Outlier-Robust Tensor PCA

Pan Zhou* Jiashi Feng*
* National University of Singapore, Singapore
pzhou@u.nus.edu elefjia@nus.edu.sg

Abstract

Low-rank tensor analysis is important for various real applications in computer vision. However, existing methods focus on recovering a low-rank tensor contaminated by Gaussian or gross sparse noise and hence cannot effectively handle outliers that are common in practical tensor data. To solve this issue, we propose an outlier-robust tensor principle component analysis (OR-TPCA) method for simultaneous low-rank tensor recovery and outlier detection. For intrinsically low-rank tensor observations with arbitrary outlier corruption, OR-TPCA is the first method that has provable performance guarantee for exactly recovering the tensor subspace and detecting outliers under mild conditions. Since tensor data are naturally high-dimensional and multi-way, we further develop a fast randomized algorithm that requires small sampling size yet can substantially accelerate OR-TPCA without performance drop. Experimental results on four tasks: outlier detection, clustering, semi-supervised and supervised learning, clearly demonstrate the advantages of our method.

1. Introduction

In this work, we consider outlier-robust tensor principle component analysis (OR-TPCA) – a new problem for tensor data analysis. As shown in Fig. 1, suppose we are given a 3-way tensor datum $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ that is a mixture of clean low-rank tensor $\boldsymbol{\mathcal{L}}_0$ and sparse outlier noise $\boldsymbol{\mathcal{E}}_0$:

$$\boldsymbol{\mathcal{X}} = \boldsymbol{\mathcal{L}}_0 + \boldsymbol{\mathcal{E}}_0.$$

OR-TPCA aims to address such a problem, *i.e.* how to exactly recover the low-rank tensor $\boldsymbol{\mathcal{L}}_0$ (or its column space) in presence of outliers. This problem is important in many real applications, such as image/video denoising and inpainting [1, 2], data mining [3], collaborative filtering [4], text analysis [5], *etc*. On one hand, recent research [6] shows that high dimensional tensor data of interest, such as videos and image collections, are usually intrinsically low-rank or approximately so. Thus, low-rank tensor data analysis is seeing increasingly more applications. On the other hand, outliers or sample-specific corruptions are common in real data due to sensor failures, malicious tampering, or other system errors [7, 8]. In these cases, large errors concentrate only on a number of samples or several parts of data. These two factors together lead to the problem of recovering low-rank tensor data from outlier corruptions. But, to date, most low-rank tensor analysis methods [2, 9–13] simply assume the noise to be Gaussian or sparse and thus cannot handle outliers or sample-specific corruptions. OR-TPCA substantially generalizes outlier-robust matrix PCA (OR-PCA) problems [7, 14, 15] by not only considering 2-way data (matrix) and providing wider applications.

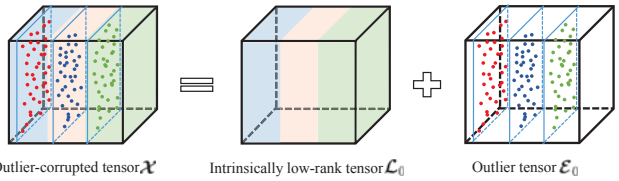


Figure 1: The problem solved by this work: isolate the clean low-rank component from outliers for an outlier-corrupted tensor.

In this work, we develop the first algorithmic solution to this problem both theoretically and practically. We perform robust low-rank analysis on the raw tensor data directly and propose an outlier-robust tensor principle component analysis (OR-TPCA) method (sharing the name with the problem) which recovers the tensor subspace and detect outliers through polynomial-time convex optimization. We also provide theoretical performance guarantee for OR-TPCA: under mild conditions, OR-TPCA can recover the column space of $\boldsymbol{\mathcal{L}}_0$ and detect outliers exactly.

Moreover, since tensor data are naturally multi-way and high-dimensional, *e.g.* long video sequences and millions-of-image collection, algorithm efficiency is critical in practical applications. Therefore, we develop a fast OR-TPCA algorithm. It first randomly samples a small fraction of samples and exactly recovers the underlying subspace from the sampled data that is also shared by the entire data. Thus, the recovered low-rank subspace is the desired one, and by utilizing it, outliers in remaining data can also be effectively detected. We further prove that when the size of randomly sampled data is lower bounded by a positive constant which is much less than the overall data number, fast OR-TPCA

exactly recovers the tensor column space and detect outliers. In fact, this is the first algorithm that can exactly solve a tensor low-rank recovery problem in linear time (*w.r.t.* the data size) with theoretical guarantees.

Recently, based on t-SVD [16, 17] and tensor tubal rank [18], Lu *et al.* [2] extend R-PCA [11] from 2-way matrix to 3-way tensor data and consider the robust tensor PCA (R-TPCA) problem:

$$\min_{\mathcal{L},\mathcal{E}} \|\mathcal{L}\|_* + \lambda\|\mathcal{E}\|_1, \text{ s.t. } \mathcal{X} = \mathcal{L} + \mathcal{E} \in \mathbb{R}^{n_1\times n_2\times n_3},$$

where $\|\mathcal{L}\|_*$ and $\|\mathcal{E}\|_1$ are tensor nuclear and ℓ_1 norms, respectively. However, R-TPCA still assumes that noise is sparse and uniformly distributed across $\mathcal{E}_0$. It cannot effectively handle outlier corruptions. Besides, we focus on a different problem and thus there exist critical differences in theory analysis and guarantees. OR-TPCA solves a problem that is more difficult for getting theoretical guarantee. Indeed, exact recovery is impossible in presence of malicious corruptions. The best one can hope for is to capture exactly or approximately some structures of the problem [14, 15]. Also our approach differs in key analysis techniques, which we believe will prove much more broadly applicable and thus be of general interest.

The contributions of this paper are as follows:

1) We propose a novel outlier-robust tensor principle component analysis (OR-TPCA) method for low-rank tensor analysis. OR-TPCA handles outliers and sample-specific corruptions which fail other low-rank tensor analysis methods.
2) We prove that OR-TPCA succeeds with high probability requiring mild conditions, even in challenging situations where the rank of $\mathcal{L}_0$ and the number of outliers $\mathcal{E}_0$ are as high as $O(n/\log(n))$ and $O(n)$ respectively. Here n is the dimension along which outliers are distributed.
3) The developed fast OR-TPCA algorithm has low complexity that is linear *w.r.t.* data size, and has provable recovery guarantee.

2. Notations and Preliminaries

2.1. Notations

For brevity, we summarize the main notations in Table 1. The *conjugate transpose* $\mathcal{A}^* \in \mathbb{C}^{n_2\times n_1\times n_3}$ of a tensor $\mathcal{A} \in \mathbb{C}^{n_1\times n_2\times n_3}$ is obtained by conjugate transposing each frontal slice and then reversing the order of transposed frontal slices 2 through n_3. $\mathcal{I} \in \mathbb{R}^{n\times n\times n_3}$ denotes the *identity tensor* whose first frontal slice is the $n\times n$ identity matrix, and other frontal slices are all zeros.

Now we consider the Discrete Fourier transformation (DFT) on tensor which is core to the tensor related definitions in Sec. 2.2. Let $\bar{\mathcal{A}} \in \mathbb{C}^{n_1\times n_2\times n_3}$ represent the DFT of $\mathcal{A} \in \mathbb{R}^{n_1\times n_2\times n_3}$ along the 3rd dimension. We can compute $\bar{\mathcal{A}}$ by the Matlab command $\bar{\mathcal{A}} = \texttt{fft}(\mathcal{A}, [], 3)$ and use the inverse DFT to obtain $\mathcal{A} = \texttt{ifft}(\bar{\mathcal{A}}, [], 3)$. We define $\bar{A} \in \mathbb{C}^{n_1n_3\times n_2n_3}$ as

$$\bar{A} = \texttt{bdiag}(\bar{\mathcal{A}}) = \begin{bmatrix} \bar{A}^{(1)} & & & \\ & \bar{A}^{(2)} & & \\ & & \ddots & \\ & & & \bar{A}^{(n_3)} \end{bmatrix},$$

where $\texttt{bdiag}(\cdot)$ unfolds the tensor $\bar{\mathcal{A}}$ to a block diagonal matrix $\bar{A}$. We further define the block circulant matrix $\texttt{bcirc}(\mathcal{A}) \in \mathbb{R}^{n_1n_3\times n_2n_3}$ of $\mathcal{A}$ as

$$\texttt{bcirc}(\mathcal{A}) = \begin{bmatrix} A^{(1)} & A^{(n_3)} & \cdots & A^{(2)} \\ A^{(2)} & A^{(1)} & \cdots & A^{(3)} \\ \vdots & \vdots & \ddots & \vdots \\ A^{(n_3)} & A^{(n_3-1)} & \cdots & A^{(1)} \end{bmatrix}. \quad (1)$$

The definitions of $\bar{A}$ and $\texttt{bcirc}(\mathcal{A})$ are the basis of tensor rank and nuclear norm that will be introduced subsequently.

