mathematics

MDPI

Article

Integrating Dual Graph Constraints into Sparse Non-Negative Tucker Decomposition for Enhanced Co-Clustering

Jing Han [1,2,†] and Linzhang Lu [1,3,*,†]

[1] School of Mathematical Sciences, Guizhou Normal University, Guiyang 550025, China; jinghanfc@163.com
[2] School of Mathematics and Physics, Anshun University, Anshun 561000, China
[3] School of Mathematical Sciences, Xiamen University, Xiamen 361005, China
* Correspondence: lzlu@xmu.edu.cn
† These authors contributed equally to this work.

Abstract

Collaborative clustering is an ensemble technique that enhances clustering performance by simultaneously and synergistically processing multiple data dimensions or tasks. This is an active research area in artificial intelligence, machine learning, and data mining. A common approach to co-clustering is based on non-negative matrix factorization (NMF). While widely used, NMF-based co-clustering is limited by its bilinear nature and fails to capture the multilinear structure of data. With the objective of enhancing the effectiveness of non-negative Tucker decomposition (NTD) in image clustering tasks, in this paper, we propose a dual-graph constrained sparse non-negative Tucker decomposition NTD (GDSNTD) model for co-clustering. It integrates graph regularization, the Frobenius norm, and an l_1 norm constraint to simultaneously optimize the objective function. The GDSNTD mode, featuring graph regularization on both factor matrices, more effectively discovers meaningful latent structures in high-order data. The addition of the l_1 regularization constraint on the factor matrices may help identify the most critical original features, and the use of the Frobenius norm may produce a more highly stable and accurate solution to the optimization problem. Then, the convergence of the proposed method is proven, and the detailed derivation is provided. Finally, experimental results on public datasets demonstrate that the proposed model outperforms state-of-the-art methods in image clustering, achieving superior scores in accuracy and Normalized Mutual Information.

Keywords: co-clustering; graph regularized; non-negative tucker decomposition; sparse

MSC: 15A69; 62H30; 68U10

Academic Editor: Guy De Tré

Received: 29 September 2025
Revised: 29 October 2025
Accepted: 30 October 2025
Published: 1 November 2025

Citation: Han, J.; Lu, L. Integrating Dual Graph Constraints into Sparse Non-Negative Tucker Decomposition for Enhanced Co-Clustering. *Mathematics* **2025**, *13*, 3494. https://doi.org/10.3390/math13213494

1. Introduction

Non-negative decomposition is a general term for a type of matrix or tensor decomposition method that requires all components in the decomposition result to be non-negative. The core idea is to decompose a non-negative data matrix or tensor into the product of several non-negative factor matrices or core tensors in order to extract interpretable latent features or components from non-negative data. The most representative non-negative decompositions are non-negative matrix factorization (NMF) and its extension in the high-dimensional domain non-negative Tucker decomposition (NTD). NMF is a common dimensionality reduction method that can extract the main features from the original data and has high interpretability [1]. When data requires more than two indices to be

uniquely identified, it should be expressed in the form of a tensor. Thus, a tensor is naturally viewed as a generalization of vectors (first-order tensors) and matrices (second-order tensors) to higher orders. Non-negative tensor decomposition is a multi-dimensional data analysis method based on matrix decomposition, which can transform high-dimensional data into low-dimensional representations and retain the main features of the original data [2]. Shcherbakova et al. investigated the advantages of non-negative Tucker decomposition [3]. Like matrix decomposition, the purpose of tensor decomposition is to extract the information or main components hidden in the original data [4,5]. The Tucker decomposition is a tensor factorization method designed for handling tensor data [6]. NTD is a concrete application of Tucker decomposition in non-negative matrix decomposition. Its core idea is to decompose a high-order non-negative tensor into the product of a core tensor and a series of factor matrices [7]. NTD not only retains the interpretability benefits of NMF but also effectively preserves the inherent multi-way structure of the original tensor data. Consequently, non-negative Tucker decomposition (NTD) is extensively utilized in a wide range of disciplines, including image analysis and text mining, to uncover the latent structures inherent in multi-way datasets.

To improve model performance, numerous regularized extensions of NMF and NTD have been developed in the recent literature. These models have demonstrated excellent performance in image clustering or data mining. Cai et al. introduced the graph-regularized non-negative matrix factorization (GNMF) algorithm, which incorporates a geometrically-based affinity graph into the NMF framework to preserve the local manifold structure of the data [8]. Sun et al. proposed graph-regularized and sparse non-negative matrix factorization with hard constraints (GSNMFC), jointly incorporating a graph regularizer and hard prior label information as well as a sparseness constraint as additional conditions to uncover the intrinsic geometrical and discriminative structures of the data space [9]. They also proposed sparse dual graph-regularized non-negative matrix factorization (SDGNMF), jointly incorporating the dual graph-regularized and sparseness constraints as additional conditions to uncover the intrinsic geometrical, discriminative structures of the data space [10]. Shang et al. proposed a novel algorithm, called graph dual regularization non-negative matrix factorization (DNMF), which simultaneously considers the geometric structures of both the data manifold and the feature manifold [11]. Long et al. proposed a novel constrained non-negative matrix factorization algorithm, called the graph-regularized discriminative non-negative matrix factorization (GDNMF), to incorporate into the NMF model both the intrinsic geometrical structure and discriminative information [12]. Saberi-Movahed et al. present a systematic analysis of NMF in dimensionality reduction, with a focus on both feature extraction and feature selection approaches [13]. Jing et al. proposed a novel semi-supervised NMF method that incorporates label regularization, basis regularization, and graph regularization [14]. Li et al. developed a manifold regularization non-negative Tucker decomposition (MR-NTD) model. To preserve the geometric information within tensor data, their method employs a manifold regularization term on the core tensor derived from the Tucker decomposition [15]. Yin and Ma incorporated this geometrically based Locally Linear Embedding (LLE) into the original NTD, thus proposing NTD-LLE for the clustering of image databases [16]. To enhance the representation learning of tensor data, Qiu et al. proposed a novel graph-regularized non-negative Tucker decomposition (GNTD) framework [17]. This method is designed to jointly extract low-dimensional parts-based representations and preserve the underlying manifold structure within the high-dimensional tensor data.

The above research demonstrates that the non-negative decomposition model significantly improves image clustering performance. However, advancements in scientific technologies have led to increasingly complex phenomena, rendering previous models

inadequate for handling the resulting complexities. Building on this foundation, a common strategy to enhance image clustering accuracy has been the development of collaborative clustering (co-clustering) frameworks. This is typically achieved by incorporating specific regularization terms that capture the relationships between different data views or clusters. Co-clustering is an ensemble learning method that performs simultaneous clustering along multiple dimensions or data views [18]. When using the non-negative matrix model, the goal of co-clustering is to simultaneously identify the clusters of both the rows and columns of the two-dimensional data matrix [19]. Del Buono and Pio present a process which aims at enhancing the performance of three-factor NMF as a co-clustering method, by identifying a clearer correlation structure represented by the block matrix [20]. Deng et al. proposed graph-regularized sparse NMF (GSNMF) and graph-regularized sparse non-negative matrix tri-factorization (GSNMTF) models. By incorporating an L_1 norm constraint on the low-dimensional matrix, they aimed to scale the data eigenvalues and enforce sparsity. This co-clustering approach has been shown to enhance the performance of standard non-negative matrix factorization models [21]. Chachlakis, Dhanaraj, Prater-Bennette, and Markopoulos presented Dynamic L_1-Tucker: an algorithm for dynamic and outlier-resistant Tucker analysis of tensor data [22]. The L_1 norm is extensively used in convex optimization problems [23]. Ahmed et al. study a tensor-structured linear regression model over the space of sparse, low Tucker-rank tensors [24]. As deep learning models become larger and more complex, sparsity is emerging as a critical consideration for enhancing efficiency and scalability, making it a central theme in the development of new image processing and data analysis methods.

As illustrated by the above studies, building upon NMF, scholars have developed numerous models to address the evolving needs of various scientific fields. The performance of the NMF model can be significantly improved through the combined constraints of graph regularization and sparsity, which directly exploit the internal structure and inherent characteristics of the data. NTD is the extension of NMF to the high-dimensional domain. However, there are relatively few co-clustering methods for constructing high-performance NTD models that leverage the internal structure and inherent characteristics of the tensor data itself. While GNTD captures graph structure and sparse NMTF promotes sparsity, neither is designed to simultaneously learn from multiple graphs while enforcing directional sparsity patterns across different data modes. To address this gap, and inspired by advancements in co-clustering NMF, this paper proposes a GDSNTD model based on GNTD for enhanced co-clustering. The new model combines graph regularization, the Frobenius norm, and the l_1 norm to simultaneously optimize the objective function. In NTD, graph regularization serves to preserve the multi-linear structure of the original data. The review of the prior literature revealed that imposing multiple graph constraints on NMF models enhances their clustering performance. Motivated by this finding, we consequently introduce dual graph constraints into the NTD framework, applying them directly to the factor matrices of a tensor. This approach allows the model to capture the intrinsic data geometry more clearly, thereby improving clustering accuracy. Furthermore, we provide updated iterative optimization rules and prove the convergence of the model. Experiments on public datasets demonstrate that the proposed method outperforms several leading state-of-the-art methods.

