Multi-Dictionary Tensor Decomposition

Maxwell McNeil and Petko Bogdanov
Computer Science, University at Albany- SUNY
{mmcneil2,pbogdanov}@albany.edu

***Abstract*—Tensor decomposition methods are popular tools for analysis of multi-way datasets from the social media, healthcare, spatio-temporal domains, and others. Widely adopted models such as Tucker and canonical polyadic decomposition (CPD) follow a data-driven philosophy: they decompose a tensor into factors that approximate the observed data well. In some cases side information is available about the tensor modes. For example, in a temporal user-item purchases tensor a user influence graph, an item similarity graph, and knowledge about seasonality or trends in the temporal mode may be available. Such side information may enable more succinct and interpretable tensor decomposition models and improved quality in downstream tasks.**

We propose a framework for Multi-Dictionary Tensor Decomposition (MDTD) which takes advantage of prior structural information about tensor modes in the form of coding dictionaries to obtain sparsely coded tensor factors. We derive a general optimization algorithm for MDTD that handles both complete inputs and inputs with missing values. MDTD handles large sparse tensors typical in many real-world application domains. We experimentally demonstrate its utility in both synthetic and real-world datasets. It learns more concise models than dictionary-free counterparts and improves (i) reconstruction quality (up to 60% smaller models coupled with reduced representation error); (ii) missing values imputation quality (two-fold MSE reduction with up to orders of magnitude time savings) and (iii) the estimation of the tensor rank. MDTD's quality improvements do not come with a running time premium: it can decompose $19GB$ datasets in less than a minute. It can also impute missing values in sparse billion-entry tensors more accurately and scalably than state-of-the-art competitors.

I. Introduction

Tensors are multi-way arrays that generalize matrix data to higher number of "dimensions" [26]. The ability of tensors to accurately model the complex relationships present in many datasets has rendered them applicable in signal processing [27], machine learning [22], chemometrics [3], and other fields. Similar to matrices, low rank decomposition models for tensors are common ways of finding patterns in multi-way data. Popular approaches like the Canonical polyadic decomposition (CPD) [3] and Tucker decomposition [29] learn directly from data without additional modeling assumptions. In many settings prior knowledge about the data generation process may also be available, for example, seasonality in a temporal mode or a network associating individuals in a user mode. In addition, downstream applications such as data imputation, clustering, and anomaly detection may benefit from imposing structure in the decomposition. Such considerations have given rise to modifications to the original CPD and Tucker models that have produced state-of-the-art performance in missing values imputation within a Bayesian framework [4], [5], improved community detection for on/off [11], periodic [16] or bursty self-exciting behavior [12], and other tasks.

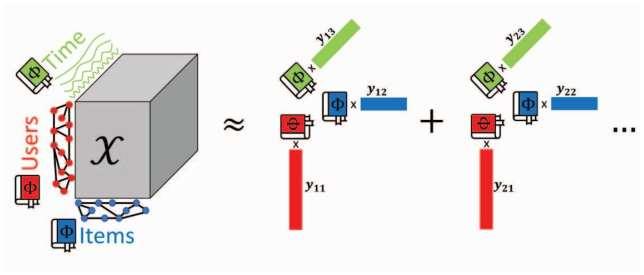


Fig. 1: The key idea behind the dictionary-based tensor decomposition model (MDTD) through a user-item-time example. MDTD can utilize graph-based dictionaries Φ for the user and item modes and a temporal dictionary for the temporal mode. The decomposition is similar to CPD decomposition in that it is a sum of rank-one factor tensors, with the key difference that factors are represented as encodings y_{ij} via the corresponding dictionaries.

Most methods employ regularization to build prior knowledge into the factorization model imposing different forms of structure: sparsity, periodicity and others. An alternative approach is to employ sparse coding for tensor factors via dictionaries [6]. Such sparse coding techniques utilize fixed dictionaries and have been widely adopted in signal and graph signal processing [20], [28], computer vision [9], machine learning [13] and data analytics [19]. The ubiquitous applications of such methods have also given rise to some standard analytical dictionaries for time series (Fourier, Ramanujan, splines) [28], graphs (graph Fourier and graph wavelets) [20], and images (wavelets, ridgelets, curvelets) [9]. Employing such dictionaries for tensor data promises to enable succinct, interpretable and efficient-to-learn models.

