Multilinear Low-Rank Tensors on Graphs & Applications

Nauman Shahid*, Francesco Grassi†, Pierre Vandergheynst*
*Signal Processing Laboratory (LTS2), EPFL, Switzerland, firstname.lastname@epfl.ch
† Politecnico di Torino, Italy, francesco.grassi@polito.it

***Abstract*—Current low-rank tensor literature lacks development in large scale processing and generalization of the low-rank concepts to graphs [8]. Motivated by the fact that the first few eigenvectors of the k_{nn}-nearest neighbors graph provide a smooth basis for the data, we propose a novel framework "Multilinear Low-Rank Tensors on Graphs (MLRTG)". The applications of our scalable method include approximate and fast methods for tensor compression, robust PCA, tensor completion and clustering. We specifically focus on Tensor Robust PCA on Graphs in this work.**

I. Introduction

For a tensor $\boldsymbol{\mathcal{Y}}^* \in \mathbb{R}^{n \times n \times n}$, let L_μ be the combinatorial Laplacian of the k_{nn}-graphs constructed between the rows of the matricized versions $Y_\mu, \forall \mu = 1, 2, 3$. Also let $L_\mu = P_\mu \Lambda_\mu P_\mu^\top$ be the eigenvalue decomposition of L_μ, where the eigenvalues Λ_μ are sorted in the increasing order. Then, $\boldsymbol{\mathcal{Y}}^*$ is said to be Multilinear Low-Rank on Graphs (MLRTG) if it can be encoded in terms of the lowest k Laplacian eigenvectors $P_{\mu k} \in \mathbb{R}^{n \times k}, \forall \mu$ as:

$$\mathrm{vec}(\boldsymbol{\mathcal{Y}}^*) = (P_{1k} \otimes P_{2k} \otimes P_{3k})\,\mathrm{vec}(\boldsymbol{\mathcal{X}}^*), \tag{1}$$

where $\mathrm{vec}(\cdot)$ denotes the vectorization, $\otimes$ denotes the kronecker product and $\boldsymbol{\mathcal{X}}^* \in \mathbb{R}^{k \times k \times k}$ is the *Graph Core Tensor (GCT)*. We call the tuple (k, k, k), where $k \ll n$, as the *Graph Multilinear Rank* of $\boldsymbol{\mathcal{Y}}^*$ and refer to a tensor from the set of all possible MLRTG as $\boldsymbol{\mathcal{Y}} \in \mathbb{MLT}$.

Throughout, we use Fast Approximate Nearest Neighbors library (FLANN) [3] for the construction of L_μ which costs $\mathcal{O}(n\log(n))$ and is parallelizable. We also assume that a fast and parallelizable framework, such as the one proposed in [9] is available for the computation of $P_{\mu k}$ which costs $\mathcal{O}(n\frac{k^2}{c})$, where c is the number of processors.

II. Tensor Robust PCA on Graphs (TRPCAG)

For any $\boldsymbol{\mathcal{Y}} \in \mathbb{MLT}$, the GCT $\boldsymbol{\mathcal{X}}$ is the most useful entity. For a clean matricized tensor Y_1 it is straight-forward to determine the matricized $\boldsymbol{\mathcal{X}}$ as $X_1 = P_{1k}^\top Y_1 P_{2,3k}$, where $P_{2,3k} = P_{1k} \otimes P_{2k} \in \mathbb{R}^{n^2 \times k^2}$. For the case of noisy $\boldsymbol{\mathcal{Y}}$, corrupted by spasre noise, one seeks a robust $\boldsymbol{\mathcal{X}}$ which is not possible without an appropriate regularization on $\boldsymbol{\mathcal{X}}$. Hence, we propose to solve the following convex minimization problem:

$$\min_{\boldsymbol{\mathcal{X}}} \|Y_1 - P_{1k} X_1 P_{2,3k}^\top\|_1 + \gamma \sum_\mu \|X_\mu\|_{*g(\Lambda_{\mu k})}, \tag{2}$$

where $\|\cdot\|_{*g(\cdot)}$ denotes the weighted nuclear norm and $g(\Lambda_{\mu k}) = \Lambda_{\mu k}^\alpha, \alpha \geq 1$, denotes the kernelized Laplacian eigenvalues as the weights for the nuclear norm minimization. Assuming the eigenvalues are sorted in ascending order, this corresponds to a higher penalization of higher singular values of X_μ which correspond to noise. Such a nuclear norm minimization on the full tensor (without weights) has appeared in earlier works [2]. This costs $\mathcal{O}(n^4)$. However, note that in our case we lift the computational burden by minimizing only the core tensor $\boldsymbol{\mathcal{X}}$. The above algorithm requires nuclear norm on $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{k \times k \times k}$ and scales with $\mathcal{O}(nk^2 + k^4)$. This is a significant complexity reduction over Tensor Robust PCA (TRPCA) [2]. We use Parallel Proximal Splitting Algorithm to solve eq. (2) as explained in Appendix of [4].

