STARK: Structured Dictionary Learning Through Rank-one Tensor Recovery

Mohsen Ghassemi, Zahra Shakeri, Anand D. Sarwate, Waheed U. Bajwa
Dept. of Electrical and Computer Engineering, Rutgers University, Piscataway, NJ 08854
{mohsen.ghassemi, zahra.shakeri, anand.sarwate, waheed.bajwa}@rutgers.edu

***Abstract*—In recent years, a class of dictionaries have been proposed for multidimensional (tensor) data representation that exploit the structure of tensor data by imposing a Kronecker structure on the dictionary underlying the data. In this work, a novel algorithm called "STARK" is provided to learn Kronecker structured dictionaries that can represent tensors of any order. By establishing that the Kronecker product of any number of matrices can be rearranged to form a rank-1 tensor, we show that Kronecker structure can be enforced on the dictionary by solving a rank-1 tensor recovery problem. Because rank-1 tensor recovery is a challenging nonconvex problem, we resort to solving a convex relaxation of this problem. Empirical experiments on synthetic and real data show promising results for our proposed algorithm.**

I. Introduction

Sparse representations of data have been widely used in a variety of information processing tasks such as data compression, feature extraction, data classification, signal denoising and inpainting, and audio processing [1]–[3]. One of the powerful techniques to obtain sparse representations is dictionary learning (DL) which can be formulated as

$$\min_{\mathbf{D},\mathbf{X}} \ \frac{1}{2}\sum_{i=1}^{L} \|\mathbf{y}_i - \mathbf{D}\mathbf{x}_i\|_2^2, \quad \text{s.t.} \quad \forall i \ \ \|\mathbf{x}_i\|_0 \leq s. \qquad (1)$$

We wish to find an overcomplete basis $\mathbf{D} \in \mathbb{R}^{m\times p}$ with unit-norm columns and dictionary coefficient matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_L] \in \mathbb{R}^{p\times L}$ such that each observation $\mathbf{y}_i$ is represented by a linear combination of no more than s columns of $\mathbf{D}$. Since this problem is not convex, it is typically solved by alternating minimization; $\mathbf{X}$ is updated using a fixed $\mathbf{D}$ and then, $\mathbf{D}$ is updated using a fixed $\mathbf{X}$ [3].

In traditional DL literature, when dealing with multidimensional data, high order data $\{\underline{\mathbf{Y}}_i\}_{i=1}^{L}$ (tensors of order 2 or higher) are vectorized and stacked in columns of an observation matrix $\mathbf{Y} = [\text{vec}(\underline{\mathbf{Y}}_1), \cdots, \text{vec}(\underline{\mathbf{Y}}_L)]$: the structure of the data is not considered in the dictionary underlying the data. In this case, any standard DL method can be used to find sparse representations of data. This simplistic method disregards the multidimensional structure in the data and does not capture the correlation and structure along different dimensions in each original "data point."

On the other hand, in structured DL methods for tensor data, the multidimentional structure in data is taken into account. There exists a class of DL algorithms for tensor data that are based on the *Tucker decomposition* [4] of tensors. The resulting "Kronecker structured" DL methods (KS-DL) assume the dictionary consists of the Kronecker product [5] of smaller subdictionaries. Such algorithms represent tensor data using many fewer parameters compared to vectorized DL techniques [6]–[9]. This is due to the fact that the number of degrees of freedom in the KS-DL problem is significantly less than the traditional DL problem; this suggests that dictionary recovery is possible with smaller sample complexity using KS-DL methods [10], [11]. These works provide KS-DL algorithms to represent second order [6]–[8] and 3rd-order tensors [9]. In [6], for example, the KS-DL objective function is minimized using a Riemannian conjugate gradient method along with a nonmonotonic line search. In other methods, the subdictionaries composing the KS dictionary are updated alternately; in [7], an approach similar to K-SVD [3] is employed that uses higher-order SVD (HOSVD) [12] to alternately update coordinate dictionaries for second order tensor data and in [9], gradient descent is used to alternately update coordinate dictionaries for third order tensor data. The dictionary update stage in these algorithms involves solving a nonconvex minimization problem. The central challenge in the theoretical analysis of such KS-DL solvers is that the dictionary update stage is nonconvex. Furthermore, these explicit models are specialized to KS problems: they do not extend to more general structures in the underlying dictionary. In contrast, Dantas et al. [8] recently proposed an algorithm to learn the sum of KS dictionaries that represent second order tensor data by adding a regularizer in the objective function.

