Randomized Tensor Robust Principal Component Analysis

Mohammad Mohammadpour Salut ¹ and David Anderson ²

¹School of Electrical and Computer Engineering
²Affiliation not available

October 30, 2023

Abstract

Removing noise from hyperspectral images can be very beneficial for improving classification accuracy. Recently, tensor robust principal component analysis (TRPCA) has been successfully employed to reduce noise in hyperspectral images. In TRPCA, a minimization involving a tensor nuclear norm and a 1-norm is employed to separate the low-rank hyperspectral image from the sparse noise. Tensor nuclear norm minimization is solved by iteratively performing tensor singular value thresholding (T-SVT). However, TRPCA possesses high computational complexity primarily due to the implementation of the T-SVT operator. The conventional approach for T-SVT is first to perform full tensor singular value decomposition (T-SVD), and then to shrink the singular values of the frontal slices in the frequency domain. However, when the solution of tensor nuclear norm minimization is a low-rank tensor, a good strategy is to incrementally find the singular values until they fall below the threshold. In this letter, we propose a randomized blocked algorithm for computing tensor singular value thresholding, and we leverage the compression attained by the fast Fourier transform (FFT) to accelerate TRPCA. Numerical experiments indicate that our method is significantly faster performing than TRPCA via the full T-SVD while maintaining classification accuracy.
Randomized Tensor Robust Principal Component Analysis

Mohammad M. Salut and David V. Anderson Senior Member, IEEE

Abstract—Removing noise from hyperspectral images can be very beneficial for improving classification accuracy. Recently, tensor robust principal component analysis (TRPCA) has been successfully employed to reduce noise in hyperspectral images. In TRPCA, a minimization involving a tensor nuclear norm and a $\ell_1$-norm is employed to separate the low-rank hyperspectral image from the sparse noise. Tensor nuclear norm minimization is solved by iteratively performing tensor singular value thresholding (T-SVT). However, TRPCA possesses high computational complexity primarily due to the implementation of the T-SVT operator. The conventional approach for T-SVT is first to perform full tensor singular value decomposition (T-SVD), and then to shrink the singular values of the frontal slices in the frequency domain. However, when the solution of tensor nuclear norm minimization is a low-rank tensor, a good strategy is to incrementally find the singular values until they fall below the threshold. In this letter, we propose a randomized blocked algorithm for computing tensor singular value thresholding, and we leverage the compression attained by the fast Fourier transform (FFT) to accelerate TRPCA. Numerical experiments indicate that our method is significantly faster performing than TRPCA via the full T-SVD while maintaining classification accuracy.

Index Terms—Randomized tensor RPCA, adaptive randomized tensor singular value thresholding, tensor nuclear norm minimization.

I. INTRODUCTION

HYPERSONTAL images have been widely used in many remote sensing applications e.g. classification [1]–[6], unmixing [7]–[10], and target detection [11], [12]. Often, hyperspectral images are contaminated with noise. Therefore, the separation of the desired information and noise is of great interest. A hyperspectral image can be naturally represented by three dimensions: two spatial and one spectral dimensions. Typically, a hyperspectral image consists of up to several hundred spectral bands, displaying high correlation across spectra. Conventional analysis and noise-reduction methods involve reshaping spectral bands into very long vectors to form the columns of a data matrix, then using principal component analysis (PCA) for spectral dimensionality reduction. Farrell et al. [13] employed PCA to investigate the effect of dimension reduction on detection of difficult targets. Chen et al. [14] proposed a denoising method based on PCA and wavelet shrinkage for high signal-to-noise ratio (SNR) hyperspectral images. However, in hyperspectral imagery, some spectral bands can have high noise levels, and PCA is susceptible to gross noise and outliers. A single large magnitude outlier can distort the principal components far from the actual subspace of interest. A variation on PCA, called robust PCA (RPCA), deals with outliers by modeling a low-dimensional subspace using PCA and creating a separate sparse additive matrix containing outliers. Another problem with vectorization of multidimensional signals is that it can break the intrinsic structure of data leading to reduced algorithm effectiveness.