The main contributions of this study are as follows:

1. We introduce dual graph constraints into the NTD framework, applying them directly to the factor matrices of a tensor. To the best of our knowledge, no existing NTD framework integrates dual graph constraints with sparse regularization simultaneously. While graph-regularized and sparse factorization techniques exist, our model GDSNTD is the first to integrate them in a unified, constrained co-clustering opti-

mization framework for NTD. We propose a new co-clustering version of the NTD model, equipped with three regularization terms: graph regularization, the Frobenius norm, and the l_1 norm. The graph regularization term captures the internal geometric structure of high-dimensional data more accurately. The l_1 norm term helps to scale original features in the factor matrices. The Frobenius norm improves the generalization ability of the model. Therefore, the co-clustering GDSNTD integrates strengths from graph-regularized and sparse factorization techniques so that the model yields a more accurate solution to the optimization problem.

2. In the novel, unified optimization objective, we leverage the L-Lipschitz condition to derive the update rules for the proposed co-clustering GDSNTD method. Subsequently, we establish the convergence of the proposed algorithm.
3. Experiments on public datasets demonstrate the effectiveness and superiority of the proposed method.

The remainder of the paper is organized as follows. In Section 2, we review the related models. In Section 3, the GDSNTD method is proposed, and its detailed inference process and the proof of convergence of the algorithm are illustrated. Section 4 presents the performance of the proposed model via experiments on various datasets. Finally, we present our conclusions in Section 5 and outline future work in Section 6.

2. Preliminaries

In this section, we review the related models.

(A) NTD

Given a non-negative data tensor $\mathcal{X} \in \mathbb{R}_+^{I_1 \times I_2 \times I_3}$, non-negative tucker decomposition (NTD) aims at decomposing the non-negative tensor $\mathcal{X}$ into a non-negative core tensor $\mathcal{G} \in \mathbb{R}_+^{I_1 \times I_2 \times I_3}$ multiplied by N non-negative factor matrices $\mathbf{U} \in \mathbb{R}_+^{I_1 \times J_1}$, $\mathbf{V} \in \mathbb{R}_+^{I_2 \times J_2}$, and $\mathbf{W} \in \mathbb{R}_+^{I_3 \times J_3}$ along each mode [7]. NTD minimizes the sum of squared residues between the data tensor $\mathcal{X}$ and the multi-linear product of core tensor $\mathcal{G}$ and factor matrices $\mathbf{U}$, $\mathbf{V}$, and $\mathbf{W}$, which is expressed as

$$F_{NTD} = \min_{\mathcal{G} \geqslant 0, \mathbf{U}, \mathbf{V}, \mathbf{W} \geqslant 0} \left\| \mathcal{X} - \mathcal{G} \times_1 \mathbf{U} \times_2 \mathbf{V} \times_3 \mathbf{W} \right\|_F^2.$$

NTD provides an effective embedding and representation for image tensor data $\mathcal{X}$. However, NTD does not consider the geometrical structure of image tensor data $\mathcal{X}$.

(B) GNTD

Qiu et al. [17] proposed the GNTD model, which incorporates the graph regularization term into the original NTD method. The GNTD is mathematically formulated as

$$F_{GNTD} = \min_{\mathcal{G} \geqslant 0, \mathbf{U}, \mathbf{V}, \mathbf{W} \geqslant 0} \left\| \mathcal{X} - \mathcal{G} \times_1 \mathbf{U} \times_2 \mathbf{V} \times_3 \mathbf{W} \right\|_F^2 + \lambda Tr(\mathbf{W}^\top \mathbf{L} \mathbf{W}),$$

where $\lambda \geqslant 0$ is the regularization parameter for balancing the importance of the graph regularization term and the reconstruction error term, tr$(\cdot)$ denotes the trace of the matrix, and $\mathbf{L}$ is a Laplacian matrix which characterizes the data manifold. $\mathbf{L}$ is defined as $\mathbf{L} = \mathbf{H} - \mathbf{A}$, and $(\mathbf{H})_{ii} = \sum_j (\mathbf{A})_{ij}$. $\mathbf{A}$ is the adjacency matrix of a p-nearest neighbor graph regarding the data points $(\mathbf{Y}_1, \ldots, \mathbf{Y}_2, \ldots)$ constructed using the $0-1$ scheme.

$$(\mathbf{A})_{ij} = \begin{cases} 1, & \text{if } \mathbf{Y}_i \in N_p(\mathbf{Y}_j) \text{ and } \mathbf{Y}_j \in N_p(\mathbf{Y}_i), \\ 0, & \text{otherwise,} \end{cases}$$

where $N_p(\mathbf{Y}_j)$ represents the set of p-nearest neighbor points of $(\mathbf{Y}_j)$.

The GNTD algorithm ingeniously combines graph constraints with non-negative constraints. This approach effectively preserves the local geometric structure of the data, enabling the learned features to better capture its intrinsic structure. As a result, GNTD significantly enhances the performance of traditional tensor decomposition models and has become a powerful tool for handling complex high-dimensional data, receiving considerable attention in machine learning and data mining.

(C) GSNMTF

The graph-regularized sparse non-negative matrix tri-factorization (GSNMTF) model incorporates graph regularization and the l_1 norm into the standard NMF objective function [21]. This integration not only preserves the geometric structure of the data and feature spaces but also promotes sparsity in the resulting factor matrices. The objective function of GSNMTF is defined as

$$F_{GSNMTF} = \min_{\mathbf{U}\geqslant 0,\mathbf{S}\geqslant 0,\mathbf{V}\geqslant 0} \left\|\mathbf{X}-\mathbf{U}\mathbf{S}\mathbf{V}^\top\right\|_F^2 + \lambda_U Tr(\mathbf{U}^\top \mathbf{L}_\mathbf{U}\mathbf{U}) + \lambda_V Tr(\mathbf{V}^\top \mathbf{L}_\mathbf{V}\mathbf{V}) + \gamma_U \left\|\mathbf{U}\right\|_1^1 + \gamma_V \left\|\mathbf{V}\right\|_1^1,$$

where tr(·) is the trace of the matrix, and γ_U and γ_V are the parameters for controlling the sparseness of the matrices $\mathbf{U}$ and $\mathbf{V}$, respectively. $\left\|\cdot\right\|_F$ and $\left\|\cdot\right\|_1$ are the Frobenius norm and l_1 norm, respectively.

3. Dual Graph Constraints into Sparse Non-Negative Tucker Decomposition (GDSNTD) Model

This section introduces the proposed GDSNTD model. We begin with the problem setup. Given a third-order tensor $\mathcal{X} \in \mathbb{R}_+^{I_1\times I_2\times I_3}$, where I_1, I_2, and I_3 are the numbers of rows, columns, and tubes of tensor, respectively. The Tucker decomposition of the tensor is given by

$$\mathcal{X} \approx \mathcal{G} \times_1 \mathbf{U} \times_2 \mathbf{V} \times_3 \mathbf{W}. \tag{1}$$

It decomposes a tensor into a core tensor multiplied by a matrix along each mode. Here the tensor $\mathcal{G} \in \mathbb{R}_+^{J_1\times J_2\times J_3}$ is called the core tensor and its entries show the level of interaction between the different components. $\mathbf{U} \in \mathbb{R}_+^{I_1\times J_1}$, $\mathbf{V} \in \mathbb{R}_+^{I_2\times J_2}$, and $\mathbf{W} \in \mathbb{R}_+^{I_3\times J_3}$ are the factor matrices (which are usually orthogonal) and can be thought of as the principal components in each mode, and $\{J_1, J_2, J_3\} \ll min\{I_1, I_2, I_3\}$.

