On the Decomposition of Interval-valued Matrices and Tensors

Francesco Di Mauro

Ph.D. Thesis
Supervisor: Maria Luisa Sapino

Università degli Studi di Torino
Dipartimento di Informatica

XXXII Ciclo - Ph.D. Programme in Computer Science
a.a. 2019/2020

Contents

List of Figures

Abstract

In many fields of computer science, matrix and tensor decomposition techniques are at the bases of many applications that rely on multi-dimensional datasets for implementing knowledge discovery tasks. Unfortunately, a major shortcoming of state-of-the-art matrix and 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. Thus, this thesis proposes a way to address this issue by extending known techniques for matrix and 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.

Chapter 1

Introduction

In this era of *big data*, increasingly large amounts of information need to be handled and analyzed, presenting us with numerous challenges. In order to address these issues, and extract useful information from the rapidly growing volumes of data, knowledge discovery approaches have been highly investigated, from regression and factor analysis, to dimensionality reduction and clustering algorithms, over numerous computer science disciplines, such as natural language and image processing, data mining, and information retrieval. Matrix and tensor factorization techniques, in particular, have emerged through the years as successful tools for discovering underlying patterns in the data.

We refer to this hidden information, which cannot be directly observed but is rather implicit in the data, as *latent semantics*, a concept derived from natural language processing that refers to the high-level concepts that can be inferred (through a mathematical model, usually based on some form of factorization) from the analysis of the terms that appear in a set of documents.

The main idea behind the factorization process is the interpretation of a matrix (but the concept can also be extended to tensors) 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 helps identifying 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, which helps identifying 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. Such applications may include:

Summarized data: Matrix (or tensor) decomposition is an expensive process, therefore, analyzing reduced or summarized datasets can ideally be more efficient, especially for implementing interactive applications [6, 7], which require a quick response. In order to do so, several observations can be grouped (or collapsed) into a single interval-valued observation, representing data as value ranges. While it may sometimes be possible to associate statistical meanings to such intervals, so that probabilistic matrix factorization techniques (such as [42]) could be leveraged, this approach may be infeasible or ineffective due to the lack of appropriate statistical representations and/or computational cost.

Conflicting data: When a dataset reflects knowledge integrated from different data sources, it might not be feasible to assign a single scalar value to each observation, while a representation as an interval of possible values might be more appropriate [7]. Moreover, when analyzing such integrated data, the resulting intervals may not have a statistical interpretation, given their different origin.

Anonymized data: Various privacy-preserving data publishing algorithms, replace exact scalar values with less specific (though semantically consistent) ranges or intervals, such as those obtained by means of the *generalization* process [51]. Since the resulting intervals do not represent any specific data distribution, and intentionally so, probabilistic techniques for data analysis are not appropriate for this kind datasets.

Imprecise data: Finally, observations made in the real-world may be subject to imprecision due to environmental factors or inherent limitations in the sensory devices.

Extending state-of-the-art factorization techniques in order to handle interval-

valued matrices and tensors 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. This thesis tries to address these challenges, presenting and evaluating the effectiveness of original decomposition techniques, while also trying to avoid an excessive increase in the computational complexity of the problem.

Specifically, the present work is organized in two main parts, respectively devoted to matrices and tensors, divided as follows:

- **I Matrix Decomposition**
  - **Chapter 2** is intended to help the reader familiarizing with the notation adopted in the text, while also introducing the main algebraic concepts that will be relied on in order to handle interval-valued matrix and tensor operations.
  - **Chapter 3** presents a brief overview of the available literature and state of the art regarding matrix factorization techniques, and how the problem of dealing with interval-valued data has been tackled so far.
  - **Chapter 4** addresses the problem of extending known factorization techniques to interval-valued matrices and presents our proposed approach to solve this task.
  - **Chapter 5** provides an experimental validation, under diverse scenarios, of the factorization techniques introduced in the previous chapter.
- **II Tensor Decomposition**
  - **Chapter 6** presents a survey on the main tensor decomposition approaches investigated in the literature, and how these processes are affected by uncertainty in the data.
  - **Chapter 7** broadens the study presented for matrices in Chapter 4 to tensors, illustrating and validating our approach on the interval-valued factorization problem.
- **Chapter 8** draws the conclusions on the present work and outlines some prospective points for the future work of this research.
- Finally, **Appendix A**, at the end of the dissertation, provides the pseudocode for the main algorithms presented in the previous chapters.

Part I

Matrix Decomposition

Chapter 2

Background and notation

This chapter briefly formalizes the mathematical notation and tools that will be adopted in the reminder of the thesis.

2.1 Interval Algebra Notation

The first aspects that need to be formalized regard the notation of interval-valued data and how the algebraic operations are extended to this domain. These will form the building blocks for the more complex operations involving matrices, such as matrix multiplication.

2.1.1 Interval representation

In the course of the thesis, an interval value, represented by the notation $\ddot{a}$, is defined as a pair of scalars

$$\ddot{a} = [a_*, a^*], \text{ with } a_*, a^* \in \mathbb{R} \text{ and } a_* \leq a^*,$$

where a_* represents the minimum value and a^* the maximum value of the interval $\ddot{a}$. The values a_* and a^*, i.e., the endpoints of the interval, are considered both included in the interval, as well as all the values between them.

It is to be noticed that if $a_* = a^*$, then the interval $\ddot{a}$ is actually a scalar.

To represent interval-valued matrices, a similar notation is adopted, following

the convention of denoting matrices by boldface capital letters. More in detail, an interval-valued matrix $\ddot{\mathbf{A}}$ is defined as

$$\ddot{\mathbf{A}} = [\mathbf{A}_*, \mathbf{A}^*], \text{ with } \mathbf{A}_*, \mathbf{A}^* \in \mathbb{R}^{n \times m} \text{ and } \mathbf{A}_* \leq \mathbf{A}^*,$$

where the inequality $\mathbf{A}_* \leq \mathbf{A}^*$ applies to every pair of elements in the interval-valued matrix, namely, $\mathbf{A}_{*[i,j]} \leq \mathbf{A}^*_{[i,j]}\ \forall i, j$[1].

2.1.2 Interval range

Given an interval $\ddot{a}$, its range (or diameter) is defined as the real number obtained by the absolute difference between the interval endpoints:

$$range(\ddot{a}) = range([a_*, a^*]) = |a^* - a_*| \in \mathbb{R}$$

2.1.3 Interval algebraic operations

Interval arithmetic, or interval computation, has been studied by mathematicians as an approach to put bounds on rounding and measurement errors, thus developing numerical methods that yield as reliable results as possible [50].

The basic idea is to represent an uncertain value as a range of possibilities. More in detail, when dealing with an uncertain value x, it is possible to work with the two ends of the interval $[a_*, a^*]$, if known, that contains x. In this sense, the variable x can assume any value between a_* and a^*, or be one of them.

Moreover, a given function f, which applied to x would also give an uncertain result, in interval arithmetic produces an interval, $[b_*, b^*]$, that covers the range of all the possible values for $f(x)$, given all $x \in [a_*, a^*]$.

A generic operator $\langle \mathtt{op} \rangle$ on two intervals $[a_*, a^*]$ and $[b_*, b^*]$ is defined as

$$[a_*, a^*] \langle \mathtt{op} \rangle [b_*, b^*] = \{a \langle \mathtt{op} \rangle b | a \in [a_*, a^*] \wedge b \in [b_*, b^*]\}$$

---

[1] In the course of the thesis, the notation $\mathbf{A}_{[i,j]}$, or $a_{i,j}$, if there is no ambiguity between the indices and other subscripts, is adopted to indicate the element at i-th row and j-th column of a matrix $\mathbf{A}$. To indicate an entire row or column, the notation will be $\mathbf{A}_{[i,:]}$ and $\mathbf{A}_{[:,j]}$ respectively. In a similar fashion, the i-th element of a vector $\mathbf{x}$ is denoted as $\mathbf{x}_{[i]}$, or x_i.

provided that $a\langle \mathrm{op} \rangle b$ is well-defined for all $a \in [a_*, a^*]$ and $b \in [b_*, b^*]$.

The four basic arithmetic operations can be expressed in terms of the combinations of the endpoints of the intervals,

$$[a_*, a^*]\langle \mathrm{op} \rangle [b_*, b^*] = [\min(a_*\langle \mathrm{op} \rangle b_*, a_*\langle \mathrm{op} \rangle b^*, a^*\langle \mathrm{op} \rangle b_*, a^*\langle \mathrm{op} \rangle b^*), \\ \max(a_*\langle \mathrm{op} \rangle b_*, a_*\langle \mathrm{op} \rangle b^*, a^*\langle \mathrm{op} \rangle b_*, a^*\langle \mathrm{op} \rangle b^*)]$$

and, more in detail, they can be defined as follows:

Addition: $[a_*, a^*] + [b_*, b^*] = [a_* + b_*, a^* + b^*]$,

Subtraction: $[a_*, a^*] - [b_*, b^*] = [a_* - b^*, a^* - b_*]$,

Multiplication: $[a_*, a^*] \cdot [b_*, b^*] = [min(a_* \cdot b_*, a_* \cdot b^*, a^* \cdot b_*, a^* \cdot b^*), \\ max(a_* \cdot b_*, a_* \cdot b^*, a^* \cdot b_*, a^* \cdot b^*)]$,

Multiplication with a scalar[2]**:** $c \cdot [a_*, a^*] = [min(c \cdot a_*, c \cdot a^*), \\ max(c \cdot a_*, c \cdot a^*)]$,

Division: $\frac{[a_*, a^*]}{[b_*, b^*]} = [a_*, a^*] \cdot \frac{1}{[b_*, b^*]} = [a_*, a^*] \cdot \left[\frac{1}{b^*}, \frac{1}{b_*}\right]$, with $0 \notin [b_*, b^*]$.

Note that, given the above definition of interval algebraic operations, more complex interval-valued operations, such as interval-valued matrix algebra, can be defined by replacing scalar addition, subtraction, multiplication and division operations with their interval-valued counterparts. For clarity, in the remainder of the thesis, the interval-product will be denoted as $\otimes$.

[2] if the scalar c is positive, the operation can be further simplified as $c \cdot [a_*, a^*] = [c \cdot a_*, c \cdot a^*]$.

Chapter 3

State of the art

This chapter provides an introduction to the main topics covered in the thesis, presenting an overview of how the matrix factorization techniques have been addressed in the literature.

3.1 Matrix Factorization

In the field of linear algebra, many matrix decomposition techniques have been devised, each finding use among a particular class of problems. In computer science, in particular, these techniques have been adopted in solving problems like *feature selection* and *dimensionality reduction* [59], which usually involve some (often linear) transformation of the vector space containing the data to help focus on a few features (or combinations of features) that best discriminate the data in a given corpus, or that can provide a good approximation of the original data, or that can help removing the redundancy in the data, thus allowing a more efficient storage.

Among the most adopted of these techniques, the Karhunen-Loeve Transform, or KLT (also known as the Principal Component Analysis, or PCA [1]), and the Singular Value Decomposition, or SVD [13], rely on the eigendecomposition to operate a transformation of the data from one vector space to a different one, possibly with a fewer number of dimensions, with the resulting basis vectors referred to as the *latent variables* [48] or the *latent semantics* of the data [13].

These techniques have the key property that the vectors selected as the dimensions of the new space are *mutually orthogonal* and, hence, *linearly independent*, thus guaranteeing no redundancy among the dimensions. However, since the Karhunen-Loeve Transform and the Singular Value Decomposition may result in negative values, Non-negative Matrix Factorization techniques (NMF) [4, 33] have been thoroughly investigated in the literature, since the non-negativity constraints on the factor matrices enable probabilistic interpretation of the results and discovery of generative models. One key disadvantage of the NMF is that it may be costly to obtain, making it difficult to be applied in applications where data is dynamic. However, several approaches have been proposed to leverage the redundancies in the data (Group Incremental NMF, or GI-NMF, [9]) or to efficiently handle data evolving over time (Evolutionary NMF, or eNMF, [54]).

On a different note, in many real world applications (such as collaborative filtering for recommender systems or image analysis), data are usually characterized by a high degree of sparsity (due to missing entries) or may suffer from the presence of numerous outliers (due to noise or corruption in an image). Since these issues are not well addressed by classical PCA approaches, many new techniques have been proposed in recent years, such as Probabilistic Matrix Factorization (PMF [42]) or Robust Bilinear Factorization (RBF [45]).

Below, four common matrix factorization strategies are briefly outlined: eigendecomposition, Singular Value Decomposition (SVD [13]), Non-negative Matrix Factorization (NMF [4, 33]), and Probabilistic Matrix Factorization (PMF [42]). In the remainder of the thesis, these techniques would be relied on as the starting point for the extension of the factorization analysis to interval-valued matrices, as described in Chapter 4.

3.1.1 Eigendecomposition of a square matrix

In linear algebra, the *eigendecomposition* (sometimes called *spectral decomposition*) is the factorization of a square matrix into a canonical form, whereby the matrix is represented in terms of its *eigenvalues* and *eigenvectors*. More in detail, a non-zero vector $\mathbf{v} \in \mathbb{R}^n$ is called an eigenvector of a square matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ if it satisfies the linear equation (referred to as the *eigenvalue equation*)

$$\mathbf{A}\mathbf{v} = \lambda \mathbf{v}$$

where the scalar λ represents the eigenvalue corresponding to the eigenvector $\mathbf{v}$. As the matrix-vector product of $\mathbf{A}$ and $\mathbf{v}$ can be interpreted as a linear transformation for the vector $\mathbf{v}$, the eigenvalue equation above simply states that the eigenvectors are those vectors that the linear transformation $\mathbf{A}$ merely elongates or shrinks, being equivalent to multiplying the magnitude of such vectors by a scalar, namely the eigenvalue λ, without affecting their direction.

The eigendecomposition of a square matrix, along with the definition of eigenvalues and eigenvectors, plays a prominent role in many linear algebra applications, including being at the basis of many factorization approaches, as the following sections illustrate.

In the reminder of the thesis, we denote the eigendecomposition af a square matrix $A \in \mathbb{R}^{n \times n}$ as

$$\text{EIG}(\mathbf{A}, k) \rightarrow [\mathbf{V}, \mathbf{\Lambda}],$$

where $\mathbf{\Lambda}$ is a $k \times k$ diagonal matrix where the elements on the diagonal are the k largest (in magnitude) eigenvalues of $\mathbf{A}$ and $\mathbf{V}$ is an $n \times k$ matrix whose columns are the corresponding eigenvectors.

3.1.2 Singular Value Decomposition (SVD)

The Singular Value Decomposition [47], or SVD for short, is a factorization method for reducing a matrix to its constituent parts by means of eigenvectors and eigenvalues extraction. More in detail, given a real matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ of rank $r \leq \min(n, m)$, it can be expressed as the product of three *factor matrices*:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}$$

where $\mathbf{U} \in \mathbb{R}^{n \times r}$, $\mathbf{V} \in \mathbb{R}^{m \times r}$, $\mathbf{\Sigma} \in \mathbb{R}^{r \times r}$, and

$$\mathbf{\Sigma} = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$$

is a non-negative diagonal matrix, whose elements, also called the *singular values* of $\mathbf{A}$, are arranged in descending order of magnitude, and correspond to the roots of the eigenvalues of both $\mathbf{A}\mathbf{A}^{\mathrm{T}} \in \mathbb{R}^{n \times n}$ and $\mathbf{A}^{\mathrm{T}}\mathbf{A} \in \mathbb{R}^{m \times m}$, i.e.,

$$\sigma_i = \sqrt{\lambda_i}, \ \forall i = 1, \ldots, r.$$

The factor matrices

$$\mathbf{U} = (\mathbf{u}_1\ \mathbf{u}_2\ \cdots\ \mathbf{u}_r)\ \text{ and }\ \mathbf{V} = (\mathbf{v}_1\ \mathbf{v}_2\ \cdots\ \mathbf{v}_r)$$

are two orthonormal[1] matrices such that the columns of $\mathbf{U}$, also called the *left singular vectors* of $\mathbf{A}$, are the eigenvectors of $\mathbf{A}\mathbf{A}^{\mathrm{T}}$, while the columns of $\mathbf{V}$, also called the *right singular vectors* of $\mathbf{A}$, are the eigenvectors of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ (see Figure 3.1).

Since the columns of the factor matrices $\mathbf{U}$ and $\mathbf{V}$ are orthonormal, it holds that

$$\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}_n\ \text{ and }\ \mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{I}_m.$$

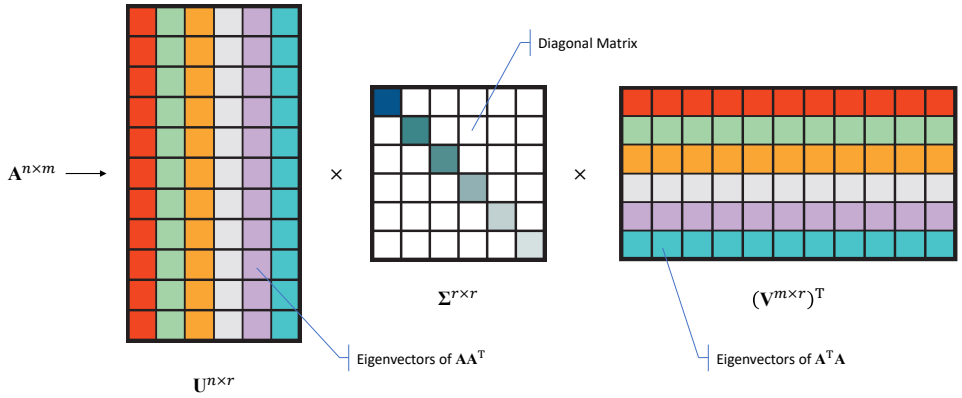


Figure 3.1: Singular Value Decomposition (SVD) of a matrix $\mathbf{A}$. The colors in $\mathbf{U}$ (columns) and $\mathbf{V}^{\mathrm{T}}$ (rows) highlight the corresponding *left* and *right singular vectors* of $\mathbf{A}$. The fading colors on the diagonal of $\mathbf{\Sigma}$ represent the decrease in magnitude of the *singular values* of $\mathbf{A}$ (non-diagonal entries are zero).

3.1.3 Non-negative Matrix Factorization (NMF)

Non-negative Matrix Factorization [4, 33], or NMF for short, identifies a group of algorithms in linear algebra where a matrix $\mathbf{A}$ that has no negative values is factorized into (usually) two factor matrices, $\mathbf{U}$ and $\mathbf{V}$, with the property that both factor matrices also have no negative elements. This non-negativity property makes the resulting matrices more suitable for numerous applications, however, since the

[1] More precisely, this is true only for a *full* SVD, with $\mathbf{U} \in \mathbb{R}^{n\times n}$ and $\mathbf{V} \in \mathbb{R}^{m\times m}$; in a *rank* SVD, $\mathbf{U}$ and $\mathbf{V}$ are only *columnwise* orthonormal, however, for the purpose of this thesis, the distinction is not crucial.

factorization problem is not exactly solvable in general, it is commonly approximated numerically.

More in detail, given a non-negative matrix $\mathbf{A} \in \mathbb{R}_+^{n \times m}$ of rank $r \leq \min(n,m)$, it is possible to find two non-negative matrices $\mathbf{U} \in \mathbb{R}_+^{n \times r}$ and $\mathbf{V} \in \mathbb{R}_+^{m \times r}$ such that the following L_2 loss function is minimized:

$$\mathcal{L}_{\text{NMF}} = \left\| \mathbf{A} - \mathbf{U}\mathbf{V}^{\text{T}} \right\|_F,$$

where $\|\cdot\|_F$ denotes the Frobenius norm (also referred to as Euclidean norm), which, for a given matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$, is defined as the square root of the sum of the squares of its elements:

$$\|\mathbf{A}\|_F = \sqrt{\sum_{i=1}^{n} \sum_{j=1}^{m} (a_{i,j})^2}$$

The approximated solutions for $\mathbf{U}$ and $\mathbf{V}$ are commonly found by means of *iterative update rules*, such as

$$\mathbf{U}_{[i,j]} \leftarrow \mathbf{U}_{[i,j]} \frac{\left(\mathbf{A}\mathbf{V}^{\text{T}}\right)_{[i,j]}}{\left(\mathbf{U}\mathbf{V}\mathbf{V}^{\text{T}}\right)_{[i,j]}},$$

$$\mathbf{V}_{[i,j]} \leftarrow \mathbf{V}_{[i,j]} \frac{\left(\mathbf{U}^{\text{T}}\mathbf{A}\right)_{[i,j]}}{\left(\mathbf{U}^{\text{T}}\mathbf{U}\mathbf{V}\right)_{[i,j]}},$$

which are proved to reach a local minimum for the loss function above.

Once a good approximation of $\mathbf{U}$ and $\mathbf{V}$ has been found, the factorization of $\mathbf{A}$ can be expressed as

$$\mathbf{A} \simeq \mathbf{U}\mathbf{V}^{\text{T}}.$$

3.1.4 Probabilistic Matrix Factorization (PMF)

Probabilistic Matrix Factorization [42], or PMF for short, is a modern factorization technique, widely adopted for collaborative filtering problems (i.e., modeling and making predictions about user preferences over a set of items), which has the advantage of scaling linearly with the number of observations and of performing well on very sparse and imbalanced datasets, where conventional approaches have particular trouble in making accurate predictions for users who have expressed very few ratings.

More in detail, let $\mathbf{R} \in \mathbb{R}^{n \times m}$ be a user-item ratings matrix of rank $r \leq \min(n,m)$, with $1 \leq i \leq n$ (number of users) and $1 \leq j \leq m$ (number of items), where $\mathbf{R}_{[i,j]}$ represents the rating that user i assigned to item j. Let then matrices $\mathbf{U} \in \mathbb{R}^{n \times r}$ and $\mathbf{V} \in \mathbb{R}^{m \times r}$ represent the user-specific and item-specific latent feature vectors respectively. Assuming that the entries in $\mathbf{R}$ are drawn from a Gaussian distribution, the conditional distribution over the observed ratings can be defined as

$$p\left(\mathbf{R}_{[i,j]} | \mathbf{U}, \mathbf{V}, \sigma^2\right) = \prod_{i=1}^{n} \prod_{j=1}^{m} \left[\mathcal{N}(\mathbf{R}_{[i,j]} | \mathbf{U}_{[i,:]} (\mathbf{V}_{[j,:]})^{\mathrm{T}}, \sigma^2)\right]^{\mathcal{I}_{ij}},$$

where

- $\mathcal{N}\left(x | \mu, \sigma^2\right)$ is the probability density function of a Gaussian distribution with mean μ and variance σ^2;
- $\mathcal{I}_{ij}$ is the indicator function that is equal to 1 if $\mathbf{R}_{[i,j]}$ is not null (i.e., if user i rated item j), and equal to 0 otherwise;
- $\mathbf{U}_{[i,:]}$ and $\mathbf{V}_{[j,:]}$ are row vectors of $\mathbf{U}$ and $\mathbf{V}$, such that $\mathbf{R}_{[i,j]} \simeq \mathbf{U}_{[i,:]} (\mathbf{V}_{[j,:]})^{\mathrm{T}}$.

Moreover, zero-mean spherical Gaussian priors are placed on $\mathbf{U}$ and $\mathbf{V}$, i.e., each latent feature vector is drawn from a multi-variate Gaussian distribution with mean 0 and precision multiple of the identity matrix $\mathbf{I}$:

$$p(\mathbf{U} | \sigma_{\mathbf{U}}^2) = \prod_{i=1}^{n} \mathcal{N}(\mathbf{U}_{[i,:]} | 0, \sigma_{\mathbf{U}}^2 \mathbf{I}),$$

$$p(\mathbf{V} | \sigma_{\mathbf{V}}^2) = \prod_{j=1}^{m} \mathcal{N}(\mathbf{V}_{[j,:]} | 0, \sigma_{\mathbf{V}}^2 \mathbf{I}).$$

The factor matrices, $\mathbf{U}$ and $\mathbf{V}$, can then be computed by minimizing the loss function

$$\mathcal{L}_{\mathrm{PMF}} = \left\|\mathbf{R} - \mathbf{U}\mathbf{V}^{\mathrm{T}}\right\|_F + \lambda_{\mathbf{U}} \|\mathbf{U}\|_F + \lambda_{\mathbf{V}} \|\mathbf{V}\|_F,$$

where $\lambda_{\mathbf{U}} = \frac{\sigma^2}{\sigma_{\mathbf{U}}^2}$, $\lambda_{\mathbf{V}} = \frac{\sigma^2}{\sigma_{\mathbf{V}}^2}$, and $\|\cdot\|_F$ denotes the Frobenius norm.

A local minimum of the loss function $\mathcal{L}_{\mathrm{PMF}}$ can be found via gradient descent on $\mathbf{U}_{[i,:]}$ and $\left(\mathbf{V}_{[j,:]}\right)^{\mathrm{T}}$:

$$\frac{\partial \mathcal{L}_{\mathrm{PMF}}}{\partial \mathbf{U}_{[i,:]}} = \sum_{j=1}^{m} \left(\mathbf{U}_{[i,:]} \left(\mathbf{V}_{[j,:]}\right)^{\mathrm{T}} - \mathbf{R}_{[i,j]}\right) \mathbf{V}_{[j,:]} + \lambda_{\mathbf{U}} \mathbf{U}_{[i,:]},$$

$$\frac{\partial \mathcal{L}_{\text{PMF}}}{\partial \left(\mathbf{V}_{[j,:]}\right)^{\mathrm{T}}} = \sum_{i=1}^{n} \left(\mathbf{U}_{[i,:]} \left(\mathbf{V}_{[j,:]}\right)^{\mathrm{T}} - \mathbf{R}_{[i,j]}\right) \left(\mathbf{U}_{[i,:]}\right)^{\mathrm{T}} + \lambda_{\mathbf{V}} \left(\mathbf{V}_{[j,:]}\right)^{\mathrm{T}},$$

and once $\mathbf{U}$ and $\mathbf{V}$ reach a good approximation, the unknown ratings in $\mathbf{R}$ can be estimated by computing

$$\widetilde{\mathbf{R}}_{[i,j]} = \mathbf{U}_{[i,:]} \left(\mathbf{V}_{[j,:]}\right)^{\mathrm{T}}.$$

3.2 Analysis of interval-valued data

In the real world, data rarely come in simple scalar form: often, the variables that need to be analyzed may take complex forms, including sets, histograms, vectors, intervals, or probability distributions [2, 37, 15, 36]. This is true, for example, when data are aggregated [16] (i.e., collected from multiple sources) or anonymized [51] (e.g., when exact information is deliberately generalized for privacy protection).

This new kind of data, referred to as *symbolic* [16], since they cannot be reduced to numbers without loosing much information, required the development of several data analysis tools, including regression [56, 20], canonical analysis [32], and multi-dimensional scaling [21]. In particular, given the popularity of latent semantic extraction techniques in data analysis, several interval-valued PCA algorithms have been proposed [3, 55, 18, 17], most of which leverage the specific statistical and geometric meanings of principal components of a system of variables.

3.2.1 The interval eigendecomposition problem

An interesting approach that we took into consideration as a benchmark in our analysis is the one proposed by the authors in [14, 44], which address the issue of extending the eigendecomposition problem to interval-valued matrices by finding interval bounds for the eigenvalues and eigenvectors of a symmetric interval-valued matrix. More specifically, given an $n \times n$ interval-valued matrix defined as

$$\ddot{\mathbf{A}} = [\mathbf{A}_C - \Delta_{\mathbf{A}}, \mathbf{A}_C + \Delta_{\mathbf{A}}],$$

with $\mathbf{A}_C$ being the center scalar matrix of $\ddot{\mathbf{A}}$ and $\Delta_{\mathbf{A}}$ its radius, such that they are both symmetrical (i.e., $\mathbf{A}_C = \mathbf{A}_C^{\mathrm{T}}$ and $\Delta_{\mathbf{A}} = \Delta_{\mathbf{A}}^{\mathrm{T}}$), the objective is to find the sets

$$\ddot{\Lambda}_i = \left[\lambda_i(\mathbf{A}) \mid \mathbf{A} \in \ddot{\mathbf{A}}\right] \text{ and } \ddot{X}_i = \left[\mathbf{x}_i(\mathbf{A}) \mid \mathbf{A} \in \ddot{\mathbf{A}}\right],\ i = 1,\ldots,n,$$

where $\lambda_i(\mathbf{A})$ and $\mathbf{x}_i(\mathbf{A})$ denote the i-th eigenvalue and, respectively, i-th eigenvector of any given matrix $\mathbf{A}$ whose elements are included in the intervals specified in $\ddot{\mathbf{A}}$, with $\mathbf{A} = \mathbf{A}_C + \delta_{\mathbf{A}}$ and $|\delta_{\mathbf{A}}| \leq \Delta_{\mathbf{A}}$. Seeking lower and upper bounds for the sets $\ddot{\Lambda}_i$ and $\ddot{X}_i$, with $i = 1,\ldots,n$, would thus provide a bounding for, respectively, the eigenvalues and eigenvectors of $\ddot{\mathbf{A}}$.

More in detail, the authors demonstrate in [14] that, if $\ddot{\mathbf{A}} = [\mathbf{A}_C - \Delta_{\mathbf{A}}, \mathbf{A}_C + \Delta_{\mathbf{A}}]$ is a real symmetric interval-valued matrix, then the eigenvalue λ_i of $\mathbf{A}$, with $\mathbf{A} \in \ddot{\mathbf{A}}$ and $i = 1,\ldots,n$, ranges over the interval

$$\ddot{\lambda}_i = [\lambda_{i*}, \lambda_i^*] = [\lambda_i(\mathbf{A}_C - \mathbf{S}_i\Delta_{\mathbf{A}}\mathbf{S}_i), \lambda_i(\mathbf{A}_C + \mathbf{S}_i\Delta_{\mathbf{A}}\mathbf{S}_i)],$$

where $\mathbf{S}_i = \mathrm{diag}\left(\mathrm{sgn}(x_{i1}),\ldots,\mathrm{sgn}(x_{in})\right)$ is a diagonal matrix whose elements in the diagonal represent the signs of the eigenvector $\mathbf{x}_i$ of $\mathbf{A}_C$.

To then find a bounding for each component of the eigenvector $\mathbf{x}_i$, the authors introduce in [44] a new numerical technique to solve the task as a linear parametric programming problem, performing a line search over a parameter $\lambda_i \in [\lambda_{i*}, \lambda_i^*]$, under the constraint

$$-\Delta_{\mathbf{A}}\,|\mathbf{x}_i| \leq (\lambda_i I - \mathbf{S}_i\mathbf{A}_C\mathbf{S}_i) \leq \Delta_{\mathbf{A}}\,|\mathbf{x}_i|,$$

where I is the $n \times n$ unit matrix and $\lambda_{i*} \leq \lambda_i \leq \lambda_i^*$.

While the approach proposed by the mentioned authors is very useful to provide bounds for the eigenvalues and eigenvectors of an interval-valued matrix, it fails to capture the inherent *latent semantics* underlaying the data, since the endpoints defining the attained bounds for the eigenvectors' components cannot be directly interpreted as the basis of a new vector space. The approach illustrated in the remainder of the thesis offers a solution to this problem by looking at a way to at least approximate a suitable basis for a vector space where to represent the latent information of interval-valued data, relying on an *interval-valued latent semantic alignment* process, which can be integrated in common matrix factorization techniques (as the ones reported in Section 3.2), allowing to extend the analysis that

rely on these facilities to non scalar data (see Chapter 4).

3.2.2 Interval-valued NMF and PMF techniques

The authors in [46] have proposed a way for extending the NMF and PMF techniques to interval-valued data, in order to resolve alignment approximation problems in face analysis and to improve rating prediction accuracy in collaborative filtering.

Interval-valued NMF. The approach taken by the authors to extended the NMF technique to interval-valued data (they refer to their approach as I-NMF) is done by extending the loss function and the iterative update rules illustrated in Section Section 3.1.3 in order to deal with the interval-valued matrices $\ddot{\mathbf{M}} = [\mathbf{M}_*, \mathbf{M}^*]$ and $\ddot{\mathbf{V}} = [\mathbf{V}_*, \mathbf{V}^*]$ (the other factor matrix, $\mathbf{U}$, is treated as scalar):

$$\mathcal{L}_{\text{I-NMF}} = \left\|\mathbf{M}_* - \mathbf{U}\mathbf{V}_*^{\mathrm{T}}\right\|_F + \left\|\mathbf{M}^* - \mathbf{U}\mathbf{V}^{*\mathrm{T}}\right\|_F,$$

where $\mathbf{U} \geq 0$, $\mathbf{V}_* \geq 0$ and $\mathbf{V}^* \geq 0$. The iterative update rules to minimize this loss function are:

$$\mathbf{U}_{[i,j]} \leftarrow \mathbf{U}_{[i,j]} \frac{\left(\mathbf{M}_*\mathbf{V}_*^{\mathrm{T}} + \mathbf{M}^*\mathbf{V}^{*\mathrm{T}}\right)_{[i,j]}}{\left(\mathbf{U}\mathbf{V}_*\mathbf{V}_*^{\mathrm{T}} + \mathbf{U}\mathbf{V}^*\mathbf{V}^{*\mathrm{T}}\right)_{[i,j]}},$$

$$\mathbf{V}_{*[i,j]} \leftarrow \mathbf{V}_{*[i,j]} \frac{\left(\mathbf{U}^{\mathrm{T}}\mathbf{M}_*\right)_{[i,j]}}{\left(\mathbf{U}^{\mathrm{T}}\mathbf{U}\mathbf{V}_*\right)_{[i,j]}},$$

$$\mathbf{V}^*_{[i,j]} \leftarrow \mathbf{V}^*_{[i,j]} \frac{\left(\mathbf{U}^{\mathrm{T}}\mathbf{M}^*\right)_{[i,j]}}{\left(\mathbf{U}^{\mathrm{T}}\mathbf{U}\mathbf{V}^*\right)_{[i,j]}}.$$

Once the update rules reach a minimum, an approximation of the interval-valued input matrix $\ddot{\mathbf{M}}$ can be evaluated as

$$\tilde{\ddot{\mathbf{M}}} = \left[\widetilde{\mathbf{M}}_*, \widetilde{\mathbf{M}}^*\right] = \left[\mathbf{U}(\mathbf{V}_*)^{\mathrm{T}}, \mathbf{U}(\mathbf{V}^*)^{\mathrm{T}}\right].$$

Interval-valued PMF. Also in [46], the authors present an Interval-valued PMF technique (referred to as I-PMF) which has the purpose of improving the task of predicting the missing entries of a user-item rating matrix drawing on the idea that an interval-valued rating system would be better suited to capture the preferences expressed by each user.

More in detail, taking a scalar rating matrix $\mathbf{R} \in \mathbb{R}^{n \times m}$, where $1 \leq i \leq n$ (number of users), $1 \leq j \leq m$ (number of items) and $\mathbf{R}_{[i,j]}$ represents the rating that user i assigned to item j, an interval-valued matrix $\ddot{\mathbf{M}} = [\mathbf{M}_*, \mathbf{M}^*]$, with $\mathbf{M}_*, \mathbf{M}^* \in \mathbb{R}^{n \times m}$, is generated by expanding each scalar rating $\mathbf{R}_{[i,j]}$ to a range

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

where the radius $\delta_{i,j}$ is evaluated in terms of the standard deviation over the set of all the ratings in $\mathbf{R}$ that are related to $\mathbf{R}_{[i,j]}$ (i.e., user i's ratings on different items than j, or the ratings for item j from users other than i)[2]:

$$\delta_{i,j} \coloneqq \alpha \cdot \mathrm{std}\left(\left\{\mathbf{R}_{[i',j']} | (i' = i \vee j' = j) \wedge (i', j') \in (\mathbf{i}, \mathbf{j})\right\}\right),$$

where $\alpha \in \mathbb{R}^+$ is a multiplicative scale coefficient.

Then, the factorization of the newly obtained interval-valued matrix $\ddot{\mathbf{M}}$ is processed in terms of a scalar factor matrix $\mathbf{U} \in \mathbb{R}^{n \times r}$, representing the user-specific latent feature vectors, and an interval-valued factor matrix $\ddot{\mathbf{V}} = [\mathbf{V}_*, \mathbf{V}^*]$, with $\mathbf{V}_*, \mathbf{V}^* \in \mathbb{R}^{m \times r}$, representing the item-specific latent feature vectors.

To evaluate these factor matrices, they first of all proceed with the introduction of additional constraints on the loss function (see equation 3.1.4 in Section 3.1.4) in order to deal with the newly defined interval-valued entries:

$$\mathcal{L}_{\text{I-PMF}} = \left\|\mathbf{M}_* - \mathbf{U}\mathbf{V}_*^{\mathrm{T}}\right\|_F + \left\|\mathbf{M}^* - \mathbf{U}\mathbf{V}^{*\mathrm{T}}\right\|_F + \lambda_{\mathbf{U}} \|\mathbf{U}\|_F + \lambda_{\mathbf{V}} \left(\|\mathbf{V}_*\|_F + \|\mathbf{V}^*\|_F\right).$$

Given this loss formulation, a local minimum can be sought by applying a gradient descent in $U_{[i,:]}$, $V_{*[j,:]}$ and $V^*_{[j,:]}$ using the following partial derivatives:

$$\frac{\partial \mathcal{L}_{\text{I-PMF}}}{\partial \mathbf{U}_{[i,:]}} = \sum_{j=1}^{m} \left(\mathbf{U}_{[i,:]} \left(\mathbf{V}_{*[j,:]}\right)^{\mathrm{T}} - \mathbf{M}_{*[i,j]}\right) \mathbf{V}_{*[j,:]} + \left(\mathbf{U}_{[i,:]} \left(\mathbf{V}^*_{[j,:]}\right)^{\mathrm{T}} - \mathbf{M}^*_{[i,j]}\right) \mathbf{V}^*_{[j,:]} + \lambda_{\mathbf{U}} \mathbf{U}_{[i,:]},$$

$$\frac{\partial \mathcal{L}_{\text{PMF}}}{\partial \left(\mathbf{V}_{*[j,:]}\right)^{\mathrm{T}}} = \sum_{i=1}^{n} \left(\mathbf{U}_{[i,:]} \left(\mathbf{V}_{*[j,:]}\right)^{\mathrm{T}} - \mathbf{R}_{[i,j]}\right) \left(\mathbf{U}_{[i,:]}\right)^{\mathrm{T}} + \lambda_{\mathbf{V}} \left(\mathbf{V}_{*[j,:]}\right)^{\mathrm{T}},$$

$$\frac{\partial \mathcal{L}_{\text{PMF}}}{\partial \left(\mathbf{V}^*_{[j,:]}\right)^{\mathrm{T}}} = \sum_{i=1}^{n} \left(\mathbf{U}_{[i,:]} \left(\mathbf{V}^*_{[j,:]}\right)^{\mathrm{T}} - \mathbf{R}_{[i,j]}\right) \left(\mathbf{U}_{[i,:]}\right)^{\mathrm{T}} + \lambda_{\mathbf{V}} \left(\mathbf{V}^*_{[j,:]}\right)^{\mathrm{T}}.$$

[2]The idea behind this approach is that the higher the variance for a user i's ratings on different items, or for the ratings of an item j from different users, the larger the interval-valued entry $\ddot{\mathbf{M}}_{[i,j]}$ should be.

Finally, once $\mathbf{U}$, $\mathbf{V}_*$ and $\mathbf{V}_*$ reach a good approximation, the unknown ratings in $\mathbf{R}$ can be estimated by computing

$$\widetilde{\mathbf{R}}_{[i,j]} = \frac{\mathbf{U}_{[i,:]}\mathbf{V}_{*[:,j]}^{\mathrm{T}} + \mathbf{U}_{[i,:]}\mathbf{V}_{[:,j]}^{*\mathrm{T}}}{2}. \tag{3.1}$$

Chapter 4

Matrix factorization with interval-valued data

This chapter provides a detailed description of how the factorization techniques illustrated in Section 3.1 can be extended for dealing with interval valued matrices, and it is based on the work by our research team, described in [34].

4.1 Interval-valued Singular Value Decomposition

As we saw in Section 3.1.2, a scalar-valued matrix $\mathbf{M} \in \mathbb{R}^{n \times m}$ can be factorized, by means of the Singular Value Decomposition (SVD), as

$$\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}.$$

Since, by definition, the factor matrices $\mathbf{U}$ and $\mathbf{V}$ are orthonormal, a common interpretation of the SVD is to regard the columns of $\mathbf{V}$ as *basis vectors* of a new vector space, also referred to as *latent semantic space*. More in detail, the columns of $\mathbf{V}$ identify a new space to map the data stored in $\mathbf{M}$ (represented by its row vectors) in such a way that the new dimensions, also referred to as *principal components*, are oriented in the directions of maximum variability of the data, in decreasing order of variance going from one dimension to the other (see Figure 4.1).

As an example, Figure 4.2 shows a scalar-valued latent semantic space superimposed on the original bidimensional space where the entries of a given matrix

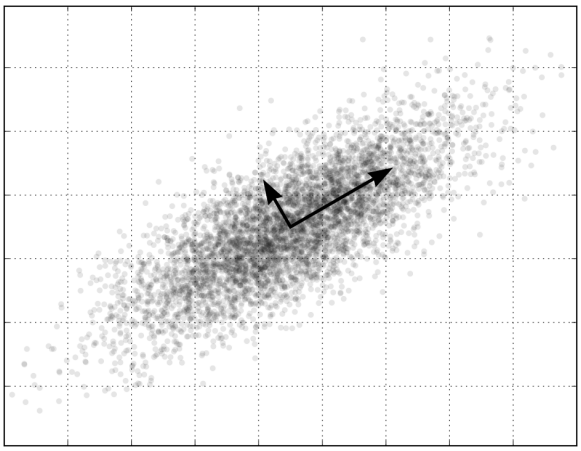


Figure 4.1: Principal components in a bidimensional space

$\mathbf{M} \in \mathbb{R}^{n\times 2}$ (where an entry i is denoted as the i-th row vector $\mathbf{M}_{[i,:]}$) are mapped. As we can see, a generic entry $\mathbf{M}_{[i,:]}$ is projected on the original space by means of the components $\mathbf{M}_{[i,1]}$ and $\mathbf{M}_{[i,2]}$, i.e., the corresponding values on the columns of $\mathbf{M}$. The same entry, once $\mathbf{M}$ is decomposed (with target rank 2 in this case) as $\mathbf{U} \in \mathbb{R}^{n\times 2}$, $\mathbf{\Sigma} = \text{diag}(\sigma_1, \sigma_2) \in \mathbb{R}^{2\times 2}$ and $\mathbf{V} \in \mathbb{R}^{2\times 2}$, can be projected on the newly found vector space (described by the columns of $\mathbf{V}$), where each component is identified by the stretched unit vectors $\sigma_1\mathbf{U}_{[i,1]}$ and $\sigma_2\mathbf{U}_{[i,2]}$.