We introduce a multi-dictionary tensor factorization (MDTD) framework that employs fixed dictionaries for joint sparse coding of the tensor factors. The key idea of our model is illustrated via a user-item-time example tensor in Fig. 1. Given prior knowledge in the form of user and item graphs as well as expectation about periodic behavior in time, we propose to employ corresponding dictionaries Φ to sparsely encode factors in a CPD-like model. For the example in the figure, we can employ a Graph Fourier Transform (GFT) dictionary for the modes with graph side information and a periodic dictionary for the temporal mode. The model is applicable to higher order tensors with any subset of modes endowed with side information, as well as to other kinds of side information and corresponding dictionaries. We propose a general optimization solution for MDTD and evaluate it on multiple tensor datasets. We demonstrate that when the

979-8-3503-0788-7/23/$31.00 ©2023 IEEE
DOI 10.1109/ICDM58522.2023.00151

side information captured by the dictionaries is well aligned with the data in the tensor, our approach enables i) orders of magnitude reduction in the model size compared to CPD and Tucker, while running in comparable time and ii) enables improved quality in several downstream tasks.

Our contributions in this paper are as follows:

• **Generality and Novelty:** We propose the first multi-dictionary tensor decomposition framework MDTD which can leverage arbitrary dictionaries for each tensor mode.

• **Parsimony and Scalability:** MDTD produces interpretable and concise representations of both real-world and synthetic tensors scaling similar to simple decomposition models and better than more complex ones. MDTD decomposes a $19GB$ tensors in 1 min and can impute missing values in tensors with billions more entries than what competitors can handle.

• **Applicability and Accuracy:** We demonstrate MDTD's utility for succinct tensor representation, and missing value imputation. Its quality dominates baselines across applications and datasets. In some cases MDTD achieves higher accuracy and a 100x speed-up compared to the fastest baseline.

II. Preliminaries

Before we define our problem of dictionary-based tensor decomposition (MDTD), we first introduce necessary preliminaries and notation. The input to our problem is a tensor $\mathcal{X}$, which is a multi-dimensional array of real numbers. We present the problem and our solutions in the context of three-way tensors for simplicity, however, both generalize seamlessly to higher order tensors. We will work with tensors of the following shape $\mathcal{X} \in \mathbb{R}^{(I\times J\times T)}$, where I, J and T are the dimensions of the modes.

CPD decomposition. MDTD can be viewed as a dictionary-based extension of the CPD decomposition of the form:

$$\mathcal{X} = \sum_{i=1}^{k} \mathcal{H}_i = \sum_{i=1}^{k} a_i \boxtimes b_i \boxtimes c_i, \tag{1}$$

where $\boxtimes$ denotes the tensor outer product and $\mathcal{H}_i$ are rank-one tensors obtained from outer tensor products of individual factors $a_i \in \mathbb{R}^I, b_i \in \mathbb{R}^J$, and $c_i \in \mathbb{R}^T$. If we stack k factor vectors a_i, b_i, and c_i into matrices $A \in \mathbb{R}^{I\times k}, B \in \mathbb{R}^{J\times k}$, and $C \in \mathbb{R}^{T\times k}$ respectively, we can express this relationship concisely as: $\mathcal{X} = [[A, B, C]]$. An in-depth introduction of CPD and other tensor models is available in [26].

Sparse dictionary coding or sparse representation modeling [24] assumes that the data can be represented via a linear combination of a few atoms from an appropriately-chosen pre-specified dictionary Φ, where both analytical and dictionaries learned from data can be employed. In its general form sparse coding solves the following problem:

$$\min_y f(y) \quad \text{s.t.} \quad x = \Phi y,$$

where x is an input signal, y is its encoding and $f(y)$ is a sparsity promoting function often instantiated as an L_1 norm.

III. Problem Formulation and Solution

In many real-world application there is a structural information associated with tensor modes. Consider, for example, users (mode 1) watching streams (mode 2) over time (mode 3) on a stream service such as Twitch. Such data can be represented by a binary tensor $\mathcal{X} \in \mathbb{R}^{(I\times J\times T)}$. It is easy to imagine that users may be associated within a friendship network and streams within a topical similarity network. More over the communities within those networks (friendship groups interested in streams featuring similar games) will likely stream based on regular daily/weakly patterns. *How can we leverage this rich structural information to learn a succinct, interpretable, and meaningful representation of the data?*

We propose to represent a tensor with structural side information through a CPD-like dictionary-based decomposition:

$$\mathcal{X} = \sum_{n=1}^{k} \Phi_1 y_{n1} \boxtimes \Phi_2 y_{n2} \boxtimes \Phi_3 y_{n3} = [[\Phi_1 Y_1, \Phi_2 Y_2, \Phi_3 Y_3]],$$

where prior knowledge in each mode is incorporated as a model-specific dictionary Φ_i and the sparse encoding of the input data through dictionaries is in matrices Y_i. Fitting the input data to such a model results in the following problem:

$$\min_{Y_1,Y_2,Y_3} \frac{1}{2}||\mathcal{X} - [[\Phi_1 Y_1, \Phi_2 Y_2, \Phi_3 Y_3]]||_F^2 + \sum_{i=1}^{3} \lambda_i \left\|Y_i\right\|_1,$$

