<table>
<thead>
<tr>
<th><strong>Matrix Factorization with Interval-Valued Data</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>This is the author's manuscript</strong></td>
</tr>
<tr>
<td><strong>Original Citation:</strong></td>
</tr>
<tr>
<td><strong>Availability:</strong></td>
</tr>
<tr>
<td>This version is available <a href="http://hdl.handle.net/2318/1726448">http://hdl.handle.net/2318/1726448</a> since 2020-02-03T22:20:21Z</td>
</tr>
<tr>
<td><strong>Published version:</strong></td>
</tr>
<tr>
<td>DOI:10.1109/TKDE.2019.2942310</td>
</tr>
<tr>
<td><strong>Terms of use:</strong></td>
</tr>
<tr>
<td>Open Access</td>
</tr>
<tr>
<td>Anyone can freely access the full text of works made available as &quot;Open Access&quot;. Works made available under a Creative Commons license can be used according to the terms and conditions of said license. Use of all other works requires consent of the right holder (author or publisher) if not exempted from copyright protection by the applicable law.</td>
</tr>
</tbody>
</table>

(Article begins on next page)
Matrix Factorization with Interval-Valued Data

<table>
<thead>
<tr>
<th>Journal:</th>
<th>Transactions on Knowledge and Data Engineering</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manuscript ID:</td>
<td>TKDE-2018-12-1264.R1</td>
</tr>
<tr>
<td>Manuscript Type:</td>
<td>Regular</td>
</tr>
<tr>
<td>Keywords:</td>
<td>Matrix factorization, Interval valued data, H.2.8.d Data mining &lt; H.2.8 Database Applications &lt; H.2 Database Management &lt; H Information Technology and Systems, H.2.8.i Mining methods and algorithms &lt; H.2.8 Database Applications &lt; H.2 Database Management &lt; H Information Technology and Sys, G.1.3.h Singular value decomposition &lt; G.1.3 Numerical Linear Algebra &lt; G.1 Numerical Analysis &lt; G Mathematics of Computing, I.2.3.l Uncertainty, &quot;fuzzy,&quot; and probabilistic reasoning &lt; H.5.2 User Interfaces &lt; H.5 Information Interfaces and Represent</td>
</tr>
</tbody>
</table>
Matrix Factorization with Interval-Valued Data

Mao-Lin Li, Francesco Di Mauro, K Selçuk Candan, and Maria Luisa Sapino

Abstract—With many applications relying on multi-dimensional datasets for decision making, matrix factorization (or decomposition) is becoming the basis for many knowledge discoveries and machine learning tasks, from clustering, trend detection, anomaly detection, to correlation analysis. Unfortunately, a major shortcoming of matrix analysis operations is that, despite their effectiveness when the data is scalar, these operations become difficult to apply in the presence of non-scalar data, as they are not designed for data that include non-scalar observations, such as intervals. Yet, in many applications, the available data are inherently non-scalar for various reasons, including imprecision in data collection, conflicts in aggregated data, data summarization, or privacy issues, where one is provided with a reduced, clustered, or intentionally noisy and obfuscated version of the data to hide information. In this paper, we propose matrix decomposition techniques that consider the existence of interval-valued data. We show that naive ways to deal with such imperfect data may introduce errors in analysis and present factorization techniques that are especially effective when the amount of imprecise information is large.

Index Terms—Matrix factorization, Interval valued data.

1 INTRODUCTION

With many machine-learning applications requiring latent semantics underlying the data sets, matrix factorization has emerged as a successful tool for discovering latent patterns in data [1], [2]: matrices are used to encode relationships among pairs of entities and data are analyzed for their latent semantics through matrix decomposition operations, such as singular value decomposition, SVD [3] or principal component analysis, PCA [4].

1.1 Challenge: Interval-Valued Data

In many applications, data need to be represented as ranges or intervals of possible values, as opposed to scalar data:

- **Summarized data.** Analyzing reduced or summarized data sets can be more efficient, especially for implementing interactive applications [5], [6]. When several observations are grouped and collapsed into a single observation, data may need to be represented as value ranges. While it may sometimes be possible to associate statistical meanings to the intervals and (assuming that appropriate generative models, probability distributions, and conditioning strategies are found) it might be possible to leverage probabilistic matrix factorization techniques, such as [7], this approach may be infeasible or ineffective due to the lack of appropriate statistical representations and/or the cost.

- **Data with conflicts.** When a data set reflects knowledge integrated from different data sources, it might not be possible to assign a single scalar weight to each observation and an interval of possible values might be a more appropriate representations [6]. Moreover, when analyzing such integrated data, the resulting intervals may not have a statistical interpretation, beyond presenting the spread (i.e., minimum and maximum values) of the data.

- **Anonymized data.** Various privacy-preserving data publishing algorithms, such as recoding techniques [8], replace precise scalar values with less precise value ranges or intervals (such as those obtained through value generalization [8]). The resulting intervals (intentionally) do not represent any specific data distribution; consequently, associating a statistical interpretation to the interval is not necessarily appropriate. This means that probabilistic techniques for data analysis are not appropriate for anonymized data sets. In Section 6, we see that the proposed approach is highly effective for interval-valued data generated through generalization.

- **Imprecise data.** Data imprecision may be caused by various reasons, including limitations in measurement. For example, minute variations in multiple facial images from a single individual may be represented using interval-valued data (see [9] and Section 6.1.2). As we further discuss in Section 6.1.3, ambiguities in users’ ratings in a collaborating filtering application may also be captured using interval-valued data [10], [11].

The key challenge in performing decompositions over interval-valued matrices is that definitions of basic algebraic operations, such as multiplication and inversion (needed to implement factorization operations), are not as straightforward for intervals as they are for scalars (see Section 2.1). Also, unlike scalars which are totally ordered, intervals are often partially ordered. Furthermore, a naive approach (which would exhaustively enumerate all possible decompositions) would significantly increase the computational complexity of the problem (which is already high for the case with scalar weights). Therefore, many basic operations need to be redefined to accommodate such non-scalar data and these need to be implemented in ways that avoid increases in computational costs.
1.2 Contributions of This Paper

In this paper, we study the problem of obtaining decompositions of interval-valued matrices:

- We present the decomposition problem for interval valued data sets. We introduce interval-valued algebra and discuss the core challenges presented by the decomposition of interval-valued data.
- We propose an interval-valued latent semantics alignment scheme and, relying on this, we develop algorithms for obtaining eigenvalue-based (such as SVD [3]) or probabilistic (such as PMF [7]) decomposition for interval-valued data matrices.
- We finally study the effectiveness of the proposed schemes in several applications, including face image analysis and collaborative filtering.

1.3 Organization of the Paper

The paper is organized as follows: We introduce the background and review the related work in Section 2. Section 3 introduces the problem and presents the key observations and mathematical formulations regarding interval-valued data. Section 4 presents the proposed interval singular value decomposition (ISVD) algorithm to obtain SVD decompositions in the presence of interval-valued data. Section 5 shows that the proposed semantic alignment technique can also be used in probabilistic matrix factorization scenarios. Section 6 reports our experimental results under diverse scenarios. We conclude the paper in Section 7.

2 BACKGROUND AND RELATED WORKS

2.1 Interval Algebra

We first formalize the definitions for interval-valued data and its algebraic operations:

**Definition 1 (Interval representation).** An interval \( a \) is a pair \( a = [a_-, a^+] \) where \( a_- \) is the minimum value and \( a^+ \) is the maximum value of the interval \( a \). If \( a_- = a^+ \), then \( a \) is scalar.

**Definition 2 (Interval span).** Given an interval \( a \), the corresponding span is computed by:

\[
\text{span}(a) = \text{span}([a_-, a^+]) = (a^+ - a_-) \in \mathbb{R}
\]

**Definition 3 (Interval algebraic operations).** We adopt the following interval-valued operations on two intervals \( [a, b]^* \) and \( [c, d]^* \):

- **Addition:** \( [a, b]^* + [c, d]^* = [a+c, b+d]^* \)
- **Subtraction:** \( [a, b]^* - [c, d]^* = [a-c, b-d]^* \)
- **Multiplication:** \( [a, b]^* \times [c, d]^* = [\min(\alpha \times b, a \times c, \alpha \times d, a \times \beta), \max(\alpha \times b, a \times c, \alpha \times d, a \times \beta)] \)

When one of the values, say \( a \), is scalar, the multiplication \( [a, b]^* \times [c, d]^* \) can be written as \( [\min(a \times b, a \times c, a \times d), \max(\alpha \times b, a \times c, a \times d)] \) and the corresponding value of the span is \( \text{span}(a \times [b, d]^*) = a \times \text{span}([b, d]^*) \).

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, and multiplication operations, with their interval-valued counterparts.

2.2 Matrix Factorization

Feature selection and dimensionality reduction techniques [13] 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. Among these transformations, Karhunen-Loeve Transform, KLT (also known as the principal component analysis, PCA [14]), and singular value decomposition, SVD [3] have the key property that the vectors selected as the dimensions of the space are mutually orthogonal and, hence, linearly independent (i.e., there is no redundancy among the dimensions). The resulting basis vectors are referred to as the latent variables [15] or the latent semantics of the data [3]. While KLT and SVD may result in negative values, in non-negative matrix factorization (NMF) [16], [17], factor matrices are non-negative and enable probabilistic interpretation of the results and discovery of generative models. Below, we outline three common matrix factorization schemes: singular value decomposition (SVD [3]), non-negative matrix factorization (NMF [16], [17]), and probabilistic matrix factorization (PMF [7]).

2.2.1 Singular Value Decomposition (SVD)

Let \( M \in \mathbb{R}^{n \times m} \) represent the input matrix. Let the rank, \( r \), be a positive integer \( r \leq \min(n, m) \). In this paper, we denote the value of \( i^{th} \) row and \( j^{th} \) column of \( M \) as \( M_{i,j} \).

The \( j^{th} \) column of \( M \) can be decomposed into \( M = UV^T \) through singular value decomposition (SVD), where

\[
U \in \mathbb{R}^{n \times r}, \text{ and } U^TU = I_r; \quad \Sigma \in \text{diag}([\sigma_1, \ldots, \sigma_r]); \quad V^T = \text{transpose}(V), V \in \mathbb{R}^{m \times r}, \text{ and } VV^T = I_m.
\]

The columns of \( U \), also called the left singular vectors of matrix \( M \), are the eigenvectors of the \( n \times n \) matrix, \( M^TM \). The columns of \( V \), or the right singular vectors of \( M \), are the eigenvectors of the \( m \times m \) matrix, \( MTM \). Note that both columns of \( U \) and columns of \( V \) are mutually orthogonal.

2.2.2 Non-negative Matrix Factorization (NMF)

Given a non-negative matrix \( M \in \mathbb{R}^{n \times m}_+ \), NMF factorizes \( M \) into two non-negative matrices \( U \in \mathbb{R}^{n \times r}_+ \) and \( V \in \mathbb{R}^{m \times r}_+ \) with target rank \( r \), which minimize the \( L_2 \) loss function

\[
\mathcal{L}_{\text{NMF}} = \|M - UV^T\|_F^2,
\]

where \( U \geq 0, V \geq 0 \) and \( \| \cdot \|_2^2 \) denotes the Frobenius norm. The approximated solutions for \( U \) and \( V \) are commonly found by iterative update rules, such as [17]

\[
U[i, j] \leftarrow U[i, j] \frac{(MV)[i, j]}{(U^TV)[i, j]} \quad \text{and} \quad V^T[i, j] \leftarrow V^T[i, j] \frac{(U^TM)[i, j]}{(U^TV)[i, j]}.
\]
[9] extended these to interval-valued matrices as follows:

\[ \mathcal{L}_{I-NMF} = \| M - UV^T \|_F^2 + \| M^* - UAV^T \|_F^2 \]

\[ U[i,j] \leftarrow U[i,j] \frac{(MV)[i,j]}{(UV^T V)[i,j]} \]

\[ V^T[i,j] \leftarrow V^T[i,j] \frac{(U^T M)[i,j]}{(U^T UV^T)[i,j]} \]

\[ V^*[i,j] \leftarrow V^*[i,j] \frac{(U^T M^*)[i,j]}{(U^T UV^*+[i,j])} \]

Note that this scheme, called I-NMF, factorizes the matrix into a scalar-valued \( U \) and an interval-valued \( V \).