In this paper, we propose a novel algorithm called "STARK" to learn KS dictionaries for Nth-order tensor data ($N \geq 2$). Our method involves adding a regularization term to the objective function of the DL problem defined in (1). The motivation for this term comes from the following realization: elements of any KS matrix can be rearranged to form a rank-1 tensor. Thus, enforcing a rank-1 constraint on such rearrangement of the dictionary results in a KS dictionary. To this end, we take advantage of low-rank tensor estimation literature [13]–[17] to add a convex regularizer that imposes low-rankness on the rearrangement tensor. This formulation has the advantage that it can be used to learn KS dictionaries as well as the case where the underlying dictionary is better approximated by sum of KS dictionaries. Our method can learn dictionaries of arbitrary tensor order; in the case of second order tensor data our general formulation coincides with that of Danita's et al. [8].

We conduct numerical experiments to validate the performance of our algorithm. We use STARK for representation of

The work of the authors was supported in part by the National Science Foundation under awards CCF-1525276 and CCF-1453073, and by the Army Research Office under award W911NF-17-1-0546.

third order synthetic and real tensor data and demonstrate that STARK outperforms vectorized DL technique K-SVD [3] and KS-DL technique K-HOSVD [7] for small sample sizes.

Notation Convention: Underlined bold upper-case, bold upper-case and lower-case letters are used to denote tensors, matrices and vectors, respectively. Lower-case letters denote scalars. We denote the Kronecker product and outer product by $\otimes$ and $\circ$, respectively, while $\times_n$ denotes the mode n product between a tensor and a matrix [18]. Norms are given by subscripts, so $\|\mathbf{v}\|_0$ and $\|\mathbf{v}\|_2$ are the ℓ_0 and ℓ_2 norms of $\mathbf{v}$, while $\|\mathbf{X}\|_2$, $\|\mathbf{X}\|_F$, and $\|\mathbf{X}\|_*$ are the spectral, Frobenius, and nuclear norms of $\mathbf{X}$, respectively. A slice of a tensor is a 2-dimensional section defined by fixing all but two of its indices. Particularly, a frontal slice of a 3-dimensional tensor is defined by fixing the third index.

II. Tucker-Based KS-DL

According to the Tucker decomposition of tensors, an N-th order data tensor $\underline{\mathbf{Y}}_i \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_N}$ can be decomposed in the following form:

$$\underline{\mathbf{Y}}_i = \underline{\mathbf{X}}_i \times_1 \mathbf{D}_1 \times_2 \mathbf{D}_2 \times_3 \cdots \times_N \mathbf{D}_N, \tag{2}$$

where $\underline{\mathbf{X}}_i \in \mathbb{R}^{p_1 \times p_2 \times \cdots \times p_N}$ denotes the core tensor and $\mathbf{D}_i$'s denote transformation matrices along each mode of $\underline{\mathbf{Y}}_i$. The vectorized version of $\underline{\mathbf{Y}}_i$ can be written as

$$\mathbf{y}_i = (\mathbf{D}_N \otimes \mathbf{D}_{N-1} \otimes \cdots \otimes \mathbf{D}_1)\, \mathbf{x}_i, \tag{3}$$

where $\mathbf{y}_i = \mathrm{vec}(\underline{\mathbf{Y}}_i)$ and $\mathbf{x}_i = \mathrm{vec}(\underline{\mathbf{X}}_i)$ [18]. Here, the structure in tensor data is being exploited using the Kronecker product of transformation matrices. Stacking L vectorized data points $\{\mathbf{y}_i\}_{i=1}^L$ in columns of a matrix $\mathbf{Y}$, we get

$$\mathbf{Y} = (\mathbf{D}_N \otimes \mathbf{D}_{N-1} \otimes \cdots \otimes \mathbf{D}_1)\, \mathbf{X}, \tag{4}$$

which is similar to the conventional DL problem, except that the dictionary $\mathbf{D}$ is Kronecker structured.

In the next section, we present our proposed KS-DL algorithm called "STructured dictionAry learning through RanK-1 Tensor recovery" (STARK), which implicitly enforces Kronecker structure on the dictionary being learned by means of a regularizer in the DL objective function.

We note that STARK can be used to enforce a more general structure called *low-separation-rank (LSR)* structure. An LSR matrix can be written as sum of a few KS matrices:

$$\mathbf{D} = \sum_{k=1}^{K} \mathbf{D}_N^k \otimes \mathbf{D}_{N-1}^k \otimes \cdots \otimes \mathbf{D}_1^k, \tag{5}$$

where the factor matrices $\{\mathbf{D}_i^k \in \mathbb{R}^{m_i \times p_i}\}_{k=1}^K$ have the same size for a fixed i, and K is the *separation rank* of $\mathbf{D}$ [19].