III. Theoretical Analysis

Although the TRPCAG based inverse problem (eq. 2) is orders of magnitude faster than the standard tensor robust PCA [2], it introduces some approximation. Our detailed theoretical analysis is presented in Theorem 2 of [4]. In simple words, the theorem states that 1) the singular vectors and values of a matrix / tensor obtained by our method are equivalent to those obtained by standard Multilinear SVD, 2) in general, the inverse problem 2 is equivalent to solving a graph regularized matrix / tensor factorization problem where the factors V_μ belong to the span of the graph eigenvectors constructed from the modes of the tensor. Furthermore, to recover an MLRTG one should have large eigen gaps $\lambda_{\mu k^*}/\lambda_{\mu k^*+1}$. This occurs when the rows of the matricized $\boldsymbol{\mathcal{Y}}$ can be clustered into k^* clusters. However, note that for most of the applications, the optimal k^* is not known. In such cases it is suggested to select a $k > k^*$. Then, the low-rank tensor recovery error is characterized by the projection of factors V_μ^* on $(k - k^*)$ extra graph eigenvectors that are used. Our experiments in [4] show that selecting a $k > k^*$ always leads to a better recovery when the exact value of k^* is not known.

IV. Experiments

We compare the qualitative performance of the low-rank and sparse decomposition of matrices and tensors obtained by various methods. Fig. 1 presents experiments on the 2D real video datasets obtained from airport and shopping mall lobbies (every frame vectorized and stacked as the columns of a matrix). The goal is to separate the static low-rank component from the sparse part (moving people) in the videos. The results of TRPCAG are compared with RPCA [1], RPCAG [5], FRPCAG [6] and CPCA [7] with a downsampling factor of 5 along the frames. Clearly, TRPCAG recovers a low-rank which is qualitatively equivalent to the other methods in a time which is 100 times less than RPCA and RPCAG and an order of magnitude less as compared to FRPCAG. Furthermore, TRPCAG requires the same time as sampling based CPCA method but recovers a better quality low-rank structure. The performance quality of TRPCAG is also evident from the point cloud experiments in Fig. 2 where we recover the low-rank point clouds of dancing people and walking dog after adding sparse noise to them.

To show the scalability of TRPCAG as compared to TRPCA, we made a video of snowfall at the campus and tried to separate the snow-fall from the low-rank background via both methods. For this 1.5GB video of dimension $1920 \times 1080 \times 500$, TRPCAG (with core tensor size $100 \times 100 \times 50$) took less than 3 minutes, whereas TRPCA [2] did not converge even in 4 hours. The result obtained via TRPCAG is visualized in Fig. 3. For more experiments and the complete paper please refer to [4].