### 2.2.3 Probabilistic Matrix Factorization (PMF)

Probabilistic matrix factorization (PMF) [7] assumes matrix entries are drawn from Gaussian distribution. In particular, probabilistic matrix factorization (PMF) \[7\] assumes matrix \( M \) is a product of two lower rank matrices \( U \) and \( V \), such that

\[ M[i,j] \sim N(U[i,:], V[j,:], \sigma^2) \]

where \( \mathcal{N} \) is the probability density function of Gaussian distribution with mean \( \mu \) and variance \( \sigma^2 \), \( Z_{ij} \) is the indicator function that is equal to 1 if \( M[i,j] \) is not null and 0 otherwise. Above, \( U[i,:] \) and \( V[j,:] \) are row vectors in \( U \) and \( V \), such that \( M[i,j] \approx U[i,:] V[j,:]^T \), and they place zero-mean spherical Gaussian priors on latent semantics \[2, 3\]. The factors, \( U \) and \( V \), are computed via the loss function

\[ \mathcal{L}_{PMF} = \| M - UV^T \|_F^2 + \lambda_U \| U \|_F^2 + \lambda_V \| V \|_F^2, \]

where \( \lambda_U = \sigma^2 / \sigma_U^2 \), \( \lambda_V = \sigma^2 / \sigma_V^2 \), and \( \| . \|_F \) denotes the Frobenius norm. A local minimum of loss function \( \mathcal{L}_{PMF} \) can be found via gradient descent in \( U[i,:) \) and \( V[j,:] \):

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

\[ \frac{\partial \mathcal{L}_{PMF}}{\partial V[j,:]} = \sum_{i=1}^{n} \left( (U[i,:]) V[j,:]^T - M[i,j] \right) U[i,:] + \lambda V[j,:]. \]

### 2.3 Analysis of Symbolic and Interval-valued Data

In the real world, data rarely comes in simple scalar form. Often variables of interest may take complex, often symbolic, forms, including sets, histograms, vectors, intervals, or probability distributions [18], [19], [20], [21]. This is especially true when data is aggregated [22] or anonymized [8]. Consequently, several data analysis tools, including regression [23], [24], canonical analysis [25], and multi-dimensional scaling [26], have been developed for symbolic and interval-valued data. Given the popularity of PCA in data analysis, several interval-valued PCA algorithms have also been proposed [27], [28], [29], [30], most of which leverage the specific statistical and geometric meanings of principal components of a system of variables. As discussed above, interval NMF and PMF [9] also have been studied to resolve alignment approximation in face analysis and rating approximation in collaborative filtering. In contrast, we develop a more general interval-valued latent semantic alignment algorithm which can be integrated in common matrix factorization approaches that directly leverages interval-valued properties.

### 3 INTERVAL-VALUED LATENT SPACES

In Section 2.2.1, we have seen that a scalar matrix can be decomposed into factor matrices (\( U \) and \( V \)) and core matrix (\( X \)). The columns in the factor matrices are referred to as latent semantics (LS) and, preferably, they are mutually orthogonal to serve as basis of the transformed space. Figure 1 shows a scalar-valued latent semantic space superimposed on the original space; the figure also shows a scalar-valued data point projected onto both original and latent spaces. Unfortunately, scalar-valued latent spaces are not sufficient to present interval-valued data.

#### 3.1 Interval-Valued Decomposition

Here, we first extend the definition of singular valued decomposition taking into account the presence of interval-valued data.

**Definition 4 (Interval-Valued Decomposition).** Given an interval-valued matrix, \( M_i \in \mathbb{R}^{n \times m} \), and a target rank \( r \leq \min(n, m) \), interval-valued SVD would decompose \( M_i \) into \( M_i \approx U_i \Sigma_i V_i^T \), such that

- \( U_i \in \mathbb{R}^{n \times r} \) and \( V_i \in \mathbb{R}^{m \times r} \) (potentially interval-valued) matrices, such that the columns of \( U_i \) and \( V_i \) are quasi-orthonormal; i.e., given column indexes \( h \) and \( l \),...
As we see in Figure 2(a), an interval-valued data entry has minimum and maximum values for each of the original basis vectors and can be considered as a (high-dimensional) box in the original space. As we also see in the figure, the transformation obtained through the decomposition process would represent each basis vector of the latent semantic space with an interval-valued column vector, \( V_i[h, i] \), bounded with a minimum and a maximum. Given this interval-valued latent space defined by the matrix \( V_i \), the projection of the data entry, defined by the row \( M_i[i, \cdot] \), onto the basis vector \( V_i[h, i] \) would be represented as an interval defined by \( U_i[i, h] \times \Sigma_i[h, h] \), where \( \times \) is an interval algebraic multiplication as defined in Section 2.1 (Figure 2(b)).

### 3.2 Imprecision of Interval-Valued Latent Spaces

Unlike in the basic SVD operation, we cannot seek for an exact decomposition of the input matrix \( M_i \), nor can we treat the constraint \( U_i \Sigma_i V_i^T = I \) as a hard constraint: The reason for both of these relaxations is implicit in the definition of interval-valued multiplication given in Section 2.1.

**Theorem 1** (Scalar Theorem for \( \times \)). Let \( a = [a_+, a_-] \) and \( b = [b_+, b_-] \) be two non-zero intervals. Let \( c = [c_+, c_-] \times b \), where \( \times \) is interval-valued matrix multiplication. Then,

\[
(c_+ = c_-) \rightarrow (a_+ = a_-) \land (b_+ = b_-).
\]

**Proof.** As we have seen in Section 2.1, \([a_+, a_-] \times [b_+, b_-] = [c_+, c_-] \times b \), where

\[
c_+ = \min(a_+ \times b_+, a_+ \times b_-, a_- \times b_+, a_- \times b_-),
\]

\[
c_- = \max(a_+ \times b_+, a_+ \times b_-, a_- \times b_+, a_- \times b_-).
\]

If, however, we are further given that \( c = a \times b \) is scalar, this implies that

\[
\min(a_+ \times b_+, a_+ \times b_-, a_- \times b_+, a_- \times b_-) = \max(a_+ \times b_+, a_+ \times b_-, a_- \times b_+, a_- \times b_-),
\]

which can only happen if either \( a \) or \( b \) is a zero interval or we have \( (a_+ = a_-) \land (b_+ = b_-) \).

In other words, the only way the multiplication of two interval values would return a scalar value is when both \([a_+, a_-]\) and \([b_+, b_-]\) themselves are scalars; i.e., \( a_+ = a_- \) and \( b_+ = b_- \). This further implies that the constraint \( U_i \Sigma_i V_i^T = I \) cannot be exactly satisfied.

**Theorem 2** (Scalar Theorem for Dot Product). Let \( x \) be a \( k \) dimensional interval-valued vector. Then the dot product of \( x \) with itself (i.e., \( xx \)) is scalar-valued only if all the entries in \( x \) are scalar-valued.

**Proof.** From the definition of dot product, we have

\[
x \cdot x = \sum_{i=1}^{k} x[i] \times x[i], \text{ where } x \in \mathbb{R}^k
\]

which, by definition of interval-valued matrix multiplication,

\[
x \cdot x = \sum_{i=1}^{k} [\min(x_+ [i]^2, x_- [i]^2), \max(x_+ [i]^2, x_- [i]^2)].
\]

Since both \( x_+ [i]^2 \) and \( x_- [i]^2 \) are non-negative, the only way the equality

\[
\sum_{i=1}^{k} \min(x_+ [i]^2, x_- [i]^2) = \sum_{i=1}^{k} \max(x_+ [i]^2, x_- [i]^2)
\]

holds is if for all \( i \), we have \( x_+ [i] = x_- [i] \).

Note that, as a corollary of above theorem, we cannot have \( XX^T \) scalar-valued unless \( X \) is scalar-valued.

**Corollary 1.** Let \( X \) be an interval-valued matrix and let \( S \) be a scalar-valued matrix. Then, the following is true:

\[
(XX^T = S) \rightarrow X = X^*.
\]

This is because for \( S \) to be scalar-valued, the diagonal of \( S \) should be scalar-valued. Since for all \( h \), we also have \( S[h] = X[h]X[h] \), then by Theorem 2, if \( S \) is scalar-valued than each and every row, \( X[h] \) (and consequently the entire matrix \( X \)) must also be scalar-valued.

**Corollary 2.** Corollary 1 further implies that \( U_i \Sigma_i V_i^T \) and \( V_i \Sigma_i^T \) cannot be equal to scalar-valued matrix, \( I \), which means that an exact decomposition of interval-valued matrices is not possible.

Given the inherent imprecision in interval-valued latent spaces, we define decomposition accuracy as follows:

**Definition 5** (Decomposition Accuracy). Given the above, let \( M_i \) be an interval-valued matrix, decomposed into \( U_i, \Sigma_i, \) and \( V_i^T \) and \( M_i \) be the interval-valued matrix obtained by

\[
\tilde{M}_i = U_i \Sigma_i V_i^T,
\]

recombining \( U_i, \Sigma_i, \) and \( V_i^T \) using the dot product operation, defined in terms of the interval algebraic operations introduced in Section 2.1. Let \( M_i, M_i^* \) be the scalar-valued matrices obtained by considering the minimum and maximum values of the intervals in \( M_i \), respectively. We measure the reconstruction accuracy in terms of the Frobenius norms of the difference matrices:

\[
\Delta(M_i, \tilde{M}_i) = \left( \frac{\|M_i - \tilde{M}_i\|}{\|M_i\|} \right); \Delta(M_i^*, \tilde{M}_i^*) = \left( \frac{\|M_i^* - \tilde{M}_i^*\|}{\|M_i^*\|} \right).
\]

To obtain a single measure of accuracy, we convert these into accuracies, \( \Theta(M_i, \tilde{M}_i) = \max(0, 1 - \Delta(M_i, \tilde{M}_i)) \) and \( \Theta(M_i^*, \tilde{M}_i^*) = \max(0, 1 - \Delta(M_i^*, \tilde{M}_i^*)) \), and combine them using Harmonic mean: \( \Theta_{HM}(M, \tilde{M}) = HarmonicMean(\Theta(M_i, \tilde{M}_i), \Theta(M_i^*, \tilde{M}_i^*)) \).

### 3.3 Interval-Valued Latent Semantic Alignment (ILSA)

Let \( V_i \) be an interval-valued factor matrix representing the latent semantics of the interval-valued data, \( M_i \). As we have discussed in Section 3.1 (and visualized in Figure 2(a)) we would expect that the columns of the \( V_i \) will be quasi-orthogonal (i.e., for all \( i \neq j \), \( V_i[i,j] V_j[i,j] \approx 0 \) and, for all \( i \), \( V_i[i,i] \approx 1 \) even though, as we have seen in Section 3.2, we cannot expect that \( V_i[i,i] = 0 \) or \( V_i[i,i] \approx 1 \). In fact, since per Theorem 2, the only way the dot product of a vector, \( x \), with itself is scalar-valued is when this vector itself is scalar-valued, the only way we can even approximate the quasi-orthonormality constraint is when for all \( i \), we
While it is known that this problem has an \( O(n^2) \) solution [31], this does not guarantee that the solution is also optimal in terms of maximizing overall alignments among the paired minimum and maximum vectors. Therefore, we consider an alternative formulation of the problem:

**Problem 2** (Optimal Min-Max Vector Alignment). Let us be given two sets of unit-length vectors, \( V_1 = \{ \vec{v}_1, \ldots, \vec{v}_r \} \) and \( V_2 = \{ \vec{v}_1^*, \ldots, \vec{v}_r^* \} \). Let us also be given the preference function between two vectors, \( x \) and \( y \), as \( \text{abs}(\cos(x, y)) \). Our goal is to find pairing/mapping \( \mu = \{ \mu_1, \ldots, \mu_r \} \) (where \( \mu_i = (\mu_{i1}, \ldots, \mu_{ir}) \)) among the vectors from \( V_1 \) and \( V_2 \), such that \( \sum_{1 \leq i \leq r} \text{abs}(\cos(\vec{v}_{\mu_{i1}}, \vec{v}_{\mu_{ir}})) \) is maximum, where \( \text{abs} \) means absolute value and \( \cos \) means cosine similarity.

This is an instance of the linear assignment problem [32] and can be solved using one of the many algorithms, such as the Hungarian algorithm with \( O(r^3) \) time complexity.

