Tensor-Train Decomposition in Presence of Interval-Valued Data

Tensor-Train Decomposition in Presence of Interval Data

Francesco Di Mauro, K Selçuk Candan, and Maria Luisa Sapino

Abstract—In many fields of computer science, tensor decomposition techniques are increasingly being adopted as the core of many applications that rely on multi-dimensional datasets for implementing knowledge discovery tasks. Unfortunately, a major shortcoming of state-of-the-art tensor analyses is that, despite their effectiveness when the data is certain, these operations become difficult to apply, or altogether inapplicable, in presence of uncertainty in the data, a circumstance common to many real-world scenarios. In this paper we propose a way to address this issue by extending the known Tensor-Train technique for tensor factorization in order to deal with uncertain data, here modeled as intervals. Working with interval-valued data, however, presents many challenges, since many algebraic operations that form the building blocks of the factorization process, as well as the properties that make these procedures useful for knowledge discovery, cannot be easily extended from their scalar counterparts, and often require some approximation (including, though it is not only the case, for keeping computational costs manageable). These challenges notwithstanding, our proposed techniques proved to be reasonably effective, and are supported by a thorough experimental validation.

Index Terms—Tensor Factorization, Tensor-Train, Uncertain Data, Interval Valued Data.

1 Introduction

Tensors, as a generalization of matrices, are commonly used for representing multidimensional data, such as user-centered document collections in the web and user interactions in social networks [1]. As for matrices, many tensor factorization operations (such as the CANDECOMP/PARAFAC [2], [3], the Tucker [4] and, more recently, the Tensor-Train [5] decompositions) have been introduced to implement various data analysis tasks, from clustering, trend and anomaly detection [1], to correlation analysis [6] and pattern discovery [7]. Initially originated in the field of psychometrics [2], tensor decomposition has been used in a large number of domains of computer science, including signal processing, computer vision, and data mining.

In simple terms, the main idea behind the matrix factorization process is the interpretation of a matrix as the *mapping* of a set of objects (the rows of the matrix) in the space identified by a set of features (the columns), such that each element of the matrix (i.e, the value that that object assumes for that particular feature) represents the projection of that object on the dimension identified by that feature. The factorization process, which can be extended also to tensors, helps identify a new set of features, initially hidden (and, as such, often referred to as forming a *latent space*) that provide a better representation of the objects, and helps identify the patterns among them that can facilitate their interpretation and analysis.

Unfortunately, a major shortcoming of matrix and tensor factorization based analyses is that, despite their effectiveness when the data is certain (i.e., *scalar*), these operations cannot be straightforwardly applied to those scenarios where data need to be represented as ranges (or *intervals*) of possible values.

Extending state-of-the-art factorization techniques in order to handle interval-valued data presents numerous challenges, since many of the basic algebraic operations needed to implement this processes are not as straightforward for intervals as they are for scalars. In this paper, building on our previous work on matrices [8], we address these challenges by presenting and evaluating the effectiveness of original decomposition techniques for tensors (following the Tensor-Train representation), while also ing to avoid an excessive increase in the computational complexity of the problem.

1.1 Background and Related Works

In the context of this paper, a *tensor* is intended as a multi-dimensional array, i.e., a generalization of a matrix with *order* (its number of dimensions, or *modes*) greater than 2. In terms of notation, we follow the convention [9] of denoting tensors (order three or higher) by boldface Euler script letters, e.g., $\boldsymbol{\mathcal{X}}$, $\boldsymbol{\mathcal{Y}}$.

The *fibers* of a tensor are the higher order analogue of matrix rows and columns, namely one-dimensional sections of the tensor along each of the modes. For a generic third-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, the fibers are referred to as *columns* (mode-1), *rows* (mode-2) and *tubes* (mode-3), and can be denoted (fixing two of the three indices) as $\mathbf{x}_{[:,i_2,i_3]}$, $\mathbf{x}_{[i_1,:,i_3]}$ and $\mathbf{x}_{[i_1,i_2,:]}$ respectively. *Slices*, instead, are two-dimensional sections, in a third-order tensor referred to as *horizontal*, *lateral* and *frontal*, defined by fixing one of the three indices, and denoted as $\mathbf{X}_{[i_1,:,:]}$, $\mathbf{X}_{[:,i_2,:]}$ and $\mathbf{X}_{[:,:,i_3]}$.

In the literature, many decomposition techniques have been applied to tensors, forming the basis for many data

- *K. Selçuk Candan is with the School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ, 85281, USA.*
  E-mail: candan@asu.edu
- *Francesco Di Mauro and Maria Luisa Sapino are with the Department of Computer Science, University of Turin, Italy.*
  E-mail: {dimauro,mlsapino}@di.unito.it

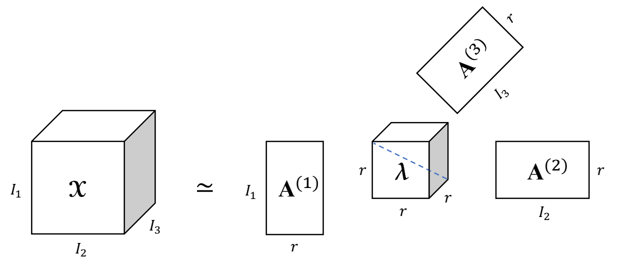


Fig. 1. Candecomp/Parafac (CP) Decomposition of a three-mode tensor $\mathcal{X}$.

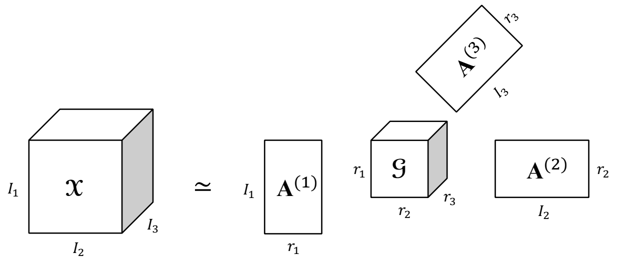


Fig. 2. Tucker Decomposition of a three-mode tensor $\mathcal{X}$.

analysis and knowledge discovery tasks. The most commonly adopted are the CANDECOMP/PARAFAC (CP) [2], [3] and the Tucker [4] factorizations. Tensor-Train [5] decomposition is recently introduced to take the complexity of the Tucker decomposition.

1.1.1 CANDECOMP/PARAFAC (CP) Decomposition

The CANDECOMP [2] (*CANonical DECOMPosition*) and PARAFAC [3] (*PARAllel FACtors*) decompositions (together known as the CP decomposition) introduced the idea of expressing a generic higher-order tensor as the sum of a finite number of rank-one tensors (*polyadic* form). More specifically, the CP Decomposition of a d-mode tensor, $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_d}$, is a weighted sum of rank-one tensors, expressed as

$$\mathcal{X} \simeq \sum_{k=1}^{r} \lambda_k \mathbf{A}^{(1)}_{[:,k]} \circ \mathbf{A}^{(2)}_{[:,k]} \circ \cdots \circ \mathbf{A}^{(d)}_{[:,k]}, \tag{1}$$

where the columns of the *factor matrices* $\mathbf{A}^{(i)} \in I_i \times r$ (with $i = 1, \cdots, d$ and $r = rank(\mathcal{X})$), combined by means of the outer product, $\circ$, form the component rank-one tensors in the summation, with $\boldsymbol{\lambda} = [\lambda_1, \lambda_2, \ldots, \lambda_r]$ absorbing the normalizing factors of each column vector.

An alternative way to view the CP decomposition is in the form of a diagonal tensor and a set of factor matrices, one for each dimension of the input tensor (see Fig. 1 for a three-mode example).

When there is an equality in (1), the equation represents an exact decomposition (also referred to as *rank decomposition*), where the rank of the tensor $\mathcal{X}$ corresponds to the smallest number of rank-one tensors that generate $\mathcal{X}$ as their sum. It is important to notice, however, that (unlike with matrices) there usually is no straightforward way to determine the rank of a given tensor (the problem is, in fact, NP-hard [10]). Thus, the most common approach to obtain a decomposition of an input tensor with good approximation, is to rely on an *Alternating Least Squares* (ALS) based method [2], [3]: the factor matrices associated to the modes of the input tensor are randomly initialized and, at each iteration, the algorithm finds a better estimation for one of the factor matrices while maintaining the others fixed; the process is then repeated for each factor matrix until a convergence condition is reached.

1.1.2 Tucker Decomposition

The Tucker decomposition can be seen as a generalization of the Singular Value Decomposition (SVD) [11] for higher-order arrays. More in detail, a d-mode tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_d}$ can be expressed, by means of the Tucker decomposition, as a core tensor multiplied by a matrix along each mode:

$$\mathcal{X} \simeq \mathcal{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \cdots \times_d \mathbf{A}^{(d)} =$$
$$= \sum_{k_1=1}^{r_1} \sum_{k_2=1}^{r_2} \cdots \sum_{k_d=1}^{r_d} \mathcal{G}_{[k_1,k_2,\ldots,k_d]} \mathbf{A}^{(1)}_{[:,k_1]} \circ \mathbf{A}^{(2)}_{[:,k_2]} \circ \cdots \circ \mathbf{A}^{(d)}_{[:,k_d]},$$

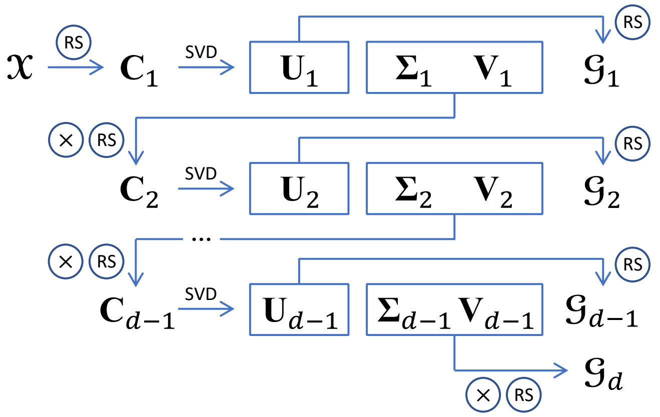
where $\mathbf{A}^{(i)} \in \mathbb{R}^{I_i \times r_i}$ are d factor matrices that can be thought of as the principal components along each mode of $\mathcal{X}$ (capturing the "group membership" relationship for the modes), while the entries of the core tensor $\mathcal{G} \in \mathbb{R}^{r_1 \times r_2 \times \cdots \times r_d}$ represent the level of interaction between the different components (capturing the relationships among the "groups"). The operator $\times_n$, referred to as the *n-mode product*, is used to multiply a tensor (specifically, its mode-n *matricization*) for a matrix. The mode-n matricization (or *unfolding*) is the operation that turns a given tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_d}$ into a matrix $\mathbf{X}^{(n)} \in \mathbb{R}^{I_n \times (I_1 \cdot \ldots \cdot I_{n-1} \cdot I_{n+1} \cdot \ldots \cdot I_d)}$, namely turning the mode-n fibers of $\mathcal{X}$ into the columns of $\mathbf{X}^{(n)}$.