3.1. Data and Feature Graphs

Many research studies have demonstrated that geometric structure not only affects the data space but also the feature space and that high-dimensional data are usually located in a low-dimensional sub-manifold of the ambient space, and this underlying geometrical information of data could be obtained by modeling a neighbor graph. We impose a graph regularization constraint on the factor matrix $\mathbf{V}$. We construct a p-nearest neighbors data space, whose vertices are $\{\mathcal{X}_{:,:,1}, \cdots, \mathcal{X}_{:,:,N}\}$. Then, we encode their geometrical information by connecting each tensor subject with its p-nearest neighbors, thereby constructing a p-nearest neighbor graph [8] with a binary $(0-1)$ weighting scheme. The weight matrix is defined as

$$[\mathbf{C}_\mathbf{V}]_{ijk} = \begin{cases} 1 & \text{if } \mathcal{X}_{j,:,k} \in \mathcal{N}_p(\mathcal{X}_{i,:,k}) \text{ and } \mathcal{X}_{i,:,k} \in N_p(\mathcal{X}_{j,:,k}), \\ 0 & \text{otherwise,} \end{cases}$$

where $\mathcal{N}_p(\mathcal{X}_{j,:,k})$ represents the set of p samples closest to $\mathcal{X}_{i,:,k}$ in the graph. The graph Laplacian of the data graph is defined as $\mathbf{L_V} = \mathbf{D_V} - \mathbf{C_V}$, where $\mathbf{D_V}$ is the diagonal degree matrix whose elements are given by $[\mathbf{D_V}]_{ii} = \sum_j [\mathbf{C_V}]_{ij}$.

Similarly, we impose a graph regularization constraint on the factor matrix $\mathbf{W}$. We can also construct a p-nearest neighbors graph of the feature space and define the weight matrix as follows:

$$[\mathbf{C_W}]_{ijk} = \begin{cases} 1 & \text{if } \mathcal{X}_{k,j,:} \in \mathcal{N}_p(\mathcal{X}_{i,j,:}) \text{ and } \mathcal{X}_{i,j,:} \in N_p(\mathcal{X}_{k,j,:}), \\ 0 & \text{otherwise,} \end{cases}$$

where $\mathcal{N}_p(\mathcal{X}_{k,j,:})$ represents the set of p samples closest to $\mathcal{X}_{i,j,:}$ in the graph. The graph Laplacian of the data graph is defined as $\mathbf{L_W} = \mathbf{D_W} - \mathbf{C_W}$.

3.2. Objective Function of GDSNTD

In this part, we present our objective function of GDSNTD, which is designed to approximate the original high-dimensional tensor by yielding cleaner and sparser low-dimensional matrices. This model is able to present the geometric structure of tensors more clearly than GNTD and makes up for the deficiency of GSNMTF in handling high-dimensional data. The proposed model has advanced the development of the NTD framework by improving its convergence properties and has facilitated its application in complex tasks such as image clustering analysis. It integrates graph regularization, the Frobenius norm, and a sparsity constraint, and its objective function is defined as

$$\begin{aligned} F_{GDSNTD} = \min_{\mathcal{G}\geqslant 0, \mathbf{U},\mathbf{V},\mathbf{W}\geqslant 0} & \left\| \mathcal{X} - \mathcal{G} \times_1 \mathbf{U} \times_2 \mathbf{V} \times_3 \mathbf{W} \right\|_F^2 + \alpha_V \left\| \mathbf{V} \right\|_F^2 + \alpha_W \left\| \mathbf{W} \right\|_F^2 + \\ & \lambda_V Tr(\mathbf{V}^\top \mathbf{L_V} \mathbf{V}) + \lambda_W Tr(\mathbf{W}^\top \mathbf{L_W} \mathbf{W}) + \gamma_V \left\| \mathbf{V} \right\|_1^1 + \gamma_W \left\| \mathbf{W} \right\|_1^1, \end{aligned} \tag{2}$$

where tr(·) is the trace of the matrix. α_V is the parameter of $\mathbf{V}$, α_W is the parameter of $\mathbf{W}$. λ_V and λ_W are the graph regularization parameters of $\mathbf{V}$ and $\mathbf{W}$, respectively. They govern the trade-off between preserving the intrinsic graph structure of the data and optimizing other objective terms. γ_V and γ_W are coefficients of the l_1 norm. They serve as a trade-off between the sparsity of the factor matrices and other objective terms. It is worth noting that the norms in the objective function are entrywise norms [25]. $\left\| \cdot \right\|_F$ and $\left\| \cdot \right\|_1$ represent the Frobenius norm and l_1 norm, respectively.

Equivalently, Equation (2) is to be rewritten in matrix form as

$$\begin{aligned} F_{GDSNTD} = \min_{\mathcal{G}\geqslant 0, \mathbf{U},\mathbf{V},\mathbf{W}\geqslant 0} & \left\| \mathbf{X}_{(1)} - \mathbf{U}\mathbf{G}_{(1)}(\mathbf{W} \otimes \mathbf{V})^\top \right\|_F^2 + \alpha_V \left\| \mathbf{V} \right\|_F^2 + \alpha_W \left\| \mathbf{W} \right\|_F^2 + \\ & \lambda_V Tr(\mathbf{V}^\top \mathbf{L_V} \mathbf{V}) + \lambda_W Tr(\mathbf{W}^\top \mathbf{L_W} \mathbf{W}) + \gamma_V \left\| \mathbf{V} \right\|_1^1 + \gamma_W \left\| \mathbf{W} \right\|_1^1, \end{aligned} \tag{3}$$

where the tensor $\mathcal{X}$ is expanded according to mode-1. When the tensor $\mathcal{X}$ is expanded according to mode-2 or mode-3, it is to be rewritten in the following forms as Equations (4) and Equation (5), respectively:

$$\begin{aligned} F_{GDSNTD} = \min_{\mathcal{G}\geqslant 0, \mathbf{U},\mathbf{V},\mathbf{W}\geqslant 0} & \left\| \mathbf{X}_{(2)} - \mathbf{V}\mathbf{G}_{(2)}(\mathbf{W} \otimes \mathbf{U})^\top \right\|_F^2 + \alpha_V \left\| \mathbf{V} \right\|_F^2 + \alpha_W \left\| \mathbf{W} \right\|_F^2 + \\ & \lambda_V Tr(\mathbf{V}^\top \mathbf{L_V} \mathbf{V}) + \lambda_W Tr(\mathbf{W}^\top \mathbf{L_W} \mathbf{W}) + \gamma_V \left\| \mathbf{V} \right\|_1^1 + \gamma_W \left\| \mathbf{W} \right\|_1^1, \end{aligned} \tag{4}$$

$$F_{GDSNTD} = \min_{\mathcal{G}\geqslant 0,\mathbf{U},\mathbf{V},\mathbf{W}\geqslant 0} \left\|\mathbf{X}_{(3)} - \mathbf{W}\mathbf{G}_{(3)}(\mathbf{V}\otimes\mathbf{U})^\top\right\|_F^2 + \alpha_V\left\|\mathbf{V}\right\|_F^2 + \alpha_W\left\|\mathbf{W}\right\|_F^2 + \lambda_V Tr(\mathbf{V}^\top\mathbf{L}_\mathbf{V}\mathbf{V}) + \lambda_W Tr(\mathbf{W}^\top\mathbf{L}_\mathbf{W}\mathbf{W}) + \gamma_V\left\|\mathbf{V}\right\|_1^1 + \gamma_W\left\|\mathbf{W}\right\|_1^1. \tag{5}$$

3.3. Inference of GDSNTD

In this section, the detailed inference is illustrated, and then, a corresponding algorithm is designed.

3.3.1. Fix $\mathbf{U}\in\mathbb{R}_+^{I_1\times J_1}$, $\mathbf{V}\in\mathbb{R}_+^{I_2\times J_2}$, $\mathbf{W}\in\mathbb{R}_+^{I_3\times J_3}$, Solve $\mathcal{G}$

Letting $\lambda_G G$ be the Lagrange multiplier [26] for constraint $\mathcal{G}\geqslant 0$, and our objective function is written in matrix form as

$$\widehat{\mathcal{L}}(\mathcal{G}) = \left\|vec(\mathcal{X}) - (\mathbf{W}\otimes\mathbf{V}\otimes\mathbf{U})vec(\mathcal{G})\right\|_F^2 + vec(\mathcal{G})^\top vec(\lambda_G). \tag{6}$$

To obtain the optimal solution, the gradient descent algorithm [27] is used to solve Equation (6). The gradient is as follows:

$$\frac{\partial\widehat{\mathcal{L}}}{\partial vec(\mathcal{G})} = 2(\mathbf{W}\otimes\mathbf{V}\otimes\mathbf{U})^\top(\mathbf{W}\otimes\mathbf{V}\otimes\mathbf{U})vec(\mathcal{G}) - 2(\mathbf{W}\otimes\mathbf{V}\otimes\mathbf{U})^\top vec(\mathcal{X}) + vec(\lambda_G). \tag{7}$$

Let $\mathcal{G}(t)$ represent the corresponding value during the tth round of updates. Therefore, using the Karush–Kuhn–Tucker (KKT) condition [27], $\frac{\partial\mathcal{L}}{\partial vec(\mathcal{G})} = 0$ and $vec(\mathcal{G})_i vec(\geq_\mathcal{G})_i = 0$, we obtain the following updating rule:

$$(vec(\mathcal{G}))_i \leftarrow (vec(\mathcal{G}))_i \frac{((\mathbf{W}\otimes\mathbf{V}\otimes\mathbf{U})^\top vec(\mathcal{X}))_i}{((\mathbf{W}\otimes\mathbf{V}\otimes\mathbf{U})^\top(\mathbf{W}\otimes\mathbf{V}\otimes\mathbf{U})vec(\mathcal{G}))_i}. \tag{8}$$

3.3.2. Fix $\mathcal{G}$, $\mathbf{V}\in\mathbf{R}_+^{I_2\times J_2}$ and $\mathbf{W}\in\mathbf{R}_+^{I_3\times J_3}$, Solve $\mathbf{U}\in\mathbf{R}_+^{I_1\times J_1}$.