Table 1: Summary of main notations in the paper.

$\mathcal{A}$	A tensor.	$\boldsymbol{a}$	A vector.
A	A matrix.	a	A scalar.
$\mathcal{I}$	The identity tensor.	$\mathcal{A}^*$	The conjugate transpose of $\mathcal{A}$.
$\mathcal{A}_{ijk}$	The (i,j,k)-th entry of $\mathcal{A}$.	$\bar{\mathcal{A}}$	The DFT of $\mathcal{A}$.
$\mathcal{A}(i,:,:)$	The i-th horizontal slice of $\mathcal{A}$	$\|\mathcal{A}\|_1$	$\|\mathcal{A}\|_1 = \sum_{ijk} \|\mathcal{A}_{ijk}\|$.
$\mathcal{A}(:,i,:)$	The i-th lateral slice of $\mathcal{A}$.	$\|\mathcal{A}\|_\infty$	$\|\mathcal{A}\|_\infty = \max_{ijk} \|\mathcal{A}_{ijk}\|$.
$\mathcal{A}(:,:,i)$	The i-th frontal slice of $\mathcal{A}$.	$\|\mathcal{A}(:,i,:)\|_F$	$\|\mathcal{A}(:,j,:)\|_F = \sqrt{\sum_{ik} \|\mathcal{A}_{ijk}\|^2}$.
$A^{(i)}$	$A^{(i)} = \mathcal{A}(:,:,i)$.	$\|\mathcal{A}\|_{2,1}$	$\|\mathcal{A}\|_{2,1} = \sum_j \|\mathcal{A}(:,j,:)\|_F$.
$\mathcal{A}(i,j,:)$	The (i,j)-th tube of $\mathcal{A}$.	$\|\mathcal{A}\|_F$	$\|\mathcal{A}\|_F = \sqrt{\sum_{ijk} \|\mathcal{A}_{ijk}\|^2}$.
$\text{rank}(A)$	The rank of matrix A.	$\|A\|_*$	Sum of the singular values of A.
$\|A\|_F$	$\|A\|_F = \sqrt{\sum_{i,j} A_{ij}^2}$.		
Θ	The index of outliers in $\mathcal{E}$.	$\mathcal{P}_\Theta$	The projection onto Θ.
$\mathcal{P}_{\Theta^\perp}$	The projection onto $\Theta^\perp$.	$\mathcal{B}(\mathcal{E})$	$\{\widetilde{\mathcal{E}} : \widetilde{\mathcal{E}}(:,i,:) = \frac{\mathcal{E}(:,i,:)}{\|\mathcal{E}(:,i,:)\|_F}$ ($i \in \Theta$); $\widetilde{\mathcal{E}}(:,i,:) = \mathbf{0}$ ($i \notin \Theta$)$\}$.
$\mathcal{P}_\mathcal{U}(\mathcal{A})$	$\mathcal{P}_\mathcal{U}(\mathcal{A}) = \mathcal{U} * \mathcal{U}^* * \mathcal{A}$.		

2.2. Preliminaries

Aiming at recovering the low-rank tensor component, we first introduce two widely used definitions on tensor rank, *i.e.* the tensor average and tubal rank.

Definition 2.1. ***(Tensor average and tubal rank)*** *[2, 19] For an arbitrary tensor $\mathcal{A} \in \mathbb{R}^{n_1\times n_2\times n_3}$, assume $\boldsymbol{r} = (rank(\bar{A}^{(1)}); \cdots; rank(\bar{A}^{(n_3)})) \in \mathbb{R}^{n_3}$. The tensor **average rank** $rank_a(\mathcal{A})$ of $\mathcal{A}$ is defined as*

$$rank_a(\mathcal{A}) = \frac{1}{n_3}\sum_{i=1}^{n_3} \boldsymbol{r}_i = \frac{1}{n_3} rank(\bar{A}) = \frac{1}{n_3} rank(\texttt{bcirc}(\mathcal{A})).$$

*The tensor **tubal rank** $rank_t(\mathcal{A})$ is defined as the number of nonzero singular tubes of $\mathcal{S}$, i.e.,*

$$rank_t(\mathcal{A}) = \#\{i : \mathcal{S}(i,i,:) \neq 0\} = \max(\boldsymbol{r}_1, \cdots, \boldsymbol{r}_{n_3}),$$

where $\mathcal{S}$ is from the t-SVD (see below) of $\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^*$.

As minimizing the tensor average or tubal rank is NP-hard, we use the tensor nuclear norm to relax the tensor average rank as in [2]. Before that, we first introduce the t-product definition, according to which we can compute the product between 3-way tensors.

Definition 2.2. ***(T-product)*** *[17] The t-product between $\mathcal{A} \in \mathbb{R}^{n_1\times n_2\times n_3}$ and $\mathcal{B} \in \mathbb{R}^{n_2\times n_4\times n_3}$ is defined as $\mathcal{A} * \mathcal{B} = fold(\texttt{bcirc}(\mathcal{A}) \cdot \texttt{unfold}(\mathcal{B})) \in \mathbb{R}^{n_1\times n_4\times n_3}$, where $\texttt{unfold}(\mathcal{A}) = [A^{(1)}; A^{(2)}; \cdots; A^{(n_3)}] \in \mathbb{R}^{n_1n_3\times n_2}$ and its inverse operator $\texttt{fold}$ is defined as $\texttt{fold}(\texttt{unfold}(\mathcal{A})) = \mathcal{A}$.*

Based on the definition of t-product, we can develop following necessary concepts. A tensor $\mathcal{P} \in \mathbb{R}^{n\times n\times n_3}$ is *orthogonal* if $\mathcal{P}^* * \mathcal{P} = \mathcal{P} * \mathcal{P}^* = \mathcal{I}$. A tensor is *f-diagonal* if each of its frontal slices is a diagonal matrix.

Definition 2.3. ***(T-SVD)*** *[17] For an arbitrary tensor $\mathcal{A} \in \mathbb{R}^{n_1\times n_2\times n_3}$, it can be factorized by T-SVD as $\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^*$, where $\mathcal{U} \in \mathbb{R}^{n_1\times n_1\times n_3}$ and $\mathcal{V} \in \mathbb{R}^{n_2\times n_2\times n_3}$ are orthogonal, and $\mathcal{S} \in \mathbb{R}^{n_1\times n_2\times n_3}$ is f-diagonal.*

Now we can give the definition of tensor nuclear norm, which is the convex envelop of the tensor average rank. With this property, we use tensor nuclear norm to depict the low-rank structure of a tensor.

Definition 2.4. ***(Tensor nuclear norm)*** *[2] The tensor nuclear norm $\|\mathcal{A}\|_*$ of a tensor $\mathcal{A} \in \mathbb{R}^{n_1\times n_2\times n_3}$ is defined as the sum of singular values of all the frontal slices of $\bar{\mathcal{A}}$, i.e.,*

$$\|\mathcal{A}\|_* = \frac{1}{n_3}\sum_{i=1}^{n_3}\|\bar{A}^{(i)}\|_* = \frac{1}{n_3}\|\bar{\mathcal{A}}\|_* = \frac{1}{n_3}\|\texttt{bcirc}(\mathcal{A})\|_*,$$

which is the convex envelop of the tensor average rank within the unit ball.

Next, we introduce the tensor column space which is related with our theoretical results.

Definition 2.5. ***(Tensor column space)*** *For an arbitrary tensor $\mathcal{A} \in \mathbb{R}^{n_1\times n_2\times n_3}$, assume that $\boldsymbol{r} = (rank(\bar{A}^{(1)}); \cdots; rank(\bar{A}^{(n_3)})) \in \mathbb{R}^{n_3}$, $r = \max_i \boldsymbol{r}_i$ and the t-SVD of $\mathcal{A}$ is $\mathcal{A} = \mathcal{U} * \mathcal{S} * \mathcal{V}^*$. Then its column space Range($\mathcal{A}$) is spanned by $\mathcal{U}_{\mathcal{A}} \in \mathbb{R}^{n_1\times r\times n_3}$, where the first $\boldsymbol{r}_i$ columns of each slice $\bar{\mathcal{U}}_{\mathcal{A}}(:,:,i)$ consist of the first $\boldsymbol{r}_i$ columns of $\bar{\mathcal{U}}(:,:,i)$ and the remaining columns are* **0***s.*

Since for a tensor $\mathcal{A}$, its tensor nuclear norm is equivalent to the sum of the nuclear norms of all frontal slices of $\bar{\mathcal{A}}$, minimizing $\|\mathcal{A}\|_*$ means recovering the low-rank subspace of each frontal slice $\bar{A}^{(i)}$ $(i = 1, \cdots, n_3)$. Thus, the column space of $\mathcal{A}$ is the union of the column spaces of all the frontal slices $\bar{A}^{(i)}$.