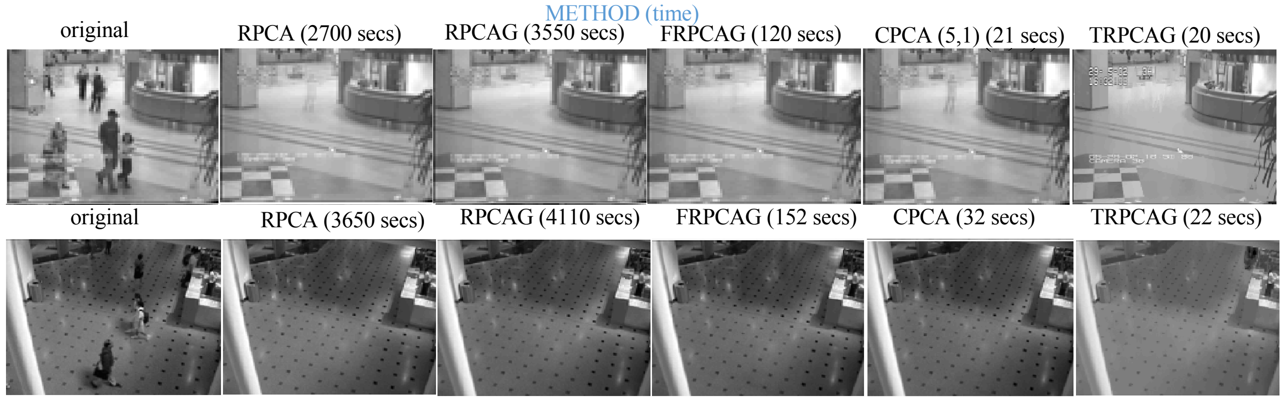

Figure 1. Performance analysis of TRPCAG for the 2D real video datasets: airport and shopping mall lobies. The goal is to separate the sparse component (moving people) from the static background. The rightmost frame in each row shows the result obtained via TRPCAG. The computation time for different methods is written on the top of frames. Clearly TRPCAG performs quite well while significantly reducing the computation time.

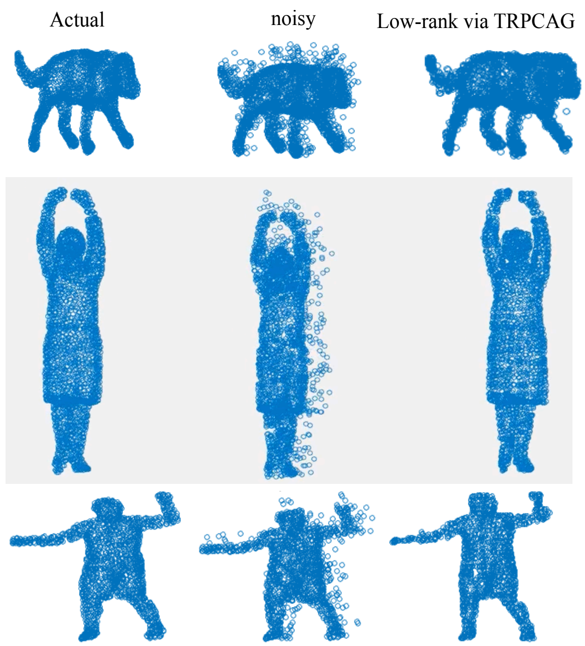

Figure 2. TRPCAG performance for recovering the low-rank point clouds of dancing people and a walking dog from the sparse noise. Actual point cloud (left), noisy point cloud (middle), recovered via TRPCAG (right).

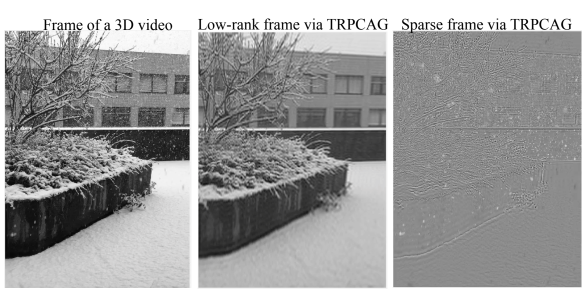

Figure 3. Low-rank recovery for a 3D video of dimension $1920 \times 1080 \times 500$ and size 1.5GB via TRPCAG. Using a core size of $100 \times 100 \times 50$, TRPCAG converged in less than 3 minutes.

REFERENCES

[1] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? Journal of the ACM (JACM), 58(3):11, 2011. 1

[2] D. Goldfarb and Z. Qin. Robust low-rank tensor recovery: Models and algorithms. SIAM Journal on Matrix Analysis and Applications, 35(1):225–253, 2014. 1

[3] M. Muja and D. Lowe. Scalable nearest neighbour algorithms for high dimensional data. 2014. 1

[4] N. Shahid, F. Grassi, and P. Vandergheynst. Multilinear low-rank tensors on graphs & applications. arXiv preprint arXiv:1611.04835, 2016. 1

[5] N. Shahid, V. Kalofolias, X. Bresson, M. Bronstein, and P. Vandergheynst. Robust principal component analysis on graphs. arXiv preprint arXiv:1504.06151, 2015. 1

[6] N. Shahid, N. Perraudin, V. Kalofolias, G. Puy, and P. Vandergheynst. Fast robust PCA on graphs. IEEE Journal of Selected Topics in Signal Processing, 10(4):740–756, 2016. 1

[7] N. Shahid, N. Perraudin, G. Puy, and P. Vandergheynst. Compressive pca for low-rank matrices on graphs. Technical report, 2016. 1

[8] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. Signal Processing Magazine, IEEE, 30(3):83–98, 2013. 1

[9] S. Si, D. Shin, I. S. Dhillon, and B. N. Parlett. Multi-scale spectral decomposition of massive graphs. In Advances in Neural Information Processing Systems, pages 2798–2806, 2014. 1