At this stage, similar to the update of $\mathcal{G}$, letting $\lambda_\mathbf{U}\mathbf{U}$ be the Lagrange multiplier for constraint $\mathbf{U}\geqslant 0$, our optimization goal is to be written in matrix form as follows:

$$\widehat{\mathcal{O}}(\mathbf{U}) = \left\|\mathbf{X}_{(1)} - \mathbf{U}\mathbf{G}_{(1)}(\mathbf{W}\otimes\mathbf{V})^\top\right\|_F^2 + Tr(\mathbf{U}^\top\lambda_\mathbf{U}). \tag{9}$$

The function (10) is to be rewritten as

$$\widehat{\mathcal{O}}(\mathbf{U}) = Tr(\mathbf{X}_{(1)}\mathbf{X}_{(1)}^\top) - 2Tr(\mathbf{X}_{(1)}(\mathbf{W}\otimes\mathbf{V})\mathbf{G}_{(1)}^\top\mathbf{U}^\top) + Tr(\mathbf{U}\mathbf{G}_{(1)}((\mathbf{W}\otimes\mathbf{V}^\top)(\mathbf{W}\otimes\mathbf{V}))\mathbf{G}_{(1)}^\top\mathbf{U}^\top) + \sum_{r=1}^{N} Tr(\mathbf{U}^\top\lambda_\mathbf{U}). \tag{10}$$

Now, the partial derivative of $\widehat{\mathcal{O}}(\mathbf{U})$ to $\mathbf{U}$ is

$$\frac{\partial\widehat{\mathcal{O}}(\mathbf{U})}{\partial(\mathbf{U})} = -2\mathbf{X}_{(1)}(\mathbf{W}\otimes\mathbf{V})\mathbf{G}_{(1)}^\top + 2\mathbf{U}\mathbf{G}_{(1)}((\mathbf{W}\otimes\mathbf{V})^\top(\mathbf{W}\otimes\mathbf{V}))\mathbf{G}_{(1)}^\top + \lambda_\mathbf{U}. \tag{11}$$

According to the (Karush–Kuhn–Tucker) (KKT) condition [27], $\frac{\partial\widehat{\mathcal{O}}(\mathbf{U})}{\partial(\mathbf{U})} = 0$, so

$$\lambda_\mathbf{U} = 2\mathbf{X}_{(1)}(\mathbf{W}\otimes\mathbf{V})\mathbf{G}_{(1)}^\top - 2\mathbf{U}\mathbf{G}_{(r)}((\mathbf{W}\otimes\mathbf{V})^\top(\mathbf{W}\otimes\mathbf{V}))\mathbf{G}_{(r)}^\top.$$

And also because $\lambda_{\mathbf{U}} \odot \mathbf{U} = 0$, $\odot$ denotes the Hadamard product, and we can obtain the following:

$$(2\mathbf{X}_{(1)}(\mathbf{W} \otimes \mathbf{V})\mathbf{G}_{(1)}^{\top} - 2\mathbf{U}\mathbf{G}_{(1)}((\mathbf{W} \otimes \mathbf{V})^{\top}(\mathbf{W} \otimes \mathbf{V}))\mathbf{G}_{(1)}^{\top}) \odot \mathbf{U} = 0. \tag{12}$$

Therefore, the update rule of $\mathbf{U}$ is

$$\mathbf{U}_{ij} \leftarrow \mathbf{U}_{ij} \frac{(\mathbf{X}_{(1)}(\mathbf{W} \otimes \mathbf{V})\mathbf{G}_{(1)}^{\top})_{ij}}{(\mathbf{U}\mathbf{G}_{(1)}((\mathbf{W} \otimes \mathbf{V})^{\top}(\mathbf{W} \otimes \mathbf{V}))\mathbf{G}_{(1)}^{\top})_{ij}}. \tag{13}$$

3.3.3. Fix $\mathcal{G}$, $\mathbf{U} \in \mathbb{R}_{+}^{I_1 \times J_1}$ and $\mathbf{W} \in \mathbb{R}_{+}^{I_3 \times J_3}$, Solve $\mathbf{V} \in \mathbb{R}_{+}^{I_2 \times J_2}$

At this stage, our goal is to minimize the function $\mathcal{P}(\mathbf{V})$. The optimization goal in (4) is to be defined as

$$\mathcal{P}(\mathbf{V}) = \widehat{\mathcal{P}}(\mathbf{V}) + \gamma_V \left\| \mathbf{V} \right\|_1^1, \tag{14}$$

where

$$\widehat{\mathcal{P}}(\mathbf{V}) = \left\| \mathbf{X}_{(2)} - \mathbf{V}\mathbf{G}_{(2)}(\mathbf{W} \otimes \mathbf{U})^{\top} \right\|_F^2 + \alpha_V \left\| \mathbf{V} \right\|_F^2 + \lambda_V Tr(\mathbf{V}^{\top}\mathbf{L}_{\mathbf{V}}\mathbf{V}) + Tr(\lambda_{\mathbf{V}}). \tag{15}$$

Since $\widehat{\mathcal{P}}(\mathbf{V}')$ is a convex function with respect to $\mathbf{V}'$ and it satisfies the L-Lipschitz condition [28] as shown in Equation (14), there exists a constant $L_1 \geqslant 0$ such that

$$\left\| \nabla\widehat{\mathcal{P}}(\mathbf{V}') - \nabla\widehat{\mathcal{P}}(\mathbf{V}) \right\| \leqslant \mathbf{L}_1 \left\| \mathbf{V}' - \mathbf{V} \right\|, \forall(\mathbf{V}', \mathbf{V}), \tag{16}$$

where $\hat{\mathcal{P}}(\mathbf{V}')$ is to be by the approximated second-order Taylor expansion near $\mathbf{V}$ as

$$\begin{aligned} \widetilde{\mathcal{P}}(\mathbf{V}) \approx & \widehat{\mathcal{P}}(\mathbf{V}') + \frac{L_1}{2}(\mathbf{V} - \mathbf{V}')^2 + \frac{\partial\widehat{\mathcal{P}}(\mathbf{V}')}{\partial(\mathbf{V}')}(\mathbf{V} - \mathbf{V}') \\ = & \frac{L_1}{2}(\mathbf{V} - (\mathbf{V}' - \frac{1}{L_1}\frac{\partial\widehat{\mathcal{P}}(\mathbf{V}')}{\partial\mathbf{V}'}))^2 + \Phi(\mathbf{V}'), \\ \Phi(\mathbf{V}') = & \widehat{\mathcal{P}}(\mathbf{V}') - \frac{1}{2L_1}(\frac{\partial\widehat{\mathcal{P}}(\mathbf{V}')}{\partial\mathbf{V}'})^2. \end{aligned} \tag{17}$$

Let $\mathbf{V}' = \mathbf{V}^{(t)}$, the minimum value of Equation (17) is then obtained as $\mathbf{V}^{(t+1)}$ and

$$\mathbf{V}^{(t+1)} = \mathbf{V}^{(t)} - \frac{1}{L_1}\frac{\partial\widehat{\mathcal{P}}(\mathbf{V}^{(t)})}{\partial(\mathbf{V}^{(t)})}. \tag{18}$$

Since $\Phi(\mathbf{V}^{(t)})$ is independent of the optimization variable $\mathbf{V}$, it can be treated as a constant. Consequently, the objective function for updating $\mathbf{V}$ simplifies to

$$\mathbf{V}(t+1) = \arg\min_{\mathbf{V}}[\frac{L_1}{2}\left\| \mathbf{V} - (\mathbf{V}^{(t)} - \tfrac{1}{L_1}\tfrac{\partial\widehat{\mathcal{P}}(\mathbf{V}^{(t)})}{\partial\mathbf{V}^{(t)}}) \right\|_2^2 + \gamma_V \left\| \mathbf{V} \right\|_1^1]. \tag{19}$$

Let $\mathbf{S}^{(t)} = \mathbf{V}^{(t)} - \frac{1}{L_1}\frac{\partial\widehat{\mathcal{P}}(\mathbf{V}^t)}{\partial\mathbf{V}^{(t)}}$, then, for solving Equation (19), we first solve $\mathbf{S}^{(t)}$ and then

$$\mathbf{V}^{(t+1)} = \arg\min_{\mathbf{V}}[\frac{L_1}{2}\left\| \mathbf{V} - \mathbf{S}^{(t)} \right\|_2^2 + \gamma_V \left\| \mathbf{V} \right\|_1^1]. \tag{20}$$