III. Enforcing Structure via Regularization

To motivate the idea behind STARK, let us consider $\mathbf{D} = \mathbf{D}_1 \otimes \mathbf{D}_2$. It turns out that the elements of $\mathbf{D}$ can be rearranged to form $\mathbf{D}^\pi = \mathbf{d}_2 \circ \mathbf{d}_1$, where $\mathbf{d}_i = \mathrm{vec}(\mathbf{D}_i)$ for $i = 1, 2$ [5]. Figure 1 illustrates this rearrangement for $\mathbf{D}$. Similarly, for $\mathbf{D} = \mathbf{D}_1 \otimes \mathbf{D}_2 \otimes \mathbf{D}_3$, we can write $\underline{\mathbf{D}}^\pi = \mathbf{d}_3 \circ \mathbf{d}_2 \circ \mathbf{d}_1$, where each frontal slice of the tensor $\underline{\mathbf{D}}^\pi$ is a scaled copy of $\mathbf{d}_3 \circ \mathbf{d}_2$. Following a similar procedure, we can show that if $\mathbf{D} = \sum_{k=1}^K \mathbf{D}_1^k \otimes \mathbf{D}_2^k \otimes \cdots \otimes \mathbf{D}_N^k$, then a certain "rearrangement" of $\mathbf{D}$ is the rank-K tensor $\underline{\mathbf{D}}^\pi = \sum_{k=1}^K \mathbf{d}_N^k \circ \mathbf{d}_{n-1}^k \circ \cdots \circ \mathbf{d}_1^k$, where $\mathbf{d}_i = \mathrm{vec}(\mathbf{D}_i)$ for $i \in [N]$. This suggests that in the structured DL problem, we can impose the LSR structure (KS when $K = 1$) on the dictionary $\mathbf{D}$ being learned by minimizing the rank of $\underline{\mathbf{D}}^\pi$.

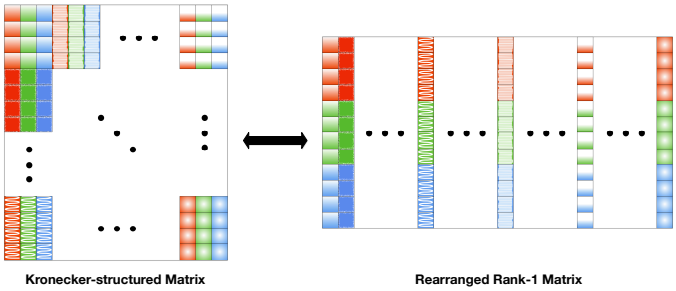

Fig. 1: Example of rearranging a KS matrix into a rank-1 matrix.

Since tensor rank is a nonconvex function, in order to make this DL problem convex with respect to $\mathbf{D}$, we use a commonly used convex proxy for the tensor rank function, the *sum-trace-norm* [15], which is defined as the average of the trace (nuclear) norms of the *unfoldings* of the tensor: $\|\underline{\mathbf{D}}\|_{\mathrm{tr}} = \frac{1}{N}\sum_{n=1}^N \left\|\mathbf{D}_{(n)}\right\|_*$. Using this convex relaxation for the rank function, the KS-DL problem has the following form:

$$\min_{\mathbf{D},\mathbf{X}} \ \frac{1}{2}\|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 + \lambda \|\underline{\mathbf{D}}^\pi\|_{\mathrm{tr}}, \ \text{ s.t. } \forall i \ \ \|\mathbf{x}_i\|_0 \le s, \tag{6}$$

where the columns of $\mathbf{D}$ have unit norm. We use alternating minimization to solve this nonconvex problem. To minimize the objective function in (6) with respect to $\mathbf{X}$, we can use any of the standard sparse coding methods. In simulations, we use orthogonal matching pursuit (OMP) [20], [21]. To update the KS dictionary $\mathbf{D}$, we use the alternating direction method of multipliers (ADMM) [22]. We describe this dictionary update step in the next section.

IV. Structured Dictionary Update Using ADMM

In this section, we discuss the dictionary update step of solving problem (6), which can be stated as

$$\min_{\mathbf{D}\in\mathbb{R}^{m\times p}} \ \frac{1}{2}\|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 + \lambda \sum_{n=1}^N \left\|\mathbf{D}_{(n)}^\pi\right\|_*. \tag{7}$$

The main issue in solving the convex dictionary update problem (7) is dealing with the interdependent nuclear norms. This makes optimization methods that use gradient information challenging. Inspired by many works in the literature on low-rank tensor estimation [13]–[16], we instead suggest the following reformulation of (7):

$$\begin{aligned} \min_{\mathbf{D},\underline{\mathbf{W}}_1,\cdots,\underline{\mathbf{W}}_N} \ & \frac{1}{2}\|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 + \lambda \sum_{n=1}^N \left\|(\mathbf{W}_n)_{(n)}\right\|_* \\ \text{s.t.} \quad & \forall n \quad \underline{\mathbf{W}}_n = \underline{\mathbf{D}}^\pi. \end{aligned} \tag{8}$$

