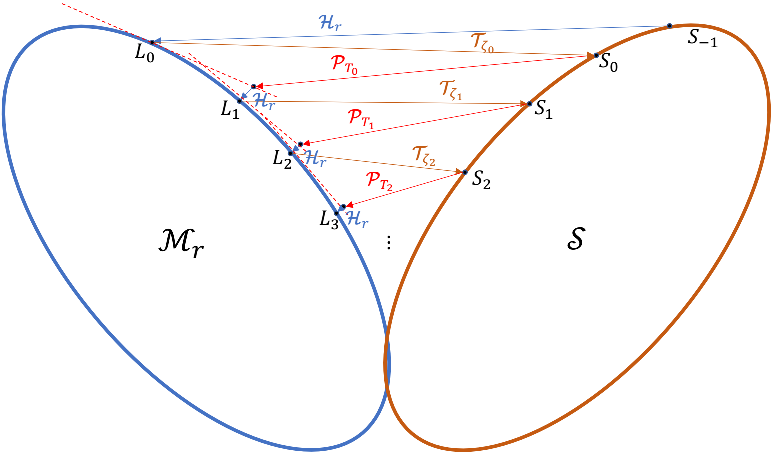
- If choose $\boldsymbol{L}_0 = 0$, then $\boldsymbol{U}_0 = \boldsymbol{V}_0 = 0$.
  - We don't want this happen.

Initialization by Two Steps of AltProj

$$\boldsymbol{L}_{-1} = 0$$
$$\zeta_{-1} = \beta_{init} \cdot \sigma_1(\boldsymbol{D})$$
$$\boldsymbol{S}_{-1} = \mathcal{T}_{\zeta_{-1}}(\boldsymbol{D} - \boldsymbol{L}_{-1})$$
$$\boldsymbol{L}_0 = \mathcal{H}_r(\boldsymbol{D} - \boldsymbol{S}_{-1})$$
$$\zeta_0 = \beta \cdot \sigma_1(\boldsymbol{D} - \boldsymbol{S}_{-1})$$
$$\boldsymbol{S}_0 = \mathcal{T}_{\zeta_0}(\boldsymbol{D} - \boldsymbol{L}_0)$$

- Two steps of AltProj give us a close enough tangent space to start with.

Illustration of Our Approach



Outline

Main Theoretical Result

Recovery Guarantee of AccAltProj

Let $\boldsymbol{L}$ and $\boldsymbol{S}$ be two matrices satisfying Assumptions A1 and A2 with

$$\alpha \leq O\left(\min\left\{\frac{1}{\mu r^2\kappa^3}, \frac{1}{\mu^{1.5} r^2\kappa}, \frac{1}{\mu^2 r^2}\right\}\right).$$

If the thresholding parameters obey $\frac{\mu r \sigma_1^L}{\sqrt{mn}\sigma_1^D} \leq \beta_{init} \leq \frac{3\mu r \sigma_1^L}{\sqrt{mn}\sigma_1^D}$ and $\beta = \frac{\mu r}{2\sqrt{mn}}$, alone with the convergence rate parameter $\gamma \in (\frac{1}{\sqrt{12}}, 1)$, then the outputs of AccAltProj satisfy

$$\|\boldsymbol{L} - \boldsymbol{L}_k\|_F \leq \varepsilon\sigma_1^L, \|\boldsymbol{S} - \boldsymbol{S}_k\|_\infty \leq \frac{\varepsilon}{\sqrt{mn}}\sigma_1^L, \text{ and } \operatorname{supp}(\boldsymbol{S}_k) \subset \operatorname{supp}(\boldsymbol{S})$$

in $O(\log_\gamma \varepsilon)$ iterations.

- Linear convergence with rate of γ.