Now, the partial derivative of $\hat{\mathcal{P}}(\mathbf{V}^{(t)})$ to $\mathbf{V}^{(t)}$ is

$$\begin{aligned}\frac{\partial\widehat{\mathcal{P}}(\mathbf{V}^{(t)})}{\partial(\mathbf{V}^{(t)})} =& -2(\mathbf{X}_{(2)}^{(t)} - \mathbf{V}^{(t)}\mathbf{G}_{(2)}^{(t)}(\mathbf{W}^{(t)} \otimes \mathbf{U}^{(t)})^{\top})(\mathbf{G}_{(2)}^{(t)}(\mathbf{W}^{(t)} \otimes \mathbf{U}^{(t)})^{\top})^{\top} + \\ & 2\alpha_V\mathbf{V}^{(t)} + 2\lambda_V\mathbf{L}_{\mathbf{V}}^{(t)}\mathbf{V}^{(t)} + \lambda_{\mathbf{V}^{(t)}} \\ & = -2\mathbf{X}_{(2)}^{(t)}(\mathbf{W}^{(t)} \otimes \mathbf{U}^{(t)}){\mathbf{G}_{(2)}^{(t)}}^{\top} + \\ & 2\mathbf{V}^{(t)}\mathbf{G}_{(2)}^{(t)}(\mathbf{W}^{(t)} \otimes \mathbf{U}^{(t)})^{\top}(\mathbf{W}^{(t)} \otimes \mathbf{U}^{(t)}){\mathbf{G}_{(2)}^{(t)}}^{\top} + \\ & 2\alpha_V\mathbf{V}^{(t)} + 2\lambda_V(\mathbf{D}_{\mathbf{V}}^{(t)} - \mathbf{C}_{\mathbf{V}}^{(t)})\mathbf{V}^{(t)} + \lambda_{\mathbf{V}}^{(t)}.\end{aligned} \tag{21}$$

where $\mathbf{D}_{\mathbf{V}}^{(t)}$ and $\mathbf{C}_{\mathbf{V}}^{(t)}$ should be solved at the same time. According to the (Karush–Kuhn–Tucker) (KKT) condition [27], $\lambda_{\mathbf{V}}\mathbf{V}^{(t)} = 0$, and $\frac{\partial\widehat{\mathcal{P}}(\mathbf{V}^{(t)})}{\partial\mathbf{V}^{(t)}} = 0$, we can obtain the following:

$$\begin{aligned}&(-2\mathbf{X}_{(2)}^{(t)}(\mathbf{W}^{(t)} \otimes \mathbf{U}^{(t)}){\mathbf{G}_{(2)}^{(t)}}^{\top} - 2\lambda_V\mathbf{C}_{\mathbf{V}}^{(t)}\mathbf{V}^{(t)} + \\ &2\mathbf{V}^{(t)}\mathbf{G}_{(2)}^{(t)}(\mathbf{W}^{(t)} \otimes \mathbf{U}^{(t)})^{T}(\mathbf{W}^{(t)} \otimes \mathbf{U}^{(t)}){\mathbf{G}_{(\mathbf{2})}^{(\mathbf{t})}}^{\top} + 2\alpha_V\mathbf{V}^{(t)} + \\ &2\lambda_V\mathbf{D}_{\mathbf{V}}^{(t)}\mathbf{V}^{(t)})\mathbf{V}^{(t)} = 0.\end{aligned} \tag{22}$$

Therefore, the update rule of $\mathbf{S}^{(t)}$ is

$$\mathbf{S}_{ij}^{(t)} = {\mathbf{V}_{ij}}^{(t)}\frac{[{\mathbf{X}_{(2)}}^{(t)}(\mathbf{W}^{(t)} \otimes \mathbf{U}^{(t)}){\mathbf{G}_{(2)}^{(t)}}^{\top} + \lambda_V\mathbf{C}_{\mathbf{V}}^{(t)}\mathbf{V}^{(t)}]_{ij}}{[\mathbf{V}^{(t)}\mathbf{G}_{(2)}^{(t)}(\mathbf{W}^{(t)} \otimes \mathbf{U}^{(t)})^{\top}(\mathbf{W}^{(t)} \otimes \mathbf{U}^{(t)}){\mathbf{G}_{(2)}^{(t)}}^{\top} + \alpha_V\mathbf{V}^{(t)} + \lambda_V\mathbf{D}_{\mathbf{V}}^{(t)}\mathbf{V}^{(t)}]_{ij}}. \tag{23}$$

To solve the objective function in Equation (19), we minimize the $\mathbf{S}_{ij}^{(t)}$ term using the gradient descent and apply a soft threshold [28] to minimize the $\left\|\mathbf{V}\right\|_1^1$ regularizer. Consequently, the update rule for $\mathbf{V}$ is derived as follows:

$${\mathbf{V}_{ij}}^{(t+1)} = \begin{cases}\mathbf{S}_{ij}^{(t)} - \frac{\gamma_V}{L_1}, & \mathbf{S}_{ij}^{(t)} > \frac{\gamma_V}{L_1}, \\ 0, & \mathbf{S}_{ij}^{(t)} \leq \frac{\gamma_V}{L_1}.\end{cases} \tag{24}$$

3.3.4. Fix $\mathcal{G}$, $\mathbf{U} \in \mathbb{R}_+^{I_1 \times J_1}$ and $\mathbf{V} \in \mathbb{R}_+^{I_2 \times J_2}$, Solve $\mathbf{W} \in \mathbb{R}_+^{I_3 \times J_3}$

Similarly, to update the factor matrix $\mathbf{W}$, we write the objective function $\mathcal{Q}(\mathbf{W})$ as follows:

$$\mathcal{Q}(\mathbf{W}) = \widehat{\mathcal{Q}}(\mathbf{W}) + \gamma_W\left\|\mathbf{W}\right\|_1^1, \tag{25}$$

where

$$\widehat{\mathcal{Q}}(\mathbf{W}) = \left\|\mathbf{X}_{(3)} - \mathbf{W}\mathbf{G}_{(3)}(\mathbf{V} \otimes \mathbf{U})^{\top}\right\|_F^2 + \alpha_W\left\|\mathbf{W}\right\|_F^2 + \lambda_W Tr(\mathbf{W}^{\top}\mathbf{L}_{\mathbf{W}}\mathbf{W}) + Tr(\lambda_{\mathbf{W}}). \tag{26}$$

Since $\widehat{\mathcal{Q}}(\mathbf{W}')$ in Equation (25) is a convex function regarding $\mathbf{W}'$, and satisfies the L-Lipschitz condition [28], there exists a constant $L_2 \geqslant 0$ such that

$$\left\|\nabla\widehat{\mathcal{Q}}(\mathbf{W}') - \nabla\widehat{\mathcal{Q}}(\mathbf{W})\right\| \leqslant L_2\left\|\mathbf{W}' - \mathbf{W}\right\|, \forall(\mathbf{W}', \mathbf{W}), \tag{27}$$

where $\nabla\widehat{\mathcal{Q}}(\mathbf{W}')$ is the derivative of $\mathbf{W}'$. $\widehat{\mathcal{Q}}(\mathbf{W}')$ is to be by the approximated second-order Taylor expansion near $\mathbf{W}$ as

$$\begin{aligned}\widetilde{\mathcal{Q}}(\mathbf{W}) \approx & \widehat{\mathcal{Q}}(\mathbf{W}') + \frac{L_2}{2}(\mathbf{W}-\mathbf{W}')^2 + \frac{\partial\widehat{\mathcal{Q}}(\mathbf{W}')}{\partial(\mathbf{W}')}(\mathbf{W}-\mathbf{W}') \\ = & \frac{L_2}{2}(\mathbf{W}-(\mathbf{W}'-\frac{1}{L_2}\frac{\partial\widehat{\mathcal{Q}}(\mathbf{W}')}{\partial\mathbf{W}'}))^2 + \Theta(\mathbf{W}'), \\ \Theta(\mathbf{W}') = & \widehat{\mathcal{Q}}(\mathbf{W}') - \frac{1}{2L_2}(\frac{\partial\widehat{\mathcal{Q}}(\mathbf{W}')}{\partial\mathbf{W}'})^2.\end{aligned} \tag{28}$$

Let $\mathbf{W}' = \mathbf{W}^{(t)}$, then, the minimum value of Equation (28) is obtained as $\mathbf{W}^{(t+1)}$ and

$$\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} - \frac{1}{L_2}\frac{\partial\widehat{\mathcal{Q}}(\mathbf{W}^t)}{\partial(\mathbf{W}^t)}. \tag{29}$$