To better utilize the inherent structure in hyperspectral images, several multidimensional denoising techniques have been developed to effectively reduce noise in hyperspectral data [15]–[18]. Recently, robust PCA (RPCA) has been extended from 2D matrices to 3-way tensors to capture multifactor structure in data. It was shown that the resulting tensor robust PCA (TRPCA) using tensor nuclear norm can exactly recover the low-rank and sparse tensors from their sum [19]. When applied to a hyperspectral image, TRPCA decomposes the image into a low-rank tensor which contains the desired hyperspectral image, and a sparse tensor which contains noise. Sun et al. [18] utilized TRPCA to improve hyperspectral classification accuracy. Results [18] indicate that TRPCA can effectively remove noise from hyperspectral image. However, TRPCA possesses high computational complexity.

Tensor nuclear norm minimization is solved iteratively by applying tensor singular value thresholding (T-SVT). The conventional approach for T-SVT is first to perform full tensor singular value decomposition (T-SVD), and then to shrink the singular values of the frontal slices in the frequency domain. However, such an approach is time-consuming and computationally inefficient since only values exceeding the threshold are ultimately needed.

Che et al. [20] proposed a fixed-rank randomized T-SVT to accelerate TRPCA by computing only the $k$ largest singular values and their associated vectors. The authors [20] applied their method to denoising of RGB images. However, a drawback of their approach is that the rank parameter has to be provided a priori. In real-world applications, the optimal rank of a data tensor is seldom known in advance. If the chosen rank is too small relative to the numerical rank of data, the low-rank approximation is poor; and, if the selected rank is too large, complexity increases and time and memory savings may be significantly reduced. The other shortcoming of assigning an equal rank to all frontal slices is that the rank of the first few frontal slices are significantly larger than the rest of them, as will be shown later.

In this letter, we propose a randomized rank-adaptive algorithm for computing tensor singular value thresholding, and we leverage the compression attained by the fast Fourier transform (FFT) to accelerate TRPCA.
II. PRELIMINARIES

In this paper, scalars are denoted by lowercase letters, e.g., \( l \). Vectors are denoted by boldface lowercase letters, e.g., \( L \). Matrices are denoted by capital letters, e.g., \( L \). Third-order tensors are denoted by Euler script letters, e.g., \( \mathcal{L} \in \mathbb{R}^{n_1 \times n_2 \times n_3} \). The \( i \)-th frontal slice of \( \mathcal{L} \) is denoted by a tensor \( \mathcal{L}_i \). The \( i \)-th frontal slice of \( \mathcal{L} \) is denoted by a matrix \( L^{(i)} \). The bar notation \( \bar{\cdot} \) and \( (\cdot) \) represent matrix multiplication and T-product, respectively. The bar notation \( (\cdot) \) represents Fourier domain and \( \mathcal{L} \) denotes the discrete Fourier transform (DFT) of \( \mathcal{L} \) which is computed by taking the Fast Fourier Transform (FFT) of \( \mathcal{L} \) along each tube. Using MATLAB notation

\[
\hat{\mathcal{L}} = \text{fft}(\mathcal{L}, [], 3)
\]

T-product. Let \( \mathcal{L} \in \mathbb{R}^{n_1 \times n_2 \times n_3} \) and \( \mathcal{M} \in \mathbb{R}^{n_2 \times n_4 \times n_3} \), the T-product \( \bar{\mathcal{N}} = \mathcal{L} * \mathcal{M} \in \mathbb{R}^{n_1 \times n_4 \times n_3} \) is obtained by multiplying each frontal slice of \( \mathcal{L} \) with its counterpart in \( \mathcal{M} \) [22–24].

\[
\bar{\mathcal{N}}^{(i)} = \hat{\mathcal{L}}^{(i)} \cdot \hat{\mathcal{M}}^{(i)}
\]

Due to the conjugate symmetry property of the DFT for real-valued data, we only need to compute \( \bar{\mathcal{N}}^{(i)} = \hat{\mathcal{L}}^{(i)} \cdot \hat{\mathcal{M}}^{(i)} \) for about half of the transformed frontal slices. The T-product is summarized in Algorithm 1 [19].