Fig. 2 illustrates the Tucker factorization for a three-mode tensor. The resemblance to Fig. 1 is no coincidence: the CP decomposition, in fact, can be viewed as a special case of Tucker, where the core tensor is super-diagonal (i.e., all its elements are zero, except on the main diagonal) and $r_1 = r_2 = \cdots = r_d$.

An efficient way to compute the Tucker decomposition of a tensor, referred to as the Higher-Order Singular Value Decomposition (HOSVD for short) [12], relies on the SVD of the matricizations of the tensor along each mode to evaluate the factor matrices, which can then be used (along with the input tensor) to evaluate the core. However, while being simple, this process does not necessarily lead to an optimal decomposition, although it can be improved by using an ALS method to find better estimates of the factor matrices [13], thus increasing the overall accuracy.

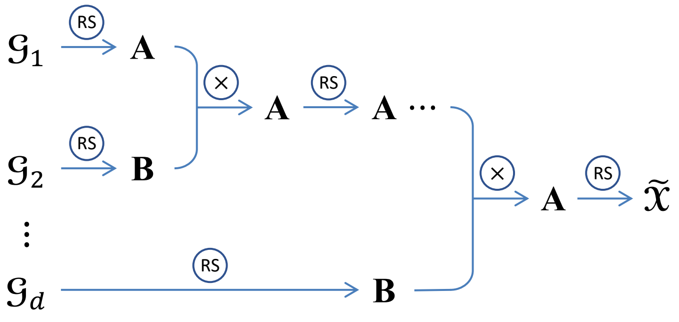
1.1.3 Tensor-Train Decomposition

The growing importance of tensor analysis in the fields of computational mathematics and computer science over the years has prompted the search for an efficient representation of a tensor by means of a small number of parameters, in order to tackle d-dimensional problems efficiently. As the number of dimensions increases, in fact, these problems cannot be handled by standard numerical methods due to the *curse of dimensionality*: everything (memory, number of operations, time of execution) grows exponentially in d.



[t].4

Fig. 3. Tensor-Train Decomposition



[t].4

Fig. 4. Tensor-Train Reconstruction

Fig. 5. The Tensor-Train factorization process.

The CP and Tucker decompositions of a tensor, illustrated in the previous sections, allow for effective representations, however, both suffer from several drawbacks.

CP guarantees a low parametric format (by means of factor matrices) but, as stated above, has the disadvantage of requiring the computation of the *tensor rank* (an NP-hard problem [10]), which not only is not guaranteed to be found, but also its numerical approximation is proven to be an ill-posed problem [14], leading to algorithms that can be stuck in local minima and might fail in providing a reliable answer even if a good approximation is known to exist.

The Tucker decomposition has the advantage of being more stable, but it suffers from requiring a number of parameters that grows exponentially with the number of dimensions of the input tensor. Thus, while proven suitable when dealing with a low number of dimensions (especially three [15]), it is rarely a good option when d gets larger. The Tensor-Train decomposition [5] tackles this issues by (a) relying entirely on the matrix Singular Value Decomposition (SVD) for its operations and (b) by representing any given d-dimensional tensor by means of d three-dimensional ones. More in detail, a d-mode tensor $\mathcal{X} \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$ is said to be in Tensor-Train form if it is expressed as d three-mode core tensors $\mathcal{G}_k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$, with $k = 1, \ldots, d$ and $r_0 = r_d = 1$. The ranks r_k, referred to as the *compression ranks*, define the accuracy of the decomposition.

Fig. 3 illustrates the process to compute the Tensor-Train form of a given tensor $\mathcal{X}$. The procedure is based on the SVD of the auxiliary matrices $\mathbf{C}_k$, obtained by *reshaping*[1] (a step notated as RS in figure) either the input tensor, to obtain $\mathbf{C}_1$, or the product of the factor matrices $\mathbf{\Sigma}_k$ and $\mathbf{V}_k^{\mathrm{T}}$ (resulting from the SVD of $\mathbf{C}_k$), to obtain the following $\mathbf{C}_{k+1}$.

The compression ranks r_k, with $k = 1, \ldots, d-1$ (as stated above, $r_0 = r_d = 1$), are defined as the *target ranks* for the SVDs of the auxiliary matrices $\mathbf{C}_k$: a full-rank decomposition at each SVD step in the process will yield an exact (and thus fully reversible) Tensor-Train decomposition of $\mathcal{X}$, while lower-rank choices will result in less accurate core tensors, with the benefit, however, of reducing their overall sizes, thus sacrificing accuracy over efficiency.

Fig. 4 illustrates the process that, taking the d cores $\mathcal{G}_1, \ldots, \mathcal{G}_d$ from the Tensor-Train decomposition, reconstructs an approximation, $\widetilde{\mathcal{X}}$, The procedure relies again on the *reshape* (either of tensors or matrices) into auxiliary matrices that are then combined by means of matrix multiplication (notated as $\otimes$ in figure).

The property of the Tensor-Train decomposition of rely-

1. Rather than a mathematical operation, the *reshape* function, common to many programming languages (MATLAB is among them), implements a way of rearranging the elements of an array (of two or more dimensions) into a new one of different size (and/or different number of dimensions), but without changing neither its number of elements nor their *linear index* (i.e., their index when treating the array as if its elements were strung out in a long column vector).

ing on simple matrix operations, such as the Singular Value Decomposition and the matrix product, proved particularly useful for the extension of the factorization analysis to interval-valued tensors, as will be described in Section 2.

Tensor train (TT) decomposition, designed to avoid the explosion of intermediary data, seeks a sequence of three-mode cores, without considering any guidelines to select the decomposition sequence, and without considering noisy data. In [?], a method for guiding the tensor train in selecting the decomposition sequence (in non noisy tensors) is introduced. The proposed GTT method leverages the data characteristics (including number of modes, length of the individual modes, density, distribution of mutual information, and distribution of entropy) as well as the target decomposition rank to pick a decomposition order that will preserve information.

1.2 Tensor Decomposition in the Presence of Uncertainty

One major problem facing the tensor decomposition approaches illustrated in the previous sections is that the factorization process can be negatively affected by noise and low quality in the data. Specifically, avoiding over-fitting to the noisy data (especially if very sparse) can be a significant challenge, particularly for large web-based user data. Recent research has shown that this issues can be addressed by modeling the decomposition as a probabilistic tensor factorization problem [16], relying on Bayesian techniques to avoid over-fitting.

This approach, however, has the significant drawbacks of assuming that all the data processed can fit in the main memory, while also ignoring the possible non-uniformities in the distribution of noise in the given input tensor. To address these issues, the authors in [17] propose a *Noise-profile adaptive Tensor Decomposition* (nTD) method which leverages a priori information about the noise in the data to partition the input tensor into multiple sub-tensors, proceeding then to decompose each of the sub-tensors by means of Bayesian probabilistic techniques, assigning the computational resources that are better suited for each subtask. In [?] authors propose a Noise-Profile Adaptive Tensor Train Decomposition method, NTTD, which leverages a model-based noise adaptive strategy. In particular, if a priori knowledge about the noise profiles of the tensor is available, it is used as the basis for a sample assignment strategy that best suits the noise distribution of the given tensor, which allows to distribute computational resources in a *noise driven* way in the decomposition process.

Other proposed methods to handle uncertainty and imprecision in the data, besides probabilistic approaches, rely on fuzzy set theory [18], where imprecise data are modeled by means of *degree of membership* to a given set (or interval) instead of crisp values. The author in [19], for example, proposes a generalization of the CP and Tucker decomposition techniques to imprecise data that are transformed into fuzzy sets by means of a *fuzzification* process.

What we propose here, instead, is a way for directly working with interval-valued high-dimensional data, a problem not yet thoroughly investigated in the literature, by relying on the findings of our work with interval-valued

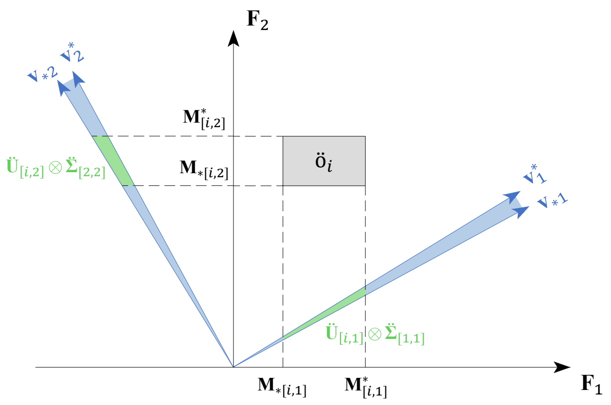


Fig. 6. Representation of a bidimensional interval latent space.

matrices [8], as the next section will explain. As we will see in the following, the proposed approach shows that in tensor train decomposition, in the presence of noisy data, also the decomposition order has an impact on the quality of the results.

2 Interval Tensor-Train Factorization

As introduced in Section 1.1.3, the Tensor-Train factorization process has the purpose of representing any given d-dimensional tensor by means of d three-dimensional ones. Likewise, for an interval-valued scenario[2], we can consider the factorization of a d-mode tensor $\ddot{\mathcal{X}} = [\mathcal{X}_*, \mathcal{X}^*]$, where $\mathcal{X}_*, \mathcal{X}^* \in \mathbb{R}^{n_1 \times n_2 \times \cdots \times n_d}$, as the set of d three-mode core tensors, such that $\ddot{\mathcal{G}}_k = [\mathcal{G}_{*k}, \mathcal{G}^*_k]$, where $\mathcal{G}_{*k}, \mathcal{G}^*_k \in \mathbb{R}^{r_{k-1} \times n_k \times r_k}$, with $k = 1, \ldots, d$ and $r_0 = r_d = 1$.