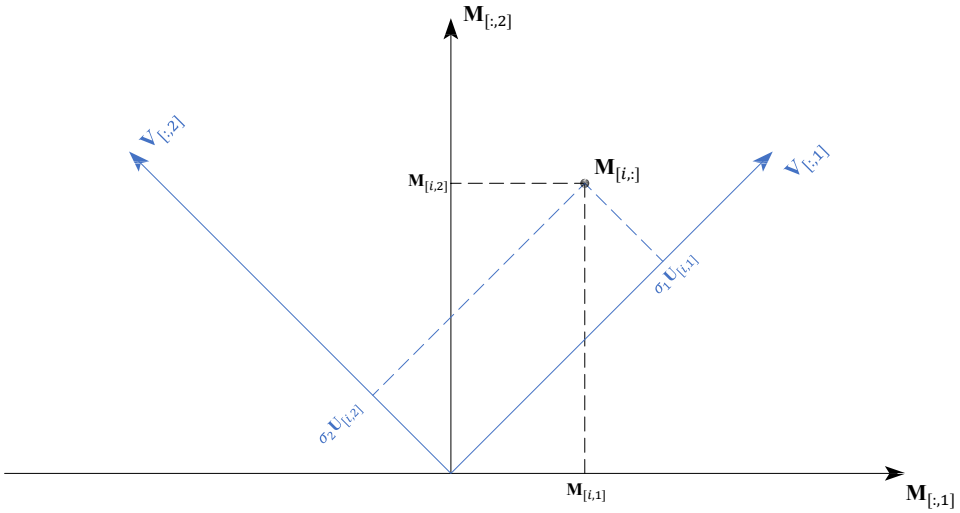


Figure 4.2: Scalar latent semantic space

In light of these considerations, the definition of SVD reported in Section 3.1.2 can readily be extended for dealing with interval-valued matrices, although with

some caveats. Given an interval-valued matrix $\ddot{\mathbf{M}} \in \mathbb{R}^{n \times m}$ and a target rank $r \leq \min(n,m)$, we can define the interval-valued SVD as the factorization

$$\ddot{\mathbf{M}} \rightarrow \left[\ddot{\mathbf{U}}, \ddot{\mathbf{\Sigma}}, \ddot{\mathbf{V}}\right], \tag{4.1}$$

where $\ddot{\mathbf{U}} \in \mathbb{R}^{n \times r}$, $\ddot{\mathbf{\Sigma}} \in \mathbb{R}^{r \times r}$ and $\ddot{\mathbf{V}} \in \mathbb{R}^{m \times r}$ are (potentially) interval-valued factor matrices, and the columns of $\ddot{\mathbf{U}}$ and $\ddot{\mathbf{V}}$ are *quasi*-orthonormal, i.e., it holds that

$$\ddot{\mathbf{U}}^{\mathrm{T}} \otimes \ddot{\mathbf{U}} \simeq \mathbf{I}_n \;\text{ and }\; \ddot{\mathbf{V}}^{\mathrm{T}} \otimes \ddot{\mathbf{V}} \simeq \mathbf{I}_m. \tag{4.2}$$

4.1.1 Imprecision of the Interval-valued SVD

Since the orthonormality constraints defined above cannot be completely enforced, a perfectly reversible decomposition by means of the factor matrices from equation 4.1 can never be reached, resulting in an inherent imprecision in the process. The reason for this unavoidable relaxation in the orthogonality constraints is rooted in how the interval multiplication is defined (see Section 2.1.3), and can be formalized by the following theorem.

Theorem 1. *Given two interval-valued vectors of the same length, $\ddot{\mathbf{x}}, \ddot{\mathbf{y}} \in \mathbb{R}^n$, their interval dot product (here denoted as $\ddot{\mathbf{x}} \odot \ddot{\mathbf{y}}$) is scalar-valued only if all the components of $\ddot{\mathbf{x}}$ and $\ddot{\mathbf{y}}$ are themselves scalar-valued, or if either one of the two vectors is zero (i.e., $\ddot{\mathbf{x}} = \mathbf{0} \vee \ddot{\mathbf{y}} = \mathbf{0}$).*

Proof. The algebraic definition of dot product between two vectors can be readily extended to the interval case as follows:

$$\ddot{\mathbf{x}} \odot \ddot{\mathbf{y}} = \sum_{i=1}^{n} \ddot{x}_i \otimes \ddot{y}_i,$$

which, by definition of interval multiplication, $\otimes$, can be expressed as

$$\begin{aligned}\ddot{\mathbf{x}} \odot \ddot{\mathbf{y}} &= \sum_{i=1}^{n} \left[\min\left(\mathbf{x}_{*[i]}\mathbf{y}_{*[i]}, \mathbf{x}_{*[i]}\mathbf{y}^*_{[i]}, \mathbf{x}^*_{[i]}\mathbf{y}_{*[i]}, \mathbf{x}^*_{[i]}\mathbf{y}^*_{[i]}\right), \max\left(\mathbf{x}_{*[i]}\mathbf{y}_{*[i]}, \mathbf{x}_{*[i]}\mathbf{y}^*_{[i]}, \mathbf{x}^*_{[i]}\mathbf{y}_{*[i]}, \mathbf{x}^*_{[i]}\mathbf{y}^*_{[i]}\right)\right] = \\ &= \left[\sum_{i=1}^{n} \min\left(\mathbf{x}_{*[i]}\mathbf{y}_{*[i]}, \mathbf{x}_{*[i]}\mathbf{y}^*_{[i]}, \mathbf{x}^*_{[i]}\mathbf{y}_{*[i]}, \mathbf{x}^*_{[i]}\mathbf{y}^*_{[i]}\right), \sum_{i=1}^{n} \max\left(\mathbf{x}_{*[i]}\mathbf{y}_{*[i]}, \mathbf{x}_{*[i]}\mathbf{y}^*_{[i]}, \mathbf{x}^*_{[i]}\mathbf{y}_{*[i]}, \mathbf{x}^*_{[i]}\mathbf{y}^*_{[i]}\right)\right].\end{aligned}$$

Since the product of two non-empty, bounded, real intervals, is itself an interval (see Theorem 2 in [25]), the only way for the overall interval dot product to be a

scalar, i.e., for the equality

$$\sum_{i=1}^{n} \min\left(\mathbf{x}_{*[i]}\mathbf{y}_{*[i]}, \mathbf{x}_{*[i]}\mathbf{y}^{*}_{[i]}, \mathbf{x}^{*}_{[i]}\mathbf{y}_{*[i]}, \mathbf{x}^{*}_{[i]}\mathbf{y}^{*}_{[i]}\right) = \sum_{i=1}^{n} \max\left(\mathbf{x}_{*[i]}\mathbf{y}_{*[i]}, \mathbf{x}_{*[i]}\mathbf{y}^{*}_{[i]}, \mathbf{x}^{*}_{[i]}\mathbf{y}_{*[i]}, \mathbf{x}^{*}_{[i]}\mathbf{y}^{*}_{[i]}\right)$$

to hold, is if $(\mathbf{x}_{*[i]} = \mathbf{x}^{*}_{[i]}) \wedge (\mathbf{y}_{*[i]} = \mathbf{y}^{*}_{[i]})$, $\forall i = 1,\ldots,n$, or if $\ddot{\mathbf{x}} = \mathbf{0} \vee \ddot{\mathbf{y}} = \mathbf{0}$, as we wanted to prove. □

Going back to the equations 4.2, we can rewrite the interval-valued matrix products (and their relation to the identity matrices) as the following interval dot products:

$$\begin{aligned}
&a)\quad \ddot{\mathbf{U}}_{[:,i]} \odot \ddot{\mathbf{U}}_{[:,i]} \simeq 1, && \forall i = 1,\ldots,r \\
&b)\quad \ddot{\mathbf{V}}_{[:,i]} \odot \ddot{\mathbf{V}}_{[:,i]} \simeq 1, && \forall i = 1,\ldots,r \\
&c)\quad \ddot{\mathbf{U}}_{[:,i]} \odot \ddot{\mathbf{U}}_{[:,j]} \simeq 0, && \forall i,j = 1,\ldots,r,\ i \neq j \\
&d)\quad \ddot{\mathbf{V}}_{[:,i]} \odot \ddot{\mathbf{V}}_{[:,j]} \simeq 0, && \forall i,j = 1,\ldots,r,\ i \neq j
\end{aligned}$$

These mean that, for the interval factor matrices $\ddot{\mathbf{U}}$ and $\ddot{\mathbf{V}}$ to be orthonormal, the equations above should all be exactly equal to either 1 or 0; however, Theorem 1 makes us conclude that:

- Equations (a) and (b) cannot be exact, because for the interval dot product to be equivalent to a non-zero scalar, also the columns of $\ddot{\mathbf{U}}$ and $\ddot{\mathbf{V}}$ should be scalar, which is not the case.
- Equations (c) and (d) cannot be exact, because for the interval dot product to be zero, at least one of the columns of $\ddot{\mathbf{U}}$ and $\ddot{\mathbf{V}}$ involved in the products should have length 0, which again cannot be the case, since, being the factor matrices *quasi*-orthonormal, their columns approximate the unit vector, i.e., a vector of length 1.

Thus, we have to conclude that $\ddot{\mathbf{U}}$ and $\ddot{\mathbf{V}}$ can never be exactly orthonormal, and that an inherent imprecision in the decomposition is inevitable.

Figure 4.3 and 4.4 help visualize the inherent imprecision of an interval-valued latent space. In the first figure, we can first of all see how a data entry $\ddot{\mathbf{M}}_{[i,:]}$ from an interval-valued matrix $\ddot{\mathbf{M}} = [\mathbf{M}_*, \mathbf{M}^*]$, where $\mathbf{M}_*, \mathbf{M}^* \in \mathbb{R}^{n\times 2}$, is mapped in the original space (bidimensional in the example): while a scalar entry is represented as a point (see Figure 4.2), here, instead, an interval-valued entry identifies a region

of the space (a rectangle in this bidimensional example, a high-dimensional box in general) bounded by the projections of the endpoints of each interval-valued component (i.e., the columns of $\ddot{\mathbf{M}}$ on the i-th row). In the same figure, the basis vectors of the latent semantic space, $\ddot{\mathbf{V}}_{[:,1]}$ and $\ddot{\mathbf{V}}_{[:,2]}$, are represented: as we can see, each vector of the interval-valued basis is actually identified by a pair of vectors, one for the minimum values in the interval-valued components (the columns of $\mathbf{V}_*$), one for the maximum values (the columns of $\mathbf{V}^*$). It is easy to see, then, why the orthonormality constraint regarding $\ddot{\mathbf{V}}$ needs to be relaxed, and where a first cause of approximation comes from.

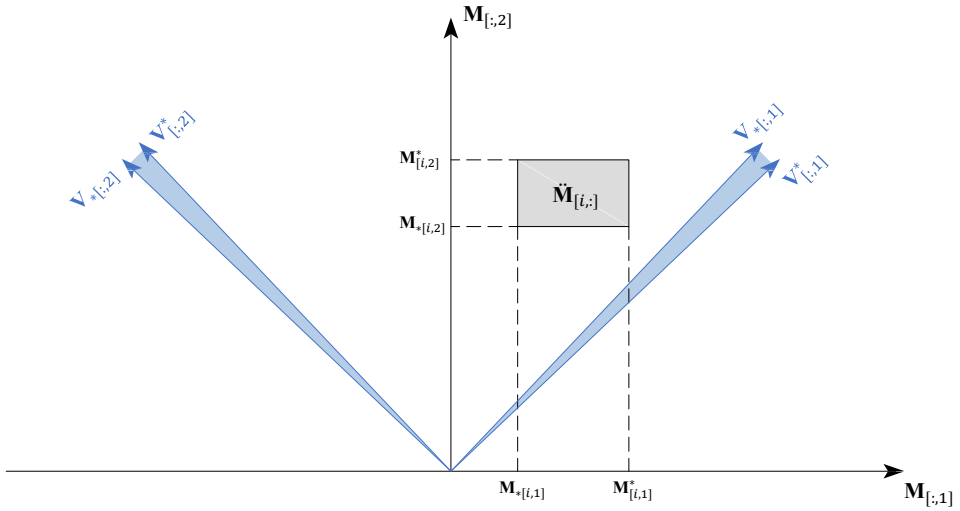


Figure 4.3: Interval-valued latent semantic space

Figure 4.4 shows how the interval-valued data entry $\ddot{\mathbf{M}}_{[i,:]}$ maps in the newly found latent semantic space, identified by the column vectors of $\ddot{\mathbf{V}}$ (approximated in figure as an orthonormal basis to help the visualization): the projection of the data entry onto each basis vectors, $\ddot{\mathbf{V}}_{[:,1]}$ and $\ddot{\mathbf{V}}_{[:,2]}$, is again represented by an interval, this time identified by the products $\ddot{\mathbf{U}}_{[i,1]} \otimes \ddot{\mathbf{\Sigma}}_{[i,1]}$ and $\ddot{\mathbf{U}}_{[i,2]} \otimes \ddot{\mathbf{\Sigma}}_{[i,2]}$, respectively. Since the interval multiplication is a costly operation (especially when matrices are involved), it usually requires to be approximated, further decreasing the accuracy of the decomposition process.

4.1.2 Accuracy measure for the interval-valued SVD

Given the inherent imprecision in the decomposition process, the recombination of the factor matrices (by means of the interval-valued matrix product) will

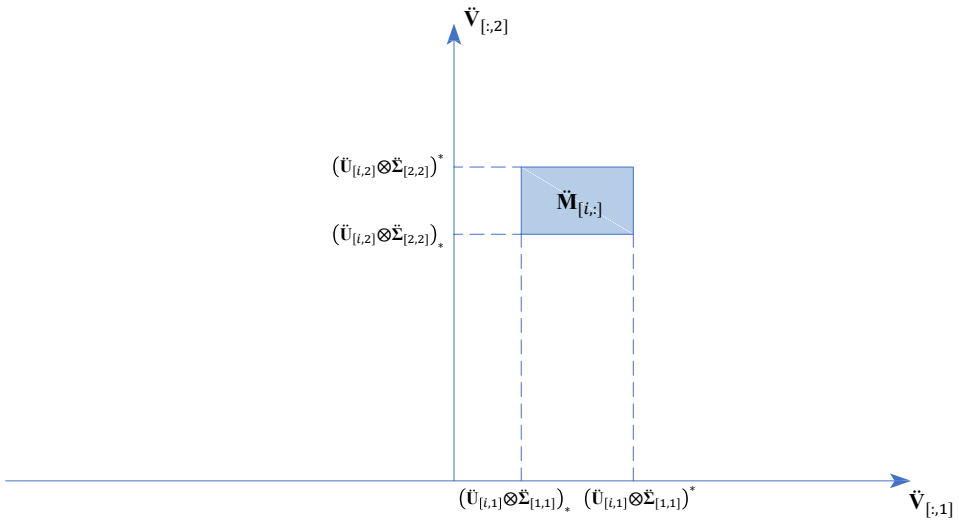


Figure 4.4: Mapping of an object in an interval-valued latent semantic space

produce an approximation $\tilde{\ddot{\mathbf{M}}}$ of the original matrix $\ddot{\mathbf{M}}$, such that

$$\tilde{\ddot{\mathbf{M}}} = \ddot{\mathbf{U}} \otimes \ddot{\mathbf{\Sigma}} \otimes \ddot{\mathbf{V}}^{\mathrm{T}},$$

where $\tilde{\ddot{\mathbf{M}}} = \left[\widetilde{\mathbf{M}}_*, \widetilde{\mathbf{M}}^*\right]$ and $\tilde{\ddot{\mathbf{M}}} \simeq \ddot{\mathbf{M}}$. The discrepancy between $\tilde{\ddot{\mathbf{M}}}$ and $\ddot{\mathbf{M}}$, can be measured by means of the normalized Frobenius norm of the difference of their endpoint matrices:

$$\Delta\left(\mathbf{M}_*, \widetilde{\mathbf{M}}_*\right) = \frac{\left\|\mathbf{M}_* - \widetilde{\mathbf{M}}_*\right\|_F}{\|\mathbf{M}_*\|_F},$$
$$\Delta\left(\mathbf{M}^*, \widetilde{\mathbf{M}}^*\right) = \frac{\left\|\mathbf{M}^* - \widetilde{\mathbf{M}}^*\right\|_F}{\|\mathbf{M}^*\|_F}.$$

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

$$\Theta\left(\mathbf{M}_*, \widetilde{\mathbf{M}}_*\right) = \max\left[0, 1 - \Delta\left(\mathbf{M}_*, \widetilde{\mathbf{M}}_*\right)\right],$$
$$\Theta\left(\mathbf{M}^*, \widetilde{\mathbf{M}}^*\right) = \max\left[0, 1 - \Delta\left(\mathbf{M}^*, \widetilde{\mathbf{M}}^*\right)\right].$$

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

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

4.2 Interval Latent Semantic Alignment (ILSA)

As described in section 4.1.1, one inherent form of imprecision in the interval SVD process (equation 4.1) is due to the impossibility of finding perfectly orthonormal factor matrices, $\ddot{\mathbf{U}}$ and $\ddot{\mathbf{V}}$. But there is also another aspect to take into consideration: if we consider the endpoints (scalar) matrices $\mathbf{M}_*$ and $\mathbf{M}^*$ that form $\ddot{\mathbf{M}}$, their factorization can be done independently (through a standard SVD), obtaining, respectively

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

We could then be tempted to complete the interval decomposition process by directly recombining the minimum and maximum factor matrices:

$$\ddot{\mathbf{U}}=\left[\mathbf{U}_*,\mathbf{U}^*\right],\ \ddot{\mathbf{\Sigma}}=\left[\mathbf{\Sigma}_*,\mathbf{\Sigma}^*\right],\text{ and }\ddot{\mathbf{V}}=\left[\mathbf{V}_*,\mathbf{V}^*\right],$$

but this approach suffers from three major issues:

1. Since the latent components in the factor matrices $\mathbf{U}_*$ and $\mathbf{V}_*$ (as well as in $\mathbf{U}^*$ and $\mathbf{V}^*$) are arranged according to the magnitude of the relative singular values (in descending order) in $\mathbf{\Sigma}_*$ (respectively $\mathbf{\Sigma}^*$), there is no guarantee that the singular vectors that express the same concept in the minimum and maximum matrices will end up in the same position, as the related singular values may end up in different places in the ordering. Figure 4.5 illustrates this issue, showing how some of the column vectors (whose latent semantics is color coded) in $\mathbf{U}_*$ and $\mathbf{U}^*$ (and, likewise, in $\mathbf{V}_*$ and $\mathbf{V}^*$) may be mismatched.

2. Moreover, any two given latent components, $\mathbf{v}_{*i}$ and $\mathbf{v}_i^*$, that are (approximately) collinear, may still be oriented in the opposite direction (in general, the orientation of the singular vectors is not relevant for the decomposition,

only their direction).

3. There is also no guarantee that the endpoints in $[\mathbf{U}_*, \mathbf{U}^*]$, $[\mathbf{\Sigma}_*, \mathbf{\Sigma}^*]$ and $[\mathbf{V}_*, \mathbf{V}^*]$ would form valid intervals, namely that $\mathbf{U}_{*[i,j]} \leq \mathbf{U}^*_{[i,j]}$, $\mathbf{V}_{*[i,j]} \leq \mathbf{V}^*_{[i,j]}$ $\forall i, j$ and $\sigma_{*i} \leq \sigma^*_i$ $\forall i$.

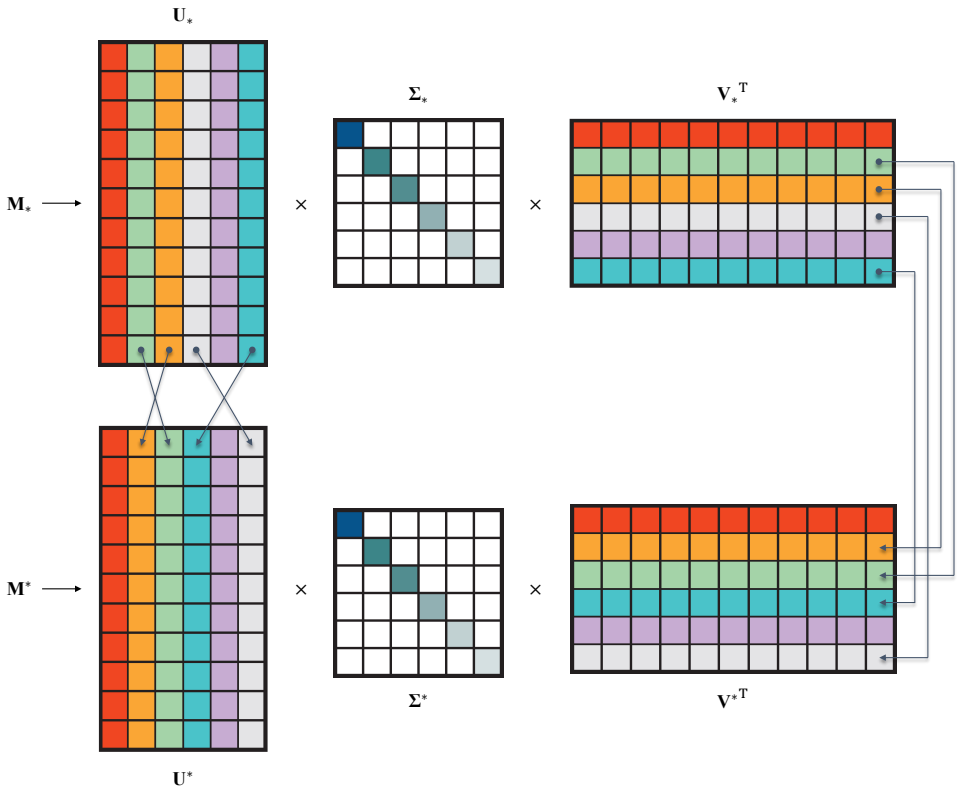


Figure 4.5: Misaligned latent semantic components: after an independent factorization of $\mathbf{M}_*$ and $\mathbf{M}^*$, the singular vectors expressing the same concepts (highlighted with the same colors in the factor matrices) need to be *semantically aligned*.

To solve this issues, we can first of all take into consideration an alternative (albeit equivalent) way to represent the SVD of a matrix $\mathbf{M} \in \mathbb{R}^{n \times m}$ of rank $r \leq \min(n,m)$, namely by means of the outer product, $\circ$, between the singular (column) vectors in $\mathbf{U} = (\mathbf{u}_1\ \mathbf{u}_2\ \cdots\ \mathbf{u}_r)$ and $\mathbf{V} = (\mathbf{v}_1\ \mathbf{v}_2\ \cdots\ \mathbf{v}_r)$, scaled by the corresponding singular values $\mathbf{\Sigma} = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}} = \sum_{i=1}^{r} \sigma_i \cdot \mathbf{u}_i \circ \mathbf{v}_i$$

This corresponds to a summation of r rank-1 matrices (see Figure 4.6), where,

since the addition is commutative, the order of the terms is irrelevant. Thanks to this property, we can address point 1 in the list above by formulating a *latent semantic alignment* problem, which would match minimum and maximum vectors, $\mathbf{V}_*$ and $\mathbf{V}^*$, according to their common semantic, without compromising the independent reconstructions results, thus forming a coherent (though approximated) interval-valued latent space.

The main idea behind this approach is that the factor matrices $\mathbf{V}_*$ and $\mathbf{V}^*$ (but the same is also true for $\mathbf{U}_*$ and $\mathbf{U}^*$) represent two different sets of latent semantics corresponding to the minimum and maximum matrices $\mathbf{M}_*$ and $\mathbf{M}^*$, respectively; yet, since they are derived from the same interval-valued input matrix, $\ddot{\mathbf{M}}$, we would expect that the singular vectors corresponding to a related latent semantics would be relatively similar to each other. In other words, if $\mathbf{v}_*$ and $\mathbf{v}^*$ are two singular vectors corresponding to a related latent semantics, we expect that $\mathbf{v}_* \simeq \mathbf{v}^*$. The *latent semantic alignment* problem can be formalized as follows:

Problem 1 (Optimal Min-Max Vector Alignment)**.** *Given two sets of unit vectors* $V_* = \{\mathbf{v}_{*1}, \mathbf{v}_{*2}, \ldots, \mathbf{v}_{*r}\}$ *and* $V^* = \{\mathbf{v}_1^*, \mathbf{v}_2^*, \ldots, \mathbf{v}_r^*\}$*, and given a cost function between any two vectors,* $\mathbf{x}$ *and* $\mathbf{y}$*, defined as* $1 - |\cos(\mathbf{x}, \mathbf{y})|$*, our goal is to find a mapping* $\mu = \{\mu_1, \mu_2, \ldots, \mu_r\}$ *(where* $\mu_i = \langle \mu_{*i}, \mu_i^* \rangle$*) among the vectors in* V_* *and* V^**, that minimizes the cumulative cost function*

$$\sum_{i=1}^{r} 1 - \left| \cos(\mathbf{v}_{*\mu_{*i}}, \mathbf{v}^*_{\mu_i^*}) \right|.$$

This is an instance of the *linear assignment* problem [5], and can be solved using one of the many algorithms in the literature, such as the Jonker-Volgenant formulation [28], which finds an optimal solution in $\mathcal{O}(r^3)$ time.

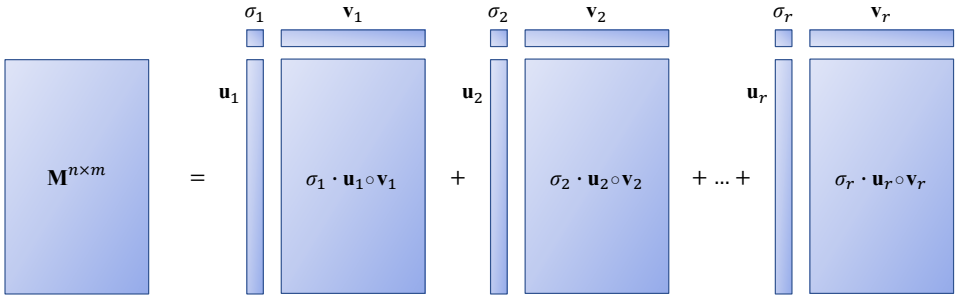


Figure 4.6: SVD as a summation of rank-1 matrices

The purpose of the mapping is to match the pairs of minimum and maximum latent components (i.e., the columns of $\mathbf{V}_*$ and $\mathbf{V}^*$) that are as aligned as possible, namely finding the pairs of vectors such that the cosine of the angle between them is as close as possible to 1 (in other words, minimizing their *cosine distance*).

More in detail, we are interested in finding those latent components that are as collinear as possible, even if oriented in the opposite direction (hence the absolute value after evaluating the cosine between the two vectors). Once a mapping μ has been found, for any $1 \leq i \leq r$ such that $\cos(\mathbf{v}_{*\mu_{*i}}, \mathbf{v}^*_{\mu^*_i})$ is negative, we can then effectively align their directions, multiplying $\mathbf{v}^*_i$ (as well as $\mathbf{u}^*_i$, to balance the SVD equation[1]) by -1, so that both $\mathbf{v}_{*i}$ and $\mathbf{v}^*_i$ will have the same orientation, solving the second issue raised in the list above.

To solve the third issue, and finally validate the interval-valued decomposition process, we replace the single entries of the factor matrices that do not lead to a valid interval with their average, compromising a bit on local accuracy (i.e., how accurately the independent SVDs of $\mathbf{M}_*$ and $\mathbf{M}^*$ could be reversed) for the general validity of the process:

$$\mathbf{U}_{*[i,j]} = \mathbf{U}^*_{[i,j]} = \frac{\mathbf{U}_{*[i,j]} + \mathbf{U}^*_{[i,j]}}{2}, \ \forall i,j \mid \mathbf{U}_{*[i,j]} > \mathbf{U}^*_{[i,j]},$$

$$\mathbf{V}_{*[i,j]} = \mathbf{V}^*_{[i,j]} = \frac{\mathbf{V}_{*[i,j]} + \mathbf{V}^*_{[i,j]}}{2}, \ \forall i,j \mid \mathbf{V}_{*[i,j]} > \mathbf{V}^*_{[i,j]},$$

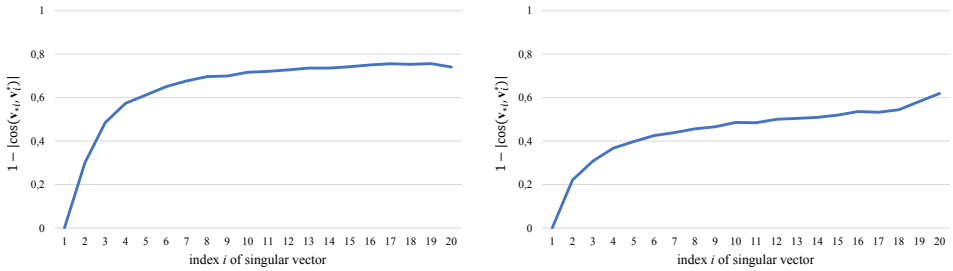
$$\sigma_{*i} = \sigma^*_i = \frac{\sigma_{*i} + \sigma^*_i}{2}, \ \forall i \mid \sigma_{*i} > \sigma^*_i.$$

Figure 4.7 shows how the remapping improves the alignment of the latent components (i.e., the columns of $\mathbf{V}_*$ and $\mathbf{V}^*$) of an interval-valued matrix $\ddot{\mathbf{M}}$. More in detail, in (a) is reported the absolute cosine distance between each pair $\langle \mathbf{v}_{*i}, \mathbf{v}^*_i \rangle$, and we can see how this metric drops (b) once the remapping is applied.The graphs in figure are obtained by evaluating the pairwise cosine distances over 1000 randomly generated sample matrices (with $n = 250$ and $m = 40$) decomposed with target rank $r = 20$, since in general many factorization-based applications work with low rank

[1] It is a property of the Singular Valued Decomposition that, for every $i = 1, \ldots, k$,

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

values (nevertheless, the results are similar also with a full-rank decomposition).



(a) Before vector remapping (b) After vector remapping

Figure 4.7: Effects of the Interval Latent Semantic Alignment (ILSA) on the latent components of an interval-valued matrix

For the experiments that generated the graphs in figure, the cost function has been evaluated considering the absolute value of the vectors cosine (thus limiting the function between 0 and 1), in order to avoid, in the (a) case, to take into account the error resulting from the orientation misalignment of the vectors, while focusing the attention on the improvement resulting in (b) just from the semantic-based remapping. In a similar way, the correction of the non-valid interval entries by averaging has not been performed for the (b) case, since it would misrepresent the results by causing a further improvement simply for the fact that the vectors would appear *closer* just for having more components in common (namely the ones replaced by their average).

The graphs also show how the alignment is better for the first latent components (corresponding to high-magnitude singular values), while it consistently drops moving along the vectors. This is a positive thing, considering that the contribution of the later dimensions (corresponding to low-magnitude singular values) tends in general to be small, and can be dropped with minimal loss of information.

4.3 Alternative decomposition options

As we saw in the previous sections, a decomposition of an interval-valued matrix $\ddot{\mathbf{M}}$ is possible, but it comes with a few drawbacks. Moreover, having interval-valued factor matrices, $\ddot{\mathbf{U}}$, $\ddot{\mathbf{\Sigma}}$ and $\ddot{\mathbf{V}}$, may not be advisable in all contexts, in particular regarding the non orthogonality of singular vectors, which could be challenging to handle in some situations more than others. Different applications, in fact, might

have different requirements in this terms, some favoring generality over precision, according to the desired outcome. Taking these aspects into account, we have considered three distinct application scenarios:

4.3.1 Decomposition Target (a): interval-valued $\ddot{\mathbf{U}}$, $\ddot{\mathbf{\Sigma}}$ and $\ddot{\mathbf{V}}$

In this most general option, all three factor matrices are treated as interval-valued, and no particular constraints are enforced. More in detail, after the independent decomposition of $\mathbf{M}_*$ and $\mathbf{M}^*$, the ILSA procedure (Section 4.2) is applied to guarantee that the latent semantic components are correctly matched and aligned in the minimum and maximum factor matrices, and that all the entries in the three interval-valued factor matrices are actually valid intervals.

4.3.2 Decomposition Target (b): scalar U and V, interval-valued $\ddot{\mathbf{\Sigma}}$

As stated above, some applications may have issues with the fact that interval-valued singular vectors (i.e., the columns of $\ddot{\mathbf{U}}$ and $\ddot{\mathbf{V}}$) cannot be orthonormal, since it is difficult to interpret them as the alternative bases of a vector space. The purpose of this strategy is to provide a set of scalar factor matrices, $\mathbf{U}$ and $\mathbf{V}$, as close as possible to being orthonormal, while capturing (at least part of) the underlying *interval information* in an interval-valued $\ddot{\mathbf{\Sigma}}$. Intuitively, this would provide alternative basis vectors (although approximated) for the rows and columns of the input matrix $\ddot{\mathbf{M}}$, and an interval-valued magnitude for each basis vector (captured by the corresponding singular value).

More in detail, an additional set of steps can be added to the procedure described for option (a), to obtain $\mathbf{U}$ and $\mathbf{V}$ as a scalar approximation of $\ddot{\mathbf{U}}$ and $\ddot{\mathbf{V}}$ and to (at least) guarantee their columns to be unit vectors, i.e., of having L2-norm equal to 1 (a property that cannot even be defined for an interval valued vector).

The first step consists in replacing all the interval-valued entries in $\ddot{\mathbf{U}}$ and $\ddot{\mathbf{V}}$ with the average of their endpoints, thus obtaining the scalar matrices $\mathbf{U}$ and $\mathbf{V}$, such that, $\forall j = 1,\ldots,r$,

$$\overline{\mathbf{U}}_{[i,j]} = \frac{\mathbf{U}_{*[i,j]} + \mathbf{U}^*_{[i,j]}}{2}, \ \forall i = 1,\ldots,n, \tag{4.3}$$

$$\overline{\mathbf{V}}_{[i,j]} = \frac{\mathbf{V}_{*[i,j]} + \mathbf{V}^*_{[i,j]}}{2}, \ \forall i = 1,\ldots,m. \tag{4.4}$$

Then, we can evaluate the L2-norms of their columns, namely, $\forall j = 1,\ldots,r$,

$$w_j^{\overline{\mathbf{U}}} = \left\| \overline{\mathbf{U}}_{[:,j]} \right\|_2 = \sqrt{\sum_{i=1}^{n} \left(\overline{\mathbf{U}}_{[i,j]} \right)^2},$$
$$w_j^{\overline{\mathbf{V}}} = \left\| \overline{\mathbf{V}}_{[:,j]} \right\|_2 = \sqrt{\sum_{i=1}^{m} \left(\overline{\mathbf{V}}_{[i,j]} \right)^2},$$

and use these weights to normalize the factor matrices $\mathbf{U}$ and $\mathbf{V}$, i.e., $\forall j = 1,\ldots,r$,

$$\mathbf{U}_{[i,j]} = \frac{\overline{\mathbf{U}}_{[i,j]}}{w_j^{\overline{\mathbf{U}}}}, \ \forall i = 1,\ldots,n,$$
$$\mathbf{V}_{[i,j]} = \frac{\overline{\mathbf{V}}_{[i,j]}}{w_j^{\overline{\mathbf{V}}}}, \ \forall i = 1,\ldots,m.$$

Moreover, since, by definition, it holds that, $\forall j = 1,\ldots,r$,

$$\sigma_i \cdot \mathbf{u}_i \circ \mathbf{v}_i \equiv \sigma_i \cdot \left(\|\mathbf{u}_i\|_2 \cdot \frac{\mathbf{u}_i}{\|\mathbf{u}_i\|_2} \right) \circ \left(\|\mathbf{v}_i\|_2 \cdot \frac{\mathbf{v}_i}{\|\mathbf{v}_i\|_2} \right) \equiv (\sigma_i \cdot \|\mathbf{u}_i\|_2 \cdot \|\mathbf{v}_i\|_2) \cdot \frac{\mathbf{u}_i}{\|\mathbf{u}_i\|_2} \circ \frac{\mathbf{v}_i}{\|\mathbf{v}_i\|_2},$$

the normalizing factors in equations 4.3 and 4.4 can also be used to update the interval-valued singular values in the diagonal of $\ddot{\mathbf{\Sigma}}$, i.e., $\forall j = 1,\ldots,r$,

$$\ddot{\sigma}_j = w_j^{\overline{\mathbf{U}}} \cdot w_j^{\overline{\mathbf{V}}} \otimes \ddot{\sigma}_j = \begin{cases} w_j^{\overline{\mathbf{U}}} \cdot w_j^{\overline{\mathbf{V}}} \cdot \overline{\sigma}_j, & \text{if } \ddot{\sigma}_j \text{ was actually scalar}^1. \\ \left[w_j^{\overline{\mathbf{U}}} \cdot w_j^{\overline{\mathbf{V}}} \cdot \sigma_{*j}, w_j^{\overline{\mathbf{U}}} \cdot w_j^{\overline{\mathbf{V}}} \cdot \sigma_j^* \right], & \text{otherwise}^2. \end{cases}$$

4.3.3 Decomposition Target (c): scalar U, Σ and V

This last option is intended for those applications that require all matrices, $\mathbf{U}$, $\mathbf{\Sigma}$ and $\mathbf{V}$, to be scalar, for example in order to be compatible with algorithms and tools that assume scalar-valued factor matrices to support recommendations. The approach follows the same steps of option (b), finding a normalization of matrices

[1] Namely, if the interval had been replace by the average of its endpoints, if they were in the wrong ordering.

[2] Note that, since the norm of a vector is by definition always positive, the endpoints in the interval are scaled, but they keep the same ordering.

$\mathbf{U}$ and $\mathbf{V}$ by means of the column weights $w_j^{\overline{\mathbf{U}}}$ and $w_j^{\overline{\mathbf{V}}}$. This time, also the singular values in the diagonal of $\ddot{\mathbf{\Sigma}}$ are averaged, as well as rescaled, resulting in a scalar valued matrix $\mathbf{\Sigma} = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r)$, such that, $\forall j = 1, \ldots, r$,

$$\sigma_j = w_j^{\overline{\mathbf{U}}} \cdot w_j^{\overline{\mathbf{V}}} \cdot \frac{\sigma_{*j} + \sigma_j^*}{2}$$

4.4 Interval SVD (ISVD) implementation

In this section, a detailed description is presented of how the Singular Value Decomposition extension to interval-valued matrices can be implemented, illustrating a set of five approaches incrementally building on each other, starting from a very naive way of approximating the data as scalar (by simply averaging all the interval-valued entries) and then applying standard factorization algorithms, to increasingly more complex methods which rely on our proposed Interval Latent Semantic Alignment (ILSA, Section 4.2) to properly factorize the data while preserving the interval information. The set of factor matrices resulting from the various decomposition methods, each with its own set of properties, can then be recombined to reach a more or less faithful representation of the original data, yielding a level of accuracy that can then be evaluated, in order to validate and compare each approach. Figure 4.8 outlines the basic steps of the decomposition and reconstruction process for each strategy, whose details are described in the following.

4.4.1 ISVD_0, a naive approach: Average and Decompose

As we formalized in Section 4.1.1, an exact decomposition of an interval-valued matrix, $\ddot{\mathbf{M}} = [\mathbf{M}_*, \mathbf{M}^*]$, can never be reached, only an approximation. Thus, since some level of approximation is inevitable, a first naive approach that can be considered (and also serve as a reference) for solving the interval SVD problem is to simplify the input interval-valued matrix into a scalar matrix by replacing each interval entry with its average, namely, by defining a matrix $\mathbf{M}_{\text{avg}} \in \mathbb{R}^{n \times m}$ such that, $\forall i = 1, \ldots, n$ and $\forall j = 1, \ldots, m$,

$$\mathbf{M}_{\text{avg}[i,j]} = \frac{\mathbf{M}_{*[i,j]} + \mathbf{M}^*_{[i,j]}}{2}.$$

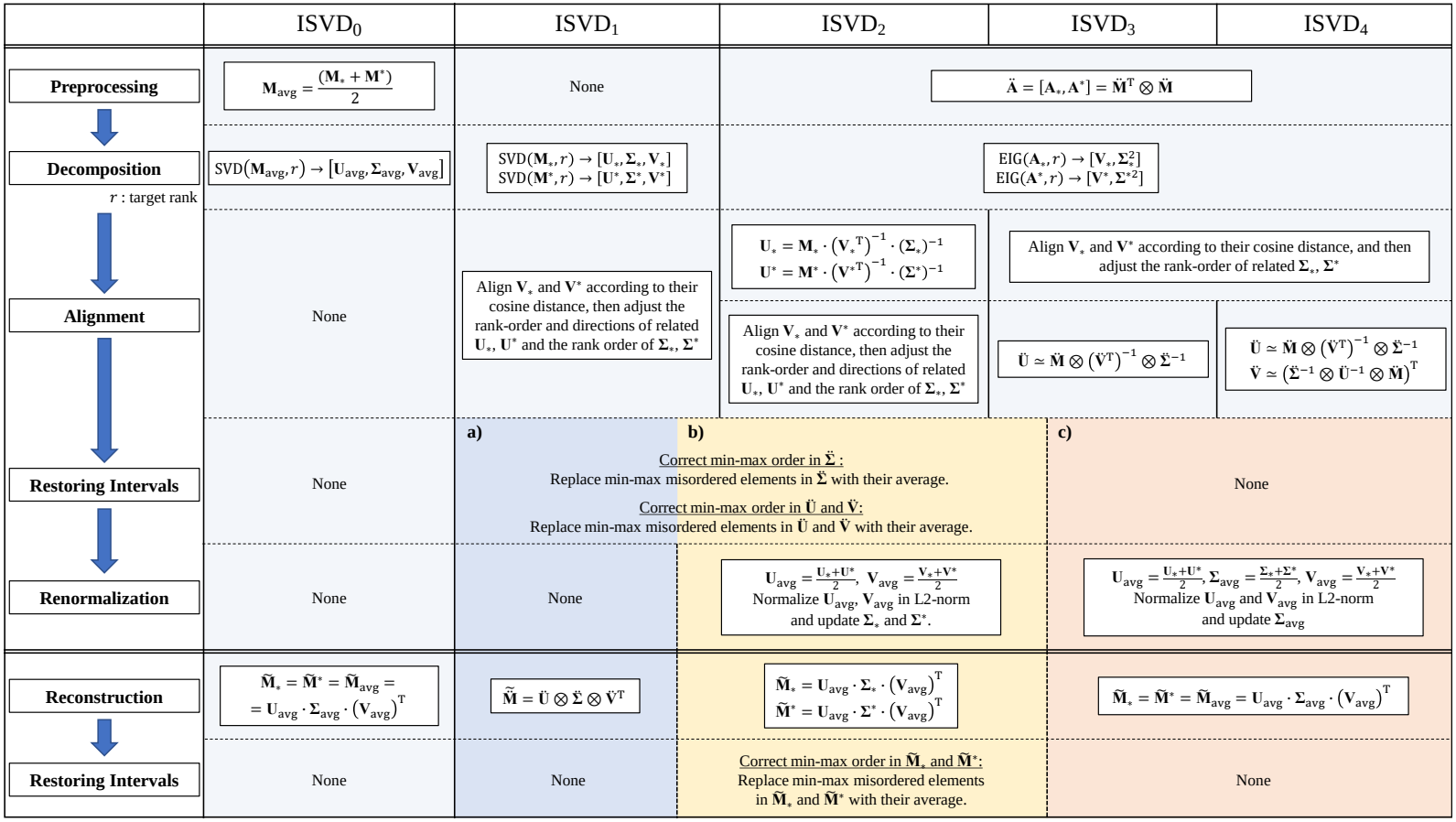


Figure 4.8: ISVD decomposition strategies for interval-valued input data

We can then approximate the interval singular value decomposition of $\ddot{\mathbf{M}}$ (with a given target rank r) by applying the standard SVD to the scalar matrix $\mathbf{M}_{\text{avg}}$, i.e.,

$$\text{ISVD}_0(\ddot{\mathbf{M}}, r) \simeq \text{SVD}(\mathbf{M}_{\text{avg}}, r).$$

This approach (outlined in the column labeled ISVD_0 in Figure 4.8) results in all scalar-valued factor matrices, $\mathbf{U}$, $\mathbf{V}$ (both orthonormal) and $\mathbf{\Sigma}$, so that it ends up being compatible only with the decomposition target-c application scenario, presented in Section 4.3.3.

Of course, an evident drawback of this naive strategy is that all the *interval information* in the original data gets lost in the process, but it may prove to be the only option, for example, in any given situation when the orthonormality of the latent components is paramount.

4.4.2 ISVD_1: Independently Decompose and Align

This second strategy (outlined in the column labeled ISVD_1 in Figure 4.8) formalizes the idea, introduced in Section 4.2, of approaching the decomposition of an interval-valued matrix $\ddot{\mathbf{M}} = [\mathbf{M}_*, \mathbf{M}^*]$ by independently applying the standard SVD (with the same target rank r) to the scalar matrices that identify its endpoints (i.e., $\mathbf{M}_*$ and $\mathbf{M}^*$), such that

$$\text{SVD}(\mathbf{M}_*, r) = \mathbf{U}_*\mathbf{\Sigma}_*\mathbf{V}_*^{\text{T}} \quad \text{and} \quad \text{SVD}(\mathbf{M}^*, r) = \mathbf{U}^*\mathbf{\Sigma}^*\mathbf{V}^{*\text{T}}.$$

Once we obtain the interval-valued factor matrices, we then apply the *Interval Latent Semantic Alignment* procedure (ILSA, Section 4.2) to match the latent components (i.e., the columns of $\mathbf{V}_*$ and $\mathbf{V}^*$) expressing the same semantics (pairing, where needed, the orientations of the singular vectors that are approximately collinear, but which point in opposite directions). In this way, we find a new *rank-order* for the columns of $\mathbf{V}_*$ and $\mathbf{V}^*$ that is then used to also rearrange the columns of $\mathbf{U}_*$ and $\mathbf{U}^*$, as well as the elements on the diagonals of $\mathbf{\Sigma}_*$ and $\mathbf{\Sigma}^*$.