Once a mapping, \( \mu_i \), is identified, then for any \( 1 \leq i \leq r \) such that \( \cos(\vec{v}_{\mu_{i1}}, \vec{v}_{\mu_{ir}}^*) \) is negative, we can realign their directions by multiplying \( \vec{v}_{\mu_{ir}}^* \) with \(-1\) so that both \( \vec{v}_{\mu_{i1}} \) and \( \vec{v}_{\mu_{ir}}^* \) are pointing in a similar direction. As we see in Figure 3, the algorithm helps improve the alignment between the minimum and maximum basis vectors, especially for low rank entries which are often critical in many analysis and recommendation applications.

### 3.4 Alternative Decomposition Strategies

We note that, given an interval-valued matrix, \( M_{ij} \), different applications may seek different types of factor matrices and core matrix: some applications may need all of the resulting matrices be interval-valued, some others may need to capture the underlying impression in the form of interval-valued factor and/or core matrices, while others may require all matrices to be scalar. In particular, we consider three distinct application semantics:

**3.4.1 Decomp. Target (a): Interval-Valued \( U_1, \Sigma_1 \), and \( V_1 \)**

In the most general decomposition strategy, which we refer to as decomposition target-a (option-a), the application captures the impression in the data and represents it in terms of imprecise factor and core matrices. We construct interval-valued entries \( U_1, \Sigma_1 \), and \( V_1 \) by combining the corresponding entries from \( U_*, \Sigma_*, \) and form \( U^*, \Sigma^*, \) \( V^* \), respectively. More specifically, for all \( 1 \leq i \leq m \) and \( 1 \leq j \leq r \), we have

\[
U_1[i,j] = \begin{cases} 
(u_{i \mu_{i1}}[i], u_{i \mu_{ir}}^*[i]), & \text{if } u_{i \mu_{i1}}[i] \leq u_{i \mu_{ir}}^*[i] \\
\text{avg}(u_{i \mu_{i1}}[i], u_{i \mu_{ir}}^*[i]), & \text{otherwise}
\end{cases}
\]

In particular, when the minimum and maximum entries in \( U_1 \) and \( U^* \) do not lead to a valid interval, we replace the entry with the mean of the values to represent the best approximation we have for that entry. For all \( 1 \leq j \leq r \), \( \Sigma_1[i,j] \) and, for all \( 1 \leq i \leq m \) and \( 1 \leq j \leq r \), \( V_1[i,j] \) are computed similarly.

**3.4.2 Decomp. Target (b): Scalar \( U \) and \( V \); Interval \( \Sigma_1 \)**

One difficulty with the above construction is that the columns of the \( U_1 \) and \( V_1 \) matrices are interval-valued and, thus, it is difficult to interpret them as the alternative basis of a vector space. In fact, while for all \( 1 \leq j \leq r \), \( \vec{u}_{i1} \) and \( \vec{u}_{ir}^* \) are themselves unit length, the same cannot be said for the interval-valued vector \( \vec{u}_i \), since the norm operation is not defined for interval-valued vectors. Therefore, in decomposition target b (option-b), the factors matrices are maintained scalar-valued (i.e., precise), whereas the core which represents the relationships among the factors are interval-valued to capture the underlying impression. We first obtain scalar \( U \) by (a) first constructing an \( X \) matrix, such that, for all
1 ≤ i ≤ n and 1 ≤ j ≤ r, \( X[i, j] = \text{avg}(\bar{a}_{u,i}, \bar{a}_{d,i}) \) and then (b) renormalizing the columns of the \( \bar{X} \) matrix:
\[
U[i, j] = \frac{X[i, j]}{\sqrt{\sum_{1 \leq h \leq n} X[h, j]^2}}
\]

Similarly, the scalar \( V \) matrix is obtained by first constructing a \( Y \) matrix and renormalizing the columns of \( Y \). Intuitively, \( U \) and \( V \) matrices are scalar-valued matrices, both consisting of unit length columns. Given these, we then obtain the core matrix \( \Sigma[i, j] \) for 1 ≤ j ≤ r as
\[
\begin{cases}
\rho_j \times \{ \Sigma_l[m, j], \mu_{h,j}, \Sigma_h^*[m, h] \}, & \text{if } \Sigma_l[m, j] \leq \Sigma_h^*[m, h]\\
\rho_j \times \text{avg}(\Sigma_l[m, j], \Sigma_h^*[m, h]), & \text{otherwise}
\end{cases}
\]

denoted by \( \rho_j = \left( \sum_{1 \leq h \leq n} X[i, j] \right) \times \left( \sum_{1 \leq h \leq m} Y[i, j] \right) \). Intuitively, we rescale the non-zero values of the interval-valued core matrix \( \Sigma_l \) to account for the renormalization factors of the corresponding columns of \( U \) and \( V \).

Option-b leads to scalar-valued factor matrices, \( U \) and \( V \), and an interval-valued core matrix, \( \Sigma_l \). Intuitively, this provides alternative basis vectors for the rows and columns of the input matrix \( M \) and an interval-valued strength for each basis vector (captured by the corresponding entries of the diagonal matrix, \( \Sigma_l \)).

3.4.3 Decomp. Target (c): Scalar \( U, \Sigma_l, \) and \( V \)
As a third option, we can obtain scalar-valued approximation for \( \Sigma \) by (a) first renormalizing \( U \) and \( V \) in L2-Norm as in Section 3.4.2 and, then, (b) replacing each diagonal entry of \( \Sigma \) with the mean value of the corresponding interval. This approach, which we refer to as decomposition target-c (option-c), leads to a scalar-valued core matrix, \( \Sigma_l \), along with scalar-valued factor matrices, \( U \) and \( V \). In this case, despite the imperfection in the input matrix, the application requires scalar-valued factors and core (e.g. to be compatible with algorithms and tools that assume scalar-valued factor and core matrices to support recommendations).

4 INTERVAL SVD (ISVD)
In this section, we will introduce alternative decomposition strategies to obtain SVD of interval-valued matrices, as defined in Section 3.1. We start with a naive approach.

4.1 Naive Approach (ISVD0): Average and Decompose
As we discussed in Section 3.2, we are not seeking an exact decomposition of interval-valued matrices, but only an approximation. Therefore, the very first approach to consider for solving the decomposition problem is to simplify the input matrix into a simple scalar matrix by representing each interval by its average (see the column labeled ISVD0 in Figure 4): Let \( M[i, j] \) be an interval-valued entry \([a_s, a_e]\) in \( M \) and let the corresponding entry in the average matrix \( M_{avg}[i, j] = \frac{a_s + a_e}{2} \). We approximate the singular value decomposition of \( M \) with rank \( r \) as
\[
\text{ISVD}_0(M, r) \simeq \text{SVD}_0(M, r) = \text{SVD}(M_{avg}, r).
\]

This approach results in \( U \) and \( V \) matrices that are scalar and orthonormal and a \( \Sigma \) core matrix that is also scalar-valued (i.e., it is compatible only with the decomposition target-c, discussed in Section 3.4).

4.2 ISVD1: Decompose and Align
In this section, we note that, in order to support a quasi-orthogonal latent-semantics space visualized in Figure 2(a), the vectors in each of \( V_1 \) (and similarly in \( V_2 \)) must be as mutually orthogonal as possible. Therefore, we can obtain \( V_1 \) and \( V_2 \) by decomposing the minimum and maximum matrices, \( M_s \) and \( M_e \), of the given interval-valued matrix, \( M_t \), separately and aligning the resulting factor/core matrices: Let us split \( M_t \) into two scalar-valued matrices \( M_s \) and \( M_e \) such that if \( M_s[i, j] \) is an interval \([a_s, a_e]\), then \( M_s[i, j] = a_s \) and \( M_e[i, j] = a_e \). Given these, we can obtain two separate matrix decompositions of rank \( r \), \( \text{SVD}(M_s, r) = U_s \Sigma_s V_{s}^T \) and \( \text{SVD}(M_e, r) = U_e \Sigma_e V_{e}^T \).

4.2.1 Latent Semantic Alignment
Once we obtain \( U_1 = [U_s, U_e], \Sigma_1 = [\Sigma_s, \Sigma_e], V_1 = [V_s, V_e] \) after decomposition, we then apply interval latent semantic alignment (ILSA, Section 3.3) to align the eigenvectors in matrix \( V_1 \) and adjust the rank-order of the rows of \( \Sigma_1 \) and the rank-order and direction of the columns in \( U_1 \). This is visualized under the column labeled ISVD1 in Figure 4.

Note that in the discussion in Section 3.3, \( V_1 \) is taken as an interval matrix, where entries in \( V_e \) are smaller than the corresponding entries in \( V_s \) (similarly for \( \Sigma_1 \) and \( U_1 \)). This, however, does not necessarily hold in practice since \( \text{SVD}(M_s, r) = U_s \Sigma_s V_{s}^T \) and \( \text{SVD}(M_e, r) = U_e \Sigma_e V_{e}^T \) can lead to situations where \( V_s[i, j] > V_e[i, j] \). This, however, does not impact the latent semantic alignment process as the algorithm does not rely on \( V[i, j] \) being \( V_s[i, j] \) and only seeks an alignment where \( V_e[i, j] \approx V_s[i, j] \).

4.2.2 Construction of \( U_1, \Sigma_1, \) and \( V_1 \) Matrices
The final step is to construct the \( U_1, \Sigma_1, \) and \( V_1 \) matrices corresponding to the singular value decomposition of the interval-valued input matrix, \( M_t \), according to the decomposition target described in Section 3.4: this leads to three schemes, ISVD1-a, with interval-valued \( U_1, \Sigma_1, \) and \( V_1 \) matrices, ISVD1-b, with scalar-valued \( U \) and \( V \) and interval-valued \( \Sigma \), and ISVD1-c, with scalar-valued \( U, \Sigma \), and \( V \).

4.3 ISVD2: Decompose, Solve, Align
The ISVD1 algorithm described in the previous section first splits the interval-valued input matrix, \( M_t \), into its minimum and maximum valued components, \( M_s \) and \( M_e \), decomposes them independently, and then aligns the resulting latent semantics before recomposing the resulting \( U_1, \Sigma_1, \) and \( V_1 \) matrices. A potential problem with this strategy is that \( M_s \) and \( M_e \) are decomposed independently and this may introduce errors in the overall decomposition. An alternative strategy would be to first finding left and/or right singular vectors of the interval-valued matrix \( M_t \) and using these to seek for the final \( U_1, \Sigma_1, \) and \( V_1 \) matrices.

Let us remember from Section 2.2.1 that when \( M \) is a scalar-valued matrix and \( M = USV^T \) is its singular value decomposition, then we have the following: (a) the columns of \( U \), also called the left singular vectors of matrix \( M \), are the eigenvectors of the \( m \times m \) matrix, \( MM^T \); (b) the columns of \( V \), or the right singular vectors of \( M \), are the eigenvectors of the \( n \times n \) matrix, \( M^TM \); and (c) the diagonal entries of \( \Sigma \), also known as the singular values of \( M \), are the square
roots of the eigenvalues of $MM^T$ and $M^TM$. Therefore, a more principled way to seek the SVD of an interval-valued matrix would be to first compute an $n \times n$ square matrix, $A = MM^T$, through interval-valued matrix multiplication (discussed in Section 2.1) and then, seek the eigenvectors and eigenvalues of this matrix.

4.3.1 Eigen-Decomposition of a Square Matrix

[33], [34] define the eigen-decomposition of an interval valued matrix $A_1$, as a solution to the $A_1\vec{v}_1 = \lambda_1\vec{v}_1$ problem, where $\vec{v}_1$ is an interval-valued eigen-vector and $\lambda$ is a scalar.