Proof Sketch

1. Show any $m \times n$ RPCA problem can be reduced to a symmetric RPCA problem.

Proof Sketch

A. Show any $m \times n$ RPCA problem can be reduced to a symmetric RPCA problem.
B. Prove the symmetric case by mathematical induction:

Proof Sketch

A. Show any $m \times n$ RPCA problem can be reduced to a symmetric RPCA problem.

B. Prove the symmetric case by mathematical induction:
   - Base case: Prove initialization by 2-step AltPorj satisfies

$$\|\boldsymbol{L}_0 - \boldsymbol{L}\|_2 \leq 8\alpha\mu r\sigma_1^L, \|\boldsymbol{S}_0 - \boldsymbol{S}\|_\infty \leq \frac{\mu r}{n}\sigma_1^L, \text{ and } \mathrm{supp}(\boldsymbol{S}_0) \subset \mathrm{supp}(\boldsymbol{S}).$$

Proof Sketch

A. Show any $m \times n$ RPCA problem can be reduced to a symmetric RPCA problem.

B. Prove the symmetric case by mathematical induction:
   - Base case: Prove initialization by 2-step AltPorj satisfies

   $$\|\boldsymbol{L}_0 - \boldsymbol{L}\|_2 \leq 8\alpha\mu r\sigma_1^L, \|\boldsymbol{S}_0 - \boldsymbol{S}\|_\infty \leq \frac{\mu r}{n}\sigma_1^L, \text{ and } \mathrm{supp}(\boldsymbol{S}_0) \subset \mathrm{supp}(\boldsymbol{S}).$$

   - Induction Step: Prove following two lemmas
     1. If $\|\boldsymbol{L}_k - \boldsymbol{L}\|_2 \leq 8\alpha\mu r\gamma^k\sigma_1^L$, $\|\boldsymbol{S}_k - \boldsymbol{S}\|_\infty \leq \frac{\mu r}{n}\gamma^k\sigma_1^L$, and $\mathrm{supp}(\boldsymbol{S}_k) \subset \mathrm{supp}(\boldsymbol{S})$, then
     $\|\boldsymbol{L}_{k+1} - \boldsymbol{L}\|_2 \leq 8\alpha\mu r\gamma^{k+1}\sigma_1^L$ and
     $\|\boldsymbol{L}_{k+1} - \boldsymbol{L}\|_\infty \leq \frac{(1-8\alpha\mu r\kappa)\mu r}{2n}\gamma^{k+1}\sigma_1^L$.

Proof Sketch

1. Show any $m \times n$ RPCA problem can be reduced to a symmetric RPCA problem.
2. Prove the symmetric case by mathematical induction:
   - Base case: Prove initialization by 2-step AltPorj satisfies

$$\|\boldsymbol{L}_0 - \boldsymbol{L}\|_2 \leq 8\alpha\mu r\sigma_1^L, \|\boldsymbol{S}_0 - \boldsymbol{S}\|_\infty \leq \frac{\mu r}{n}\sigma_1^L, \text{ and } \mathrm{supp}(\boldsymbol{S}_0) \subset \mathrm{supp}(\boldsymbol{S}).$$

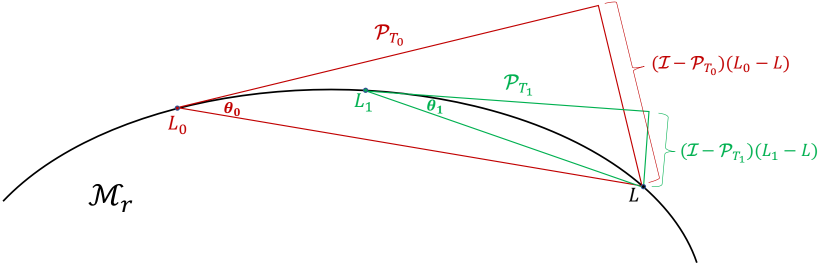
   - Induction Step: Prove following two lemmas
     1. If $\|\boldsymbol{L}_k - \boldsymbol{L}\|_2 \leq 8\alpha\mu r\gamma^k\sigma_1^L$, $\|\boldsymbol{S}_k - \boldsymbol{S}\|_\infty \leq \frac{\mu r}{n}\gamma^k\sigma_1^L$, and $\mathrm{supp}(\boldsymbol{S}_k) \subset \mathrm{supp}(\boldsymbol{S})$, then $\|\boldsymbol{L}_{k+1} - \boldsymbol{L}\|_2 \leq 8\alpha\mu r\gamma^{k+1}\sigma_1^L$ and $\|\boldsymbol{L}_{k+1} - \boldsymbol{L}\|_\infty \leq \frac{(1-8\alpha\mu r\kappa)\mu r}{2n}\gamma^{k+1}\sigma_1^L$.
     2. If $\|\boldsymbol{L}_{k+1} - \boldsymbol{L}\|_\infty \leq \frac{(1-8\alpha\mu r\kappa)\mu r}{2n}\gamma^{k+1}\sigma_1^L$, then $\|\boldsymbol{S}_{k+1} - \boldsymbol{S}\|_\infty \leq \frac{\mu r}{n}\gamma^{k+1}\sigma_1^L$ and $\mathrm{supp}(\boldsymbol{S}_{k+1}) \subset \mathrm{supp}(\boldsymbol{S})$.