3. Outlier-Robust Tensor PCA

We first detail outlier-robust tensor PCA (OR-TPCA), followed by its optimization, and finally analyze its performance theoretically. Let $n_{(1)} = \max(n_1, n_2)$ and $n_{(2)} = \min(n_1, n_2)$, which will be used in Sec. 3.2, 3.3 and 4.2.

3.1. Formulation of OR-TPCA

Without loss of generality, this paper assumes that outliers are distributed along the 2nd dimension of a tensor $\mathcal{X}$, *i.e.*, $\mathcal{E}(:,i,:)$ denotes the possible outliers. Note that outliers can also be distributed along the other dimensions and the developed approach and analysis can be applied directly. In most cases outliers are very sparse compared with the data size. So we can use tensor $\ell_{2,1}$ norm to characterize this sparsity property. Then, OR-TPCA recovers the low-rank tensor component through

$$\min_{\mathcal{L},\mathcal{E}} \|\mathcal{L}\|_* + \lambda\|\mathcal{E}\|_{2,1}, \text{ s.t. } \mathcal{X} = \mathcal{L} + \mathcal{E}, \qquad (2)$$

where $\|\cdot\|_*$ denotes the tensor nuclear norm.

By Definition 2.4, we know that the tensor nuclear norm of $\mathcal{L}$ is equivalent to the nuclear norm (with a factor $1/n_3$) of the block circulant matrix $\texttt{bcirc}(\mathcal{L})$ in Eqn. (1). As pointed out in [2], compared with other matricizations along certain dimension, such as the Tucker rank [20], the block circulant matricization may preserve more spacial relationships among entries and thus better depict the low-rank structure of the tensor. We also note that the tensor nuclear norm is also equivalent to the sum of the nuclear norms of all frontal slices of $\bar{\mathcal{L}}$, which is the DFT of $\mathcal{L}$ along the 3rd dimension. Thus, minimizing the tensor nuclear norm of $\mathcal{L}$ is equivalent to recovering the underlying low-rank structure of each frontal slice $\bar{L}^{(i)}$ $(i = 1, \cdots, n_3)$. This means that it recovers the subspaces of data in Fourier domain, *i.e.*, the subspaces of all the frontal slices $\bar{L}^{(i)}$, while the noise is distributed in the original space. If $n_3 = 1$, then the t-product of 3-way tensors reduces to the standard matrix-matrix product and the tensor nuclear norm degenerates to the matrix nuclear norm. OR-TPCA will reduce to OR-PCA [7, 14, 15]. So OR-PCA is a special case of ours.

3.2. Optimization

We use ADMM [21] to solve problem (2) because of its efficiency and convergence guarantee in solving this problem. The optimization is summarized in Algorithm 1. See Supplementary Material for deduction details. At each iteration, $\mathcal{L}_{k+1}$ and $\mathcal{E}_{k+1}$ have closed form solutions and their computational complexity at each iteration is $O\left(n_{(1)}n_{(2)}^2 n_3 + n_1 n_2 n_3 \log(n_3)\right)$. Note that the computation and memory costs of OR-TPCA are much lower than those of OR-PCA and R-PCA. This is because the main computation and memory costs lie in SVDs involved in these methods and OR-TPCA only requires n_3 SVDs of $n_1 \times n_2$ matrices (the lateral slices $\mathcal{L}_{k+1}^{(i)}$), while in OR-PCA and R-PCA, they have to compute the SVD of the whole data matrix which is much larger and hence need much more computational resource and memory. Note that our optimization method can be implemented in parallel, since at each iteration all lateral slices $\bar{L}_{k+1}^{(i)}$ $(i = 1, \cdots, n_3)$ of $\bar{\mathcal{L}}$ can be parallelly updated which is the main computation cost when updating $\mathcal{L}_{k+1}$, and when updating $\mathcal{E}_{k+1}$, its frontal slices $\mathcal{E}_{k+1}(:,i,:)$ $(i = 1, \cdots, n_2)$ can also be parallelly computed (see Supplementary Material). But for fairness, we adopt the serial updating scheme in our implementation, which is also very fast (see Sec. 5.2.2).

3.3. Exact Subspace Recovery Guarantees

Similar to low-rank matrix recovery [7, 14, 15], exactly separating $\mathcal{X}$ as the low-rank term $\mathcal{L}_0$ plus the outlier-sparse term $\mathcal{E}_0$ is impossible in the following two cases: (1) the true low-rank term $\mathcal{L}_0$ is sparse; (2) the true sparse outliers $\mathcal{E}_0$ are low-rank. To avoid these cases, we need two

mild conditions (assumptions).

Tensor Column-Incoherence Condition on Clean Low-rank Data: This condition is widely used for evaluating the sparsity of a matrix and we generalize it to tensors here. For a tensor $\mathcal{L} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, suppose $\text{rank}_t(\mathcal{L}) = r$ and its skinny t-SVD is $\mathcal{U} * \mathcal{S} * \mathcal{V}^*$. For $\mathcal{U} \in \mathbb{R}^{n_1 \times r \times n_3}$ and $\mathcal{V} \in \mathbb{R}^{n_2 \times r \times n_3}$, we have $\mathcal{U}^* * \mathcal{U} = \mathcal{I}$ and $\mathcal{V}^* * \mathcal{V} = \mathcal{I}$. Then the tensor column-incoherence condition with parameter μ_1 is defined as

$$\mu_1 \geq \frac{n_2 n_3}{r} \max_{i=1,\cdots,n_2} \|\mathcal{V}^* * \mathring{\mathbf{e}}_i\|_F^2, \tag{3}$$

where $\mathring{\mathbf{e}}_i$ is of size $n_2 \times 1 \times n_3$ with the $(i,1,1)$-th entry equal to 1 and the rest equal to **0**s. μ_1 measures how far the tensor is from a column sparse one, and if μ_1 is small, the tensor $\mathcal{L}$ is not column sparse, hence avoiding the first case.

Unambiguity Condition on Outliers: To distinguish low-rank term from outliers, we also require that outliers are not in the subspace of the low-rank clean data and are not low-rank [14, 15]. To avoid this case, similar to matrix OR-PCA [15], we introduce an unambiguity condition on outliers $\mathcal{E}$:

$$\|\mathcal{B}(\mathcal{E})\| \leq \sqrt{\log(n_2)}/4. \tag{4}$$

Note that many noise models satisfy the above condition, including *i.i.d.* Gaussian noise. Indeed, (4) holds as long as the directions of the nonzero lateral slices of $\mathcal{B}(\mathcal{E})$ scatter sufficiently randomly. No matter how many outliers are present, (4) can guarantee the outliers to be not low-rank.

Main Results: Let Range($\mathcal{L}_0$) denote the column space of $\mathcal{L}_0$. Now we present our main results in Theorem 1.

Theorem 1. *Assume Range*$(\mathcal{L}_0) =$ *Range*$(\mathcal{P}_{\Theta_0^\perp}(\mathcal{L}_0))$ *and* $\mathcal{E}_0 \notin$ *Range*$(\mathcal{L}_0)$. *Then any optimal solution* $(\mathcal{L}_0 + \mathcal{H}, \mathcal{E}_0 - \mathcal{H})$ *to problem* (2) *with* $\lambda = 1/\sqrt{\log(n_2)}$ *exactly recovers the tensor column space* $\mathcal{U}_0$ *of* $\mathcal{L}_0$ *and the support set* Θ_0 *of* $\mathcal{E}_0$ *with a probability at least* $1 - c_1 n_{(1)}^{-10}$, *where* c_1 *is a positive constant, if the support set* Θ_0 *is uniformly distributed among all sets of cardinality* $|\Theta_0|$ *and*

$$\textit{rank}_t(\mathcal{L}_0) \leq \frac{\rho_r n_2}{\mu_1 \log(n_{(1)})} \quad \textit{and} \quad |\Theta_0| \leq \rho_s n_2 n_3,$$

where ρ_r *and* ρ_s *are two constants,* $\mathcal{L}_0 + \mathcal{P}_{\Theta_0}\mathcal{P}_{\mathcal{U}_0}(\mathcal{H})$ *satisfies the column-incoherence condition* (3) *and* $\mathcal{E}_0 - \mathcal{P}_{\Theta_0}\mathcal{P}_{\mathcal{U}_0}(\mathcal{H})$ *satisfies the unambiguity condition* (4).