[33], [35] provide bounds and linear-programming based solutions to solve this problem. In contrast, [36] reformulates the problem as seeking interval eigen-values of scalar-valued eigen-vectors. In this paper, we consider a more general solution, where we seek interval-valued eigen-values, corresponding to interval eigen-vectors: Let $A_1 = M_1^TM_1$ be an interval-valued square matrix. For simplicity, let us assume that $A_1$ is a $3 \times 3$ interval matrix:

$$A_1 = \begin{bmatrix} a_x & a_y & a_z \\ d_x & d_y & d_z \\ g_x & g_y & g_z \end{bmatrix} = \begin{bmatrix} [a_x, a_y, a_z] \\ [d_x, d_y, d_z] \\ [g_x, g_y, g_z] \end{bmatrix}$$

Let $\lambda_1 = [\lambda_x, \lambda_y, \lambda_z]$ be an interval-valued eigenvalue of $A_1$ and $\vec{v}_1 = [x, y, z]^T$ is the corresponding interval-valued eigenvector. Since (by definition) $A_1\vec{v}_1 = \lambda_1\vec{v}_1$, we have

$$\begin{bmatrix} a_x & a_y & a_z \\ d_x & d_y & d_z \\ g_x & g_y & g_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \lambda_x \cdot x \\ \lambda_y \cdot y \\ \lambda_z \cdot z \end{bmatrix}$$

By applying interval-valued algebra and simple matrix algebra, we can rewrite the above equality as

$$\begin{bmatrix} a_x & a_y & a_z \\ d_x & d_y & d_z \\ g_x & g_y & g_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_x \cdot x + a_y \cdot y + a_z \cdot z \\ d_x \cdot x + d_y \cdot y + d_z \cdot z \\ g_x \cdot x + g_y \cdot y + g_z \cdot z \end{bmatrix} = [\lambda_x, \lambda_y, \lambda_z] \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

It is easy to see that the above equality can be split into a minimum and maximum component:

$$(a_x \cdot x + a_y \cdot y + a_z \cdot z, d_x \cdot x + d_y \cdot y + d_z \cdot z, g_x \cdot x + g_y \cdot y + g_z \cdot z) = [\lambda_x, \lambda_y, \lambda_z] \cdot (x, y, z)$$

In other words, to obtain the eigen-decomposition of the original interval-valued matrix, $A_1$, we can instead seek the eigen-decompositions of the matrices $A_1$, $A_2$, where

$$A_2 = \begin{bmatrix} a_x & b_y & c_z \\ d_x & e_y & f_z \\ g_x & h_y & i_z \end{bmatrix}$$

$$A^* = \begin{bmatrix} a^+_x & b^+_y & c^+_z \\ d^+_x & e^+_y & f^+_z \\ g^+_x & h^+_y & i^+_z \end{bmatrix}$$

4.3.2 Recovery of the $U_*$ and $U^*$ Matrices

Given the eigen-decomposition of $A_1 = M_1^TM_1$, which provides us $V_1$ and $V^*$ and the $\Lambda_1$ and $\Lambda^*$, the next step of the process involves computation of the $U_*$ and $U^*$ matrices using $V_1$, $V^*$, $\Lambda_1 = (\Sigma_1^2)$, and $\Lambda^* = (\Sigma^2)$ matrices. This can be achieved by directly relying on the definition of SVD: $U_* = M_1((V_1)^T)-1(\Sigma_1)^{-1}$ and $U^* = M^*(V^*)^T)-1(\Sigma^*)^{-1}$.

4.3.3 Latent Semantic Alignment and Construction of $U_1$, $\Sigma_1$, and $V_1$ Matrices

Once the $U_*$, $U^*$, $V_*$ and $V^*$ are obtained, we can construct the $U_1$, $\Sigma_1$, and $V_1$ matrices by following the same
steps in Section 4.2.1. These steps are visualized under the column labeled ISVD₃ in Figure 4. Note that, as also discussed in Section 4.2.1, the decompositions of $A_*$ and $A^*$ can lead to situations where $V_1[i,j] > V^*[i,j]$ and/or $\Sigma_1[i,j] > \Sigma^*[i,j]$. Similarly, $U_*$ and $U^*$ computed using these matrices can contain cases where $U_1[i,j] > U^*[i,j]$. As before, these entries do not negatively impact the latent semantic alignment process and, thus, are corrected as part of the final step of the decomposition process (Section 4.3) which returns (ISVD₃-a) interval-valued $U_1$, $\Sigma_1$, and $V_1$ matrices, (ISVD₃-b) scalar-valued $U$ and $V$ and interval-valued $\Sigma_1$, or (ISVD₃-c) scalar-valued $U, \Sigma$, and $V$.

### 4.4 ISVD₄: Decompose, Align, Solve

In this section, we further consider an equivalent approach where, instead of obtaining $U_*$ and $U^*$ matrices independently, we (a) first solve for the $V_1$ and $\Sigma_1$ matrices, (b) align the $V_1$ and $V^*$ matrices, and then (c) use these to compute the $U_1$ matrix through interval-valued matrix inversion (Figure 4). Let, as before, $A_1$ be an $n \times n$ square matrix, $A_1 = M_1M_1^T$, obtained through interval-valued matrix multiplication (Section 2.1). Let matrices $V_1$ and $V^*$ capture the eigenvectors and $\Sigma_1$ and $\Sigma^*$ be diagonal matrices encoding the square roots of the eigenvalues of matrices $A_*$ and $A^*$, respectively, as described in Section 4.3.1.

#### 4.4.1 Latent Semantic Alignment

Given the $V_1$ and $V^*$ matrices along with the corresponding $\Sigma_1$ and $\Sigma^*$, we first seek the latent semantic alignment through ILSA, as described in Section 3.3. Given the latent alignment, we can next compute the interval-valued $V_1$ matrix as in Section 3.4.1. The interval-valued core matrix $\Sigma_1$ is also computed similarly, using the $A_*$ and $A^*$ matrices.

#### 4.4.2 Recovery of the matrix $U_1$

Given the interval valued matrices $V_1$ and $\Sigma_1$, we can then compute the interval-valued $U_1$ matrix relying on the definition of SVD: $U_1 = M_1((V_1)^T)^{-1}(\Sigma_1)^{-1}$. This, however, requires us to invert interval-valued matrices. We discuss how to achieve this next.

#### 4.4.2.1 Inverse of a Non-Negative Interval-Valued, $\Sigma_1$

Let $S = \Sigma_1$ be a $r \times r$ diagonal matrix, where the entries in the diagonal may be interval values and the rest of the entries are 0. The entries in the diagonal are non-negative since they are obtained through square-roots of eigenvalues. We seek an $r \times r$ diagonal matrix, $S^{-1}$, such that $S^{-1}S = I$, where $I$ is a $k \times k$ interval-valued diagonal matrix, approximately equal to the identity matrix. More specifically, for all $1 \leq i \leq r$, we have $I(i,i) = [1 - \epsilon_i, 1 + \epsilon_i]$, where $0 \leq \epsilon_i \leq 1$.

Given the above, let $S(i,i) = [s_{i,i}, s_{i,i}^+]$ and $S^{-1}(i,i) = [\sigma_{i,i}, \sigma_{i,i}^+]$. We seek $\sigma_{i,i}$ and $\sigma_{i,i}^+$ values that minimize the value of $\epsilon_i$ subject to the constraints (1) $s_{i,i} \times \sigma_{i,i} = 1 - \epsilon_i$, (2) $s_{i,i}^+ \times \sigma_{i,i}^+ = 1 + \epsilon_i$, (3) $\sigma_{i,i} \leq \sigma_{i,i}^+$, and (4) $0 \leq \epsilon_i \leq 1$. Note that the first two constraints can be combined to obtain $\sigma_{i,i} = \frac{1}{s_{i,i} - \epsilon_i}$ and $\sigma_{i,i}^+ = \frac{1 + \epsilon_i}{s_{i,i}^+}$, which, together with the third constraint, lead to the inequality $0 \leq \frac{s_{i,i}^+ - s_{i,i}}{s_{i,i}^+ + s_{i,i}} \leq \epsilon_i \leq 1$.

This constraint shows that $\epsilon_i$ is minimum when it is equal to $\frac{s_{i,i}^+ - s_{i,i}}{s_{i,i}^+ + s_{i,i}}$, and this case works when $\sigma_{i,i} = \sigma_{i,i}^+$. Therefore, elements in interval inverse matrix $S^{-1}$ have only scalar values, where $\sigma_{i,i} = \sigma_{i,i}^+$. Additionally, when $\sigma_{i,i} = \sigma_{i,i}^+ = \sigma_i$, the equations, $s_{i,i} \times \sigma_i = 1 - \epsilon_i$, and $s_{i,i}^+ \times \sigma_i = 1 + \epsilon_i$, together can be used to get $\sigma_i = \frac{1}{s_{i,i} + s_{i,i}^+}$.

#### 4.4.2.2 Inverse of an Interval-Valued Matrix, $V_1$

Note that the above inversion process is applicable to only square, diagonal interval-valued matrices. Unfortunately, the factor matrix $V_1$, is not a diagonal matrix. Therefore, its inverse cannot be obtained using this method. We therefore, need to approximate $V_1^{-1}$, by (a) first computing $V_{avg}$ by replacing each interval in $V_1$ with its mean, (b) inverting $V_{avg}$, and (c) returning $V_{avg}^{-1}$ as $V_1^{-1}$. Note that, one difficulty with this approach is that $V_{avg}$ may not always be invertible. We, therefore, first check whether $V_{avg}$ is well-conditioned and, if that is not the case, use Moore-Penrose pseudo-inverse for $V_{avg}^{-1}$ (or alternatively, revert back to ISVD₃). This can be considered as a generalization of the inverse matrix – we adapt SVD to compute pseudo-inverse and replace singular values smaller than 0.1 with zero.

#### 4.4.3 Construction of $U_1, \Sigma_1$, and $V_1$ Matrices

Given $M_1$ and the inverses of the factor matrix $V_1$ and the core matrix $\Sigma_1$, we compute the factor matrix $U_1$ using the equation $U_1 = M_1((V_1)^T)^{-1}(\Sigma_1)^{-1}$. This is visualized under the column labeled ISVD₄ in Figure 4.

As discussed in Sections 4.2.1 and 4.3.3, the decompositions of $A_*$ and $A^*$ can lead to situations where $V_1[i,j] > V^*[i,j]$ and/or $\Sigma_1[i,j] > \Sigma^*[i,j]$. These entries do not negatively impact the latent semantic alignment process. They, moreover, also do not impact the computation of $U_1 = M_1((V_1)^T)^{-1}(\Sigma_1)^{-1}$, since (as described above) both $\Sigma_1^{-1}$ and $((V_1)^T)^{-1}$ involve averaging of the interval-valued entries. Therefore, these misordered elements are corrected as part of the final step of the decomposition process described in Section 3.4.3. Intuitively, this recomputation step identifies minimum and maximum factor matrices, which themselves may not always be invertible. We, therefore, first check whether $V_{avg}$ is well-conditioned and, if that is not the case, use Moore-Penrose pseudo-inverse for $V_{avg}^{-1}$ (or alternatively, revert back to ISVD₃). This can be considered as a generalization of the inverse matrix – we adapt SVD to compute pseudo-inverse and replace singular values smaller than 0.1 with zero.

### 4.5 ISVD₄: Decompose, Align, Solve, Recompute

In this section, we consider a variant of ISVD₃, which inherits its benefits, but further reduces the degree of imprecision in the resulting interval-valued factor matrices.

#### 4.5.1 Recomputation of the Left Singular Vectors

As we discussed in the previous sections, the decomposition of interval-valued matrices potentially results in interval-valued factor matrices – as discussed in Section 3.3, the smaller are the sizes of the intervals in these factor matrices, the more precise are the resulting factors. Therefore, as discussed before, it is important to keep these intervals smaller (or equivalently, keep the minimum and maximum versions of the eigenvectors similar) as long as this does not introduce inaccuracies in the overall decomposition.

The ISVD₄ approach follows the same steps as ISVD₃, up to the computation of the left singular vectors matrix, $U_1$, using the (aligned) right singular-values, $V_1$, obtained through eigen-decomposition of minimum and maximum matrices. At that point, instead of using the original $V_1$ factor matrix, ISVD₄ introduces an extra step to recompute the right singular vector matrix relying on the definition of SVD: $V_1 = M_1((V_1)^T)^{-1}(\Sigma_1)^{-1}$. Intuitively, this recomputation step identifies minimum and maximum factor matrices, $V_1$ and $V^*$, that are compatible with the $U_1$ factor, which itself benefits from the alignment step for the left-singular vectors.