Key Properties of Tangent Space Projection (1)

- If $\boldsymbol{L}$, $\boldsymbol{L}_k$ are two rank-r matrices, then

$$\|(\mathcal{I}-\mathcal{P}_{T_k})(\boldsymbol{L}-\boldsymbol{L}_k)\|_2 \leq \frac{\|\boldsymbol{L}-\boldsymbol{L}_k\|_2^2}{\sigma_r^L}.$$

  – Make the local convergence analysis possible.



Key Properties of Tangent Space Projection (2)

- If $\boldsymbol{L}_k$ is μ-incoherent and $\boldsymbol{S}$ is α-sparse, then

$$\|\mathcal{P}_{T_k}\boldsymbol{S}\|_\infty \leq 3\alpha\mu r\|\boldsymbol{S}\|_\infty.$$

  - $\mathcal{P}_{T_k}\boldsymbol{S}$ is not sparse.
  - Ensures sparsity can still be used for error bound after tangent space projection.

Overview

Algorithm	Updating S	Updating L	Tolerance of p	Iterations needed
AccAltProj	$\boldsymbol{O(n^2)}$	$\boldsymbol{O(rn^2)}$	$O\left(\frac{1}{\max\{\mu r^2\kappa^3, \mu^{1.5}r^2\kappa, \mu^2 r^2\}}\right)$	$\boldsymbol{O(\log(\frac{1}{\epsilon}))}$
AltProj	$O(rn^2)$	$O(r^2n^2)$	$\boldsymbol{O\left(\frac{1}{\mu r}\right)}$	$\boldsymbol{O(\log(\frac{1}{\epsilon}))}$
GD	$O(n^2 + pn^2\log(pn))$	$\boldsymbol{O(rn^2)}$	$O\left(\frac{1}{\max\{\mu r^{1.5}\kappa^{1.5}, \mu r\kappa^2\}}\right)$	$O(\kappa\log(\frac{1}{\epsilon}))$

*complexities in table are based on $\boldsymbol{D} \in \mathbb{R}^{n\times n}$

We have the best per-iteration complexity, with the best order of convergence, in the class of provable non-convex Robust PCA algorithms.

Outline

Outline

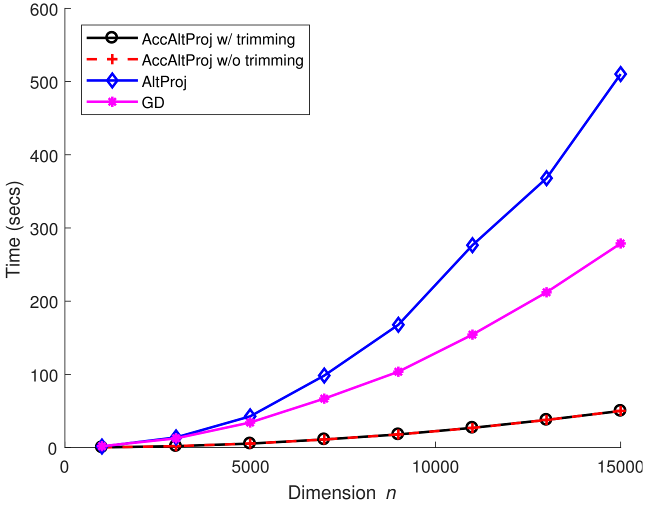
Setup

- $\boldsymbol{L} = \boldsymbol{P}\boldsymbol{Q}^T$, where $\boldsymbol{P}, \boldsymbol{Q} \in \mathbb{R}^{n \times r}$ are random matrices where their entries are drawn i.i.d. from standard normal distribution.
- $\mathrm{supp}(\boldsymbol{S})$ is sampled uniformly without replacement.
- The values of non-zero entries of $\boldsymbol{S}$ are drawn i.i.d from uniform distribution over the domain $[-c \cdot \mathbb{E}(|L_{ij}|), c \cdot \mathbb{E}(|L_{ij}|)]$.
  - Smaller c gives a harder splitting task (unless it becomes too small to be outliers, like 10^{-8})