In this formulation, although the nuclear norms are associated with one another through the introduced constraint, we can

decouple the minimization problem into separate subproblems. In particular, we can solve the objective function (8) using ADMM, which involves decoupling the problem into independent subproblems by forming the following augmented Lagrangian function:

$$\mathcal{L}_\gamma(\underline{\mathbf{D}}^\pi, \widetilde{\underline{\mathbf{W}}}, \widetilde{\underline{\mathbf{A}}}) = \frac{1}{2}\|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 + \sum_{n=1}^{N}\Big(\lambda \left\|(\mathbf{W}_n)_{(n)}\right\|_* - \langle \underline{\mathbf{A}}_n,\ \underline{\mathbf{D}}^\pi - \underline{\mathbf{W}}_n\rangle + \frac{\gamma}{2}\|\ \underline{\mathbf{D}}^\pi - \underline{\mathbf{W}}_n\|_F^2\Big), \quad (9)$$

where $\widetilde{\underline{\mathbf{W}}} = \left[\underline{\mathbf{W}}_1^T, \cdots,\ \underline{\mathbf{W}}_N^T\right]^T$ and $\widetilde{\underline{\mathbf{A}}} = \left[\underline{\mathbf{A}}_1^T, \cdots,\ \underline{\mathbf{A}}_N^T\right]^T$. Here, the inner product of two tensors is defined as the inner product of their vectorizations.

Finally, to find the gradient of (9) with respect to $\underline{\mathbf{D}}^\pi$, we rewrite the Lagrangian function in the following form

$$\mathcal{L}_\gamma(\underline{\mathbf{D}}^\pi, \widetilde{\underline{\mathbf{W}}}, \widetilde{\underline{\mathbf{A}}}) = \frac{1}{2}\|\mathbf{y} - \mathcal{T}(\underline{\mathbf{D}}^\pi)\|_2^2 + \sum_{n=1}^{N}\Big(\lambda \left\|(\mathbf{W}_n)_{(n)}\right\|_* - \langle \underline{\mathbf{A}}_n,\ \underline{\mathbf{D}}^\pi - \underline{\mathbf{W}}_n\rangle + \frac{\gamma}{2}\|\ \underline{\mathbf{D}}^\pi - \underline{\mathbf{W}}_n\|_F^2\Big). \quad (10)$$

Here, we defined $\mathbf{y} = \mathrm{vec}(\mathbf{Y})$ and $\mathcal{T}(\underline{\mathbf{D}}^\pi) = \mathrm{vec}(\mathbf{D}\mathbf{X}) = \widetilde{\mathbf{X}}^T\mathbf{\Pi}\,\mathrm{vec}(\underline{\mathbf{D}}^\pi)$, where $\widetilde{\mathbf{X}} = \mathbf{X}\otimes\mathbf{I}_m$ and $\mathbf{\Pi}$ is a permutation matrix such that $\mathbf{\Pi}\,\mathrm{vec}(\underline{\mathbf{D}}^\pi) = \mathrm{vec}(\mathbf{D})$.

In the rest of this section, we briefly discuss derivation of the permutation matrix as well as the update steps of ADMM. Due to lack of space, we leave the details to an extended version of this paper.

A. The Permutation Matrix

The permutation matrix $\mathbf{\Pi}$ represents a linear transformation that maps the elements of $\mathrm{vec}(\mathbf{D})$ to $\mathrm{vec}(\underline{\mathbf{D}}^\pi)$. Given index l of $\mathrm{vec}(\mathbf{D})$ and the corresponding mapped index l' of $\mathrm{vec}(\underline{\mathbf{D}}^\pi)$, our strategy for finding the permutation matrix is to define l' as a function of l. To this end, we first find the the corresponding row and column indices (i, j) of matrix $\mathbf{D}$ from the lth element of $\mathrm{vec}(\mathbf{D})$. Then, we find the index of the element of interest on the Nth order rearranged tensor $\underline{\mathbf{D}}^\pi$, and finally, we find its location l' on $\mathrm{vec}(\underline{\mathbf{D}}^\pi)$. Note that the permutation matrix is only a function of the dimensions of the factor matrices. We leave the formal explanation of this procedure to an extended version of this paper.