3. This also applies to ISVD₄ described next.
(a) \(\cos(V_s[i], V^*[i])\) and \(\cos(U_s[i], U^*[i])\) before the recomputation of \(V_s\) (the higher the better)

(b) \(\cos(V_s[i], V^*[i])\) and \(\cos(U_s[i], U^*[i])\) after the recomputation of \(V_s\) (the higher the better)

Fig. 5: Cosine similarities among min and max vectors: the higher the cosine similarity, the more similar the minimum and maximum factor matrices are, indicating higher precision in the decomposition (averaged for 100 matrices as described in Table 1, Section 6, default configuration)

Figure 5 visualizes how matrix re-computation affects eigenvectors’ alignment. In the figure, we plot cosine similarities between the corresponding eigenvectors in \(V_s\) and \(V^*\), considered in increasing order of singular values – note that, the more similar the minimum and maximum versions of the eigenvectors is (i.e., the more precise the factors are), the higher is the cosine similarity. The figure reports the cosine similarity values before recomputation and after recomputation. As we see in Figure 5, for all phases, the first few eigenvectors (with the largest singular values) have the highest cosine similarities (indicating that they are more precise). On the other hand, Figure 5(a) shows that the alignment between the corresponding vectors in \(U_s\) and \(U^*\) (obtained in ISVD, through the equation \(U_t = M_t((V_t)^T)^{-1}(\Sigma_v)^{-1}\)) is very high, despite using less precise \(V_t\) vectors in the construction: intuitively, the equation used for the \(U_s\) and \(U^*\) matrices weight the contribution of the more precise entries in the \(V_s\) and \(V^*\) matrices through multiplication with the inverses of the singular values.

ISVD leverages this “corrective” behavior to re-obtain the \(V_s\) and \(V^*\) matrices using an extra step \(V_t = M(V_t)^T)^{-1}\Sigma_v^{-1}\). The result is visualized in Figure 5(b): as we see here, after the recomputation step, the \(V_s\) and \(V^*\) matrices become much more similar, indicating more precise factor matrices, an improvement which (as we detail in Section 6) contributes to more accurate decompositions.

4.5.2 Construction of \(U_t\), \(\Sigma_t\), and \(V_t\) Matrices

After re-computing interval-valued factor matrices \(V_s\) and \(V^*\), we can follow the same process described in the previous sections to reconstruct \(U_t\), \(\Sigma_t\), and \(V_t\). The steps are visualized in the column labeled ISVD in Figure 4. As before, depending on the decomposition target, this process can lead to interval-valued matrices in \(ISVD_t\)-a, scalar-valued \(U\), \(V\) and interval-valued \(\Sigma_t\) in \(ISVD_t\)-b, or scalar-valued \(U\), \(\Sigma\) and \(V\) in \(ISVD_t\)-c.

5 INTERVAL-VALUED PMF (IPMF)

In the previous section, we discussed several techniques to leverage latent semantic alignment to implement SVD on interval-valued data. We now show that latent alignment of minimum and maximum basis vectors can also be effective in alternative factorization schemes, such as probabilistic matrix factorization (PMF) introduced in Section 2.2.3.

We first consider interval-valued PMF (I-PMF) formulation presented in [9], which takes as input an interval-valued matrix \(M_t \in \mathbb{R}^{n \times m}\) (of probabilities) and introduces extra constraints, representing the minimum and maximum terms, in the loss function presented in Section 2.2.3:

\[
\mathcal{L}_{I-PMF} = \sum_{i,j} (V_s[i,j] - M_t[i,j])^2 + \lambda_U U_s[i,j]^2 + \lambda_V V_s[i,j]^2,
\]

where \(V_s\) and \(V^*\) represent the minimum and maximum in interval matrix \(M_t\). Given this loss formulation, a local minimum can be sought by applying a gradient descent in \(U_s[i,j]\), \(V_s[i,j]\) and \(V^*[i,j]\) using the following partial derivatives:

\[
\frac{\partial \mathcal{L}_{I-PMF}}{\partial U_s[i,j]} = \sum_{j=1}^n (V_s[i,j] - M_t[i,j]) V_s[i,j] + \lambda_U U_s[i,j] - \lambda_U U_t[i,j],
\]

\[
\frac{\partial \mathcal{L}_{I-PMF}}{\partial V_s[i,j]} = \sum_{i=1}^n (V_s[i,j] - M_t[i,j]) U_s[i,j] + \lambda_V V_s[i,j] T + \lambda_V V_t[i,j] T,
\]

\[
\frac{\partial \mathcal{L}_{I-PMF}}{\partial V^*[i,j]} = \sum_{i=1}^n (V_s[i,j] - M_t[i,j]) U_s[i,j] + \lambda_V V^*[i,j] T.
\]

In this paper, we note that, while the above formulation adopts extra constraints in loss function to capture interval property of the data, it fails to consider the need for alignment in interval-valued latent semantic spaces, as discussed in Section 3.3. Hence, we propose semantically aligned interval-valued PMF formulation (AI-PMF), which adjusts the rank-order of eigen-vectors in \(V_s\) and \(V^*\) in each gradient descent iteration, after obtaining \(U_s\), \(V_s\) and \(V^*\) from the above equations:

\[
V_{aligned} = \{V_{aligned}, V^*_{aligned}\} = ILSA(V_s, V^*).
\]

As we see in the next section, this latent semantic alignment provides significant improvements in decomposition accuracy of interval-valued (probabilistic) data.

6 EXPERIMENTS

In this section, we evaluate the proposed techniques in the presence of different distributions of interval-valued data. We also demonstrate that the proposed approaches can be applied to various real-world applications, including face image analysis and collaborative filtering.

6.1 Datasets

We use both synthetic and real datasets to evaluate the proposed matrix decomposition techniques in the context of several data reconstruction and classification scenarios.
TABLE 1: Experiment parameters for synthetic data (bold values are defaults)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Matrix dimension</td>
<td>40 × 250, 250 × 40, 25 × 400, 400 × 250, 250 × 400</td>
</tr>
<tr>
<td>Matrix density</td>
<td>percentage of 0-values: 0%, 50%, 90%</td>
</tr>
<tr>
<td>Interval density</td>
<td>5%, 25%, 50%, 75%, 100%</td>
</tr>
<tr>
<td>Interval intensity</td>
<td>5%, 25%, 50%, 75%, 100%</td>
</tr>
<tr>
<td>Target rank</td>
<td>uniform data: 5; 10, 20, 40, 100, full anonymized and social media data: 5%; 50%; 100%</td>
</tr>
</tbody>
</table>

6.1.1 Synthetic Datasets

Table 1 provides an overview of the synthetic data scenarios we have considered in this paper. As the table shows, we have considered matrices of different sizes, dimensionalities, densities, and intensities: interval-values are generated by picking a cell in the matrix (based on the value of the interval density parameter) and replacing the corresponding scalar value by an interval whose size is determined by an interval intensity parameter—given the intensity value $X$, the scope of the interval is uniformly selected between $0\%$ and $X\%$ of the minimum value of the cell.

We considered two types of interval-valued data:

- **uniform** matrices do not have inherent structures—the cells and interval-values are selected uniformly randomly subject to constraints described in Table 1;
- **anonymized** matrices are obtained through generalization of random matrices; therefore, the resulting interval-valued matrices represent the required degree of anonymization. In particular, we consider 4 generalization levels: (a) $L1$ divides the domain into 100, (b) $L2$ into 50, (c) $L3$ into 20, and (d) $L4$ into 5 generalization intervals—note that the higher the generalization level, the larger the corresponding intervals and the more anonymized the data. Given these, we generalize the input data to generate three matrices with different mixtures of anonymization levels: high anon. (L1:10%, L2:20%, L3:30%, L4:40%), medium anon. (L1:25%, L2:25%, L3:25%, L4:25%), and low anon. (L1:40%, L2:30%, L3:20%, L4:10%).

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

6.1.2 ORL Face Dataset

ORL face dataset [9] contains 32x32 pixels facial images of the 40 individuals. The data set contains 10 images for each individual. These 400 images are organized in the form of a matrix $M \in \mathbb{R}^{n \times m}$, where $n = 400$, $m = 1024$; i.e., each row represents a person’s face and each column represents the pixel value at specific coordinates. We then follow the strategy proposed in [9] to generate an interval-valued matrix for the same data set and use these matrices for the following applications:

- **Reconstruction**: In this task, we attempt to reconstruct the original data matrix from the low-rank approximations and use root-mean square error to assess the reconstruction accuracy.
- **Clustering-based Classification**: We evaluate accuracy using F1-score.
- **NN-Classification**: In this task, the goal is to use the (scalar-valued) $U$ factor for NMF based schemes and $U \times S$ (more specifically $[U_s \times U_s, U_s \times S^*]$), since we have interval valued matrices factor for SVD based schemes to identify an individual among the 40 that are in the data set. For this purpose, we randomly select 50% rows per individual as training data and remaining rows are used as test data. The classification is done using 1-NN with Euclidean distance. When necessary, for interval-valued data, we use the following interval-valued Euclidean distance function:

$$\text{dist}(a_{int}, b_{int}) = \sqrt{(a_s - b_s)^2 + (a^* - b^*)^2}.$$  

We will evaluate accuracy using F1-score.

- **Clustering-based Classification**: With the ORL face data set, we also consider classification through K-means (where $K = 40$) clustering, using Euclidean distances as described above. Normalized mutual information (NMI) [37] is used to assess cluster quality relative to given class labels.

We repeated each task 100 times and are reporting averages.

6.1.3 Social Media Data Sets

6.1.3.1 Ciao and Epinions Datasets: We have performed experiments with two ratings data sets: Epinions, Ciao [11]. For both Epinions (22K users, 27 categories, matrix density=0.26, interval density= 0.49, interval intensity=2.44/4, full rank=27) and Ciao (7K users, 28 categories, matrix density=0.28, interval density= 0.44, interval intensity=2.20/4, full rank=28) data sets, we have considered...
TABLE 2: Comparison of the alternative approaches (for Option-b) with varying parameters
(the higher the Harmonic mean, the better the result)
lower reconstruction accuracies. We note that, while the relatively simpler ISVD₀ appears to be competitive against ISVD₂ in the reconstruction task, in the classification tasks discussed next, performs less effectively.

### 6.4.2 NN-based Classification

Figure 8(b) shows classification performance for various algorithms. As we see here, the best classification results are obtained using ISVD₁ and ISVD₂ techniques, both of which leverage latent semantic alignment, under low (20-30) rank decomposition. Since classification task only relies on $U$ factors and the core matrix, the ISVD₃ and ISVD₄ techniques (which attempt to further improve the latent space defined by the $V$ factor) do not provide additional benefits (and in fact hurt the classification accuracies). It is important to note that NMF and I-NMF competitors, both, provide significantly lower accuracies than ISVD-based approaches.

### 6.4.3 Clustering-based Classification

The clustering results, shown in Figure 8(c), re-confirm the NN-based classification results discussed above. Once again, the best results are obtained using ISVD₁ and ISVD₂ techniques, both of which leverage latent semantic alignment, under low (≤10) rank decomposition.

Finally, in Table 3 we consider the original face data (scalar vectors) and processed interval-valued vectors and compare them to ISVD₂ based clustering for both accuracy and execution time. As we see in this table, adding interval information (scalar vectors vs. interval vectors) significantly improves accuracy, but at a significant execution time cost. The proposed interval-valued decomposition strategy cuts down the clustering time significantly, while matching the accuracy (at rank 20) provided by the interval vectors.
TABLE 3: Accuracy and execution time for clustering-based classification using original data vector, interval-valued data vectors, and low-rank interval-value based data

<table>
<thead>
<tr>
<th>Accuracy in terms of NMI; the higher, the better</th>
<th>Exec. Time (decomp.+k-means) in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>res.</td>
<td>scalar vectors</td>
</tr>
<tr>
<td>32 × 32</td>
<td>0.69</td>
</tr>
<tr>
<td>64 × 64</td>
<td>0.68</td>
</tr>
</tbody>
</table>

6.5 Results based on Social Media Data

6.5.1 Reconstruction Results

Figure 9 presents reconstruction accuracies for different target ranks and decomposition semantics on three interval-valued data sets, Ciao, Epinions, and MovieLens.

These results re-confirm that, in general, decomposition target option-b (with interval-valued core and scalar factors) leads to best results, except for very low-rank decompositions, where option-a (with interval-valued core and factors) becomes a better option.

For Ciao and Epinion data sets, we see that, as in the data sets considered earlier, ISVD_3 and ISVD_4, which perform latent alignment early, lead to better results. ISVD_1 and ISVD_2, which perform late-alignment of the factors, are more effective than ISVD_3 and ISVD_4 only for very low rank decompositions. The MovieLens data set also has a similar behavior: for relatively large decomposition ranks, ISVD_3 and ISVD_4, which perform latent alignment early, lead to better results. For low rank decompositions (r ≤ 10) the benefit of early alignment is lost and ISVD_1 and ISVD_2 are (slightly) more effective.