Setup - Continue

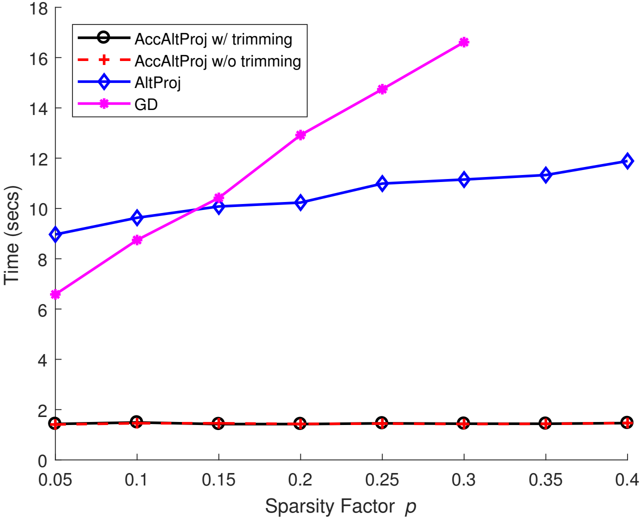
- Compare among three non-convex algorithms:
  - GD
    - ☐ Use 1.1α for sparsification $\mathcal{F}$
  - AltProj
    - ☐ Only time the r^{th} stage
  - AccAltProj with Trimming
    - ☐ Use 1.1μ for trimming
  - AccAltProj without Trimming
    - ☐ Skip the step $\widetilde{\boldsymbol{L}}_k = \mathrm{Trim}(\boldsymbol{L}_k, \mu)$
- PROPACK is used for large size truncated SVD
  - Good for AltProj which uses it every iteration

Speed Comparison: Dimension vs Runtime



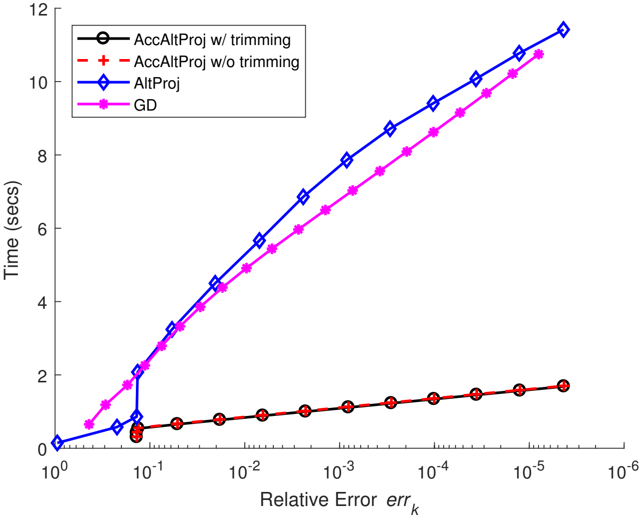
$r = 5, \alpha = 0.1, c = 1$, stop at $err_k < 10^{-4}$.

Speed Comparison: Sparsity vs Runtime



$r = 5, c = 1, n = 2500$, stop at $err_k < 10^{-4}$.

Speed Comparison: Relative Error vs Runtime



$r = 5, \alpha = 0.1, c = 1, n = 2500.$

Rate of Success for Varying c and α

$c = 0.2$	0.3	0.35	0.4	0.45	0.5	0.55	0.6
AccAltProj w/ trimming	10	10	10	10	10	10	10
AccAltProj w/o trimming	10	10	10	10	10	10	10
AltProj	10	10	10	10	10	10	10
GD	10	10	10	0	0	0	0

$c = 1$	0.3	0.35	0.4	0.45	0.5	0.55	0.6
AccAltProj w/ trimming	10	10	10	10	10	10	10
AccAltProj w/o trimming	10	10	10	10	10	10	10
AltProj	10	10	10	10	10	10	10
GD	10	10	10	10	9	0	0

$c = 5$	0.3	0.35	0.4	0.45	0.5	0.55	0.6
AccAltProj w/ trimming	10	10	10	10	10	10	10
AccAltProj w/o trimming	10	10	10	10	10	10	10
AltProj	10	10	10	10	10	10	10
GD	10	10	10	10	10	10	7

$r = 5$, $n = 2500$.