Algorithm 1 T-Product

Input: \( \mathcal{L} \in \mathbb{R}^{n_1 \times n_2 \times n_3} \), \( \mathcal{M} \in \mathbb{R}^{n_2 \times n_4 \times n_3} \)
Output: \( \bar{\mathcal{N}} \in \mathbb{R}^{n_1 \times n_4 \times n_3} \)

1: \( \mathcal{L} = \text{fft}(\mathcal{L}, [], 3) \); \( \mathcal{M} = \text{fft}(\mathcal{M}, [], 3) \)
2: for \( i = 1 : \lceil \frac{n_1+1}{2} \rceil \) do
3: \( \bar{\mathcal{N}}^{(i)} = \hat{\mathcal{L}}^{(i)} \cdot \hat{\mathcal{M}}^{(i)} \)
4: end for
5: for \( i = \lceil \frac{n_1+1}{2} \rceil + 1 : n_3 \) do
6: \( \bar{\mathcal{N}}^{(i)} = \text{conjugation}(\bar{\mathcal{N}}^{(n_3-i+2)}) \)
7: end for
8: \( \bar{\mathcal{N}} = \text{ifft}(\bar{\mathcal{N}}, [], 3) \)

Conjugate transpose. The conjugate transpose of a tensor \( \mathcal{L} \in \mathbb{C}^{n_1 \times n_2 \times n_3} \) is obtained by conjugate transposing each of the frontal slices and reversing the order of frontal slices 2 through \( n_3 \), \( \bar{\mathcal{L}}^H \in \mathbb{C}^{n_2 \times n_1 \times n_3} \).

Identity tensor. \( I \in \mathbb{R}^{n_1 \times n_1 \times n_3} \) is an identity tensor having its first frontal slice being the \( n_1 \times n_1 \) identity matrix, and zeros everywhere else. The Fourier transform of identity tensor (\( I \)) along the third dimension is denoted as \( \bar{I} \). Every frontal slice of \( \bar{I} \) is an identity matrix \( \bar{I}^{(i)} = I \).

Orthogonal tensor. A tensor \( \mathcal{U} \in \mathbb{R}^{n_1 \times n_1 \times n_3} \) is orthogonal if \( \mathcal{U} \ast \bar{\mathcal{U}}^T = \bar{\mathcal{U}}^T \ast \mathcal{U} = I \).

F-diagonal tensor. A tensor is F-diagonal if each frontal slice is a diagonal matrix [22–24].

T-SVT. T-SVT of data tensor \( \mathcal{L} \in \mathbb{R}^{n_1 \times n_2 \times n_3} \) is obtained by soft-thresholding of the singular values of the frontal slices in the frequency domain \( (\hat{S} - \tau)_+ = \max(\hat{S} - \tau, 0) \). \( \mathcal{D}_+ (\cdot) \) denotes the T-SVT operator. Algorithm 2 summarizes tensor singular value thresholding procedure [19].

Algorithm 2 T-SVT

Input: \( \mathcal{L} \in \mathbb{R}^{n_1 \times n_2 \times n_3} \)
Output: \( \hat{\mathcal{L}} = \mathcal{D}_+ (\mathcal{L}) \)

1: \( \hat{\mathcal{L}} = \text{fft}(\mathcal{L}, [], 3) \)
2: for \( i = 1 : \lceil \frac{n_1+1}{2} \rceil \) do
3: \( [\bar{U}_i, \bar{V}_i] = \text{svd}(\hat{\mathcal{L}}^{(i)}) \)
4: \( \hat{\mathcal{L}}^{(i)} = \bar{U}_i \cdot (\bar{S} - \tau)_+ \cdot \bar{V}_i^H \)
5: end for
6: for \( i = \lceil \frac{n_1+1}{2} \rceil + 1 : n_3 \) do
7: \( \hat{\mathcal{L}}^{(i)} = \text{conjugation}(\hat{\mathcal{L}}^{(n_3-i+2)}) \)
8: end for
9: \( \hat{\mathcal{L}} = \text{ifft}(\hat{\mathcal{L}}, [], 3) \)