6.5.2 Collaborative Filtering/Prediction Results

Figure 10 shows collaborative filtering performance in RMSE with MovieLens-100K dataset for PMF, I-PMF and AI-PMF. 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 > 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 considered in Section 3.3.

7 Conclusions

In this paper, we noted that many applications generate interval-valued data, yet, existing data analysis tools (such as matrix-decomposition operations) assume that all observations are scalar-valued. Building on this observation, we presented eigen-vector based and probabilistic matrix-decomposition techniques (interval singular value decomposition (ISVD) and aligned interval probability matrix factorization (AI-PMF), respectively) that are designed for interval-valued data sets. Experiments, under diverse data scenarios and application semantics, showed that state-of-the-art linear-programming based eigen-decomposition techniques are ineffective for interval-valued SVD. Results also showed that by carefully combining interval- and matrix-algebra operations, we can improve matrix decomposition accuracies and further improve the performance in image classification, images clustering and collaborative filtering in the presence of interval-valued data.

Acknowledgments

Supported by NSF#163381, NSF#1610282, and EU-H2020 grant (“FourCmodeling”) No 690817.

References

Fig. 10: Collaborative filtering (RMSE, the lower, the better).


Mao-Lin Li is a computer science PhD student at the Arizona State University. His research interests include management and analysis of imprecise, graph structured data, with applications including collaborative filtering and predictive analysis.

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

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

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

In this document, we provide supporting materials, including pseudo-codes of the algorithms described in the paper, along with the detailed description of the data generation processes for the ORL faces and MovieLens datasets:

## CONTENTS

A Pseudocodes for Major Supporting Functions

A.1 Interval-valued Matrix Multiplication 1
A.2 Vector and Matrix Average Replacement 2
A.3 Inverse of a Non-Negative Interval-Valued Core Matrix 2
A.4 L2-Norm based Matrix Normalization 2

B Pseudocodes for Interval-Valued Latent Semantic Alignment (ILSA) 3

C Pseudocodes for ISVD Algorithms

C.1 ISVD$^0$ (Naive Approach): Average and Decompose 3
C.2 ISVD$^1$: Decompose and Align 3
C.3 ISVD$^2$: Decompose, Solve, Align 4
C.4 ISVD$^3$: Decompose, Align, Solve 5
C.5 ISVD$^4$: Decompose, Align, Solve, Re-compute 5

D Pseudocodes for Reconstruction Algorithms

D.1 Reconstruction Target (a): Interval-valued $U_\Sigma, \Sigma V_i$ 6
D.2 Reconstruction Target (b): Scalar $U, V_i$ 7
D.3 Reconstruction Target (c): Scalar $U, \Sigma, V$ 7

E Pseudocodes For Aligned Interval Probabilistic Matrix Factorization (AI-PMF) 7

F ORL Face and MovieLens Datasets

F.1 Intervals in the ORL Face Dataset 7
F.2 Intervals in the MovieLens Dataset 8

G Link for Datasets and Source Codes 8

## A PSEUDOCODES FOR MAJOR SUPPORTING FUNCTIONS

### A.1 Interval-valued Matrix Multiplication

Interval-valued matrix multiplication is discussed in Section 2.1 of the paper and used throughout the paper as a core matrix algebraic operation.

#### Algorithm 1 Interval-valued Matrix Multiplication

Input:

Interval-valued matrices $A_\Sigma = [A_\Sigma, A^*] \in \mathbb{R}^{n_A \times m_A}$, $B_\Sigma = [B_\Sigma, B^*] \in \mathbb{R}^{m_B \times m_B}$, with $m_A = n_B$.

Output:

Interval-valued matrix $Res_\Sigma = [Res_\Sigma, Res^*] \in \mathbb{R}^{m_B \times n_A}$.

1: procedure INTERVAL-MAT-MULT($Res_\Sigma$)
2:   Initialize $Res_\Sigma, T_1, T_2, T_3, T_4 \in \mathbb{R}^{m_B \times n_A}$
3:   $T_1 \leftarrow A_\Sigma \cdot B_\Sigma$
4:   $T_2 \leftarrow A_\Sigma \cdot B^*$
5:   $T_3 \leftarrow A^* \cdot B_\Sigma$
6:   $T_4 \leftarrow A^* \cdot B^*$
7:   for $i \leftarrow 1$ to $m_B$
8:     for $j \leftarrow 1$ to $n_A$
9:       $Res_\Sigma[i, j] \leftarrow \min\{T_1[i, j], T_2[i, j], T_3[i, j], T_4[i, j]\}$
10:      $Res^*[i, j] \leftarrow \max\{T_1[i, j], T_2[i, j], T_3[i, j], T_4[i, j]\}$
12:   end for
13: end for
14: return $Res_\Sigma$
15: end procedure
A.2 Vector and Matrix Average Replacement

Average replacement is a mechanism to correct vectors and matrices that include entries that are not proper intervals. Several algorithms potentially introduce such incorrectly formed intervals (where the minimum values are larger than maximum values) in their intermediary steps. As discussed in Sections 4.2.1, 4.3.3, and 4.4.3 of the revised manuscript, this does not lead to mathematical problems as the following intermediary operations do not inherently require that the input matrices are min-max ordered. However, these vectors and matrices need to be corrected before they are returned to the user as the output of the decomposition process.

Algorithm 2 Vector Average Replacement

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

1: procedure AVERAGE-REPLACEMENT-VEC($v_\ell$)
2: Initialize $\hat{v}_\ell \in \mathbb{R}^{1 \times n}$
3: $\hat{v}_\ell \leftarrow v_\ell$
4: 
5: for $i \leftarrow 1$ to $n$
6: if $v_\ell[i] > v^*[i]$ then
7: $\hat{v}_\ell[i] \leftarrow v_\ell[i] + v^*[i]$
8: $\hat{v}^*[i] \leftarrow \hat{v}_\ell[i]$
9: end if
10: end for
11: return $\hat{v}_\ell$
12: end procedure

Algorithm 3 Matrix Average Replacement

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

1: procedure AVERAGE-REPLACEMENT-MAT($M_\ell$)
2: Initialize $\hat{M}_\ell \in \mathbb{R}^{n \times m}$
3: $\hat{M}_\ell \leftarrow M_\ell$
4: 
5: for $i \leftarrow 1$ to $n$
6: for $j \leftarrow 1$ to $m$
7: if $M_\ell[i, j] > M^*[i, j]$ then
8: $M_\ell[i, j] \leftarrow \frac{M_\ell[i, j] + M^*[i, j]}{2}$
9: $M^*[i, j] \leftarrow \frac{M_\ell[i, j] + M^*[i, j]}{2}$
10: end if
11: end for
12: end for
13: return $\hat{M}_\ell$
14: end procedure

A.3 Inverse of a Non-Negative Interval-Valued Core Matrix

Several ISVD algorithms require the inverse of a non-negative interval-valued core matrix. This is discussed in Section 4.4.2.1 of the manuscript.

Algorithm 4 Inverse of a Non-Negative Interval-Valued $\Sigma_\ell$

Input: Interval-valued matrix $\Sigma_\ell = [\Sigma_\ell, \Sigma^*] \in \mathbb{R}^{n \times r}$, target rank $r$.
Output: Scalar matrix $\Sigma^{inv} \in \mathbb{R}^{n \times r}$, approximating the inverse of the input matrix $\Sigma_\ell$.

1: procedure INVERSE-$\Sigma(\Sigma_\ell, r)$
2: Initialize $\Sigma^{inv} \in \mathbb{R}^{n \times r}$: $\forall i, j \Sigma^{inv}[i, j] \leftarrow 0$
3: for $i \leftarrow 1$ to $r$
4: if $\Sigma_\ell[i, i] = 0 \land \Sigma^*[i, i] = 0$ then
5: $\Sigma^{inv}[i, i] \leftarrow 0$
6: else if $\Sigma_\ell[i, i] = 0 \land \Sigma^*[i, i] \neq 0$ then
7: $\Sigma^{inv}[i, i] \leftarrow 2 \frac{\Sigma^*[i, i]}{\Sigma^*[i, i] + \Sigma^*[i, i]}$
8: else if $\Sigma_\ell[i, i] \neq 0 \land \Sigma^*[i, i] = 0$ then
9: $\Sigma^{inv}[i, i] \leftarrow 2 \frac{\Sigma^*[i, i]}{\Sigma^*[i, i] + \Sigma^*[i, i]}$
10: else
11: $\Sigma^{inv}[i, i] \leftarrow 2 \frac{\Sigma^*[i, i] \Sigma^*[i, i]}{\Sigma^*[i, i] + \Sigma^*[i, i]}$
12: end if
13: end for
14: return $\Sigma^{inv}$
15: end procedure

A.4 L2-Norm based Matrix Normalization

As we see in Figure 4 and discuss in Section 3.4 of the manuscript, results may need to be L2-normalized before being returned to the end-user.

Algorithm 5 L2-Norm Matrix Normalization

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

1: procedure NORM-MAT($A$)
2: Initialize $\hat{A} \in \mathbb{R}^{n \times m}$
3: Initialize $colNorm \in \mathbb{R}^{1 \times m}$
4: 
5: for $j \leftarrow 1$ to $n$
6: $colNorm[j] \leftarrow ||A[:, j]||_2$
7: for $i \leftarrow 1$ to $m$
8: $A[i, j] \leftarrow \frac{A[i, j]}{colNorm[j]}$
9: end for
10: end for
11: return $\hat{A}, colNorm$
12: end procedure
B PSEUDOCODES FOR INTERVAL-VALUED LATENT SEMANTIC ALIGNMENT (ILSA)

Interval-valued latent semantic alignment, discussed in Section 3.3 of the manuscript, is a key process, helping combine separately obtained minimum and maximum basis vectors to form an interval-valued latent space. As discussed in the paper, the problem can be formulated as an instance of the linear assignment problem [32]. Below, we provide the pseudocode of the algorithm we use in our implementation for the latent semantic alignment process.

Algorithm 6 Interval-Valued Latent Semantic Alignment (ILSA)

Input:
Interval-valued matrix \( V_1 = [V_s, V^*] \in \mathbb{R}^{n \times r} \) with \( V_s = \{\vec{v}_1, \ldots, \vec{v}_s\} \), \( V^* = \{\vec{v}_1^*, \ldots, \vec{v}_r^*\} \), target rank \( r \).

Output:
Indices vector \( mappingIdx \in \mathbb{R}^{1 \times r} \) (for aligning the interval-valued components of \( V_1 \)), where \( mappingIdx[j] \) is the remapped index for \( \vec{v}_{s,j} \) and \( \vec{v}_{r,j} \), indicator vector \( dirFlag \in \mathbb{R}^{1 \times r} \) (to swap direction of misaligned \( V_1 \) components), where \( dirFlag[j] = 1 \) if \( \vec{v}_{s,j} \) needs to be swapped to better align with \( \vec{v}_{r,j}^* \), otherwise \( dirFlag[j] = 0 \).