The above results demonstrate that with high probability OR-TPCA can exactly recover $\mathcal{P}_{\Theta_0^\perp}(\mathcal{L}_0)$, *i.e.*, the clean data in $\mathcal{X}$, and the support set Θ_0 of $\mathcal{E}_0$. But it does not mean that OR-TPCA can never recover the corrupted samples. Actually, if a sample is not severely corrupted, it is possible to remove the noise, which is demonstrated in [15, 22, 23] and our experiments (see Fig. 4). Besides, Theorem 1 is also applicable when the 3rd dimension is 1. So the theoretical guarantee of OR-PCA in [15] is also a special case of our theorem.

Algorithm 1 Outlier-Robust Tensor PCA (OR-TPCA)

Input: Tensor data $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$.
Initialize: $\mathcal{L}_0 = \mathcal{E}_0 = \mathcal{J}_0 = \mathbf{0}$, $\lambda = 1/\sqrt{\log(n_2)}$, $\gamma = 1.1$, $\beta_0 = 1e-5$, $\beta_{\max} = 1e+8$, $\epsilon = 1e-8$, and $k = 0$.
While not converged **do**
1. Fix $\mathcal{E}_k$. Update $\mathcal{L}_{k+1}$ by
$$\mathcal{L}_{k+1} = \arg\min_{\mathcal{L}} \|\mathcal{L}\|_* + \frac{\beta_k}{2}\|\mathcal{X} - \mathcal{L} - \mathcal{E}_k + \frac{\mathcal{J}_k}{\beta_k}\|_F^2.$$
2. Fix $\mathcal{L}_{k+1}$. Update $\mathcal{E}_{k+1}$ by
$$\mathcal{E}_{k+1} = \arg\min_{\mathcal{E}} \lambda\|\mathcal{E}\|_{2,1} + \frac{\beta_k}{2}\|\mathcal{X} - \mathcal{L}_{k+1} - \mathcal{E} + \frac{\mathcal{J}_k}{\beta_k}\|_F^2.$$
3. $\mathcal{J}_{k+1} = \mathcal{J}_k + \beta_k(\mathcal{X} - \mathcal{L}_{k+1} - \mathcal{E}_{k+1})$.
4. $\beta_{k+1} = \min(\gamma\beta_k, \beta_{\max})$.
5. Check the convergence conditions:
$$\|\mathcal{E}_{k+1} - \mathcal{E}_k\|_\infty \leq \varepsilon, \quad \|\mathcal{L}_{k+1} - \mathcal{L}_k\|_\infty \leq \varepsilon,$$
$$\|\mathcal{X} - \mathcal{L}_{k+1} - \mathcal{E}_{k+1}\|_\infty \leq \varepsilon.$$
6. $k = k + 1$.
end while
Output: $\mathcal{L}_{k+1}$ and $\mathcal{E}_{k+1}$.

4. The Fast OR-TPCA Algorithm

As aforementioned, tensor data are usually large-scale, such as long video sequences and millions-of-image collection, and thus we propose a fast OR-TPCA algorithm. It has two steps which are given below.

4.1. Sketch of Fast OR-TPCA

(1) Seed Tensor Recovery: Since directly solving the OR-TPCA problem with the whole data is very time-consuming when the data scale is very large, we divide the whole tensor of interest into two tensors of smaller sizes. One is called "seed tensor", which is used for recovering the subspace of the whole tensor. More concretely, we first randomly sample a sub-tensor $\mathcal{X}_l \in \mathbb{R}^{n_1 \times k \times n_3}$ from $\mathcal{X}$, where k is much smaller than n_2. Accordingly, $\mathcal{X}$, $\mathcal{L}$, and $\mathcal{E}$ are respectively partitioned into

$$\mathcal{X} = [\mathcal{X}_l, \mathcal{X}_r], \ \mathcal{L}_0 = [\mathcal{L}_l, \mathcal{L}_r], \ \mathcal{E}_0 = [\mathcal{E}_l, \mathcal{E}_r].$$

Then fast OR-TPCA first recovers $\mathcal{L}_l$ from $\mathcal{X}_l$ by solving a small-sized OR-TPCA problem (2). As compared with n_2, k is very small (see Sec. 4.2), the computation of recovering $\mathcal{L}_l$ is much cheaper than recovering the whole $\mathcal{L}_0$.

(2) Tensor $\ell_{2,1}$ Filtering: As $\mathcal{X}_l$ is randomly selected, $\mathcal{L}_l$ spans the same subspace as $\mathcal{L}_0$ with high probability. Indeed, this is guaranteed (see Theorem 2). Thus, there must exist a tensor $\mathcal{Q}$ such that

$$\mathcal{L}_r = \mathcal{L}_l * \mathcal{Q}.$$

As outliers $\mathcal{E}$ are sparse, $\mathcal{E}_r$ is also sparse and thus can be depicted by the $\ell_{2,1}$ norm. So we can find $\mathcal{Q}$ by solving

$$\min_{\mathcal{E}_r, \mathcal{Q}} \|\mathcal{E}_r\|_{2,1}, \quad \text{s.t. } \mathcal{X}_r = \mathcal{L}_l * \mathcal{Q} + \mathcal{E}_r. \tag{5}$$

Since the DFT is conducted along the 3rd dimension, the lateral slices are independent of each other. Then, with the definition $\|\mathcal{E}_r\|_{2,1} = \sum_{i=1}^{n_2} \|\mathcal{E}_r(:,i,:)\|_F$, we can further

Algorithm 2 Fast OR-TPCA

Input: Tensor data $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times n_3}$ and parameter s/n_2.
1. Randomly sample each lateral slice of $\mathcal{X}$ by $i.i.d.$ Bernoulli distribution $\text{Ber}(s/n_2)$ to construct $\mathcal{X}_l$ and the remaining lateral slices are for constructing $\mathcal{X}_r$.
2. Compute the clean data $\mathcal{L}_l$ and outliers $\mathcal{E}_l$ by solving
$(\mathcal{L}_l, \mathcal{E}_l) = \underset{\mathcal{L}', \mathcal{E}'}{\text{argmin}} \ \|\mathcal{L}'\|_* + \lambda \|\mathcal{E}'\|_{2,1}, \ \text{s.t.} \ \mathcal{X}_l = \mathcal{L}' + \mathcal{E}'.$
3. Compute $\mathcal{L}_r$ and $\mathcal{E}_r$ in $\mathcal{X}_r$ by the closed form solution to problem (6).
Output: $\mathcal{L} = [\mathcal{L}_l, \mathcal{L}_r]$ and $\mathcal{E} = [\mathcal{E}_l, \mathcal{E}_r]$.

divide problem (5) into $(n_2 - k)$ sub-problems along the 2nd dimension and the i-th sub-problem is written as

$$\min_{\mathcal{E}_r^i, \mathcal{Q}^i} \ \| \mathcal{E}_r^i \|_F, \quad \text{s.t.} \ \mathcal{X}_r^i = \mathcal{L}_l * \mathcal{Q}^i + \mathcal{E}_r^i, \qquad (6)$$

where $\mathcal{X}_r^i$, $\mathcal{E}_r^i$ and $\mathcal{Q}^i$ denote $\mathcal{X}_r(:, i, :)$, $\mathcal{E}_r(:, i, :)$ and $\mathcal{Q}(:, i, :)$, respectively. As problem (6) is a least square problem, it admits closed-form solution $\mathcal{E}_r^i = \mathcal{X}_r^i - \mathcal{P}_{\mathcal{U}_{\mathcal{L}_l}}(\mathcal{X}_r^i)$, where $\mathcal{U}_{\mathcal{L}_l}$ is the tensor column space of $\mathcal{L}_l$. We can further obtain $\mathcal{L}_r^i = \mathcal{P}_{\mathcal{U}_{\mathcal{L}_l}}(\mathcal{X}_r^i)$. We summarize the fast algorithm in Algorithm 2.

4.2. Guarantees for Fast OR-TPCA

Here we analyze the exact recovery ability of Algorithm 2. For the randomly selected $\mathcal{X}_l \in \mathbb{R}^{n_1 \times k \times n_3}$, when the value of k is lower bounded by a positive constant, Step 1 in Algorithm 2 guarantees that $\mathcal{P}_{\Theta_0^\perp}(\mathcal{X}_l)$ exactly spans the desired column space $\text{Range}(\mathcal{L}_0)$ with high probability. By Theorem 1, Step 2 can exactly recover the true low-rank structure of $\mathcal{L}_l$ and detect outliers $\mathcal{E}_l$ with high probability. Accordingly, the remaining normal samples can be represented by $\mathcal{L}_l$ while outliers cannot be represented. Our main results are stated in Theorem 2.