However, while the standard Tensor-Train Decomposition relies on the classical SVD algorithm (see Fig. 3), here, we first of all need to devise an extension of the Singular Value Decomposition in order to deal with the interval-valued auxiliary matrices that we encounter along the process.

2.1 Interval-Valued Matrix Factorization

In [8], we proposed a set of tools to tackle interval valued matrix factorisation by extending both eigenvector-based algorithms (such as the Singular Value Decomposition [11], or SVD, and the Non-negative Matrix Factorization [20], or NMF) as well as probabilistic state-of-the-art techniques (such as the Probabilistic Matrix Factorization [21], or PMF) to interval-valued data.

In particular, regarding the SVD, the approach follows a generalization of the factorization process to interval-valued matrices such that all the resulting factor matrices are interval-valued, i.e.:

$$\ddot{\mathbf{M}} \rightarrow \left[\ddot{\mathbf{U}}, \ddot{\mathbf{\Sigma}}, \ddot{\mathbf{V}}\right].$$

2. In terms of notation, we indicate an interval $\ddot{a}$ as a pair of scalars $\ddot{a} = [a_*, a^*]$, with $a_*, a^* \in \mathbb{R}$ and $a_* \leq a^*$, where a_* represents the minimum value and a^* the maximum value of the interval $\ddot{a}$.
Similarly, a generic array (either a matrix or a tensor), $\ddot{\mathbf{A}}$, can be defined as $\ddot{\mathbf{A}} = [\mathbf{A}_*, \mathbf{A}^*]$, with $\mathbf{A}_*, \mathbf{A}^* \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_d}$ and $\mathbf{A}_{*i_1,\ldots,i_n} \leq \mathbf{A}^*_{i_1,\ldots,i_n}$ $\forall i_1, \ldots, i_n$.

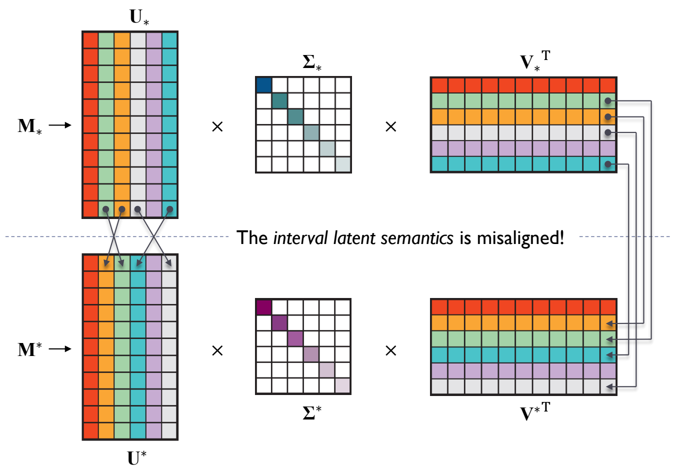

$\mathbf{U}_*$
$\mathbf{\Sigma}_*$
$\mathbf{V}_*^{\mathrm{T}}$
$\mathbf{M}_* \rightarrow$
The *interval latent semantics* is misaligned!
$\mathbf{M}^* \rightarrow$
$\mathbf{\Sigma}^*$
$\mathbf{V}^{*\mathrm{T}}$
$\mathbf{U}^*$

Fig. 7. Matching of the components of an interval latent semantic space.

A naive way to solve this task could be to separately apply the scalar SVD to the endpoint matrices, such that

$$\mathbf{M}_* = \mathbf{U}_*\mathbf{\Sigma}_*\mathbf{V}_*^{\mathrm{T}} \quad \text{and} \quad \mathbf{M}^* = \mathbf{U}^*\mathbf{\Sigma}^*\mathbf{V}^{*\mathrm{T}}.$$

However, this leads to the following issues:

1) An important property of the standard SVD is that both the left and right singular vectors (i.e., the columns of $\mathbf{U}$ and $\mathbf{V}$, respectively) are mutually orthogonal. However, after an independent decomposition of $\mathbf{M}_*$ and $\mathbf{M}^*$, each *interval latent component* (i.e., each column vector of the factor matrix $\ddot{\mathbf{V}}$) is identified by a pair of vectors, one for the minimum and one for the maximum values. Fig. 6, which represents a bidimensional interval latent space projected over the feature space which maps the objects of an interval-valued matrix $\ddot{\mathbf{M}}$, shows intuitively how the orthogonality constraint on the interval latent components cannot be respected, unless both are scalar. In the paper, we presented a formal demonstration of this issue, that leads us to conclude that the interval decomposition process cannot be exact, and some approximation is inevitable.
2) A second problem that arises from an independent decomposition is that the minimum and maximum singular vectors that express the same concepts may be *mismatched*, since their order depends on the magnitude of the relative singular values, which may be different. Fig. 6 illustrates this issue, identifying the latent components that express the same concepts with the same colors.
3) A third issue is related to the fact that what matters in defining a latent space is the direction of its basis vectors, but not their orientation. This means that two vectors that form an interval latent component could be very close in direction, expressing a similar semantics, but their direction could be opposite.
4) Lastly, the endpoints of the factor matrices resulting from the independent decomposition process may not necessarily form valid intervals, where each minimum value is actually less or equal to the maximum. For the decomposition process to be consistent, we need all the entries of the factor matrices to form valid intervals.

In order to solve these issues, the main idea behind our approach is that of guaranteeing that the latent semantic components in the minimum and maximum factor matrices resulting from the decomposition of an interval-valued matrix are always correctly matched and aligned, a goal that can be achieved by means of our proposed *Interval Latent Semantic Alignment* (ILSA, for short) procedure.

More in detail, given an interval-valued matrix $\ddot{\mathbf{M}} = [\mathbf{M}_*, \mathbf{M}^*]$, our proposed Interval-Valued Singular Value Decomposition (ISVD) approach can be briefly summarized in the following steps:

1) Independently decompose the minimum and maximum matrices, such that $\mathbf{M}_* = \mathbf{U}_*\mathbf{\Sigma}_*\mathbf{V}_*^{\mathrm{T}}$ and $\mathbf{M}^* = \mathbf{U}^*\mathbf{\Sigma}^*\mathbf{V}^{*\mathrm{T}}$.
2) Align the components expressing to the same latent semantic by means of the ILSA procedure.
3) Restore the interval-valued entries of the factor matrices such that $\mathbf{U}_{*[i,j]} \leq \mathbf{U}^*_{[i,j]}$, $\mathbf{V}_{*[i,j]} \leq \mathbf{V}^*_{[i,j]}$ $\forall i,j$ and $\sigma_{*i} \leq \sigma^*_i$ $\forall i$ (where the inequalities do not hold, we replace the wrong ordered entries with their average).

2.2 Interval Tensor-Train Decomposition

Once the factorization problem for interval-valued matrices has been tackled, we are ready to define a general strategy for computing the Tensor-Train form of a given interval-valued tensor. Fig. 8 illustrates our Interval Tensor-Train Decomposition procedure that, taking a d-dimensional interval-valued tensor $\ddot{\mathfrak{X}}$ as input, returns d interval-valued core tensors $\ddot{\mathfrak{G}}_k$, with $k = 1, \ldots, d$:

2.2.1 ReShape

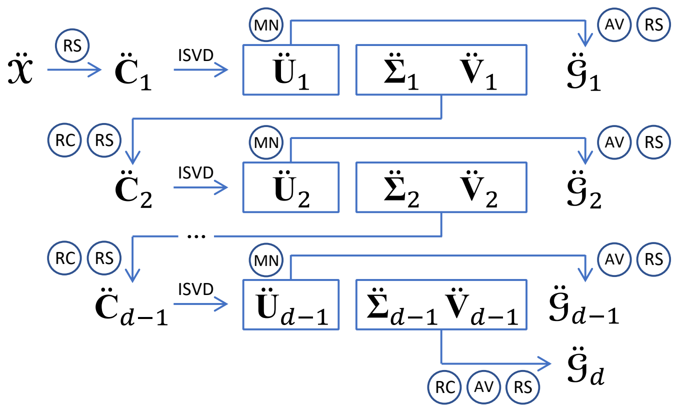
(RS) The **ReShape** function, already introduced for the scalar scenario (see Section 1.1.3), can be readily extended to the interval-valued case, where it just operates separately for the minimum and maximum endpoints of the matrix (or tensor) that needs to be reshaped (i.e., whose entries need to be rearranged to obtain an array of a different size).

2.2.2 ReCombination

(RC) The **ReCombination** function represents the extension of the standard matrix product to the interval-valued factor matrices $\ddot{\mathbf{\Sigma}}_k$ and $\ddot{\mathbf{V}}_k$ (obtained from the ISVD of $\ddot{\mathbf{C}}_k$), that need to be recombined in order to obtain the next auxiliary matrix, $\ddot{\mathbf{C}}_{k+1}$, and proceed with the next step of the decomposition procedure. More specifically, we contemplate four alternatives, according to whether we want to keep the factor matrices interval-valued or scalar (in which case the average of their endpoints is considered, by evaluating $\overline{\mathbf{V}}_k = \frac{\mathbf{V}_{k*}+\mathbf{V}_{k*}}{2}$ and $\overline{\mathbf{\Sigma}}_k = \frac{\mathbf{\Sigma}_{k*}+\mathbf{\Sigma}^*_k}{2}$):

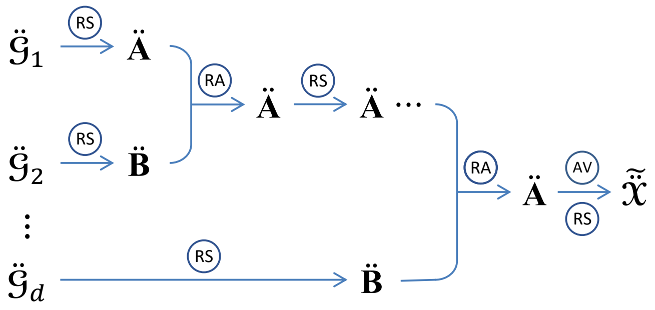
(a) $\ddot{\mathbf{C}}_{k+1} = \ddot{\mathbf{\Sigma}}_k \otimes \ddot{\mathbf{V}}_k$;

(b) $\ddot{\mathbf{C}}_{k+1} = \left[\mathbf{\Sigma}_{k*} \cdot \overline{\mathbf{V}}_k, \mathbf{\Sigma}^*_k \cdot \overline{\mathbf{V}}_k\right]$;



[t].4

Fig. 8. Interval Tensor-Train Decomposition



[t].4

Fig. 9. Interval Tensor-Train Reconstruction

Fig. 10. The Interval Tensor-Train factorization process.