where the first term is the data fit and the second term encourages sparsity in encodings Y_i in the form of an L_1 regularization. This form of sparsity is typical when using dictionaries to avoid overfitting and ill-posed problems. Increasing the sparsity balance parameters λ_i encourages sparser solutions for corresponding modes and allows us to control the complexity/size of the learned model. To prevent the model from fitting missing/unobserved values we also introduce a zero-one mask Ω which is a tensor of the same size as $\mathcal{X}$. Our overall MDTD objective is:

$$\min_{Y_1,Y_2,Y_3} \frac{1}{2}||\Omega \boxdot (\mathcal{X} - [[\Phi_1 Y_1, \Phi_2 Y_2, \Phi_3 Y_3]])||_F^2 + \sum_{i=1}^{3} \lambda_i \left\|Y_i\right\|_1, \tag{2}$$

where $\boxdot$ denotes the element-wise product. It is important to note that if a dictionary (or side information) is not available for some of the modes in a given application, a trivial identity dictionary $\Phi_i = I$ and a corresponding 0 sparsity cost ($\lambda_i = 0$) will allow that mode to be fit as in a regular CPD model.

A. MDTD optimization algorithm and complexity

We present the overall optimization algorithm in the case of tensors with missing values in Alg. 1. Detailed derivations are available in [18]. We first initialize all variables (Step 1) and pre-compute eigenvalue decompositions of $\Phi^T\Phi$ for non-orthogonal dictionaries (Steps 2-6). In the main loop of the algorithm (Steps 7-28) we iteratively update each mode's factors (Steps 8-24) and update the missing value imputation matrix (Step 25) until convergence. In Steps 9-11 we compute the factors for modes that are not currently being updated

Algorithm 1 MDTD (with missing values)

```
Input: Input $\mathcal{X}$, mask $\Omega$, dictionaries $\Phi_i$, $k$, $\lambda_i$, $\rho_i$
 1: Initialize $Y_i = Z_i$ uniformly random, and $\Gamma_i = 0$ for all modes, set $\mathcal{D} = \mathcal{X}$
 2: for  i= 1 to #modes do
 3:     if $\Phi_i^T \Phi_i \neq I$ then
 4:         $E_{d,i} \Lambda_{d,i} E_{d,i}^T = \Phi_i^T \Phi_i$
 5:     end if
 6: end for
 7: while  not converged  do
 8:     for  i= 1 to #modes do
 9:         set $j \neq l \neq i$ and $j < l$
10:         $A = \Phi_j Y_j$
11:         $B = \Phi_l Y_l$
12:         if $\Phi_i^T \Phi_i = I$ then
13:             $Y_i = (\Phi_i^T D_i^T (B \odot A) + \rho_i Z_i - \Gamma_i^\tau)(B^T B \boxdot A^T A + \rho_i I)^{-1}$
14:         else
15:             $E_v p_v E_v^T = B^T B \boxdot A^T A$
16:             $C = \Phi_i^T D_i^T (B \odot A) + \rho_i Z_i - \Gamma_i^\tau$
17:             $Y_i = E_{d,i} [(E_{d,i}^T C E_v) \oslash (\mathbf{p}_{d,i} * \mathbf{p}_v^T + \rho_i)] E_v^T$
18:         end if
19:         $S_f = max(Y_{i,f})$, for $f$ from 1 to $k$
20:         $Y_i = Y_i \oslash S$
21:         $H^{(i)} = Y_i - \frac{\Gamma_i^\tau}{\rho_i}$.
22:         $Z_{i,jl} = sign\left(H_{jl}^{(i)}\right) \times max\left(\left|H_{jl}^{(i)}\right| - \frac{\lambda_i}{\rho_i}, 0\right)$
23:         $\Gamma_i^{\tau+1} = \Gamma_i^\tau + \rho_i (Z_i - Y_i)$
24:     end for
25:     $\mathcal{D} = ([[S \boxdot \Phi_1 Y_1, \Phi_2 Y_2, \Phi_3 Y_3]] + \lambda_d \Omega \odot \mathcal{X}) \oslash (\mathcal{I} + \lambda_d \Omega)$
26:     $\tau \leftarrow \tau + 1$
27:     Convergence condition: $\left|f^{t+1} - f^t\right| \leq \varepsilon$, where $f^{t+1}$ and $f^t$ are the objective values of Eq. 2 at iterations $t+1$ and $t$.
28: end while
```

through their respected dictionaries Φ and coding matrices Y. The updates for the factor of a given Y_i depend on whether the corresponding dictionary Φ_i is orthonormal. If Φ_i is orthonormal, we have a direct update (Step 13). The update for non-orthonormal dictionaries Φ_i employ the pre-computed eigendecompositions of their dictionaries and require three steps (15-17) based on our derivations in [18].

