Learned Robust PCA: A Scalable Deep Unfolding Approach for High-Dimensional Outlier Detection

HanQin Cai (UCLA) Jialin Liu (Alibaba DAMO) Wotao Yin (Alibaba DAMO)

Overview

- We proposed a novel feedforward-recurrent-mixed neural network (FRMNN) model to solve robust PCA (RPCA).
- An exact recovery guarantee is established under mild conditions.
- The proposed model is computational efficient and scalable to high-dimensional RPCA problems.
- The link of codes: `https://github.com/caesarcai/LRPCA`.

Introduction

Problem settings. Recover $\boldsymbol{X}_\star$ from corrupted observation $\boldsymbol{Y}$

$$\boldsymbol{Y} = \boldsymbol{X}_\star + \boldsymbol{S}_\star \in \mathbb{R}^{n_1 \times n_2}, \tag{1}$$

where $\boldsymbol{X}_\star$ is a rank-r data matrix and $\boldsymbol{S}_\star$ is a sparse outlier matrix.
- For simplicity, we assume $n = n_1 = n_2$ in this poster.
- The general case is discussed in our paper.

Joint minimization. Recover $\boldsymbol{X}_\star$ and $\boldsymbol{S}_\star$ simultaneously from $\boldsymbol{Y}$:

$$\begin{aligned} &\underset{\boldsymbol{X},\boldsymbol{S}}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{X}+\boldsymbol{S}-\boldsymbol{Y}\|_\mathrm{F}^2 \\ &\text{subject to} \ \ \mathrm{rank}(\boldsymbol{X}) \le r, \quad \boldsymbol{S} \text{ is } \alpha\text{-sparse}. \end{aligned} \tag{2}$$

- Enforcing $\boldsymbol{X}$ be to low-rank usually requires full/truncated SVD.
- Computational cost is high for large-scale problems.

Non-convex minimization. Let $\boldsymbol{X} = \boldsymbol{L}\boldsymbol{R}^\top$ ($\boldsymbol{L},\boldsymbol{R} \in \mathbb{R}^{n\times r}$) to avoid the low-rank constraint:

$$\begin{aligned} &\underset{\boldsymbol{L},\boldsymbol{R},\boldsymbol{S}}{\text{minimize}} \quad \frac{1}{2}\|\boldsymbol{L}\boldsymbol{R}^\top+\boldsymbol{S}-\boldsymbol{Y}\|_\mathrm{F}^2 \\ &\text{subject to} \ \ \boldsymbol{S} \text{ is } \alpha\text{-sparse}. \end{aligned} \tag{3}$$

ScaledGD. Scaled Gradient Descent is a state-of-the-art iterative algorithm for solving (3):

$$\begin{aligned} \boldsymbol{L}_{k+1} &= \boldsymbol{L}_k - \eta_{k+1}(\boldsymbol{L}_k\boldsymbol{R}_k^\top + \boldsymbol{S}_{k+1} - \boldsymbol{Y})\boldsymbol{R}_k(\boldsymbol{R}_k^\top\boldsymbol{R}_k)^{-1} \\ \boldsymbol{R}_{k+1} &= \boldsymbol{R}_k - \eta_{k+1}(\boldsymbol{L}_k\boldsymbol{R}_k^\top + \boldsymbol{S}_{k+1} - \boldsymbol{Y})^\top\boldsymbol{L}_k(\boldsymbol{L}_k^\top\boldsymbol{L}_k)^{-1} \\ \boldsymbol{S}_{k+1} &= \mathcal{T}_{\widetilde{\alpha}}(\boldsymbol{Y} - \boldsymbol{L}_k\boldsymbol{R}_k^\top), \end{aligned} \tag{4}$$

where $\mathcal{T}_{\widetilde{\alpha}}$ keeps the largest $\widetilde{\alpha}$ entries per row and per column as outliers.
- It requires a partial sorting in every row and column.
- $\widetilde{\alpha}$ is usually taken as $1.5\alpha - 2\alpha$.
- It is expensive when α is relatively large.

Soft-thresholding. We propose to use a simple operator to replace $\mathcal{T}_{\widetilde{\alpha}}$:

$$[\mathcal{S}_\zeta(\boldsymbol{M})]_{i,j} = \begin{cases} 0, & |[\boldsymbol{M}]_{i,j}| \le \zeta; \\ \mathrm{sign}([\boldsymbol{M}]_{i,j})(|[\boldsymbol{M}]_{i,j}| - \zeta), & \text{otherwise.} \end{cases}$$

- It's computationally simple.
- With proper chosen thresholds ζ, it provides even faster convergence.
- Practically, such good ζ can be trained from data.