Outline

Setup

- Each frame in the video is vectorized, and becomes a row (or a column) in a data matrix.
- The static background is the low rank component of the matrix.
- The moving objects, i.e., foreground, forms the sparse components of the matrix.

Demo

https://youtu.be/k6uaeQky2sc

Comparison of Video Outputs

- Shoppingmall(S): 256×320 frame size; 1000 frames
- Restaurant(R): 120×160 frame size; 3055 frames

	AccAltProj w/ trim		AccAltProj w/o trim		AltProj		GD	
	runtime	μ	runtime	μ	runtime	μ	runtime	μ
S	$38.98s$	2.12	$38.79s$	2.26	$82.97s$	2.13	$161.1s$	2.85
R	$28.09s$	5.16	$27.94s$	5.25	$69.12s$	5.28	$107.3s$	6.07

Trimming helps the consistency a litter bit among the frames of the background, while only uses slightly more time.

AccAltProj has been published as:

Cai, HanQin, Jian-Feng Cai, and Ke Wei. "Accelerated alternating projections for robust principal component analysis." *The Journal of Machine Learning Research* 20.1 (2019): 685-717.

Code can be found at:

`https://github.com/caesarcai/AccAltProj_for_RPCA`

Outline

Form a Hankel Matrix

Given a complex vector $\boldsymbol{x} \in \mathbb{C}^n$, we can form a corresponding Hankel matrix $\mathcal{H}(\boldsymbol{x}) \in \mathbb{C}^{n_1 \times n_2}$ by

$$\mathcal{H}(\boldsymbol{x}) = \begin{bmatrix} x_0 & x_1 & x_2 & \cdots & x_{n_2-1} \\ x_1 & x_2 & x_3 & \cdots & x_{n_2} \\ x_2 & x_3 & x_4 & \cdots & x_{n_2+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{n_1-1} & x_{n_1} & x_{n_1+1} & \cdots & x_{n-1} \end{bmatrix}$$

where $n_1 + n_2 = n + 1$.

- We try to keep the matrix (almost) square: when n is odd, we choose $n_1 = n_2 = \frac{n+1}{2}$; when n is even, we choose $n_1 = \frac{n}{2} + 1$ and $n_2 = \frac{n}{2}$.

*We try to using consistent notation with the paper, which is unfortunately inconsistent with the notation in previous sections.

Low-Rank Hankel Matrix

Not all Hankel matrices are low-rank, but low-rank Hankel matrix appears in many applications:

- Spectrally sparse signal
  - Magnetic resonance image (MRI)
  - Nuclear magnetic resonance (NMR) spectroscopy
  - Fluorescence microscopy
  - Analog-to-digital conversion
  - Earthquake-induced vibration data
- Dynamic sampling
- Auto-regression

Hankel Outlier Detection Problem

- To recover the vector $\boldsymbol{x}$ from the corrupted observation $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{s} \in \mathbb{C}^n$ (or equivalently $\mathcal{H}(\boldsymbol{y}) = \mathcal{H}(\boldsymbol{x}) + \mathcal{H}(\boldsymbol{s}) \in \mathbb{C}^{n_1 \times n_2}$), we want to solve the minimization problem:

$$\begin{aligned} &\underset{\boldsymbol{x},\boldsymbol{s}}{\text{minimize}} \, \|\boldsymbol{y} - \boldsymbol{x} - \boldsymbol{s}\|_2 \\ &\text{subject to } \operatorname{rank}(\mathcal{H}(\boldsymbol{x})) \le r, \|\boldsymbol{s}\|_0 \le \alpha n \end{aligned} \tag{5}$$

- The sparse vector $\boldsymbol{s}$ can be:
  - Impulse noise in MRI/NMR signals.
  - Corrupted time snap in dynamic sampling
  - Black swan events on financial marketing