Similarly, since $\Psi(\mathbf{W}^{(t)})$ is independent of $\mathbf{W}^t$, the objective function updated with $\mathbf{W}$ can be written as

$$\mathbf{W}^{(t+1)} = \arg\min_{\mathbf{W}}[\frac{L_2}{2}\left\|\mathbf{W}-(\mathbf{W}^{(t)}-\tfrac{1}{L_2}\frac{\partial\widehat{\mathcal{Q}}(\mathbf{W}^{(t)})}{\partial\mathbf{W}^{(t)}})\right\|_2^2 + \gamma_W\left\|\mathbf{W}\right\|_1^1]. \tag{30}$$

Let $\mathbf{R}^{(t)} = \mathbf{W}^{(t)} - \frac{1}{L_2}\frac{\partial\widehat{\mathcal{Q}}(\mathbf{W}^t)}{\partial\mathbf{W}^{(t)}}$, then, for solving Equation (30), we first solve $\mathbf{R}^{(t)}$ and then

$$\mathbf{W}^{(t+1)} = \arg\min_{\mathbf{W}}[\frac{L_2}{2}\left\|\mathbf{W}-\mathbf{R}^{(t)}\right\|_2^2 + \gamma_W\left\|\mathbf{W}\right\|_1^1]. \tag{31}$$

Now, the partial derivative of $\widehat{\mathcal{Q}}(\mathbf{W}^{(t)})$ to $\mathbf{W}^{(t)}$ is

$$\begin{aligned}\frac{\partial\widehat{\mathcal{Q}}(\mathbf{W}^{(t)})}{\partial(\mathbf{W}^{(t)})} = & -2(\mathbf{X}_{(3)}^{(t)} - \mathbf{W}^{(t)}\mathbf{G}_{(3)}^{(t)}(\mathbf{V}^{(t)}\otimes\mathbf{U}^{(t)})^\top)(\mathbf{G}_{(3)}^{(t)}(\mathbf{V}^{(t)}\otimes\mathbf{U}^{(t)})^\top)^\top + \\ & 2\alpha_W\mathbf{W}^{(t)} + 2\lambda_W\mathbf{L}_{\mathbf{W}^{(t)}}\mathbf{W}^{(t)} + \lambda_{\mathbf{W}^{(t)}} \\ = & -2\mathbf{X}_{(3)}^{(t)}(\mathbf{V}^{(t)}\otimes\mathbf{U}^{(t)}){\mathbf{G}_{(3)}^{(t)}}^\top + \\ & 2\mathbf{W}^{(t)}\mathbf{G}_{(3)}^{(t)}(\mathbf{V}^{(t)}\otimes\mathbf{U}^{(t)})^\top(\mathbf{V}^{(t)}\otimes\mathbf{U}^{(t)}){\mathbf{G}_{(3)}^{(t)}}^\top + \\ & 2\alpha_3\mathbf{W}^{(t)} + 2\lambda_W(\mathbf{D}_{\mathbf{W}}^{(t)} - \mathbf{C}_{\mathbf{W}}^{(t)})\mathbf{W}^{(t)} + \lambda_{\mathbf{W}}^{(t)},\end{aligned} \tag{32}$$

where $\mathbf{D}_{\mathbf{W}}^{(t)}$ and $\mathbf{C}_{\mathbf{W}}^{(t)}$ should be solved at the same time. According to the (Karush–Kuhn–Tucker) (KKT) condition [27], $\lambda_{\mathbf{W}}\mathbf{W}^{(t)} = 0$, and $\frac{\partial\widehat{\mathcal{Q}}(\mathbf{W}^{(t)})}{\partial\mathbf{W}^{(t)}} = 0$, we can obtain the following:

$$\begin{aligned}&(-2\mathbf{X}_{(3)}^{(t)}(\mathbf{V}^{(t)}\otimes\mathbf{U}^{(t)}){\mathbf{G}_{(3)}^{(t)}}^\top - 2\lambda_W\mathbf{C}_{\mathbf{W}}^{(t)}\mathbf{W}^{(t)} + \\ &2\mathbf{W}^{(t)}\mathbf{G}_{(3)}^{(t)}(\mathbf{V}^{(t)}\otimes\mathbf{U}^{(t)})^\top(\mathbf{V}^{(t)}\otimes\mathbf{U}^{(t)}){\mathbf{G}_{(\mathbf{3})}^{(\mathbf{t})}}^\top + 2\alpha_W{\mathbf{A}^{(3)}}^{(t)} + \\ &2\lambda_W\mathbf{D}_{\mathbf{W}}^{(t)}\mathbf{W}^{(t)})\mathbf{W}^{(t)} = 0.\end{aligned} \tag{33}$$

Therefore, the update rule of $\mathbf{R}^{(t)}$ is

$$\mathbf{R}_{ij}^{(t)} = {\mathbf{W}_{ij}}^{(t)}\frac{[{\mathbf{X}_{(3)}}^{(t)}(\mathbf{V}^{(t)}\otimes\mathbf{U}^{(t)}){\mathbf{G}_{(3)}^{(t)}}^\top + \lambda_W\mathbf{C}_{\mathbf{W}}^{(t)}\mathbf{W}^{(t)}]_{ij}}{[\mathbf{W}^{(t)}\mathbf{G}_{(3)}^{(t)}(\mathbf{V}^{(t)}\otimes\mathbf{U}^{(t)})^\top(\mathbf{V}^{(t)}\otimes\mathbf{U}^{(t)}){\mathbf{G}_{(3)}^{(t)}}^\top + \alpha_W\mathbf{W}^{(t)} + \lambda_W\mathbf{D}_{\mathbf{W}}^{(t)}\mathbf{W}^{(t)}]_{ij}}. \tag{34}$$

Similarly, we minimize $\mathbf{R}_{ij}^{(t)}$ and $\left\|\mathbf{W}\right\|_1^1$ using gradient descent and soft thresholding, respectively. This leads to the following update rule for $\mathbf{W}$:

$$\mathbf{W}_{ij}^{(t+1)} = \begin{cases} \mathbf{R}_{ij}^{(t)} - \frac{\gamma_W}{L_2}, & \mathbf{R}_{ij}^{(t)} > \frac{\gamma_W}{L_2}, \\ 0, & \mathbf{R}_{ij}^{(t)} \leq \frac{\gamma_W}{L_2}. \end{cases} \tag{35}$$

3.4. Algorithm Design

According to the above reasoning, a clustering algorithm based on GDSNTD is summarized, with its detailed steps outlined in Algorithm 1.

Algorithm 1: GDSNTD for Clustering

Input:
Data tensor $\mathcal{X} \in \mathbb{R}_+^{I_1 \times I_2 \times I_3}$. The algorithm parameters J_1, J_2, J_3, p, and regularization
parameters α_V, α_W, λ_V, λ_W, γ_V, γ_W. The stopping criterion and the maximum number
of iterations **maxiter**.
Output: Core tensor $\mathcal{G}$ and factors $\mathbf{U}, \mathbf{V}, \mathbf{W}$;
Ensure:
Clustering results;
repeat
for $k = 1, 2, \ldots,$ maxiter do
1: Update $\mathcal{G}, \mathbf{U}, \mathbf{V}, \mathbf{W}$, and using (8), (13), (24), (35), respectively.
if $re < \epsilon$
Return $\mathcal{G}, \mathbf{U}, \mathbf{V}, \mathbf{W}$
end if
end for
until converges
2: Run k-means on $\mathbf{W}$ to get clustering result.
3: **return Clustering results**

We employ a stopping criterion based on the relative change in the objective function value. The criterion is defined as

$$re = \frac{f^{(k-1)} - f^{(k)}}{f^{(k)}},$$

where $f^{(k-1)}$ and $f^{(k)}$ are the objective function values at the $(k-1)$-th and k-th iterations, respectively. The iterations are terminated when $re < \epsilon$, with ϵ being a predefined convergence tolerance [29].

3.5. Convergence Analysis

Since the update rule for $\mathcal{G}$ is identical to that in [17], this subsection employs the auxiliary function approach exclusively to prove the convergence of the updating rules given in Equations (8), (13), (24) and (35). We now present the following theorem:

Theorem 1. *For $\mathcal{G} \geqslant 0$, $\mathbf{U} \geqslant 0$, $\mathbf{V} \geqslant 0$, and $\mathbf{W} \geqslant 0$, the objective function in (2) is non-increasing under the updating rules in (8), (13), (24), and (35) if and only if $\mathbf{U} \geqslant 0$, $\mathbf{V} \geqslant 0$, $\mathbf{W} \geqslant 0$, and $\mathcal{G}$ are at a stationary point.*

To prove Theorem 1, we first give a definition and several lemmas.