(c) $\ddot{\mathbf{C}}_{k+1} = [\overline{\mathbf{\Sigma}}_k \cdot \mathbf{V}_{k*}, \overline{\mathbf{\Sigma}}_k \cdot \mathbf{V}^*_k]$;

(d) $\overline{\mathbf{C}}_{k+1} = \overline{\mathbf{\Sigma}}_k \cdot \overline{\mathbf{V}}_k$.

2.2.3 Validation

AV The **AVerage** function has the purpose of validating the interval-valued entries of the output core tensors (i.e., guaranteeing that $\mathcal{G}_{k*[l,m,n]} \leq \mathcal{G}^*_{k[l,m,n]} \forall k, l, m, n$) and operates by replacing the wrongly ordered entries in each core tensor $\ddot{\mathcal{G}}_k$ with their average:

$$\mathcal{G}_{k*[l,m,n]} := \mathcal{G}^*_{k[l,m,n]} := \frac{\mathcal{G}_{k*[l,m,n]} + \mathcal{G}^*_{k[l,m,n]}}{2},$$

for each entries such that $\mathcal{G}_{k*[l,m,n]} > \mathcal{G}^*_{k[l,m,n]}$.

MN The **MiNimization** function has been devised as an optimization of the decomposition process, and has the purpose of reducing the number of entries of a factor matrix that need to be forced as scalar during the validation process. Let us consider a pair of interval-valued factor matrices $\ddot{\mathbf{U}} = [\mathbf{U}_*, \mathbf{U}^*]$, with $\mathbf{U}_*, \mathbf{U}^* \in \mathbb{R}^{n\times r}$, and $\ddot{\mathbf{V}} = [\mathbf{V}_*, \mathbf{V}^*]$, with $\mathbf{V}_*, \mathbf{V}^* \in \mathbb{R}^{m\times r}$, and let us suppose that some of their entries do not represent valid intervals (i.e., the endpoints of these entries are misordered and would need to be replaced with their average). We can then define the following two quantities, one for each column vector k of $\ddot{\mathbf{U}}$ and $\ddot{\mathbf{V}}$, for $k = 1, \ldots, r$:

$$\Delta^U_k = \sum_{i=1}^{n} \mathbf{U}^*_{[i,k]} - \mathbf{U}_{*[i,k]},$$

$$\Delta^V_k = \sum_{i=1}^{n} \mathbf{V}^*_{[i,k]} - \mathbf{V}_{*[i,k]}.$$

The signs (either positive or negative) of Δ^U_k and Δ^V_k can give us an idea of whether, in the validation process, we would preserve more of the interval information by leaving the k column vectors of $\ddot{\mathbf{U}}$ and $\ddot{\mathbf{V}}$ as they are, or if it would be better to swap their signs (a legitimate operation, since it is a property of the singular value decomposition that the directions of a corresponding pair of left and right singular vectors can be swapped without compromising the overall decomposition process[3]).

However, precisely because each adjustment on $\ddot{\mathbf{U}}$ would necessarily affect also $\ddot{\mathbf{V}}$, we have to either prioritize one of the two (sacrificing the interval information in the other), or try to reach a compromise, loosing as least interval information as possible from both by looking locally at each pair of singular vectors, $\ddot{\mathbf{U}}_{[:,k]}$ and $\ddot{\mathbf{V}}_{[:,k]}$, prioritizing either the left or the right one according to how much interval

3. Note that, for every $i = 1, \ldots, k$,

$$\sigma_i \cdot \mathbf{u}_i \cdot \mathbf{v}_i^{\mathrm{T}} \equiv \sigma_i \cdot (-\mathbf{u}_i) \cdot (-\mathbf{v}_i^{\mathrm{T}}).$$

information each one carries (namely, not just looking at the signs of Δ_k^U and Δ_k^V, but also at their magnitude).

More specifically, for every $k = 1, \dots, r$, we have the following three options for deciding when it is best to swap the direction of a pair of column vectors $\ddot{\mathbf{U}}_{[:,k]}$ and $\ddot{\mathbf{V}}_{[:,k]}$:

1) **Prioritize $\ddot{\mathbf{U}}$:**

$$\textbf{if } \Delta_k^U < 0 \textbf{ then } \ddot{\mathbf{U}}_{[:,k]} := -1 \cdot \ddot{\mathbf{U}}_{[:,k]} \wedge \ddot{\mathbf{V}}_{[:,k]} := -1 \cdot \ddot{\mathbf{V}}_{[:,k]}$$

2) **Prioritize $\ddot{\mathbf{V}}$:**

$$\textbf{if } \Delta_k^V < 0 \textbf{ then } \ddot{\mathbf{U}}_{[:,k]} := -1 \cdot \ddot{\mathbf{U}}_{[:,k]} \wedge \ddot{\mathbf{V}}_{[:,k]} := -1 \cdot \ddot{\mathbf{V}}_{[:,k]}$$

3) **Compromise between $\ddot{\mathbf{U}}$ and $\ddot{\mathbf{V}}$:**

$$\begin{array}{ll} \textbf{if} & \left(|\Delta_k^U| > |\Delta_k^V| \wedge \Delta_k^U < 0\right) \vee \left(|\Delta_k^V| > |\Delta_k^U| \wedge \Delta_k^V < 0\right) \\ \textbf{then} & \ddot{\mathbf{U}}_{[:,k]} := -1 \cdot \ddot{\mathbf{U}}_{[:,k]} \wedge \ddot{\mathbf{V}}_{[:,k]} := -1 \cdot \ddot{\mathbf{V}}_{[:,k]} \end{array}$$

The decision on which option to choose at each **MiNimization** step of the Interval Tensor-Train Decomposition (see Fig. 3) is application-specific and is related to whether we expect the bulk of the interval information to be collected in one of the resulting core tensors over the others. In this case, supposing $\ddot{\mathcal{G}}_k$ being the core of interest, the best thing to do is to prioritize for $\ddot{\mathbf{V}}$ (option 2) along each step of the decomposition process that leads to its extraction, in this way helping the interval information trickle down each step of the procedure, and lastly prioritize for the factor matrix $\ddot{\mathbf{U}}_k$ (option 1) that will be reshaped as the core tensor $\ddot{\mathcal{G}}_k$ that we are actually interested in.

If, instead, we presume that all the interval information would be equally spread over all the cores (or, more generally, no one in particular), the best choice is to try to lower the overall number of entries that need to be averaged between both $\ddot{\mathbf{U}}$ and $\ddot{\mathbf{V}}$ (option 3) along each step of the decomposition procedure.

2.3 Interval Tensor-Train Reconstruction

Fig. 9 illustrates the Interval Tensor-Train reconstruction process, which, similarly to what happens for the scalar scenario (see Fig. 4), takes the d interval-valued cores $\ddot{\mathcal{G}}_1, \dots, \ddot{\mathcal{G}}_d$ obtained with the Interval Tensor-Train Decomposition procedure (Section 2) and returns either the original tensor $\ddot{\mathcal{X}}$, or an arbitrarily close approximation, $\tilde{\ddot{\mathcal{X}}}$, according to the accuracy of the initial decomposition (i.e., if the decomposition was either *full-rank* or *low-rank*):

(RS) For the **ReShape** function the same observations made for the decomposition procedure are valid (see Section 2).

(AV) The **AVerage** function, again used for validating the wrong interval-valued entries, is applied in the last step, just before returning the output tensor.

(RA) The **ReAssembling** function replaces what in the scalar scenario was represented by the standard matrix product, and is the equivalent of the **ReCombination** function described for the decomposition procedure. Here, we can again define four options for the merging of the auxiliary matrices $\ddot{\mathbf{A}}$ and $\ddot{\mathbf{B}}$, according to which one we want to keep as interval-valued or scalar:

(α) $\ddot{\mathbf{A}} := \ddot{\mathbf{A}} \otimes \ddot{\mathbf{B}}$;

(β) $\ddot{\mathbf{A}} := \left[\mathbf{A}_* \cdot \overline{\mathbf{B}}, \mathbf{A}^* \cdot \overline{\mathbf{B}}\right]$;

(γ) $\ddot{\mathbf{A}} := \left[\overline{\mathbf{A}} \cdot \mathbf{B}_*, \overline{\mathbf{A}} \cdot \mathbf{B}^*\right]$;

(δ) $\overline{\mathbf{A}} := \overline{\mathbf{A}} \cdot \overline{\mathbf{B}}$.

The decision on which option to use at each **ReAssembling** step of the reconstruction procedure is, again, application-specific and related to the disposition of the interval information in the input core tensors. The next section will help clarify this aspect, presenting a general strategy that leads to high accuracies in the reconstruction process, according to where the interval information is concentrated in the input core tensors.

3 Decomposition and Reconstruction Strategies

In this section, we define general strategies for the assignment of the **ReCombination** and **ReAssembling** procedures (presented in Sections 2.2 and 2.3 respectively) in order to verify the overall accuracy of our approach in recovering the interval information after a *full-rank* decomposition and reconstruction iteration, taking into account the inherent imprecision that emerges in the process, as described in Section 2.1.

3.1 A General Strategy for the TT Reconstruction

One of the conclusions that we draw from the experiments in our previous work [8] is that, when two interval-valued matrices need to be multiplied together, it is generally better to avoid the interval-valued matrix product and choose, if possible, a scalar product, where one of the two matrices is averaged. Since in fact, as we mentioned in Section 2.1, the interval latent components deriving from the decomposition process are inherently approximated, the interval matrix product has the effect of aggravating the inaccuracy, leading to poor reconstruction results. In light of this fact, in the following experiments we will not take into account options a and α in the **ReCombination** and **ReAssembling** procedures, respectively.