We normalize learned factors in Steps $19-20$ by dividing each factor by its maximum value. Similar normalization is commonly used in CPD algorithms to ensure that the scale of each factor is bounded [14]. Finally, we update proxy variables and Lagrangian coefficients following the ADMM updates in Steps (19-23). When the input tensor does not have missing values, or their imputation is not necessary (i.e., we simply need a decomposition), we omit step 25 and simply replace all unfoldings D_i with the unfolding of the input tensor X_i elsewhere in the algorithm. The three steps of Alg. 1 which dominate the computational complexity are (i) the matrix inversion in step 13, which runs in $O(k^3)$ (ii) the tensor reconstruction in step 25 $[[S \boxdot \Phi_1 Y_1, \Phi_2 Y_2, \Phi_3 Y_3]]$ involving the Khatri–Rao product of three matrices of sizes $I \times k$,$J \times k$ and $J \times k$ with complexity $O(IJTk)$ and (iii) the product $\Phi_i^T D_i^T (B \odot A)$ in steps 13 and 16. Let Φ_i^T be $p_i \times m_i$, D_i^T be of size $m_i \times m_j m_l$ and $(B \odot A)$ be $m_j m_l \times k$, then the complexity of the latter step is $O(p_i m_i m_j m_l + p_i m_j m_l k)$ if one performs $\Phi_i^T D_i^T$ first or $O(p_i m_i k + m_i m_j m_l k)$ if $D_i^T (B \odot A)$ is performed first. The model rank k and the number of dictionary atoms p_i are the two hyperparameters that directly affect the overall complexity. The typical motivation behind tensor decomposition is that real-world tensors are often of low rank, i.e., ($k < p_i$). Assuming also that the number of atoms is of the same order as the size of the associate tensor mode ($p_i = O(m_i)$) leads to an asymptotic running time similar to dictionary-free updates such as ALS-based CPD. Reconstructing the full tensor $\mathcal{D}$ with missing values in Step 25, requires materializing a potentially dense large tensor even if the input and the number of missing values are relatively sparse. We discuss an alternative scalable solution for this step for the case of large sparse tensors in [18].

IV. Experimental Evaluation

We compare MDTD to baselines on (i) model quality, (ii) size, and (iii) missing value imputation. We preform and task and data specific grid search for all methods when appropriate. To facilitate reproducibility we include a document detailing the parameters selected, how they were set for each method, and additional material such as dictionary construction formulas with our code at http://cs.albany.edu/~petko/lab/code.html. For MDTD and TGSD We utilize commonly adopted dictionaries for graph (GFT [25]) and temporal (Ramanujan [28] and Spline [10]) modes in our experimental evaluation. A concise summary of these bases can be found in [19]. We add the time cost of dictionary creation to the total running time of MDTD in all tables. Different variations of our method are denoted by MDTD followed by the dictionary abbreviations. For example, MDTD with a Spline dictionary on the first mode, GFT on the second, and no dictionary for the third would be denoted as MDTD SG. We denote variations of the matrix dictionary decomposition baselines TGSD [19] similarly. In [18], we also evaluate the ability of MDTD to perform rank estimation.

A. Experimental Setup

Datasets. We employ synthetic data and three real world datasets for evaluation, including a spatial dataset (*Crime*), social interactions from *Reality Mining (RM)*, and data from content exchange (*Twitch*). We provide their statistics in Tbl. I and describe each dataset in what follows.

- *Synthetic Data.* We generate 3-way synthetic datasets according to 2 distinct GFT dictionaries generated from two stochastic block model (SBM) graphs and a Ramanujan periodic dictionary (max period 10 and 400 time steps). Communities in both SBM graphs contain half of all possible internal edges and an equal number of external edges. The first (smallest eigenvalue) 50 and 30 Laplacian eigenvectors respectively are used as dictionaries. We generate 10 sparsely encoded factors for each mode with 75% nonzero atom loadings set to uniformly random values in $[0, 1]$. We form a tensor product of dictionary-encoded factors and add Gaussian noise at SNR=20 to the tensor. Synthetic samples and code to generate them can be found within our implementation available at https://www.cs.albany.edu/~petko/lab/code.html
- *Twitch* [23] consists of followers viewing the content of streamers. An entry represents a follower watching a stream during a given hour. We select the top 5000, 8000, and 8000