Then, the interval-valued factor matrices can be formalized as $\ddot{\mathbf{U}} = [\mathbf{U}_*, \mathbf{U}^*]$, $\ddot{\mathbf{\Sigma}} = [\mathbf{\Sigma}_*, \mathbf{\Sigma}^*]$ and $\ddot{\mathbf{V}} = [\mathbf{V}_*, \mathbf{V}^*]$ and finally validated by replacing the endpoints of the entries that do not form valid intervals with their average (as also described in Section 4.2). This last step is actually related to the intended decomposition

target: option ISVD$_1$-a (Section 4.3.1), meant for applications that call for all three factor matrices to be interval-valued, would require a correction for all $\ddot{\mathbf{U}}$, $\ddot{\mathbf{\Sigma}}$ and $\ddot{\mathbf{V}}$; options ISVD$_1$-b (Section 4.3.2), which leads to scalar-valued factor matrices, $\mathbf{U}_{\text{avg}}$ and $\mathbf{V}_{\text{avg}}$, would require the correction to be applied only to $\ddot{\mathbf{\Sigma}}$, since the averaging to obtain $\mathbf{U}_{\text{avg}}$ and $\mathbf{V}_{\text{avg}}$ already makes up for the non-valid interval-valued entries in $\ddot{\mathbf{U}}$ and $\ddot{\mathbf{V}}$, respectively; finally, option ISVD$_1$-c (Section 4.3.3), which results in all scalar-valued factor matrices, $\mathbf{U}_{\text{avg}}$, $\mathbf{\Sigma}_{\text{avg}}$ and $\mathbf{V}_{\text{avg}}$, does not require any correction at all, since, again, the correction is implicit in the averaging process.

4.4.3 ISVD$_2$: Decompose, Solve, Align

The ISVD$_1$ algorithm described in the previous section first splits the interval-valued input matrix $\ddot{\mathbf{M}}$ into its (scalar-valued) minimum and maximum entries, $\mathbf{M}_*$ and $\mathbf{M}^*$, decomposes them independently by means of the standard SVD, and then applies the ILSA procedure to match the latent components, providing in output the interval-valued factor matrices $\ddot{\mathbf{U}}$, $\ddot{\mathbf{\Sigma}}$ and $\ddot{\mathbf{V}}$.

A way to improve this strategy (as outlined in the column labeled ISVD$_2$ in Figure 4.8) can be derived from the definition of the standard SVD reported in Section 3.1.2, in particular considering that, when $\mathbf{M}$ is a scalar-valued matrix and $\mathbf{M} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^{\mathrm{T}}$ is its singular value decomposition, it holds that:

- the columns of $\mathbf{U}$, also called the *left singular vectors* of matrix $\mathbf{M}$, are the eigenvectors of the matrix $\mathbf{M}\mathbf{M}^{\mathrm{T}} \in \mathbb{R}^{n \times n}$;
- the columns of $\mathbf{V}$, or the *right singular vectors* of $\mathbf{M}$, are the eigenvectors of the $\mathbf{M}^{\mathrm{T}}\mathbf{M} \in \mathbb{R}^{m \times m}$;
- the diagonal entries of $\mathbf{\Sigma}$, also known as the *singular values* of $\mathbf{M}$, are the square roots of the eigenvalues of both $\mathbf{M}\mathbf{M}^{\mathrm{T}}$ and $\mathbf{M}^{\mathrm{T}}\mathbf{M}$.

Thus, a more principled way to seek the SVD of an interval-valued matrix $\ddot{\mathbf{M}}$ would be to first compute an $n \times n$ interval-valued matrix, $\ddot{\mathbf{A}} = \ddot{\mathbf{M}}^{\mathrm{T}} \otimes \ddot{\mathbf{M}}$, through interval-valued matrix multiplication (discussed in Section 2.1) and then, by means of the eigendecomposition of this matrix, obtain the $\ddot{\mathbf{\Sigma}}^2$ and $\ddot{\mathbf{V}}$ matrices[2].

[2] Alternatively, we could get $\ddot{\mathbf{U}}$, instead of $\ddot{\mathbf{V}}$, by means of the eigendecomposition of $\ddot{\mathbf{M}} \otimes \ddot{\mathbf{M}}^{\mathrm{T}}$

Eigendecomposition of an interval-valued matrix, Ä

The problem of extending the eigendecomposition (defined in Section 3.1.1) to interval-valued matrices has been tackled by the linear algebra community under various approaches. As we reported in Section 3.2.1, the authors in [14, 44] tackle this problem by looking for interval bounds for the eigenvectors and the eigenvalues, by means of linear-programming based algorithms.

What we are interested in here is a more general solution, actually seeking *interval-valued eigenvectors* (and the corresponding *interval-valued eigenvalues*) that could form a consistent vector space where the latent information of the interval-valued data can be represented. Specifically, we look for a solution to the following eigenvalue equation:

$$\ddot{\mathbf{A}} \otimes \ddot{\mathbf{v}} = \ddot{\lambda} \otimes \ddot{\mathbf{v}}$$

Let us assume, for simplicity, that $\ddot{\mathbf{A}} = [\mathbf{A}_*, \mathbf{A}^*]$ is a 3×3 interval-valued square matrix

$$\ddot{\mathbf{A}} = \begin{bmatrix} [a_*, a^*] & [b_*, b^*] & [c_*, c^*] \\ [d_*, d^*] & [e_*, e^*] & [f_*, f^*] \\ [g_*, g^*] & [h_*, h^*] & [i_*, i^*] \end{bmatrix},$$

$\ddot{\lambda} = [\lambda_*, \lambda^*]$ be an interval-valued eigenvalue of $\mathbf{A}$ and

$$\ddot{\mathbf{v}} = \begin{bmatrix} [x_*, x^*] \\ [y_*, y^*] \\ [z_*, z^*] \end{bmatrix},$$

be the corresponding interval-valued eigenvector. The eigenvalue equation can then be rewritten as

$$\begin{bmatrix} [a_*, a^*] & [b_*, b^*] & [c_*, c^*] \\ [d_*, d^*] & [e_*, e^*] & [f_*, f^*] \\ [g_*, g^*] & [h_*, h^*] & [i_*, i^*] \end{bmatrix} \otimes \begin{bmatrix} [x_*, x^*] \\ [y_*, y^*] \\ [z_*, z^*] \end{bmatrix} = [\lambda_*, \lambda^*] \otimes \begin{bmatrix} [x_*, x^*] \\ [y_*, y^*] \\ [z_*, z^*] \end{bmatrix}.$$

Unfortunately, finding a solution for a linear system of equations like this is not at all straightforward, but we can at least try to find an approximation assuming,

for starters, that the eigenvector $\ddot{\mathbf{v}}$ is actually scalar, i.e.,

$$\mathbf{v} = \begin{bmatrix} x \\ y \\ z \end{bmatrix},$$

so that the eigenvalue equation can be formulated as

$$\begin{bmatrix} [a_*, a^*] & [b_*, b^*] & [c_*, c^*] \\ [d_*, d^*] & [e_*, e^*] & [f_*, f^*] \\ [g_*, g^*] & [h_*, h^*] & [i_*, i^*] \end{bmatrix} \otimes \begin{bmatrix} x \\ y \\ z \end{bmatrix} = [\lambda_*, \lambda^*] \otimes \begin{bmatrix} x \\ y \\ z \end{bmatrix}.$$

If we now assume that all the components of $\mathbf{v}$, as well as the endpoints of $\ddot{\lambda}$, are non-negative, by applying interval-valued algebra (in particular, the scalar-interval multiplication reported in Section 2.1.3) and simple matrix algebra, the above equality can be rewritten as

$$\begin{bmatrix} [a_*x + b_*y + c_*z,\ a^*x + b^*y + c^*z] \\ [d_*x + e_*y + f_*z,\ d^*x + e^*y + f^*z] \\ [g_*x + h_*y + i_*z,\ g^*x + h^*y + i^*z] \end{bmatrix} = \begin{bmatrix} [\lambda_* x,\ \lambda^* x] \\ [\lambda_* y,\ \lambda^* y] \\ [\lambda_* z,\ \lambda^* z] \end{bmatrix}$$

which can then be easily split into a minimum and maximum components:

$$\begin{bmatrix} a_*x + b_*y + c_*z \\ d_*x + e_*y + f_*z \\ g_*x + h_*y + i_*z \end{bmatrix} = \begin{bmatrix} \lambda_* x \\ \lambda_* y \\ \lambda_* z \end{bmatrix}$$

$$\begin{bmatrix} a^*x + b^*y + c^*z \\ d^*x + e^*y + f^*z \\ g^*x + h^*y + i^*z \end{bmatrix} = \begin{bmatrix} \lambda^* x \\ \lambda^* y \\ \lambda^* z \end{bmatrix}.$$

In other words, to obtain an approximation of the eigendecomposition of the original interval-valued matrix $\ddot{\mathbf{A}}$, we could seek the independent eigendecompositions of the matrices $\mathbf{A}_*$ and $\mathbf{A}^*$, as

$$\mathbf{A}_* \mathbf{v} = \lambda_* \mathbf{v}$$

and

$$\mathbf{A}^* \mathbf{v} = \lambda^* \mathbf{v}$$

where

$$\mathbf{A}_* = \begin{bmatrix} a_* & b_* & c_* \\ d_* & e_* & f_* \\ g_* & h_* & i_* \end{bmatrix} \text{ and } \mathbf{A}^* = \begin{bmatrix} a^* & b^* & c^* \\ d^* & e^* & f^* \\ g^* & h^* & i^* \end{bmatrix}.$$

The problem is that, in general, it may be hard to find two eigenvalues of $\mathbf{A}_*$ and $\mathbf{A}^*$ sharing the same eigenvector. Reasonably, the independent decompositions would also imply separate eigenvectors, as in

$$\mathbf{A}_*\mathbf{v}_1 = \lambda_1\mathbf{v}_1$$

and

$$\mathbf{A}^*\mathbf{v}_2 = \lambda_2\mathbf{v}_2,$$

hopefully having $\mathbf{v}_1$ and $\mathbf{v}_2$ as close as possible. We could then define an interval-valued eigenvector, $\ddot{\mathbf{v}} = [\mathbf{v}_*, \mathbf{v}^*]$, such that, for every i,

$$\ddot{\mathbf{v}}_i = \begin{cases} [\mathbf{v}_{1i}, \mathbf{v}_{2i}], & \text{if } \mathbf{v}_{1i} \leq \mathbf{v}_{2i}. \\ \frac{\mathbf{v}_{1i}+\mathbf{v}_{2i}}{2} & \text{otherwise.} \end{cases}$$

and an interval-valued eigenvalue, $\ddot{\lambda} = [\lambda_*, \lambda^*]$, such that, for every i,

$$\ddot{\lambda}_i = \begin{cases} [\lambda_{1i}, \lambda_{2i}], & \text{if } \lambda_{1i} \leq \lambda_{2i}. \\ \frac{\lambda_{1i}+\lambda_{2i}}{2} & \text{otherwise.} \end{cases}$$

As we can see, the eigendecomposition of an interval-valued matrix presents the same problems found for the ISVD (see Section 4.1.1), namely that the eigenvector, as well as the singular vectors, may not be aligned, and the interval-valued entries of the matrices involved may not be valid. Luckily, these problems can be addressed with the same approach used for the ISVD, namely, the Interval Latent Semantic Alignment (ILSA), presented in Section 4.2.

More in detail, going back to our case, we can initially seek the eigendecomposition of the interval-valued (square) matrix $\ddot{\mathbf{A}} = \ddot{\mathbf{M}}^{\mathrm{T}} \otimes \ddot{\mathbf{M}}$ by independently decomposing the endpoints matrices $\mathbf{A}_*$ and $\mathbf{A}^*$, namely, given a target rank $r \leq n$,

$$\text{EIG}(\mathbf{A}_*, r) \rightarrow [\mathbf{V}_*, \mathbf{\Lambda}_*], \text{ and } \text{EIG}(\mathbf{A}^*, r) \rightarrow [\mathbf{V}^*, \mathbf{\Lambda}^*], \tag{4.5}$$

where $\mathbf{V}_*$ and $\mathbf{V}^*$ are the singular values of $\mathbf{M}_*$ and $\mathbf{M}^*$, respectively, and $\mathbf{\Sigma}_* = \sqrt{\mathbf{\Lambda}_*}$ and $\mathbf{\Sigma}^* = \sqrt{\mathbf{\Lambda}^*}$ are the relative singular values.

Given these matrices, we can then obtain the left singular vectors by solving the SVD equations $\mathbf{M}_* = \mathbf{U}_*\mathbf{\Sigma}_*\mathbf{V}_*^{\mathrm{T}}$ and $\mathbf{M}^* = \mathbf{U}^*\mathbf{\Sigma}^*\mathbf{V}^{*\mathrm{T}}$, by, respectively, $\mathbf{U}_*$ and $\mathbf{U}^*$, i.e.,

$$\mathbf{U}_* = \mathbf{M}_* \cdot \left(\mathbf{V}_*^{\mathrm{T}}\right)^{-1} \cdot \left(\mathbf{\Sigma}_*\right)^{-1} \quad \text{and} \quad \mathbf{U}_* = \mathbf{M}^* \cdot \left(\mathbf{V}^{*\mathrm{T}}\right)^{-1} \cdot \left(\mathbf{\Sigma}^*\right)^{-1}.$$

Once all the factor matrices have been evaluated, we can then proceed with the alignment of the latent components by means of the ILSA procedure and with the correction of non-valid intervals in their entries, thus providing consistent interval-valued factor matrices $\ddot{\mathbf{U}}$, $\ddot{\mathbf{\Sigma}}$ and $\ddot{\mathbf{V}}$. Once again, according to the decomposition target option (either a, b, or c) the factor matrices can be averaged or left as interval-valued, producing the final output of the procedure.

4.4.4 ISVD$_3$: Decompose, Align, Solve

This strategy (outlined in the column labeled ISVD$_3$ in Figure 4.8) follows the same approach as the previous one, up to the factorization of the interval-valued matrix $\ddot{\mathbf{A}} = \ddot{\mathbf{M}}^{\mathrm{T}} \otimes \ddot{\mathbf{M}}$ through the independent eigendecomposition of its endpoints matrices $\mathbf{A}_*$ and $\mathbf{A}^*$ (equation 4.5). In contrast with ISVD$_2$, here we promptly address the alignment issue in the resulting factor matrices by means of the ILSA procedure[3], in order to produce interval-valued matrices, $\ddot{\mathbf{\Sigma}}$ and $\ddot{\mathbf{V}}$, which are already semantically consistent. Only at this point the left singular values are computed, solving the ISVD equation $\ddot{\mathbf{M}} \simeq \ddot{\mathbf{U}} \otimes \ddot{\mathbf{\Sigma}} \otimes \ddot{\mathbf{V}}^{\mathrm{T}}$ by $\ddot{\mathbf{U}}$, i.e.,

$$\ddot{\mathbf{U}} \simeq \ddot{\mathbf{M}} \otimes \left(\ddot{\mathbf{V}}^{\mathrm{T}}\right)^{-1} \otimes \ddot{\mathbf{\Sigma}}^{-1}.$$

However, this last step requires the computation of the inverse of two interval-valued matrices, a not so straightforward procedure, that needs to be discussed.

[3]Excluding the correction of the non-valid interval entries in $\ddot{\mathbf{\Sigma}}$ and $\ddot{\mathbf{V}}$, which can be postponed as the last step of the entire procedure, when all the interval-valued matrices need to be validated in order to provide a consistent output, both semantically and in terms of interval-valued entries.

Inverse of a non-negative and diagonal interval-valued matrix, $\ddot{\mathbf{\Sigma}}$

The interval-valued factor matrix $\ddot{\mathbf{\Sigma}} = [\mathbf{\Sigma}_*, \mathbf{\Sigma}^*]$, with $\mathbf{\Sigma}_*, \mathbf{\Sigma}^* \in \mathbb{R}^{r \times r}$ is a diagonal matrix such that, $\forall i = 1, \ldots, r$,

$$\ddot{\mathbf{\Sigma}}_{[i,i]} = \ddot{\sigma}_i = [\sigma_{*i}, \sigma_i^*], \text{ with } \sigma_{*i} \leq \sigma_i^*$$

By definition, an $r \times r$ diagonal matrix $\mathbf{D} = \text{diag}(d_1, d_2, \ldots, d_r)$, is invertible if all the elements on its diagonal are non-zero, i.e., if $\forall i = 1, \ldots, r$, $d_i \neq 0$, and its inverse is defined as the also diagonal matrix $\mathbf{D}^{-1} = \text{diag}(\frac{1}{d_1}, \frac{1}{d_2}, \ldots, \frac{1}{d_r})$, such that the elements in the diagonal are the reciprocal of the entries in $\mathbf{D}$.

Known this, and given the definition of the division of two intervals (see Section 2.1.3), we could assume that the inverse of an interval-valued matrix $\ddot{\mathbf{\Sigma}}$ could be defined as the (diagonal) interval-valued matrix $\ddot{\mathbf{\Sigma}}^{-1}$ such that, $\forall i = 1, \ldots, r$,

$$\ddot{\mathbf{\Sigma}}^{-1}_{[i,i]} = \ddot{\tau}_i = \frac{1}{[\sigma_{*i}, \sigma_i^*]} = 1 \cdot \left[\frac{1}{\sigma_i^*}, \frac{1}{\sigma_{*i}}\right] = \left[\frac{1}{\sigma_i^*}, \frac{1}{\sigma_{*i}}\right].$$

The problem with this definition, however, is that (as we already saw in Section 4.1.1) the product of two intervals is itself an interval (see Theorem 2 in [25]), hence, an important property of the inversion, i.e., that $\mathbf{D} \cdot \mathbf{D}^{-1} = \mathbf{I}$, cannot be extended to interval-valued diagonal matrices.

What can be done is to find an approximation of the inversion process such that the product with the original matrix is as close as possible to the identity, namely, find $\ddot{\mathbf{\Sigma}}^{-1} = \text{diag}(\ddot{\tau}_1, \ddot{\tau}_2, \ldots, \ddot{\tau}_r)$ such that

$$\ddot{\mathbf{\Sigma}} \otimes \ddot{\mathbf{\Sigma}}^{-1} = \widetilde{\ddot{\mathbf{I}}}_r \,,$$

where $\widetilde{\ddot{\mathbf{I}}}_r$ is an interval-valued diagonal matrix (approximately equal to the identity matrix $\mathbf{I}_r$) such that, $\forall i = 1, \ldots, r$,

$$\widetilde{\ddot{\mathbf{I}}}_{r[i,i]} = [1 - \varepsilon_i, 1 + \varepsilon_i], \text{ with } 0 \leq \varepsilon_i \leq 1.$$

Our problem can be formulated as finding, $\forall i = 1, \ldots, r$, an interval $\ddot{\tau}_i = [\tau_{*i}, \tau_i^*]$ such that ε_i is minimum, and is subject to the following constraints:

1. $\sigma_{*i} \cdot \tau_{*i} = 1 - \varepsilon_i$;
2. $\sigma_i^* \cdot \tau_i^* = 1 + \varepsilon_i$;

3. $\tau_{*i} \leq \tau_i^*$,
4. $0 \leq \varepsilon_i \leq 1$.

By rewriting constraints (1) and (2) as $\tau_{*i} = \frac{1-\varepsilon_i}{\sigma_{*i}}$ and $\tau_i^* = \frac{1+\varepsilon_i}{\sigma_i^*}$, and by combining them under constraint (3), we obtain the following equality:

$$\frac{1-\varepsilon_i}{\sigma_{*i}} \leq \frac{1+\varepsilon_i}{\sigma_i^*};$$

$$\frac{(1-\varepsilon_i)\sigma_i^* - (1+\varepsilon_i)\sigma_{*i}}{\sigma_{*i} \cdot \sigma_i^*} \leq 0;$$

$$\frac{\sigma_i^* - \sigma_{*i}}{\sigma_{*i} \cdot \sigma_i^*} \leq \varepsilon \cdot \frac{\sigma_i^* + \sigma_{*i}}{\sigma_{*i} \cdot \sigma_i^*};$$

$$\frac{\sigma_i^* - \sigma_{*i}}{\sigma_i^* + \sigma_{*i}} \leq \varepsilon_i;$$

which, combined with constraint (4), and considering that, by definition, $\sigma_i^* \geq \sigma_{*i}$, can be rewritten as

$$0 \leq \frac{\sigma_i^* - \sigma_{*i}}{\sigma_i^* + \sigma_{*i}} \leq \varepsilon_i \leq 1.$$

From this inequality, we can conclude, first of all, that ε_i would be 0 only if $\sigma_{*i} = \sigma_i^*$, namely if and only if $\ddot{\mathbf{\Sigma}}$ would be scalar which, unfortunately, is not our case. And we can also conclude that ε_i is minimum when it is equal to $\frac{\sigma_i^* - \sigma_{*i}}{\sigma_i^* + \sigma_{*i}}$, i.e., the solution of the equation

$$\frac{\sigma_i^* - \sigma_{*i}}{\sigma_i^* + \sigma_{*i}} = \varepsilon_i, \tag{4.6}$$

which holds, going back to constraints (1) and (2), if and only if $\tau_{*i} = \tau_i^*$. This means that, for better approximating the identity matrix, we need to settle for an inverse of $\ddot{\mathbf{\Sigma}}$ that needs to be scalar.

In particular, considering $\tau_{*i} = \tau_i^* = \tau_i$, and rewriting constraints (1) and (2) as $\sigma_{*i} = \frac{1-\varepsilon_i}{\tau_i}$ and $\sigma_i^* = \frac{1+\varepsilon_i}{\tau_i}$, we can rewrite equation 4.6 as

$$\frac{\frac{1+\varepsilon_i}{\tau_i} - \frac{1-\varepsilon_i}{\tau_i}}{\sigma_i^* + \sigma_{*i}} = \varepsilon_i,$$

and solve by τ_i:

$$\frac{1+\varepsilon_i - 1 - \varepsilon_i}{\tau_i(\sigma_i^* + \sigma_{*i})} = \varepsilon_i;$$

$$\tau_i = \frac{2}{\sigma_i^* + \sigma_{*i}},$$

so that we can finally define the inverse of $\ddot{\mathbf{\Sigma}}$ as the scalar diagonal matrix $\ddot{\mathbf{\Sigma}}^{-1} = \text{diag}(\tau_1, \tau_2, \ldots, \tau_r)$ such that, $\forall i = 1, \ldots, r$,

$$\ddot{\mathbf{\Sigma}}^{-1}_{[i,i]} = \tau_i = \frac{2}{\sigma_i^* + \sigma_{*i}}.$$

Inverse of an interval-valued matrix, $\ddot{\mathbf{V}}$

The inversion process detailed above is applicable to only square, diagonal interval-valued matrices, but, unfortunately, the factor matrix $\ddot{\mathbf{V}}$ is not diagonal. Therefore, since there is no straightforward way to invert a generic interval-valued matrix, we need to resort to an approximation that leverages the known inversion of a scalar-valued matrix. More in detail, given our interval valued matrix $\ddot{\mathbf{V}} = [\mathbf{V}_*, \mathbf{V}^*]$, where $\mathbf{V}_*, \mathbf{V}^* \in \mathbb{R}^{m \times r}$, we take the following steps:

- we compute an approximation of $\ddot{\mathbf{V}}$ by replacing its interval-valued entries with the average of their endpoints, i.e., we compute $\mathbf{V}_{\text{avg}}$ such that, $\forall i = 1, \ldots, m$ and $\forall j = 1, \ldots, r$,

$$\mathbf{V}_{\text{avg}[i,j]} = \frac{\mathbf{V}_{*[i,j]} + \mathbf{V}^*_{[i,j]}}{2};$$

- we invert $\mathbf{V}_{\text{avg}}$ by means of the know inversion algorithms for scalar-valued matrices;
- we return an approximation of $\ddot{\mathbf{V}}^{-1}$ as $\mathbf{V}^{-1}_{\text{avg}}$.

It is to be noticed, that this approach may present some difficulties, namely that the matrix $\mathbf{V}_{\text{avg}}$, though scalar-valued, may not always be invertible. We, therefore, first need to check whether $\mathbf{V}_{\text{avg}}$ is *well-conditioned* (i.e., if its *condition number*[4] is over a certain threshold) and, if that is not the case, use the Moore-Penrose pseudo-inverse [41] to compute an approximation of $\mathbf{V}^{-1}_{\text{avg}}$. If the condition number were to

[4]The condition number of a matrix can be leveraged to infer how accurately such matrix can be inverted: if the condition number is not too much larger than 1, then the matrix is said to be *well-conditioned*, meaning that its inverse can be computed with good accuracy. If the condition number however gets very large (namely, the matrix is *ill-conditioned*), the computation of its inverse could be prone to large numerical errors.

be too high, the best choice could be to revert back to the ISVD_2 strategy.

Now that all the factor matrices have been evaluated, we can proceed with the correction of non-valid intervals in their entries, thus providing consistent interval-valued factor matrices $\ddot{\mathbf{U}}$, $\ddot{\mathbf{\Sigma}}$ and $\ddot{\mathbf{V}}$. As for the previous strategies, according to the decomposition target option (either a, b, or c) the factor matrices can be averaged or left as interval, producing the final output of the procedure.

4.4.5 ISVD_4: Decompose, Align, Solve, Recompute

In this section, we consider a variant of the ISVD_3 strategy, which not only inherits its benefits, but also further reduces the degree of imprecision in the resulting interval-valued factor matrices, $\ddot{\mathbf{U}}$ and $\ddot{\mathbf{V}}$, as the experimental validation (Chapter 5) will confirm.

More in detail, the ISVD_4 approach follows the same steps of ISVD_3 up to the computation of the left singular vectors matrix, obtained solving the ISVD equation $\ddot{\mathbf{M}} \simeq \ddot{\mathbf{U}} \otimes \ddot{\mathbf{\Sigma}} \otimes \ddot{\mathbf{V}}^{\mathrm{T}}$ by $\ddot{\mathbf{U}}$, i.e.,

$$\ddot{\mathbf{U}} \simeq \ddot{\mathbf{M}} \otimes \left(\ddot{\mathbf{V}}^{\mathrm{T}}\right)^{-1} \otimes \ddot{\mathbf{\Sigma}}^{-1},$$

where the right interval-valued singular vectors (i.e., the columns of $\ddot{\mathbf{V}}$), obtained through the independent eigendecomposition of $\mathbf{A}_*$ and $\mathbf{A}_*$, had already been aligned, by means of the ILSA procedure.

At this point, an extra step is introduced, namely the recomputation of these same right singular vectors, $\ddot{\mathbf{V}}$, by solving, again, the ISVD equation $\ddot{\mathbf{M}} \simeq \ddot{\mathbf{U}} \otimes \ddot{\mathbf{\Sigma}} \otimes \ddot{\mathbf{V}}^{\mathrm{T}}$, this time by $\ddot{\mathbf{V}}$:

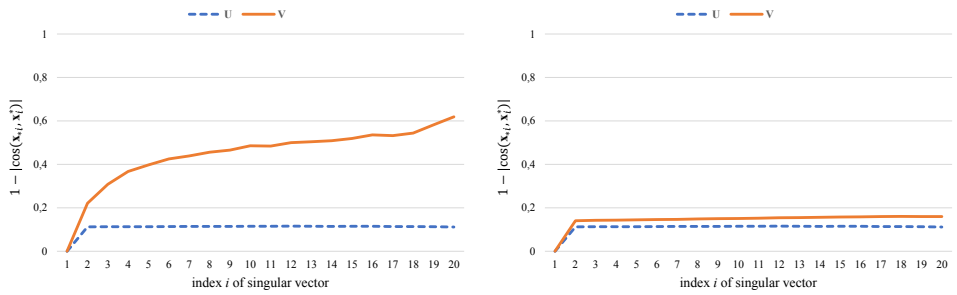
$$\ddot{\mathbf{V}} \simeq \left(\ddot{\mathbf{\Sigma}}^{-1} \otimes \ddot{\mathbf{U}}^{-1} \otimes \ddot{\mathbf{M}}\right)^{\mathrm{T}}$$

The basic idea behind this strategy is related to what we have discussed in Section 4.1.1 regarding the inherent imprecision in the factorization of an interval-valued matrix. In particular, we came to the conclusion that interval-valued factor matrices, $\ddot{\mathbf{U}}$ and $\ddot{\mathbf{V}}$, can never be orthonormal, hence making it difficult to interpret their columns as the basis of a transformed vector space. We then saw that the alignment of the latent components according to their semantics (i.e., the ILSA procedure, described in Section 4.2) mitigates this problem by matching the singular vectors which are closer, reducing the overall (cosine) distance between each

pair $\mathbf{v}_*$ and $\mathbf{v}^*$ (see Figure 4.7), approaching a situation where the interval-valued components are scalar, and approximate more easily a vector basis (thus proving that the similarity between interval-valued factor matrices is crucial for capturing the latent semantics in interval-valued data).

Here we try to take this idea a bit further, attempting to bring the minimum and maximum components of the interval-valued singular vectors in $\ddot{\mathbf{V}}$ even closer, on the one hand loosing some accuracy in terms of the factorized data (i.e., how well the singular vectors would work in an independent reconstruction of the minimum and maximum values), but on the other getting a more truthful representation of the data in the transformed vector space, hopefully reaching a favorable compromise that would increase the accuracy of the overall interval decomposition process.

Figure 4.9 helps visualize this approach, showing how the matrix recomputation affects the alignment of the minimum and maximum components of the interval-valued singular vectors, both for $\ddot{\mathbf{U}}$ and $\ddot{\mathbf{V}}$. In the figure, we plot the absolute cosine distance between each corresponding pair of column vectors $\mathbf{x}_{*i}$ and $\mathbf{x}_i^*$ (where $\mathbf{x}_{*i}$ can either be $\mathbf{U}_{*[:,i]}$ or $\mathbf{V}_{*[:,i]}$, and similarly for $\mathbf{x}_i^*$), considered in increasing order of singular values [5]. Specifically, Figures 4.9(a) and 4.9(b) report the cosine distance values *before recomputation* and *after recomputation* of $\ddot{\mathbf{V}}$, basically illustrating the differences between the strategies ISVD_3 and ISVD_4.



(a) Before recomputation of $\ddot{\mathbf{V}}$ (b) After recomputation of $\ddot{\mathbf{V}}$

Figure 4.9: Effects of the recomputation of the right singular vectors $\ddot{\mathbf{V}}$

First of all, what we can already see in Figure 4.9(a) is that the distances be-

[5] Note that, the lower the cosine distance, the more similar the minimum and maximum components of the singular vectors

tween the corresponding vectors in $\mathbf{U}_*$ and $\mathbf{U}^*$, obtained in ISVD_3 through the equation

$$\ddot{\mathbf{U}} \simeq \ddot{\mathbf{M}} \otimes \left(\ddot{\mathbf{V}}^{\mathrm{T}}\right)^{-1} \otimes \ddot{\mathbf{\Sigma}}^{-1},$$

are already pretty low, especially if compared with the vectors in $\ddot{\mathbf{V}}$, suggesting that the process of deriving $\ddot{\mathbf{U}}$ through the inversion of $\ddot{\mathbf{V}}$ and $\ddot{\mathbf{\Sigma}}$ (which, as we saw in Section 4.4.4, requires some averaging) *smoothens* the differences between each pair $\mathbf{u}_{i*}$ and $\mathbf{u}_i^*$, bringing them closer.

Figure 4.9(b) shows how the same is true for $\mathbf{V}_*$ and $\mathbf{V}^*$: as we can see, the cosine distance between each pair of column vectors $\mathbf{v}_{i*}$ and $\mathbf{v}_i^*$ gets noticeably smaller after the re-computation and, as the experiments will confirm (Chapter 5), they are in this way able to more precisely capture (w.r.t. ISVD_3) the significant latent semantics in the interval-valued data, thus leading to an overall improvement in terms of the accuracy of the interval decomposition process.

4.4.6 ISVD complexity analysis

A complexity analysis for the ISVD algorithms presented in the previous sections can be performed by looking at the local complexity of the primitive procedures that form each step in the overall decomposition processes. Specifically, we can draw general conclusions by inspecting in detail the ISVD_4 implementation, given that it is the most demanding in terms of resources and, as the experimental validation presented in Chapter 5 will confirm, is in general the most robust option.

Preprocessing. Following the chart in Figure 4.8, we can see that first step in the ISVD_4 procedure (i.e., the *preprocessing* of the input data) has the purpose of computing the interval-valued symmetric square matrix $\ddot{\mathbf{A}}$, whose factorization will provide the latent components ot the input matrix $\ddot{\mathbf{M}}$. The implementation of this step relies on the interval-valued matrix product, $\otimes$, an operation that, like its scalar counterpart, has time complexity $\mathcal{O}(n^3)$, supposing a general matrix $\ddot{\mathbf{M}}$ of size $n \times n$. To reduce the computational impact of this step, we have implemented (see Appendix A.1.1) an approximated algorithm that leverages the efficiency of matrix operations in MATLAB by avoiding for-loops, trading a bit of accuracy[6] for

[6]It is important to notice that if both the factor matrices are non-negative, which is the case for most applications, their product by means of our algorithm will be exact.

a huge advantage in terms of execution times.

In terms of space complexity, the number of elements of matrix $\ddot{\mathbf{A}}$, of size $m \times m$, can represent an issue in case the input matrix $\ddot{\mathbf{A}}$, of size $n \times m$, is such that $m \ll n$. In these circumstances, if the latent semantics in the data allows it, it would be preferable to evaluate $\ddot{\mathbf{A}} = \ddot{\mathbf{M}} \otimes \ddot{\mathbf{M}}^{\mathrm{T}}$ and operate the latent semantic alignment on the left singular vectors, $\ddot{\mathbf{U}}$, instead that on the right ones, $\ddot{\mathbf{V}}$.

Decomposition. The factorization of the interval-valued matrix $\ddot{\mathbf{A}}$ is done by independently applying the eigendecomposition to the scalar endpoints matrices $\mathbf{A}_*$ and $\mathbf{A}^*$, both of size $m \times m$. The eigendecomposition procedure is implemented in MATLAB by means of the QR Algorithm [40] which, for an $m \times m$ real and symmetric matrix has complexity $\mathcal{O}(m^3)$.

Alignment. The latent semantic alignment step, as detailed in Section 4.2 (see Problem 1), requires the solution of a *linear assignment* problem, in order to find the optimal matching between the r columns of $\mathbf{V}_*$ and $\mathbf{V}^*$. An optimal solution to this problem can be found, by means of known algorithms such as the Jonker-Volgenant [28], in time $\mathcal{O}(r^3)$. However, in order to reduce the computation time to $\mathcal{O}(r^2)$, it is possible to consider the alignment task as a *stable marriage* (or *stable matching*) problem [26], trading off the optimality of the solution for the speed of execution. More specifically, a matching is considered *stable* when there does not exist any non-matched vectors $\mathbf{v}_{i*}$, $\mathbf{v}_j^*$ which would both be closer to each other than to the vectors they are paired under the current assignment. Such a solution, however, is not optimal, since it does not guarantee that the overall distances between the paired vectors are minimized.

Since the remaining steps in the procedure (*restoring intervals* and *renormalization*) are just $\mathcal{O}(n^2)$, we can conclude, by looking at the worst-case step in the chain, that the overall complexity of ISVD_4 (and, by extension, of all the ISVD approaches) is $\mathcal{O}(n^3)$.

4.5 Aligned Interval-valued PMF (AI-PMF)

In the previous sections of this chapter, we focused our attention on how to extend the Singular Value Decomposition technique to interval-valued data. Here we show that applying our proposed Interval-Valued Latent Semantic Alignment

(ILSA) method for processing interval-valued basis vectors can also be effective in alternative factorization algorithms, such as Probabilistic Matrix Factorization (PMF), introduced in Section 3.1.4.

As we reported in Section 3.2, various efforts to adapt PMF models to interval-valued data has been investigated in the literature. Our approach is built on the work proposed in [46], reported in Section 3.2.2, where the authors present an Interval-valued PMF technique (referred to as I-PMF) with the purpose of improving the prediction of the missing entries of a user-item rating matrix.

More specifically, although I-PMF adopts extra constraints in the loss function to capture the interval nature of the data, in our opinion it fails to consider the need for alignment in the interval-valued latent semantics spaces, as discussed in Section 4.2.

Hence, we propose a *semantically aligned* interval-valued PMF formulation (AI-PMF for short), which adjusts the rank-order of the eigen-vectors in $\mathbf{V}_*$ and $\mathbf{V}^*$ in each gradient descent iteration, after obtaining $\mathbf{U}$, $\mathbf{V}_*$ and $\mathbf{V}^*$ from the above equations:

$$\ddot{\mathbf{V}}_{\text{aligned}} = [\mathbf{V}_{*\text{aligned}}, \mathbf{V}^*_{\text{aligned}}] = \text{ILSA}(\mathbf{V}_*, \mathbf{V}^*).$$

As we will see in the next chapter, the latent semantic alignment procedure yields significant improvements in the decomposition accuracy of interval-valued (probabilistic) data.

Chapter 5

Experiments on Interval-valued matrix factorization

The purpose of this chapter is to provide a thorough description of the set of experiments that we devised to validate our approach on the factorization of interval-valued data presented in Chapter 4, while also comparing our decomposition strategies with state of the art algorithms on different fields, including *face image analysis* and *collaborative filtering*. Specifically, various scenarios have been considered in choosing a sample base for our experiments, ranging from synthetic data, specifically designed to test various kinds of interval distributions in a controlled manner, to real-world datasets, in order to prove the actual effectiveness of our decomposition strategies.

Let's start with a description of the datasets that we relied on for the experiments.

5.1 Choosing the right datasets

As stated above, we relied upon both synthetic and readily available real datasets to evaluate the proposed interval-valued matrix decomposition techniques in the context of several data reconstruction and classification scenarios.

5.1.1 Synthetic datasets

The purpose of generating a synthetic set of interval-valued matrices was aimed at providing an easy to control scenario, were various parameters could be purposefully set in order to evaluate our approach over a range of test cases as diverse as possible. More in detail, we relied on the following set of parameters (summarized in Table 5.1):

Matrix size: we considered various possible sizes for the test matrices, to see how it would affect the decomposition performances; this include having matrices with more rows than column, or vice-versa, while keeping the same number of entries overall (10 thousand for the first three sets), and also matrices with more elements (100 thousand for the last two sets), to see how this would affect the time of execution.

Matrix density: this parameter indicates the percentage of zero-valued entries (randomly placed) in a matrix, and is meant to test how the *sparsity* of a matrix affects its decomposition process. It is quite common, in fact, to have quite sparse matrices in various domain, for example in the case of user-item rating matrices in collaborative filtering applications.

Interval density: this parameter determines the ratio between scalar and interval-valued entries of a matrix: 5% density indicates that only 5 (non-zero) entries out of 100 are interval-valued, while 100% indicates a fully interval-valued matrix. In this way we can test precisely how the interval-valued nature of a matrix affects its decomposition.

Interval intensity: another aspect to take into account when generating an interval-valued matrix is how wide the range of each entry should be. More in detail, 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.

Target rank: this parameter is related to the actual 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, low if we want to extract just a few latent semantics components).

We then consider two types of interval-valued data, *uniform* and *anonymized*:

Parameter	Values
Matrix size	**40** $\times$ **250**, 250 $\times$ 40, 25 $\times$ 400, 400 $\times$ 250, 250 $\times$ 400
Matrix density	percentage of 0-values: **0**%, 50%, 90%
Interval density	5%, 25%, 50%, 75%, 90%, 95%, 99%, **100**%
Interval intensity	10%, 25%, 50%, 75%, **100**%
Target rank	Uniform data: 5, 10, **20**, 40, 100, full
	Anonymized and social media data: 5%; 50%; 100%

Table 5.1: Experiment parameters for synthetic data (values in bold are defaults)

1. Uniform matrices do not have inherent structures – the entries are generated according to the parameters described above. More in detail, once a set of parameters has been selected, the process of generating a synthetic interval-valued matrix $\ddot{\mathbf{M}}$ starts by randomly creating a scalar matrix $\mathbf{M}$ with values in a domain that goes from 1 to 100. Then, according to the chosen *matrix density*, a percentage of the entries is set to zero. At this point, some of the non-zero scalar entries remaining (how many depends on the *interval density*) are replaced by an interval, whose range is evaluated from the initial scalar value and the selected *interval intensity*. For example, given a scalar value $\mathbf{M}_{[i,j]}$ and an interval intensity value of X%, we generate the interval-valued entry $\ddot{\mathbf{M}}_{[i,j]}$ as

$$\ddot{\mathbf{M}}_{[i,j]} = \left[\mathbf{M}_{[i,j]}, \mathbf{M}_{[i,j]} + \mathbf{M}_{[i,j]} \cdot x \sim \mathcal{U}\left(0, \frac{X}{100}\right)\right],$$

where x is extracted form a uniform distribution, $\mathcal{U}$, between 0 and the interval intensity value X.

2. Anonymized matrices are obtained by means of a *generalization* process applied to random scalar matrices in order to get to a specific degree of anonymization; in particular, we start as before by creating a random scalar matrix $\mathbf{M}$ with values in a domain that goes from 1 to 100, then we consider four *generalization levels* (in increasing order of anonymization[1]), each characterizing a subset of the

[1]The higher the generalization level, the larger the corresponding intervals replacing each scalar entry and the more anonymized the data.

Anonymization	$\mathbf{L}_1$	$\mathbf{L}_2$	$\mathbf{L}_3$	$\mathbf{L}_4$
Low	40%	30%	20%	10%
Medium	25%	25%	25%	25%
High	10%	20%	30%	40%

Table 5.2: Percentages of elements in each generalization level for three degree of anonymization

entries of $\mathbf{M}$:

- L_1: the entries of $\mathbf{M}$ that will end up in this subset are simply rounded to the nearest integer, down to generate the minimum endpoint, up for the maximum, so that a given scalar entry $\mathbf{M}_{[i,j]}$ is generalized as $\ddot{\mathbf{M}}_{[i,j]} = [\lfloor \mathbf{M}_{[i,j]} \rfloor, \lceil \mathbf{M}_{[i,j]} \rceil]$;
- L_2: for this level of generalization, the entire range of values in the domain (1 to 100) is divided into a series of 50 bins, and each entry of $\mathbf{M}_{[i,j]}$ is replaced by the endpoints of the bin in which it falls into, so, for example, a scalar entry $\mathbf{M}_{[i,j]} = 2.35$, that ends up in the bin $b_k = [2,4]$ would be generalized as the interval-valued entry $\ddot{\mathbf{M}}_{[i,j]} = [2,4]$;
- L_3: the generalization process is the same as with L_2, only here the number of bins to divide the entries in is 20 (thus increasing the level of generalization);
- L_4: finally, with 5 bins, the highest degree of anonymization is reached.

Given these, we then define three classes of anonymization (*low*, *medium* and *high*) by assigning different percentages of elements to each generalization level, (as summarized in Table 5.2): the rationale is to assign more and more elements to higher levels of generalization in order to reach an overall higher degree of anonymization in the matrix.

For each scenario, we created 100 random matrices and the results presented are averages of the corresponding runs.

5.1.2 The ORL face dataset

The Olivetti Research Laboratory (ORL) face dataset [43] 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. More in detail, these 400 images, each of 32×32 pixels, are stored in form of a single matrix, $\mathbf{M} \in \mathbb{N}_{+}^{n \times m}$, with $n = 400$, $m = 1024$, where each row stores a person's face image and each column entry represents the pixel value at that specific coordinate (a pixel is encoded as a gray-scale level from 0 to 255).

We then adopt the strategy proposed in [46] to generate an interval-valued matrix $\ddot{\mathbf{M}}$ starting from $\mathbf{M}$, 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 the same individual. For example, in the first row of pictures in Figure 5.1, 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, the proposed solution by the authors in [46] is to 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.