Definition 1 ([1]). *$\mathcal{H}(a, a')$ is an auxiliary function for $\mathcal{J}(a)$ if the conditions $\mathcal{H}(a, a') \geq \mathcal{J}(a)$ and $\mathcal{H}(a, a) = \mathcal{J}(a)$ are satisfied.*

Lemma 1 ([1]). *If $\mathcal{H}(a, a')$ is an auxiliary function for $\mathcal{J}(a)$, then $\mathcal{J}(a)$ is non-increasing under the updating rule*

$$a^{t+1} = \underset{a}{argmin}\mathcal{H}(a, a^t). \tag{36}$$

Proof. $\mathcal{J}(a^{t+1}) \leq \mathcal{H}(a^{t+1}, a^t) \leq \mathcal{H}(a^t, a^t) \leq \mathcal{J}(a^t)$. □

The equality $\mathcal{J}(a^{t+1}) = \mathcal{J}(a^t)$ holds only if a^t is a local minimum of $\mathcal{H}(a, a^t)$. By iterating the update rule (36), a^t converges to the local minimum of $\mathcal{H}(a, a^t)$.

Next, we provide a detailed proof of the convergence for the above updating rule for $\mathbf{U}$ using an auxiliary function. Recall the function from Equation (3):

$$\widehat{\mathcal{J}}(\mathbf{U}) = \left\| \mathbf{X}_{(1)} - \mathbf{U}\mathbf{G}_{(1)}(\mathbf{W} \otimes \mathbf{V})^\top \right\|_F^2. \tag{37}$$

We can obtain

$$\begin{aligned} \widehat{\mathcal{J}}'_{ij}({\mathbf{U}_{ij}}^{(t)}) = \frac{\partial \widehat{\mathcal{J}}({\mathbf{U}_{ij}}^{(t)})}{\partial({\mathbf{U}_{ij}}^{(t)})} = [-2\mathbf{X}_{(1)}^{(t)}(\mathbf{W} \otimes \mathbf{V})^{(t)}{\mathbf{G}_{(1)}}^{(t)^\top} + \\ 2\mathbf{U}^{(t)}{\mathbf{G}_{(1)}}^{(t)}((\mathbf{W} \otimes \mathbf{V})^{(t)^\top}(\mathbf{W} \otimes \mathbf{V})^{(t)}){\mathbf{G}_{(r)}}^{(t)^\top}]_{ij}. \end{aligned} \tag{38}$$

$$\widehat{\mathcal{J}}''_{ij}({\mathbf{U}_{ij}}^{(t)}) = [2{\mathbf{G}_{(1)}}^{(t)}((\mathbf{W} \otimes \mathbf{V})^{(t)^\top}(\mathbf{W} \otimes \mathbf{V})^{(t)}){\mathbf{G}_{(1)}}^{(t)^\top}]_{jj}. \tag{39}$$

Essentially, it is sufficient to prove that each $\widehat{\mathcal{J}}_{ij}(\mathbf{U}_{ij})$ is non-increasing under the update rule.

Lemma 2. *The function*

$$\begin{aligned} \widehat{\mathcal{H}}(\mathbf{U}_{ij}, {\mathbf{U}_{ij}}^t) = \widehat{\mathcal{J}}_{ij}({\mathbf{U}_{ij}}^t) + \widehat{\mathcal{J}}'_{ij}({\mathbf{U}_{ij}}^t)(\mathbf{U}_{ij} - {\mathbf{U}_{ij}}^t) + \\ \frac{[\mathbf{U}^t\mathbf{G}_{(1)}((\mathbf{W} \otimes \mathbf{V})^\top(\mathbf{W} \otimes \mathbf{V}))\mathbf{G}_{(1)}{}^\top]_{jj}}{{\mathbf{U}_{ij}}^t}(\mathbf{U}_{ij} - {\mathbf{U}_{ij}}^t)^2 \end{aligned} \tag{40}$$

is an auxiliary function for $\widehat{\mathcal{J}}_{ij}(\mathbf{U}_{ij})$, where the matrix $\mathbf{U}^t = ({\mathbf{U}_{ij}}^t)$.

Proof. Obviously, $\widetilde{\mathcal{H}}({\mathbf{U}_{ij}}^t, {\mathbf{U}_{ij}}^t) = \widehat{\mathcal{J}}_{ij}({\mathbf{U}_{ij}}^t)$. According to Definition 1, we only need to show that $\widetilde{\mathcal{H}}(\mathbf{U}_{ij}, {\mathbf{U}_{ij}}^t) \geq \widehat{\mathcal{J}}_{ij}(\mathbf{U}_{ij})$. We first obtain the Taylor series expansion of $\widehat{\mathcal{J}}_{ij}(\mathbf{U}_{ij})$ at ${\mathbf{U}_{ij}}^t$ to be

$$\begin{aligned} \widetilde{\mathcal{J}}_{ij}(\mathbf{U}_{ij}) &= \mathcal{J}_{ij}({\mathbf{U}_{ij}}^t) + \widehat{\mathcal{J}}'_{ij}({\mathbf{U}_{ij}}^t)(\mathbf{U}_{ij} - {\mathbf{U}_{ij}}^t) + \frac{1}{2}\mathcal{J}''_{ij}({\mathbf{U}_{ij}}^t)(\mathbf{U}_{ij} - {\mathbf{U}_{ij}}^t)^2 \\ &= \mathcal{J}_{ij}({\mathbf{U}_{ij}}^t) + \mathcal{J}'_{ij}({\mathbf{U}_{ij}}^t)(\mathbf{U}_{ij} - {\mathbf{U}_{ij}}^t) + [\mathbf{G}_{(1)}((\mathbf{W} \otimes \mathbf{V})^\top(\mathbf{W} \otimes \mathbf{V}))\mathbf{G}_{(1)}{}^\top]_{jj}(\mathbf{U}_{ij} - {\mathbf{U}_{ij}}^t)^2. \end{aligned} \tag{41}$$

By Equation (40), $\mathcal{H}(\mathbf{U}_{ij}, {\mathbf{U}_{ij}}^t) \geq \mathcal{J}_{ij}(\mathbf{U}_{ij})$ is equivalent to

$$\frac{[\mathbf{U}^t\mathbf{G}_{(1)}((\mathbf{W} \otimes \mathbf{V})^\top(\mathbf{W} \otimes \mathbf{V}))\mathbf{G}_{(1)}^\top]_{jj}}{{\mathbf{U}_{ij}}^t} \geq [\mathbf{G}_{(r)}((\mathbf{W} \otimes \mathbf{V})^\top(\mathbf{W} \otimes \mathbf{V}))\mathbf{G}_{(1)}^\top]_{jj}. \tag{42}$$

We can obtain

$$\begin{aligned}[\mathbf{U}^t\mathbf{G}_{(1)}((\mathbf{W}\otimes\mathbf{V})^\top(\mathbf{W}\otimes\mathbf{V}))\mathbf{G}_{(1)}^\top]_{jj} &= \mathbf{U}_{il}{}^t[\mathbf{G}_{(1)}((\mathbf{W}\otimes\mathbf{V})^\top(\mathbf{W}\otimes\mathbf{V}))\mathbf{G}_{(1)}^\top]_{lj} \\ &\geq \mathbf{U}_{ij}{}^t[\mathbf{G}_{(1)}((\mathbf{W}\otimes\mathbf{V})^\top(\mathbf{W}\otimes\mathbf{V}))\mathbf{G}_{(1)}^\top]_{jj},\end{aligned} \tag{43}$$

Therefore, the inequality $\widetilde{\mathcal{H}}(\mathbf{U}_{ij},\mathbf{U}_{ij}{}^t) \geq \widehat{\mathcal{J}}_{ij}(\mathbf{U}_{ij})$ holds. □

Analogously, the convergence under the update rule in Equation (24) can be proven. Next, we provide a detailed proof of convergence for the **V** update rule using an auxiliary function. Consider the following function:

$$\mathcal{J}(\mathcal{V}) = \widehat{\mathcal{J}} + \gamma_V\left\|\mathbf{V}\right\|_1^1, \tag{44}$$

where

$$\widehat{\mathcal{J}}(\mathbf{V}) = \left\|\mathbf{X}_{(2)} - \mathbf{V}\mathbf{G}_{(2)}(\mathbf{W}\otimes\mathbf{U})^\top\right\|_F^2 + \alpha_V\left\|\mathbf{U}\right\|_F^2 + \lambda_V Tr(\mathbf{V}^\top\mathbf{L}_\mathbf{V}\mathbf{V}), \tag{45}$$