III. PROPOSED ALGORITHM

Our proposed algorithm performs T-SVT in an efficient iterative manner. It takes as inputs the data tensor \( \mathcal{L} \in \mathbb{R}^{n_1 \times n_2 \times n_3} \) with \( n_1 \geq n_2 \) and block size \( b \ll n_2 \). The T-SVT operator shrinks the singular values of the frontal slices of the input tensor in the Fourier domain; therefore, it suffices to obtain only those values exceeding the threshold. However, in practice, it is hard to guess the optimal rank of each frontal slice for a given threshold \( \tau \), so the SVD is performed incrementally in blocks as summarized here. Let \( \hat{\mathcal{L}}^{(i)} \in \mathbb{C}^{n_1 \times n_2} \) to be the \( i \)-th frontal slice of the frequency-domain input data tensor \( \hat{\mathcal{L}} \). Suppose that the optimal rank of \( \hat{\mathcal{L}}^{(i)} \) is \( r(i) \) such that \( (k(i) - 1) \cdot b < r(i) \leq k(i) \cdot b \) for some integer \( k(i) \) where \( r \) and \( k \in \mathbb{N}^{n_3} \). For each \( \hat{\mathcal{L}}^{(i)} \), the algorithm iterates to build up the SVD, each iteration computing \( b \) singular values with their corresponding singular vectors of \( \hat{\mathcal{L}}^{(i)} \).

A. Algorithm Details

First, we compute the fast Fourier transform of \( \mathcal{L} \) along the third dimension. \( \hat{\mathcal{L}} \) denotes the FFT of \( \mathcal{L} \). Our algorithm can be split into four computational stages: Stage 1. We perform T-QB factorization on the input data tensor \( \mathcal{L} \) by extending randomized QB approximation [25, 26] from 2D matrices to 3-way tensors. We utilize the random sampling technique to learn the dominant subspace of \( \mathcal{L} \). The aim is to compute the \( j \)-th block of the singular values and the corresponding singular...
Algorithm 3 Rapid Tensor Singular Value Thresholding

Input: \( L \in \mathbb{R}^{n_1 \times n_2 \times n_3}, \tau > 0, P, b, K \)

Output: \( \hat{L} = D_2(L) \)

1: \( \hat{L} = \text{fft}(L) \)
2: \( \hat{L} = 0; \quad z_c = 0 \)
3: for \( i = 1 : \left\lceil \frac{n_i + 1}{2} \right\rceil \) do
4: \( \hat{U} = []; \quad \hat{S} = []; \quad \hat{V} = [] \)
5: converge = false
6: while converge do
7: \( \Omega_i = \text{randn}(n_2, b) \)
8: \( \hat{Q}_i \approx \text{qr}(L(i) \cdot \Omega_i) \)
9: for \( j = 1 : P \) do
10: \( \hat{Q}_j \approx \text{qr}(L(i) \cdot L(i)^H \cdot \hat{Q}) \)
11: end for
12: \( \hat{B} = \hat{Q}^H \cdot \hat{L}^{(i)} \)
13: \( \hat{U}, \hat{S}, \hat{V} = \text{svd}({\hat{B}}) \)
14: \( \hat{U} = \hat{U}, \quad \hat{Q} \cdot \hat{U} \)
15: \( \hat{S} = \text{diag}(\hat{S}) \)
16: \( \hat{V} = \hat{V}, \hat{V} \)
17: if \( \hat{S}(\text{end}) < \tau \) then
18: converge = true
19: else
20: \( \hat{L}^{(i)} = \hat{L}^{(i)} - \hat{Q} \cdot \hat{B} \)
21: end if
22: end while
23: \( r = \text{length}(\text{find}(\hat{S} > \tau)) \)
24: if \( r \geq 1 \) then
25: \( \hat{L}^{(i)} = \hat{U}(:,1:r) \cdot \text{diag}(\hat{S}(1:r) - \tau) \cdot \hat{V}(1:r)^H \)
26: \( z_c = 0 \)
27: else
28: \( z_c = z_c + 1 \)
29: if \( z_c \geq K \) then
30: break
31: end if
32: end for
33: \( \hat{L}^{(i)} = \text{conj}(\hat{L}^{(n_3 - i + 2)}) \)
34: end for
35: \( \hat{L} = \text{fft}(\hat{L}, \), 3) 