Theorem 2. *Assume that each lateral slice of $\mathcal{X}$ is sampled by $i.i.d.$ Bernoulli distribution $Ber(s/n_2)$ for constructing $\mathcal{X}_l$ and all the assumptions in Theorem 1 are fulfilled for the pair $(\mathcal{L}_l, \mathcal{E}_l)$. Then Algorithm 2 exactly recovers the tensor column space of $\mathcal{L}_0$ and the support set Θ_0 of $\mathcal{E}_0$ with a probability at least $1 - \delta$, provided that*

$$s \geq \max\left(c_2 \mu_1 r \log(n_{(1)}), 2\mu_1 r \log\left(\frac{r}{\delta}\right)\right), \qquad (7)$$

where c_2 is a constant, $r = rank_t(\mathcal{L}_0)$ and μ_1 denotes the tensor column-incoherence parameter in Eqn. (3).

Note that if $(\mathcal{L}_0, \mathcal{E}_0)$ obeys the assumptions in Theorem 1, the pair $(\mathcal{L}_l, \mathcal{E}_l)$ also meets them. Thus, actually this fast algorithm does not require more strict conditions than solving the original problem. Also this fast algorithm is very useful when dealing with large-scale tensor data. This is because generally the rank of large-scale data is much smaller than the size. Hence we can just set s/n_2 very small (*e.g.* 6%) and the randomly selected $\mathcal{X}_l$ obeys Eqn. (7) with high probability, which is verified in Sec. 5.2.3. Indeed, by Bernoulli trial property in [23], the sampled number k obeys

Table 2: Exact recovery on random problems of varying sizes.
$r = \text{rank}_t(\mathcal{L}_0) = 0.15n, k = 0.4n, \lambda = 1/\sqrt{\log(n)}, \Theta' = \Theta_0^\perp$.

n	r	k	$\text{rank}_t(\widetilde{\mathcal{L}})$	$\frac{\|\mathcal{P}_{\mathcal{U}_0} - \mathcal{P}_{\widetilde{\mathcal{U}}}\|_F}{\|\mathcal{P}_{\mathcal{U}_0}\|_F}$	$\frac{\|\mathcal{P}_{\Theta'}(\mathcal{L}_0) - \mathcal{P}_{\Theta'}(\widetilde{\mathcal{L}})\|_F}{\|\mathcal{P}_{\Theta'}(\mathcal{L}_0)\|_F}$	$\text{dist}(\Theta_0, \widetilde{\Theta})$
60	9	24	9	4.634e-15	5.518e-15	0
100	15	40	15	2.754e-15	3.322e-15	0
200	30	80	30	2.858e-15	2.870e-15	0

Table 3: Exact recovery on random problems of varying noise magnitudes and different kinds of noise.
$n = 80, r = \text{rank}_t(\mathcal{L}_0) = 0.15n, k = 0.4n, \lambda = 1/\sqrt{\log(n)}, \Theta' = \Theta_0^\perp$.

Outliers	$\text{rank}_t(\widetilde{\mathcal{L}})$	$\frac{\|\mathcal{P}_{\mathcal{U}_0} - \mathcal{P}_{\widetilde{\mathcal{U}}}\|_F}{\|\mathcal{P}_{\mathcal{U}_0}\|_F}$	$\frac{\|\mathcal{P}_{\Theta'}(\mathcal{L}_0) - \mathcal{P}_{\Theta'}(\widetilde{\mathcal{L}})\|_F}{\|\mathcal{P}_{\Theta'}(\mathcal{L}_0)\|_F}$	$\text{dist}(\Theta_0, \widetilde{\Theta})$
$\mathcal{N}(0, 0.01)$	12	5.007e-14	6.559e-14	0
$\mathcal{N}(0, 1)$	12	3.521e-15	3.753e-15	0
$\mathcal{N}(0, 100)$	12	2.306e-15	2.913e-15	0
$\text{Bin}(1, -1)$	12	3.980e-15	4.229e-15	0

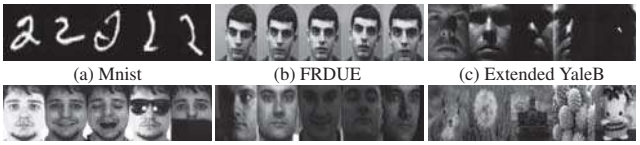
$k \in [0.5s, 2s]$ with a probability at least $1 - n_2^{-10}$ when sampling each lateral slice of $\mathcal{X}$ by $\text{Ber}(s/n_2)$. So this fast algorithm only needs a small fraction of samples and utilizes them to exactly recover the low-rank structure of the whole data, and then recovers the remaining samples one by one. Such a mechanism allows it to be applied to many tasks, such as supervised learning, video summary, *etc*.

5. Experiments

5.1. Evaluation on Synthetic Data

We first test the performance of OR-TPCA on recovering the low-rank tensor from synthetic data and verify that its performance is consistent with the implication of Theorem 1. We generate a tensor $\mathcal{X} = \mathcal{L}_0 + \mathcal{E}_0 \in \mathbb{R}^{n_1 \times n_2 \times n_3}$, where $\mathcal{L}_0$ is low-rank and $\mathcal{E}_0$ contains sparse outliers as follows. We produce a rank_t-r tensor $\mathcal{L}_0 = \mathcal{A} * \mathcal{B}$, where the entries of $\mathcal{A} \in \mathbb{R}^{n_1 \times r \times n_3}$ and $\mathcal{B} \in \mathbb{R}^{r \times n_2 \times n_3}$ are from $i.i.d.$ $\mathcal{N}(0, 1)$. We uniformly select k lateral slices of $\mathcal{E}_0$ as outliers whose entries obey $i.i.d.$ $\mathcal{N}(0, 1)$ and the support set of $\mathcal{E}_0$ is denoted by Θ_0. The remaining entries in $\mathcal{E}_0$ are **0**s. For simplicity, we set $n_1 = n_2 = n_3 = n$.

To verify that OR-TPCA can perform well for various tensor sizes, noise magnitudes and different kinds of noise, we conduct two different experiments. (1) We consider the tensors whose dimension varies as $n = 60$, 100, 200 and report the recovery error of OR-TPCA in Table 2. (2) We test the cases that the entries of $\mathcal{E}_0$ follow $i.i.d.$ $\mathcal{N}(0, 0.01)$, $\mathcal{N}(0, 1)$, $\mathcal{N}(0, 100)$, and the distribution $\text{Bin}(1, -1)$ which *i.i.d.* produces 1 or -1 with probability 0.5, and report the performance in Table 3. We run every experiment for 20 times and report the average results. Note that $\widetilde{\mathcal{L}}$ denotes the recovered tensor and the support set $\widetilde{\Theta}$ of nonzero $\widetilde{\mathcal{E}}(:, i, :)$ is the recovered outlier support set. From Table 2, the recovered tubal rank is exactly equal to r and the relative errors $\|\mathcal{P}_{\mathcal{U}_0} - \mathcal{P}_{\widetilde{\mathcal{U}}}\|_F / \|\mathcal{P}_{\mathcal{U}_0}\|_F$ and $\|\mathcal{P}_{\Theta_0^\perp}(\mathcal{L}_0) - \mathcal{P}_{\Theta_0^\perp}(\widetilde{\mathcal{L}})\|_F / \|\mathcal{P}_{\Theta_0^\perp}(\mathcal{L}_0)\|_F$ are very small, even less than 10^{-10}, where $\mathcal{P}_{\mathcal{U}} = \mathcal{U} * \mathcal{U}^*$. The Hamming distance $\text{dist}(\Theta_0, \widetilde{\Theta})$ between Θ_0 and $\widetilde{\Theta}$ is always 0. These results testify that OR-TPCA can exactly recover the tubal



(a) Mnist (b) FRDUE (c) Extended YaleB

(d) AR (e) PIE (f) MIRFLICKR-25k

Database	#Class	Number of each class	Total number	Size	Difficulty
Mnist	10	100	1000	28×28	def.
FRDUE	153	≈ 20	3059	100×90	def. and pos.
YaleB	38	≈ 68	2414	96×84	ill.
AR	100	26	2600	165×120	ill., exp. and occ.
PIE	68	≈ 170	11554	96×96	ill., exp. and pos.

(g) Descriptions of the testing datasets. ("def.", "pos.", "ill.", "exp." and "occ." are short for "deformation", "pose", "illumination", "expression" and "occlusion", respectively.)