This decision affects in particular the reconstruction process: if we want to avoid the interval-valued product between two auxiliary matrices $\ddot{\mathbf{A}}$ and $\ddot{\mathbf{B}}$, in fact, it is easy to see that only one core tensor can be taken as interval-valued in the process, since only one branch of the procedure represented in Fig. 9 can carry interval-valued variables at any time in order to avoid any two auxiliary matrices from being both interval-valued when they need to be combined.

From these considerations, we can define a general strategy to assign the **ReAssembling** steps of the reconstruction procedure, precisely according to which core we want to keep as interval-valued. This strategy is summarized in Table 1, which shows, for any core tensor $\ddot{\mathcal{G}}_k$ obtained from the decomposition procedure that we want to keep as interval-valued, which set of options for the **ReAssembling** steps needs to be assigned. These steps will allow the interval information, stored in the chosen core tensor, to be preserved along the branches of the reconstruction procedure and end up in the output approximation of the initial tensor.

TABLE 1
ReAssembling options according to which core tensor is selected as the interval-valued one.

Interval core	ReAssembling step options					
$\ddot{\mathfrak{G}}_1$	β	β	β	$\cdots$	β	β
$\ddot{\mathfrak{G}}_2$	γ	β	β	$\cdots$	β	β
$\ddot{\mathfrak{G}}_3$	δ	γ	β	$\cdots$	β	β
$\vdots$				$\ddots$		
$\ddot{\mathfrak{G}}_d$	δ	δ	δ	$\cdots$	δ	γ
	$\textcircled{\scriptsize RA}_1$	$\textcircled{\scriptsize RA}_2$	$\textcircled{\scriptsize RA}_3$	$\cdots$	$\textcircled{\scriptsize RA}_{d-2}$	$\textcircled{\scriptsize RA}_{d-1}$

TABLE 2
ReCombination options in the Interval Tensor-Train Decomposition procedure according to which core tensor we would want to keep as interval-valued in the reconstruction process.

Interval core	ReCombination step options					
$\ddot{\mathfrak{G}}_1$	d	d	d	$\cdots$	d	d
$\ddot{\mathfrak{G}}_2$	c	d	d	$\cdots$	d	d
$\ddot{\mathfrak{G}}_3$	c	c	d	$\cdots$	d	d
$\vdots$				$\ddots$		
$\ddot{\mathfrak{G}}_d$	c	c	c	$\cdots$	c	c
	$\textcircled{\scriptsize RC}_1$	$\textcircled{\scriptsize RC}_2$	$\textcircled{\scriptsize RC}_3$	$\cdots$	$\textcircled{\scriptsize RC}_{d-2}$	$\textcircled{\scriptsize RC}_{d-1}$

3.2 A General Strategy for the TT Decomposition

Since, following the strategy for the reconstruction procedure presented above, we assume that only one core tensor can be taken as interval-valued, we make sure that already in the decomposition process as much of the interval information from the original tensor as possible will end up in that particular core. Our approach to enforce this outcome is twofold:

1) First of all, we make sure that the interval information in the input tensor propagates through each step in the decomposition by selecting an option for the **ReCombination** step that would keep either $\ddot{\mathbf{\Sigma}}$ or $\ddot{\mathbf{V}}$ as interval (i.e., option b or c, respectively).
   In particular, we expect option c (i.e., keep $\ddot{\mathbf{V}}$ as interval-valued, while averaging the singular values as $\overline{\mathbf{\Sigma}}$) to be best suited in general, since in this way, though we loose the exact scaling provided by the singular values, we keep the more important interval-information stored in the singular vectors, which defines the interval latent space. Option b would be more suited in the very special cases in which the interval information ends up only in the singular values, while the singular vectors are the same, namely when the maximum entries in the input matrix are a multiple (for the same constant) of the minimum values.
   Once, in this way, we reach the ISVD step that returns the $\ddot{\mathbf{U}}_k$ factor matrix that would be reshaped into the desired core tensor $\ddot{\mathfrak{G}}_k$, then there is no need for the next steps to still process interval information (we could actually risk to propagate *unwanted* interval information in form of noise), so, from then on, we can keep both $\overline{\mathbf{\Sigma}}$ and $\overline{\mathbf{V}}$ as scalar (after all, the following core tensors would be averaged anyway). We can in this way further reduce the branching factor regarding the possible options for the **ReCombination** steps in the decomposition, obtaining also in this case a general assignment strategy, which is reported in Table 2.
2) We also leverage the **MiNimization** function (introduced in Section 2) to make sure that as less of the interval information as possible gets lost in the decomposition process: when we reach the step of the decomposition procedure wherein we get the factor matrix $\ddot{\mathbf{U}}$ that is the one that will end up forming the core tensor that we intend to keep as interval-valued, we *prioritize* for $\ddot{\mathbf{U}}$ (so that, as explained in Section 2, it will loose as less information as possible in the validation process), otherwise we prioritize for $\ddot{\mathbf{V}}$ (including the case in which we are interested in keeping the last core tensor as interval-valued, since it will be the outcome of the recombination of $\ddot{\mathbf{\Sigma}}_{d-1}$ and $\ddot{\mathbf{V}}_{d-1}$).

In the next section, we wil lasess the impact of the proposed general strategies for TT decomposition and reconstruction.

4 Experiments

In order to validate our approach on the factorization of interval-valued tensors, we have devised a set of experiments, with real and synthetic data sets, that evaluate the accuracy of the reconstruction (see Fig. 9) of a d-dimensional tensor $\ddot{\mathfrak{X}}$ from its core tensors $\ddot{\mathfrak{G}}_1, \ddot{\mathfrak{G}}_2, \ldots, \ddot{\mathfrak{G}}_d$ obtained from the decomposition procedure (see Fig. 8). In particular, we are interested in finding whether the **ReCombination** and **ReAssembling** strategies presented in the previous section lead to the best reconstruction accuracy, minimizing the difference between the input tensor $\ddot{\mathfrak{X}}$ and its approximation $\tilde{\ddot{\mathfrak{X}}}$.

4.1 Reconstruction Accuracy

The discrepancy between $\ddot{\mathfrak{X}}$ and $\tilde{\ddot{\mathfrak{X}}}$, can be measured by means of the normalized Frobenius norm of the difference of their endpoint tensors, namely,

$$\Delta\left(\mathfrak{X}_*, \tilde{\mathfrak{X}}_*\right) = \frac{\left\|\mathfrak{X}_* - \tilde{\mathfrak{X}}_*\right\|_F}{\|\mathfrak{X}_*\|_F},$$
$$\Delta\left(\mathfrak{X}^*, \tilde{\mathfrak{X}}^*\right) = \frac{\left\|\mathfrak{X}^* - \tilde{\mathfrak{X}}^*\right\|_F}{\|\mathfrak{X}^*\|_F}.$$

These values can be converted into accuracies, and, for ease of interpretation, clipped between 0 and 1:

$$\Theta\left(\mathfrak{X}_*, \tilde{\mathfrak{X}}_*\right) = \max\left[0, 1 - \Delta\left(\mathfrak{X}_*, \tilde{\mathfrak{X}}_*\right)\right],$$
$$\Theta\left(\mathfrak{X}^*, \tilde{\mathfrak{X}}^*\right) = \max\left[0, 1 - \Delta\left(\mathfrak{X}^*, \tilde{\mathfrak{X}}^*\right)\right].$$

Finally, these separate measures of accuracy are combined in a single one through their harmonic mean:

$$\Theta_{\mathrm{HM}}\left(\ddot{\mathfrak{X}}, \tilde{\ddot{\mathfrak{X}}}\right) = \begin{cases} \frac{2 \cdot \Theta\left(\mathfrak{X}_*, \tilde{\mathfrak{X}}_*\right) \cdot \Theta\left(\mathfrak{X}^*, \tilde{\mathfrak{X}}^*\right)}{\Theta\left(\mathfrak{X}_*, \tilde{\mathfrak{X}}_*\right) + \Theta\left(\mathfrak{X}^*, \tilde{\mathfrak{X}}^*\right)}, & \text{if } \Theta\left(\mathfrak{X}_*, \tilde{\mathfrak{X}}_*\right) + \Theta\left(\mathfrak{X}^*, \tilde{\mathfrak{X}}^*\right) \neq 0. \\ 0, & \text{otherwise.} \end{cases}$$

In the following, we will refer to this accuracy measure, Θ_{HM}, simply as *harmonic mean* (HM for short).

4.2 Experiments on Synthetic Datasets

For this first set of experiments, we relied on a dataset where the input tensors have been randomly generated, in order to provide an easy to control testing scenario were various parameters can be purposefully set, thus allowing us to evaluate the most interesting aspects of our approach. For the reported experiments, the following parameters have been taken into account:

- **Tensor size:** we considered 5-dimensional $8 \times 8 \times 8 \times 8 \times 8$ tensors (for a total of 32768 entries each). It is important to notice that, while the input tensor has the same size for each mode, the core tensors resulting from the decomposition (according to the definition presented at the beginning of Section 2) may vary substantially in size. In particular, considering this input size and a *full-rank* factorization (see point below for details), we get:
  - $\mathcal{G}_{*1}, \mathcal{G}_1^* \in \mathbb{R}^{1\times8\times8}$ (64 entries).
  - $\mathcal{G}_{*2}, \mathcal{G}_2^* \in \mathbb{R}^{8\times8\times64}$ (4096 entries).
  - $\mathcal{G}_{*3}, \mathcal{G}_3^* \in \mathbb{R}^{64\times8\times64}$ (32768 entries).
  - $\mathcal{G}_{*4}, \mathcal{G}_4^* \in \mathbb{R}^{64\times8\times8}$ (4096 entries).
  - $\mathcal{G}_{*5}, \mathcal{G}_5^* \in \mathbb{R}^{8\times8\times1}$ (64 entries).
- **Interval density:** this parameter determines the ratio between scalar and interval-valued entries of a tensor: here this parameter is set to 50% (half of the entries in each tensor is set as interval valued).
- **Interval intensity:** another aspect to take into account when generating an interval-valued tensor is how wide the range of each entry should be. Here, to get an interval-valued entry starting from a scalar one, given an intensity value X, the range of the interval is uniformly selected between 0% and X% of the original scalar entry. Also this parameter is set to 50% (the range of each interval entry is uniformly selected between 0% and 50% of the original scalar value).
- **Target rank:** this parameter is related to the ISVD decomposition process, and varies according to objective of each particular application (generally speaking, full or high rank if we are interested in the accuracy of the reconstruction, which is the case here, low if we want to extract just a few latent semantic components for other kinds of analysis). Here, at each ISVD step in the procedure, a *full-rank* decomposition is performed.
- **Interval information disposition:** one aspect that is interesting to explore regarding interval-valued tensors is how the intervals are disposed in the data. Here we consider two options, one where the interval-valued entries are *randomly placed* in the tensor, and one where the interval information is *localized* along one of the modes. Fig. 13 illustrates this concept for a 3-dimensional $16 \times 16 \times 16$ tensor (for ease of visualization), (a) with random intervals, (b) with the intervals along mode-1 (i.e., the horizontal slices of the tensor). The same idea can be extended to 5-dimensional tensors, where the interval information can be localized along any of the five modes, which we refer to as I, J, K, L and M.