B. ADMM Update Rules

Recall that each iteration of ADMM consists of the following steps [22]:

$$\underline{\mathbf{D}}^\pi(t+1) = \underset{\underline{\mathbf{D}}^\pi}{\mathrm{argmin}}\ \mathcal{L}_\gamma(\underline{\mathbf{D}}^\pi, \widetilde{\underline{\mathbf{W}}}(t), \widetilde{\underline{\mathbf{A}}}(t)), \quad (11)$$

$$\widetilde{\underline{\mathbf{W}}}(t+1) = \underset{\widetilde{\underline{\mathbf{W}}}}{\mathrm{argmin}}\ \mathcal{L}_\gamma(\underline{\mathbf{D}}^\pi(t+1), \widetilde{\underline{\mathbf{W}}}, \widetilde{\underline{\mathbf{A}}}(t)), \quad (12)$$

$$\widetilde{\underline{\mathbf{A}}}(t+1) = \widetilde{\underline{\mathbf{A}}}(t) - \gamma\left(\underline{\mathbf{H}}\underline{\mathbf{D}}^\pi(t+1) - \widetilde{\underline{\mathbf{W}}}(t+1)\right), \quad (13)$$

where $\underline{\mathbf{H}}$ is the vertical concatenation of N instances of $\underline{\mathbf{I}}$, the identity operator on $\in \mathbb{R}^{m_1p_1\times\cdots\times m_Np_N}$.

The solution to problem (11) is found by taking the gradient of $\mathcal{L}_\gamma(\cdot)$ with respect to $\underline{\mathbf{D}}^\pi$ and setting it to zero. Suppressing the iteration index t for ease of notation, we have

$$\frac{\partial\mathcal{L}_\gamma}{\partial\underline{\mathbf{D}}^\pi} = \mathcal{T}^*(\mathcal{T}(\underline{\mathbf{D}}^\pi) - \mathbf{y}) - \sum_{n=1}^{N}\underline{\mathbf{A}}_n + \sum_{n=1}^{N}\gamma\left(\underline{\mathbf{D}}^\pi - \underline{\mathbf{W}}_n\right),$$

where $\mathcal{T}^*$ denotes the adjoint of the linear operator $\mathcal{T}$ [13]. Setting the gradient to zero results in

$$\mathcal{T}^*(\mathcal{T}(\underline{\mathbf{D}}^\pi)) + \gamma N\ \underline{\mathbf{D}}^\pi = \mathcal{T}^*(\mathbf{y}) + \sum_{n=1}^{N}\left(\underline{\mathbf{A}}_n + \gamma\underline{\mathbf{W}}_n\right). \quad (14)$$

Equivalently, we have,

$$\mathrm{vec}^{-1}\left(\left[\mathbf{\Pi}^T\widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T\mathbf{\Pi} + \gamma N\mathbf{I}\right]\mathrm{vec}(\underline{\mathbf{D}}^\pi)\right) = \mathrm{vec}^{-1}(\mathbf{\Pi}^T\widetilde{\mathbf{X}}\mathbf{y}) + \sum_{n=1}^{N}\left(\underline{\mathbf{A}}_n + \gamma\underline{\mathbf{W}}_n\right). \quad (15)$$

Therefore, the update rule for $\underline{\mathbf{D}}^\pi$ is

$$\underline{\mathbf{D}}^\pi(t+1) = \mathrm{vec}^{-1}\Big(\left[\mathbf{\Pi}^T\widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^T\mathbf{\Pi} + \gamma N\mathbf{I}_{mp}\right]^{-1} \cdot\left[\mathbf{\Pi}^T\widetilde{\mathbf{X}}\mathbf{y} + \mathrm{vec}\left(\sum_{n=1}^{N}\left(\underline{\mathbf{A}}_n(t) + \gamma\underline{\mathbf{W}}_n(t)\right)\right)\right]\Big). \quad (16)$$

To update $\widetilde{\underline{\mathbf{W}}}$, we can break the second step (12) into solving N independent subproblems (suppressing the index t):

$$\min_{\underline{\mathbf{W}}_n}\ \mathcal{L}_\mathcal{W} = \lambda\left\|(\mathbf{W}_n)_{(n)}\right\|_* - \langle\underline{\mathbf{A}}_n,\ \underline{\mathbf{D}}^\pi - \underline{\mathbf{W}}_n\rangle + \frac{\gamma}{2}\|\ \underline{\mathbf{D}}^\pi - \underline{\mathbf{W}}_n\|_F^2.$$

The objective function of this problem can be reformulated as

$$\mathcal{L}_\mathcal{W} = \lambda\left\|(\mathbf{W}_n)_{(n)}\right\|_* + \frac{\gamma}{2}\left\|(\mathbf{W}_n)_{(n)} - \left(\underline{\mathbf{D}}^\pi_{(n)} - \frac{(\mathbf{A}_n)_{(n)}}{\gamma}\right)\right\|_F^2 + \mathrm{const}. \quad (17)$$

The minimizer of an objective function of the form (17) is

$$\mathrm{shrink}\left((\mathbf{D}^\pi)_{(n)} - \frac{1}{\gamma}(\mathbf{A}_n)_{(n)},\ \frac{\lambda}{\gamma}\right), \quad (18)$$

where $\mathrm{shrink}(\mathbf{M}, \tau)$ is the shrinkage operator that applies soft-thresholding at level τ on the singular values of M (see Theorem 2.1 in [23] for details). Therefore,

$$\underline{\mathbf{W}}_n(t+1) = \mathrm{refold}\Big(\mathrm{shrink}\Big(\Big(\mathbf{D}^\pi(t+1)\Big)_{(n)} - \frac{1}{\gamma}\left(\mathbf{A}_n(t)\right)_{(n)},\ \frac{\lambda}{\gamma}\Big)\Big). \quad (19)$$

where $\mathrm{refold}(\cdot)$ is the inverse of the unfolding operator. The summary of our DL method is provided in Algorithm 1.