denotes the power iteration scheme to improve the quality of the low-rank approximation.

\[
\hat{Y} = \hat{L}^{(i)} \cdot \Omega_j^{(i)}
\]

where \( P \) specifies the number of power iterations. Typically, \( P \in \{1, 2\} \) suffices in practice. \( \hat{Y}_P \in \mathbb{C}^{n_1 \times b} \) has the same orthogonal basis as \( \hat{Y} \) but its singular values decay faster than those of \( \hat{Y} \). The orthonormal basis \( \hat{Q} \in \mathbb{C}^{n_1 \times b} \), representing the range of the samples matrix \( \hat{Y}_P \), is approximated by performing QR decomposition on \( \hat{Y}_P \).

\[
[\hat{Q}, \sim] = \text{qr}(\hat{Y}_P)
\]

We utilize the orthonormal matrix \( \hat{Q} \) to project the high-dimensional frontal slice \( \hat{L}_j^{(i)} \) to a low-dimensional space.

\[
\hat{B} = \hat{Q}^H \cdot \hat{L}_j^{(i)}
\]

Stage 2. We perform singular value decomposition on the small matrix \( \hat{B} \in \mathbb{C}^{k \times n_2} \) to obtain a factorization

\[
\hat{B} = \hat{U} \cdot \hat{S} \cdot \hat{V}^H
\]

Thus, the partial SVD of \( \hat{L}_j^{(i)} \) is efficiently computed as:

\[
\hat{U}_j^{(i)} = \hat{Q} \cdot \hat{U}, \quad \hat{S}_j^{(i)} = \hat{S}, \quad \hat{V}_j^{(i)} = \hat{V}
\]

Stage 3. The residual update is computed using Eq. (10) as:

\[
\hat{L}_{j+1}^{(i)} = \hat{L}_j^{(i)} - \hat{Q} \cdot \hat{B}
\]

It is important to note that the subspace of \( \hat{L}_j^{(i)} \) is always orthogonal to the columns of \( \hat{U}_m^{j} \cdot \hat{U}_m^{i} \), which ensures that orthogonality is strictly maintained in the columns of \( \hat{U}_j^{(i)} \) throughout the iterations.

Stage 4. Stages 1-3 are repeated until at least one of the singular values falls below the threshold \( \tau \). Then, the SVT of \( \hat{L}_j^{(i)} \) is computed as

\[
\hat{L}_j^{(i)} = \hat{U}^{(i)} \cdot \left( \hat{S}^{(i)} - \tau \right) \cdot \hat{V}^{(i)H}
\]