	Recombination strategy	HM according to strategy	Overall best HM	Perc. Of error
$\mathcal{G}_1$	d d d d	0,9248	0,9248	0,00%
$\mathcal{G}_2$	c d d d	0,9247	0,9249	0,02%
$\mathcal{G}_3$	c c d d	**0,9380**	**0,9380**	0,00%
$\mathcal{G}_4$	c c c d	0,9201	0,9244	0,43%
$\mathcal{G}_5$	c c c c	0,9188	0,9245	0,57%

Fig. 14. Reconstruction accuracies for tensors with **random interval information** (*the greener the cell, the better the result*, best option highlighted in bold – the tables are best viewed in color).

More in detail, given a scalar tensor $\mathcal{X} \in \mathbb{R}^{I\times J\times K\times L\times M}$ (with $I = J = K = L = M = 8$), whose entries are randomly generated from a uniform distribution of real values from 1 to 100, we generate the interval-valued tensor $\ddot{\mathcal{X}}$, according to the the disposition of the interval information, as follows:

Non-Localized Interval Information

We uniformly select 50% (according to the *interval density* parameter) of the scalar entries in $\mathcal{X}$ and we replace them with an interval defined as

$$\ddot{\mathcal{X}}_{[i,j,k,l,m]} = \left[\mathcal{X}_{[i,j,k,l,m]}, \mathcal{X}_{[i,j,k,l,m]} + \mathcal{X}_{[i,j,k,l,m]} \cdot \alpha \sim \mathcal{U}\left(0, 0.5\right)\right].$$

Localized Interval Information

We uniformly select 50% (according to the *interval density* parameter) of the first mode indices, $i \in [1, \ldots, I]$, and we replace the scalar slices they identify, $\mathcal{X}_{[i,:,:,:,:]}$, with interval-valued ones

$$\ddot{\mathcal{X}}_{[i,:,:,:,:]} = \left[\mathcal{X}_{[i,:,:,:,:]}, \mathcal{X}_{[i,:,:,:,:]} \cdot \alpha \sim \mathcal{U}\left(1, 1.5\right)\right];$$

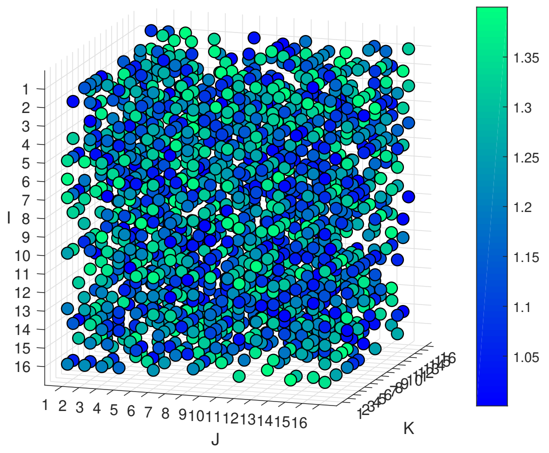
the same process is then repeated along the other modes J, K, L and M to generate the rest of the localized interval information dataset.

For each presented scenario, we created 100 random tensors and the results reported in the following are the average of each corresponding run.

4.2.1 Non-Localized Interval Information Experiments

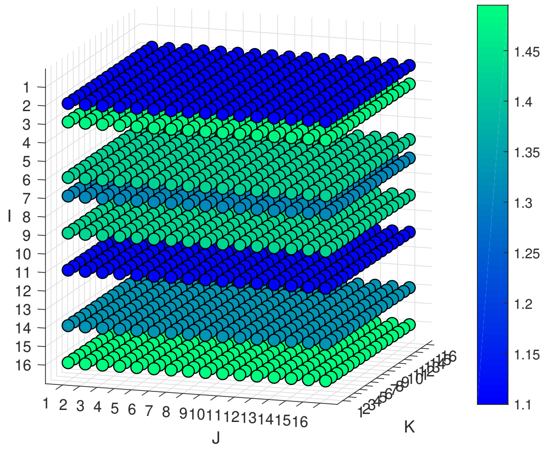
Fig. 14 illustrates the results for a run of experiments over the synthetic dataset with the interval information randomly placed in each tensor. The figure reports the Harmonic Mean (HM) results (averaged over 100 runs and color-coded from white, poor reconstruction, to green, best reconstruction) for five possible reconstruction schemes, according to which one of the five core tensors, $\ddot{\mathcal{G}}_1$ to $\ddot{\mathcal{G}}_5$, is kept as interval. We compare the reconstruction accuracy obtained following our general strategy for the assignment of the **ReCombination** steps (see Table 2) with the overall best result obtained by running all the possible combinations of options b, c and d at each of the four ISVD steps in the decomposition procedure. In the last column we then report the difference between the two results as a percentage of error (color-coded from white to red, as the error increases).

From the reported results we can conclude that:



[t].4

Fig. 11. Random placement of the interval information



[t].4

Fig. 12. Localized placement of the interval information (here along mode I)

Fig. 13. Disposition of the interval information in a 3D tensor, represented as the ratio between maximum and minimum data entries.

- when the interval information in the input tensor is randomly placed, taking any of the core tensors as interval-valued gives a reasonably good accuracy;
- however, we get a slightly better result (highlighted in light green in figure) if we take $\ddot{\mathcal{G}}_3$ as interval-valued, which seems to be related, intuitively, to the fact that it is the one with the largest size (and, consequently, the one with more entries) of all, thus being able to maintain more of the original interval information;
- our strategy, though not providing the best option for each core tensor reconstruction, is optimal in the overall best case: in order to get the best accuracy, while keeping $\ddot{\mathcal{G}}_3$ as interval, it is important to let the interval information propagate through the ISVD steps by using option c, i.e., keeping $\ddot{\mathbf{V}}_k$ as interval-valued and $\ddot{\mathbf{\Sigma}}_k$ as scalar. After the third ISVD step, from which we extract the factor matrix $\ddot{\mathbf{U}}_3$ (which after a reshape becomes $\ddot{\mathcal{G}}_3$), both $\ddot{\mathbf{\Sigma}}_k$ and $\ddot{\mathbf{V}}_k$ might as well be treated as scalar, choosing option d for the following **ReCombination** steps.

4.2.2 Localized Interval Information Experiments

For this set of experiments we considered a synthetic dataset where the interval information is localized along only one of the five modes in the input tensors at a time. Specifically, Fig. 15 reports the reconstruction accuracies (following the same format as the previous test case scenario) for each set of experiments where the interval information is localized along mode I, J, K, L and M, respectively. We again compare our strategy for the **ReCombination** options assignment at each ISVD step of the decomposition with the overall best combination of options between b, c and d. Finally, Fig. 16 summarizes these results by comparing the best option from each result set.

These results help us conclude that:

- the localization of the interval information along one of the input tensor's mode affects the choice on which one of the core tensors is best keeping as interval-valued. In particular, it is always the case that if the interval information is along mode k, then it is best to keep $\ddot{\mathcal{G}}_k$ as interval;
- our strategy for choosing the **ReCombination** options at each step of the decomposition procedure is once again confirmed to be guaranteeing the best

I	Recombination strategy	HM according to strategy	Overall best HM	Perc. Of error
$\mathcal{G}_1$	d d d d	**0,9820**	**0,9820**	0,00%
$\mathcal{G}_2$	c d d d	0,9504	0,9504	0,00%
$\mathcal{G}_3$	c c d d	0,9482	0,9482	0,00%
$\mathcal{G}_4$	c c c d	0,9185	0,9189	0,04%
$\mathcal{G}_5$	c c c c	0,9164	0,9190	0,27%

J	Recombination strategy	HM according to strategy	Overall best HM	Perc. Of error
$\mathcal{G}_1$	d d d d	0,9290	0,9290	0,00%
$\mathcal{G}_2$	c d d d	**0,9398**	**0,9398**	0,00%
$\mathcal{G}_3$	c c d d	0,9376	0,9376	0,00%
$\mathcal{G}_4$	c c c d	0,9113	0,9283	1,70%
$\mathcal{G}_5$	c c c c	0,9096	0,9286	1,90%

K	Recombination strategy	HM according to strategy	Overall best HM	Perc. Of error
$\mathcal{G}_1$	d d d d	0,9290	0,9290	0,00%
$\mathcal{G}_2$	c d d d	0,9331	0,9331	0,00%
$\mathcal{G}_3$	c c d d	**0,9690**	**0,9690**	0,00%
$\mathcal{G}_4$	c c c d	0,9326	0,9326	0,00%
$\mathcal{G}_5$	c c c c	0,9290	0,9290	0,00%

L	Recombination strategy	HM according to strategy	Overall best HM	Perc. Of error
$\mathcal{G}_1$	d d d d	0,9290	0,9290	0,00%
$\mathcal{G}_2$	c d d d	0,9331	0,9331	0,00%
$\mathcal{G}_3$	c c d d	**0,9443**	**0,9443**	0,00%
$\mathcal{G}_4$	c c c d	0,9441	0,9441	0,00%
$\mathcal{G}_5$	c c c c	0,9137	0,9286	1,50%

M	Recombination strategy	HM according to strategy	Overall best HM	Perc. Of error
$\mathcal{G}_1$	d d d d	0,9290	0,9290	0,00%
$\mathcal{G}_2$	c d d d	0,9331	0,9331	0,00%
$\mathcal{G}_3$	c c d d	0,9443	0,9443	0,00%
$\mathcal{G}_4$	c c c d	0,9449	0,9449	0,00%
$\mathcal{G}_5$	c c c c	**0,9531**	**0,9531**	0,00%

Fig. 15. Reconstruction accuracies for tensors with **interval information localized** along each mode I, J, K, L and M (*the greener the cell, the better the result*, best option highlighted in bold – the tables are best viewed in color).

accuracy in almost any scenario (except for the case when the interval information is located along mode L, where, due to the difference in size, choosing $\ddot{\mathcal{G}}_3$ as interval-valued instead of $\ddot{\mathcal{G}}_4$ gives a very slight advantage; the difference however is negligible), according to which core tensor we intend to keep as interval-valued;

- the summary results in Fig. 16 imply that, whenever possible, keeping the first core as interval-valued would be the overall optimal strategy, suggesting that it is better to *extract* the interval information *as soon as possible* in the decomposition procedure. This would imply, in general, that, given a generic tensor where the interval information is localized along one mode, it would be best to first re-orient the tensor in order to have specific mode as the first one, leading to the first core resulting from the decomposition procedure to preserve the interval information in the input data.