Dataset statistics						MDTD			TGSD			CPD			Tucker			TT		
Dataset	m_1	m_2	Prior	m_3	Prior	SSE	NNZ	time	SSE	NNZ	time	SSE	NNZ	time	SSE	NNZ	time	SSE	NNZ	time
Syn	200	300	Graph	400	Period	**500**	**1045**	2	522	508K	35	605	8829	2.4	681	8100	**.5**	607	80K	1.5
RM	94	94	Graph	719	Hours	**7M**	**44K**	.38	8M	267K	12	**7M**	64K	.49	**7M**	100K	**.31**	**7M**	180K	1.2
Crime	77	24	Hours	6186	Days	**2.19K**	**3794**	.23	2.32K	7k	89	2.20K	6K	**.04**	2.20K	6K	.69	3.08K	14M	3.6
Twitch-S	5000	300	Graph	200	Hours	**1.7M**	**9K**	17*	16M	8M	9K	1.8M	55K	**1***	1.8M	56K	6*	16M	225M	90
Twitch-M	8000	500	Graph	200	Hours	**5M**	**14K**	36*	**5M**	455K	76K	**5M**	87K	**3***	**5M**	88K	9*	**5M**	700K	178*
Twitch-L	8000	3000	Graph	500	Hours	**20M**	**10K**	71*	/	/	/	**20M**	115K	**8***	**20M**	116K	22*	/	/	/

TABLE I: Summary of datasets and comparison to baselines on decomposition quality, size and running time in seconds. Column m_i show the size of the i-th tensor mode, while Prior specifies the type of side information available which in turn informs the choice of dictionary for MDTD and TGSD. All datasets have a graph prior for m_1. We explicitly denote the dictionaries used each dataset in Fig.2. *Method time was recorded using sparse tensor representation.

most active users and the top 300, 500 and 3000 most active streamers from this dataset to form three versions of increasing size from this dataset: Twitch-S, Twitch-M, and Twitch-L respectively. The follower graph is based on co-viewing of the same stream with edge weights proportional to the number of hours the users co-viewed any stream. Similarly, we create a streamer graph based on shared viewership.

• *Reality Mining (RM)* [8] tracks the 94 users at MIT. Each entry represents the number of messages exchanged between a pair within a 12 hour time-span. We create a weighted graph based on the number of messages exchanged and employ its GFT as a dictionary for the first two modes.

• *Crime* [7] tracks the number of crimes that occurred in Chicago over 17 years starting in 2001. The first mode corresponds to 77 community areas of Chicago. Each entry in the tensor represents the number of crimes that took place in a particular community during a one hour period hour on a particular day (day slices are stacked to form the tensor). We utilized a map of Chicago to create an associated network by connecting neighboring communities.

Decomposition baselines. We compare MDTD to CPD [3] and Tucker decomposition [29], both implemented in Matlab's tensor toolbox [2]. We also compare to tensor train decomposition (TT) [21], utilizing the authors implementation. These approaches represent the state-of-the-art for low-rank tensor representation. We also compare to TGSD [19], a dictionary-based decomposition method for matrices by independetly applying it to graph-time or graph-graph tensor slices.

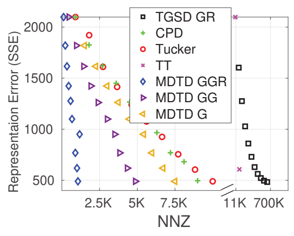
Missing value imputation baselines. We compare the quality of MDTD for missing value imputation to that of CP-WOPT [1] which employs CPD factorization by fitting only known values. We also compare to two Bayesian factorization approaches designed for imputation of missing values in road traffic datasets: BATF [4] and BCGP [5]. These methods also employ a CPD-like decomposition, but regularize the factor matrix to align to Bayesian priors. We also compare to TRLRF [30] which learns a low-rank latent space to fill in missing values; CoSTCo [17] which utilizes a convolutional neural network to learn nonlinear dependencies among factors to impute missing values; and SOFIA [15], an outlier-, seasonality-, and trend-aware tensor factorization technique for missing value imputation in temporal tensors. Finally, we also compare to TGSD [19] which can impute missing matrix values and thus apply it to one tensor slice at a time.

Metrics: We measure quality of representation as the sum of squared error (SSE) and the quality of missing value imputation in terms of mean squared error (MSE). We quantify a model's size by the number of its non-zero (NNZ) coefficients. We also measure running times for each method in seconds. COSTCO as a deep learning model was run on a Tesla V100 PCIe GPU with 16GB of RAM. All other baselines were run on a Intel(R) Xeon(R) Gold 6138 CPU @ 2.00GHz with 251G of RAM.