This procedure is followed for all frontal slices \( \hat{L}_j^{(i)} \) where \( i = 1 : \left\lceil \frac{n_i + 1}{2} \right\rceil \). However, computing the entire SVT of the frontal slices may not be necessary due to the compression attained by performing FFT along each tube—the frontal slices containing low-frequency components typically possess significantly larger singular values than those containing high frequency components. If the dominant singular values of \( K \) consecutive frontal slices are all less than the threshold \( S(k)_{i+K} < \tau \), the rest of the frontal slices \( S(k)_{i+K+1} \) are very unlikely to produce any singular values larger than the threshold, and our algorithm does not perform SVT on those slices to accelerate the computation \( \hat{L}_j^{(i)} \)\[i = 1 : \left\lceil \frac{n_i + 1}{2} \right\rceil \] = 0. A summary of our rapid tensor singular value thresholding method is given in Algorithm 3. The computational complexity of performing FFT on the input data tensor \( L \) is \( O(n_1 n_2 n_3 \log n_3) \) flops. The complexity of performing our proposed randomized blocked SVT on \( m \) frontal slices where \( m \leq \left\lceil \frac{n_i + 1}{2} \right\rceil \) is \( O(\sum_{i=1}^{m} (k(i)C + n_1 r(i) + n_1 n_2 r(i))) \) flops where \( C = n_2 b + (P + 1)n_1 b^2 + 2(P + 1)n_1 n_2 b + n_2 b^2 \), and \( k(i) \) and \( r(i) \) represent the number of blocks (iterations) and the optimal rank of the \( i \)-th frontal slice, respectively.
FIG. 1: a) shows the magnitude of the Fourier coefficients of the vectorization of the frontal slices of the hyperspectral data sets. b) shows the vector of the first ten dominant singular values of each frontal slice in the frequency domain. c) shows the rank of each frontal slice in 10-th, 20-th, 30-th, and 40-th iterations and the last iteration (green lines) obtained by TRPCA [19]. d) shows the rank of each frontal slice obtained by our algorithm.

TABLE I: Performance comparison of classification accuracy for different methods.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Data size</th>
<th>Number of classes</th>
<th>SVM</th>
<th>PCA</th>
<th>RPCA [27]</th>
<th>Ours</th>
<th>Tensor RPCA [19]</th>
<th>Fixed-rank R-TRPCA [20]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Botswana</td>
<td>$1476 \times 145 \times 256$</td>
<td>14</td>
<td>91.8</td>
<td>91.4</td>
<td>92.1</td>
<td>98.5</td>
<td>98.4</td>
<td>98.5</td>
</tr>
<tr>
<td>Kennedy Space Center</td>
<td>$512 \times 176 \times 614$</td>
<td>13</td>
<td>92.5</td>
<td>89.1</td>
<td>92.7</td>
<td>98.5</td>
<td>98.4</td>
<td>98.5</td>
</tr>
<tr>
<td>Pavia Centre</td>
<td>$1096 \times 102 \times 715$</td>
<td>9</td>
<td>98.7</td>
<td>98.7</td>
<td>98.6</td>
<td>99.5</td>
<td>99.5</td>
<td>99.5</td>
</tr>
<tr>
<td>Pavia University</td>
<td>$610 \times 103 \times 340$</td>
<td>9</td>
<td>90.5</td>
<td>90.1</td>
<td>90.2</td>
<td>97.8</td>
<td>97.7</td>
<td>97.7</td>
</tr>
<tr>
<td>Salinas</td>
<td>$512 \times 204 \times 217$</td>
<td>16</td>
<td>92.8</td>
<td>92.7</td>
<td>92.8</td>
<td>99.3</td>
<td>99.2</td>
<td>99.3</td>
</tr>
<tr>
<td>Average accuracy</td>
<td></td>
<td></td>
<td>93.26</td>
<td>92.4</td>
<td>93.28</td>
<td>98.72</td>
<td>98.72</td>
<td>98.7</td>
</tr>
</tbody>
</table>

TABLE II: Comparisons of CPU time of RPCA algorithms (sec).

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Botswana</td>
<td>114</td>
<td>415</td>
<td>146</td>
<td>373</td>
</tr>
<tr>
<td>KSC</td>
<td>318</td>
<td>795</td>
<td>332</td>
<td>345</td>
</tr>
<tr>
<td>Pavia Centre</td>
<td>217</td>
<td>522</td>
<td>230</td>
<td>391</td>
</tr>
<tr>
<td>Pavia University</td>
<td>55</td>
<td>167</td>
<td>68</td>
<td>108</td>
</tr>
<tr>
<td>Salinas</td>
<td>81</td>
<td>331</td>
<td>92</td>
<td>164</td>
</tr>
<tr>
<td>Elapsed time</td>
<td>783</td>
<td>2250</td>
<td>868</td>
<td>1381</td>
</tr>
</tbody>
</table>