In order to generate an interval-valued entry $\ddot{\mathbf{M}}_{[i,j]}$ starting from a scalar value $\mathbf{M}_{[i,j]}$, 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

$$\ddot{\mathbf{M}}_{[i,j]} = \left[\mathbf{M}_{[i,j]} - \delta_{i,j}, \mathbf{M}_{[i,j]} + \delta_{i,j}\right]. \tag{5.1}$$

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

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

where $\alpha \in \mathbb{R}^{+}$ is a multiplicative scale coefficient, r is the coordinates' radius circumscribing the surrounding of $\mathbf{M}_{[i,j]}$, and the notation $x^{(i,j)}$ and $y^{(i,j)}$ is used to represent the coordinates $\langle x, y \rangle$ of the j-th pixel in the i-th image, i.e., the j-th element on the row vector $\mathbf{M}_{[i,:]}$. For the experiments presented later on, we set $r = 5$ and $\alpha = 2.5$.

The second row of pictures in Figure 5.1 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 $\ddot{\mathbf{M}}_{[i,j]}$).

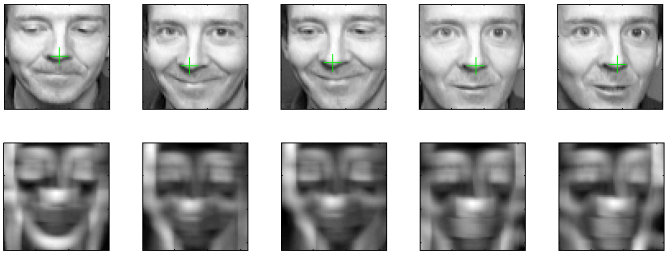


Figure 5.1: An example of how the radius matrices are generated from a set of face images from the ORL dataset

The approach taken by the authors in [46] to address the factorization requirements in face analysis is, as we detailed in Section 3.2.2, to extended the NMF technique to interval-valued data (they refer to their approach as I-NMF).

Instead, our approach for the factorization of this kind of interval-valued matrices relies again on the ISVD algorithms illustrated in Section 4.4, given that the non-negativity constraint doesn't affect the procedure. We will thus compare our results with the ones obtained by the authors in [46] over a set of experiments regarding the following three applications:

Nearest Neighbor Classification: For this task, the dataset is divided (by randomly selecting 50% of the rows for each individual) in a *training set* and a *test set*. The goal is to then assign the faces in the test set to the correct individual. The classification is performed on the principal components that represent the people-latent semantic of the input data. More in detail, for the I-NMF approach these principal components correspond to the columns of the (scalar) matrix $\mathbf{U}$, while for our ISVD approach, they need to be evaluated as the combination of both $\ddot{\mathbf{U}}$ and $\ddot{\mathbf{\Sigma}}$, namely $[\mathbf{U}_*\mathbf{\Sigma}_*, \mathbf{U}^*\mathbf{\Sigma}]$, either directly as interval-valued (ISVD_{1-4}), or averaged (ISVD_0). The classification is done using a 1-nearest neighbor classifier with Euclidean distance, that, when necessary, is extended to interval-valued data as the function that, taken the intervals $\ddot{a} = [a_*, a^*]$ and $\ddot{b} = [b_*, b^*]$ as input, evaluates their distance as:

$$d(\ddot{a}, \ddot{b}) = \sqrt{(a_* - b_*)^2 + (a^* - b^*)^2}.$$

We then evaluate the accuracy by means of the F_1-score measure, defined as the harmonic mean of the precision and recall of the classifier:

$$F_1 = \left(\frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} \right) = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}.$$

Clustering-based Classification: Here we try to classify the pictures in the ORL dataset through the K-means algorithm, attempting to cluster the 400 pictures in 40 clusters (hence $k = 40$), one for each individual. As in the previous task, the analysis is performed on the principal components that represent the people-latent semantic of the input data, and the Euclidean distance is used to measure the separation of the pictures in the latent space. The clustering accuracy between the known classification C and the retrieved result R is then measured by means of the *Normalized Mutual Information* (NMI), a similarity measure derived from information theory [10], defined as:

$$\text{NMI}(C,R) = \frac{\text{I}(C,R)}{\sqrt{\text{H}(C) \cdot \text{H}(R)}},$$

where $I(C,R)$ is the *mutual information* between C and R, defined as

$$\text{I}(C,R) = \text{I}(C) - \text{I}(C|R),$$

which provides a sound indication of the shared information between the two clusterings, while the normalizing factor is evaluated by means of the entropy H, which measures the *degree of disorder* inside each cluster.

Reconstruction: In this task, we attempt to reconstruct an approximation $\widetilde{\mathbf{M}}$ of the original data matrix $\mathbf{M}$ from a low-rank decomposition of the interval-valued matrix $\ddot{\mathbf{M}}$, and then we evaluate the reconstruction accuracy. For the I-NMF approach in [46], after decomposing the interval-valued matrix $\ddot{\mathbf{M}}$, obtaining the factor matrices $\mathbf{U}$ and $\ddot{\mathbf{V}}$ (the first scalar, the second interval-valued), the estimated scalar matrix $\widetilde{\mathbf{M}}$ is reconstructed through the following computation:

$$\widetilde{\mathbf{M}} = \frac{\mathbf{U}\mathbf{V}_*^{\text{T}} + \mathbf{U}\mathbf{V}^{*\text{T}}}{2}$$

To achieve a fair and consistent comparison, we compute the decomposition of $\ddot{\mathbf{M}}$ by means of the ISVD algorithms presented in Section 4.4, but, for the reconstruction options that return an interval valued approximation in terms

of the matrices $\widetilde{\mathbf{M}}_*$ and $\widetilde{\mathbf{M}}^*$, we also take their average to get an estimation of $\mathbf{M}$, i.e.,

$$\widetilde{\mathbf{M}} = \frac{\widetilde{\mathbf{M}}_* + \widetilde{\mathbf{M}}^*}{2}$$

We then evaluate the accuracy of both approaches by means of the Root-Mean Square Error (RMSE):

$$\text{RMSE}(\widetilde{\mathbf{M}}, \mathbf{M}) = \sqrt{\frac{\sum_{i=1}^{n}\sum_{j=1}^{m}(\widetilde{\mathbf{M}}_{[i,j]} - \mathbf{M}_{[i,j]})^2}{n \cdot m}}.$$

For each task, the accuracies are evaluated by averaging 100 different runs.

5.1.3 The social media datasets

In order to test our approaches also on data that are typically used in collaborative filtering scenarios, we took in consideration the following social media datasets:

Ciao and Epinions

On these two ratings data sets, *Ciao* and *Epinions* [52], we have performed a set of interval-valued matrix reconstruction experiments, in order to compare the accuracies of our ISVD strategies illustrated in Section 4.4 among each other. In particular, for both datasets we have considered *user-category* rating matrices, where each non-zero entry corresponds to the range of ratings that a user provided for the items of a given category. In this way, we simulate a scenario in which users are allowed to provide a range of ratings (modeled as interval-valued data), instead of a single one.

The characteristics and test parameters for the two datasets are:

Ciao

- 22 thousand users;
- 27 categories;
- Matrix Density= 26% of zero entries;
- Interval Density= 49%;
- Interval Intensity= 24.4%;
- Target Rank: 5%, 50%, 100% of the number of categories.

Epinions

- 7 thousand users;
- 28 categories;
- Matrix Density= 28% of zero entries;
- Interval Density= 44%;
- Interval Intensity= 22%;
- Target Rank: 5%, 50%, 100% of the number of categories.

After obtaining an approximation of the input data through the decomposition process, we evaluate the performance of the ISVD strategies by means of the harmonic mean, as described in Section 4.1.2.

MovieLens

The MovieLens dataset [22], made available by the GroupLens research group[2], contains 100 thousand ratings for 1682 movies (categorized in 19 genres) by 943 users, making the entries of a rating matrix $\mathbf{R} \in \mathbb{R}^{943\times1682}$. We rely on this dataset to evaluate the performance of our approaches in terms of reconstruction and missing ratings prediction (collaborative filtering).

For the **reconstruction** evaluation, we generate a *user-genre* interval-valued matrix $\ddot{\mathbf{M}} = [\mathbf{M}_*, \mathbf{M}^*]$, with $\mathbf{M}_*, \mathbf{M}^* \in \mathbb{R}^{943\times19}$, where each entry $\ddot{\mathbf{M}}_{[i,j]}$ corresponds to the range of ratings user i provided for all the movies of genre j, i.e., given G_j as the set of all the movie ids of genre j:

$$\mathbf{M}_{*[i,j]} = \min\left(\mathbf{R}_{[i,k]} | k \in G_j\right),$$
$$\mathbf{M}^*_{[i,j]} = \max\left(\mathbf{R}_{[i,k]} | k \in G_j\right).$$

We then apply our ISVD strategies to factorize $\ddot{\mathbf{M}}$ trying different target ranks (5%, 50%, 100% of the number of categories) and then, after reconstructing an approximation $\tilde{\ddot{\mathbf{M}}}$ of the input data, we evaluate the accuracy of the overall process by means of the harmonic mean, as described in Section 4.1.2.

For the **collaborative-filtering** evaluation, we followed the procedure explained

[2]http://www.grouplens.org

by the authors in [46], and illustrated in Section 3.2.2, to generate the interval-valued rating matrix $\ddot{\mathbf{M}}$ defined as

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

where the parameter $\delta_{i,j}$ takes into account the variability of user i's rating habit, as well as the variability of the evaluations on movie j, to transform a scalar rating $\mathbf{R}_{[i,j]}$ into a more general *preference degree*, $\ddot{\mathbf{M}}_{[i,j]}$. The missing ratings can then be computed (through equation 3.1 in Section 3.2.2) reconstructing the original matrix after a low-rank approximation.

We then use the Root-Mean Square Error (RMSE) to evaluate the accuracy of the reconstruction-based rating predictions, comparing the I-PMF factorization method of the authors in [46], with our AI-PMF decomposition (see Section 4.5), in order to show the benefits of a latent semantic alignment approach.

5.2 Competitors

As we discussed in Section 3.2.1, the authors in [14, 44] proposed a linear-programming based approach to solve the eigendecomposition problem for interval-valued data by looking for interval bounds to the eigenvector and eigenvalues. We have, therefore, implemented and experimented also with their proposed solution. We denote their linear-programming based approaches as LP_x, where the subscript x denotes one of the three application semantics (or options), a, b or c, described in Section 4.3.

As described in Section 5.1.2, we also compare our proposed ISVD approaches with NMF and I-NMF [46] for the face analysis tasks: data reconstruction and classification.

For collaborative filtering with social media data, discussed in Section 5.1.3, we used PMF and I-PMF [46] as competitors to our AI-PMF approach.

5.3 Experiments results

In this section we present the results for the sets of experiments described in Section 5.1 on the various datasets, synthetic and real-world.

5.3.1 Experiments on synthetic datasets

Uniform Data

Figure 5.2 provides an overview of the accuracy and execution times for the default configuration of the first batch of experiments described in Section 5.1.1. These results help us drawing the following conclusions:

- we obtain the highest accuracies using the $ISVD_{\#}$-b class of techniques[3] (returning both scalar-valued factors and interval-valued core); in particular, the highest overall accuracy is provided by $ISVD_4$-b, which leverages both semantic alignment and latent space recomputation techniques, as described in Section 4.4.5;
- the naive approach, $ISVD_0$, is the fastest overall, but does not match the accuracy of the $ISVD_{\#}$-b class, which capture the interval information in the core matrix;
- the $ISVD_{\#}$-a class of techniques (returning both interval-valued factors and core) does not lead to a high overall reconstruction accuracy, proving that relying on the interval matrix product for the reconstruction is not a good option;
- the $ISVD_{\#}$-c class of techniques (returning all factor matrices as scalar) has similar results to the $ISVD_0$ strategy, since the approximations involved in the latent semantic alignment process (which, at this point, would be redundant) have a slightly negative impact on the final accuracy – this reconstruction option is thus indicated only with $ISVD_0$ strategy;
- the linear-programing (LP_x) based competitors' [14, 44] approach shows poor accuracies and massive execution times: the reason for this poor performance is that, since these approaches look at finding interval bounds

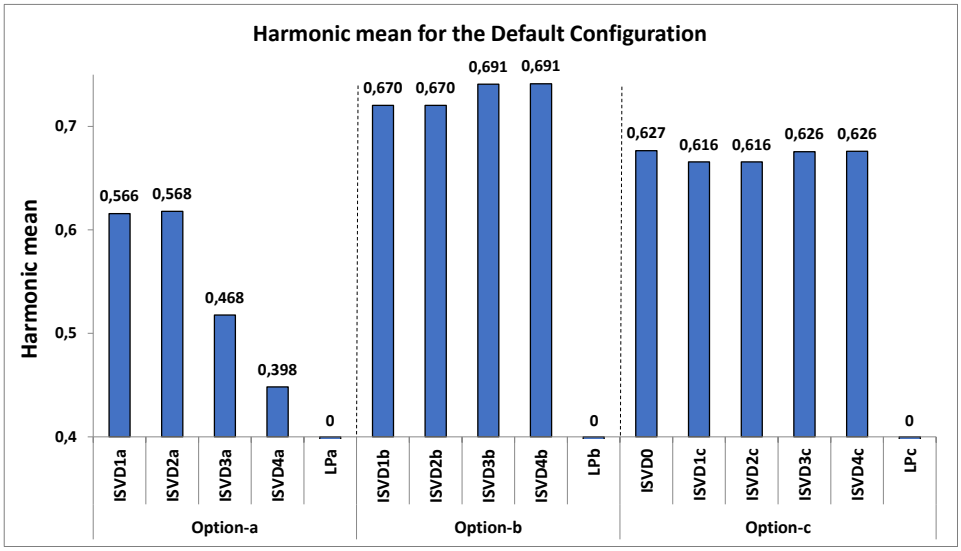
[3] Here, $ISVD_{\#}$ is a shorthand for the set $\{ISVD_1, ISVD_2, ISVD_3, ISVD_4\}$ of algorithms that rely on latent semantic alignment techniques for decomposing the given interval-valued input matrix.

for eigenvectors and eigenvalues, they are not very suited for the task at stake, failing to provide a valid representation of the latent information of the interval-valued data in a transformed vector space. Moreover, looking for the bounds of each component of every eigenvector is necessarily a very resource-intensive task.

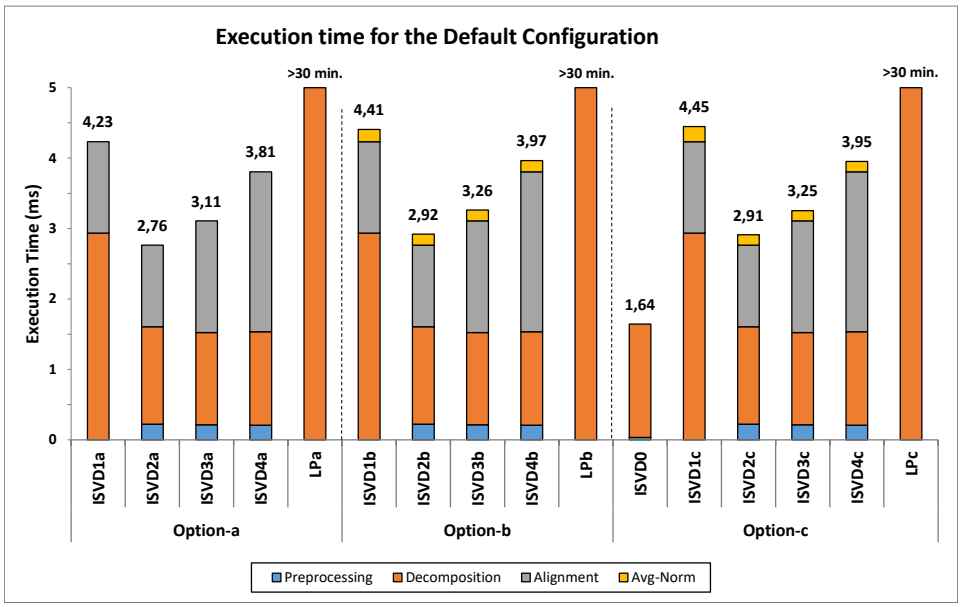
Table 5.3 shows the accuracy results for $ISVD_{\#}$-b class of techniques (which, as we have seen above, provide the best overall accuracies) under the various configurations reported in Table 5.1. In particular, for each set of experiments, we vary one of the parameters at a time (according to its possible values), while keeping the others as default. As a reference, we also included the results for $ISVD_0$, since it is the fastest alternative option[4] (although less accurate). From these experiments, we can conclude that:

- In Table 5.3(a) we see how the size of the input matrices affects the decomposition process: as expected, we get higher accuracies for smaller matrices, however, the relative performances between the different approaches stay the same, with $ISVD_4$-b still providing the best accuracies.
- In Table 5.3(b), we see that all algorithms perform more accurately with denser data (i.e., less sparse input matrices), however, again, their relative performances stay the same.
- In Table 5.3(c), we see that $ISVD_4$-b is fairly robust also against different interval densities, proving its efficacy also with input matrices entirely made of interval-valued entries. In comparison, $ISVD_0$ worsen quickly as the interval density increases (as expected, since more of the interval information gets lost in the averaging processes).
- Table 5.3(d), where the intensities of the interval-valued entries varies (i.e., we get progressively larger intervals), shows a similar pattern: $ISVD_0$ is competitive only when the interval intensity is very low ($\sim$10%), while $ISVD_{\#}$-b strategies prove robust even against large interval intensities with, again, $ISVD_4$-b providing the highest effectiveness overall.
- Finally, Table 5.3(e) shows the impact of different decomposition target

[4] We left out the LP_x based competitors results since, as stated above, this approach performs very poorly for this kind of data.



(a) Accuracy



(b) Execution time breakdown

Figure 5.2: Comparison of the alternative approaches for **Synthetic Uniform Data** reconstruction – default configuration (*the higher the Harmonic mean, the better the result*)

Matrix size	**ISVD$_0$**	**ISVD$_1$-b**	**ISVD$_2$-b**	**ISVD$_3$-b**	**ISVD$_4$-b**
25-by-400	0.697	0.758	0.758	**0.789**	**0.789**
40-by-250	0.627	0.670	0.670	**0.691**	**0.691**
250-by-40	0.627	0.670	0.670	**0.691**	**0.691**
400-by-250	0.503	0.535	0.535	0.541	**0.542**
250-by-400	0.503	0.535	0.535	0.541	**0.542**

(a) Varying matrix size

Matrix density	**ISVD$_0$**	**ISVD$_1$-b**	**ISVD$_2$-b**	**ISVD$_3$-b**	**ISVD$_4$-b**
0%	0.627	0.670	0.670	**0.691**	**0.691**
50%	0.493	0.511	0.511	0.530	**0.530**
90%	0.425	0.431	0.431	0.451	**0.451**

(b) Varying matrix density

Interval density	**ISVD$_0$**	**ISVD$_1$-b**	**ISVD$_2$-b**	**ISVD$_3$-b**	**ISVD$_4$-b**
10%	**0.702**	0.689	0.689	**0.702**	**0.702**
25%	0.678	0.659	0.659	0.682	**0.683**
75%	0.641	0.657	0.657	**0.681**	**0.681**
100%	0.627	0.670	0.670	**0.691**	**0.691**

(c) Varying interval density

Interv. intensity	**ISVD$_0$**	**ISVD$_1$-b**	**ISVD$_2$-b**	**ISVD$_3$-b**	**ISVD$_4$-b**
10%	0.708	**0.709**	**0.709**	**0.709**	**0.696**
25%	0.702	0.704	0.704	**0.708**	**0.709**
75%	0.658	0.681	0.681	**0.698**	**0.698**
100%	0.627	0.670	0.670	**0.691**	**0.691**

(d) Varying interval intensity

Target Rank	**ISVD$_0$**	**ISVD$_1$-b**	**ISVD$_2$-b**	**ISVD$_3$-b**	**ISVD$_4$-b**
5	0.501	0.537	0.537	**0.539**	**0.539**
10	0.546	0.585	0.585	0.591	**0.592**
20	0.627	0.670	0.670	**0.691**	**0.691**
40	0.758	0.814	0.814	**0.895**	**0.896**

(e) Varying decomposition target rank

Table 5.3: Comparison of the alternative ISVD$_\#$-b reconstruction approach with varying parameters over the **Uniform Synthetic Data** (*the higher the Harmonic Mean, the better the result*)

ranks. Here, we see that, when the rank is very low, ISVD_0 provides a somewhat competitive accuracy. However, as the decomposition rank increases (i.e., as we try to maintain more detailed information while we factorize the input data matrix), ISVD_4-b provides increasingly better accuracy.

Overall, the reported experiments show that ISVD_4-b provides the best accuracy under all data scenarios considered, while the fastest alternative, ISVD_0, is not able, in general, to provide competitive results.

Anonymized Data

As we discussed in Section 5.1.1, we have also run experiments on synthetic interval-valued data generated by means of a generalization process that simulates an anonymized dataset. Figure 5.3 shows how our ISVD strategies perform over this kind of data (measuring the accuracy through the harmonic mean), considering three levels of generalization (*high*, *medium*, and *low*) and, for each of those, three decomposition target ranks (100%, 50% and 5%). We left off the results for the LP_x based comptetitors since, again, they don't perform well for this kind of tasks, leading to harmonic mean accuracies lower than 0.01.

From the results we can see that, once again, the $\text{ISVD}_\#$-b strategy (which returns interval-valued $\ddot{\mathbf{\Sigma}}$ and scalar factor matrices $\mathbf{U}$ and $\mathbf{V}$) provides the best overall accuracy, except for the very low-rank (5%) decomposition under the low-privacy scenario (Figure 5.3(c)), where the interval-valued entries are fewer and with closer endpoints. In this case, decomposition target *a*, which maintains the interval information not only for the core, but also for the factor matrices, leads to slightly better results.

Once again, these results confirm that ISVD_4-b provides the overall best accuracy in the most part of the scenarios inspected.

Anonymization High Privacy [10%, 20%, 30%, 40%]		100% rank		50% rank		5% rank	
		H-mean	Order	H-mean	Order	H-mean	Order
Option a	ISVD1	0,7485	13	0,6517	12	0,5116	5
	ISVD2	0,7506	12	0,6531	11	0,5116	5
	ISVD3	0,7699	10	0,6542	10	0,5006	12
	ISVD4	0,7687	11	0,6335	13	0,4701	13
Option b	ISVD1	0,8693	6	0,6906	3	0,5124	3
	ISVD2	0,8693	6	0,6906	3	0,5124	3
	ISVD3	0,9349	**1**	0,7029	2	0,5126	**1**
	ISVD4	0,9349	**1**	0,7030	**1**	0,5126	**1**
Option c	ISVD0	0,8780	3	0,6850	5	0,5011	7
	ISVD1	0,8323	8	0,6731	8	0,5008	10
	ISVD2	0,8323	8	0,6731	8	0,5008	10
	ISVD3	0,8761	4	0,6846	7	0,5010	9
	ISVD4	0,8761	4	0,6847	6	0,5010	8

(a) Anonymized data – High privacy

Anonymization Medium Privacy [25%, 25%, 25%, 25%]		100% rank		50% rank		5% rank	
		H-mean	Order	H-mean	Order	H-mean	Order
Option a	ISVD1	0,7401	13	0,6501	13	0,5133	5
	ISVD2	0,7421	12	0,6515	12	0,5133	5
	ISVD3	0,7728	11	0,6590	11	0,5073	12
	ISVD4	0,8138	10	0,6611	10	0,4902	13
Option b	ISVD1	0,8672	6	0,6906	6	0,5140	3
	ISVD2	0,8672	6	0,6906	6	0,5140	3
	ISVD3	0,9338	2	0,7030	2	0,5143	**1**
	ISVD4	0,9339	**1**	0,7031	**1**	0,5143	**1**
Option c	ISVD0	0,9003	3	0,6934	3	0,5080	7
	ISVD1	0,8462	8	0,6809	8	0,5077	10
	ISVD2	0,8462	8	0,6809	8	0,5077	10
	ISVD3	0,8977	5	0,6929	5	0,5080	8
	ISVD4	0,8978	4	0,6930	4	0,5080	8

(b) Anonymized data – Medium privacy

Anonymization Low Privacy [40%, 30%, 20%, 10%]		100% rank		50% rank		5% rank	
		H-mean	Order	H-mean	Order	H-mean	Order
Option a	ISVD1	0,5828	11	0,4788	10	0,2465	**1**
	ISVD2	0,6024	10	0,4889	9	0,2465	**1**
	ISVD3	0,5456	12	0,4459	13	0,2385	7
	ISVD4	0,5456	12	0,4562	12	0,2181	13
Option b	ISVD1	0,7648	3	0,4788	10	0,2447	3
	ISVD2	0,7648	3	0,5827	3	0,2447	3
	ISVD3	0,8047	**1**	0,5993	**1**	0,2446	5
	ISVD4	0,8047	**1**	0,5993	**1**	0,2446	5
Option c	ISVD0	0,7630	5	0,5777	4	0,2360	11
	ISVD1	0,7337	8	0,5646	7	0,2364	8
	ISVD2	0,7337	8	0,5646	7	0,2364	8
	ISVD3	0,7630	5	0,5775	6	0,2361	10
	ISVD4	0,7630	5	0,5776	5	0,2360	11

(c) Anonymized data – Low privacy

Figure 5.3: Accuracy comparison for **Anonymized Synthetic Data** for different target ranks (*the greener the cell, the better the result* – the tables are best viewed in color)

5.3.2 Experiments on the ORL Face datasets

With this set of experiments, we were particularly interested in evaluating the performances of our ISVD approach in terms of classification accuracy in a real-world scenario (in this case being able to assign a face image to a specific individual). From what we observe in the results, it appears that the interval latent features that we are able to extract (especially with low rank decompositions) are indeed very effective at identifying the salient characteristics of an individual. We also report the results for the reconstruction experiments since, even if the data in exam are not particularly suited for this task, they can still provide insight on the overall accuracy of the decomposition process.

Nearest Neighbor-based classification

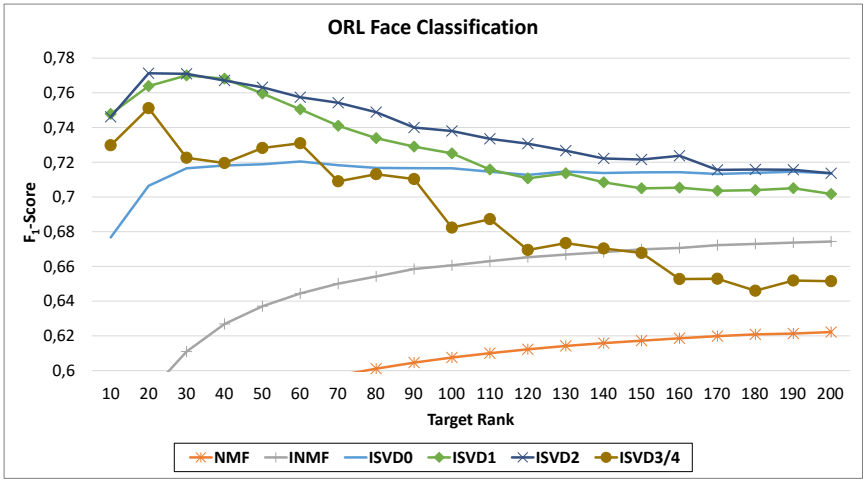
Figure 5.4(a) shows the results for a set of experiments on the Nearest Neighbor classification task described in Section 5.1.2. The performance of the various approaches (our ISVD strategies compared to NMF and I-NMF) is measured by means of the F_1-score function, varying the target rank of the decomposition from 10 (2.5%) to 200 (50%).

As we can see, the best classification results are obtained using the ISVD_1 and ISVD_2 techniques, both of which leverage the latent semantic alignment, and this is particularly evident under low rank scenarios ($r = 20$ or $r = 30$). It is important to notice that, since the classification task only relies on the interval-valued $\ddot{\mathbf{U}}$ factors and core matrix $\ddot{\mathbf{\Sigma}}$, the ISVD_3 and ISVD_4 techniques (which attempt to further improve the latent space defined by the $\ddot{\mathbf{V}}$ factor) do not provide, as expected, additional benefits: they in fact actually hurt the classification accuracy, by bringing the interval-valued vectors closer, while losing some of the interval information.

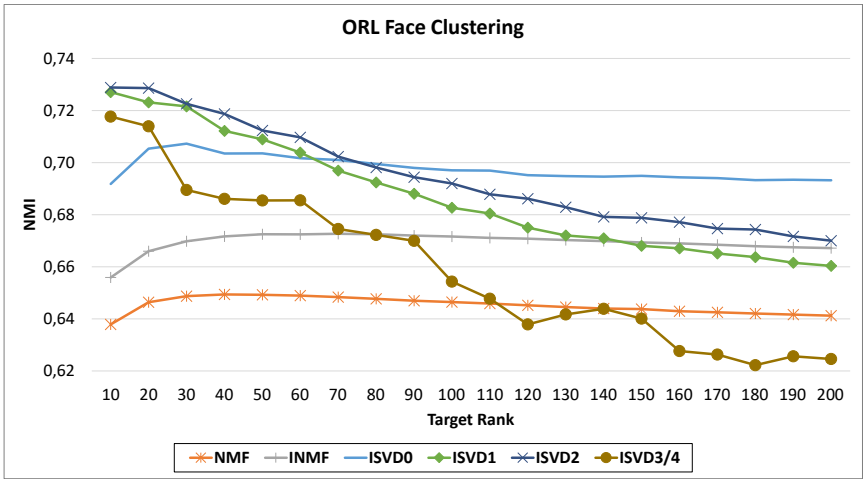
The NMF and I-NMF competitors, both provide significantly lower accuracies than our ISVD-based approaches, however their relation confirms the advantage of leveraging the interval information in the data for this kind of task.

Clustering-based classification

Figure 5.4(b) shows the results for the clustering-based classification task (also described in Section 5.1.2) based on the the K-means algorithm, measuring the



(a) NN-based Classification (*the higher, the better*)



(b) Clustering-based Classification (*the higher, the better*)

ORL Face Reconstruction		rank = 200		rank = 100		rank = 10	
		RMSE	Order	RMSE	Order	RMSE	Order
Option a	ISVD1	26,82	11	24,26	11	26,10	9
	ISVD2	23,30	10	24,14	10	25,71	4
	ISVD3	14,91	4	15,99	4	53,01	10
	ISVD4	21,56	9	19,74	5	113,50	11
Option b	ISVD1	19,33	5	20,30	6	25,96	5
	ISVD2	19,50	7	20,46	8	25,98	7
	ISVD3	8,84	3	11,75	3	23,09	3
	ISVD4	4,96	**2**	9,18	**2**	21,58	**2**
Option c	ISVD0	4,46	**1**	8,65	**1**	21,18	**1**
	ISVD1	19,33	5	20,30	6	25,96	5
	ISVD2	19,50	7	20,46	8	25,98	7
	ISVD3	8,84	3	11,75	3	23,09	3
	ISVD4	4,96	**2**	9,18	**2**	21,58	**2**
	NMF	16,73	7	18,32	7	23,80	6
	I-NMF	19,02	8	20,29	9	25,12	7

(c) Reconstruction (*the lower, the better*)

Figure 5.4: (a) Reconstruction and (b,c) Classification results for the **ORL face dataset** experiments

Clustering on:	rows of $\mathbf{M}$	rows of $\ddot{\mathbf{M}}$	$\ddot{\mathbf{U}}\ddot{\mathbf{\Sigma}}$ (ISVD$_2$, $r = 20$)
Accuracy (NMI)	0.69	0.72	0.72
Exectution time (sec)	0.04	1.14	0.11 (0.08 + 0.03)

Table 5.4: Accuracy and execution time (*decomposition* + *clustering*, or just *clustering* time) for clustering-based classification using original data row vectors, interval-valued data row vectors, and low-rank interval-valued principal components

accuracy by means of the Normalized Mutual Information (NMI). The results confirm the outcome of the NN-based classification task discussed above showing that, once again, the best accuracies are obtained relying on the ISVD$_1$ and ISVD$_2$ techniques (both of which leverage the latent semantic alignment), especially under low rank ($r \simeq 10$) decomposition scenarios.

Finally, in Table 5.4 we report the results (in terms of accuracy, as NMI, and time of execution, in seconds) for a set of experiments devised for comparing three types of clustering:

1. A clustering directly applied to the raw scalar data, namely the row vectors of the input matrix $\mathbf{M}$ (where each row represents a face in the dataset).
2. A clustering applied to the interval-valued data (generated by means of equation 5.1 in Section 5.1.2), namely the row vectors of $\ddot{\mathbf{M}}$.
3. A clustering applied to the principal components, i.e., $\ddot{\mathbf{U}}\ddot{\mathbf{\Sigma}}$, obtained through the ISVD$_2$ decomposition with target rank 20 (i.e., the best option from the experiments reported in Figure 5.4(b)).

As we see in Table 5.4, adding interval information already significantly improves the clustering accuracy, however this comes at a significant execution time cost. Our proposed interval-valued decomposition strategy cuts down the execution time significantly, almost being comparable with the scalar approach (although adding a bit of overhead for the decomposition process), while matching the accuracy provided by the interval-valued option.

Reconstruction

Here we inspect the reconstruction accuracy of our ISVD approaches in comparison with the NMF and I-NMF strategies, as described in Section 5.1.2.

Figure 5.4(c) shows the reconstruction performances (in terms of Root Mean Square Error, RMSE) of the considered options for three classes of target rank: 50% ($r = 200$), 25% ($r = 100$) and 2.5% ($r = 10$). As we can see, ISVD_0 provides the overall best reconstruction accuracies (slightly overtaking ISVD_4), while the competitor techniques, NMF and I-NMF, perform poorly in comparison with most of our methods.

Looking at these results, it seems to be the case that for this particular task the introduction of interval information does not provide a sufficient advantage to justify the overhead required for the decomposition process. In fact, since the generated intervals are symmetrical (see equation 5.1 in Section 5.1.2), the average of the interval-valued matrix matches exactly the original data, so that the ISVD_0 process becomes analogous to the standard SVD applied to the original scalar matrix. This seems to be true also looking at how NMF and I-NMF compare to one another, with the standard decomposition working better than the one extended to manage interval-valued data. It is still interesting to see, however, how ISVD_4 doesn't loose too much information, proving to be a reasonably effective decomposition approach even under unfavorable conditions.

5.3.3 Experiments on social media datasets

Reconstruction

Figure 5.5 presents, in a similar way as for the synthetic datasets, the reconstruction accuracy (again in terms of harmonic mean) of our ISVD approach, this time on three real world datasets, *Ciao*, *Epinions*, and *MovieLens*, where the interval-valued data has been generated as described in Section 5.1.3.

These results are comparable, confirming that, in general, the decomposition strategies $\text{ISVD}_\#$-b (with interval-valued core and scalar factor matrices) lead to the best results, except for very low-rank decompositions, where option $\text{ISVD}_\#$-a (with interval-valued core and factor matrices) performs better.

More in detail, for the Ciao and Epinion datasets, we see that $ISVD_3$ and $ISVD_4$ (which perform latent alignment early), lead to better results, while $ISVD_1$ and $ISVD_2$ (which perform the latent alignment later in the process) are more effective than $ISVD_3$ and $ISVD_4$ only for very low rank decompositions.

The MovieLens dataset also has a similar behavior: for relatively large decomposition ranks, $ISVD_3$ and $ISVD_4$ lead again to better results, while for low rank decompositions ($r \leq 10$) the benefit of early alignment is lost and $ISVD_1$ and $ISVD_2$ are (slightly) more effective. However, the behavior w.r.t. the other two dataset is a bit different for medium (50%) target rank decompositions, where $ISVD_1$ and $ISVD_2$ are here more effective than $ISVD_3$ and $ISVD_4$.

Prediction (Collaborative Filtering)

In this set of experiments we show how our proposed latent semantic alignment method can be applied to (interval) probabilistic matrix factorization to achieve more effective performances in collaborative filtering. Figure 5.6 shows the accuracies (in terms of Root Mean Square Error, RMSE) of the PMF, I-PMF and (our) AI-PMF approaches on the MovieLens dataset, varying the target rank of the decomposition from 10 (2.5%) to 200 (50%).

As the chart shows, the prediction accuracy of all algorithms improves as we consider higher decomposition ranks and the proposed latent semantic alignment based approach, AI-PMF, leads to better prediction performance than both PMF and I-PMF for decomposition ranks greater than 60.

Most importantly, however, AI-PMF always performs better than I-PMF indicating that, as expected, latent semantic alignment helps achieve better handling of interval-valued factors, also when considering factors with probabilistic interpretations, rather than eigen-vector based interpretations, as they were considered in Section 4.2.

Ciao		100% rank (=28)		50% rank (=14)		5% rank (=1)	
		H-mean	Order	H-mean	Order	H-mean	Order
Option a	ISVD1	0,6446	13	0,4746	12	0,2475	**1**
	ISVD2	0,6614	11	0,4844	10	0,2475	**1**
	ISVD3	0,6453	12	0,4740	13	0,2417	7
	ISVD4	0,6706	10	0,4843	11	0,2280	13
Option b	ISVD1	0,8047	6	0,5518	3	0,2463	3
	ISVD2	0,8047	6	0,5518	3	0,2463	3
	ISVD3	0,8383	**1**	0,5640	2	0,2462	5
	ISVD4	0,8383	**1**	0,5641	**1**	0,2462	6
Option c	ISVD0	0,8072	3	0,5518	3	0,2403	12
	ISVD1	0,7806	8	0,5410	8	0,2406	8
	ISVD2	0,7806	8	0,5410	8	0,2406	8
	ISVD3	0,8072	3	0,5517	7	0,2404	10
	ISVD4	0,8072	3	0,5517	6	0,2403	11

(a) Ciao data set

Epinion		100% rank (=27)		50% rank (=14)		5% rank (=1)	
		H-mean	Order	H-mean	Order	H-mean	Order
Option a	ISVD1	0,5828	11	0,4788	11	0,2465	**1**
	ISVD2	0,6024	10	0,4889	10	0,2465	**1**
	ISVD3	0,5456	13	0,4459	13	0,2385	7
	ISVD4	0,5710	12	0,4562	12	0,2181	13
Option b	ISVD1	0,7648	3	0,5827	3	0,2447	3
	ISVD2	0,7648	3	0,5827	3	0,2447	3
	ISVD3	0,8047	**1**	0,5993	2	0,2446	5
	ISVD4	0,8047	**1**	0,5993	**1**	0,2446	6
Option c	ISVD0	0,7630	5	0,5777	5	0,2360	11
	ISVD1	0,7337	8	0,5646	8	0,2364	8
	ISVD2	0,7337	8	0,5646	8	0,2364	8
	ISVD3	0,7630	5	0,5775	7	0,2361	10
	ISVD4	0,7630	5	0,5776	6	0,2360	11

(b) Epinion data set

MovieLens		100% rank (=19)		50% rank (=10)		5% rank (=1)	
		H-mean	Order	H-mean	Order	H-mean	Order
Option a	ISVD1	0,6332	6	0,6156	6	0,5469	**1**
	ISVD2	0,7367	5	0,7200	5	0,5469	**1**
	ISVD3	0,5269	12	0,5368	12	0,4452	7
	ISVD4	0,2146	13	0,2673	13	0,2423	13
Option b	ISVD1	0,7651	3	0,7298	**1**	0,5356	3
	ISVD2	0,7651	3	0,7298	**1**	0,5356	3
	ISVD3	0,7676	**1**	0,7282	4	0,5334	5
	ISVD4	0,7676	**1**	0,7287	3	0,5317	6
Option c	ISVD0	0,5813	9	0,5530	10	0,3995	11
	ISVD1	0,5889	7	0,5645	7	0,4119	8
	ISVD2	0,5889	7	0,5645	7	0,4119	8
	ISVD3	0,5813	9	0,5540	9	0,4049	10
	ISVD4	0,5813	9	0,5529	11	0,3992	12

(c) Movielens data set

Figure 5.5: Accuracy comparison for **Social Media Data** for different target ranks (*the greener the cell, the better the result* – the tables are best viewed in color)

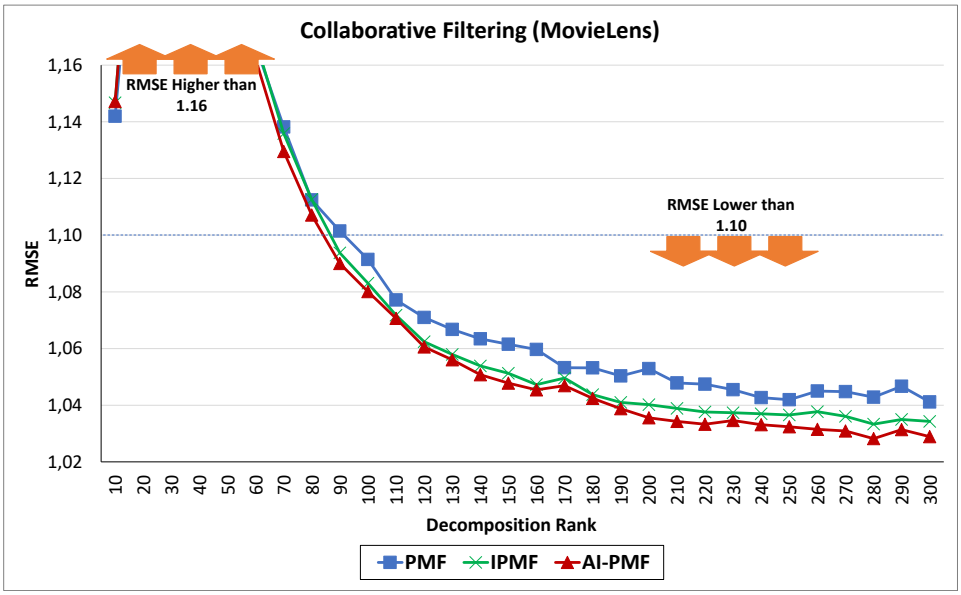


Figure 5.6: Collaborative filtering for the **MovieLens Dataset** (*the lower, the better*)

5.4 Conclusions

The set of experiments presented in this chapter confirmed our expectations regarding the decomposition techniques (illustrated in Chapter 4) applied to interval-valued datasets, both synthetic and real-world ones.

In fact, our ISVD approaches (Section 4.4) proved efficient for the **reconstruction** task, where we were interested in evaluating the accuracy of our approach in providing a truthful representation of the data in the transformed vector space while, at the same time, not loosing too much of the interval information in the original data. Specifically, the ISVD4 method, designed to improve the interval-valued latent-space representation, is confirmed to be the better suited for this task, particularly following the (b) option (see Figure 4.8 for details) for the reconstruction steps, reaching the best compromise in keeping the interval information while avoiding the introduction of too much noise.

As regards the **classification** task, we followed the intuition of the authors in [46] of relying on interval-valued data for highlighting the facial features of the ORL face dataset for improving the overall accuracy, but thanks to our Interval Latent Semantic Alignment (ILSA, Section 4.2) method, we managed to get even

better results, thanks to a more faithful factorization of the interval-valued components. The ISVD2 strategy, in particular, proved to be the most efficient, since (although less accurate in representing an interval-valued latent semantic space w.r.t. ISVD3 and ISVD4) it better discriminates the features (represented by the weighted left singular vectors) that help identifying the specific characteristics of a given individual.

In a similar way, the results for the **collaborative filtering** experiments show that the latent semantic alignment helps improving the prediction capability of probabilistic models (see AI-PMF, Section 4.5) when dealing with interval-valued data.

Part II

Tensor Decomposition

Chapter 6

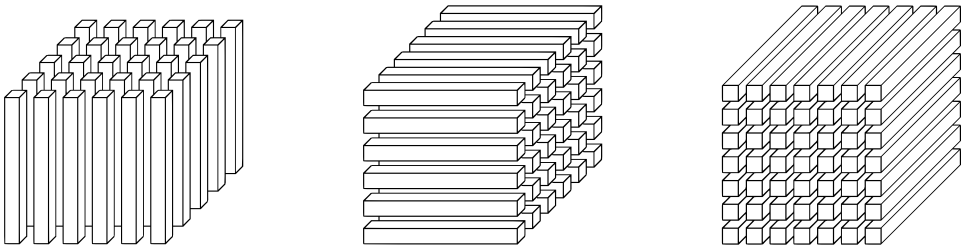
Tensors and Tensor Factorization

In the context of this thesis, a *tensor* is intended as a multi-dimensional array, i.e., a generalizations of a matrix with *order* (its number of dimensions, or *modes*) greater than 2. More in detail, while a vector can be thought of as a tensor of order one and a matrix as a tensor of order two, a generic tensor is intended as an array of order three or higher [1].