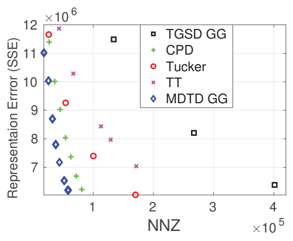
B. Succinct decomposition

We compare the accuracy of representation versus the size of the models when decomposing a tensor. We vary the decomposition rank for all methods but TT as well as the level of sparsity enforced in MDTD and TGSD (through the λ_i parameters). Since Tensor-Train (TT) is capable of tuning its optimal rank for a given error level we vary the error level to obtain decompositions of varying sparsity and quality. We report the Pareto-optimal models for all methods in terms of reconstruction error (SSE) versus model size measured as NNZ. We do not count the fixed dictionary entries towards the NNZ. These dictionaries are results of preset analytical functions and can be generated efficiently on demand as discussed in [18]. Tbl. I (right-most columns) summarizes the SSE and NNZ for one specific setting on all datasets. We select this setting by fixing a SSE level for MTDM and reporting the closest SSE regime of baselines. This allows us to compare methods in terms of model size (NNZ) for approximately similar SSE. MDTD produces the most succinct representations and its running time is comparable to the fast baselines CPD and Tucker. TT decomposes a tensor into a series of smaller tensors, leading to a typically large number of representation parameters. While TGSD also employs dictionaries, its model sizes and running time are both larger as it is a matrix (non-tensor) baseline and cannot take advantage of 3-way dependencies in the data. TGSD and TT were not able to scale to Twitch-L due to time (running time exceed 24 hrs) and memory constraints (memory exceed 64 GB) respectively.

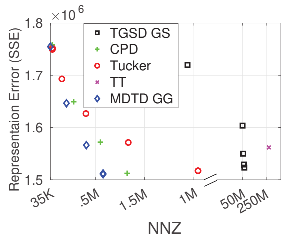
We present the full spectrum of regimes for competing techniques in Fig. 2. For our Synthetic graph-graph-time dataset we include versions of MDTD that utilize increasing set of dictionaries to serve as an ablation study evaluating the advantage of multi-dictionary decomposition 2(a). Specifically, MDTD GGR employs all three dictionaries, MDTD GG employs only the graph dictionaries, while MDTD G employs a graph dictionary only for the first mode. The joint benefit of using multiple dictionaries for encoding is evident from this comparison. The reduction in model size is super-linear with the number of dictionaries employed at the same level of SSE. For almost perfect fit (SSE ≈ 0), the single dictionary version



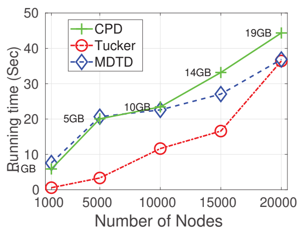
(a) Synthetic



(b) RM



(c) Twitch-S



(d) Syn run time vs #nodes

Fig. 2: Comparison of the quality and model size of MDTD and baselines CPD and Tucker on the synthetic (a), RM (b) and Twitch-S (c) datasets. In synthetic, we experiment with versions of our models with increasing number of dictionaries, while for real datasets we report the best models (MDTD GG). Different models are obtained by varying the model rank of the competing techniques and the sparsity parameters for MDTD. Only Pareto-optimal models are shown in each of these figures. (d) Scalability comparison of MDTD, CPD, and Tucker for increasing number of nodes in a synthetic dataset

	%	15	45	75	15	45	75	15	45	75
MDTD	MSE	**4.1**	**4.3**	**4.1**	.43	.43	**.43**	**.006**	**.006**	**.006**
	time	**39**	**13**	**115**	**.8**	**.7**	**.7**	**1K**	**3K**	18K
TGSD	MSE	**4.1**	**4.3**	4.3	.47	.47	.47	.007	.008	/
	time	1K	3K	2K	166	247	301	22K	40K	/
CP-WOPT	MSE	17K	20K	47K	.42	.49	211	3	40	40
	time	1K	1K	1K	1K	251	284	7K	15K	**14K**
BGCP	MSE	8.7	9.2	10.4	.51	.55	.64	.009	.008	.012
	time	42K	55K	58K	7K	5K	6K	27K	26K	26K
BATF	MSE	250	50.9	1K	**.41**	**.42**	**.43**	/	/	/
	time	3K	4K	6K	775	622	668	/	/	/
TRFL	MSE	5.4	7.0	7.3	.83	.86	1.18	/	/	/
	time	1K	1K	1K	1K	1K	1K	/	/	/
CoSTCo	MSE	5.4	5.1	5.0	.43	.43	**.43**	/	/	/
	time	6K	5K	2K	55K	33K	15K	/	/	/
SOFIA	MSE	4.7	5.2	5.2	.50	.45	.45	**.006**	**.006**	**.006**
	time	1K	1K	1K	36	39	33	20K	16K	17K

TABLE II: Comparison of the quality (MSE) and running time (seconds) for dense missing value imputation between MDTD and baselines on the real-world datasets. MDTD utilizes GGS for RM and Twitch and GSS for Crime. For TGSD we report results employing the best performing GFT+spline (GS) dictionary combination across datasets. Settings in which baselines did not complete within 24 hours are marked by the symbol "/".

MDTD G requires 75% of Tucker's (CPD's) coefficients, the two-dictionary version MDTD GG requires less than 50% of those coefficients, while the 3-dictionary version MDTD GGR requires only 10% of the coefficients. This super-linear improvement is due to the interaction of the dictionaries in the multi-way data and is also observed for sparser models of higher SSE. TGSD GR is employed on graph-time slices and is unable to utilize dependencies among all three modes leading to a huge gap in model size compared to alternatives.