4.3 Experiments on a Real-World Dataset

To test our approach with real-world data, we relied on the Olivetti Research Laboratory (ORL) face dataset [22], which was used for interval valued matrix factorization evaluation in [8], [23]. The dataset is composed of a collection of images portraying the faces (in an upright position and in frontal view) of 40 different people. For each individual, a total of 10 photos have been taken, each one under slightly different conditions, either regarding the light exposure, the position of the face w.r.t. the center of the image, or the left-right rotation of the head. We thus have 400 images, each represented as a 32×32 pixels matrix, $\mathbf{M}_k \in \mathbb{N}_{+}^{32\times 32}$, with $k = 1, \ldots, 400$, where each entry $\mathbf{M}_{k[i,j]}$ represents the pixel intensity value at the specific coordinates (i, j) in the matrix (a pixel is encoded as a gray-scale level from 0 to 255).

Best HM according to interval disposition (following the recombination strategy)		
I	$\mathcal{G}_1$	**0,9820**
J	$\mathcal{G}_2$	0,9398
K	$\mathcal{G}_3$	0,9690
L	$\mathcal{G}_4$	0,9441
M	$\mathcal{G}_5$	0,9531

Fig. 16. Summary for the **localized interval information** experiments (best reconstruction accuracy for each mode I, J, K, L and M).

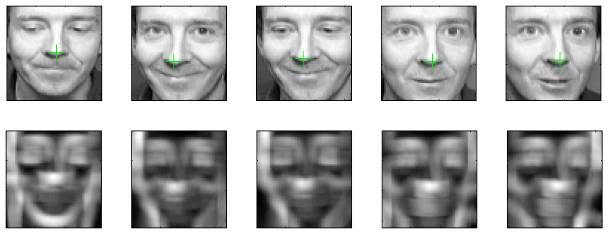


Fig. 17. An example of how the radius matrices are generated from a set of face images from the ORL dataset.

We then adopt a strategy similar to the one proposed by the authors in [23] to generate an interval-valued matrix $\ddot{\mathbf{M}}_k$ starting from $\mathbf{M}_k$, which is based on the assumption that face analysis can be seriously hindered from a lack of alignment between the facial features in different pictures of a same individual. For example, in the first row of pictures in Fig. 17, we can see how the position of the tip of the nose (marked with a cross), can be located in different coordinates in the image. Since standard decomposition strategies are particularly sensitive to this lack of alignment, authors in [23] replace the scalar brightness level of the pixels corresponding to each facial feature (i.e., around their edges) with an interval, and then applying interval-valued decomposition techniques, that may be more tolerant to such alignment errors.

More specifically, in order to generate an interval-valued entry $\ddot{\mathbf{M}}_{k[i,j]}$ starting from a scalar value $\mathbf{M}_{k[i,j]}$, a radius $\delta_{i,j}$ is evaluated, leading to a larger interval for the pixels whose surroundings are characterized by a high brightness variance (i.e., the pixel on a feature's edge or in its proximity), as follows

$$\ddot{\mathbf{M}}_{k[i,j]} = \left[\mathbf{M}_{k[i,j]} - \delta_{i,j}, \mathbf{M}_{k[i,j]} + \delta_{i,j}\right].$$

The radius $\delta_{i,j}$ for a given pixel $\mathbf{M}_{k[i,j]}$ is evaluated as the brightness variance of the pixels in the surrounding of

Frontal	Recombination strategy	HM according to strategy	Overall best HM	Perc. Of error
$\mathcal{G}_1$	d d	0,8924	0,8924	0,00%
$\mathcal{G}_2$	c d	**0,9281**	**0,9281**	0,00%
$\mathcal{G}_3$	c c	0,7882	0,7882	0,00%

Lateral	Recombination strategy	HM according to strategy	Overall best HM	Perc. Of error
$\mathcal{G}_1$	d d	0,8924	0,8924	0,00%
$\mathcal{G}_2$	c d	**0,9226**	**0,9226**	0,00%
$\mathcal{G}_3$	c c	0,8134	0,8134	0,00%

Horizontal	Recombination strategy	HM according to strategy	Overall best HM	Perc. Of error
$\mathcal{G}_1$	d d	0,7933	0,7933	0,00%
$\mathcal{G}_2$	c d	**0,9256**	**0,9256**	0,00%
$\mathcal{G}_3$	c c	0,7967	0,7967	0,00%

Fig. 22. Reconstruction accuracies for the three orientations of **ORL faces dataset** tensors (*the greener the cell, the better the result*, best option highlighted in bold – the tables are best viewed in color.

$\mathbf{M}_{k[i,j]}$:

$$\delta_{i,j} := \alpha \cdot \text{std}\left(\left\{\mathbf{M}_{k[i',j']} \,|\, \left(|i'-i| \leq r\right) \wedge \left(|j'-j| \leq r\right)\right\}\right),$$

where $\alpha \in \mathbb{R}^+$ is a multiplicative scale coefficient and r is the coordinates' radius circumscribing the surroundings of $\mathbf{M}_{k[i,j]}$ (for the experiments presented, $r = 5$ and $\alpha = 2.5$).

The second row of pictures in Fig. 17 shows the radius matrices corresponding to each face image on the first row, with lighter gray level representing a larger radius (and thus a larger interval in $\ddot{\mathbf{M}}_{k[i,j]}$).

We then arranged the 400 interval-valued matrices as 40 tensors by collecting the 10 photos characterizing each individual and ran a similar set of experiments to the one reported in Section 4.2 for synthetic data, evaluating the reconstruction accuracy of our proposed interval Tensor-Train factorization approach. As Fig. 21 shows, we chose three different ways of storing each set of images, obtaining tensors of different sizes that, in turn, end up in the following core tensors:

- **Frontal**: $\mathcal{X}_*, \mathcal{X}^* \in \mathbb{R}^{32\times32\times10}$
  - $\mathcal{G}_{*1}, \mathcal{G}_1^* \in \mathbb{R}^{1\times32\times32}$ (1024 entries).
  - $\mathcal{G}_{*2}, \mathcal{G}_2^* \in \mathbb{R}^{32\times32\times10}$ (10240 entries).
  - $\mathcal{G}_{*3}, \mathcal{G}_3^* \in \mathbb{R}^{10\times10\times1}$ (100 entries).
- **Lateral**: $\mathcal{X}_*, \mathcal{X}^* \in \mathbb{R}^{32\times10\times32}$
  - $\mathcal{G}_{*1}, \mathcal{G}_1^* \in \mathbb{R}^{1\times32\times32}$ (1024 entries).
  - $\mathcal{G}_{*2}, \mathcal{G}_2^* \in \mathbb{R}^{32\times10\times32}$ (10240 entries).
  - $\mathcal{G}_{*3}, \mathcal{G}_3^* \in \mathbb{R}^{32\times32\times1}$ (1024 entries).
- **Horizontal**: $\mathcal{X}_*, \mathcal{X}^* \in \mathbb{R}^{10\times32\times32}$
  - $\mathcal{G}_{*1}, \mathcal{G}_1^* \in \mathbb{R}^{1\times10\times10}$ (100 entries).
  - $\mathcal{G}_{*2}, \mathcal{G}_2^* \in \mathbb{R}^{10\times32\times32}$ (10240 entries).
  - $\mathcal{G}_{*3}, \mathcal{G}_3^* \in \mathbb{R}^{32\times32\times1}$ (1024 entries).

Fig. 22 reports the reconstruction accuracies (again as the Harmonic Mean, HM) for each set of experiments where the input interval-valued tensors are oriented in on of the three possible ways (frontal, lateral and horizontal), while Fig. 23 summarizes the best outcomes for each orientation.

From these results, we can conclude that:

Best HM according to tensor orientation (following the recombination strategy)		
Frontal	$\mathcal{G}_2$	**0,9281**
Lateral	$\mathcal{G}_2$	0,9226
Horizontal	$\mathcal{G}_2$	0,9256

Fig. 23. Summary for the **ORL faces dataset** tensors reconstruction accuracy (best outcome for each orientation).

- the strategies that we have devised for the assignment of the **ReCombination** and **ReAssembling** steps in the decomposition and reconstruction procedures (summarized in Tables 2 and 1, respectively) consistently lead to best results;
- on a real-world dataset, as expected, the disposition of the interval-valued information along the tensors' modes cannot be as clear-cut as the synthetic scenarios that we presented in Section 4.2, and, since the difference in size is pretty relevant, in general the best outcome is obtained by keeping the largest core tensors ($\ddot{\mathcal{G}}_2$ in all three cases) as interval-valued, similarly to the case where the interval information is randomly placed;
- the summary results in Fig. 23 also show us that, keeping the core tensors sizes equal, there is an advantage in analyzing the faces in their frontal orientation, so that $\ddot{\mathcal{G}}_2$, besides being the largest core, can keep more of the interval-information that defines the characteristic features of each individual.