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 $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3}$, the fibers are referred to as *columns* (mode-1), *rows* (mode-2) and *tubes* (mode-3), see Figure 6.1, 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*, see Figure 6.2, defined by fixing one of the three indices, and denoted as $\mathbf{X}_{[i_1,:,:]}$, $\mathbf{X}_{[:,i_2,:]}$ and $\mathbf{X}_{[:,:,i_3]}$.

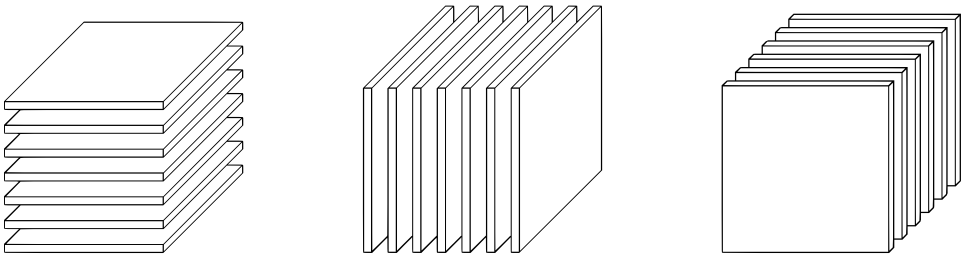
In literature, many decomposition techniques have been applied to tensors, forming the basis for many data analysis and knowledge discovery tasks, from clustering, trend and anomaly detection [30], correlation analysis [49], to pattern discovery [27]. Initially originated in the field of psychometrics [8], tensor decom-

[1] In terms of notation, we follow the convention [29] of denoting higher-order tensors (order three or higher) by boldface Euler script letters, e.g., $\mathcal{X}$, $\mathcal{Y}$.



(a) Mode-1 fibers (columns) (b) Mode-2 fibers (rows) (c) Mode-3 fibers (tubes)

Figure 6.1: Tensor Fibers



(a) Horizontal slices (b) Lateral slices (c) Frontal slices

Figure 6.2: Tensor Slices

position has been used in a large number of domains of computer science, including signal processing, computer vision, and data mining.

In the following, two of the most adopted tensor factorization techniques, CP and Tucker decompositions [53], are briefly outlined, followed by a description of a more recent approach to the problem, the *Tensor-Train* decomposition [38], which would serve as the starting point for the extension of the factorization analysis to interval-valued tensors, as will be described in Chapter 7.

6.1 CP Decomposition

The CANDECOMP [8] (*CANonical DECOMPosition*) and PARAFAC [23] (*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

$$\mathcal{X} \simeq \sum_{k=1}^{r} \lambda_k \mathbf{A}^{(1)}_{[:,k]} \circ \mathbf{A}^{(2)}_{[:,k]} \circ \cdots \circ \mathbf{A}^{(d)}_{[:,k]}, \tag{6.1}$$

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 $\lambda = [\lambda_1, \lambda_2, \ldots, \lambda_r]$ 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 Figure 6.3 for a three-mode example).

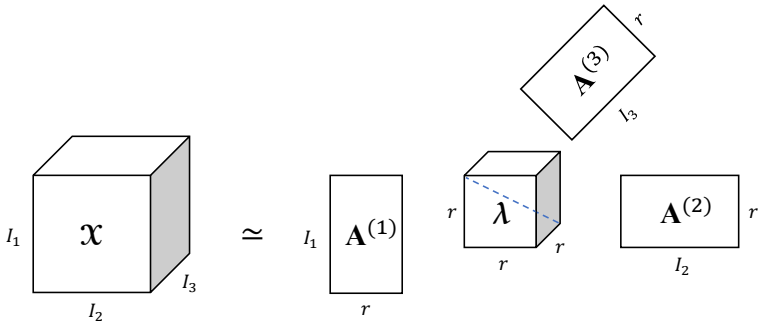


Figure 6.3: Candecomp/Parafac (CP) Decomposition of a three-mode tensor $\mathcal{X}$ into three factor matrices (one for each mode) and a diagonal core tensor (whose non-zero elements absorb the normalizing factors of each column vector in the factor matrices).

When there is an equality in 6.1, 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 [24]). 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 [8, 23]: 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.

6.2 Tucker Decomposition

The Tucker decomposition can be seen as a generalization of the Singular Value Decomposition (SVD) 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:

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

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 $\mathbf{X}^{(n)}$.

Figure 6.4 illustrates the Tucker factorization for a three-mode tensor. The resemblance to Figure 6.3 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 $r_1 = r_2 = \cdots = r_d$.

The most efficient way to compute the Tucker decomposition of a tensor, referred to as the Higher-Order Singular Value Decomposition (HOSVD for short) [11], 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 [31], thus increasing the accuracy.

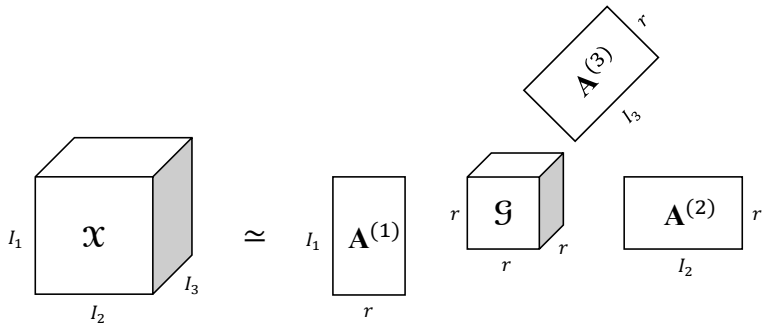


Figure 6.4: Tucker Decomposition of a three-mode tensor $\mathcal{X}$ into three factor matrices (which are usually orthogonal and can be thought of as the *principal components* in each mode) and a core tensor (whose entries show the level of interaction between the different components).

6.3 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.

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

The CP decomposition 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 [24]), which not only is not guaranteed to be found, but also its numerical approximation is proven to be an ill-posed problem [12], 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.

On the other hand, 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 suit-

able when dealing with a low number of dimensions (especially three, [39]), it is rarely a good option when d gets larger.

The Tensor-Train decomposition [38] 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 *compression ranks*, define the accuracy of the decomposition, which not only can be exact (*full-rank decomposition*), but can also be approximated with any given accuracy ε (a common practice in scientific computing when it is necessary to prioritize efficiency over accuracy), such that

$$\frac{\left\| \mathcal{X} - \widetilde{\mathcal{X}} \right\|_F}{\|\mathcal{X}\|_F} \leq \varepsilon,$$

where $\widetilde{\mathcal{X}}$ is the approximation of $\mathcal{X}$, and $\|\cdot\|_F$ is the Frobenius norm.

Figure 6.5 illustrates the process to compute the Tensor-Train form of a given tensor $\mathcal{X}$. The procedure is based on the Singular Value Decomposition of the auxiliary matrices $\mathbf{C}_k$, obtained by *reshaping*[2] (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.

---

[2] 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).

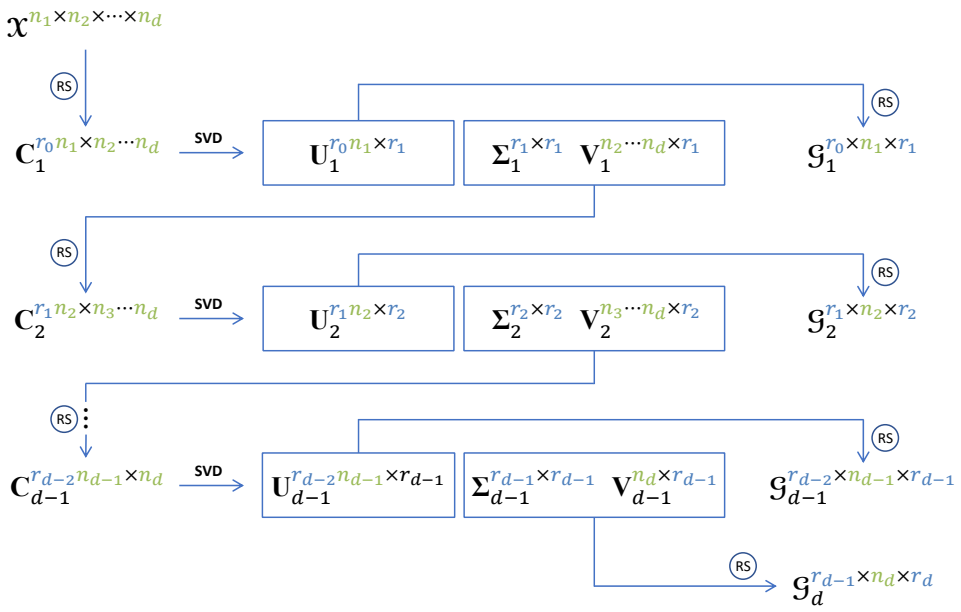


Figure 6.5: Tensor-Train Decomposition

Figure 6.6 illustrates the process that, taking the d cores $\mathcal{G}$ of the Tensor-Train decomposition, reconstructs either the original tensor $\mathcal{X}$, or a close approximation, $\widetilde{\mathcal{X}}$, according to the accuracy of the initial decomposition. The procedure relies again on the *reshape* (either of tensors or matrices) into auxiliary matrices that are easily combined by means of matrix multiplication (notated as $\otimes$ in figure).

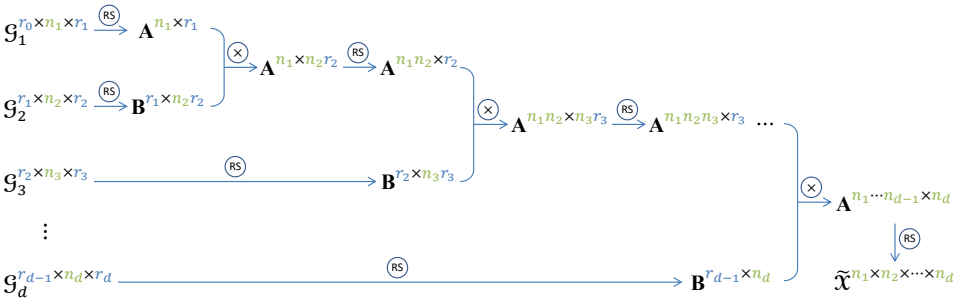


Figure 6.6: Tensor-Train Reconstruction

The property of the Tensor-Train decomposition of relying 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 Chapter 7.

6.4 Tensor decomposition in 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 from 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 [57], 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 [35] 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.

Other proposed methods to handle uncertainty and imprecision in the data, besides probabilistic approaches, rely on fuzzy set theory [58], 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 three-way 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-value high-dimensional data, a problem not yet thoroughly investigated in the literature, by relying on the findings of our work with matrices, presented in the first part of the thesis.

Chapter 7

Tensor factorization with interval-valued data

This chapter describes how the techniques explored in Chapter 4 for dealing with interval-valued matrices can be leveraged to also factorize interval-valued tensors. In particular, as a case study, we present an extension of the *tensor train* decomposition (introduced in section 6.3), taking advantage of the fact that its core operations just involve matrices, allowing us to exploit the interval-valued decomposition strategies (ISVD) already developed (see Section 4.4).

7.1 Interval-valued Tensor-Train Decomposition (ITTD)

As we saw in section 6.3, the Tensor-Train decomposition is mainly based on the matrix Singular Value Decomposition (SVD) and has the purpose of representing any given d-dimensional tensor by means of d three-dimensional ones.

For an interval-valued scenario, 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 $r_0 = r_d = 1$.

Figure 7.1 illustrates the Interval Tensor-Train Decomposition (ITTD) procedure, that computes the Tensor-Train form of a given interval-valued tensor $\ddot{\mathcal{X}}$. While the standard tensor train decomposition relied on the SVD, here, since at each step k the auxiliary matrices $\ddot{\mathbf{C}}_k$ are also interval-valued, we can apply one of our ISVD strategies to obtain their factorization in terms of the factor matrices $\ddot{\mathbf{U}}_k$, $\ddot{\mathbf{V}}_k$ and $\ddot{\mathbf{\Sigma}}_k$. However, since we are dealing with interval-valued entries, a few more measures need to be taken into account, as detailed by the following operations (also notated in figure):

(RS) The **ReShape** function, already introduced for the scalar scenario, is easily extended to the interval-valued case, since 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).

(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 $\ddot{\mathbf{C}}_{k+1}$ and proceed with the next step of the decomposition procedure. More specifically, we contemplate four options, according to whether we want to keep them either as interval-valued or scalar (in which case the average of their endpoints is considered, as $\overline{\mathbf{V}}_k = \frac{\mathbf{V}_{k*} + \mathbf{V}_{k*}}{2}$ 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) $\ddot{\mathbf{C}}_{k+1} = \left[\mathbf{\Sigma}_{k*} \cdot \overline{\mathbf{V}}_k, \mathbf{\Sigma}_k^* \cdot \overline{\mathbf{V}}_k\right]$;

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

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

(AV) The **AVerage** function has the purpose of validating the interval-valued entries of the output core tensors, and operates (similarly to the validation step of the ISVD procedure, discussed in 4.2) by replacing the wrongly ordered entries in each core tensor $\ddot{\mathcal{G}}_k$ with their average.

(MN) The **MiNimization** function is intended as a sort of optimization of the decomposition process, and has the purpose of reducing the number of entries of a factor matrix that need to be made scalar during the validation process described in the previous point (thus avoiding the interval information that

they carry to be lost in the averaging process). More in detail, let's 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's suppose that some of their entries are not valid intervals (i.e., their endpoints are misordered and would need to be replaced with their average). We can then define the following set of variables, one for each column vector k of $\ddot{\mathbf{U}}$ and $\ddot{\mathbf{V}}$, with $k = 1, \ldots, r$:

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

where the signs (either positive or negative) of Δ_k^U and Δ_k^V give us an idea of whether, in the validation process, we would save more of the interval information leaving the k column vectors as they are, or if it would be better to swap their signs.

As we already saw in Section 4.2, in fact, a property of the singular value decomposition is that we can swap the directions of a corresponding pair of left and right singular vectors without compromising the overall decomposition process[1]. However, precisely because each adjustment on $\ddot{\mathbf{U}}$ would also affect $\ddot{\mathbf{V}}$, we have to either prioritize one of the two, or reach a compromise. Specifically, for every $k = 1, \ldots, r$, we have the following three options:

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

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

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

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

3. **Reach a compromise between $\ddot{\mathbf{U}}$ and $\ddot{\mathbf{V}}$**:

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

[1] Specifically, for every $i = 1, \ldots, k$, it holds that $\sigma_i \cdot \mathbf{u}_i \cdot \mathbf{v}_i^{\mathrm{T}} \equiv \sigma_i \cdot (-\mathbf{u}_i) \cdot (-\mathbf{v}_i^{\mathrm{T}})$.

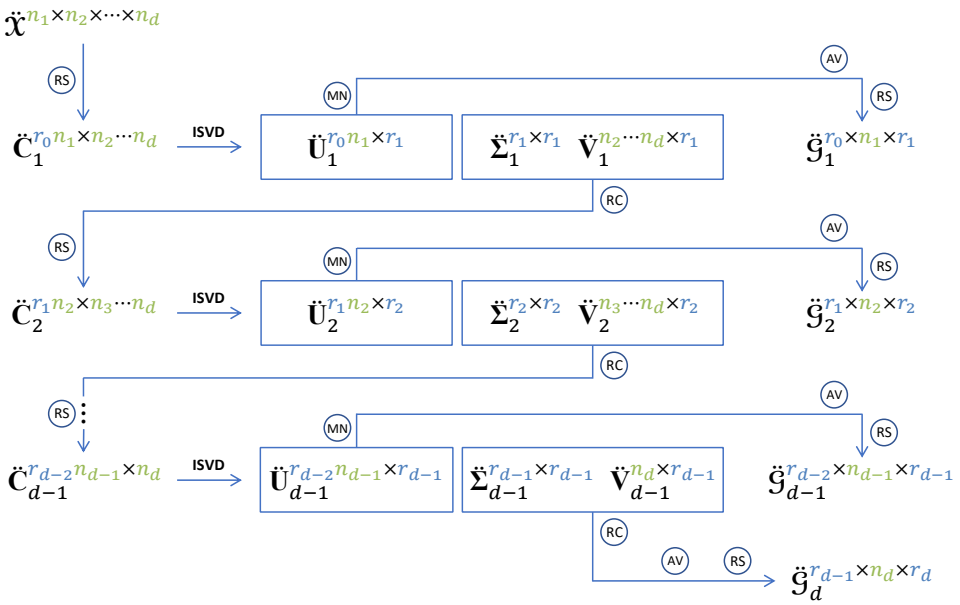


Figure 7.1: Interval Tensor-Train Decomposition (ITTD) of an n-mode interval-valued tensor. Colors help visualize the relationship between the input tensor sizes (in green) and the auxiliary matrices ranks (in blue) with the core tensors sizes.

The decision on which option to choose at each step of the decomposition is application-oriented and, specifically, it's related to whether we would like to prioritize one of the resulting cores over the others (maybe because we expect the bulk of the interval information to be collected in it), in which case we would prioritize for $\ddot{\mathbf{V}}$ (option 2) for each step that leads to its extraction, and lastly for $\ddot{\mathbf{U}}$ (option 1) when we actually extract it. If instead we presume that all the interval information would be equally spread over al the cores, the best choice is to try to lower the overall number of elements that need to be averaged between both $\ddot{\mathbf{U}}$ and $\ddot{\mathbf{V}}$ (option 3).

7.2 Interval-valued Tensor-Train Reconstruction

Figure 7.2 illustrates the Interval Tensor-Train reconstruction process, which, similarly to what happens in the scalar scenario (see Figure 6.6), takes the d cores $\ddot{\mathcal{G}}$ from the ITTD procedure and returns either the original tensor $\ddot{\mathcal{X}}$, or a close approximation, $\ddot{\widetilde{\mathcal{X}}}$, according to the accuracy of the initial decomposition (i.e., if the decomposition was either *full-rank* or *low-rank*).

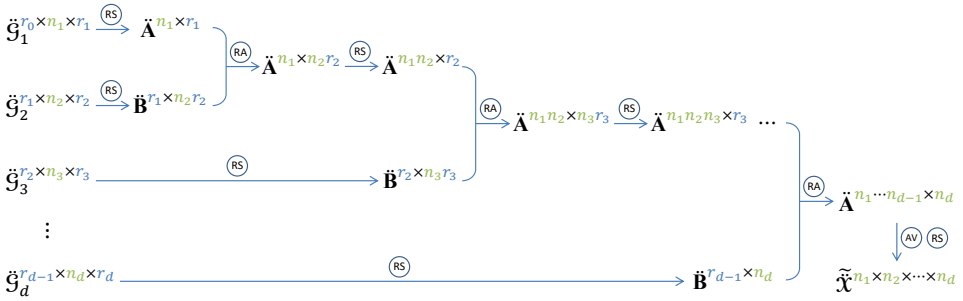


Figure 7.2: Interval Tensor-Train Reconstruction of an n-mode interval-valued tensor. Colors help visualize the relationship between the core tensors sizes (in green) and the auxiliary matrices ranks (in blue) with the output tensor sizes.

As for the decomposition, the interval-valued nature of the data requires the introduction of following additional operations:

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

(AV) The **AVerage** function, used for validating the interval-valued entries, is here 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 matrix product, and is equivalent to the **ReCombination** function of the ITTD procedure. Here, again, we have four options for the recombination 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) $\ddot{\mathbf{A}} := \overline{\mathbf{A}} \cdot \overline{\mathbf{B}}$.

7.3 Experiments on interval-valued tensor factorization

In order to validate our approach on the decomposition of interval-valued tensors, we have devised a set of experiments that evaluate the accuracy of the reconstruction of a d-dimensional tensor $\ddot{\mathcal{X}}$ from its cores $\ddot{\mathcal{G}}_1, \ddot{\mathcal{G}}_2, \ldots, \ddot{\mathcal{G}}_d$ obtained from the ITTD procedure.

In particular, we are interested in finding the **ReCombination** and **ReAssembling** assignments (at each step of the decomposition and reconstruction procedures, respectively) that lead to the best accuracy, i.e., that minimize the *harmonic mean* function[2] $\Theta_{\text{HM}}\left(\ddot{\mathcal{X}}, \tilde{\ddot{\mathcal{X}}}\right)$ between the input tensor $\ddot{\mathcal{X}}$ and its approximation $\tilde{\ddot{\mathcal{X}}}$.

From the experiments on interval-valued matrices, reported in Chapter 5, we can draw the following conclusions, that will help us setting up the test cases:

- the best strategy for decomposing an interval-valued matrix is, in general, ISVD_4 (detailed in Section 4.4.5), so we will rely on this method for the decomposition steps in the ITTD procedure (indicated as ISVD in Figure 7.1).
- When two interval-valued matrices need to be multiplied, it is better in general to avoid the interval-valued matrix product, and choose, if possible, a scalar product, where one of the two matrices is averaged. In light of this fact, in the following experiments we will not take into account options a and α in the **ReCombination** and **ReAssembling**, respectively.

The second point in the list affects, first of all, the reconstruction procedure: if we want to avoid the interval-valued product between two matrices, 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 Figure 7.2 can carry interval-valued variables at any time. From these consideration, 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. This strategy is summarized in Table 7.1, which shows, for each core tensor $\ddot{\mathcal{G}}_k$ obtained from the ITTD 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 trickle down the branches of the reconstruction procedure and end up in the final approximation of the initial tensor.

[2]Note that the harmonic mean function Θ_{HM}, presented in Section 4.1.2, can be readily extended from matrices to tensors considering that the Frobenius norm of a tensor is simply the extension of the one for a matrix to higher dimensions, namely, given a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_d}$,

$$\|\mathcal{X}\|_F = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_d=1}^{I_d} (x_{i_1,i_2,\ldots,i_d})^2}$$

Interval core	**ReAssembling step options**					
$\ddot{\mathcal{G}}_1$	β	β	β	$\cdots$	β	β
$\ddot{\mathcal{G}}_2$	γ	β	β	$\cdots$	β	β
$\ddot{\mathcal{G}}_3$	δ	γ	β	$\cdots$	β	β
$\vdots$				$\ddots$		
$\ddot{\mathcal{G}}_d$	δ	δ	δ	$\cdots$	δ	γ
	RA_1	RA_2	RA_3	$\cdots$	RA_{d-2}	RA_{d-3}

Table 7.1: **ReAssembling** options according to which core tensor is selected as the interval-valued one

Since in this formulation of the reconstruction procedure only one core tensor can be taken as interval-valued, it seems appropriate to make sure that already in the decomposition process most of the interval information from the original tensor will end up in that particular core. Our approach to facilitate this outcome is twofold:

1. First of all, we can make sure that the interval information propagates through each step in the decomposition by selecting an option for the **ReCombination** step that would keep either $\ddot{\mathbf{\Sigma}}_k$ or $\ddot{\mathbf{V}}$ as interval (i.e., option *b* or *c*, respectively). Once we reach the ISVD step that returns the $\ddot{\mathbf{U}}$ factor matrix that would be reshaped into the desired core tensor, then there is no need for the next steps to still process interval information, so, from then on, we can keep $\overline{\mathbf{\Sigma}}$ 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 7.2.

2. We can also leverage the **MiNimization** step 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 ITTD 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, then we *prioritize* $\ddot{\mathbf{U}}$ (so that, as explained in Section 7.1, 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

Interval core	ReCombination step options					
$\ddot{\mathcal{G}}_1$	d	d	d	$\cdots$	d	d
$\ddot{\mathcal{G}}_2$	b/c	d	d	$\cdots$	d	d
$\ddot{\mathcal{G}}_3$	b/c	b/c	d	$\cdots$	d	d
$\vdots$				$\ddots$		
$\ddot{\mathcal{G}}_d$	b/c	b/c	b/c	$\cdots$	b/c	b/c
	RC_1	RC_2	RC_3	$\cdots$	RC_{d-2}	RC_{d-3}

Table 7.2: **ReCombination** options in the ITTD procedure according to which core tensor we would want to keep as interval-valued in the reconstruction

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 $\ddot{\mathbf{V}}_{d-1}$).

7.3.1 Experiments on synthetic datasets

The experiments reported in this section have been devised with the purpose of finding the assignment (or assignments) for the **ReCombination** steps in the ITTD procedure that lead to the best approximation for a given input tensor, after its decomposition and reconstruction.

The dataset we relied on have been randomly generated, following a similar approach to the one discussed for the interval-valued matrices scenario (see Section 5.1.1). More in detail, the following parameters have been specified:

Tensor size: we considered 5-dimensional $8 \times 8 \times 8 \times 8 \times 8$ tensors (for a total of 32768 entries each), for which the decomposition and reconstruction processes (tailored for this particular size) are illustrated in Figures 7.3 and 7.4, respectively.

Interval density: this parameter is set to 50% (half of the entries in each tensor is set as interval valued).

Interval intensity: 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: at each ISVD step in the ITTD 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. Figure 7.5 illustrates this concept for a 3D $16 \times 16 \times 16$ 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 5D tensors, where the interval information can be localized along any of the five modes.

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 type of noise, as follows:

- **Random 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

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

- **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

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

  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 type of noise, we created 100 random tensors and the results presented in the following are the average of each corresponding run.

Random interval information experiments

Figure 7.6 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 results (averaged over 100 runs and color-coded from

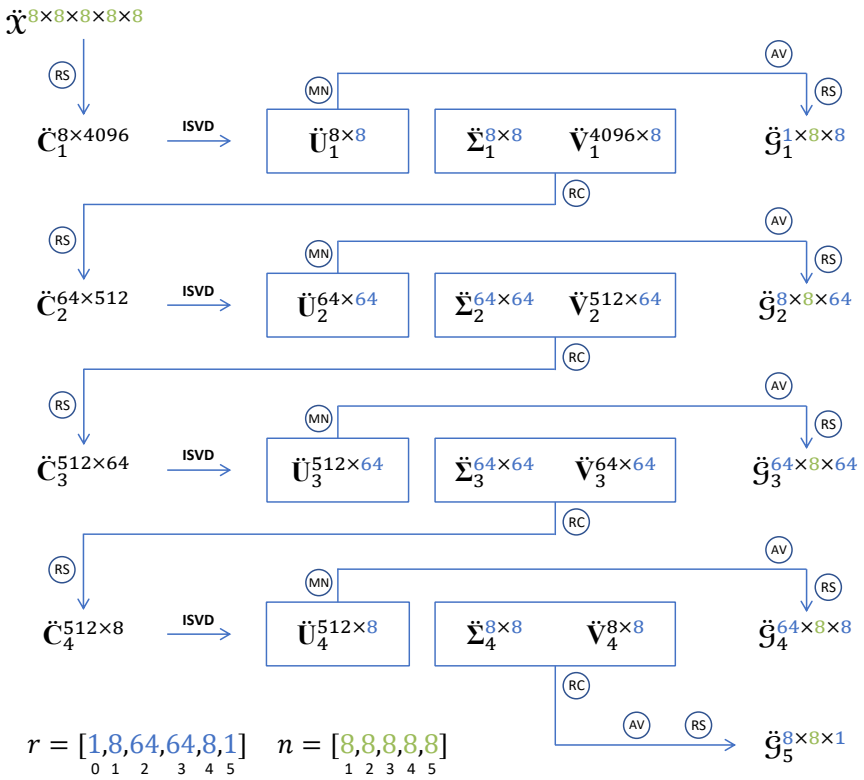


Figure 7.3: Instantiation of the Interval Tensor-Train Decomposition (ITTD) (see Figure 7.1) procedure for 5-dimensional ($8 \times 8 \times 8 \times 8 \times 8$) tensors. Colors help visualize the relationship between the input tensor sizes (in green) and the auxiliary matrices ranks (in blue) with the core tensors sizes. Notice also how the number of elements are substantially different from one core tensor to the other, with $\ddot{\mathcal{G}}_3$ being the largest by a significant margin.

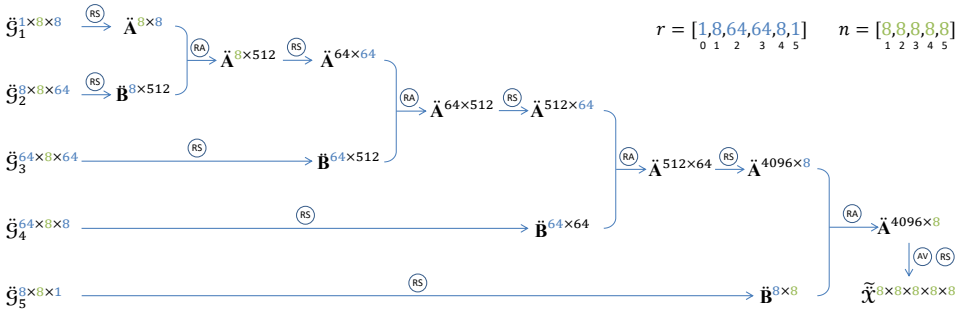
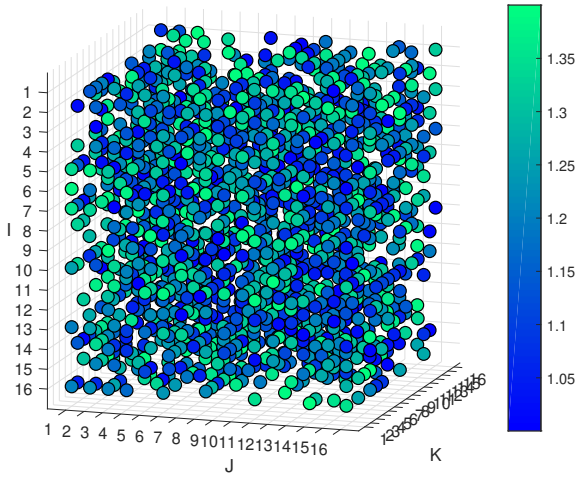
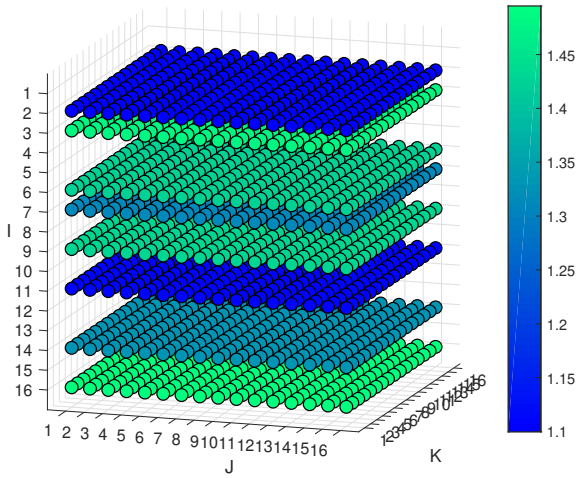


Figure 7.4: Instantiation of the Interval Tensor-Train Reconstruction procedure (see Figure 7.2) for 5-dimensional ($8 \times 8 \times 8 \times 8 \times 8$) tensors. Colors help visualize the relationship between the core tensors sizes (in green) and the auxiliary matrices ranks (in blue) with the output tensor sizes.



(a) Random placement of the interval information



(b) Localized placement of the interval information

Figure 7.5: Disposition of the interval information in a 3D tensor, represented as the ratio between maximum and minimum values

red, poor reconstruction, to green, best reconstruction) for each possible combination of the **ReCombination** options (*b*, *c* and *d*) at each of the four ISVD steps in the decomposition procedure (see Figure 7.3). Each decomposition scenario is then evaluated over five possible reconstruction schemes, according to which of the core tensors, $\ddot{\mathcal{G}}_1$ to $\ddot{\mathcal{G}}_5$, we want to keep as interval.

From these results we can conclude that:

- when the interval information is randomly placed in the tensors, 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, which seems to be related, intuitively, to the fact that (as we can see in Figure 7.4) 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;
- in terms of the **ReCombination** options, our strategy (summarized in Table 7.2) seems to hold true: 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 (in this case, option *c* is the best option, keeping $\ddot{\mathbf{V}}_k$ as interval-valued and $\ddot{\mathbf{\Sigma}}_k$ 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[3].

Localized interval information experiments

For this set of experiments we considered a synthetic dataset where the interval information is localized along any of the modes in the input tensors. Specifically, Figures 7.7 to 7.11 report the reconstruction accuracies (in terms of the average Harmonic Mean over 100 samples, with the results color-coded from red, poor reconstruction, to green, best reconstruction) for each set of experiments where the interval information is localized along mode *I* to *M*, respectively. We again search for the assignment of **ReCombination** options (*b*, *c* and *d*) at each of the four

[3]Notice that the first nine results in the column relative to $\ddot{\mathcal{X}}_3$ in figure 7.6 are equivalent, and they are just ordered alphabetically according to the recombination options.

$\mathcal{G}_1$ as interval		$\mathcal{G}_2$ as interval		$\mathcal{G}_3$ as interval		$\mathcal{G}_4$ as interval		$\mathcal{G}_5$ as interval	
Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options
0,924781737	[b b b b]	0,924903241	[b b b b]	0,938030943	[c c b b]	0,92440716	[b c b b]	0,924508243	[b b b c]
0,924781737	[b b b c]	0,924903241	[b b b c]	0,938030943	[c c b c]	0,92440716	[b c b c]	0,924507046	[b b b b]
0,924781737	[b b b d]	0,924903241	[b b b d]	0,938030943	[c c b d]	0,92440716	[b c b d]	0,924496331	[b c b c]
0,924781737	[b b c b]	0,924903241	[b b c b]	0,938030943	[c c c b]	0,924405153	[b c c b]	0,924495103	[b c b b]
0,924781737	[b b c c]	0,924903241	[b b c c]	0,938030943	[c c c c]	0,924405153	[b c c c]	0,924480652	[b c c c]
0,924781737	[b b c d]	0,924903241	[b b c d]	0,938030943	[c c c d]	0,924405153	[b c c d]	0,924479337	[b c c b]
0,924781737	[b b d b]	0,924903241	[b c b b]	0,938030943	[c c d b]	0,924383779	[b b c b]	0,924476022	[b b c c]
0,924781737	[b b d c]	0,924903241	[b c b c]	0,938030943	[c c d c]	0,924383779	[b b c c]	0,924474804	[b b c b]
0,924781737	[b b d d]	0,924903241	[b c b d]	0,938030943	[c c d d]	0,924383779	[b b c d]	0,921901021	[c b b c]
0,924781737	[b c b b]	0,924903241	[b c c b]	0,924370483	[b b b b]	0,924381144	[b b b b]	0,921899913	[c b b b]
0,924781737	[b c b c]	0,924903241	[b c c c]	0,924370483	[b b b c]	0,924381144	[b b b c]	0,921863653	[c b c c]
0,924781737	[b c b d]	0,924903241	[b c c d]	0,924370483	[b b b d]	0,924381144	[b b b d]	0,921862241	[c b c b]
0,924781737	[b c c b]	0,924903241	[b c d b]	0,924370483	[b b c b]	0,921832284	[b c b b]	0,918835383	[c c c c]
0,924781737	[b c c c]	0,924903241	[b c d c]	0,924370483	[b b c c]	0,921832284	[b c b c]	0,918804178	[c c c b]
0,924781737	[b c c d]	0,924903241	[b c d d]	0,924370483	[b b c d]	0,921832284	[b c b d]	0,9186875	[c c b c]
0,924781737	[b c d b]	0,924903241	[b d b b]	0,924370483	[b b d b]	0,92182547	[b b b b]	0,918685267	[c c b b]
0,924781737	[b c d c]	0,924903241	[b d b c]	0,924370483	[b b d c]	0,92182547	[b b b c]	0,904019975	[b b b d]
0,924781737	[b c d d]	0,924903241	[b d b d]	0,924370483	[b b d d]	0,92182547	[b b b d]	0,904019975	[b b c d]
0,924781737	[b d b b]	0,924903241	[b d c b]	0,924343108	[b c b b]	0,92011705	[b c c b]	0,904019975	[b b d b]
0,924781737	[b d b c]	0,924903241	[b d c c]	0,924343108	[b c b c]	0,92011705	[b c c c]	0,904019975	[b b d c]
0,924781737	[b d b d]	0,924903241	[b d c d]	0,924343108	[b c b d]	0,92011705	[b c c d]	0,904019975	[b b d d]
0,924781737	[b d c b]	0,924903241	[b d d b]	0,924343108	[b c c b]	0,918573575	[b c b d]	0,904019975	[b c b d]
0,924781737	[b d c c]	0,924903241	[b d d c]	0,924343108	[b c c c]	0,918573575	[b c b b]	0,904019975	[b c c d]
0,924781737	[b d c d]	0,924903241	[b d d d]	0,924343108	[b c c d]	0,918573575	[b c b c]	0,904019975	[b c d b]
0,924781737	[b d d b]	0,924903241	[b b d b]	0,924343108	[b c d b]	0,904019975	[b b d b]	0,904019975	[b c d c]
0,924781737	[b d d c]	0,924903241	[b b d c]	0,924343108	[b c d c]	0,904019975	[b b d c]	0,904019975	[b c d d]
0,924781737	[b d d d]	0,924903241	[b b d d]	0,924343108	[b c d d]	0,904019975	[b b d d]	0,904019975	[b d b b]
0,924781737	[d b b b]	0,924748989	[c b b b]	0,921724704	[c b b b]	0,904019975	[b c d b]	0,904019975	[b d b c]
0,924781737	[d b b c]	0,924748989	[c b b c]	0,921724704	[c b b c]	0,904019975	[b c d c]	0,904019975	[b d b d]
0,924781737	[d b b d]	0,924748989	[c b b d]	0,921724704	[c b b d]	0,904019975	[b c d d]	0,904019975	[b d c b]
0,924781737	[d b c b]	0,924748989	[c b c b]	0,921724704	[c b c b]	0,904019975	[b d b b]	0,904019975	[b d c c]
0,924781737	[d b c c]	0,924748989	[c b c c]	0,921724704	[c b c c]	0,904019975	[b d b c]	0,904019975	[b d c d]
0,924781737	[d b c d]	0,924748989	[c b c d]	0,921724704	[c b c d]	0,904019975	[b d b d]	0,904019975	[b d d b]
0,924781737	[d b d b]	0,924748989	[c b d b]	0,921724704	[c b d b]	0,904019975	[b d c b]	0,904019975	[b d d c]
0,924781737	[d b d c]	0,924748989	[c b d c]	0,921724704	[c b d c]	0,904019975	[b d c c]	0,904019975	[b d d d]
0,924781737	[d b d d]	0,924748989	[c b d d]	0,921724704	[c b d d]	0,904019975	[b d c d]	0,904019975	[d b b b]
0,924781737	[d c b b]	0,924748989	[c d b b]	0,904019975	[b d b b]	0,904019975	[b d d b]	0,904019975	[d b b c]
0,924781737	[d c b c]	0,924748989	[c d b c]	0,904019975	[b d b c]	0,904019975	[b d d c]	0,904019975	[d b b d]
0,924781737	[d c b d]	0,924748989	[c d b d]	0,904019975	[b d b d]	0,904019975	[b d d d]	0,904019975	[d b c b]
0,924781737	[d c c b]	0,924748989	[c d c b]	0,904019975	[b d c b]	0,904019975	[d b b b]	0,904019975	[d b c c]
0,924781737	[d c c c]	0,924748989	[c d c c]	0,904019975	[b d c c]	0,904019975	[d b b c]	0,904019975	[d b c d]
0,924781737	[d c c d]	0,924748989	[c d c d]	0,904019975	[b d c d]	0,904019975	[d b b d]	0,904019975	[d b d b]
0,924781737	[d c d b]	0,924748989	[c d d b]	0,904019975	[b d d b]	0,904019975	[d b c b]	0,904019975	[d b d c]
0,924781737	[d c d c]	0,924748989	[c d d c]	0,904019975	[b d d c]	0,904019975	[d b c c]	0,904019975	[d b d d]
0,924781737	[d c d d]	0,924748989	[c d d d]	0,904019975	[b d d d]	0,904019975	[d b c d]	0,904019975	[d c b b]
0,924781737	[d d b b]	0,92138904	[c c b b]	0,904019975	[d b b b]	0,904019975	[d b d b]	0,904019975	[d c b c]
0,924781737	[d d b c]	0,92138904	[c c b c]	0,904019975	[d b b c]	0,904019975	[d b d c]	0,904019975	[d c b d]
0,924781737	[d d b d]	0,92138904	[c c b d]	0,904019975	[d b b d]	0,904019975	[d b d d]	0,904019975	[d c c b]
0,924781737	[d d c b]	0,92138904	[c c c b]	0,904019975	[d b c b]	0,904019975	[d c b b]	0,904019975	[d c c c]
0,924781737	[d d c c]	0,92138904	[c c c c]	0,904019975	[d b c c]	0,904019975	[d c b c]	0,904019975	[d c c d]
0,924781737	[d d c d]	0,92138904	[c c c d]	0,904019975	[d b c d]	0,904019975	[d c b d]	0,904019975	[d c d b]
0,924781737	[d d d b]	0,92138904	[c c d b]	0,904019975	[d b d b]	0,904019975	[d c c b]	0,904019975	[d c d c]
0,924781737	[d d d c]	0,92138904	[c c d c]	0,904019975	[d b d c]	0,904019975	[d c c c]	0,904019975	[d c d d]
0,924781737	[d d d d]	0,92138904	[c c d d]	0,904019975	[d b d d]	0,904019975	[d c c d]	0,904019975	[d d b b]
0,922138002	[c b b b]	0,904019975	[d b b b]	0,904019975	[d c b b]	0,904019975	[d c d b]	0,904019975	[d d b c]
0,922138002	[c b b c]	0,904019975	[d b b c]	0,904019975	[d c b c]	0,904019975	[d c d c]	0,904019975	[d d b d]
0,922138002	[c b b d]	0,904019975	[d b b d]	0,904019975	[d c b d]	0,904019975	[d c d d]	0,904019975	[d d c b]
0,922138002	[c b c b]	0,904019975	[d b c b]	0,904019975	[d c c b]	0,904019975	[d d b b]	0,904019975	[d d c c]
0,922138002	[c b c c]	0,904019975	[d b c c]	0,904019975	[d c c c]	0,904019975	[d d b c]	0,904019975	[d d c d]
0,922138002	[c b c d]	0,904019975	[d b c d]	0,904019975	[d c c d]	0,904019975	[d d b d]	0,904019975	[d d d b]
0,922138002	[c b d b]	0,904019975	[d b d b]	0,904019975	[d c d b]	0,904019975	[d d c b]	0,904019975	[d d d c]
0,922138002	[c b d c]	0,904019975	[d b d c]	0,904019975	[d c d c]	0,904019975	[d d c c]	0,904019975	[d d d d]
0,922138002	[c b d d]	0,904019975	[d b d d]	0,904019975	[d c d d]	0,904019975	[d d c d]	0,90185101	[c b b d]
0,922138002	[c d b b]	0,904019975	[d c b b]	0,904019975	[d d b b]	0,904019975	[d d d b]	0,90185101	[c b c d]
0,922138002	[c d b c]	0,904019975	[d c b c]	0,904019975	[d d b c]	0,904019975	[d d d c]	0,90185101	[c b d b]
0,922138002	[c d b d]	0,904019975	[d c b d]	0,904019975	[d d b d]	0,904019975	[d d d d]	0,90185101	[c b d c]
0,922138002	[c d c b]	0,904019975	[d c c b]	0,904019975	[d d c b]	0,90185101	[c b d b]	0,90185101	[c b d d]
0,922138002	[c d c c]	0,904019975	[d c c c]	0,904019975	[d d c c]	0,90185101	[c b d c]	0,90185101	[c d b b]
0,922138002	[c d c d]	0,904019975	[d c c d]	0,904019975	[d d c d]	0,90185101	[c b d d]	0,90185101	[c d b c]
0,922138002	[c d d b]	0,904019975	[d c d b]	0,904019975	[d d d b]	0,90185101	[c d b b]	0,90185101	[c d b d]
0,922138002	[c d d c]	0,904019975	[d c d c]	0,904019975	[d d d c]	0,90185101	[c d b c]	0,90185101	[c d c b]
0,922138002	[c d d d]	0,904019975	[d c d d]	0,904019975	[d d d d]	0,90185101	[c d b d]	0,90185101	[c d c c]
0,918878016	[c c b b]	0,904019975	[d d b b]	0,90185101	[c d b b]	0,90185101	[c d c b]	0,90185101	[c d c d]
0,918878016	[c c b c]	0,904019975	[d d b c]	0,90185101	[c d b c]	0,90185101	[c d c c]	0,90185101	[c d d b]
0,918878016	[c c b d]	0,904019975	[d d b d]	0,90185101	[c d b d]	0,90185101	[c d c d]	0,90185101	[c d d c]
0,918878016	[c c c b]	0,904019975	[d d c b]	0,90185101	[c d c b]	0,90185101	[c d d b]	0,90185101	[c d d d]
0,918878016	[c c c c]	0,904019975	[d d c c]	0,90185101	[c d c c]	0,90185101	[c d d c]	0,899169549	[c c b d]
0,918878016	[c c c d]	0,904019975	[d d c d]	0,90185101	[c d c d]	0,90185101	[c d d d]	0,899169549	[c c c d]
0,918878016	[c c d b]	0,904019975	[d d d b]	0,90185101	[c d d b]	0,899169549	[c c d b]	0,899169549	[c c d b]
0,918878016	[c c d c]	0,904019975	[d d d c]	0,90185101	[c d d c]	0,899169549	[c c d c]	0,899169549	[c c d c]
0,918878016	[c c d d]	0,904019975	[d d d d]	0,90185101	[c d d d]	0,899169549	[c c d d]	0,899169549	[c c d d]

Figure 7.6: Reconstruction accuracies for tensors with random interval information (*the greener the cell, the better the result*, best option highlighted in light green – the tables are best viewed in color)

ISVD step in the decomposition procedure that leads to the highest reconstruction accuracy, highlighting the best (or one of the best, in case of a tie) set of assignments in light green. Finally, Figure 7.12 summarizes these results by comparing the best option from each 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 noise 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 ITTD procedure is once again confirmed to be reasonable, guaranteeing the best accuracy (or one of the best in case of tie) in each scenario, according to which core tensor we intend to keep as interval-valued;
- the results in Figure 7.12 imply that, whether possible, keeping the first core as interval-valued would be the overall optimal solution, 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 so that that specific mode becomes the one related to the first core resulting from the decomposition procedure.