Figure 2: Examples and descriptions of the five testing datasets and MIRFLICKR-25k.

Table 4: AUC of outlier detection and clustering results (ACC, NMI and PUR) on FRDUE (averaged over 20 random runs).

#Outlier	Metric	k-means	R-PCA	OR-PCA	LRR	SNN	R-TPCA	OR-TPCA
100	AUC	—	0.865	0.943	0.914	0.869	0.875	**0.951**
	ACC	0.407	0.436	0.520	0.492	0.491	0.499	**0.564**
	NMI	0.693	0.720	0.785	0.769	0.726	0.766	**0.818**
	PUR	0.427	0.482	0.552	0.499	0.522	0.527	**0.640**
200	AUC	—	0.823	0.886	0.892	0.827	0.835	**0.934**
	ACC	0.374	0.430	0.481	0.474	0.456	0.461	**0.531**
	NMI	0.650	0.694	0.750	0.716	0.724	0.728	**0.779**
	PUR	0.405	0.479	0.518	0.523	0.486	0.492	**0.566**

rank, column space $\boldsymbol{\mathcal{U}}_0$, clean data $\boldsymbol{\mathcal{P}}_{\Theta_0^\perp}(\boldsymbol{\mathcal{L}}_0)$ and the support set Θ_0 of outliers. Table 3 testifies that OR-TPCA is very robust to various noise and corresponding magnitudes. These results fully verify the conclusions in Theorem 1.

5.2. Evaluation on Real Applications

Here we compare our method with other state-of-the-art low-rank factorization methods including R-PCA [11], OR-PCA [15], LRR (with the $\ell_{2,1}$ norm) [24], SNN [12] and R-TPCA [2] on four tensor analysis tasks, *i.e.* outlier detection, clustering, semi-supervised and supervised classification. Fig. 2 gives the brief introduction of five testing databases, including one handwriting dataset Mnist and four face databases, *i.e.* Face Recognition Data of University of Essex[1] (FRDUE), Extended YaleB [25], AR [26] and PIE [27]. In this paper, all classification performance is evaluated by ridge regression (RR), which is defined as

$$\min_{\boldsymbol{W}} \|\boldsymbol{Y} - \boldsymbol{W}\boldsymbol{B}\|_F^2 + \nu\|\boldsymbol{W}\|_F^2, \tag{8}$$

where $\boldsymbol{B}$ is the recovered data by these compared methods, and $\boldsymbol{Y}$ is the label matrix. For tensor based methods (*i.e.* SNN, R-TPCA and OR-TPCA), we vectorize sample matrices for classification or clustering. We do not tune the parameter of RR for our method and set $\nu = 1$ in Sec. 5.2.1 and 5.2.2 and $\nu = 30$ in Sec. 5.2.3, while in other compared methods, their parameters ν are tuned to be the best for different tasks on different datasets. For fairness, we set the regularization parameters of R-PCA [11], OR-PCA [15], RT-PCA [2] and our OR-TPCA as $1/\sqrt{\max(n_1', n_2')}$, $1/\sqrt{\log(n_2')}$, $1/\sqrt{\max(n_1, n_2)n_3}$ and $1/\sqrt{\log(n_2)}$ respectively, where the data matrix size processed by R-PCA and OR-PCA is $n_1' \times n_2'$ and outliers are columnwisely distributed, while in R-TPCA and our OR-TPCA, the data size is $n_1 \times n_2 \times n_3$ and outliers are distributed along the 2nd dimension. These parameter settings are provided by the authors [2, 11, 15]. We manually tune the parameters of LRR and SNN.

[1] http://cswww.essex.ac.uk/mv/allfaces/

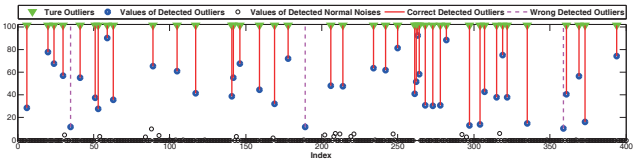


Figure 3: Outlier detection results of OR-TPCA on the first 400 samples in FRDUE with 100 outliers. **Best viewed in color pdf file.**

Table 5: Clustering results (ACC, NMI and PUR) and segmentation error (ERR, %) on Mnist (averaged over 20 random runs).

Metric	k-means	R-PCA	OR-PCA	LRR	SNN	R-TPCA	OR-TPCA
ACC	0.490	0.512	0.520	0.538	0.559	0.781	**0.826**
NMI	0.468	0.500	0.495	0.513	0.535	0.803	**0.880**
PUR	0.532	0.563	0.574	0.568	0.586	0.823	**0.865**
ERR	0.575	0.491	0.479	0.461	0.452	0.258	**0.110**

5.2.1 Outlier Detection

Theorem 1 implies the optimal solution $\boldsymbol{\mathcal{E}}$ can help detect potential outliers in data: when the data $\boldsymbol{\mathcal{P}}_{\Theta_0^\perp}(\boldsymbol{\mathcal{X}})$ are clean, the support set Θ of nonzero $\boldsymbol{\mathcal{E}}(:,i,:)$ reveals the outlier location as in Sec. 5.1. But real data $\boldsymbol{\mathcal{P}}_{\Theta_0^\perp}(\boldsymbol{\mathcal{X}})$ is more often noisy, leading to more nonzero $\boldsymbol{\mathcal{E}}(:,i,:)$. As $\|\boldsymbol{\mathcal{E}}(:,i,:)\|_F^2$ corresponding to outliers are much larger than normal ones, we can use k-means to cluster all $\|\boldsymbol{\mathcal{E}}(:,i,:)\|_F^2$ into two classes (outliers vs. non-outliers) for outlier detection.

To investigate the effectiveness of OR-TPCA in presence of outliers, we construct a dataset by combining FRDUE with the MIRFLICKR-25k dataset (containing 25,000 images for retrieval evaluation) [28]. For FRDUE, we only use the first 40 subjects, resulting in 800 authentic samples from a low-rank subspace. Then we randomly extract 100 and 200 images from MIRFLICKR-25k as outliers. Fig. 2 (f) shows some examples of MIRFLICKR-25k.

We use AUC to evaluate the performance of outlier detection which is computed based on the detection results by k-means conducting on $\boldsymbol{\mathcal{E}}$. We adopt following clustering metrics to measure the quality of the recovered subspace: accuracy (ACC), normalized mutual information (NMI) and purity (PUR). In the experiments, we first remove the outliers detected by k-means (some normal samples may also be removed) and use the remaining recovered data (possibly including undetected outliers) for clustering. We then report the clustering results of normal samples (the normal samples detected as outliers by k-means are given wrong labels). Table 4 summarizes the experimental results. Our OR-TPCA achieves the best outlier detection and clustering

(a) Original (b) R-TPCA (c) OR-TPCA (d) Original (e) R-TPCA (f) OR-TPCA (g) Original (h) R-TPCA (i) OR-TPCA

Figure 4: Examples of face image denoising results on Extended YaleB. **Best viewed in color pdf file.**

Table 6: Classification accuracy (%) and average running time (in seconds) under *semi-supervised learning* setting on the three face testing databases. The numbers, *e.g.* 1, 2, 3, denote the training numbers per person.

Methods	FRDUE				Extended YaleB						AR					
	1	2	3	time	3	6	9	12	15	time	1	2	3	4	5	time
RR	87.4±1.0	92.2±0.7	93.9±0.5	—	56.6±1.8	74.2±1.4	81.5±1.4	86.0±0.9	88.3±1.0	—	39.6±1.9	64.8±1.3	77.5±1.2	84.9±0.8	87.8±0.5	—
R-PCA	90.5±0.5	94.5±0.5	95.4±0.3	2.88	61.5±1.1	79.9±1.0	87.7±1.0	92.1±0.9	92.9±0.7	1.99	43.8±1.2	73.0±1.0	84.5±0.8	89.8±0.6	92.4±0.5	6.23
OR-PCA	90.6±0.7	95.0±0.5	96.0±0.4	1.85	61.8±1.3	80.5±1.1	88.3±0.8	92.9±0.7	93.4±0.5	1.26	43.1±1.1	74.1±0.9	84.9±0.8	89.3±0.6	93.2±0.6	4.61
LRR	90.3±0.6	95.1±0.4	96.6±0.4	3.16	58.9±1.5	72.9±1.3	79.3±1.0	86.0±0.8	88.3±0.7	2.04	45.1±1.6	68.1±0.9	77.6± 0.7	86.0± 0.6	89.4± 0.6	6.41
SNN	92.2±0.7	95.9±0.5	96.9±0.4	4.01	63.1±1.2	81.7±0.9	89.8±0.5	93.6±0.3	94.1±0.3	2.67	45.7±1.2	75.3±1.0	86.6±0.7	91.1±0.5	94.1±0.4	6.89
R-TPCA	95.6±0.5	97.9±0.4	98.7±0.4	0.76	65.1±1.1	84.5±0.8	92.0±0.7	95.7±0.5	97.3±0.5	0.38	49.3±1.2	78.9±0.8	89.7±0.7	93.6±0.6	96.0±0.6	1.43
OR-TPCA	**97.3**±0.6	**98.6**±0.4	**99.2**±0.2	**0.69**	**71.4**±1.1	**90.3**±0.9	**95.9**±0.6	**98.2**±0.4	**98.6**±0.3	**0.24**	**56.8**±0.8	**87.2**±0.7	**95.0**±0.6	**98.1**±0.4	**99.0**±0.2	**1.03**

performance, because (1) it takes advantage of the tensor multi-dimension structure; (2) it uses the $\ell_{2,1}$ norm that can better depict outliers than the ℓ_1 norm. Fig. 3 shows some outliers detected by OR-TPCA. We also observe that outlier detection can boost the clustering results. These methods benefit from detecting outliers first and achieve better clustering results than vanilla k-means.