V. Numerical Experiments

We compare the performance of STARK with two methods: KSVD [3], as a non-structured DL method, and K-HOSVD [7], which is a structured DL method that explicitly enforces Kronecker structure on the dictionary. We compare the performance of these methods are compared for synthetic 2-dimensional and 3-dimensional data as well as 3-dimensional real-world data.

a) Synthetic Data: For synthetic data, we randomly generate the dictionary $\mathbf{D}$ and the sparse coefficient matrix $\mathbf{X}$ to construct the observation matrix $\mathbf{Y} = \mathbf{D}\mathbf{X}$. We generate a KS dictionary as $\mathbf{D} = \mathbf{D}_1 \otimes \mathbf{D}_2 \otimes \mathbf{D}_3$ (and $\mathbf{D} = \mathbf{D}_1 \otimes \mathbf{D}_2$ for 2-dimensional data) with unit-norm columns according to the following procedure. The elements of the subdictionaries $\mathbf{D}_1$ through $\mathbf{D}_3$ are chosen i.i.d from a Gaussian distribution $\mathcal{N}(0,1)$, and then the columns of the subdictionaries are normalized. For generating $\mathbf{X}$, we select the locations of the s nonzero elements of each column uniformly at random. The values of those elements are sampled i.i.d. from $\mathcal{N}(0,1)$. In the learning process, the dictionary $\mathbf{D}$ is initialized using random columns of the observation matrix $\mathbf{Y}$. The experiments were run for 20 Monte Carlo iterations and for various training sample sizes and the resulting dictionaries were tested on a set of 10000 test samples. For 2nd-order tensor data we selected $m_1 = 4$, $p_1 = 12$, $m_2 = 6$, $p_2 = 8$, $s = 5$, and for 3rd-order tensor data we selected $m_1 = 2$, $p_1 = 4$, $m_2 = 5$, $p_2 = 10$, $m_3 = 5$, $p_3 = 5$, and $s = 10$.

The results of our experiments on synthetic data are shown in Figure 2. We compared our method to K-SVD and K-HOSVD. For K-HOSVD, we used algorithm provided by the authors, which actually enforces a Khatri-Rao structure on the dictionary rather than KS. STARK outperforms both K-SVD and K-HOSVD for all training sample sizes, especially when the number of training samples is small. The improvement over K-SVD can be attributed to the lower sample complexity of structured DL models. We conjecture that one reason for the improvement over K-HOSVD is that the dictionary update in STARK is a convex problem and thus the algorithm is less prone to getting stuck in a poor local minimum.

b) Real Data: For these experiments, we compare the denoising performance of the three methods on two RGB images, `Peppers` and `Lena`, which are $512 \times 512 \times 2$ and $512 \times 512 \times 3$ tensors, respectively. We corrupt the images using additive white Gaussian noise with $\sigma = 50$. To construct the training data set, we extract overlapping 6×6 patches from each image and treat each patch as a data point. Then we compare the denoising performances of the methods based on the resulting peak signal to noise ratio (PSNR) of the reconstructed images [24].

Algorithm 1 Structured Dictionary Learning through Rank-1 Tensor Recovery (STARK)

Require: $\mathbf{Y}$, $\mathbf{\Pi}$, $s > 0$, $\lambda > 0$, $\gamma > 0$
initialize: $\mathbf{D}(0)$, $\mathbf{X}(0)$, , $\underline{\widetilde{\mathbf{A}}}(0)$, $\underline{\widetilde{\mathbf{W}}}(0)$
while $\|\mathbf{Y} - \mathbf{D}(\tau)\mathbf{X}(\tau)\|_F > \epsilon$ **do**
  Sparse coding stage: Use OMP to update $\mathbf{X}(\tau)$.
  Dictionary update stage:
  while $\left\|\underline{\widetilde{\mathbf{A}}}(t) - \underline{\widetilde{\mathbf{A}}}(t-1)\right\|_F > \epsilon$ **do**
    Update $\underline{\mathbf{D}}^{\pi}(t)$ according to update rule (16)
    for all $i \in [N]$ **do**
      Update $\underline{\widetilde{\mathbf{W}}}_n(t)$ according to update rule (19)
    end for
    for all $n \in [N]$ **do**
      $\underline{\mathbf{A}}_n(t+1) = \underline{\mathbf{A}}_n(t) - \gamma\left(\underline{\mathbf{D}}^{\pi}(t+1) - \underline{\mathbf{W}}_n(t+1)\right)$
    end for
  end while
  Normalize columns of $\mathbf{D}(\tau)$
end while
return $\mathbf{D}(\tau)$, $\mathbf{X}(\tau)$