$\mathcal{G}_1$ as interval		$\mathcal{G}_2$ as interval		$\mathcal{G}_3$ as interval		$\mathcal{G}_4$ as interval		$\mathcal{G}_5$ as interval	
Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options
0,982005205	[d b b b]	0,950402887	[c b b b]	0,94821447	[c c b b]	0,918920595	[b c c b]	0,919030144	[b c c c]
0,982005205	[d b b c]	0,950402887	[c b b c]	0,94821447	[c c b c]	0,918920595	[b c c c]	0,919028196	[b c c b]
0,982005205	[d b b d]	0,950402887	[c b b d]	0,94821447	[c c b d]	0,918920595	[b c c d]	0,919017594	[b b c c]
0,982005205	[d b c b]	0,950402887	[c b c b]	0,94821447	[c c c b]	0,918873405	[b b c b]	0,919016418	[b b c b]
0,982005205	[d b c c]	0,950402887	[c b c c]	0,94821447	[c c c c]	0,918873405	[b b c c]	0,918993395	[b c b c]
0,982005205	[d b c d]	0,950402887	[c b c d]	0,94821447	[c c c d]	0,918873405	[b b c d]	0,918992515	[b c b b]
0,982005205	[d b d b]	0,950402887	[c b d b]	0,94821447	[c c d b]	0,918857591	[b c b b]	0,918988367	[b b b c]
0,982005205	[d b d c]	0,950402887	[c b d c]	0,94821447	[c c d c]	0,918857591	[b c b c]	0,918987951	[b b b b]
0,982005205	[d b d d]	0,950402887	[c b d d]	0,94821447	[c c d d]	0,918857591	[b c b d]	0,916384701	[c c c b]
0,982005205	[d c b b]	0,950402887	[c c b b]	0,920338745	[b c b b]	0,918853161	[b b b b]	0,91636668	[c c c c]
0,982005205	[d c b c]	0,950402887	[c c b c]	0,920338745	[b c b c]	0,918853161	[b b b c]	0,916159922	[c b c c]
0,982005205	[d c b d]	0,950402887	[c c b d]	0,920338745	[b c b d]	0,918853161	[b b b d]	0,916159249	[c b c b]
0,982005205	[d c c b]	0,950402887	[c c c b]	0,920338745	[b c c b]	0,918481484	[c c c b]	0,916149953	[c b b c]
0,982005205	[d c c c]	0,950402887	[c c c c]	0,920338745	[b c c c]	0,918481484	[c c c c]	0,916149611	[c b b b]
0,982005205	[d c c d]	0,950402887	[c c c d]	0,920338745	[b c c d]	0,918481484	[c c c d]	0,916127289	[c c b b]
0,982005205	[d c d b]	0,950402887	[c c d b]	0,920338745	[b c d b]	0,916713412	[c c b b]	0,916127089	[c c b c]
0,982005205	[d c d c]	0,950402887	[c c d c]	0,920338745	[b c d c]	0,916713412	[c c b c]	0,906248657	[d b b b]
0,982005205	[d c d d]	0,950402887	[c c d d]	0,920338745	[b c d d]	0,916713412	[c c b d]	0,906248657	[d b b c]
0,982005205	[d d b b]	0,950402887	[c d b b]	0,918919732	[b b b b]	0,916016292	[c b b b]	0,906248657	[d b b d]
0,982005205	[d d b c]	0,950402887	[c d b c]	0,918919732	[b b b c]	0,916016292	[c b b c]	0,906248657	[d b c b]
0,982005205	[d d b d]	0,950402887	[c d b d]	0,918919732	[b b b d]	0,916016292	[c b b d]	0,906248657	[d b c c]
0,982005205	[d d c b]	0,950402887	[c d c b]	0,918919732	[b b c b]	0,916001679	[c b c b]	0,906248657	[d b c d]
0,982005205	[d d c c]	0,950402887	[c d c c]	0,918919732	[b b c c]	0,916001679	[c b c c]	0,906248657	[d b d b]
0,982005205	[d d c d]	0,950402887	[c d c d]	0,918919732	[b b c d]	0,916001679	[c b c d]	0,906248657	[d b d c]
0,982005205	[d d d b]	0,950402887	[c d d b]	0,918919732	[b b d b]	0,906248657	[d b b b]	0,906248657	[d b d d]
0,982005205	[d d d c]	0,950402887	[c d d c]	0,918919732	[b b d c]	0,906248657	[d b b c]	0,906248657	[d c b b]
0,982005205	[d d d d]	0,950402887	[c d d d]	0,918919732	[b b d d]	0,906248657	[d b b d]	0,906248657	[d c b c]
0,963362235	[b b b b]	0,922447982	[b b b b]	0,916469444	[c b b b]	0,906248657	[d b c b]	0,906248657	[d c b d]
0,963362235	[b b b c]	0,922447982	[b b b c]	0,916469444	[c b b c]	0,906248657	[d b c c]	0,906248657	[d c c b]
0,963362235	[b b b d]	0,922447982	[b b b d]	0,916469444	[c b b d]	0,906248657	[d b c d]	0,906248657	[d c c c]
0,963362235	[b b c b]	0,922447982	[b b c b]	0,916469444	[c b c b]	0,906248657	[d b d b]	0,906248657	[d c c d]
0,963362235	[b b c c]	0,922447982	[b b c c]	0,916469444	[c b c c]	0,906248657	[d b d c]	0,906248657	[d c d b]
0,963362235	[b b c d]	0,922447982	[b b c d]	0,916469444	[c b c d]	0,906248657	[d b d d]	0,906248657	[d c d c]
0,963362235	[b b d b]	0,922447982	[b b d b]	0,916469444	[c b d b]	0,906248657	[d c b b]	0,906248657	[d c d d]
0,963362235	[b b d c]	0,922447982	[b b d c]	0,916469444	[c b d c]	0,906248657	[d c b c]	0,906248657	[d d b b]
0,963362235	[b b d d]	0,922447982	[b b d d]	0,916469444	[c b d d]	0,906248657	[d c b d]	0,906248657	[d d b c]
0,963362235	[b c b b]	0,922447982	[b c b b]	0,906248657	[d b b b]	0,906248657	[d c c b]	0,906248657	[d d b d]
0,963362235	[b c b c]	0,922447982	[b c b c]	0,906248657	[d b b c]	0,906248657	[d c c c]	0,906248657	[d d c b]
0,963362235	[b c b d]	0,922447982	[b c b d]	0,906248657	[d b b d]	0,906248657	[d c c d]	0,906248657	[d d c c]
0,963362235	[b c c b]	0,922447982	[b c c b]	0,906248657	[d b c b]	0,906248657	[d c d b]	0,906248657	[d d c d]
0,963362235	[b c c c]	0,922447982	[b c c c]	0,906248657	[d b c c]	0,906248657	[d c d c]	0,906248657	[d d d b]
0,963362235	[b c c d]	0,922447982	[b c c d]	0,906248657	[d b c d]	0,906248657	[d c d d]	0,906248657	[d d d c]
0,963362235	[b c d b]	0,922447982	[b c d b]	0,906248657	[d b d b]	0,906248657	[d d b b]	0,906248657	[d d d d]
0,963362235	[b c d c]	0,922447982	[b c d c]	0,906248657	[d b d c]	0,906248657	[d d b c]	0,898473249	[b b b d]
0,963362235	[b c d d]	0,922447982	[b c d d]	0,906248657	[d b d d]	0,906248657	[d d b d]	0,898473249	[b b c d]
0,963362235	[b d b b]	0,922447982	[b d b b]	0,906248657	[d c b b]	0,906248657	[d d c b]	0,898473249	[b b d b]
0,963362235	[b d b c]	0,922447982	[b d b c]	0,906248657	[d c b c]	0,906248657	[d d c c]	0,898473249	[b b d c]
0,963362235	[b d b d]	0,922447982	[b d b d]	0,906248657	[d c b d]	0,906248657	[d d c d]	0,898473249	[b b d d]
0,963362235	[b d c b]	0,922447982	[b d c b]	0,906248657	[d c c b]	0,906248657	[d d d b]	0,898473249	[b c b d]
0,963362235	[b d c c]	0,922447982	[b d c c]	0,906248657	[d c c c]	0,906248657	[d d d c]	0,898473249	[b c c d]
0,963362235	[b d c d]	0,922447982	[b d c d]	0,906248657	[d c c d]	0,906248657	[d d d d]	0,898473249	[b c d b]
0,963362235	[b d d b]	0,922447982	[b d d b]	0,906248657	[d c d b]	0,898473249	[b b d b]	0,898473249	[b c d c]
0,963362235	[b d d c]	0,922447982	[b d d c]	0,906248657	[d c d c]	0,898473249	[b b d c]	0,898473249	[b c d d]
0,963362235	[b d d d]	0,922447982	[b d d d]	0,906248657	[d c d d]	0,898473249	[b b d d]	0,898473249	[b d b b]
0,957351605	[c b b b]	0,906248657	[d b b b]	0,906248657	[d d b b]	0,898473249	[b c b b]	0,898473249	[b d b c]
0,957351605	[c b b c]	0,906248657	[d b b c]	0,906248657	[d d b c]	0,898473249	[b c d c]	0,898473249	[b d b d]
0,957351605	[c b b d]	0,906248657	[d b b d]	0,906248657	[d d b d]	0,898473249	[b c d d]	0,898473249	[b d c b]
0,957351605	[c b c b]	0,906248657	[d b c b]	0,906248657	[d d c b]	0,898473249	[b d b b]	0,898473249	[b d c c]
0,957351605	[c b c c]	0,906248657	[d b c c]	0,906248657	[d d c c]	0,898473249	[b d b c]	0,898473249	[b d c d]
0,957351605	[c b c d]	0,906248657	[d b c d]	0,906248657	[d d c d]	0,898473249	[b d b d]	0,898473249	[b d d b]
0,957351605	[c b d b]	0,906248657	[d b d b]	0,906248657	[d d d b]	0,898473249	[b d c b]	0,898473249	[b d d c]
0,957351605	[c b d c]	0,906248657	[d b d c]	0,906248657	[d d d c]	0,898473249	[b d c c]	0,898473249	[b d d d]
0,957351605	[c b d d]	0,906248657	[d b d d]	0,906248657	[d d d d]	0,898473249	[b d c d]	0,896118892	[c b b d]
0,957351605	[c c b b]	0,906248657	[d c b b]	0,898473249	[b d b b]	0,898473249	[b d d b]	0,896118892	[c b c d]
0,957351605	[c c b c]	0,906248657	[d c b c]	0,898473249	[b d b c]	0,898473249	[b d d c]	0,896118892	[c b d b]
0,957351605	[c c b d]	0,906248657	[d c b d]	0,898473249	[b d b d]	0,898473249	[b d d d]	0,896118892	[c b d c]
0,957351605	[c c c b]	0,906248657	[d c c b]	0,898473249	[b d c b]	0,896118892	[c b d b]	0,896118892	[c b d d]
0,957351605	[c c c c]	0,906248657	[d c c c]	0,898473249	[b d c c]	0,896118892	[c b d c]	0,896118892	[c c b d]
0,957351605	[c c c d]	0,906248657	[d c c d]	0,898473249	[b d c d]	0,896118892	[c b d d]	0,896118892	[c c c d]
0,957351605	[c c d b]	0,906248657	[d c d b]	0,898473249	[b d d b]	0,896118892	[c c d b]	0,896118892	[c c d b]
0,957351605	[c c d c]	0,906248657	[d c d c]	0,898473249	[b d d c]	0,896118892	[c c d c]	0,896118892	[c c d c]
0,957351605	[c c d d]	0,906248657	[d c d d]	0,898473249	[b d d d]	0,896118892	[c c d d]	0,896118892	[c c d d]
0,957351605	[c d b b]	0,906248657	[d d b b]	0,896118892	[c d b b]	0,896118892	[c d b b]	0,896118892	[c d b b]
0,957351605	[c d b c]	0,906248657	[d d b c]	0,896118892	[c d b c]	0,896118892	[c d b c]	0,896118892	[c d b c]
0,957351605	[c d b d]	0,906248657	[d d b d]	0,896118892	[c d b d]	0,896118892	[c d b d]	0,896118892	[c d b d]
0,957351605	[c d c b]	0,906248657	[d d c b]	0,896118892	[c d c b]	0,896118892	[c d c b]	0,896118892	[c d c b]
0,957351605	[c d c c]	0,906248657	[d d c c]	0,896118892	[c d c c]	0,896118892	[c d c c]	0,896118892	[c d c c]
0,957351605	[c d c d]	0,906248657	[d d c d]	0,896118892	[c d c d]	0,896118892	[c d c d]	0,896118892	[c d c d]
0,957351605	[c d d b]	0,906248657	[d d d b]	0,896118892	[c d d b]	0,896118892	[c d d b]	0,896118892	[c d d b]
0,957351605	[c d d c]	0,906248657	[d d d c]	0,896118892	[c d d c]	0,896118892	[c d d c]	0,896118892	[c d d c]
0,957351605	[c d d d]	0,906248657	[d d d d]	0,896118892	[c d d d]	0,896118892	[c d d d]	0,896118892	[c d d d]

Figure 7.7: Reconstruction accuracies for tensors with interval information localized along the first mode, I (*the greener the cell, the better the result*, best option highlighted in light green – the tables are best viewed in color)

$\mathcal{G}_1$ as interval		$\mathcal{G}_2$ as interval		$\mathcal{G}_3$ as interval		$\mathcal{G}_4$ as interval		$\mathcal{G}_5$ as interval	
Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options
0,929048675	[b b b b]	0,93983634	[c b c b]	0,937622165	[c c b b]	0,92832208	[b b c b]	0,928612239	[b c c c]
0,929048675	[b b b c]	0,93983634	[c d b b]	0,937622165	[c c b c]	0,92832208	[b b c c]	0,928612239	[b c c b]
0,929048675	[b b b d]	0,93983634	[c d b c]	0,937622165	[c c b d]	0,92832208	[b b c d]	0,928590868	[b b c c]
0,929048675	[b b c b]	0,93983634	[c d b d]	0,937622165	[c c c b]	0,928321467	[b b b b]	0,92859086	[b b c b]
0,929048675	[b b c c]	0,93983634	[c d c b]	0,937622165	[c c c c]	0,928321467	[b b b c]	0,928589665	[b c b c]
0,929048675	[b b c d]	0,93983634	[c d c c]	0,937622165	[c c c d]	0,928321467	[b b b d]	0,928589663	[b c b b]
0,929048675	[b b d b]	0,93983634	[c d c d]	0,937622165	[c c d b]	0,928303009	[b c c b]	0,928589326	[b b b c]
0,929048675	[b b d c]	0,93983634	[c d d b]	0,937622165	[c c d c]	0,928303009	[b c c c]	0,928589322	[b b b b]
0,929048675	[b b d d]	0,93983634	[c d d c]	0,937622165	[c c d d]	0,928303009	[b c c d]	0,909665095	[c c c b]
0,929048675	[b c b b]	0,93983634	[c d d d]	0,92812364	[b b b b]	0,928301536	[b c b b]	0,909655419	[c b c c]
0,929048675	[b c b c]	0,93983634	[c b b b]	0,92812364	[b b b c]	0,928301536	[b c b c]	0,909654902	[c b c b]
0,929048675	[b c b d]	0,93983634	[c b b c]	0,92812364	[b b b d]	0,928301536	[b c b d]	0,909646424	[c c c c]
0,929048675	[b c c b]	0,93983634	[c b b d]	0,92812364	[b b c b]	0,911275853	[c c c b]	0,909605557	[c b b c]
0,929048675	[b c c c]	0,93983634	[c b c c]	0,92812364	[b b c c]	0,911275853	[c c c c]	0,909605152	[c b b b]
0,929048675	[b c c d]	0,93983634	[c b c d]	0,92812364	[b b c d]	0,911275853	[c c c d]	0,909455225	[c c b b]
0,929048675	[b c d b]	0,93983634	[c b d b]	0,92812364	[b b d b]	0,90992911	[c c b b]	0,909455007	[c c b c]
0,929048675	[b c d c]	0,93983634	[c b d c]	0,92812364	[b b d c]	0,90992911	[c c b c]	0,906248657	[b b b d]
0,929048675	[b c d d]	0,93983634	[c b d d]	0,92812364	[b b d d]	0,90992911	[c c b d]	0,906248657	[b b c d]
0,929048675	[b d b b]	0,939617233	[c c b b]	0,928121105	[b c b b]	0,909474205	[c b b b]	0,906248657	[b b d b]
0,929048675	[b d b c]	0,939617233	[c c b c]	0,928121105	[b c b c]	0,909474205	[c b b c]	0,906248657	[b b d c]
0,929048675	[b d b d]	0,939617233	[c c b d]	0,928121105	[b c b d]	0,909474205	[c b b d]	0,906248657	[b b d d]
0,929048675	[b d c b]	0,939617233	[c c c b]	0,928121105	[b c c b]	0,909461429	[c b c b]	0,906248657	[b c b d]
0,929048675	[b d c c]	0,939617233	[c c c c]	0,928121105	[b c c c]	0,909461429	[c b c c]	0,906248657	[b c c d]
0,929048675	[b d c d]	0,939617233	[c c c d]	0,928121105	[b c c d]	0,909461429	[c b c d]	0,906248657	[b c d b]
0,929048675	[b d d b]	0,939617233	[c c d b]	0,928121105	[b c d b]	0,906248657	[b b d b]	0,906248657	[b c d c]
0,929048675	[b d d c]	0,939617233	[c c d c]	0,928121105	[b c d c]	0,906248657	[b b d c]	0,906248657	[b c d d]
0,929048675	[b d d d]	0,939617233	[c c d d]	0,928121105	[b c d d]	0,906248657	[b b d d]	0,906248657	[b d b b]
0,929048675	[d b b b]	0,92907181	[b b b b]	0,90960417	[c b b b]	0,906248657	[b c b b]	0,906248657	[b d b c]
0,929048675	[d b b c]	0,92907181	[b b b c]	0,90960417	[c b b c]	0,906248657	[b c d c]	0,906248657	[b d b d]
0,929048675	[d b b d]	0,92907181	[b b b d]	0,90960417	[c b b d]	0,906248657	[b c d d]	0,906248657	[b d c b]
0,929048675	[d b c b]	0,92907181	[b b c b]	0,90960417	[c b c b]	0,906248657	[b d b b]	0,906248657	[b d c c]
0,929048675	[d b c c]	0,92907181	[b b c c]	0,90960417	[c b c c]	0,906248657	[b d b c]	0,906248657	[b d c d]
0,929048675	[d b c d]	0,92907181	[b b c d]	0,90960417	[c b c d]	0,906248657	[b d b d]	0,906248657	[b d d b]
0,929048675	[d b d b]	0,92907181	[b b d b]	0,90960417	[c b d b]	0,906248657	[b d c b]	0,906248657	[b d d c]
0,929048675	[d b d c]	0,92907181	[b b d c]	0,90960417	[c b d c]	0,906248657	[b d c c]	0,906248657	[b d d d]
0,929048675	[d b d d]	0,92907181	[b b d d]	0,90960417	[c b d d]	0,906248657	[b d c d]	0,906248657	[d b b b]
0,929048675	[d c b b]	0,92907181	[b c b b]	0,906248657	[b d b b]	0,906248657	[b d d b]	0,906248657	[d b b c]
0,929048675	[d c b c]	0,92907181	[b c b c]	0,906248657	[b d b c]	0,906248657	[b d d c]	0,906248657	[d b b d]
0,929048675	[d c b d]	0,92907181	[b c b d]	0,906248657	[b d b d]	0,906248657	[b d d d]	0,906248657	[d b c b]
0,929048675	[d c c b]	0,92907181	[b c c b]	0,906248657	[b d c b]	0,906248657	[d b b b]	0,906248657	[d b c c]
0,929048675	[d c c c]	0,92907181	[b c c c]	0,906248657	[b d c c]	0,906248657	[d b b c]	0,906248657	[d b c d]
0,929048675	[d c c d]	0,92907181	[b c c d]	0,906248657	[b d c d]	0,906248657	[d b b d]	0,906248657	[d b d b]
0,929048675	[d c d b]	0,92907181	[b c d b]	0,906248657	[b d d b]	0,906248657	[d b c b]	0,906248657	[d b d c]
0,929048675	[d c d c]	0,92907181	[b c d c]	0,906248657	[b d d c]	0,906248657	[d b c c]	0,906248657	[d b d d]
0,929048675	[d c d d]	0,92907181	[b c d d]	0,906248657	[b d d d]	0,906248657	[d b c d]	0,906248657	[d c b b]
0,929048675	[d d b b]	0,92907181	[b d b b]	0,906248657	[d b b b]	0,906248657	[d b d b]	0,906248657	[d c b c]
0,929048675	[d d b c]	0,92907181	[b d b c]	0,906248657	[d b b c]	0,906248657	[d b d c]	0,906248657	[d c b d]
0,929048675	[d d b d]	0,92907181	[b d b d]	0,906248657	[d b b d]	0,906248657	[d b d d]	0,906248657	[d c c b]
0,929048675	[d d c b]	0,92907181	[b d c b]	0,906248657	[d b c b]	0,906248657	[d c b b]	0,906248657	[d c c c]
0,929048675	[d d c c]	0,92907181	[b d c c]	0,906248657	[d b c c]	0,906248657	[d c b c]	0,906248657	[d c c d]
0,929048675	[d d c d]	0,92907181	[b d c d]	0,906248657	[d b c d]	0,906248657	[d c b d]	0,906248657	[d c d b]
0,929048675	[d d d b]	0,92907181	[b d d b]	0,906248657	[d b d b]	0,906248657	[d c c b]	0,906248657	[d c d c]
0,929048675	[d d d c]	0,92907181	[b d d c]	0,906248657	[d b d c]	0,906248657	[d c c c]	0,906248657	[d c d d]
0,929048675	[d d d d]	0,92907181	[b d d d]	0,906248657	[d b d d]	0,906248657	[d c c d]	0,906248657	[d d b b]
0,909947618	[c b b b]	0,906248657	[d b b b]	0,906248657	[d c b b]	0,906248657	[d c d b]	0,906248657	[d d b c]
0,909947618	[c b b c]	0,906248657	[d b b c]	0,906248657	[d c b c]	0,906248657	[d c d c]	0,906248657	[d d b d]
0,909947618	[c b b d]	0,906248657	[d b b d]	0,906248657	[d c b d]	0,906248657	[d c d d]	0,906248657	[d d c b]
0,909947618	[c b c b]	0,906248657	[d b c b]	0,906248657	[d c c b]	0,906248657	[d d b b]	0,906248657	[d d c c]
0,909947618	[c b c c]	0,906248657	[d b c c]	0,906248657	[d c c c]	0,906248657	[d d b c]	0,906248657	[d d c d]
0,909947618	[c b c d]	0,906248657	[d b c d]	0,906248657	[d c c d]	0,906248657	[d d b d]	0,906248657	[d d d b]
0,909947618	[c b d b]	0,906248657	[d b d b]	0,906248657	[d c d b]	0,906248657	[d d c b]	0,906248657	[d d d c]
0,909947618	[c b d c]	0,906248657	[d b d c]	0,906248657	[d c d c]	0,906248657	[d d c c]	0,906248657	[d d d d]
0,909947618	[c b d d]	0,906248657	[d b d d]	0,906248657	[d c d d]	0,906248657	[d d c d]	0,890691925	[c b c d]
0,909947618	[c d b b]	0,906248657	[d c b b]	0,906248657	[d d b b]	0,906248657	[d d d b]	0,890691925	[c b b d]
0,909947618	[c d b c]	0,906248657	[d c b c]	0,906248657	[d d b c]	0,906248657	[d d d c]	0,890691925	[c b d b]
0,909947618	[c d b d]	0,906248657	[d c b d]	0,906248657	[d d b d]	0,906248657	[d d d d]	0,890691925	[c b d c]
0,909947618	[c d c b]	0,906248657	[d c c b]	0,906248657	[d d c b]	0,890691925	[c b d b]	0,890691925	[c b d d]
0,909947618	[c d c c]	0,906248657	[d c c c]	0,906248657	[d d c c]	0,890691925	[c b d c]	0,890691925	[c d b b]
0,909947618	[c d c d]	0,906248657	[d c c d]	0,906248657	[d d c d]	0,890691925	[c b d d]	0,890691925	[c d b c]
0,909947618	[c d d b]	0,906248657	[d c d b]	0,906248657	[d d d b]	0,890691925	[c d b b]	0,890691925	[c d b d]
0,909947618	[c d d c]	0,906248657	[d c d c]	0,906248657	[d d d c]	0,890691925	[c d b c]	0,890691925	[c d c b]
0,909947618	[c d d d]	0,906248657	[d c d d]	0,906248657	[d d d d]	0,890691925	[c d b d]	0,890691925	[c d c c]
0,909772758	[c c b b]	0,906248657	[d d b b]	0,890691925	[c d b b]	0,890691925	[c d c b]	0,890691925	[c d c d]
0,909772758	[c c b c]	0,906248657	[d d b c]	0,890691925	[c d b c]	0,890691925	[c d c c]	0,890691925	[c d d b]
0,909772758	[c c b d]	0,906248657	[d d b d]	0,890691925	[c d b d]	0,890691925	[c d c d]	0,890691925	[c d d c]
0,909772758	[c c c b]	0,906248657	[d d c b]	0,890691925	[c d c b]	0,890691925	[c d d b]	0,890691925	[c d d d]
0,909772758	[c c c c]	0,906248657	[d d c c]	0,890691925	[c d c c]	0,890691925	[c d d c]	0,890533581	[c c b d]
0,909772758	[c c c d]	0,906248657	[d d c d]	0,890691925	[c d c d]	0,890691925	[c d d d]	0,890533581	[c c c d]
0,909772758	[c c d b]	0,906248657	[d d d b]	0,890691925	[c d d b]	0,890533581	[c c d b]	0,890533581	[c c d b]
0,909772758	[c c d c]	0,906248657	[d d d c]	0,890691925	[c d d c]	0,890533581	[c c d c]	0,890533581	[c c d c]
0,909772758	[c c d d]	0,906248657	[d d d d]	0,890691925	[c d d d]	0,890533581	[c c d d]	0,890533581	[c c d d]

Figure 7.8: Reconstruction accuracies for tensors with interval information localized along the second mode, J (*the greener the cell, the better the result*, best option highlighted in light green – the tables are best viewed in color)

$\mathcal{G}_1$ as interval		$\mathcal{G}_2$ as interval		$\mathcal{G}_3$ as interval		$\mathcal{G}_4$ as interval		$\mathcal{G}_5$ as interval	
Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options
0,929048675	[c d b b]	0,933109207	[c b b b]	0,969000273	[c c b b]	0,932627566	[c c c b]	0,928994244	[c c c c]
0,929048675	[c d b c]	0,933109207	[c b b c]	0,969000273	[c c b c]	0,932627566	[c c c c]	0,928964615	[c c c b]
0,929048675	[c d b d]	0,933109207	[c b b d]	0,969000273	[c c b d]	0,932627566	[c c c d]	0,92861358	[b c c c]
0,929048675	[c d c b]	0,933109207	[c b c b]	0,969000273	[c c c b]	0,929375888	[c c b b]	0,928613579	[b c c b]
0,929048675	[c d c c]	0,933109207	[c b c c]	0,969000273	[c c c c]	0,929375888	[c c b c]	0,928608405	[c c b c]
0,929048675	[c d c d]	0,933109207	[c b c d]	0,969000273	[c c c d]	0,929375888	[c c b d]	0,928608004	[c c b b]
0,929048675	[c d d b]	0,933109207	[c b d b]	0,969000273	[c c d b]	0,928446671	[c b c b]	0,928599471	[c b c c]
0,929048675	[c d d c]	0,933109207	[c b d c]	0,969000273	[c c d c]	0,928446671	[c b c c]	0,928599254	[c b c b]
0,929048675	[c d d d]	0,933109207	[c b d d]	0,969000273	[c c d d]	0,928446671	[c b c d]	0,928591507	[b b b c]
0,929048675	[b b b b]	0,933109207	[c c c b]	0,92853121	[c b b b]	0,928437629	[c b b b]	0,928591504	[b b b b]
0,929048675	[b b b c]	0,933109207	[c c c c]	0,92853121	[c b b c]	0,928437629	[c b b c]	0,928589531	[b c b c]
0,929048675	[b b b d]	0,933109207	[c c c d]	0,92853121	[c b b d]	0,928437629	[c b b d]	0,92858953	[b c b b]
0,929048675	[b b c b]	0,933109207	[c c d b]	0,92853121	[c b c b]	0,92831276	[b c c b]	0,928588432	[b b c c]
0,929048675	[b b c c]	0,933109207	[c c d c]	0,92853121	[c b c c]	0,92831276	[b c c c]	0,928588427	[b b c b]
0,929048675	[b b c d]	0,933109207	[c c d d]	0,92853121	[c b c d]	0,92831276	[b c c d]	0,928576612	[c b b c]
0,929048675	[b b d b]	0,933109207	[c d b b]	0,92853121	[c b d b]	0,928311914	[b c b b]	0,928576598	[c b b b]
0,929048675	[b b d c]	0,933109207	[c d b c]	0,92853121	[c b d c]	0,928311914	[b c b c]	0,906248657	[b b b d]
0,929048675	[b b d d]	0,933109207	[c d b d]	0,92853121	[c b d d]	0,928311914	[b c b d]	0,906248657	[b b c d]
0,929048675	[c c b b]	0,933109207	[c d c b]	0,928472463	[b c b b]	0,928311524	[b b c b]	0,906248657	[b b d b]
0,929048675	[c c b c]	0,933109207	[c d c c]	0,928472463	[b c b c]	0,928311524	[b b c c]	0,906248657	[b b d c]
0,929048675	[c c b d]	0,933109207	[c d c d]	0,928472463	[b c b d]	0,928311524	[b b c d]	0,906248657	[b b d d]
0,929048675	[c c c c]	0,933109207	[c d d b]	0,928472463	[b c c b]	0,9283105	[b b b b]	0,906248657	[b c b d]
0,929048675	[c c d b]	0,933109207	[c d d c]	0,928472463	[b c c c]	0,9283105	[b b b c]	0,906248657	[b c c d]
0,929048675	[c c d c]	0,933109207	[c d d d]	0,928472463	[b c c d]	0,9283105	[b b b d]	0,906248657	[b c d b]
0,929048675	[c c d d]	0,933109207	[c c b b]	0,928472463	[b c d b]	0,906248657	[b b d b]	0,906248657	[b c d c]
0,929048675	[d b b b]	0,933109207	[c c b c]	0,928472463	[b c d c]	0,906248657	[b b d c]	0,906248657	[b c d d]
0,929048675	[d b b c]	0,933109207	[c c b d]	0,928472463	[b c d d]	0,906248657	[b b d d]	0,906248657	[b d b b]
0,929048675	[d b b d]	0,929071724	[b b b b]	0,928471211	[b b b b]	0,906248657	[b c d b]	0,906248657	[b d b c]
0,929048675	[d b c b]	0,929071724	[b b b c]	0,928471211	[b b b c]	0,906248657	[b c d c]	0,906248657	[b d b d]
0,929048675	[d b c c]	0,929071724	[b b b d]	0,928471211	[b b b d]	0,906248657	[b c d d]	0,906248657	[b d c b]
0,929048675	[d b c d]	0,929071724	[b b c b]	0,928471211	[b b c b]	0,906248657	[b d b b]	0,906248657	[b d c c]
0,929048675	[d b d b]	0,929071724	[b b c c]	0,928471211	[b b c c]	0,906248657	[b d b c]	0,906248657	[b d c d]
0,929048675	[d b d c]	0,929071724	[b b c d]	0,928471211	[b b c d]	0,906248657	[b d b d]	0,906248657	[b d d b]
0,929048675	[d b d d]	0,929071724	[b b d b]	0,928471211	[b b d b]	0,906248657	[b d c b]	0,906248657	[b d d c]
0,929048675	[d c b b]	0,929071724	[b b d c]	0,928471211	[b b d c]	0,906248657	[b d c c]	0,906248657	[b d d d]
0,929048675	[d c b c]	0,929071724	[b b d d]	0,928471211	[b b d d]	0,906248657	[b d c d]	0,906248657	[c b b d]
0,929048675	[d c b d]	0,929071724	[b d b b]	0,906248657	[b d b b]	0,906248657	[b d d b]	0,906248657	[c b c d]
0,929048675	[d c c b]	0,929071724	[b d b c]	0,906248657	[b d b c]	0,906248657	[b d d c]	0,906248657	[c b d b]
0,929048675	[d c c c]	0,929071724	[b d b d]	0,906248657	[b d b d]	0,906248657	[b d d d]	0,906248657	[c b d c]
0,929048675	[d c c d]	0,929071724	[b d c b]	0,906248657	[b d c b]	0,906248657	[c b d b]	0,906248657	[c b d d]
0,929048675	[d c d b]	0,929071724	[b d c c]	0,906248657	[b d c c]	0,906248657	[c b d c]	0,906248657	[c c b d]
0,929048675	[d c d c]	0,929071724	[b d c d]	0,906248657	[b d c d]	0,906248657	[c b d d]	0,906248657	[c c c d]
0,929048675	[d c d d]	0,929071724	[b d d b]	0,906248657	[b d d b]	0,906248657	[c c d b]	0,906248657	[c c d b]
0,929048675	[d d b b]	0,929071724	[b d d c]	0,906248657	[b d d c]	0,906248657	[c c d c]	0,906248657	[c c d c]
0,929048675	[d d b c]	0,929071724	[b d d d]	0,906248657	[b d d d]	0,906248657	[c c d d]	0,906248657	[c c d d]
0,929048675	[d d b d]	0,929071724	[b c d b]	0,906248657	[c d b b]	0,906248657	[c d b b]	0,906248657	[c d b b]
0,929048675	[d d c b]	0,929071724	[b c d c]	0,906248657	[c d b c]	0,906248657	[c d b c]	0,906248657	[c d b c]
0,929048675	[d d c c]	0,929071724	[b c d d]	0,906248657	[c d b d]	0,906248657	[c d b d]	0,906248657	[c d b d]
0,929048675	[d d c d]	0,929071724	[b c b b]	0,906248657	[c d c b]	0,906248657	[c d c b]	0,906248657	[c d c b]
0,929048675	[d d d b]	0,929071724	[b c b c]	0,906248657	[c d c c]	0,906248657	[c d c c]	0,906248657	[c d c c]
0,929048675	[d d d c]	0,929071724	[b c b d]	0,906248657	[c d c d]	0,906248657	[c d c d]	0,906248657	[c d c d]
0,929048675	[d d d d]	0,929071724	[b c c b]	0,906248657	[c d d b]	0,906248657	[c d d b]	0,906248657	[c d d b]
0,929048675	[b c b b]	0,929071724	[b c c c]	0,906248657	[c d d c]	0,906248657	[c d d c]	0,906248657	[c d d c]
0,929048675	[b c b c]	0,929071724	[b c c d]	0,906248657	[c d d d]	0,906248657	[c d d d]	0,906248657	[c d d d]
0,929048675	[b c b d]	0,906248657	[d b b b]	0,906248657	[d b b b]	0,906248657	[d b b b]	0,906248657	[d b b b]
0,929048675	[b c c b]	0,906248657	[d b b c]	0,906248657	[d b b c]	0,906248657	[d b b c]	0,906248657	[d b b c]
0,929048675	[b c c c]	0,906248657	[d b b d]	0,906248657	[d b b d]	0,906248657	[d b b d]	0,906248657	[d b b d]
0,929048675	[b c c d]	0,906248657	[d b c b]	0,906248657	[d b c b]	0,906248657	[d b c b]	0,906248657	[d b c b]
0,929048675	[b c d b]	0,906248657	[d b c c]	0,906248657	[d b c c]	0,906248657	[d b c c]	0,906248657	[d b c c]
0,929048675	[b c d c]	0,906248657	[d b c d]	0,906248657	[d b c d]	0,906248657	[d b c d]	0,906248657	[d b c d]
0,929048675	[b c d d]	0,906248657	[d b d b]	0,906248657	[d b d b]	0,906248657	[d b d b]	0,906248657	[d b d b]
0,929048675	[b d b b]	0,906248657	[d b d c]	0,906248657	[d b d c]	0,906248657	[d b d c]	0,906248657	[d b d c]
0,929048675	[b d b c]	0,906248657	[d b d d]	0,906248657	[d b d d]	0,906248657	[d b d d]	0,906248657	[d b d d]
0,929048675	[b d b d]	0,906248657	[d c b b]	0,906248657	[d c b b]	0,906248657	[d c b b]	0,906248657	[d c b b]
0,929048675	[b d c b]	0,906248657	[d c b c]	0,906248657	[d c b c]	0,906248657	[d c b c]	0,906248657	[d c b c]
0,929048675	[b d c c]	0,906248657	[d c b d]	0,906248657	[d c b d]	0,906248657	[d c b d]	0,906248657	[d c b d]
0,929048675	[b d c d]	0,906248657	[d c c b]	0,906248657	[d c c b]	0,906248657	[d c c b]	0,906248657	[d c c b]
0,929048675	[b d d b]	0,906248657	[d c c c]	0,906248657	[d c c c]	0,906248657	[d c c c]	0,906248657	[d c c c]
0,929048675	[b d d c]	0,906248657	[d c c d]	0,906248657	[d c c d]	0,906248657	[d c c d]	0,906248657	[d c c d]
0,929048675	[b d d d]	0,906248657	[d c d b]	0,906248657	[d c d b]	0,906248657	[d c d b]	0,906248657	[d c d b]
0,929048675	[c b b b]	0,906248657	[d c d c]	0,906248657	[d c d c]	0,906248657	[d c d c]	0,906248657	[d c d c]
0,929048675	[c b b c]	0,906248657	[d c d d]	0,906248657	[d c d d]	0,906248657	[d c d d]	0,906248657	[d c d d]
0,929048675	[c b b d]	0,906248657	[d d b b]	0,906248657	[d d b b]	0,906248657	[d d b b]	0,906248657	[d d b b]
0,929048675	[c b c b]	0,906248657	[d d b c]	0,906248657	[d d b c]	0,906248657	[d d b c]	0,906248657	[d d b c]
0,929048675	[c b c c]	0,906248657	[d d b d]	0,906248657	[d d b d]	0,906248657	[d d b d]	0,906248657	[d d b d]
0,929048675	[c b c d]	0,906248657	[d d c b]	0,906248657	[d d c b]	0,906248657	[d d c b]	0,906248657	[d d c b]
0,929048675	[c b d b]	0,906248657	[d d c c]	0,906248657	[d d c c]	0,906248657	[d d c c]	0,906248657	[d d c c]
0,929048675	[c b d c]	0,906248657	[d d c d]	0,906248657	[d d c d]	0,906248657	[d d c d]	0,906248657	[d d c d]
0,929048675	[c b d d]	0,906248657	[d d d b]	0,906248657	[d d d b]	0,906248657	[d d d b]	0,906248657	[d d d b]
0,929048675	[c c c b]	0,906248657	[d d d c]	0,906248657	[d d d c]	0,906248657	[d d d c]	0,906248657	[d d d c]
0,929048675	[c c c d]	0,906248657	[d d d d]	0,906248657	[d d d d]	0,906248657	[d d d d]	0,906248657	[d d d d]

Figure 7.9: Reconstruction accuracies for tensors with interval information localized along the third mode, K (*the greener the cell, the better the result*, best option highlighted in light green – the tables are best viewed in color)