5.2.2 Unsupervised and Semi-Supervised Learning

As aforementioned, low-rank subspace methods have been applied to image denoising and aligning. In this experiment, we also apply OR-TPCA for removing noise and corruptions on face and handwritten digits via recovering their subspaces, considering authentic handwriting and face images approximately lie on a union of low-rank subspaces [2, 11, 24, 29–35]. Meanwhile, the shadows, facial expressions and occlusions on face images displayed in Fig. 2 are more like contiguous noise and hence the $\ell_{2,1}$ norm can characterize them better, compared with the conventional ℓ_1 norm. Since the tensor nuclear norm is orientation dependent, we find that organizing the images along the 3rd direction to form a $w \times h \times n$ tensor provides slightly better results. R-TPCA also performs better when its processed tensor is constructed in this way. Besides, this construction way leads to higher computational efficiency, since the sample number n is usually much larger than its dimension w and h. Otherwise, we have to compute decompositions of w or h matrices of sizes $n \times h$ or $w \times n$.

We evaluate OR-TPCA on one handwritten digits database Mnist and three face datasets including FRDUE, Extended YaleB and AR. We use ACC, NMI, PUR and segmentation error (ERR) to evaluate the clustering results on Mnist; adopt the classification accuracy as performance metric on the three face datasets. We also report the average algorithm running time (total denoising time divided by the sample number). We run each classification experiment with a specific training size over 10 random train/test splits of the recovered data and report the average results.

Table 7: Subspace recovery performance of fast OR-TPCA on PIE. ("#TSP" is short for "#Training sample per person".)

#TSP	6	10	14	18	22	26	30
$\|\mathcal{P}_{\mathcal{U}_i} - \mathcal{P}_{\widetilde{\mathcal{U}}}\|_F / \|\mathcal{P}_{\mathcal{U}}\|_F$	5.1e-01	2.7e-15	2.7e-15	2.7e-15	2.7e-15	2.8e-15	2.7e-15

Table 5 summarizes the clustering results on Mnist. OR-TPCA consistently outperforms the baselines for all the four metrics. It improves by 13.8% over the second best R-TPCA in terms of ERR. Table 6 displays the classification accuracy on the three face datasets. All methods achieve impressive good classification results on FRDUE partially since it is easier than Extended YaleB and AR (see Fig. 2). Though Extended YaleB and AR are more challenging, OR-TPCA still performs better than other methods across all the settings, especially for insufficient training samples cases (*e.g.* ≤ 6 and 3 training samples per person on Extended YaleB and AR, respectively). This clearly proves the superior robustness of OR-TPCA. Also, the results demonstrate tensor based methods, *i.e.* SNN, R-TPCA and OR-TPCA, achieve higher classification accuracy, as they take advantage of the multi-dimensional structure of the tensor instead of directly vectorizing samples like the matrix based baselines. We also observe that $\ell_{2,1}$ based methods, *i.e.* OR-TPCA and OR-PCA, usually outperform their ℓ_1 counterparts, *i.e.* R-TPCA and R-PCA. It is because the $\ell_{2,1}$ norm can better detect the contiguous noise. This is confirmed by results given in Fig. 4, where OR-TPCA de-shadows and recovers faces much better than R-TPCA.

We also observe that OR-TPCA is much faster than others: R-PCA, OR-PCA and LRR need to perform SVD over a very large data matrix and SNN decomposes three large matrices (unfold the tensor along three modes). Conversely, OR-TPCA needs to do SVD on n matrices however each matrix is much smaller than those involved in the baselines.

Table 8: Classification accuracy (%) and total running time (in hours) under *semi-supervised learning* setting on PIE. ("#TSP" is short for "#Training sample per person".)

#TSP	5	10	15	20	time
RR	63.1±1.7	76.0±1.1	83.4±1.2	85.8±0.9	—
R-PCA	66.1±1.2	79.5±0.9	85.8±0.7	88.1±0.3	>20
OR-PCA	63.2±1.4	78.3±1.0	86.5±0.8	88.9±0.4	>20
LRR	67.6±1.5	80.2±1.0	86.2±0.6	88.5±0.5	>20
SNN	69.0±1.3	82.0±0.6	87.3±0.5	89.9±0.3	>20
R-TPCA	70.1±1.1	83.1±0.8	88.5±0.6	90.9±0.3	1.74
Fast OR-TPCA (10)	73.3±1.2	85.0±0.7	89.5±0.4	**91.8**±0.2	**0.14**
Fast OR-TPCA (30)	**73.5**±1.1	84.5±0.7	89.6±0.6	91.6±0.3	0.22
OR-TPCA (all)	73.2±1.1	**85.3**±0.6	**89.7**±0.5	91.7±0.2	1.51

5.2.3 Experiments on Fast OR-TPCA

Here we evaluate the performance of our fast OR-TPCA on PIE, a large-scale face dataset, and run every experiment for 10 times for the average performance. We first examine that given a small fraction of the data, whether fast OR-TPCA can exactly recover the subspace of the whole data. For this, we vary the sampled face numbers per subject from 6 to 30. We construct the corresponding tensor $h \times n \times w$ when given n images of size $h \times w$. We compute the relative error $\|\mathcal{P}_{\mathcal{U}_i} - \mathcal{P}_{\widetilde{\mathcal{U}}}\|_F / \|\mathcal{P}_{\widetilde{\mathcal{U}}}\|_F$ for performance measure, where $\mathcal{P}_{\mathcal{U}} = \mathcal{U} * \mathcal{U}^*$; $\mathcal{U}_i$ is the column space recovered by fast OR-TPCA using only i samples per person and $\widetilde{\mathcal{U}}$ is the column space obtained using all samples. From Table 7, one can observe that, when given a few samples ($i = 6$), there is a quality gap between $\mathcal{U}_i$ and $\widetilde{\mathcal{U}}$. But using more samples (larger i), the performance gap decreases rapidly. Even when the selected samples occupy less than 6% (about 10 samples per person), the relative error is already as low as 10^{-14}. This clearly demonstrates ability and sample-efficiency of fast OR-TPCA on recovering the subspace, consistent with conclusions of Theorem 2.

We then conduct semi-supervised classification experiments to further evaluate fast OR-TPCA. We randomly select 10, 30 and all samples from each person to construct the "seed tensor" used for fast OR-TPCA. In contrast, other compared methods utilize the whole data for recovery. Table 8 reports the average accuracy and total running time (total denoising time). Both OR-TPCA and fast OR-TPCA outperform the baselines. OR-PCA and fast OR-TPCA achieve similar accuracy with 10 and 30 samples per person. Using 10 samples per person as the "seed tensor", fast OR-TPCA is 10× faster than R-TPCA and the original OR-TPCA, and is 100× faster than other baselines, clearly demonstrating the high efficiency of fast OR-TPCA.