We also compare the representation quality and model size of MDTD to that of CPD, Tucker, TT and TGSD on the RM and Twitch-S datasets in Figs. 2(b),2(c). In both experiments we employ GFT dictionaries for two modes for MDTD. Adding a temporal dictionary for the third mode in these datasets did not enable improvements on this task, indicating that the temporal behavior does not allow a significantly sparser encoding via the (Spline and Ramanujan) dictionaries we considered. It is important to note, however, that for the application of missing value imputation (Sec. IV-C), the Twitch dataset benefits from a spline dictionary. MDTD dominates all baselines at all levels of SSE and enables up to 5-fold reduction of the model size compared to TGSD on the RM and Twitch datasets.

C. Missing values imputation

We next evaluate the utility of MDTD for predicting missing values and compare it against baselines specifically designed for this task. We remove a set percentage (from 15% to 75%) of values at random from a given tensor and then compare the accuracy of competing imputations on these held-out values measured in terms of mean squared error (MSE) and running time measured in seconds. To tune all methods we perform a grid search over their hyper-parameters and select the configurations which produced the smallest MSE on a validation set for all datasets with the exception of Twitch-S. We found that this dataset is too large for some competing methods to grid search their hyper-parameters extensively. To ensure a fair comparison despite this, we set the rank of all models to 50, set MDTD's and TGSD's $\lambda_i = .0001$ for all i, and use the default parameters for other competitors.

All results from this experiment are presented in Tbl.II. Our method is consistently the best or close to the best method in terms of MSE and almost always much faster than alternatives. In RM MDTD is tied for the overall best performance with TGSD in terms of MSE, however, it is an order of magnitude faster. On the largest dataset in this experiment Twitch-S, MDTD's performance is the best in terms of both MSE and running time with the exception of when 75% of the values are missing. In that regime CP-WOPT is 22% faster, however, its MSE is more than 3 orders of magnitude worse. In Crime, MDTD is a very close second to BATF in terms MSE, but has up to 10 orders of magnitude speed-up against the latter. Notably, for this task of missing value imputation (unlike the decomposition task) dictionary encoding for all modes resulted in optimal MDTD models. In particular, we employed a spline dictionary for the temporal mode in, RM, Twitch-S, and the Crime datasets, effectively enforcing smoothness in time to help impute missing values in addition to smoothness on the respective graphs associated with non-temporal modes.

Real world tensors are often sparse. Thus, in the extended version [18] we introduce and experiment with a sparse imputation scheme for MDTD which allows us to impute missing values in larger sparse tensors (up to 12 billion entries). We demonstrate that MDTD scales to such large inputs, performs faster imputation and achieves lower MSE than all competitors.

D. Scalability

We also compare the scalability of MDTD to that of CPD and Tucker models on Synthetic data. We exclude TGSD and TT from this comparison since they are both significantly slower as demonstrated in all experiments above (See Tbl. I). We record the time it takes for CPD, Tucker and MDTD algorithms converge under the same convergence criteria ($\epsilon = 10^{-4}$). In Fig. 2(d) we vary the number of nodes in the first mode while holding other modes fixed to their default sizes. We utilize MDTD GGR in all settings. We also annotate the size of the input tensor in GB to illustrate the scale of the inputs considered. While Tucker is the fastest among the three competitors for small sizes, MDTD is a close second. As the size of the tensors grows, MDTD closes the gap to Tucker. For example, at 20k nodes their running times are on par. MDTD method is highly scalable regardless of its more complex objective and relatively less-optimized implementation (note that Tucker's and CPD's implementations well optimized library). In particular MDTD is able to decompose 19 gigabyte tensors in under 1 minute, making it applicable to large real-world datasets. Additional scalability and convergence experiments demonstrating similar trends can be found in [18].

V. Conclusion

In this paper we introduced a flexible and general framework for dictionary decomposition of tensors, named MDTD. Our framework produced succinct low-rank representations for both synthetic and real-world tensors by jointly employing dictionaries for multiple modes in the data. We demonstrated that our proposed ADMM optimization for MDTD converges to a high quality solution on par with CPD and Tucker in many settings. Moreover, the resulting factors were shown to be advantageous through their utility for succinct representation, their capability of estimating the ground truth rank, and their ability to accurately model the underlying patterns in the data in the presence of missing values. Our code and sample synthetic datasets are available at https://cs.albany.edu/~petko/lab/code.html.

Acknowledgement

This research is funded by the NSF SC&C grant CMMI-1831547.

References

[1] E. Acar, D. M. Dunlavy, T. G. Kolda, and M. Mørup. Scalable tensor factorizations for incomplete data. *Chemometrics and Intelligent Laboratory Systems*, 106(1):41–56, 2011.