1: # Main procedure
2: procedure ILSA\((V_1, r)\)
3: \( simMat, dirFlag \leftarrow \text{PAIRSIM}(V_1, r) \)
4: \( mappingIdx \leftarrow \text{MAPPING}(simMat, r) \)
5: return \( mappingIdx, dirFlag \)
6: end procedure

8: # Compute pairwise cosine similarity
9: procedure \( \text{PAIRSIM}(V_1, r) \)
10: Initialize \( simMat \in \mathbb{R}^{r \times r} \)
11: Initialize \( dirFlag \in \mathbb{R}^{1 \times r} \)
12: for \( i \leftarrow 1 \) to \( r \) do
13: \hspace{1em} for \( j \leftarrow 1 \) to \( r \) do
14: \hspace{2em} if \( \cos(\vec{v}_{s,i}, \vec{v}_{r,j}^*) < 0 \) then \( \triangleright \) cosine similarity
15: \hspace{2em} \( simMat[i, j] \leftarrow \cos(\vec{v}_{s,i}, -\vec{v}_{r,j}^*) \)
16: \hspace{2em} dirFlag[i] = 1
17: \hspace{2em} else
18: \hspace{2.5em} \( simMat[i, j] \leftarrow \cos(\vec{v}_{s,i}, \vec{v}_{r,j}^*) \)
19: \hspace{2em} dirFlag[i] = 0
20: end if
21: end for
22: end for
23: return \( simMat, dirFlag \)
24: end procedure

26: # Find the mapping maximizing the pairwise similarity between \( V_s \) and \( V^* \)
27: procedure \( \text{MAPPING}(simMat, r) \)
28: Initialize \( mappingIdx \in \mathbb{R}^{1 \times r} \)
29: for \( j \leftarrow 1 \) to \( r \) do
30: \hspace{1em} \( mappingIdx[j] \leftarrow i \mid \exists k : simMat[k, j] < simMat[i, j] \)
31: end for
32: \( spareIdx \leftarrow i \mid \exists j : mappingIdx[j] = i \) \hspace{1em} if \( spareIdx \neq \emptyset \) then
33: conflictIdx \(_i \leftarrow i \mid \exists j_1, j_2 : mappingIdx[j_1] = i \wedge mappingIdx[j_2] = i \wedge j_1 \neq j_2 \)
34: for \( i \in \text{conflictIdx} \)
35: \hspace{1em} \( conflictIdx^* \leftarrow j \mid mappingIdx[j] = i \)
36: \hspace{2em} sort(\( conflictIdx^* \), \( simMat[i, j] \), descending)
37: \hspace{2em} conflictIdx^* \leftarrow conflictIdx^* \setminus \{conflictIdx^*[1]\}
38: for \( j \in \text{conflictIdx}^* \)
39: \hspace{3em} \( mappingIdx[j] \leftarrow k \mid k \in spareIdx \wedge \exists h : simMat[h, j] < simMat[k, j] \wedge h \in spareIdx \)
40: \hspace{2em} spareIdx \leftarrow spareIdx \setminus \{k\}
41: end for
42: end for
43: end if
44: return \( mappingIdx \)
45: end procedure

C PSEUDOCODES FOR ISVD ALGORITHMS

As visualized in Figure 4 and discussed in Section 4 of the manuscript, in the paper, we propose four different approaches (five including a naive approach) to obtain interval-valued singular valued decompositions. These different techniques differ in how they obtain initial minimum and maximum decompositions, in which order they apply the latent semantic alignment, and how they revise the factor matrices. Below, we provide detailed pseudocodes of these algorithms.

C.1 ISVD\(_0\) (Naive Approach): Average and Decompose

The naive ISVD\(_0\) technique is discussed in Section 4.1 of the manuscript. See also Figure 4.

Algorithm 7 ISVD\(_0\)

Input:
Interval-valued matrix \( M_1 = [M_s, M^*] \in \mathbb{R}^{n \times m} \), target rank \( r \).

Output:
Scalar-valued factor matrices \( U_{avg} \in \mathbb{R}^{n \times r}, V_{avg} \in \mathbb{R}^{m \times r} \) and scalar-valued singular matrix \( \Sigma_{avg} \in \mathbb{R}^{r \times r} \).

1: procedure ISVD\(_0\)(\(M_1, r\))
2: \( M_{avg} \leftarrow \frac{M_s + M^*}{2} \)
3: \( U_{avg}, \Sigma_{avg}, V_{avg} \leftarrow \text{SVD}(M_{avg}, r) \) \( \triangleright \) Singular value decomposition
4: return \( U_{avg}, \Sigma_{avg}, V_{avg} \)
5: end procedure

C.2 ISVD\(_1\): Decompose and Align

The ISVD\(_1\) technique, which obtains minimum and maximum values of \( U, V, \) and \( \Sigma \) matrices independently before applying latent semantic alignment, is discussed in Section 4.2 of the manuscript. See also Figure 4.

Algorithm 8 ISVD\(_1\)
Input: Interval-valued matrix $M_t = [M_s, M^*] \in \mathbb{R}^{n \times m}$, target rank $r$, decomposition target option $optDT$.

Output: Decomposition target option (a):
Interval-valued factor matrices $U_t \in \mathbb{R}^{n \times r}, V_t \in \mathbb{R}^{m \times r}$ and interval-valued singular matrix $\Sigma_t \in \mathbb{R}^{r \times r}$.

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

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

1: \textbf{procedure} ISVD$_1$(M$_t$, r, optDT)
2: \hspace{1em} U$_s$, $\Sigma_s$, V$_s$ ← SVD(M$_s$, r) \hspace{1em} \triangleright \text{Singular Value Decomposition}
3: \hspace{1em} U$^*$, $\Sigma^*$, V$^*$ ← SVD(M$^*$, r)
4: \hspace{1em} mappingIdx, dirFlag ← ILSA(V$_s$, V$^*$) \hspace{1em} \triangleright \text{See Algorithm 6}
5: \hspace{1em} return
6: \textbf{end procedure}

Algorithm 6

1: \textbf{procedure} ISVD$_2$(M$_t$, r, optDT)
2: \hspace{1em} A$_t$ ← INTERVAL-MAT-MULT(M$_t$, M$_t$) \hspace{1em} \triangleright \text{See Algorithm 1}
3: \hspace{1em} V$_s$, $\Sigma^2_s$ ← Eig(A$_s$, r) \hspace{1em} \triangleright \text{Eigen Decomposition}
4: \hspace{1em} V$^*$, $\Sigma^2_s$ ← Eig(A$^*$, r)
5: \hspace{1em} U$_s$ ← M$_s$. (V$^T$)$^{-1}$. (\Sigma$^2_s$)$^{-1}$
6: \hspace{1em} U$^*$ ← M$^*$. (V$^T$)$^{-1}$. (\Sigma$^2_s$)$^{-1}$
7: \hspace{1em} mappingIdx, dirFlag ← ILSA(V$_s$, V$^*$) \hspace{1em} \triangleright \text{See Algorithm 6}
8: \hspace{1em} return
9: \textbf{end procedure}

C.3 ISVD$_2$: Decompose, Solve, Align
Unlike the ISVD$_1$ technique, which obtains minimum and maximum values of $U$, $V$, and $\Sigma$ matrices independently, the ISVD$_2$ technique first obtain the minimum and maximum values of $V$ and $\Sigma$ matrices and then uses these two solve for the minimum and maximum matrices for $U$. This is discussed in Section 4.3 of the manuscript. See Figure 4.

Algorithm 9 ISVD$_2$

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

Output: Decomposition target option (a):
Interval-valued factor matrices $U_t \in \mathbb{R}^{n \times r}, V_t \in \mathbb{R}^{m \times r}$ and interval-valued singular matrix $\Sigma_t \in \mathbb{R}^{r \times r}$.

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

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

1: \textbf{procedure} ISVD$_2$(M$_t$, r, optDT)
2: \hspace{1em} A$_t$ ← INTERVAL-MAT-MULT(M$_t$, M$_t$) \hspace{1em} \triangleright \text{See Algorithm 1}
3: \hspace{1em} V$_s$, $\Sigma^2_s$ ← Eig(A$_s$, r) \hspace{1em} \triangleright \text{Eigen Decomposition}
4: \hspace{1em} V$^*$, $\Sigma^2_s$ ← Eig(A$^*$, r)
5: \hspace{1em} U$_s$ ← M$_s$. (V$^T$)$^{-1}$. (\Sigma$^2_s$)$^{-1}$
6: \hspace{1em} U$^*$ ← M$^*$. (V$^T$)$^{-1}$. (\Sigma$^2_s$)$^{-1}$
7: \hspace{1em} mappingIdx, dirFlag ← ILSA(V$_s$, V$^*$) \hspace{1em} \triangleright \text{See Algorithm 6}
8: \hspace{1em} return
9: \textbf{end procedure}

\textbf{end if}

39: \textbf{end procedure}
The ISVD₃ technique applies the latent alignment process before solving for the U matrix. In addition, unlike ISVD₂, it uses interval-valued matrix algebra to solve for the interval-valued U matrix directly, instead of solving for the minimum and maximum matrices of U separately. This is discussed in Section 4.4 of the manuscript and visualized in Figure 4.

Algorithm 10 ISVD₃

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

Output: Decomposition target option (a): Interval-valued factor matrices \( U_1 \in \mathbb{R}^{n \times r}, V_1 \in \mathbb{R}^{m \times r} \) and interval-valued singular matrix \( \Sigma_1 \in \mathbb{R}^{r \times r} \).

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

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

1: procedure ISVD₃ \((M_I, r, \text{optDT}, \text{condTh})\)
2: \( \bar{A}_1 \leftarrow \text{INTERVAL-MAT-MULT}(M_I^T, M_I) \)  
3: \( \Sigma_1 \leftarrow \text{Eig}(\bar{A}_1, r) \)  
4: \( \bar{V}^* \leftarrow \text{Eig}(\bar{A}_1^*, r) \)  
5: \( \text{mappingIdx}, \text{dirFlag} \leftarrow \text{ILSA}(\bar{V}^*, \bar{V}^*) \)  
6: for \( j \leftarrow 1 \) to \( r \) do  
7: \( \bar{V}[:, j] \leftarrow \bar{V}[:, \text{mappingIdx}[j]] \)  
8: \( \Sigma_1[:, j] \leftarrow \text{Eig}(A_{1, \text{dirFlag}}) \)  
9: if \( \text{dirFlag}[j] = 1 \) then \( \text{Swap misaligned vectors} \)  
10: end if
11: end for
12: end procedure

C.4 ISVD₃: Decompose, Align, Solve

C.5 ISVD₄: Decompose, Align, Solve, Recompute

The final, and most complete, algorithm, ISVD₄, operates similarly to ISVD₃. However, after obtaining interval-valued U matrix, it re-solves for the interval-valued V matrix. As discussed in Section 4.5 of the manuscript and visualized in Figures 4 and 5) the reason for this additional step in ISVD₄ is that the computation of the eigenvectors in U benefits from the latent-space alignment process in the previous step, but the eigenvectors in V are the original eigenvectors (even if they have been realigned). Our hypothesis is that recomputing V (using the properly estimated U) values will lead to eigenvectors that are better representative of the intervals than the original estimates of V. In fact, as we see by comparing Figure 5(a) and Figure 5(b), the re-estimation of V after computation of U leads to...
tighter intervals in $V$ (i.e., more precise eigenvectors). This is also confirmed in the experiments (Section 6).

**Algorithm 11 ISVD**

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

**Output:**
Decomposition target option (a):
Interval-valued factor matrices $U_i, V_i \in \mathbb{R}^{n \times r}$ and interval-valued singular matrix $\Sigma_i \in \mathbb{R}^{r \times r}$.

Decomposition target option (b):
Scalar-valued factor matrices $U_{avg} \in \mathbb{R}^{n \times r}, V_{avg} \in \mathbb{R}^{m \times r}$ and interval-valued singular matrix $\Sigma_{avg} \in \mathbb{R}^{r \times r}$.

Decomposition target option (c):
Scalar-valued factor matrices $U_{avg} \in \mathbb{R}^{n \times r}, V_{avg} \in \mathbb{R}^{m \times r}$ and scalar-valued singular matrix $\Sigma_{avg} \in \mathbb{R}^{r \times r}$.