Now we apply fast OR-TPCA to large-scale real supervised classification. By robustly recovering the training-test shared subspace from noisy training data, fast OR-TPCA can boost supervised classification performance. In this experiment, we compare fast OR-TPCA with R-PCA, OR-PCA and R-TPCA baselines. SNN and LRR is not applicable here. Note that for supervised classification, the baselines have to recover the "seed matrix (tensor)" constructed by training data first and use them to denoise and classify

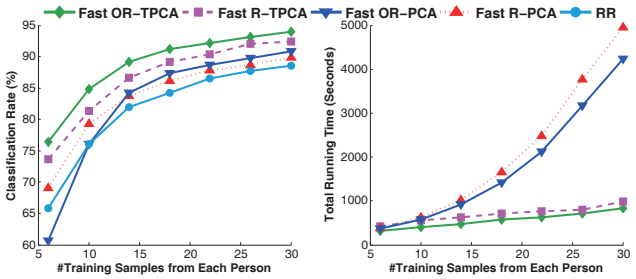


Figure 5: Classification accuracy (%) and total running time (denoising time on training and testing samples, in seconds) of fast algorithms under *supervised learning* setting on PIE.

testing data, similar to our fast OR-TPCA. This also gives their *accelerated version*. However, they do not enjoy performance guarantee as ours. Note that a fast R-PCA is proposed in [36] though no guarantees provided. We use this fast R-PCA as baseline. We evaluate all the methods on PIE, by randomly selecting i (from 6 to 30) training images per person and testing on the remaining images.

Fig. 5 shows that OR-TPCA always provides the highest classification accuracy. Besides, we can make following observations. First, the performance of all the methods improves with using more training samples. For our proposed OR-TPCA, such behavior is consistent with Theorem 2. When the training number is sufficiently large (*e.g.* 10 per person), the tensor subspace can be recovered correctly, giving better classification performance. We also compare the total running time (denoising time on training and testing samples). Note that RR has no denoising time. When training number per person is 18, fast OR-TPCA performs 2× faster than fast OR-PCA and R-PCA. Moreover, the running time of fast OR-TPCA increase slowly when the training size increases. When processing k training samples of size $h \times w$, SVD consumes most computation time per iteration. The cost of fast OR-TPCA, R-PCA and OR-PCA per iteration is $O(kw^2h + kwh\log(w))$, $O(k^2wh)$ and $O(k^2wh)$ respectively. So fast OR-TPCA is much more efficient than them. These consistent results testify that fast OR-TPCA can recover low-rank tensor better and more efficiently.

6. Conclusions

We proposed an outlier-robust tensor principle component analysis (OR-TPCA) method for low-rank tensor analysis. For large-scale tensor data, we further developed a fast algorithm that effectively speeds up OR-TPCA. We also applied this fast algorithm for supervised classification. Extensive experimental results demonstrate that our method obtains better performance on various learning tasks effectively and efficiently compared with state-of-the-arts.

7. Acknowledgements

Jiashi Feng was partially supported by National University of Singapore startup grant R-263-000-C08-133 and Ministry of Education of Singapore AcRF Tier One grant R-263-000-C21-112.

References

[1] J. Liu, P. Musialski, P. Wonka, and J. Ye, "Tensor completion for estimating missing values in visual data," *IEEE TPAMI*, vol. 35, no. 1, pp. 208–220, 2013.

[2] C. Lu, J. Feng, Y. Chen, W. Liu, Z. Lin, and S. Yan, "Tensor robust principal component analysis: Exact recovery of corrupted low-rank tensors via convex optimization," in *IEEE CVPR*, 2016.

[3] M. Mørup, "Applications of tensor (multiway array) factorizations and decompositions in data mining," *Wiley Interdisciplinary Reviews Data Mining and Knowledge Discovery*, vol. 1, no. 1, pp. 24–40, 2011.

[4] A. Karatzoglou, X. Amatriain, L. Baltrunas, and N. Oliver, "Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering," in *Proc. ACM Conf. Recommender Systems*, pp. 79–86, 2010.

[5] M. Collins and S. Cohen, "Tensor decomposition for fast parsing with latent-variable PCFGs," in *NIPS*, pp. 2519–2527, 2012.

[6] T. Kolda and B. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.

[7] H. Xu, C. Caramanis, and S. Sanghavi, "Robust PCA via outlier pursuit," in *NIPS*, 2010.

[8] M. Balcan and H. Zhang, "Noise-tolerant life-long matrix completion via adaptive sampling," in *NIPS*, pp. 2955–2963, 2016.

[9] F. D. L. Torre and M. J. Black, "A framework for robust subspace learning," *IJCV*, vol. 54, no. 1-3, pp. 117–142, 2003.

[10] P. J. Huber, *Robust statistics*. Springer, 2011.

[11] E. J. Candès, X. Li, Y. Ma, and J. Wright, "Robust principal component analysis?," *Journal of the ACM*, vol. 58, no. 3, p. 11, 2011.

[12] B. Huang, C. Mu, D. Goldfarb, and J. Wright, "Provable low-rank tensor recovery," *Optimization Online*, vol. 4252, p. 2, 2014.

[13] A. Anandkumar, P. Jain, Y. Shi, and U. Niranjan, "Tensor vs matrix methods: Robust tensor decomposition under block sparse perturbations," *arXiv:1510.04747*, 2015.

[14] H. Xu, C. Caramanis, and S. Sanghavi, "Robust PCA via outlier pursuit," *IEEE TIT*, vol. 58, no. 5, pp. 3047–3064, 2012.

[15] H. Zhang, Z. Lin, C. Zhang, and E. Chang, "Exact recoverability of robust PCA via outlier pursuit with tight recovery bounds," in *AAAI*, 2015.

[16] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, 2009.

[17] M. E. Kilmer and C. D. Martin, "Factorization strategies for third-order tensors," *Linear Algebra and its Applications*, vol. 435, no. 3, pp. 641–658, 2011.

[18] Z. Zhang, G. Ely, S. Aeron, N. Hao, and M. Kilmer, "Novel methods for multilinear data completion and de-noising based on tensor-SVD," in *IEEE CVPR*, 2014.

[19] M. Kilmer, K. Braman, N. Hao, and R. Hoover, "Third-order tensors as operators on matrices: A theoretical and computational framework with applications in imaging," *SIAM Journal on Matrix Analysis and Applications*, vol. 34, no. 1, pp. 148–172, 2013.

[20] L. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, no. 3, pp. 279–311, 1966.

[21] Z. Lin, R. Liu, and Z. Su, "Linearized alternating direction method with adaptive penalty for low-rank representation," in *NIPS*, 2011.

[22] R. Liu, Z. Lin, F. D. la Torre, and Z. Su, "Fixed-rank representation for unsupervised visual learning," in *IEEE CVPR*, pp. 598–605, 2012.

[23] H. Zhang, Z. Lin, and C. Zhang, "Completing low-rank matrices with corrupted samples from few coefficients in general basis," *IEEE TIT*, vol. 62, no. 8, pp. 4748–4768, 2016.

[24] G. Liu, Z. Lin, and Y. Yu, "Robust subspace segmentation by low-rank representation," in *ICML*, 2010.

[25] A. S. Georghiades, P. N. Belhumeur, and D. J. Kriegman, "From few to many: Illumination cone models for face recognition under variable lighting and pose," *IEEE TPAMI*, vol. 23, no. 6, pp. 643–660, 2001.

[26] A. Martinez and R. Benavente, "The AR face database," *CVC Tech. Rep. 24*, Jun. 1998.

[27] T. Sim, S. Baker, and M. Bsat, "The CMU pose, illumination, and expression database," *IEEE TPAMI*, vol. 25, pp. 1615–1618, 2003.

[28] M. J. Huiskes and M. S. Lew, "The MIR flickr retrieval evaluation," in *ACM MIR*, 2008.

[29] E. Elhamifar and R. Vidal, "Sparse subspace clustering," in *IEEE CVPR*, 2009.

[30] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE TPAMI*, vol. 31, pp. 210–227, 2009.

[31] P. Zhou, Z. Lin, and C. Zhang, "Integrated low-rank-based discriminative feature learning for recognition," *IEEE TNNLS*, vol. 27, no. 5, pp. 1080–1093, 2016.

[32] P. Zhou, C. Zhang, and Z. Lin, "Bilevel model based discriminative dictionary learning for recognition," *IEEE TIP*, vol. 26, no. 3, pp. 1173–1187, 2017.

[33] H. Zhang, Z. Lin, C. Zhang, and J. Gao, "Relations among some low rank subspace recovery models," *Neural Computation*, vol. 27, no. 9, pp. 1915–1950, 2015.

[34] H. Zhang, Z. Lin, C. Zhang, and J. Gao, "Robust latent low rank representation for subspace clustering," *Neurocomputing*, vol. 145, pp. 369–373, 2014.

[35] Y. Wang, C. Xu, C. Xu, and D. Tao, "Beyond RPCA: Flattening complex noise in the frequency domain," in *AAAI*, 2017.

[36] R. Liu, Z. Lin, Z. Su, and J. Gao, "Linear time principal component pursuit and its extensions using ℓ_1 filtering," *Neurocomputing*, vol. 142, pp. 529–541, 2014.