Algorithm and Analysis

LRPCA. We propose Learned Robust PCA:

$$\begin{aligned} \boldsymbol{S}_{k+1} &= \mathcal{S}_{\zeta_{k+1}}(\boldsymbol{Y} - \boldsymbol{L}_k\boldsymbol{R}_k^\top) \\ \boldsymbol{L}_{k+1} &= \boldsymbol{L}_k - \eta_{k+1}(\boldsymbol{L}_k\boldsymbol{R}_k^\top + \boldsymbol{S}_{k+1} - \boldsymbol{Y})\boldsymbol{R}_k(\boldsymbol{R}_k^\top\boldsymbol{R}_k)^{-1} \\ \boldsymbol{R}_{k+1} &= \boldsymbol{R}_k - \eta_{k+1}(\boldsymbol{L}_k\boldsymbol{R}_k^\top + \boldsymbol{S}_{k+1} - \boldsymbol{Y})^\top\boldsymbol{L}_k(\boldsymbol{L}_k^\top\boldsymbol{L}_k)^{-1}. \end{aligned} \tag{5}$$

We prove that there exists a parameter-sequence $\{\zeta_k, \eta_k\}$ that guarantees (5) converge.

Theorem 1 (Guaranteed recovery.) *Suppose that $\boldsymbol{X}_\star$ is a rank-r matrix with μ-incoherence and $\boldsymbol{S}_\star$ is an α-sparse matrix with $\alpha \le \frac{1}{10^4\mu r^{3/2}\kappa}$. If we set the thresholding values $\zeta_0 = \|\boldsymbol{X}_\star\|_\infty$ and $\zeta_k = \|\boldsymbol{L}_{k-1}\boldsymbol{R}_{k-1}^\top - \boldsymbol{X}_\star\|_\infty$ for $k \ge 1$ for LRPCA, the iterates of LRPCA satisfy*

$$\|\boldsymbol{L}_k\boldsymbol{R}_k^\top - \boldsymbol{X}_\star\|_\mathrm{F} \le 0.03(1-0.6\eta)^k\sigma_r(\boldsymbol{X}_\star) \quad \text{and} \quad \mathrm{supp}(\boldsymbol{S}_k) \subseteq \mathrm{supp}(\boldsymbol{S}_\star),$$

with the step sizes $\eta_k = \eta \in [\frac{1}{4}, \frac{8}{9}]$.

- guaranteed linear convergence with rate $1 - 0.6\eta$
- allow larger step sizes (ScaledGD allows step step up to $\frac{2}{3}$)
- The $\{\zeta_k\}$ formula is dependent on $\boldsymbol{X}_\star$ that cannot be used directly. We propose to use machine learning to calculate $\{\zeta_k\}$.
- The step size $\{\eta_k\}$ can also be learned to adapt to specific data sets.

Model and Training Techniques

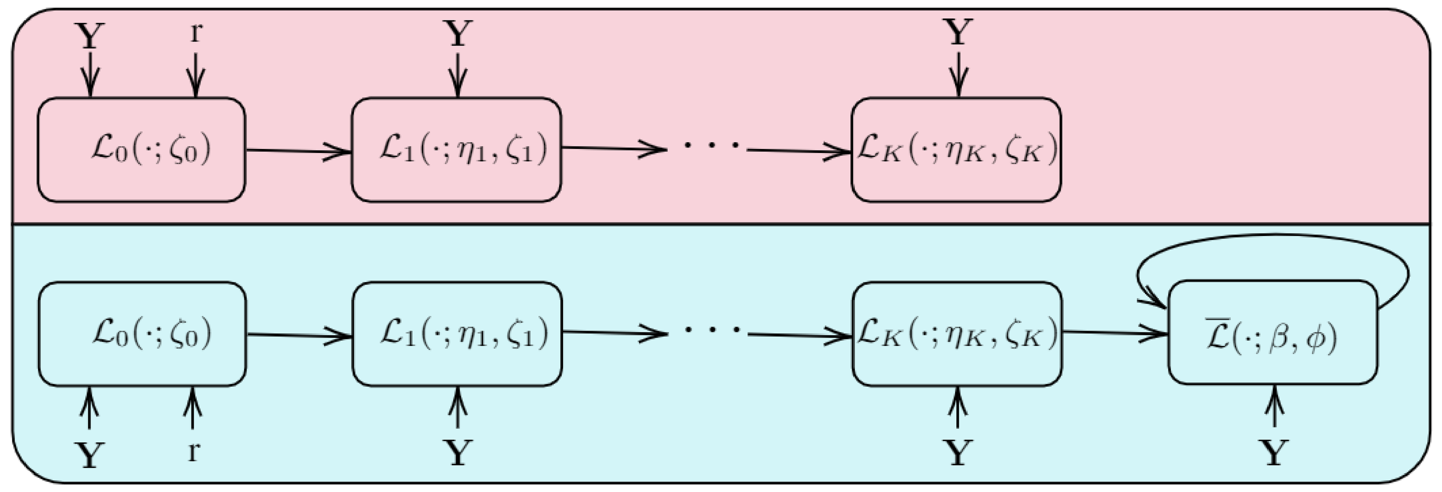
- Unfold (K-iteration) LRPCA as a Feedforward Neural Network (FNN)
- Collect training data set $\mathcal{D}_{\text{train}}$ consisting of $(\boldsymbol{Y}, \boldsymbol{X}_\star)$ pairs.
- Learn the parameters $\Theta = \{\zeta_k, \eta_k\}_{k=0}^K$ by fitting $\boldsymbol{X}_\star$ with $\boldsymbol{X}_K$:

$$\underset{\Theta}{\text{minimize}} \ \mathbb{E}_{(\boldsymbol{Y},\boldsymbol{X}_\star)\sim\mathcal{D}_{\text{train}}}\|\boldsymbol{X}_K(\boldsymbol{Y},\Theta) - \boldsymbol{X}_\star\|_\mathrm{F}^2,$$

where $\boldsymbol{X}_K = \boldsymbol{L}_K\boldsymbol{R}_K^\top$ is determined by $\boldsymbol{Y}$ and Θ based on (5).

The learned parameter Θ can be used on a certain type of RPCA problems that share the distribution of the training examples.

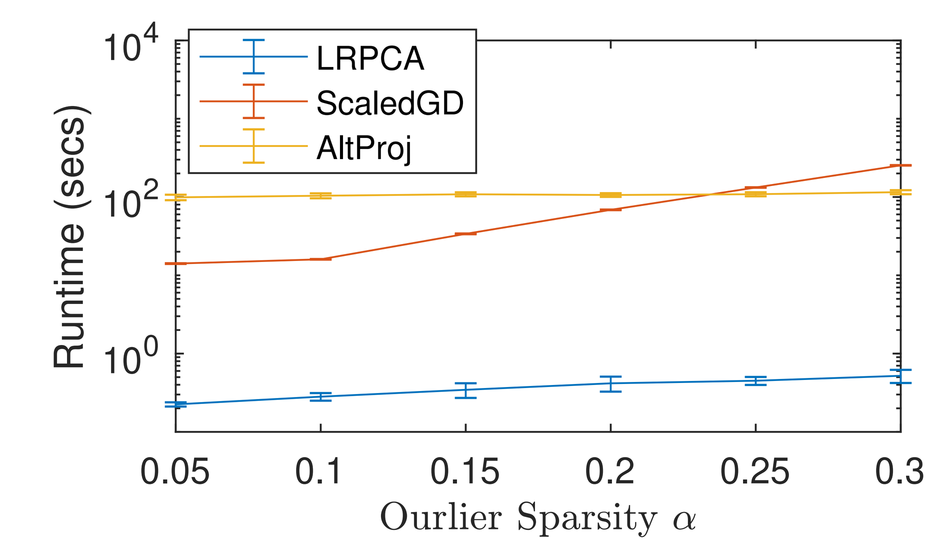
Feedforward-Recurrent-Mixed Neural Network (FRMNN). To extend our model to (infinite) more iterations, we concatenate a Recurrent Neural Network (RNN) after the K-layer FNN:



Numerical Validation

We compare our algorithm LRPCA [1] with two STOA algorithms: ScaledGD [2] and AltProj [3].

Synthetic data. We randomly generate instances following the rules in [4] and show runtime comparison with error bar for varying outlier sparsity α. Problem dimension $d = 3000$ and rank $r = 5$. All algorithms halt when $\|\boldsymbol{Y} - \boldsymbol{X} - \boldsymbol{S}\|_\mathrm{F}/\|\boldsymbol{Y}\|_\mathrm{F} < 10^{-4}$.



Real data. We apply LRPCA to video background subtraction and use VIRAT video dataset as our benchmark. Each frame of the videos is vectorized and become a matrix column, and all frames of a video form a data matrix. The static backgrounds are the low-rank component of the data matrices and moving objects can be viewed as outliers. To achieve the same accuracy, LRPCA runs much faster than both ScaledGD and AltProj in all verification tests:

Video Name	Frame Size	Frame Number	Runtime (secs)		
			LRPCA	ScaledGD	AltProj
ParkingLot1	320×180	965	**16.01**	260.45	63.04
ParkingLot2	320×180	2149	**33.95**	639.03	144.50
ParkingLot3	480×270	1110	**38.85**	662.08	166.91
StreeView	480×270	1034	**33.73**	626.05	167.66

Visualization of the "StreetView" instance:

References

[1] H. Cai, J. Liu, and W. Yin, "Learned robust PCA: A scalable deep unfolding approach for high-dimensional outlier detection," in *Advances in Neural Information Processing Systems*, 2021.

[2] T. Tong, C. Ma, and Y. Chi, "Accelerating ill-conditioned low-rank matrix estimation via scaled gradient descent," *Journal of Machine Learning Research*, 2021.

[3] P. Netrapalli, U. Niranjan, S. Sanghavi, A. Anandkumar, and P. Jain, "Non-convex robust PCA," in *Advances in Neural Information Processing Systems*, 2014.

[4] X. Yi, D. Park, Y. Chen, and C. Caramanis, "Fast algorithms for robust PCA via gradient descent," in *Advances in Neural Information Processing Systems*, 2016.