AAP-Hankel

Accelerated Alternating Projections for Hankel Recovery

Initialization, set $k = 0$

while $\|\boldsymbol{y} - \boldsymbol{x}_k - \boldsymbol{s}_k\| / \|\boldsymbol{z}\| \geq \epsilon$ **do**

$\widetilde{\boldsymbol{L}}_k = \mathrm{Trim}(\boldsymbol{L}_k, \mu)$

$\boldsymbol{L}_{k+1} = \mathcal{D}_r \mathcal{P}_{\widetilde{T}_k} \mathcal{H}(\boldsymbol{y} - \boldsymbol{s}_k)$

$\boldsymbol{x}_{k+1} = \mathcal{H}^{\dagger}(\boldsymbol{L}_{k+1})$

$\zeta_{k+1} = \beta \left(\sigma_{r+1} \left(\mathcal{P}_{\widetilde{T}_k} \mathcal{H}(\boldsymbol{z} - \boldsymbol{s}_k) \right) + \gamma^{k+1} \sigma_1 \left(\mathcal{P}_{\widetilde{T}_k} \mathcal{H}(\boldsymbol{z} - \boldsymbol{s}_k) \right) \right)$

$\boldsymbol{s}_{k+1} = \mathcal{T}_{\zeta_{k+1}}(\boldsymbol{z} - \boldsymbol{x}_{k+1})$

$k = k + 1$

end while

- Trim: Enforce incoherence on a low rank matrix
- $\mathcal{D}_r$: the best rank r approximation
- $\mathcal{P}_{\widetilde{T}_k}$: Projection onto subspace $\widetilde{T}_k = \{\widetilde{\boldsymbol{U}}_k \boldsymbol{A}^T + \boldsymbol{B} \widetilde{\boldsymbol{V}}_k^* \mid \boldsymbol{A} \in \mathbb{R}^{n_2 \times r}, \boldsymbol{B} \in \mathbb{R}^{n_1 \times r}\}$
- $\mathcal{H}^{\dagger}$, normalized dual of $\mathcal{H}$ such that $\mathcal{H}^{\dagger}\mathcal{H} = \mathcal{I}$
- $\mathcal{T}_{\zeta}$: Hard thresholding

Computational Complexity of Updating x

- Due to the construction of Hankel matrix, a Hankel matrix and vector multiplication can be re-formula to a vector-vector convolution, which can be computed via FFT. Hence, a Hankel matrix and matrix multiplication only needs $O(rn\log(n))$ flops.

- With a SVD of $2r \times 2r$ matrix + two QR-decompositions, we need $O(nr^2 + n\log(n)r + r^3)$ flops for updating $\boldsymbol{L}$.

Computational Complexity of Updating x

- Due to the construction of Hankel matrix, a Hankel matrix and vector multiplication can be re-formula to a vector-vector convolution, which can be computed via FFT. Hence, a Hankel matrix and matrix multiplication only needs $O(rn\log(n))$ flops.

- With a SVD of $2r \times 2r$ matrix + two QR-decompositions, we need $O(nr^2 + n\log(n)r + r^3)$ flops for updating $\boldsymbol{L}$.

- $\boldsymbol{x}_{k+1} = \mathcal{H}^\dagger(\boldsymbol{L}_{k+1}) = \sum_{i=1}^r \boldsymbol{\Sigma}_{k+1}^{(i,i)} \omega_j \sum_{a+b=j} \boldsymbol{U}_{k+1}^{(a,i)} \overline{\boldsymbol{V}}_{k+1}^{(b,i)}, \qquad \forall j$
  - ω_j is the weight for averaging j^{th} anti-diagonal
  - Convolutions can be computed by FFT. r of the convolutions result $O(n\log(n)r)$ complexity.

- Overall, need $O(nr^2 + n\log(n)r + r^3)$ flops for updating $\boldsymbol{x}$.

Assumptions

A1 $\mathcal{H}(\boldsymbol{x}) \in \mathbb{C}^{n_1 \times n_2}$ is μ-incoherence: there exists a numerical constant $\mu_0 > 0$ such that

$$\sigma_r(\boldsymbol{E}_L^* \boldsymbol{E}_L) \geq \frac{n_1}{\mu_0}, \quad \sigma_r(\boldsymbol{E}_R^* \boldsymbol{E}_R) \geq \frac{n_2}{\mu_0},$$

where $\mathcal{H}(\boldsymbol{x}) = \boldsymbol{E}_L \boldsymbol{D} \boldsymbol{E}_R^T$ is its Vandermonde decomposition.