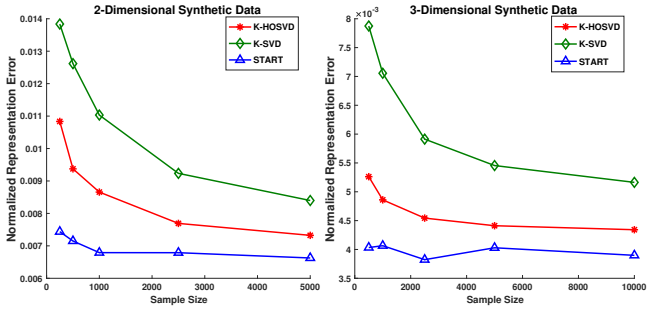

Fig. 2: Normalized Representation Error on Synthetic Data.

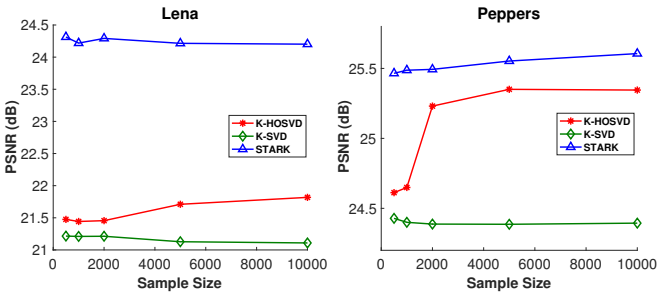

Fig. 3: Denoising Performance on Real Data (PSNR).

Figure 3 shows the results averaged over 10 Monte Carlo iterations. We can see the denoising performance of STARK is superior to both K-SVD and K-HOSVD for all training sample sizes. This is in part due to the fact that for real-world data, the underlying dictionaries may not be KS. STARK allows $\underline{\mathbf{D}}^{\pi}$ to have rank higher than 1, meaning the algorithm can use a larger number of parameters (K times as many when $\mathrm{rank}(\underline{\mathbf{D}}^{\pi}) = K$) to approximate the true dictionary.

VI. Conclusion

In this paper we showed that the Kronecker product of N matrices can be rearranged to form an Nth order rank-1 tensor. Based on this, we proposed a novel structured dictionary learning method for multidimensional data that enforces LSR structure in the dictionary through imposing a low-rank structure on the rearranged tensor. In particular, Kronecker structure can be enforced by imposing a rank-1 constraint on the rearranged tensor. Since the low-rank tensor recovery problem is a nonconvex problem, we resort to solving its convex relaxation, namely, minimizing the sum-trace-norm of the rearranged tensor, which is a convex proxy for tensor rank. We used ADMM for solving the dictionary update stage of this structured DL problem. Our experiments on both synthetic and real data showed that when the sample size is small, our method considerably outperforms both K-SVD, which returns unstructured dictionaries, and K-HOSVD, a KS-DL method that directly finds the subdictionaries.

REFERENCES

[1] K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, T.-W. Lee, and T. J. Sejnowski, "Dictionary learning algorithms for sparse representation," *Neural computation*, vol. 15, no. 2, pp. 349–396, February 2003. [Online]. Available: https://doi.org/10.1162/089976603762552951

[2] M. Elad, J.-L. Starck, P. Querre, and D. L. Donoho, "Simultaneous cartoon and texture image inpainting using morphological component analysis (MCA)," *Appl. and Computational Harmonic Anal.*, vol. 19, no. 3, pp. 340–358, November 2005. [Online]. Available: https://doi.org/10.1016/j.acha.2005.03.005

[3] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. Signal Process.*, vol. 54, no. 11, pp. 4311–4322, November 2006. [Online]. Available: https://doi.org/10.1109/TSP.2006.881199

[4] L. R. Tucker, "Implications of factor analysis of three-way matrices for measurement of change," *Problems in Measuring Change*, pp. 122–137, 1963.