[2] Brett, T. W. Bader, G. Kolda, et al. Tensor toolbox for matlab, version 3.2.1,, 2021. www.tensortoolbox.org.

[3] R. Bro. Parafac. tutorial and applications. *Chemometrics and Intelligent Laboratory Systems*, 38(2):149–171, 1997.

[4] X. Chen, Z. He, Y. Chen, Y. Lu, and J. Wang. Missing traffic data imputation and pattern discovery with a bayesian augmented tensor factorization model. *Transportation Research Part C: Emerging Technologies*, 104:66–77, 2019.

[5] X. Chen, Z. He, and L. Sun. A bayesian tensor decomposition approach for spatiotemporal traffic data imputation. *Transportation research part C: emerging technologies*, 98:73–84, 2019.

[6] J. E. Cohen and N. Gillis. Dictionary-based tensor canonical polyadic decomposition. *IEEE Transactions on Signal Processing*, 66(7):1876–1889, 2018.

[7] C. P. Department. Crimes - 2001 to present: City of chicago: Data portal, Feb 2022.

[8] N. Eagle and A. S. Pentland. Reality mining: sensing complex social systems. *Personal and ubiquitous computing*, 10(4):255–268, 2006.

[9] M.-J. Fadili, J.-L. Starck, and F. Murtagh. Inpainting and zooming using sparse representations. *The Computer Journal*, 52(1):64–79, 2009.

[10] V. Goepp, O. Bouaziz, and G. Nuel. Spline regression with automatic knot selection. *arXiv preprint arXiv:1808.01770*, 2018.

[11] A. Gorovits, E. Gurjal, V. Papalexakis, and P. Bogdanov. Larc: Learning activity-regularized overlapping communities across time. In *ACM SIGKDD*, 2018.

[12] A. Gorovits, L. Zhang, E. Gujral, E. Papalexakis, and P. Bogdanov. *Mining Bursty Groups from Interaction Data*, page 596–605. Association for Computing Machinery, New York, NY, USA, 2021.

[13] K. Gregor and Y. LeCun. Learning fast approximations of sparse coding. In *Proceedings of the 27th international conference on international conference on machine learning*, pages 399–406, 2010.

[14] D. Hong, T. G. Kolda, and J. A. Duersch. Generalized canonical polyadic tensor decomposition. *SIAM Review*, 62(1):133–163, 2020.

[15] D. Lee and K. Shin. Robust factorization of real-world tensor streams with patterns, missing values, and outliers. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 840–851. IEEE, 2021.

[16] A. G. Lin Zhang and P. Bogdanov. PERCeIDs: periodic community detection. In *IEEE ICDM (ICDM)*, 2019.

[17] H. Liu, Y. Li, M. Tsang, and Y. Liu. Costco: A neural tensor completion model for sparse tensors. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 324–334, 2019.

[18] M. McNeil and P. Bogdanov. Multi-dictionary tensor decomposition. In *23rd IEEE International Conference on Data Mining*, 2023.

[19] M. McNeil, L. Zhang, and P. Bogdanov. Temporal graph signal decomposition. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 1191–1201, 2021.

[20] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst. Graph signal processing: Overview, challenges, and applications. *IEEE*, 106(5):808–828, 2018.

[21] I. V. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[22] E. E. Papalexakis, C. Faloutsos, and N. D. Sidiropoulos. Parcube: Sparse parallelizable tensor decompositions. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 521–536. Springer, 2012.

[23] J. Rappaz, J. McAuley, and K. Aberer. Recommendation on live-streaming platforms: Dynamic availability and repeat consumption. In *Fifteenth ACM Conference on Recommender Systems*, pages 390–399, 2021.

[24] R. Rubinstein, A. M. Bruckstein, and M. Elad. Dictionaries for sparse representation modeling. *Proceedings of the IEEE*, 98(6):1045–1057, 2010.

[25] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst. The emerging field of signal processing on graphs. *IEEE Signal Proc. Magazine*, 2013.

[26] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos. Tensor decomposition for signal processing and machine learning. *IEEE Transactions on Signal Processing*, 65(13):3551–3582, 2017.

[27] N. D. Sidiropoulos, G. B. Giannakis, and R. Bro. Blind parafac receivers for ds-cdma systems. *IEEE Transactions on Signal Processing*, 48(3):810–823, 2000.

[28] S. V. Tenneti and P. P. Vaidyanathan. Nested periodic matrices and dictionaries: New signal representations for period estimation. *IEEE Trans. Signal Processing*, 63(14):3736–3750, 2015.

[29] L. R. Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[30] L. Yuan, C. Li, D. Mandic, J. Cao, and Q. Zhao. Tensor ring decomposition with rank minimization on latent space: An efficient approach for tensor completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 9151–9158, 2019.