- In the undamped spectrally sparse signals, this assumption is guaranteed with at least $\frac{2}{n}$ separations between the non-zero frequencies [LiaoFannjiang'2016].

A2 $\boldsymbol{s} \in \mathbb{C}^n$ is α-sparse: no more than αn non-zero entries in the vector.

- this implies that $\mathcal{H}(\boldsymbol{s})$ is no more than 2α-sparse matrix.

Main Theoretical Result of AAP-Hankel

Recovery Guarantee of AAP-Hankel

Let $\boldsymbol{x}$ and $\boldsymbol{s}$ be two matrices satisfying Assumptions A1 and A2 with

$$\alpha \lesssim O\left(\min\left\{\frac{1}{\mu r^2\kappa^3}, \frac{1}{\mu^{1.5} r^2\kappa}, \frac{1}{\mu^2 r^2}\right\}\right).$$

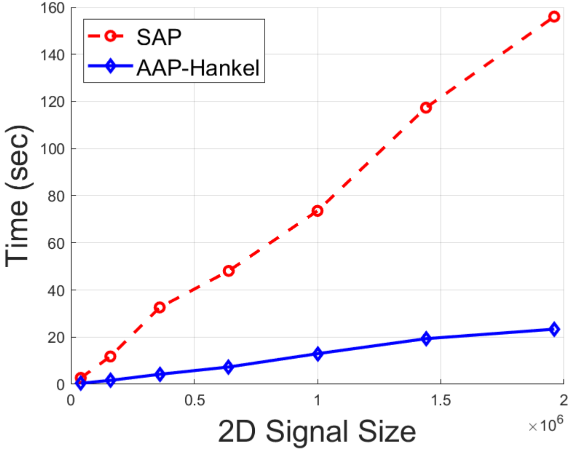
If the thresholding parameters obey $\frac{\mu r \sigma_1^{\boldsymbol{x}}}{\sqrt{n_1 n_2}\sigma_1^{\boldsymbol{z}}} \leq \beta_{init} \leq \frac{3\mu r \sigma_1^{\boldsymbol{x}}}{\sqrt{n_1 n_2}\sigma_1^{\boldsymbol{z}}}$ and $\beta = \frac{\mu r}{2\sqrt{n_1 n_2}}$, alone with the convergence rate parameter $\gamma \in (\frac{1}{\sqrt{12}}, 1)$, then the outputs of Algorithm 1 satisfy

$$\|\mathcal{H}(\boldsymbol{x} - \boldsymbol{x}_k)\|_F \leq \varepsilon \sigma_1^{\boldsymbol{x}}, \|\boldsymbol{s} - \boldsymbol{s}_k\|_\infty \leq \frac{\varepsilon}{\sqrt{mn}}\sigma_1^{\boldsymbol{x}}, \text{ and } \operatorname{supp}(\boldsymbol{s}_k) \subset \operatorname{supp}(\boldsymbol{s})$$

in $O(\log_\gamma \varepsilon)$ iterations.

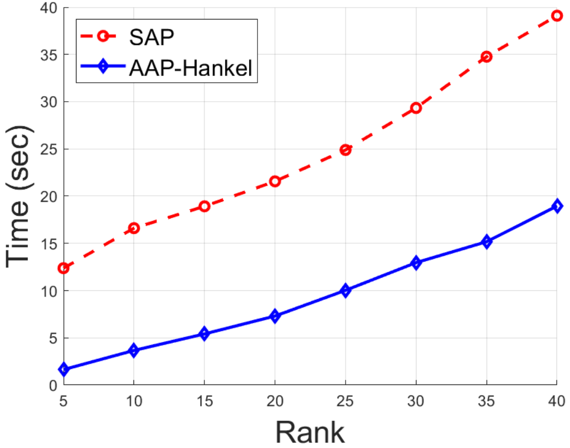
- $\sigma_1^{\boldsymbol{x}}$ denotes the largest singular value of $\mathcal{H}(\boldsymbol{x})$.
- Linear convergence with rate of γ.