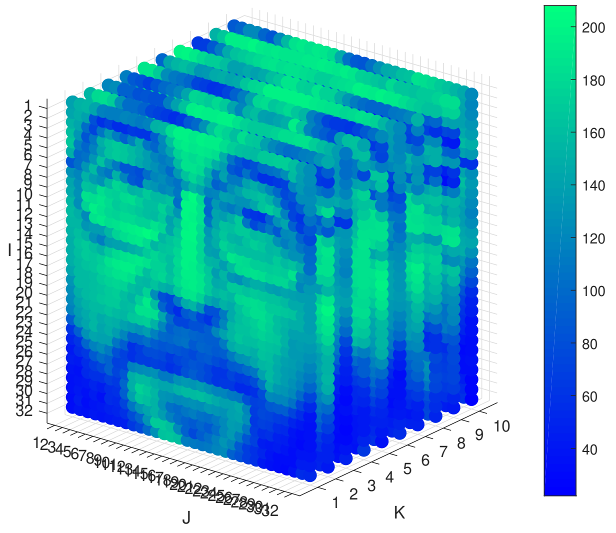
5 Conclusions

In this paper, we presented a generalization of the tensor decomposition technique in order to deal with interval-valued tensors, the motivation being that, although many applications involve this kind of data, existing analysis tools assume in general that all observations are scalar-valued.

Building on our previous work [8] for interval-valued matrices (based on the main idea of guaranteeing that the latent semantic components in the minimum and maximum factor matrices need to be always matched and aligned), we proposed an extension of the state-of-the-art Tensor-Train factorization algorithm, and we then proposed strategies for effective decompositions. These strategies have been validated through a set of experiments both on synthetic and real-world datasets. The results of our investigation show the efficacy of our proposed approach and offer a first attempt at analyzing how the disposition of the interval information along the modes of an interval-valued tensor can help the knowledge discovery tasks based on the analysis of underlying patterns in the data.

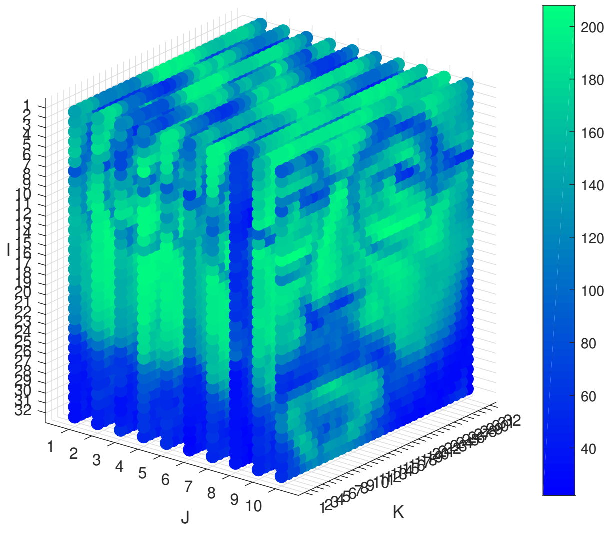
Acknowledgments

This work has been supported by NSF grants #1633381, #1909555, #1629888, #2026860, #1827757, DOD grant W81XWH-19-1-0514, and a DOE CYDRES grant. Experiments for the paper were conducted using NSF testbed: Chameleon: A Large-Scale Re-configurable Experimental Environment for Cloud Research.



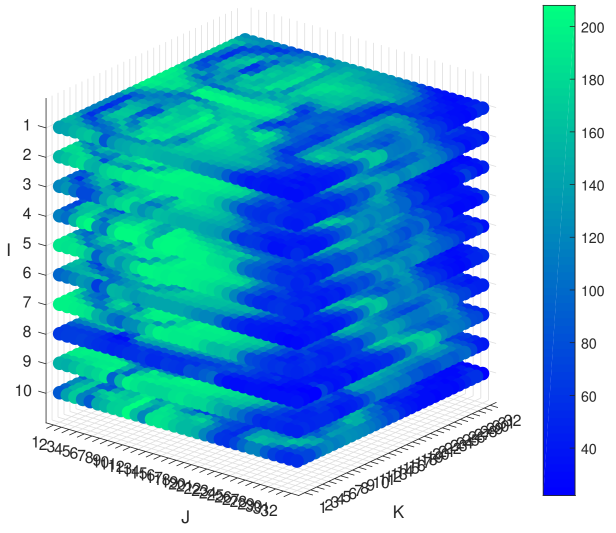
[t].3

Fig. 18. Frontal slices $(32 \times 32 \times 10)$



[t].3

Fig. 19. Lateral slices $(32 \times 10 \times 32)$



[t].3

Fig. 20. Horizontal slices $(10 \times 32 \times 32)$

Fig. 21. The ten pictures of one of the forty subjects stored as a tensor in three different orientations.

References

[1] T. G. Kolda and J. Sun, "Scalable tensor decompositions for multi-aspect data mining," in *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008)*, 2008.

[2] J. Carroll and J.-J. Chang, "Analysis of individual differences in multidimensional scaling via an n-way generalization of Eckart-Young decomposition," *Psychometrika*, vol. 35, no. 3, pp. 283–319, September 1970. [Online]. Available: https://ideas.repec.org/a/spr/psycho/v35y1970i3p283-319.html

[3] R. A. Harshman, "Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multi-modal factor analysis," *UCLA Working Papers in Phonetics*, vol. 16, pp. 1–84, 1970.

[4] L. R. Tucker, "Some mathematical notes on three-mode factor analysis," *Psychometrika*, vol. 31, pp. 279–311, 1966c.

[5] I. V. Oseledets, "Tensor-train decomposition," *SIAM J. Sci. Comput.*, vol. 33, no. 5, p. 22952317, Sep. 2011. [Online]. Available: https://doi.org/10.1137/090752286

[6] J. Sun, S. Papadimitriou, and S. Y. Philip, "Window-based tensor analysis on high-dimensional and multi-aspect streams," in *Sixth International Conference on Data Mining (ICDM'06)*. IEEE, 2006, pp. 1076–1080.

[7] B. Jeon, I. Jeon, L. Sael, and U. Kang, "Scout: Scalable coupled matrix-tensor factorization-algorithm and discoveries," in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 2016, pp. 811–822.

[8] M.-L. Li, F. Di Mauro, K. S. Candan, and M. L. Sapino, "Matrix factorization with interval-valued data," *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[9] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Rev.*, vol. 51, no. 3, p. 455500, Aug. 2009. [Online]. Available: https://doi.org/10.1137/07070111X

[10] R. A. Harshman and M. E. Lundy, "Uniqueness proof for a family of models sharing features of tucker's three-mode factor analysis and parafac/candecomp," *Psychometrika*, vol. 61, no. 1, pp. 133–154, 1996.

[11] G. W. Stewart, "On the early history of the singular value decomposition," vol. 35, no. 4, pp. 551–566, Dec. 1993.

[12] L. De Lathauwer, B. De Moor, and J. Vandewalle, "A multilinear singular value decomposition," *SIAM journal on Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1253–1278, 2000.

[13] P. M. Kroonenberg and J. De Leeuw, "Principal component analysis of three-mode data by means of alternating least squares algorithms," *Psychometrika*, vol. 45, no. 1, pp. 69–97, 1980.

[14] V. De Silva and L.-H. Lim, "Tensor rank and the ill-posedness of the best low-rank approximation problem," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 3, pp. 1084–1127, 2008.

[15] I. V. Oseledets, D. Savostianov, and E. E. Tyrtyshnikov, "Tucker dimensionality reduction of three-dimensional arrays in linear time," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 3, pp. 939–956, 2008.

[16] M. Li, K. S. Candan, and M. L. Sapino, "GTT: guiding the tensor train decomposition," in *Similarity Search and Applications - 13th International Conference, SISAP 2020, Copenhagen, Denmark, September 30 - October 2, 2020, Proceedings*, ser. Lecture Notes in Computer Science, vol. 12440. Springer, 2020, pp. 187–202.

[17] L. Xiong, X. Chen, T.-K. Huang, J. Schneider, and J. Carbonell, "Temporal collaborative filtering with bayesian probabilistic tensor factorization," 12 2010, pp. 211–222.

[18] X. Li, K. S. Candan, and M. L. Sapino, "ntd: noise-profile adaptive tensor decomposition," in *Proceedings of the 26th International Conference on World Wide Web*, 2017, pp. 243–252.

[19] ——, "Noise adaptive tensor train decomposition for low-rank embedding of noisy data," in *Similarity Search and Applications - 13th International Conference, SISAP 2020, Copenhagen, Denmark, September 30 - October 2, 2020, Proceedings*, ser. Lecture Notes in Computer Science, vol. 12440. Springer, 2020, pp. 203–217.

[20] L. A. Zadeh, "Fuzzy sets," *Information and control*, vol. 8, no. 3, pp. 338–353, 1965.

[21] P. Giordani, "Three-way analysis of imprecise data," *Journal of multivariate analysis*, vol. 101, no. 3, pp. 568–582, 2010.

[22] D. D. Lee and H. S. Seung, "Learning the parts of objects by nonnegative matrix factorization," *Nature*, vol. 401, pp. 788–791, 1999.

[23] R. Salakhutdinov and A. Mnih, "Probabilistic matrix factorization," in *NIPS*, 2007, pp. 1257–1264.

[24] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in *Proceedings of 1994 IEEE Workshop on Applications of Computer Vision*, Dec 1994, pp. 138–142.

[25] Z. Shen, L. Du, X. Shen, and Y. Shen, "Interval-valued matrix factorization with applications," in *2010 IEEE International Conference on Data Mining*, Dec 2010, pp. 1037–1042.

Francesco Di Mauro is a PhD at the Computer Science Department of the University of Torino. His research interests include decomposition of imprecise (such as interval-valued) matrices and tensors to support efficient and effective data analysis with real-world data sets.

K. Selçuk Candan is a professor of computer science and engineering at Arizona State University. His research is in the area of management and analysis of non-traditional and imprecise (such as multimedia, web, and scientific) data. He is an ACM Distinguished Scientist.

Maria Luisa Sapino is a Full Professor at the University of Torino. Her research interests are in the area of heterogeneous and multimedia data management, with emphasis on the development of efficient techniques for tensor based big data analysis and on indexing, classification, and querying of (possibly multivariate) time series.