1: **procedure** ISVD$_4(M_i, r, \text{optDT}, \text{condTh})$
2: \hspace{1em} $A_i \leftarrow \text{INTERVAL-MAT-MULT}(M_i, M_i^T)$ \hspace{1em} \(\triangleright \text{See Algorithm 1}\)
3: \hspace{1em} $V_*, \Sigma^2_* \leftarrow \text{Eig}(A_*, r) \hspace{1em} \triangleright \text{Eigen Decomposition}\)
4: \hspace{1em} $V^*, \Sigma_*^2 \leftarrow \text{Eig}(A_*^*, r)$
5: \hspace{1em} $\text{mappingIdx, dirFlag} \leftarrow \text{ILSA}(V_*, V^*) \hspace{1em} \triangleright \text{See Algorithm 6}\)
6: \hspace{1em} for $j \leftarrow 1$ to $r$ do \hspace{1em} \triangleright \text{Apply remapping}\)
7: \hspace{2em} $V_[:, j] \leftarrow V_[:, \text{mappingIdx}[j]]$
8: \hspace{2em} $\Sigma_[, j] \leftarrow \Sigma_[, j, \text{mappingIdx}[j]]$
9: \hspace{1em} if $\text{dirFlag}[j] = 1$ then \hspace{1em} \triangleright \text{Swap misaligned vectors}\)
10: \hspace{2em} $V_[:, j] \leftarrow V_[:, j] \cdot (-1)$
11: \hspace{1em} end if
12: \hspace{1em} end for
13: \hspace{1em} $V_{inv} \leftarrow V_* + V^*$
14: \hspace{1em} $\text{condNum} \leftarrow \text{cond}(V_{avg})$
15: \hspace{1em} $n, m \leftarrow \text{size}(V_{avg})$
16: \hspace{1em} if $\text{condNum} > \text{condTh}r \vee n \neq m$ then
17: \hspace{2em} $V_{inv} \leftarrow \text{pinv}(V_{avg}) \hspace{1em} \triangleright \text{Moore-Penrose pseudo-inverse}\)
18: \hspace{1em} else
19: \hspace{2em} $V_{inv} \leftarrow V_{avg}^{-1}$
20: \hspace{1em} end if
21: \hspace{1em} $\Sigma_{inv} \leftarrow \text{INVERSE}\Sigma(\Sigma_*, \Sigma^*) \hspace{1em} \triangleright \text{See Algorithm 4}\)
22: \hspace{1em} $U_1 \leftarrow \text{INTERVAL-MAT-MULT}(M_i, (V_{avg} \cdot \Sigma_{inv})) \hspace{1em} \triangleright \text{See Algorithm 1}\)
23: \hspace{1em} end procedure

**Algorithm 12 Reconstruction Target (a)**

**Input:**
Interval-valued factor matrices $U_i \in \mathbb{R}^{n \times r}, V_i \in \mathbb{R}^{m \times r}$ and interval-valued singular matrix $\Sigma_i \in \mathbb{R}^{r \times r}$.

**Output:**
Interval-valued matrix $\tilde{M_i} = [M_*, \tilde{M}^*] \in \mathbb{R}^{n \times m}$.

1: **procedure** TARGET$_a(U_i, \Sigma_i, V_i)$
2: \hspace{1em} $T_i \leftarrow \text{INTERVAL-MAT-MULT}(U_i, \Sigma_i) \hspace{1em} \triangleright \text{See Algorithm 1}\)
3: \hspace{1em} $\tilde{M_i} \leftarrow \text{INTERVAL-MAT-MULT}(T_i, V_i^T)$

D **Pseudocodes for Reconstruction Algorithms**

While it is not part of the decomposition process itself, we sometimes need to (approximately) reconstruct the input matrix (see Figure 4 in the main manuscript) – for example, to use the reconstructed matrix as a tool to estimate the missing values in the original matrix. In the experiments reported in the paper, we use reconstruction to evaluate the accuracy of the decomposition process. In this section, we provide pseudocodes describing the matrix reconstruction processes depending on which decomposition target has been selected by the user (see Section 3.4 and Figure 4).

D.1 **Reconstruction Target (a): Interval-valued** $U_i, \Sigma_i, V_i$

**Algorithm 12 Reconstruction Target (a)**

**Input:**
Interval-valued factor matrices $U_i \in \mathbb{R}^{n \times r}, V_i \in \mathbb{R}^{m \times r}$ and interval-valued singular matrix $\Sigma_i \in \mathbb{R}^{r \times r}$.

**Output:**
Interval-valued matrix $\tilde{M_i} = [M_*, \tilde{M}^*] \in \mathbb{R}^{n \times m}$.
D.2 Reconstruction Target (b): Scalar $U$, $V_1$, Interval $\Sigma_1$

Algorithm 13 Reconstruction Target (b)

Input:
Scalar-valued factor matrices $U_{avg} \in \mathbb{R}^{n \times r}$, $V_{avg} \in \mathbb{R}^{m \times r}$ and interval-valued singular matrix $\Sigma_{avg} \in \mathbb{R}^{r \times r}$.

Output:
Interval-valued matrix $\tilde{M}_1 = [\tilde{M}_*, \tilde{M}^*] \in \mathbb{R}^{n \times m}$.

1: procedure REPTAR(b)($U_{avg}$, $\Sigma_{avg}$, $V_{avg}$)
2: $\tilde{M}_* \leftarrow U_{avg} \cdot \Sigma_{avg} \cdot V_{avg}^T$
3: $\tilde{M}^* \leftarrow U_{avg} \cdot \Sigma_{avg} \cdot V_{avg}^T$
4: $\tilde{M}_1 \leftarrow [\tilde{M}_*, \tilde{M}^*]$
5: $\tilde{M}_1 \leftarrow \text{AVERAGE-REPLACEMENT-MAT}(\tilde{M}_1)$ \> See Algorithm 3
6: return $\tilde{M}_1$
7: end procedure

D.3 Reconstruction Target (c): Scalar $U$, $\Sigma$, $V$

Algorithm 14 Reconstruction Target (c)

Input:
Scalar-valued factor matrices $U_{avg} \in \mathbb{R}^{n \times r}$, $V_{avg} \in \mathbb{R}^{m \times r}$ and scalar-valued singular matrix $\Sigma_{avg} \in \mathbb{R}^{r \times r}$.

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

1: procedure REPTAR(c)($U_{avg}$, $\Sigma_{avg}$, $V_{avg}$)
2: $\tilde{M}_{avg} \leftarrow U_{avg} \cdot \Sigma_{avg} \cdot V_{avg}^T$
3: return $\tilde{M}_{avg}$
4: end procedure

E PSEUDOCODES FOR ALIGNED INTERVAL PROBABILISTIC MATRIX FACTORIZATION (AI-PMF)

As discussed in Section 5 of the manuscript, 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 projections. Below, we provide the pseudocode of the AI-PMF algorithm.

Algorithm 15 AI-PMF

Input:
Interval-valued matrix $M_1 = [M_*, M^*] \in \mathbb{R}^{n \times m}$, target rank $r$, number of epoch $nEpoch$ and number of batches $nBatch$.

Output:
Scalar-valued factor matrix $\tilde{U} \in \mathbb{R}^{n \times r}$ and interval-valued factor matrix $\tilde{V}_1 \in \mathbb{R}^{m \times r}$.

1: procedure AI-PMF($M_1$, $r$, $nEpoch$, $nBatch$)
2: Initialize random $\tilde{U} \in \mathbb{R}^{n \times r}$
3: Initialize random $\tilde{V}_1 = [\tilde{V}_*, \tilde{V}^*] \in \mathbb{R}^{m \times r}$
4: $\text{batchSize} \leftarrow \frac{\text{nBatch}}{nBatch}$
5: for $c \leftarrow 1$ to $nEpoch$ do
6: Shuffle and divide row vectors in $M_1$ into $\text{batchSize}$ batches.
7: for each batch $B \in \mathbb{R}^{\text{batchSize} \times m}$ do \> See Section 5
8: for $reIdx_i \leftarrow 1$ to $\text{batchSize}$ do
9: $i \leftarrow \text{row index in } M_1 \text{ that mapping to row index } reIdx_i \text{ in } B$
10: for $reIdx_j \leftarrow 1$ to $m$ do
11: $j \leftarrow \text{column index in } M_1 \text{ that mapping to column index } reIdx_j \text{ in } B$
12: $U[i, \cdot] \leftarrow U[i, \cdot] - \frac{\partial E_{I-PMF}}{\partial U[i, \cdot]}$
13: $V_*[j, \cdot] \leftarrow V_*[j, \cdot] - \frac{\partial E_{I-PMF}}{\partial V_*[j, \cdot]}$
14: $\tilde{V}_*[j, \cdot] \leftarrow \tilde{V}_*[j, \cdot] - \frac{\partial E_{I-PMF}}{\partial \tilde{V}_*[j, \cdot]}$
15: end for
16: end for
17: end for
18: $\text{mappingIdx}, \text{dirFlag} \leftarrow \text{ILSA}(\tilde{V}_*, \tilde{V}^*)$ \> See Algorithm 6
19: for $j \leftarrow 1$ to $r$ do \> Apply remapping
20: $V_*[\cdot, j] \leftarrow V_*[\cdot, \text{mappingIdx}[j]]$
21: if $\text{dirFlag}[j] = 1$ then \> Swap misaligned vectors
22: $V_*[\cdot, j] \leftarrow V_*[\cdot, j] \cdot (-1)$
23: end if
24: end for
25: return $U, \tilde{V}_1$
26: end procedure

F ORL FACE AND MOVIELENS DATASETS

In order to have consistent evaluation against PMF and I-PMF techniques reported in [9], we followed the procedure reported in that paper to generate the interval values for the ORL face dataset and MovieLens datasets. While, we include a reference to that paper in the original manuscript and provide an overview of the data generation process, we provide additional details in this supplementary file.

F.1 Intervals in the ORL Face Dataset

ORL face dataset contains 32x32 pixels facial images of 40 individuals. The data set contains 10 images for each individual. These 400 images are organized in the form of a matrix $M \in \mathbb{R}^{n \times m}$, where $n = 400$, $m = 1024$; i.e., each row represents a person’s face and each column represents the pixel value at a different coordinate.

[9] states that the value of a pixel with coordinates $(x, y)$ $x \in \{1, \ldots, d_x\}$, $y \in \{1, \ldots, d_y\}$ corresponds to a pixel with coordinates $(x + \Delta x, y + \Delta y)$, $0 \leq \Delta x, \Delta y \leq r$ in the corresponding image. The $ith$ face is represented by a vector $X_i$, with dimensionality $d = d_x \times d_y$. Here, $(x^{(i)}, y^{(i)})$ denotes the coordinates of pixel in the $ith$ image which corresponds to the $jth$ element in vector $X_i$, namely $X_{ij}$.

Given these, [9] defines the following set of the entries in $X$ for each $X_{ij}$

$$S^{(r)}_{ij} = \{X_{ij} \mid |x^{(i,j)} - x^{(i,j)}| \leq r \land |y^{(i,j)} - y^{(i,j)}| \leq r\} \quad (1)$$
The elements in $S^r_{ij}$ correspond to pixels around $(x^{i,j}, y^{i,j})$ within range $r$. Intuitively, $X_{ij}$ corresponds to a value that can be represented as the interval $[\min(S^r_{ij}), \max(S^r_{ij})]$

However, [9] further argues that this interval is not robust in practice and it constructs an alternative interval, $I(X_{ij})$, that takes into account the standard deviation of the values in $S^r_{ij}$. More specifically, given a range $r$ and a multiplicative scale coefficient, $\alpha \in \mathbb{R}^+$, the interval corresponding to the center point $X_{ij}$ is defined as

$$I(X_{ij}) = [X_{ij} - \delta_{ij}, X_{ij} + \delta_{ij}]$$

(2)

where

$$\delta_{ij} := \alpha \cdot \text{std}(S^r_{ij}).$$

(3)

### F.2 Intervals in the MovieLens Dataset

MovieLens dataset includes 19 genres, 100K ratings, 943 users (objects) and 1682 movies (features). We use this dataset to evaluate the performance for (i) reconstruction evaluation in Section 6.5.1 and (ii) collaborative filtering evaluation in Section 6.5.2.

For reconstruction evaluation, we generate a user-genre interval matrix, where each interval corresponds to the range of ratings a user provided for movies of a given genre. The definition of this interval-valued matrix $R = [R_*, R^*] = \mathbb{R}^{n \times g}$ will be:

$$R_*[i, j] = \min(\text{rating from user } i \text{ to genre } j)$$

$$R^*[i, j] = \max(\text{rating from user } i \text{ to genre } j)$$

(4)

where, $n$ is number of users and $r$ is number of genres.

For collaborative-filtering evaluation, we followed the procedure reported in [9] to generate the interval values for the collaborative filtering dataset: Let $X_{ij}$ denote the rating from user $i$ to movie $j$. [9] first defines a corresponding observations set

$$S_{ij} = \{X_{i'j'} \mid (i' = i \lor j' = j) \land (i', j') \in (i, j)\},$$

(5)

which consists of all ratings by the $i$th user or for the $j$th movie. Given this set, and a multiplicative scale coefficient, $\alpha \in \mathbb{R}^+$, [9] associates the interval

$$I(X_{ij}) = [X_{ij} - \delta_{ij}, X_{ij} + \delta_{ij}]$$

(6)

to $X_{ij}$, where

$$\delta_{ij} := \alpha \cdot \text{std}(S_{ij}).$$

(7)

### G LINK FOR DATASETS AND SOURCE CODES

Our implementation of the algorithms proposed in the paper and the datasets used in the experiments are available at

https://shorturl.at/fuN35