Speed Comparison: Dimension vs Runtime



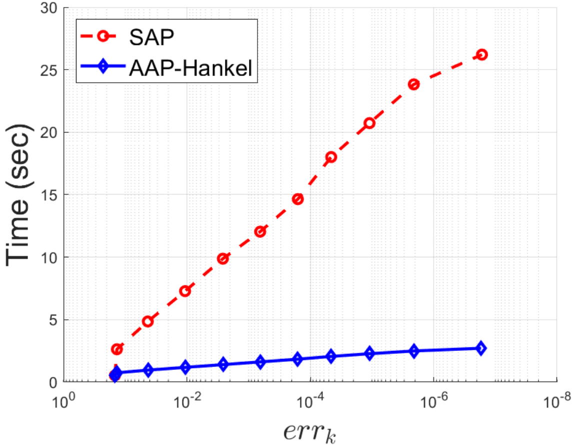
$r = 5, \alpha = 0.1, c = 1$, stop at $err_k < 10^{-4}$.

Speed Comparison: Sparsity vs Runtime



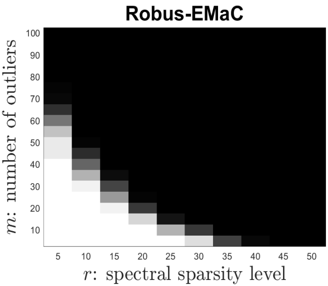
$n = 400^2, \alpha = 0.1, c = 1$, stop at $err_k < 10^{-4}$.

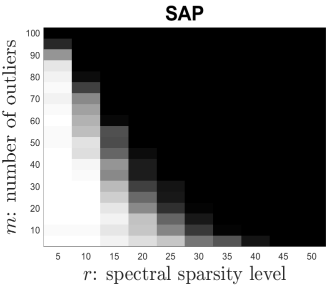
Speed Comparison: Relative Error vs Runtime

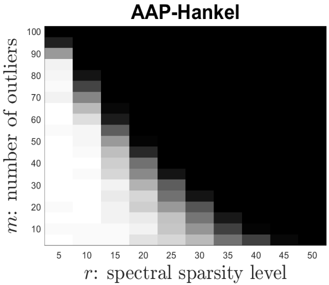


$n = 400^2, r = 5, \alpha = 0.1, c = 1.$

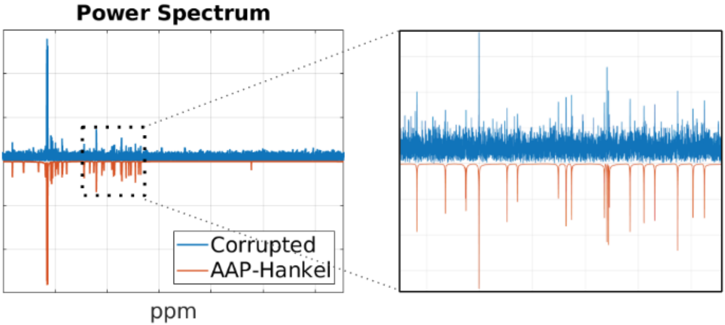
Phase Transition







Impulse Corrupted NMR Singal



$n = 32768$, 50% corruption.

AAP-Hankel is under review, you can found it on arXiv:

Cai, HanQin, Jian-Feng Cai, Tianming Wang, and Guojian Yin. "Fast and Robust Spectrally Sparse Signal Recovery: A Provable Non-Convex Approach via Robust Low-Rank Hankel Matrix Reconstruction." *arXiv preprint arXiv*:1910.05859 (2019).

Code can be found at:

`https://github.com/caesarcai/AAP-Hankel`

Outline

Conclusion

We present a new algorithm for Robust PCA and extend it to low-rank Hankel recovery problem.

- Non-convex with theoretical guarantee
- Best per-iteration complexity
- Guaranteed linear convergence
- Significant speed advantage over the other state-of-the-art algorithms
- High robustness in practice

Future Work

- Recovery guarantee proof for "without Trimming"
- Relax tolerance of α to match our experimental results
- Study partial observed Robust PCA
  - Acceleration with guaranteed recovery
- Study recovery stability with additive noise
  - Theoretically and practically

Thank you!

Any Question?