[5] C. F. Van Loan, "The ubiquitous Kronecker product," *J. Computational and Appl. Math.*, vol. 123, no. 1, pp. 85–100, November 2000. [Online]. Available: https://doi.org/10.1016/S0377-0427(00)00393-9

[6] S. Hawe, M. Seibert, and M. Kleinsteuber, "Separable dictionary learning," in *Proc. IEEE Conf. Comput. Vision and Pattern Recognition (CVPR)*, June 2013, pp. 438–445. [Online]. Available: https://doi.org/10.1109/CVPR.2013.63

[7] F. Roemer, G. Del Galdo, and M. Haardt, "Tensor-based algorithms for learning multidimensional separable dictionaries," in *Proc. IEEE Int. Conf. Acoustics, Speech and Signal Process. (ICASSP)*, May 2014, pp. 3963–3967. [Online]. Available: https://doi.org/10.1109/ICASSP.2014.6854345

[8] C. F. Dantas, M. N. da Costa, and R. da Rocha Lopes, "Learning dictionaries as a sum of Kronecker products," *IEEE Signal Process. Lett.*, vol. 24, no. 5, pp. 559–563, March 2017. [Online]. Available: https://doi.org/10.1109/LSP.2017.2681159

[9] S. Zubair and W. Wang, "Tensor dictionary learning with sparse Tucker decomposition," in *Proc. IEEE 18th Int. Conf. Digital Signal Process. (DSP)*, July 2013, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ICDSP.2013.6622725

[10] Z. Shakeri, W. U. Bajwa, and A. D. Sarwate, "Minimax lower bounds for Kronecker-structured dictionary learning," in *Proc. 2016 IEEE Int. Symp. Inf. Theory*, July 2016, pp. 1148–1152. [Online]. Available: https://doi.org/10.1109/ISIT.2016.7541479

[11] ——, "Minimax lower bounds on dictionary learning for tensor data," *arXiv preprint arXiv:1608.02792*, August 2016. [Online]. Available: https://arxiv.org/abs/1608.02792

[12] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM J. Matrix Analy. and Applicat.*, vol. 21, no. 4, pp. 1253–1278, 2000. [Online]. Available: https://doi.org/10.1137/S0895479896305696

[13] S. Gandy, B. Recht, and I. Yamada, "Tensor completion and low-n-rank tensor recovery via convex optimization," *Inverse Problems*, vol. 27, no. 2, p. 025010, January 2011. [Online]. Available: https://doi.org/10.1088/0266-5611/27/2/025010

[14] B. Romera-Paredes, H. Aung, N. Bianchi-Berthouze, and M. Pontil, "Multilinear multitask learning," in *Proc. 30th Int. Conf. Mach. Learn. (ICML)*, vol. 28, no. 3, Atlanta, Georgia, USA, June 2013, pp. 1444–1452. [Online]. Available: http://proceedings.mlr.press/v28/romera-paredes13.html

[15] K. Wimalawarne, M. Sugiyama, and R. Tomioka, "Multitask learning meets tensor factorization: Task imputation via convex optimization," in *Proc. Advances in Neural Inform. Process. Syst. (NIPS)*, 2014, pp. 2825–2833. [Online]. Available: http://dl.acm.org/citation.cfm?id=2969033.2969142

[16] B. Huang, C. Mu, D. Goldfarb, and J. Wright, "Provable low-rank tensor recovery," *Optimization-Online*, vol. 4252, p. 2, February 2014. [Online]. Available: http://www.optimization-online.org/DB_FILE/2014/02/4252.pdf

[17] J. Liu, P. Musialski, P. Wonka, and J. Ye, "Tensor completion for estimating missing values in visual data," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 1, pp. 208–220, January 2013. [Online]. Available: http://doi.org/10.1109/TPAMI.2012.39

[18] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, August 2009. [Online]. Available: https://doi.org/10.1137/07070111X

[19] T. Tsiligkaridis and A. O. Hero, "Covariance estimation in high dimensions via Kronecker product expansions," *IEEE Trans. Signal Process.*, vol. 61, no. 21, pp. 5347–5360, November 2013. [Online]. Available: https://doi.org/10.1109/TSP.2013.2279355

[20] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," in *Proc. 27th Asilomar Conf. Signals, Syst. and Comput.*, November 1993, pp. 40–44 vol.1. [Online]. Available: https://doi.org/10.1109/ACSSC.1993.342465

[21] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inf. Theory*, vol. 53, no. 12, pp. 4655–4666, December 2007. [Online]. Available: https://doi.org/10.1109/TIT.2007.909108

[22] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, January 2011. [Online]. Available: https://doi.org/10.1561/2200000016

[23] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," *SIAM J. Optimization*, vol. 20, no. 4, pp. 1956–1982, March 2010. [Online]. Available: https://doi.org/10.1137/080738970

[24] A. Hore and D. Ziou, "Image quality metrics: PSNR vs. SSIM," in *Proc. IEEE int. conf. Pattern recognition (ICPR)*, August 2010, pp. 2366–2369. [Online]. Available: https://doi.org/10.1109/ICPR.2010.579