IV. RESULTS

We evaluated the performance of our method using five hyperspectral image data sets namely ‘Botswana,’ ‘Kennedy Space Center,’ ‘Pavia Centre,’ ‘Pavia University,’ and ‘Salinas’[^1]. Table 1 shows the image size and the number of bands ($n_2$) for each data set. All experiments are run on MATLAB 2022a using a computer with AMD Ryzen 7 2700X 3.70 GHz and 32 GB memory.


To highlight the compression attained by the FFT, we computed the FFT of the hyperspectral images along the third dimension. Then, we vectorized each frontal slice to form a matrix, and displayed the magnitude of the Fourier coefficients. As Fig. 1a shows, the magnitude of the coefficients of the frontal slices containing low frequency components are significantly larger than those containing high frequency components. To highlight the variation of the magnitude of the singular values across the frontal slices, we computed the SVD of each frontal slice in the Fourier domain and displayed the vector of the first ten dominant singular values of each frontal slice. As Fig. 1b shows, the frontal slices containing low frequency components possess substantially larger singular values than those containing high frequency components. Therefore, when the given threshold ($\tau$) is larger than the dominant singular values of $K$ consecutive frontal slices, the singular values of the remaining frontal slices which contain higher frequency components do not likely exceed the threshold. This quality enables us to refrain from performing SVT on the remaining frontal slices, reducing the computation.

We utilized TRPCA [19] for denoising of the hyperspectral images. Fig. 1c shows the optimal rank of each frontal slice in
10-th, 20-th, 30-th, and 40-th iterations and the last iteration (green lines). As the figure shows, the rank of the frontal slices containing high frequency components are significantly smaller than those containing the low frequency components. We employ our proposed T-SVT algorithm to accelerate TRPCA [19]. We considered the block size $b = 5$, the number of power iterations $P = 1$ and $K = 10$. Fig. 1d shows the rank of each frontal slice estimated by our method. As the figure shows the optimal rank estimated by our adaptive randomized algorithm is very similar to that of [19]. Moreover, the majority of frontal slices do not have any singular values larger than the threshold even until 20-th iterations. As explained in the previous section, our algorithm skips performing SVT on them which speeds up the computation.

We compared the classification performance on data after processing with various algorithms for noise reduction, namely, our proposed algorithm, PCA on vectorized data, RPCA on vectorized data [27], TRPCA [19], and fixed-rank randomized TRPCA [20]. As it has been mentioned, the fixed-rank randomized T-SVT [20] cannot estimate the optimal rank for a given threshold. Therefore, we set the tubal rank $r = 10$ for the fixed-rank R-TRPCA [20]. After applying the aforementioned methods to the hyperspectral images, each data set is randomly divided into a 10/90 training and testing, and support vector machine (SVM) is utilized for classification. We also applied the SVM classification to the original data sets to use them as a benchmark. Table 1 shows the classification accuracy of each method. PCA and RPCA (2D-based methods) were not effective in improving classification accuracy, whereas, tensor-based methods effectively improved the classification accuracy. The classification accuracy of our method is as good as TRPCA via full T-SVD [19]. Table 2 shows the time elapsed for each RPCA algorithms. Results indicate that our method is faster than other RPCA algorithms, and is up to 4 times faster than the conventional TRPCA [19].

V. CONCLUSION

Tensor singular value thresholding is the key subroutine for solving tensor nuclear norm minimization that arises from low-rank tensor recovery problems such as TRPCA. We developed a randomized blocked tensor singular value thresholding scheme to accelerate TRPCA. Numerical experiments on numerous hyperspectral data sets indicate that classification accuracy following the application of our method is as good as when using TRPCA via full T-SVD. Results show that our algorithm is generally a few times faster than conventional TRPCA.

REFERENCES