$\mathcal{G}_1$ as interval		$\mathcal{G}_2$ as interval		$\mathcal{G}_3$ as interval		$\mathcal{G}_4$ as interval		$\mathcal{G}_5$ as interval	
Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options
0,929048675	[b c b b]	0,933109207	[c b b b]	0,944289019	[c c b b]	0,944128581	[c c c b]	0,928617234	[b c c c]
0,929048675	[b c b c]	0,933109207	[c b b c]	0,944289019	[c c b c]	0,944128581	[c c c c]	0,928617231	[b c c b]
0,929048675	[b c b d]	0,933109207	[c b b d]	0,944289019	[c c b d]	0,944128581	[c c c d]	0,928599454	[c b c c]
0,929048675	[b c c b]	0,933109207	[c d b b]	0,944289019	[c c c b]	0,928259941	[c b c b]	0,928599247	[c b c b]
0,929048675	[b c c c]	0,933109207	[c d b c]	0,944289019	[c c c c]	0,928259941	[c b c c]	0,928595043	[b b b c]
0,929048675	[b c c d]	0,933109207	[c d b d]	0,944289019	[c c c d]	0,928259941	[c b c d]	0,92859504	[b b b b]
0,929048675	[b c d b]	0,933109207	[c d c b]	0,944289019	[c c d b]	0,928242171	[b c c b]	0,928592673	[b c b c]
0,929048675	[b c d c]	0,933109207	[c d c c]	0,944289019	[c c d c]	0,928242171	[b c c c]	0,928592664	[b c b b]
0,929048675	[b c d d]	0,933109207	[c d c d]	0,944289019	[c c d d]	0,928242171	[b c c d]	0,928589173	[b b c c]
0,929048675	[d b b b]	0,933109207	[c d d b]	0,929193044	[c b b b]	0,928240483	[b c b b]	0,928589168	[b b c b]
0,929048675	[d b b c]	0,933109207	[c d d c]	0,929193044	[c b b c]	0,928240483	[b c b c]	0,928584066	[c b b c]
0,929048675	[d b b d]	0,933109207	[c d d d]	0,929193044	[c b b d]	0,928240483	[b c b d]	0,928583959	[c b b b]
0,929048675	[d b c b]	0,933109207	[c b c b]	0,929193044	[c b c b]	0,928234019	[c b b b]	0,913732209	[c c c b]
0,929048675	[d b c c]	0,933109207	[c b c c]	0,929193044	[c b c c]	0,928234019	[c b b c]	0,91365019	[c c c c]
0,929048675	[d b c d]	0,933109207	[c b c d]	0,929193044	[c b c d]	0,928234019	[c b b d]	0,913610689	[c c b b]
0,929048675	[d b d b]	0,933109207	[c b d b]	0,929193044	[c b d b]	0,928168543	[b b c b]	0,913595907	[c c b c]
0,929048675	[d b d c]	0,933109207	[c b d c]	0,929193044	[c b d c]	0,928168543	[b b c c]	0,906248657	[b b b d]
0,929048675	[d b d d]	0,933109207	[c b d d]	0,929193044	[c b d d]	0,928168543	[b b c d]	0,906248657	[b b c d]
0,929048675	[d c b b]	0,929071724	[b b b b]	0,928030283	[b c b b]	0,928168408	[b b b b]	0,906248657	[b b d b]
0,929048675	[d c b c]	0,929071724	[b b b c]	0,928030283	[b c b c]	0,928168408	[b b b c]	0,906248657	[b b d c]
0,929048675	[d c b d]	0,929071724	[b b b d]	0,928030283	[b c b d]	0,928168408	[b b b d]	0,906248657	[b b d d]
0,929048675	[d c c b]	0,929071724	[b b c b]	0,928030283	[b c c b]	0,91345514	[c c b b]	0,906248657	[b c b d]
0,929048675	[d c c c]	0,929071724	[b b c c]	0,928030283	[b c c c]	0,91345514	[c c b c]	0,906248657	[b c c d]
0,929048675	[d c c d]	0,929071724	[b b c d]	0,928030283	[b c c d]	0,91345514	[c c b d]	0,906248657	[b c d b]
0,929048675	[d c d b]	0,929071724	[b b d b]	0,928030283	[b c d b]	0,906248657	[b b d b]	0,906248657	[b c d c]
0,929048675	[d c d c]	0,929071724	[b b d c]	0,928030283	[b c d c]	0,906248657	[b b d c]	0,906248657	[b c d d]
0,929048675	[d c d d]	0,929071724	[b b d d]	0,928030283	[b c d d]	0,906248657	[b b d d]	0,906248657	[b d b b]
0,929048675	[d d b b]	0,929071724	[b c b b]	0,928030134	[b b b b]	0,906248657	[b c b b]	0,906248657	[b d b c]
0,929048675	[d d b c]	0,929071724	[b c b c]	0,928030134	[b b b c]	0,906248657	[b c d c]	0,906248657	[b d b d]
0,929048675	[d d b d]	0,929071724	[b c b d]	0,928030134	[b b b d]	0,906248657	[b c d d]	0,906248657	[b d c b]
0,929048675	[d d c b]	0,929071724	[b c c b]	0,928030134	[b b c b]	0,906248657	[b d b b]	0,906248657	[b d c c]
0,929048675	[d d c c]	0,929071724	[b c c c]	0,928030134	[b b c c]	0,906248657	[b d b c]	0,906248657	[b d c d]
0,929048675	[d d c d]	0,929071724	[b c c d]	0,928030134	[b b c d]	0,906248657	[b d b d]	0,906248657	[b d d b]
0,929048675	[d d d b]	0,929071724	[b c d b]	0,928030134	[b b d b]	0,906248657	[b d c b]	0,906248657	[b d d c]
0,929048675	[d d d c]	0,929071724	[b c d c]	0,928030134	[b b d c]	0,906248657	[b d c c]	0,906248657	[b d d d]
0,929048675	[d d d d]	0,929071724	[b c d d]	0,928030134	[b b d d]	0,906248657	[b d c d]	0,906248657	[c b b d]
0,929048675	[b b b b]	0,929071724	[b d b b]	0,906248657	[b d b b]	0,906248657	[b d d b]	0,906248657	[c b c d]
0,929048675	[b b b c]	0,929071724	[b d b c]	0,906248657	[b d b c]	0,906248657	[b d d c]	0,906248657	[c b d b]
0,929048675	[b b b d]	0,929071724	[b d b d]	0,906248657	[b d b d]	0,906248657	[b d d d]	0,906248657	[c b d c]
0,929048675	[b b c b]	0,929071724	[b d c b]	0,906248657	[b d c b]	0,906248657	[c b d b]	0,906248657	[c b d d]
0,929048675	[b b c c]	0,929071724	[b d c c]	0,906248657	[b d c c]	0,906248657	[c b d c]	0,906248657	[c d b b]
0,929048675	[b b c d]	0,929071724	[b d c d]	0,906248657	[b d c d]	0,906248657	[c b d d]	0,906248657	[c d b c]
0,929048675	[b b d b]	0,929071724	[b d d b]	0,906248657	[b d d b]	0,906248657	[c d b b]	0,906248657	[c d b d]
0,929048675	[b b d c]	0,929071724	[b d d c]	0,906248657	[b d d c]	0,906248657	[c d b c]	0,906248657	[c d c b]
0,929048675	[b b d d]	0,929071724	[b d d d]	0,906248657	[b d d d]	0,906248657	[c d b d]	0,906248657	[c d c c]
0,929048675	[b d b b]	0,917345364	[c c b b]	0,906248657	[c d b b]	0,906248657	[c d c b]	0,906248657	[c d c d]
0,929048675	[b d b c]	0,917345364	[c c b c]	0,906248657	[c d b c]	0,906248657	[c d c c]	0,906248657	[c d d b]
0,929048675	[b d b d]	0,917345364	[c c b d]	0,906248657	[c d b d]	0,906248657	[c d c d]	0,906248657	[c d d c]
0,929048675	[b d c b]	0,917345364	[c c c b]	0,906248657	[c d c b]	0,906248657	[c d d b]	0,906248657	[c d d d]
0,929048675	[b d c c]	0,917345364	[c c c c]	0,906248657	[c d c c]	0,906248657	[c d d c]	0,906248657	[d b b b]
0,929048675	[b d c d]	0,917345364	[c c c d]	0,906248657	[c d c d]	0,906248657	[c d d d]	0,906248657	[d b b c]
0,929048675	[b d d b]	0,917345364	[c c d b]	0,906248657	[c d d b]	0,906248657	[d b b b]	0,906248657	[d b b d]
0,929048675	[b d d c]	0,917345364	[c c d c]	0,906248657	[c d d c]	0,906248657	[d b b c]	0,906248657	[d b c b]
0,929048675	[b d d d]	0,917345364	[c c d d]	0,906248657	[c d d d]	0,906248657	[d b b d]	0,906248657	[d b c c]
0,929048675	[c b b b]	0,906248657	[d b b b]	0,906248657	[d b b b]	0,906248657	[d b c b]	0,906248657	[d b c d]
0,929048675	[c b b c]	0,906248657	[d b b c]	0,906248657	[d b b c]	0,906248657	[d b c c]	0,906248657	[d b d b]
0,929048675	[c b b d]	0,906248657	[d b b d]	0,906248657	[d b b d]	0,906248657	[d b c d]	0,906248657	[d b d c]
0,929048675	[c b c b]	0,906248657	[d b c b]	0,906248657	[d b c b]	0,906248657	[d b d b]	0,906248657	[d b d d]
0,929048675	[c b c c]	0,906248657	[d b c c]	0,906248657	[d b c c]	0,906248657	[d b d c]	0,906248657	[d c b b]
0,929048675	[c b c d]	0,906248657	[d b c d]	0,906248657	[d b c d]	0,906248657	[d b d d]	0,906248657	[d c b c]
0,929048675	[c b d b]	0,906248657	[d b d b]	0,906248657	[d b d b]	0,906248657	[d c b b]	0,906248657	[d c b d]
0,929048675	[c b d c]	0,906248657	[d b d c]	0,906248657	[d b d c]	0,906248657	[d c b c]	0,906248657	[d c c b]
0,929048675	[c b d d]	0,906248657	[d b d d]	0,906248657	[d b d d]	0,906248657	[d c b d]	0,906248657	[d c c c]
0,929048675	[c d b b]	0,906248657	[d c b b]	0,906248657	[d c b b]	0,906248657	[d c c b]	0,906248657	[d c c d]
0,929048675	[c d b c]	0,906248657	[d c b c]	0,906248657	[d c b c]	0,906248657	[d c c c]	0,906248657	[d c d b]
0,929048675	[c d b d]	0,906248657	[d c b d]	0,906248657	[d c b d]	0,906248657	[d c c d]	0,906248657	[d c d c]
0,929048675	[c d c b]	0,906248657	[d c c b]	0,906248657	[d c c b]	0,906248657	[d c d b]	0,906248657	[d c d d]
0,929048675	[c d c c]	0,906248657	[d c c c]	0,906248657	[d c c c]	0,906248657	[d c d c]	0,906248657	[d d b b]
0,929048675	[c d c d]	0,906248657	[d c c d]	0,906248657	[d c c d]	0,906248657	[d c d d]	0,906248657	[d d b c]
0,929048675	[c d d b]	0,906248657	[d c d b]	0,906248657	[d c d b]	0,906248657	[d d b b]	0,906248657	[d d b d]
0,929048675	[c d d c]	0,906248657	[d c d c]	0,906248657	[d c d c]	0,906248657	[d d b c]	0,906248657	[d d c b]
0,929048675	[c d d d]	0,906248657	[d c d d]	0,906248657	[d c d d]	0,906248657	[d d b d]	0,906248657	[d d c c]
0,913949869	[c c b d]	0,906248657	[d d b b]	0,906248657	[d d b b]	0,906248657	[d d c b]	0,906248657	[d d c d]
0,913949869	[c c c b]	0,906248657	[d d b c]	0,906248657	[d d b c]	0,906248657	[d d c c]	0,906248657	[d d d b]
0,913949869	[c c b b]	0,906248657	[d d b d]	0,906248657	[d d b d]	0,906248657	[d d c d]	0,906248657	[d d d c]
0,913949869	[c c b c]	0,906248657	[d d c b]	0,906248657	[d d c b]	0,906248657	[d d d b]	0,906248657	[d d d d]
0,913949869	[c c c c]	0,906248657	[d d c c]	0,906248657	[d d c c]	0,906248657	[d d d c]	0,894106975	[c c b d]
0,913949869	[c c c d]	0,906248657	[d d c d]	0,906248657	[d d c d]	0,906248657	[d d d d]	0,894106975	[c c c d]
0,913949869	[c c d b]	0,906248657	[d d d b]	0,906248657	[d d d b]	0,894106975	[c c d b]	0,894106975	[c c d b]
0,913949869	[c c d c]	0,906248657	[d d d c]	0,906248657	[d d d c]	0,894106975	[c c d c]	0,894106975	[c c d c]
0,913949869	[c c d d]	0,906248657	[d d d d]	0,906248657	[d d d d]	0,894106975	[c c d d]	0,894106975	[c c d d]

Figure 7.10: Reconstruction accuracies for tensors with interval information localized along the fourth mode, L (*the greener the cell, the better the result*, best option highlighted in light green – the tables are best viewed in color)

$\mathcal{G}_1$ as interval		$\mathcal{G}_2$ as interval		$\mathcal{G}_3$ as interval		$\mathcal{G}_4$ as interval		$\mathcal{G}_5$ as interval	
Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options
0,929048675	[b b b c]	0,933109207	[c b b b]	0,944289019	[c c b b]	0,944912824	[c c c b]	0,953128196	[c c c c]
0,929048675	[b b b d]	0,933109207	[c b b c]	0,944289019	[c c b c]	0,944912824	[c c c c]	0,92641883	[c b c c]
0,929048675	[b c b b]	0,933109207	[c b b d]	0,944289019	[c c b d]	0,944912824	[c c c d]	0,926389731	[c b c b]
0,929048675	[b c b c]	0,933109207	[c b c b]	0,944289019	[c c c b]	0,92873038	[c b c b]	0,926377354	[c b b c]
0,929048675	[b c b d]	0,933109207	[c b c c]	0,944289019	[c c c c]	0,92873038	[c b c c]	0,926366232	[c b b b]
0,929048675	[b c d b]	0,933109207	[c b c d]	0,944289019	[c c c d]	0,92873038	[c b c d]	0,926308468	[b c c c]
0,929048675	[b c d c]	0,933109207	[c b d b]	0,944289019	[c c d b]	0,928699299	[c b b b]	0,926307153	[b c c b]
0,929048675	[b c d d]	0,933109207	[c b d c]	0,944289019	[c c d c]	0,928699299	[c b b c]	0,92630591	[b c b c]
0,929048675	[b d b b]	0,933109207	[c b d d]	0,944289019	[c c d d]	0,928699299	[c b b d]	0,926305701	[b b c c]
0,929048675	[b d b c]	0,933109207	[c d b b]	0,929193044	[c b b b]	0,928681554	[b c c b]	0,926305553	[b c b b]
0,929048675	[b d b d]	0,933109207	[c d b c]	0,929193044	[c b b c]	0,928681554	[b c c c]	0,926305469	[b b c b]
0,929048675	[b d c b]	0,933109207	[c d b d]	0,929193044	[c b b d]	0,928681554	[b c c d]	0,926304584	[b b b c]
0,929048675	[b d c c]	0,933109207	[c d c b]	0,929193044	[c b c b]	0,928678948	[b c b b]	0,926304417	[b b b b]
0,929048675	[b d c d]	0,933109207	[c d c c]	0,929193044	[c b c c]	0,928678948	[b c b c]	0,918590982	[c c c b]
0,929048675	[b d d b]	0,933109207	[c d c d]	0,929193044	[c b c d]	0,928678948	[b c b d]	0,917064562	[c c b c]
0,929048675	[b d d c]	0,933109207	[c d d b]	0,929193044	[c b d b]	0,92867804	[b b c b]	0,915911901	[c c b b]
0,929048675	[b d d d]	0,933109207	[c d d c]	0,929193044	[c b d c]	0,92867804	[b b c c]	0,906248657	[b b b d]
0,929048675	[c b b c]	0,933109207	[c d d d]	0,929193044	[c b d d]	0,92867804	[b b c d]	0,906248657	[b b c d]
0,929048675	[c b b d]	0,929071724	[b b b c]	0,928030283	[b c b b]	0,928677921	[b b b b]	0,906248657	[b b d b]
0,929048675	[c b c b]	0,929071724	[b b d b]	0,928030283	[b c b c]	0,928677921	[b b b c]	0,906248657	[b b d c]
0,929048675	[d b b b]	0,929071724	[b b d c]	0,928030283	[b c b d]	0,928677921	[b b b d]	0,906248657	[b b d d]
0,929048675	[d b b c]	0,929071724	[b b d d]	0,928030283	[b c c b]	0,914570318	[c c b b]	0,906248657	[b c b d]
0,929048675	[d b b d]	0,929071724	[b c b b]	0,928030283	[b c c c]	0,914570318	[c c b c]	0,906248657	[b c c d]
0,929048675	[d b c b]	0,929071724	[b c c b]	0,928030283	[b c c d]	0,914570318	[c c b d]	0,906248657	[b c d b]
0,929048675	[d b c c]	0,929071724	[b c c d]	0,928030283	[b c d b]	0,906248657	[b b d b]	0,906248657	[b c d c]
0,929048675	[d b c d]	0,929071724	[b c d b]	0,928030283	[b c d c]	0,906248657	[b b d c]	0,906248657	[b c d d]
0,929048675	[d b d b]	0,929071724	[b c d c]	0,928030283	[b c d d]	0,906248657	[b b d d]	0,906248657	[b d b b]
0,929048675	[d b d c]	0,929071724	[b c d d]	0,928030134	[b b b b]	0,906248657	[b c d b]	0,906248657	[b d b c]
0,929048675	[d b d d]	0,929071724	[b b b b]	0,928030134	[b b b c]	0,906248657	[b c d c]	0,906248657	[b d b d]
0,929048675	[d c b b]	0,929071724	[b b b d]	0,928030134	[b b b d]	0,906248657	[b c d d]	0,906248657	[b d c b]
0,929048675	[d c b c]	0,929071724	[b b c b]	0,928030134	[b b c b]	0,906248657	[b d b b]	0,906248657	[b d c c]
0,929048675	[d c b d]	0,929071724	[b b c c]	0,928030134	[b b c c]	0,906248657	[b d b c]	0,906248657	[b d c d]
0,929048675	[d c c b]	0,929071724	[b b c d]	0,928030134	[b b c d]	0,906248657	[b d b d]	0,906248657	[b d d b]
0,929048675	[d c c c]	0,929071724	[b c b c]	0,928030134	[b b d b]	0,906248657	[b d c b]	0,906248657	[b d d c]
0,929048675	[d c c d]	0,929071724	[b c b d]	0,928030134	[b b d c]	0,906248657	[b d c c]	0,906248657	[b d d d]
0,929048675	[d c d b]	0,929071724	[b c c c]	0,928030134	[b b d d]	0,906248657	[b d c d]	0,906248657	[c b b d]
0,929048675	[d c d c]	0,929071724	[b d b b]	0,906248657	[b d b b]	0,906248657	[b d d b]	0,906248657	[c b c d]
0,929048675	[d c d d]	0,929071724	[b d b c]	0,906248657	[b d b c]	0,906248657	[b d d c]	0,906248657	[c b d b]
0,929048675	[d d b b]	0,929071724	[b d b d]	0,906248657	[b d b d]	0,906248657	[b d d d]	0,906248657	[c b d c]
0,929048675	[d d b c]	0,929071724	[b d c b]	0,906248657	[b d c b]	0,906248657	[c b d b]	0,906248657	[c b d d]
0,929048675	[d d b d]	0,929071724	[b d c c]	0,906248657	[b d c c]	0,906248657	[c b d c]	0,906248657	[c d b b]
0,929048675	[d d c b]	0,929071724	[b d c d]	0,906248657	[b d c d]	0,906248657	[c b d d]	0,906248657	[c d b c]
0,929048675	[d d c c]	0,929071724	[b d d b]	0,906248657	[b d d b]	0,906248657	[c d b b]	0,906248657	[c d b d]
0,929048675	[d d c d]	0,929071724	[b d d c]	0,906248657	[b d d c]	0,906248657	[c d b c]	0,906248657	[c d c b]
0,929048675	[d d d b]	0,929071724	[b d d d]	0,906248657	[b d d d]	0,906248657	[c d b d]	0,906248657	[c d c c]
0,929048675	[d d d c]	0,917345364	[c c b b]	0,906248657	[c d b b]	0,906248657	[c d c b]	0,906248657	[c d c d]
0,929048675	[d d d d]	0,917345364	[c c b c]	0,906248657	[c d b c]	0,906248657	[c d c c]	0,906248657	[c d d b]
0,929048675	[b b b b]	0,917345364	[c c b d]	0,906248657	[c d b d]	0,906248657	[c d c d]	0,906248657	[c d d c]
0,929048675	[b b c b]	0,917345364	[c c c b]	0,906248657	[c d c b]	0,906248657	[c d d b]	0,906248657	[c d d d]
0,929048675	[b b c c]	0,917345364	[c c c c]	0,906248657	[c d c c]	0,906248657	[c d d c]	0,906248657	[d b b b]
0,929048675	[b b c d]	0,917345364	[c c c d]	0,906248657	[c d c d]	0,906248657	[c d d d]	0,906248657	[d b b c]
0,929048675	[b b d b]	0,917345364	[c c d b]	0,906248657	[c d d b]	0,906248657	[d b b b]	0,906248657	[d b b d]
0,929048675	[b b d c]	0,917345364	[c c d c]	0,906248657	[c d d c]	0,906248657	[d b b c]	0,906248657	[d b c b]
0,929048675	[b b d d]	0,917345364	[c c d d]	0,906248657	[c d d d]	0,906248657	[d b b d]	0,906248657	[d b c c]
0,929048675	[b c c b]	0,906248657	[d b b b]	0,906248657	[d b b b]	0,906248657	[d b c b]	0,906248657	[d b c d]
0,929048675	[b c c c]	0,906248657	[d b b c]	0,906248657	[d b b c]	0,906248657	[d b c c]	0,906248657	[d b d b]
0,929048675	[b c c d]	0,906248657	[d b b d]	0,906248657	[d b b d]	0,906248657	[d b c d]	0,906248657	[d b d c]
0,929048675	[b c b b]	0,906248657	[d b c b]	0,906248657	[d b c b]	0,906248657	[d b d b]	0,906248657	[d b d d]
0,929048675	[c b c c]	0,906248657	[d b c c]	0,906248657	[d b c c]	0,906248657	[d b d c]	0,906248657	[d c b b]
0,929048675	[c b c d]	0,906248657	[d b c d]	0,906248657	[d b c d]	0,906248657	[d b d d]	0,906248657	[d c b c]
0,929048675	[c b d b]	0,906248657	[d b d b]	0,906248657	[d b d b]	0,906248657	[d c b b]	0,906248657	[d c b d]
0,929048675	[c b d c]	0,906248657	[d b d c]	0,906248657	[d b d c]	0,906248657	[d c b c]	0,906248657	[d c c b]
0,929048675	[c b d d]	0,906248657	[d b d d]	0,906248657	[d b d d]	0,906248657	[d c b d]	0,906248657	[d c c c]
0,929048675	[c d b b]	0,906248657	[d c b b]	0,906248657	[d c b b]	0,906248657	[d c c b]	0,906248657	[d c c d]
0,929048675	[c d b c]	0,906248657	[d c b c]	0,906248657	[d c b c]	0,906248657	[d c c c]	0,906248657	[d c d b]
0,929048675	[c d b d]	0,906248657	[d c b d]	0,906248657	[d c b d]	0,906248657	[d c c d]	0,906248657	[d c d c]
0,929048675	[c d c b]	0,906248657	[d c c b]	0,906248657	[d c c b]	0,906248657	[d c d b]	0,906248657	[d c d d]
0,929048675	[c d c c]	0,906248657	[d c c c]	0,906248657	[d c c c]	0,906248657	[d c d c]	0,906248657	[d d b b]
0,929048675	[c d c d]	0,906248657	[d c c d]	0,906248657	[d c c d]	0,906248657	[d c d d]	0,906248657	[d d b c]
0,929048675	[c d d b]	0,906248657	[d c d b]	0,906248657	[d c d b]	0,906248657	[d d b b]	0,906248657	[d d b d]
0,929048675	[c d d c]	0,906248657	[d c d c]	0,906248657	[d c d c]	0,906248657	[d d b c]	0,906248657	[d d c b]
0,929048675	[c d d d]	0,906248657	[d c d d]	0,906248657	[d c d d]	0,906248657	[d d b d]	0,906248657	[d d c c]
0,913949869	[c c b b]	0,906248657	[d d b b]	0,906248657	[d d b b]	0,906248657	[d d c b]	0,906248657	[d d c d]
0,913949869	[c c b c]	0,906248657	[d d b c]	0,906248657	[d d b c]	0,906248657	[d d c c]	0,906248657	[d d d b]
0,913949869	[c c b d]	0,906248657	[d d b d]	0,906248657	[d d b d]	0,906248657	[d d c d]	0,906248657	[d d d c]
0,913949869	[c c c b]	0,906248657	[d d c b]	0,906248657	[d d c b]	0,906248657	[d d d b]	0,906248657	[d d d d]
0,913949869	[c c c c]	0,906248657	[d d c c]	0,906248657	[d d c c]	0,906248657	[d d d c]	0,894106975	[c c c d]
0,913949869	[c c c d]	0,906248657	[d d c d]	0,906248657	[d d c d]	0,906248657	[d d d d]	0,894106975	[c c b d]
0,913949869	[c c d b]	0,906248657	[d d d b]	0,906248657	[d d d b]	0,894106975	[c c d b]	0,894106975	[c c d b]
0,913949869	[c c d c]	0,906248657	[d d d c]	0,906248657	[d d d c]	0,894106975	[c c d c]	0,894106975	[c c d c]
0,913949869	[c c d d]	0,906248657	[d d d d]	0,906248657	[d d d d]	0,894106975	[c c d d]	0,894106975	[c c d d]

Figure 7.11: Reconstruction accuracies for tensors with interval information localized along the fifth mode, M (*the greener the cell, the better the result*, best option highlighted in light green – the tables are best viewed in color)

Noise along mode I		Noise along mode J		Noise along mode K		Noise along mode L		Noise along mode M	
$\mathcal{G}_1$ as interval		$\mathcal{G}_2$ as interval		$\mathcal{G}_3$ as interval		$\mathcal{G}_4$ as interval		$\mathcal{G}_5$ as interval	
Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options	Average HM (decr. order)	Recomb. options
0,982005205	[d d d d]	0,93983634	[c d d d]	0,969000273	[c c d d]	0,944128581	[c c c d]	0,953128196	[c c c c]

Figure 7.12: Best reconstruction accuracies for tensors with interval information localized along each mode (*the greener the cell, the better the result*, best option highlighted in light green – the tables are best viewed in color)

Chapter 8

Conclusions and Future Work

We have presented, in the foregoing pages, a generalization of the factorization process in order to deal with interval-valued matrices and 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 this observation, we have proposed a set of tools to tackle this problem by extending both eigenvector-based and probabilistic state-of-the-art techniques. These approaches have then been validated by a set of experiments under diverse data scenarios and application semantics, in order to prove their robustness and efficiency.

More in detail, Chapter 4, in the first part of the thesis, illustrates the challenges that emerged when trying to extend known matrix factorization techniques (such as Singular Value Decomposition, SVD and Probabilistic Matrix Factorization, PMF) to interval-valued data, and we proposed a set of algorithms (ISVD and AI-PMF, respectively) to address them.

The main idea behind our approach is that of guaranteeing that the latent semantic components in the minimum and maximum factor matrices are always correctly matched and aligned, a goal that can be achieved by means of the Interval Latent Semantic Alignment (ILSA, Section 4.2) procedure. Particular effort has been devoted to an incremental study for the implementation of the ISVD strategy (see Section 4.4), starting from a very naive approach (ISVD_0) and gradually

building up to more elaborate versions (from ISVD_1 to ISVD_4), each adding to the previous one by either including additional steps or by improving existing ones.

We then conclude the chapter by presenting a method for improving on the work of the authors in [46], who already proposed an approach for extending the PMF method to interval-valued data. Also in this case, we relied on the latent semantic alignment intuition for defining our AI-PMF strategy (Section 4.5).

All the presented approaches have then been validated with a set of experiments over various scenarios and under different conditions, as detailed in Chapter 5, proving their effectiveness in dealing with applications such as matrix reconstruction, image classification and collaborative filtering.

The second part of the thesis is dedicated to the application of the techniques proposed for matrices to tackle the tensor factorization problem in presence of interval-valued data. Specifically, after a brief overview of the state of the art (Chapter 6), we present an extension of the *tensor train* decomposition algorithm based on the ISVD strategy (Chapter 7). The chapter also presents a set of experiments (on synthetic data) devised to provide a first validation of our approach. In particular, we assess the accuracy of the decomposition and reconstruction process when dealing with different types of interval information, either with the interval-valued data uniformly distributed in the input tensors, or *localized* along one particular mode.

In conclusion, this dissertation has presented an investigation on the interval-valued data factorization problem that is in no way intended to be exhaustive, and we believe there is much progress to be made. In fact, our decision of modeling uncertainty as intervals is just one among many (probabilistic, fuzzy, etc.) and it could be interesting to investigate how our approach could be improved by taking in considerations aspects from each one of those. This could provide a more robust approach for dealing with real-world scenarios where uncertain data are not easily categorized.

Regarding the tensor decomposition approach presented in Chapter 7, a lot of work is still to be done in terms of validation, especially in real-world scenarios: being a relatively unexplored field of study, we expect plenty of opportunities for our interval-valued based approach to find useful applications. An interesting aspect which emerged from our analysis, and that certainly requires further investiga-

tion, is how the disposition of the interval information inside the input tensors can be exploited to drive the decomposition process. From what we saw with synthetic data, in fact, a clear-cut case where the interval-valued data are concentrated along only one of the modes has a noticeable impact on how these information should be preserved in the factorization, but it would be interesting to see how a more complex (i.e., multi-modal) arrangement would require the algorithm to adapt, either by defining a general strategy through a preliminary analysis of the data, or by tailoring the parameters at each step of the process.

Appendices

Appendix A

Main Functions Pseudocode

This appendix reports the pseudocode for the main functions related to the interval-valued matrix factorization problem that we introduced throughout the thesis.

A.1 Pseudocode for the Supporting Functions

First of all, we provide here the pseudocode for some of the key supporting functions introduced in the thesis, including interval-valued matrix multiplication, average replacement for vectors and matrices with invalid interval-valued entries, the function for evaluating the inverse of a non-negative diagonal interval-valued matrix, and the L2-norm based matrix normalization.

A.1.1 Interval-valued Matrix Multiplication

Interval-valued matrix multiplication, introduced in Section 2.1, is a core matrix algebraic operation that is often relied on when is necessary to extend the standard matrix product.

Algorithm 1 Interval-valued Matrix Multiplication

Input:
Interval-valued matrices $\ddot{\mathbf{A}} = [\mathbf{A}_*, \mathbf{A}^*] \in \mathbb{R}^{n_A \times m_A}$, $\ddot{\mathbf{B}} = [\mathbf{B}_*, \mathbf{B}^*] \in \mathbb{R}^{n_B \times m_B}$, with $m_A = n_B$.

Output:
Interval-valued matrix $\ddot{\mathbf{R}} = [\mathbf{R}_*, \mathbf{R}^*] \in \mathbb{R}^{m_B \times n_A}$.

1: **procedure** INTERVAL-MAT-MULT($\ddot{\mathbf{A}}, \ddot{\mathbf{B}}$)
2: Initialize $\ddot{\mathbf{R}}, \mathbf{T}_1, \mathbf{T}_2, \mathbf{T}_3, \mathbf{T}_4 \in \mathbb{R}^{m_B \times n_A}$
3: $\mathbf{T}_1 \leftarrow \mathbf{A}_* \cdot \mathbf{B}_*$
4: $\mathbf{T}_2 \leftarrow \mathbf{A}_* \cdot \mathbf{B}^*$
5: $\mathbf{T}_3 \leftarrow \mathbf{A}^* \cdot \mathbf{B}_*$
6: $\mathbf{T}_4 \leftarrow \mathbf{A}^* \cdot \mathbf{B}^*$
7:
8: **for** $i \leftarrow 1$ *to* m_B **do**
9: **for** $j \leftarrow 1$ *to* n_A **do**
10: $\mathbf{R}_{*[i,j]} \leftarrow \min\{\mathbf{T}_{1[i,j]}, \mathbf{T}_{2[i,j]}, \mathbf{T}_{3[i,j]}, \mathbf{T}_{4[i,j]}\}$
11: $\mathbf{R}^*_{[i,j]} \leftarrow \max\{\mathbf{T}_{1[i,j]}, \mathbf{T}_{2[i,j]}, \mathbf{T}_{3[i,j]}, \mathbf{T}_{4[i,j]}\}$
12: **end for**
13: **end for**
14: **return** $\ddot{\mathbf{R}}$
15: **end procedure**

A.1.2 Vector and Matrix Average Replacement

The *average replacement* is a mechanism introduced (see Section 4.2) in order to correct vectors and matrices that include entries that are not valid intervals. Several steps of the decomposition procedures potentially introduce such incorrectly formed intervals (where the minimum values are larger than the maximum) in their intermediary steps: these proposed functions help to validate the final factorization output, in order to provide coherent interval-valued matrices.

Algorithm 2 Vector Average Replacement

Input:
Interval-valued vector $\ddot{\mathbf{v}} = [\mathbf{v}_*, \mathbf{v}^*] \in \mathbb{R}^{1 \times n}$.

Output:
Interval-valued vector $\ddot{\mathbf{w}} = [\mathbf{w}_*, \mathbf{w}^*] \in \mathbb{R}^{1 \times n}$.

procedure AVERAGE-REPLACEMENT-VEC($\ddot{\mathbf{v}}$)
    Initialize $\ddot{\mathbf{w}} \in \mathbb{R}^{1\times n}$
    $\ddot{\mathbf{w}} \leftarrow \ddot{\mathbf{v}}$

    for $i \leftarrow 1$ *to* n **do**
        if $\mathbf{v}_{*[i]} > \mathbf{v}^{*}[i]$ **then**
            $\mathbf{w}_{*[i]} \leftarrow \frac{\mathbf{v}_{*[i]}+\mathbf{v}^{*}_{[i]}}{2}$
            $\mathbf{w}^{*}_{[i]} \leftarrow \mathbf{w}_{*[i]}$
        end if
    end for
    return $\ddot{\mathbf{w}}$
end procedure

Algorithm 3 Matrix Average Replacement

Input:
Interval-valued matrix $\ddot{\mathbf{M}} = [\mathbf{M}_{*}, \mathbf{M}^{*}] \in \mathbb{R}^{n\times m}$.

Output:
Interval-valued matrix $\ddot{\mathbf{N}} = [\mathbf{N}_{*}, \mathbf{N}^{*}] \in \mathbb{R}^{n\times m}$.

procedure AVERAGE-REPLACEMENT-MAT($\ddot{\mathbf{M}}$)
    Initialize $\ddot{\mathbf{N}} \in \mathbb{R}^{n\times m}$
    $\ddot{\mathbf{N}} \leftarrow \ddot{\mathbf{M}}$

    for $i \leftarrow 1$ *to* n **do**
        for $j \leftarrow 1$ *to* m **do**
            if $\mathbf{M}_{*[i,j]} > \mathbf{M}^{*}_{[i,j]}$ **then**
                $\mathbf{N}_{*[i,j]} \leftarrow \frac{\mathbf{M}_{*[i,j]}+\mathbf{M}^{*}_{[i,j]}}{2}$
                $\mathbf{N}^{*}_{[i,j]} \leftarrow \mathbf{N}_{*[i,j]}$
            end if
        end for
    end for
    return $\ddot{\mathbf{N}}$
end procedure

A.1.3 Inverse of a diagonal non-negative interval-valued matrix

Several ISVD algorithms require the inverse of a non-negative interval-valued core matrix $\ddot{\mathbf{\Sigma}}$. As discussed in Section 4.4.4, we can only provide an approximation of the inverse problem, which can be implemented as follows.

Algorithm 4 Inverse of a diagonal non-negative interval-valued matrix $\ddot{\mathbf{\Sigma}}$

Input:
Interval-valued matrix $\ddot{\mathbf{\Sigma}} = [\mathbf{\Sigma}_*, \mathbf{\Sigma}^*] \in \mathbb{R}^{r \times r}$, target rank r.

Output:
Scalar matrix $\mathbf{\Sigma}^{\text{inv}} \in \mathbb{R}^{r \times r}$, approximating the inverse of the input matrix $\ddot{\mathbf{\Sigma}}$.

procedure INVERSE-$\Sigma(\ddot{\mathbf{\Sigma}}, r)$
  Initialize $\mathbf{\Sigma}^{\text{inv}} \in \mathbb{R}^{r \times r} : \forall i, j \; \mathbf{\Sigma}^{\text{inv}}_{[i,j]} \leftarrow 0$
  for $i \leftarrow 1$ *to* r **do**
    if $\mathbf{\Sigma}_{*[i,i]} = 0 \wedge \mathbf{\Sigma}^*_{[i,i]} = 0$ **then**
      $\mathbf{\Sigma}_{\text{inv}[i,i]} \leftarrow 0$
    else if $\mathbf{\Sigma}_{*[i,i]} = 0 \wedge \mathbf{\Sigma}^*_{[i,i]} \neq 0$ **then**
      $\mathbf{\Sigma}_{\text{inv}[i,i]} \leftarrow \frac{2}{\mathbf{\Sigma}^*_{[i,i]}}$
    else if $\mathbf{\Sigma}_{*[i,i]} \neq 0 \wedge \mathbf{\Sigma}^*_{[i,i]} = 0$ **then**
      $\mathbf{\Sigma}_{\text{inv}[i,i]} \leftarrow \frac{2}{\mathbf{\Sigma}_{*[i,i]}}$
    else
      $\mathbf{\Sigma}_{\text{inv}[i,i]} \leftarrow \frac{2}{\mathbf{\Sigma}_{*[i,i]} + \mathbf{\Sigma}^*_{[i,i]}}$
    end if
  end for
  return $\mathbf{\Sigma}^{\text{inv}}$
end procedure

A.1.4 L2-Norm based Matrix Normalization

As we see in 4.8 and discuss in Section 4.2 of the thesis, results may need to be L2-normalized before being returned to the end-user.

Algorithm 5 L2-Norm Matrix Normalization

Input:
Scalar matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$.

Output:
Scalar matrix $\hat{\mathbf{A}} \in \mathbb{R}^{n \times m}$, vector of $\mathbf{A}$ columns norms $\mathbf{w} \in \mathbb{R}^{1 \times m}$.

procedure NORM-MAT($\mathbf{A}$)
    Initialize $\hat{\mathbf{A}} \in \mathbb{R}^{n \times m}$
    Initialize $\mathbf{w} \in \mathbb{R}^{1 \times m}$

    for $j \leftarrow 1$ *to* n **do**
        $\mathbf{w}_{[j]} \leftarrow \|\mathbf{A}_{[:,j]}\|_2$
        for $i \leftarrow 1$ *to* m **do**
            $\hat{\mathbf{A}}_{[i,j]} \leftarrow \frac{\hat{\mathbf{A}}_{[i,j]}}{\mathbf{w}_{[j]}}$
        end for
    end for
    return $\hat{\mathbf{A}}, \mathbf{w}$
end procedure

A.2 Pseudocode for Interval-Valued Latent Semantic Alignment (ILSA)

The Interval-valued Latent Semantic Alignment (ILSA), discussed in Section 4.2, is a key process of our proposed decomposition strategy, helping combine separately obtained minimum and maximum basis vectors to form a coherent interval-valued latent space. Here we provide the pseudocode for the algorithm we used in our implementation of the latent semantic alignment process.

Algorithm 6 Interval-Valued Latent Semantic Alignment (ILSA)

Input:
Interval-valued matrix $\ddot{\mathbf{V}} = [\mathbf{V}_*, \mathbf{V}^*] \in \mathbb{R}^{n \times r}$, with $\mathbf{V}_* = \{\mathbf{v}_{*1}, \ldots, \mathbf{v}_{*r}\}$,
$\mathbf{V}^* = \{\mathbf{v}^*_1, \ldots, \mathbf{v}^*_r\}$, target rank r.

Output:

Indices vector $\mathbf{mappingIdx} \in \mathbb{R}^{1\times r}$ (for aligning the interval-valued components of $\ddot{\mathbf{V}}$), where $\mathbf{mappingIdx}_j$ is the remapped index for $\mathbf{v}_{*j}$, $\mathbf{u}_{*j}$ and σ_{*j}.
Indicator vector $\mathbf{dirFlag} \in \mathbb{R}^{1\times r}$ (to swap direction of misaligned $\ddot{\mathbf{V}}$ components), where $\mathbf{dirFlag}_{[j]} = 1$ if $\mathbf{v}_{*j}$ needs to be swapped to better align with $\mathbf{v}^*_j$, otherwise $\mathbf{dirFlag}_{[j]} = 0$.

```
# Main procedure
procedure ILSA(V̈, r)
    simMat, dirFlag ← PAIRSIM(V̈, r)
    mappingIdx ← MAPPING(simMat, r)
    return mappingIdx, dirFlag
end procedure

# Compute pairwise cosine similarity
procedure PAIRSIM(V̈, r)
    Initialize simMat ∈ ℝ^{r×r}
    Initialize dirFlag ∈ ℝ^{1×r}
    for i ← 1 to r do
        for j ← 1 to r do
            if cos(v_{*i}, v*_j) < 0 then                ▷ cosine similarity
                simMat_{[i,i]} ← cos(v_{*i}, −v*_j)
                dirFlag_{[i]} = 1
            else
                simMat_{[i,i]} ← cos(v_{*i}, v*_j)
                dirFlag_{[i]} = 0
            end if
        end for
    end for
    return simMat, dirFlag
end procedure

# Find the mapping maximizing the pairwise similarity between V_* and V*
procedure MAPPING(simMat, r)
    Initialize mappingIdx ∈ ℝ^{1×r}
    for j ← 1 to r do
        mappingIdx_j ← i | ∄k : simMat_{[k,j]} < simMat_{[i,i]}
```

31: **end for**
32: **spareidx** $\leftarrow i \mid \nexists j : \mathbf{mappingIdx}_j = i$
33: **if spareidx** $\neq \varnothing$ **then**
34: **conflictIdx**$_*$ $\leftarrow i \mid \exists j_1, j_2 : \mathbf{mappingIdx}[j_1] = i \wedge$
35: $\wedge$ **mappingIdx**$[j_2] = i \wedge j_1 \neq j_2$
36: **for** $i \in$ **conflictIdx**$_*$ **do**
37: **conflictIdx**$^* \leftarrow j \mid \mathbf{mappingIdx}_j = i$
38: sort(**conflictIdx***, **simMat**$_{[i,i]}$, *descending*)
39: **conflictIdx**$^* \leftarrow$ **conflictIdx**$^* \setminus \{\mathbf{conflictIdx}^*_{[1]}\}$
40: **for** $j \in$ **conflictIdx*** **do**
41: **mappingIdx**$_j \leftarrow k \mid k \in$ **spareidx** $\wedge$
42: $\wedge \nexists h :$ **simMat**$_{[h,j]} <$ **simMat**$_{[k,j]} \wedge h \in$ **spareidx**
43: **spareidx** $\leftarrow$ **spareidx** $\setminus \{k\}$
44: **end for**
45: **end for**
46: **end if**
47: **return mappingIdx**
48: **end procedure**

A.3 Pseudocodes for ISVD Algorithms

As discussed in Section 4.4 (and summarized in Figure 4.8), we propose four different approaches (five including the naive one, ISVD_0) to decompose interval-valued matrices. Here we provide the detailed pseudocode for each one of these ISVD algorithms.

A.3.1 ISVD_0 (Naive Approach): Average and Decompose

The naive ISVD_0 algorithm, which simply operates by averaging the input interval-valued entries, is discussed in Section 4.4.1.

Algorithm 7 ISVD_0

Input:
Interval-valued matrix $\ddot{\mathbf{M}} = [\mathbf{M}_*, \mathbf{M}^*] \in \mathbb{R}^{n \times m}$, target rank r.

Output:
Scalar-valued factor matrices $\mathbf{U}_{\text{avg}} \in \mathbb{R}^{n\times r}, \mathbf{V}_{\text{avg}} \in \mathbb{R}^{m\times r}$ and scalar-valued singular matrix $\mathbf{\Sigma}_{\text{avg}} \in \mathbb{R}^{r\times r}$.

1: **procedure** $\text{ISVD}_0(\ddot{\mathbf{M}}, r)$
2: $\quad \mathbf{M}_{\text{avg}} \leftarrow \frac{\mathbf{M}_* + \mathbf{M}^*}{2}$
3: $\quad \mathbf{U}_{\text{avg}}, \mathbf{\Sigma}_{\text{avg}}, \mathbf{V}_{\text{avg}} \leftarrow \text{SVD}(\mathbf{M}_{\text{avg}}, r)$ ▷ Singular value decomposition
4: $\quad$ **return** $\mathbf{U}_{\text{avg}}, \mathbf{\Sigma}_{\text{avg}}, \mathbf{V}_{\text{avg}}$
5: **end procedure**

A.3.2 ISVD_1: Independently Decompose and Align

The ISVD_1 algorithm is described in Section 4.4.2 and is implemented as follows.

Algorithm 8 ISVD_1

Input:
Interval-valued matrix $\ddot{\mathbf{M}} = [\mathbf{M}_*, \mathbf{M}^*] \in \mathbb{R}^{n\times m}$, target rank r, decomposition target option $optDT$.

Output:
Decomposition target option (a):
Interval-valued factor matrices $\ddot{\mathbf{U}} \in \mathbb{R}^{n\times r}, \ddot{\mathbf{V}} \in \mathbb{R}^{m\times r}$ and interval-valued singular matrix $\ddot{\mathbf{\Sigma}} \in \mathbb{R}^{r\times r}$.

Decomposition target option (b):
Scalar-valued factor matrices $\mathbf{U}_{\text{avg}} \in \mathbb{R}^{n\times r}, \mathbf{V}_{\text{avg}} \in \mathbb{R}^{m\times r}$ and interval-valued singular matrix $\ddot{\mathbf{\Sigma}} \in \mathbb{R}^{r\times r}$.

Decomposition target option (c):
Scalar-valued factor matrices $\mathbf{U}_{\text{avg}} \in \mathbb{R}^{n\times r}, \mathbf{V}_{\text{avg}} \in \mathbb{R}^{m\times r}$ and scalar-valued singular matrix $\mathbf{\Sigma}_{\text{avg}} \in \mathbb{R}^{r\times r}$.

1: **procedure** $\text{ISVD}_1(\ddot{\mathbf{M}}, r, optDT)$
2: $\quad \mathbf{U}_*, \mathbf{\Sigma}_*, \mathbf{V}_* \leftarrow \text{SVD}(\mathbf{M}_*, r)$ ▷ Singular Value Decomposition
3: $\quad \mathbf{U}^*, \mathbf{\Sigma}^*, \mathbf{V}^* \leftarrow \text{SVD}(\mathbf{M}^*, r)$

4: **mappingIdx**, **dirFlag** $\leftarrow$ ILSA$(\mathbf{V}_*, \mathbf{V}^*)$ ▷ See Algorithm 6
5:
6: **for** $j \leftarrow 1$ *to* r **do**
7: $\mathbf{U}_{*[:,j]} \leftarrow \mathbf{U}_{*[:,\textbf{mappingIdx}_j]}$ ▷ Apply remapping
8: $\mathbf{V}_{*[:,j]} \leftarrow \mathbf{V}_{*[:,\textbf{mappingIdx}_j]}$
9: $\mathbf{\Sigma}_{*[j,j]} \leftarrow \mathbf{\Sigma}_{*[j,\textbf{mappingIdx}_j]}$
10: **if** $\textbf{dirFlag}_{[j]} = 1$ **then** ▷ Swap misaligned vectors
11: $\mathbf{U}_{*[:,j]} \leftarrow \mathbf{U}_{*[:,j]} \cdot (-1)$
12: $\mathbf{V}_{*[:,j]} \leftarrow \mathbf{V}_{*[:,j]} \cdot (-1)$
13: **end if**
14: **end for**
15:
16: $\ddot{\mathbf{U}} \leftarrow$ AVERAGE-REPLACEMENT-MAT$(\ddot{\mathbf{U}})$ ▷ See Algorithms 3 and 2
17: $\ddot{\mathbf{V}} \leftarrow$ AVERAGE-REPLACEMENT-MAT$(\ddot{\mathbf{V}})$
18: $\text{diag}(\ddot{\mathbf{\Sigma}}) \leftarrow$ AVERAGE-REPLACEMENT-VEC$(\text{diag}(\ddot{\mathbf{\Sigma}}))$
19:
20: **if** $optDT =$ 'a' **then**
21: **return** $\ddot{\mathbf{U}}, \ddot{\mathbf{\Sigma}}, \ddot{\mathbf{V}}$
22: **else if** $optDT =$ 'b' **then**
23: $\mathbf{U}_{\text{avg}} \leftarrow \frac{\mathbf{U}_* + \mathbf{U}^*}{2}$
24: $\mathbf{V}_{\text{avg}} \leftarrow \frac{\mathbf{V}_* + \mathbf{V}^*}{2}$
25: $\mathbf{U}_{\text{avg}}, \mathbf{w}_U \leftarrow$ NORM-MAT$(\mathbf{U}_{\text{avg}})$ ▷ See Algorithm 5
26: $\mathbf{V}_{\text{avg}}, \mathbf{w}_V \leftarrow$ NORM-MAT$(\mathbf{V}_{\text{avg}})$
27: $\mathbf{\Sigma}_* \leftarrow (\mathbf{w}_U \cdot \mathbf{w}_V) \cdot \mathbf{\Sigma}_*$
28: $\mathbf{\Sigma}^* \leftarrow (\mathbf{w}_U \cdot \mathbf{w}_V) \cdot \mathbf{\Sigma}^*$
29: **return** $\mathbf{U}_{\text{avg}}, \ddot{\mathbf{\Sigma}}, \mathbf{V}_{\text{avg}}$
30: **else if** $optDT =$ 'c' **then**
31: $\mathbf{U}_{\text{avg}} \leftarrow \frac{\mathbf{U}_* + \mathbf{U}^*}{2}$
32: $\mathbf{\Sigma}_{\text{avg}} \leftarrow \frac{\mathbf{\Sigma}_* + \mathbf{\Sigma}^*}{2}$
33: $\mathbf{V}_{\text{avg}} \leftarrow \frac{\mathbf{V}_* + \mathbf{V}^*}{2}$
34: $\mathbf{U}_{\text{avg}}, \mathbf{w}_U \leftarrow$ NORM-MAT$(\mathbf{U}_{\text{avg}})$ ▷ See Algorithm 5
35: $\mathbf{V}_{\text{avg}}, \mathbf{w}_V \leftarrow$ NORM-MAT$(\mathbf{V}_{\text{avg}})$
36: $\mathbf{\Sigma}_{\text{avg}} \leftarrow (\mathbf{w}_U \cdot \mathbf{w}_V) \cdot \mathbf{\Sigma}_{\text{avg}}$
37: **return** $\mathbf{U}_{\text{avg}}, \mathbf{\Sigma}_{\text{avg}}, \mathbf{V}_{\text{avg}}$
38: **end if**
39: **end procedure**

A.3.3 ISVD$_2$: Decompose, Solve, Align

The ISVD$_2$ algorithm is described in Section 4.4.3 and is implemented as follows.

Algorithm 9 ISVD$_2$

Input:
Interval-valued matrix $\ddot{\mathbf{M}} = [\mathbf{M}_*, \mathbf{M}^*] \in \mathbb{R}^{n \times m}$, target rank r, decomposition target option $optDT$.

Output:
Decomposition target option (a):
Interval-valued factor matrices $\ddot{\mathbf{U}} \in \mathbb{R}^{n \times r}, \ddot{\mathbf{V}} \in \mathbb{R}^{m \times r}$ and interval-valued singular matrix $\ddot{\mathbf{\Sigma}} \in \mathbb{R}^{r \times r}$.

Decomposition target option (b):
Scalar-valued factor matrices $\mathbf{U}_{\text{avg}} \in \mathbb{R}^{n \times r}, \mathbf{V}_{\text{avg}} \in \mathbb{R}^{m \times r}$ and interval-valued singular matrix $\ddot{\mathbf{\Sigma}} \in \mathbb{R}^{r \times r}$.

Decomposition target option (c):
Scalar-valued factor matrices $\mathbf{U}_{\text{avg}} \in \mathbb{R}^{n \times r}, \mathbf{V}_{\text{avg}} \in \mathbb{R}^{m \times r}$ and scalar-valued singular matrix $\mathbf{\Sigma}_{\text{avg}} \in \mathbb{R}^{r \times r}$.

procedure ISVD$_2$($\ddot{\mathbf{M}}, r, optDT$)
  $\ddot{\mathbf{A}} \leftarrow$ INTERVAL-MAT-MULT$(\ddot{\mathbf{M}}^{\mathrm{T}}, \ddot{\mathbf{M}})$ ▷ See Algorithm 1
  $\mathbf{V}_*$, $\mathbf{\Sigma}_*^2 \leftarrow \text{EIG}(A_*, r)$ ▷ Eigen Decomposition
  $\mathbf{V}^*$, $\mathbf{\Sigma}^{*2} \leftarrow \text{EIG}(A^*, r)$
  $\mathbf{U}_* \leftarrow \mathbf{M}_* \cdot (\mathbf{V}_*^T)^{-1} \cdot (\mathbf{\Sigma}_*)^{-1}$
  $\mathbf{U}^* \leftarrow \mathbf{M}^* \cdot (\mathbf{V}^{*\mathrm{T}})^{-1} \cdot (\mathbf{\Sigma}^*)^{-1}$
  mappingIdx, **dirFlag** $\leftarrow \text{ILSA}(\mathbf{V}_*, \mathbf{V}^*)$ ▷ See Algorithm 6

  for $j \leftarrow 1$ *to* r **do**
    $\mathbf{U}_{*[:,j]} \leftarrow \mathbf{U}_{*[:,\textbf{mappingIdx}_j]}$ ▷ Apply remapping
    $\mathbf{V}_{*[:,j]} \leftarrow \mathbf{V}_{*[:,\textbf{mappingIdx}_j]}$
    $\mathbf{\Sigma}_{*[j,j]} \leftarrow \mathbf{\Sigma}_{*[j,\textbf{mappingIdx}_j]}$
    if $\textbf{dirFlag}_j = 1$ **then** ▷ Swap misaligned vectors

$\mathbf{U}_{*[:,j]} \leftarrow \mathbf{U}_{*[:,j]} \cdot (-1)$
$\mathbf{V}_{*[:,j]} \leftarrow \mathbf{V}_{*[:,j]} \cdot (-1)$
end if
end for

$\ddot{\mathbf{U}} \leftarrow$ AVERAGE-REPLACEMENT-MAT$(\ddot{\mathbf{U}})$ ▷ See Algorithms 3 and 2
$\ddot{\mathbf{V}} \leftarrow$ AVERAGE-REPLACEMENT-MAT$(\ddot{\mathbf{V}})$
$\text{diag}(\ddot{\mathbf{\Sigma}}) \leftarrow$ AVERAGE-REPLACEMENT-VEC$(\text{diag}(\ddot{\mathbf{\Sigma}}))$

if $optDT =$ 'a' **then**
return $\ddot{\mathbf{U}}, \ddot{\mathbf{\Sigma}}, \ddot{\mathbf{V}}$
else if $optDT =$ 'b' **then**
$\mathbf{U}_{\text{avg}} \leftarrow \frac{\mathbf{U}_* + \mathbf{U}^*}{2}$
$\mathbf{V}_{\text{avg}} \leftarrow \frac{\mathbf{V}_* + \mathbf{V}^*}{2}$
$\mathbf{U}_{\text{avg}}, \mathbf{w}_U \leftarrow$ NORM-MAT$(\mathbf{U}_{\text{avg}})$ ▷ See Algorithm 5
$\mathbf{V}_{\text{avg}}, \mathbf{w}_V \leftarrow$ NORM-MAT$(\mathbf{V}_{\text{avg}})$
$\mathbf{\Sigma}_* \leftarrow (\mathbf{w}_U \cdot \mathbf{w}_V) \cdot \mathbf{\Sigma}_*$
$\mathbf{\Sigma}^* \leftarrow (\mathbf{w}_U \cdot \mathbf{w}_V) \cdot \mathbf{\Sigma}^*$
return $\mathbf{U}_{\text{avg}}, \ddot{\mathbf{\Sigma}}, \mathbf{V}_{\text{avg}}$
else if $optDT =$ 'c' **then**
$\mathbf{U}_{\text{avg}} \leftarrow \frac{\mathbf{U}_* + \mathbf{U}^*}{2}$
$\mathbf{\Sigma}_{\text{avg}} \leftarrow \frac{\mathbf{\Sigma}_* + \mathbf{\Sigma}^*}{2}$
$\mathbf{V}_{\text{avg}} \leftarrow \frac{\mathbf{V}_* + \mathbf{V}^*}{2}$
$\mathbf{U}_{\text{avg}}, \mathbf{w}_U \leftarrow$ NORM-MAT$(\mathbf{U}_{\text{avg}})$ ▷ See Algorithm 5
$\mathbf{V}_{\text{avg}}, \mathbf{w}_V \leftarrow$ NORM-MAT$(\mathbf{V}_{\text{avg}})$
$\mathbf{\Sigma}_{\text{avg}} \leftarrow (\mathbf{w}_U \cdot \mathbf{w}_V) \cdot \mathbf{\Sigma}_{\text{avg}}$
return $\mathbf{U}_{\text{avg}}, \mathbf{\Sigma}_{\text{avg}}, \mathbf{V}_{\text{avg}}$
end if
end procedure

A.3.4 ISVD$_3$: Decompose, Align, Solve

The ISVD$_3$ algorithm is described in Section 4.4.4 and is implemented as follows.

Algorithm 10 ISVD_3

Input:
Interval-valued matrix $\ddot{\mathbf{M}} = [\mathbf{M}_*, \mathbf{M}^*] \in \mathbb{R}^{n \times m}$, target rank r, decomposition target option $optDT$, condition number threshold $condThr$.

Output:
Decomposition target option (a):
Interval-valued factor matrices $\ddot{\mathbf{U}} \in \mathbb{R}^{n \times r}, \ddot{\mathbf{V}} \in \mathbb{R}^{m \times r}$ and interval-valued singular matrix $\ddot{\mathbf{\Sigma}} \in \mathbb{R}^{r \times r}$.

Decomposition target option (b):
Scalar-valued factor matrices $\mathbf{U}_{\text{avg}} \in \mathbb{R}^{n \times r}, \mathbf{V}_{\text{avg}} \in \mathbb{R}^{m \times r}$ and interval-valued singular matrix $\ddot{\mathbf{\Sigma}} \in \mathbb{R}^{r \times r}$.

Decomposition target option (c):
Scalar-valued factor matrices $\mathbf{U}_{\text{avg}} \in \mathbb{R}^{n \times r}, \mathbf{V}_{\text{avg}} \in \mathbb{R}^{m \times r}$ and scalar-valued singular matrix $\mathbf{\Sigma}_{\text{avg}} \in \mathbb{R}^{r \times r}$.

1: **procedure** $\text{ISVD}_3(\ddot{\mathbf{M}}, r, optDT, condTh)$
2: $\quad\ddot{\mathbf{A}} \leftarrow \text{INTERVAL-MAT-MULT}(\ddot{\mathbf{M}}^{\text{T}}, \ddot{\mathbf{M}})$ ▷ See Algorithm 1
3: $\quad\mathbf{V}_*, \mathbf{\Sigma}_*^2 \leftarrow \text{EIG}(A_*, r)$ ▷ Eigen Decomposition
4: $\quad\mathbf{V}^*, \mathbf{\Sigma}^{*2} \leftarrow \text{EIG}(A^*, r)$
5: $\quad$**mappingIdx**, **dirFlag** $\leftarrow \text{ILSA}(\mathbf{V}_*, \mathbf{V}^*)$ ▷ See Algorithm 6
6:
7: $\quad$**for** $j \leftarrow 1$ *to* r **do** ▷ Apply remapping
8: $\quad\quad\mathbf{V}_{*[:,j]} \leftarrow \mathbf{V}_{*[:,\textbf{mappingIdx}_j]}$
9: $\quad\quad\mathbf{\Sigma}_{*[j,j]} \leftarrow \mathbf{\Sigma}_{*[j,\textbf{mappingIdx}_j]}$
10: $\quad\quad$**if dirFlag**$_j = 1$ **then** ▷ Swap misaligned vectors
11: $\quad\quad\quad\mathbf{V}_{*[:,j]} \leftarrow \mathbf{V}_{*[:,j]} \cdot (-1)$
12: $\quad\quad$**end if**
13: $\quad$**end for**
14:
15: $\quad\mathbf{V}_{\text{avg}} \leftarrow \frac{\mathbf{V}_* + \mathbf{V}^*}{2}$
16: $\quad condNum \leftarrow \text{COND}(\mathbf{V}_{\text{avg}})$ ▷ Condition Number
17: $\quad n, m \leftarrow \text{SIZE}(\mathbf{V}_{\text{avg}})$
18: $\quad$**if** $condNum > condThr \vee n \neq m$ **then**

$\mathbf{V}_{\text{avg}}^{\text{inv}} \leftarrow \text{PINV}(\mathbf{V}_{\text{avg}})$ ▷ Moore-Penrose pseudo-inverse
else
$\mathbf{V}_{\text{avg}}^{\text{inv}} \leftarrow (\mathbf{V}_{\text{avg}})^{-1}$
end if
$\mathbf{\Sigma}^{\text{inv}} \leftarrow \text{INVERSE-}\Sigma(\mathbf{\Sigma}_*, \mathbf{\Sigma}^*)$ ▷ See Algorithm 4
$\ddot{\mathbf{U}} \leftarrow \text{INTERVAL-MAT-MULT}(\ddot{\mathbf{M}}, (\mathbf{V}_{\text{avg}}^{\text{inv}} \cdot \mathbf{\Sigma}^{\text{inv}}))$ ▷ See Algorithm 1

$\ddot{\mathbf{U}} \leftarrow \text{AVERAGE-REPLACEMENT-MAT}(\ddot{\mathbf{U}})$ ▷ See Algorithms 3 and 2
$\ddot{\mathbf{V}} \leftarrow \text{AVERAGE-REPLACEMENT-MAT}(\ddot{\mathbf{V}})$
$\text{diag}(\ddot{\mathbf{\Sigma}}) \leftarrow \text{AVERAGE-REPLACEMENT-VEC}(\text{diag}(\ddot{\mathbf{\Sigma}}))$

if $optDT =$ 'a' **then**
return $\ddot{\mathbf{U}}, \ddot{\mathbf{\Sigma}}, \ddot{\mathbf{V}}$
else if $optDT =$ 'b' **then**
$\mathbf{U}_{\text{avg}} \leftarrow \frac{\mathbf{U}_* + \mathbf{U}^*}{2}$
$\mathbf{V}_{\text{avg}} \leftarrow \frac{\mathbf{V}_* + \mathbf{V}^*}{2}$
$\mathbf{U}_{\text{avg}}, \mathbf{w}_U \leftarrow \text{NORM-MAT}(\mathbf{U}_{\text{avg}})$ ▷ See Algorithm 5
$\mathbf{V}_{\text{avg}}, \mathbf{w}_V \leftarrow \text{NORM-MAT}(\mathbf{V}_{\text{avg}})$
$\mathbf{\Sigma}_* \leftarrow (\mathbf{w}_U \cdot \mathbf{w}_V) \cdot \mathbf{\Sigma}_*$
$\mathbf{\Sigma}^* \leftarrow (\mathbf{w}_U \cdot \mathbf{w}_V) \cdot \mathbf{\Sigma}^*$
return $\mathbf{U}_{\text{avg}}, \ddot{\mathbf{\Sigma}}, \mathbf{V}_{\text{avg}}$
else if $optDT =$ 'c' **then**
$\mathbf{U}_{\text{avg}} \leftarrow \frac{\mathbf{U}_* + \mathbf{U}^*}{2}$
$\mathbf{\Sigma}_{\text{avg}} \leftarrow \frac{\mathbf{\Sigma}_* + \mathbf{\Sigma}^*}{2}$
$\mathbf{V}_{\text{avg}} \leftarrow \frac{\mathbf{V}_* + \mathbf{V}^*}{2}$
$\mathbf{U}_{\text{avg}}, \mathbf{w}_U \leftarrow \text{NORM-MAT}(\mathbf{U}_{\text{avg}})$ ▷ See Algorithm 5
$\mathbf{V}_{\text{avg}}, \mathbf{w}_V \leftarrow \text{NORM-MAT}(\mathbf{V}_{\text{avg}})$
$\mathbf{\Sigma}_{\text{avg}} \leftarrow (\mathbf{w}_U \cdot \mathbf{w}_V) \cdot \mathbf{\Sigma}_{\text{avg}}$
return $\mathbf{U}_{\text{avg}}, \mathbf{\Sigma}_{\text{avg}}, \mathbf{V}_{\text{avg}}$
end if
end procedure

A.3.5 ISVD$_4$: Decompose, Align, Solve, Recompute

The ISVD$_4$ algorithm is described in Section 4.4.5 and is implemented as follows.

Algorithm 11 ISVD$_4$

Input:
Interval-valued matrix $\ddot{\mathbf{M}} = [\mathbf{M}_*, \mathbf{M}^*] \in \mathbb{R}^{n \times m}$, target rank r, decomposition target option $optDT$, condition number threshold $condThr$.

Output:
Decomposition target option (a):
Interval-valued factor matrices $\ddot{\mathbf{U}} \in \mathbb{R}^{n \times r}, \ddot{\mathbf{V}} \in \mathbb{R}^{m \times r}$ and interval-valued singular matrix $\ddot{\mathbf{\Sigma}} \in \mathbb{R}^{r \times r}$.

Decomposition target option (b):
Scalar-valued factor matrices $\mathbf{U}_{\text{avg}} \in \mathbb{R}^{n \times r}, \mathbf{V}_{\text{avg}} \in \mathbb{R}^{m \times r}$ and interval-valued singular matrix $\ddot{\mathbf{\Sigma}} \in \mathbb{R}^{r \times r}$.

Decomposition target option (c):
Scalar-valued factor matrices $\mathbf{U}_{\text{avg}} \in \mathbb{R}^{n \times r}, \mathbf{V}_{\text{avg}} \in \mathbb{R}^{m \times r}$ and scalar-valued singular matrix $\mathbf{\Sigma}_{\text{avg}} \in \mathbb{R}^{r \times r}$.

procedure ISVD$_4$($\ddot{\mathbf{M}}, r, optDT, condTh$)
  $\ddot{\mathbf{A}} \leftarrow$ INTERVAL-MAT-MULT$(\ddot{\mathbf{M}}^{\mathrm{T}}, \ddot{\mathbf{M}})$ ▷ See Algorithm 1
  $\mathbf{V}_*, \mathbf{\Sigma}_*^2 \leftarrow \text{EIG}(A_*, r)$ ▷ Eigen Decomposition
  $\mathbf{V}^*, \mathbf{\Sigma}^{*2} \leftarrow \text{EIG}(A^*, r)$
  mappingIdx, **dirFlag** $\leftarrow$ ILSA$(\mathbf{V}_*, \mathbf{V}^*)$ ▷ See Algorithm 6

  for $j \leftarrow 1$ *to* r **do** ▷ Apply remapping
    $\mathbf{V}_{*[:,j]} \leftarrow \mathbf{V}_{*[:,\textbf{mappingIdx}_j]}$
    $\mathbf{\Sigma}_{*[j,j]} \leftarrow \mathbf{\Sigma}_{*[j,\textbf{mappingIdx}_j]}$
    if dirFlag$_j = 1$ **then** ▷ Swap misaligned vectors
      $\mathbf{V}_{*[:,j]} \leftarrow \mathbf{V}_{*[:,j]} \cdot (-1)$
    end if
  end for

  $\mathbf{V}_{\text{avg}} \leftarrow \frac{\mathbf{V}_* + \mathbf{V}^*}{2}$
  $condNum \leftarrow \text{COND}(\mathbf{V}_{\text{avg}})$ ▷ Condition Number
  $n, m \leftarrow \text{SIZE}(\mathbf{V}_{\text{avg}})$

18: **if** $condNum > condThr \vee n \neq m$ **then**
19: $\quad \mathbf{V}_{\text{avg}}^{\text{inv}} \leftarrow \text{PINV}(\mathbf{V}_{\text{avg}})$ ▷ Moore-Penrose pseudo-inverse
20: **else**
21: $\quad \mathbf{V}_{\text{avg}}^{\text{inv}} \leftarrow \mathbf{V}_{\text{avg}}^{-1}$
22: **end if**
23: $\boldsymbol{\Sigma}^{\text{inv}} \leftarrow \text{INVERSE-}\Sigma(\boldsymbol{\Sigma}_*, \boldsymbol{\Sigma}^*)$ ▷ See Algorithm 4
24: $\ddot{\mathbf{U}} \leftarrow \text{INTERVAL-MAT-MULT}(\ddot{\mathbf{M}}, (\mathbf{V}_{\text{avg}}^{\text{inv}} \cdot \boldsymbol{\Sigma}^{\text{inv}}))$ ▷ See Algorithm 1
25:
26: $\mathbf{U}_{\text{avg}} \leftarrow \frac{\mathbf{U}_* + \mathbf{U}^*}{2}$
27: $condNum \leftarrow \text{COND}(\mathbf{U}_{\text{avg}})$ ▷ Condition Number
28: $n, m \leftarrow \text{SIZE}(\mathbf{U}_{\text{avg}})$
29: **if** $condNum > condThr \vee n \neq m$ **then**
30: $\quad \mathbf{U}_{\text{avg}}^{\text{inv}} \leftarrow \text{PINV}(\mathbf{U}_{\text{avg}})$ ▷ Moore-Penrose pseudo-inverse
31: **else**
32: $\quad \mathbf{U}_{\text{avg}}^{\text{inv}} \leftarrow (\mathbf{U}_{\text{avg}})^{-1}$
33: **end if**
34: $\ddot{\mathbf{V}} \leftarrow \left[\text{INTERVAL-MAT-MULT}((\boldsymbol{\Sigma}^{\text{inv}} \cdot \mathbf{V}_{\text{avg}}^{\text{inv}}), \ddot{\mathbf{M}})\right]^{\text{T}}$ ▷ See Algorithm 1
35:
36: $\ddot{\mathbf{U}} \leftarrow \text{AVERAGE-REPLACEMENT-MAT}(\ddot{\mathbf{U}})$ ▷ See Algorithms 3 and 2
37: $\ddot{\mathbf{V}} \leftarrow \text{AVERAGE-REPLACEMENT-MAT}(\ddot{\mathbf{V}})$
38: $\text{diag}(\ddot{\boldsymbol{\Sigma}}) \leftarrow \text{AVERAGE-REPLACEMENT-VEC}(\text{diag}(\ddot{\boldsymbol{\Sigma}}))$
39:
40: **if** $optDT =$ 'a' **then**
41: $\quad$ **return** $\ddot{\mathbf{U}}, \ddot{\boldsymbol{\Sigma}}, \ddot{\mathbf{V}}$
42: **else if** $optDT =$ 'b' **then**
43: $\quad \mathbf{U}_{\text{avg}} \leftarrow \frac{\mathbf{U}_* + \mathbf{U}^*}{2}$
44: $\quad \mathbf{V}_{\text{avg}} \leftarrow \frac{\mathbf{V}_* + \mathbf{V}^*}{2}$
45: $\quad \mathbf{U}_{\text{avg}}, \mathbf{w}_U \leftarrow \text{NORM-MAT}(\mathbf{U}_{\text{avg}})$ ▷ See Algorithm 5
46: $\quad \mathbf{V}_{\text{avg}}, \mathbf{w}_V \leftarrow \text{NORM-MAT}(\mathbf{V}_{\text{avg}})$
47: $\quad \boldsymbol{\Sigma}_* \leftarrow (\mathbf{w}_U \cdot \mathbf{w}_V) \cdot \boldsymbol{\Sigma}_*$
48: $\quad \boldsymbol{\Sigma}^* \leftarrow (\mathbf{w}_U \cdot \mathbf{w}_V) \cdot \boldsymbol{\Sigma}^*$
49: $\quad$ **return** $\mathbf{U}_{\text{avg}}, \ddot{\boldsymbol{\Sigma}}, \mathbf{V}_{\text{avg}}$
50: **else if** $optDT =$ 'c' **then**
51: $\quad \mathbf{U}_{\text{avg}} \leftarrow \frac{\mathbf{U}_* + \mathbf{U}^*}{2}$
52: $\quad \boldsymbol{\Sigma}_{\text{avg}} \leftarrow \frac{\boldsymbol{\Sigma}_* + \boldsymbol{\Sigma}^*}{2}$
53: $\quad \mathbf{V}_{\text{avg}} \leftarrow \frac{\mathbf{V}_* + \mathbf{V}^*}{2}$

54: $\mathbf{U}_{\text{avg}}, \mathbf{w}_U \leftarrow$ NORM-MAT$(\mathbf{U}_{\text{avg}})$ ▷ See Algorithm 5
55: $\mathbf{V}_{\text{avg}}, \mathbf{w}_V \leftarrow$ NORM-MAT$(\mathbf{V}_{\text{avg}})$
56: $\mathbf{\Sigma}_{\text{avg}} \leftarrow (\mathbf{w}_U \cdot \mathbf{w}_V) \cdot \mathbf{\Sigma}_{\text{avg}}$
57: **return** $\mathbf{U}_{\text{avg}}, \mathbf{\Sigma}_{\text{avg}}, \mathbf{V}_{\text{avg}}$
58: **end if**
59: **end procedure**

A.4 Pseudocodes for Reconstruction Algorithms

Here we provide the pseudocode for the matrix reconstruction process and its variations according to which decomposition target has been selected by the user (as described in Section 4.3 and summarized in Figure 4.8).

A.4.1 Reconstruction Target (a): Interval-valued $\ddot{\mathbf{U}}$, $\ddot{\mathbf{\Sigma}}$, $\ddot{\mathbf{V}}$

Algorithm 12 Reconstruction Target (a)

Input:
Interval-valued factor matrices $\ddot{\mathbf{U}} \in \mathbb{R}^{n \times r}, \ddot{\mathbf{V}} \in \mathbb{R}^{m \times r}$ and interval-valued singular matrix $\ddot{\mathbf{\Sigma}} \in \mathbb{R}^{r \times r}$.

Output:
Interval-valued matrix $\widetilde{\ddot{\mathbf{M}}} = \left[\widetilde{\mathbf{M}}_*, \widetilde{\mathbf{M}}^*\right] \in \mathbb{R}^{n \times m}$.

1: **procedure** TARGET$_a$($\ddot{\mathbf{U}}$, $\ddot{\mathbf{\Sigma}}$, $\ddot{\mathbf{V}}$)
2: $\ddot{\mathbf{T}} \leftarrow$ INTERVAL-MAT-MULT$(\ddot{\mathbf{U}}, \ddot{\mathbf{\Sigma}})$ ▷ See Algorithm 1
3: $\widetilde{\ddot{\mathbf{M}}} \leftarrow$ INTERVAL-MAT-MULT$(\ddot{\mathbf{T}}, \ddot{\mathbf{V}}^{\mathrm{T}})$
4: $\widetilde{\ddot{\mathbf{M}}} \leftarrow \ddot{\mathbf{U}} \cdot \ddot{\mathbf{\Sigma}} \cdot \ddot{\mathbf{V}}^{\mathrm{T}}$
5: **return** $\widetilde{\ddot{\mathbf{M}}}$
6: **end procedure**

A.4.2 Reconstruction Target (b): Scalar U, $\ddot{\mathbf{V}}$, Interval $\ddot{\mathbf{\Sigma}}$

Algorithm 13 Reconstruction Target (b)

Input:

Scalar-valued factor matrices $\mathbf{U}_{\text{avg}} \in \mathbb{R}^{n \times r}$, $\mathbf{V}_{\text{avg}} \in \mathbb{R}^{m \times r}$ and interval-valued singular matrix $\ddot{\mathbf{\Sigma}} \in \mathbb{R}^{r \times r}$.

Output:
Interval-valued matrix $\tilde{\ddot{\mathbf{M}}} = \left[\widetilde{\mathbf{M}}_*, \widetilde{\mathbf{M}}^*\right] \in \mathbb{R}^{n \times m}$.

1: **procedure** TARGET$_b$($\mathbf{U}_{\text{avg}}$, $\ddot{\mathbf{\Sigma}}$, $\mathbf{V}_{\text{avg}}$)
2: $\quad \widetilde{\mathbf{M}}_* \leftarrow \mathbf{U}_{\text{avg}} \cdot \mathbf{\Sigma}_* \cdot \mathbf{V}_{\text{avg}}^{\text{T}}$
3: $\quad \widetilde{\mathbf{M}}^* \leftarrow \mathbf{U}_{\text{avg}} \cdot \mathbf{\Sigma}^* \cdot \mathbf{V}_{\text{avg}}^{\text{T}}$
4: $\quad \tilde{\ddot{\mathbf{M}}} = \left[\widetilde{\mathbf{M}}_*, \widetilde{\mathbf{M}}^*\right]$
5: $\quad \tilde{\ddot{\mathbf{M}}} \leftarrow$ AVERAGE-REPLACEMENT-MAT$(\ddot{\mathbf{M}})$ ▷ See Algorithm 3
6: $\quad$ **return** $\tilde{\ddot{\mathbf{M}}}$
7: **end procedure**

A.4.3 Reconstruction Target (c): Scalar U, Σ, *V*

Algorithm 14 Reconstruction Target (c)

Input:
Scalar-valued factor matrices $\mathbf{U}_{\text{avg}} \in \mathbb{R}^{n \times r}$, $\mathbf{V}_{\text{avg}} \in \mathbb{R}^{m \times r}$ and scalar-valued singular matrix $\mathbf{\Sigma}_{\text{avg}} \in \mathbb{R}^{r \times r}$.

Output:
Scalar-valued matrix $\widetilde{\mathbf{M}}_{\text{avg}} \in \mathbb{R}^{n \times m}$.

1: **procedure** TARGET$_c$($\mathbf{U}_{\text{avg}}$, $\mathbf{\Sigma}_{\text{avg}}$, $\mathbf{V}_{\text{avg}}$)
2: $\quad \widetilde{\mathbf{M}}_{\text{avg}} \leftarrow \mathbf{U}_{\text{avg}} \cdot \mathbf{\Sigma}_{\text{avg}} \cdot \mathbf{V}_{\text{avg}}^{\text{T}}$
3: $\quad$ **return** $\widetilde{\mathbf{M}}_{\text{avg}}$
4: **end procedure**

A.5 Aligned Interval Probabilistic Matrix Factorization (AI-PMF)

As discussed in Section 4.5, the aligned interval probabilistic matrix factorization (AI-PMF) algorithm leverages the ILSA technique to obtain an interval-valued latent semantic space to help reduce the negative impact of interval-valued projec-

tions. Below, we provide the pseudocode of the AI-PMF algorithm.

Algorithm 15 AI-PMF

Input:
Interval-valued matrix $\ddot{\mathbf{M}} = [\mathbf{M}_*, \mathbf{M}^*] \in \mathbb{R}^{n\times m}$, target rank r, number of epoch $nEpoch$ and number of batches $nBatch$.

Output:
Scalar-valued factor matrix $\mathbf{U} \in \mathbb{R}^{n\times r}$ and interval-valued factor matrix $\ddot{\mathbf{V}} \in \mathbb{R}^{m\times r}$.

procedure AI-PMF($\ddot{\mathbf{M}}, r, nEpoch, nBatch$)
  Initialize random $\mathbf{U} \in \mathbb{R}^{n\times r}$
  Initialize random $\ddot{\mathbf{V}} = [\mathbf{V}_*, \mathbf{V}^*] \in \mathbb{R}^{m\times r}$
  $batchSize \leftarrow \frac{n}{nBatch}$
  for $e \leftarrow 1$ *to* $nEpoch$ **do**
    Shuffle and divide row vectors in $\ddot{\mathbf{M}}$ into $nBatch$ batches.
    for *each batch* $\mathbf{B} \in \mathbb{R}^{batchSize\times m}$ **do**
      for $\mathbf{reIdx}_i \leftarrow 1$ *to* $batchSize$ **do**
        $i \leftarrow$ row index in $\ddot{\mathbf{M}}$ mapping to row index $\mathbf{reIdx}_i$ in $\mathbf{B}$
        for $\mathbf{reIdx}_j \leftarrow 1$ *to* m **do**
          $j \leftarrow$ column ind. in $\ddot{\mathbf{M}}$ mapping to column ind. $\mathbf{reIdx}_j$ in $\mathbf{B}$
          $\mathbf{U}_{[i,:]} = \mathbf{U}_{[i,:]} - \frac{\partial \mathcal{L}_{I-PMF}}{\partial \mathbf{U}_{[i,:]}}$
          ${\mathbf{V}_*}_{[j,:]}^{\mathrm{T}} = {\mathbf{V}_*}_{[j,:]}^{\mathrm{T}} - \frac{\partial \mathcal{L}_{I-PMF}}{\partial {\mathbf{V}_*}_{[j,:]}^{\mathrm{T}}}$
          ${\mathbf{V}^*}_{[j,:]}^{T} = {\mathbf{V}^*}_{[j,:]}^{T} - \frac{\partial \mathcal{L}_{I-PMF}}{\partial {\mathbf{V}^*}_{[j,:]}^{T}}$
        end for
      end for
    end for
  end for
  $\mathbf{mappingIdx}, \mathbf{dirFlag} \leftarrow \mathrm{ILSA}(\mathbf{V}_*, \mathbf{V}^*)$ ▷ See Algorithm 6

  for $j \leftarrow 1$ *to* r **do** ▷ Apply remapping
    $\mathbf{V}_{*[:,j]} \leftarrow \mathbf{V}_{*[:,\mathbf{mappingIdx}_j]}$
    if $\mathbf{dirFlag}_j = 1$ **then** ▷ Swap misaligned vectors
      $\mathbf{V}_{*[:,j]} \leftarrow \mathbf{V}_{*[:,j]} \cdot (-1)$
    end if

end for
return $\mathbf{U}, \ddot{\mathbf{V}}$
end procedure

Bibliography

[1] M.W. Berry and D.I. Martin. Component analysis for information retrieval. in handbook of parallel computing and statistics., 2004.

[2] L. Billard. and E. Diday, editors. *Symbolic Data Analysis: Conceptual Statistics and Data Mining*. Wiley, 2007.

[3] L. Billard and J. Le-Rademacher. Principal component analysis for interval data. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4(6):535–540, 2012.

[4] S. S. Bucak and B. Gunsel. Video content representation by incremental non-negative matrix factorization. In *2007 IEEE International Conference on Image Processing*, volume 2, pages II – 113–II – 116, Sept 2007.

[5] R. Burkard, M. Dell'Amico, and S. Martello. *Assignment Problems*. Society for Industrial and Applied Mathematics, 2012.

[6] K. Selçuk Candan, Huiping Cao, Yan Qi, and Maria Luisa Sapino. Alphasum: size-constrained table summarization using value lattices. In *EDBT*, pages 96–107, 2009.

[7] Huiping Cao, K. Selçuk Candan, and Maria Luisa Sapino. Skynets: searching for minimum trees in graphs with incomparable edge weights. In *CIKM*, pages 1775–1784, 2011.

[8] J. Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "Eckart-Young" decomposition. *Psychometrika*, 35(3):283–319, September 1970.

[9] Xilun Chen and K. Selçuk Candan. Gi-nmf: Group incremental non-negative matrix factorization on data streams. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, page 1119–1128, New York, NY, USA, 2014. Association for Computing Machinery.

[10] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, New York, NY, USA, 2006.

[11] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[12] Vin De Silva and Lek-Heng Lim. Tensor rank and the ill-posedness of the best low-rank approximation problem. *SIAM Journal on Matrix Analysis and Applications*, 30(3):1084–1127, 2008.

[13] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[14] Assem Deif. The interval eigenvalue problem. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, 71(1):61–64, 1991.

[15] Edwin Diday. Principal component analysis for bar charts and metabins tables. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 6(5):403–430, 2013.

[16] Edwin Diday. Thinking by classes in data science: the symbolic data analysis paradigm. *Wiley Interdisciplinary Reviews: Computational Statistics*, 8(5):172–205, 2016.

[17] A. Douzal-Chouakria, L. Billard, and E. Diday. Principal component analysis for interval-valued observations. *Statistical Analysis and Data Mining*, 4(2):229–246, 2011.

[18] Federica Gioia and Carlo N. Lauro. Principal component analysis on interval data. *Computational Statistics*, 21(2):343–363, Jun 2006.

[19] Paolo Giordani. Three-way analysis of imprecise data. *Journal of multivariate analysis*, 101(3):568–582, 2010.

[20] Paolo Giordani. Lasso-constrained regression analysis for interval-valued data. *Advances in Data Analysis and Classification*, 9(1):5–19, Mar 2015.

[21] P.J.F. Groenen, S. Winsberg, O. RodrÃguez, and E. Diday. I-scal: Multidimensional scaling of interval dissimilarities. *Computational Statistics and Data Analysis*, 51(1):360 – 378, 2006.

[22] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, December 2015.

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

[24] Richard A Harshman and Margaret E Lundy. Uniqueness proof for a family of models sharing features of tucker's three-mode factor analysis and parafac/-candecomp. *Psychometrika*, 61(1):133–154, 1996.

[25] Timothy Hickey, Qun Ju, and Maarten H Van Emden. Interval arithmetic: From principles to implementation. *Journal of the ACM (JACM)*, 48(5):1038–1068, 2001.

[26] Kazuo Iwama and Shuichi Miyazaki. A survey of the stable marriage problem and its variants. In *International conference on informatics education and research for knowledge-circulating society (ICKS 2008)*, pages 131–136. IEEE, 2008.

[27] ByungSoo Jeon, Inah Jeon, Lee Sael, and U Kang. Scout: Scalable coupled matrix-tensor factorization-algorithm and discoveries. In *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*, pages 811–822. IEEE, 2016.

[28] Roy Jonker and Anton Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340, 1987.

[29] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Rev.*, 51(3):455–500, August 2009.

[30] Tamara G. Kolda and Jimeng Sun. Scalable tensor decompositions for multi-aspect data mining. In *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008)*, 2008.

[31] Pieter M Kroonenberg and Jan De Leeuw. Principal component analysis of three-mode data by means of alternating least squares algorithms. *Psychometrika*, 45(1):69–97, 1980.

[32] N. Carlo Lauro, Rosanna Verde, and Antonio Irpino. *Generalized Canonical Analysis*, pages 313–330. John Wiley and Sons, Ltd, 2008.

[33] Daniel D. Lee and H. Sebastian Seung. Learning the parts of objects by nonnegative matrix factorization. *Nature*, 401:788–791, 1999.

[34] Mao-Lin Li, Francesco Di Mauro, K Selcuk Candan, and Maria Luisa Sapino. Matrix factorization with interval-valued data. *IEEE Transactions on Knowledge and Data Engineering*, 2019.

[35] Xinsheng Li, K Selçuk Candan, and Maria Luisa Sapino. ntd: noise-profile adaptive tensor decomposition. In *Proceedings of the 26th International Conference on World Wide Web*, pages 243–252, 2017.

[36] Sun Makosso-Kallyth and Edwin Diday. Adaptation of interval pca to symbolic histogram variables. *Advances in Data Analysis and Classification*, 6(2):147–159, Jul 2012.

[37] Monique Noirhomme-Fraiture and Paula Brito. Far beyond the classical data models: symbolic data analysis. *Statistical Analysis and Data Mining*, 4(2):157–170, 2011.

[38] I. V. Oseledets. Tensor-train decomposition. *SIAM J. Sci. Comput.*, 33(5):2295–2317, September 2011.

[39] Ivan V Oseledets, DV Savostianov, and Eugene E Tyrtyshnikov. Tucker dimensionality reduction of three-dimensional arrays in linear time. *SIAM Journal on Matrix Analysis and Applications*, 30(3):939–956, 2008.

[40] Beresford N. Parlett. The qr algorithm. *Computing in Science and Engg.*, 2(1):38–42, January 2000.

[41] Roger Penrose. A generalized inverse for matrices. In *Mathematical proceedings of the Cambridge philosophical society*, volume 51, pages 406–413. Cambridge University Press, 1955.

[42] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *NIPS*, pages 1257–1264, 2007.

[43] 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*, pages 138–142, Dec 1994.

[44] N. P. Seif, S. Hashem, and A. S. Deif. Bounding the eigenvectors for symmetric interval matrices. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift für Angewandte Mathematik und Mechanik*, 72(3):233–236, 1992.

[45] Fanhua Shang, Yuanyuan Liu, Hanghang Tong, James Cheng, and Hong Cheng. Robust bilinear factorization with missing and grossly corrupted observations. *Information Sciences*, 307:53–72, 2015.

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

[47] G. W. Stewart. On the early history of the singular value decomposition. 35(4):551–566, December 1993.

[48] J. Sun, S. Papadimitriou, and C. Faloutsos. Online latent variable detection in sensor networks. In *21st International Conference on Data Engineering (ICDE'05)*, pages 1126–1127, April 2005.

[49] Jimeng Sun, Spiros Papadimitriou, and S Yu Philip. Window-based tensor analysis on high-dimensional and multi-aspect streams. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 1076–1080. IEEE, 2006.

[50] Teruo Sunaga. Theory of an interval algebra and its application to numerical analysis. *Japan Journal of Industrial and Applied Mathematics*, 26(2):125–143, October 2009.

[51] Latanya Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5):571–588, October 2002.

[52] J. Tang, H. Gao, and H. Liu. mTrust: Discerning multi-faceted trust in a connected world. In *Proceedings of the fifth ACM international conference on Web search and data mining*, pages 93–102. ACM, 2012.

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

[54] Fei Wang, Hanghang Tong, and Ching-Yung Lin. Towards evolutionary nonnegative matrix factorization. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[55] Huiwen Wang, Rong Guan, and Junjie Wu. Cipca: Complete-information-based principal component analysis for interval-valued data. *Neurocomputing*, 86:158 – 169, 2012.

[56] Huiwen Wang, Rong Guan, and Junjie Wu. Linear regression of interval-valued data based on complete information in hypercubes. *Journal of Systems Science and Systems Engineering*, 21(4):422–442, Dec 2012.

[57] Liang Xiong, Xi Chen, Tzu-Kuo Huang, Jeff Schneider, and Jaime Carbonell. Temporal collaborative filtering with bayesian probabilistic tensor factorization. pages 211–222, 12 2010.

[58] Lotfi A Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

[59] Zheng (Zheng Alan) Zhao. *Spectral feature selection for data mining*. Chapman & Hall/CRC data mining and knowledge discovery series. CRC Press, Boca Raton, FL, 2012.