We derive

$$\begin{aligned}\widehat{\mathcal{J}}'_{ij}(\mathbf{V}_{ij}{}^{(t)}) =& \frac{\partial\widehat{\mathcal{J}}_{ij}(\mathbf{V}_{ij}{}^{(t)})}{\partial(\mathbf{V}_{ij}{}^{(t)})} = [-2\mathbf{X}_{(2)}^{(t)}(\mathbf{W}^{(t)}\otimes\mathbf{U}^{(t)})\mathbf{G}_{(2)}^{(t)}{}^\top + \\ & 2\mathbf{V}^{(t)}\mathbf{G}_{(2)}^{(t)}((\mathbf{W}^{(t)}\otimes\mathbf{U}^{(t)})^\top(\mathbf{W}^{(t)}\otimes\mathbf{U}^{(t)}))\mathbf{G}_{(2)}^{(t)}{}^\top + \\ & 2\alpha_V\mathbf{V}^{(t)} + 2\lambda_V\mathbf{D}_\mathbf{V}^{(t)}\mathbf{V}^{(t)} - 2\lambda_V\mathbf{C}_\mathbf{V}^{(t)}\mathbf{V}^{(t)}]_{ij},\end{aligned} \tag{46}$$

$$\begin{aligned}\widehat{\mathcal{J}}''_{ij}(\mathbf{V}_{ij}{}^{(t)}) =& [2\mathbf{G}_{(2)}^{(t)}(\mathbf{W}^{(t)}\otimes\mathbf{U}^{(t)})^T(\mathbf{W}^{(t)}\otimes\mathbf{U}^{(t)})\mathbf{G}_{(2)}^{(t)}{}^T + \\ & 2\alpha_V + 2\lambda_V\mathbf{D}_\mathbf{V}^{(t)} - 2\lambda_V\mathbf{C}_\mathbf{V}^{(t)}]_{jj}.\end{aligned} \tag{47}$$

Essentially, the updating rule is element wise, so it is sufficient to prove each $\widehat{\mathcal{J}}_{ij}(\mathbf{V}_{ij})$ is non-increasing under the update rule.

Lemma 3. *The function*

$$\widehat{\mathcal{H}}(\mathbf{V}_{ij},\mathbf{V}_{ij}{}^t) = \widehat{\mathcal{J}}_{ij}(\mathbf{V}_{ij}{}^t) + \widehat{\mathcal{J}}'_{ij}(\mathbf{V}_{ij}{}^t)(\mathbf{V}_{ij} - \mathbf{V}_{ij}{}^t) + \frac{M_1+M_2}{\mathbf{V}_{ij}{}^t}(\mathbf{V}_{ij} - \mathbf{V}_{ij}{}^t)^2 \tag{48}$$

is an auxiliary function for $\widehat{\mathcal{J}}_{ij}(\mathbf{V}_{ij})$, where the matrix $\mathbf{V}^t = (\mathbf{V}_{ij}{}^t)$, and

$$M_1 = \mathbf{V}^{(t)}\mathbf{G}_{(2)}^{(t)}(\mathbf{W}^{(t)}\otimes\mathbf{U}^{(t)})^\top(\mathbf{W}^{(t)}\otimes\mathbf{U}^{(t)})\mathbf{G}_{(2)}^{(t)}{}^\top,$$

$$M_2 = \alpha_V\mathbf{V}^{(t)} + \lambda_V\mathbf{D}_\mathbf{V}^{(t)}\mathbf{V}^{(t)}.$$

Proof. Obviously, $\widetilde{\mathcal{H}}(\mathbf{V}_{ij}{}^t,\mathbf{V}_{ij}{}^t) = \widehat{\mathcal{J}}_{ij}(\mathbf{V}_{ij}{}^t)$. According to Definition 1, we only need to show that $\widetilde{\mathcal{H}}(\mathbf{V}_{ij},\mathbf{V}_{ij}{}^t) \geq \widehat{\mathcal{J}}_{ij}(\mathbf{V}_{ij})$. We first obtain the Taylor series expansion of $\widehat{\mathcal{J}}_{ij}(\mathbf{V}_{ij})$ at $\mathbf{V}_{ij}{}^t$ to be

$$\begin{aligned}&\widetilde{\mathcal{J}}_{ij}(\mathbf{V}_{ij}) = \mathcal{J}_{ij}(\mathbf{V}_{ij}{}^t) + \widehat{\mathcal{J}}'_{ij}(\mathbf{V}_{ij}{}^t)(\mathbf{V}_{ij} - \mathbf{V}_{ij}{}^t) + \frac{1}{2}\mathcal{J}''_{ij}(\mathbf{V}_{ij}{}^t)(\mathbf{V}_{ij} - \mathbf{V}_{ij}{}^t)^2 \\ &= \mathcal{J}_{ij}(\mathbf{V}_{ij}{}^t) + \mathcal{J}'_{ij}(\mathbf{V}_{ij}{}^t)(\mathbf{V}_{ij} - \mathbf{V}_{ij}{}^t) + [\mathbf{G}_{(2)}^{(t)}((\mathbf{W}^{(t)}\otimes\mathbf{U}^{(t)})^\top(\mathbf{W}^{(t)}\otimes\mathbf{U}^{(t)}))\mathbf{G}_{(2)}^{(t)}{}^\top \\ &+ \alpha_V + \lambda_V\mathbf{D}_\mathbf{V}^{(t)} - \lambda_V\mathbf{C}_\mathbf{V}^{(t)}]_{jj}(\mathbf{V}_{ij} - \mathbf{V}_{ij}{}^t)^2.\end{aligned} \tag{49}$$

By Equation (48) and Equation (49), $\mathcal{H}(\mathbf{V}_{ij}, {\mathbf{V}_{ij}}^t) \geq \mathcal{J}_{ij}(\mathbf{V}_{ij})$ is equivalent to

$$\frac{[M_1 + M_2]_{ij}}{{\mathbf{V}_{ij}}^t} \geq [\mathbf{G}_{(2)}^{(t)}(\mathbf{W}^{(t)} \otimes \mathbf{U}^{(t)})^\top(\mathbf{W}^{(t)} \otimes \mathbf{U}^{(t)}){\mathbf{G}_{(2)}^{(t)}}^\top + \alpha_V + \lambda_V \mathbf{D}_{\mathbf{V}}^{(t)} - \lambda_V \mathbf{C}_{\mathbf{V}}^{(t)}]_{jj}. \quad (50)$$

We can obtain

$$\begin{aligned} &[\mathbf{V}^{(t)}\mathbf{G}_{(2)}^{(t)}((\mathbf{W}^{(t)} \otimes \mathbf{U}^{(t)})^\top(\mathbf{W}^{(t)} \otimes \mathbf{U}^{(t)}){\mathbf{G}_{(2)}^{(t)}}^\top]_{ij} \\ &= {\mathbf{V}_{il}}^t[\mathbf{G}_{(2)}^{(t)}((\mathbf{W}^{(t)} \otimes \mathbf{U}^{(t)})^\top(\mathbf{W}^{(t)} \otimes \mathbf{U}^{(t)}){\mathbf{G}_{(2)}^{(t)}}^\top]_{lj} \\ &\geq {\mathbf{V}_{ij}}^t[\mathbf{G}_{(2)}^{(t)}((\mathbf{W}^{(t)} \otimes \mathbf{U}^{(t)})^\top(\mathbf{W}^{(t)} \otimes \mathbf{U}^{(t)}){\mathbf{G}_{(2)}^{(t)}}^\top]_{jj}, \end{aligned} \quad (51)$$

and

$$[\lambda_V \mathbf{D}_{\mathbf{V}}^{(t)}\mathbf{V}^{(t)}]_{ij} = {\mathbf{V}_{il}}^t[\lambda_V \mathbf{D}_{\mathbf{V}}^{(t)}]_{lj} \geq {\mathbf{V}_{ij}}^t[\lambda_V \mathbf{D}_{\mathbf{V}}^{(t)}]_{jj} \geq {\mathbf{V}_{ij}}^t[\lambda_V(\mathbf{D}_{\mathbf{V}}^{(t)} - \mathbf{V}_{\mathbf{V}}^{(t)})]_{jj}, \quad (52)$$

Therefore, the inequality $G(\mathbf{V}_{ij}, {\mathbf{V}_{ij}}^t) \geq F_{ij}(\mathbf{V}_{ij})$ holds. □

Thus, Equation (50) holds, and $\mathcal{H}(\mathbf{V}_{ij}, {\mathbf{V}_{ij}}^t) \geq \mathcal{J}_{ij}(\mathbf{V}_{ij})$. According to Equation (23),

$$\begin{aligned} \mathbf{S}_{ij}^{(t)} =& {\mathbf{V}_{ij}}^{(t)} \frac{[{\mathbf{X}_{(2)}}^{(t)}(\mathbf{W}^{(t)} \otimes \mathbf{U}^{(t)}){\mathbf{G}_{(2)}^{(t)}}^\top + \lambda_V \mathbf{C}_{\mathbf{V}}^{(t)}\mathbf{V}^{(t)}]_{ij}}{[\mathbf{V}^{(t)}\mathbf{G}_{(2)}^{(t)}(\mathbf{W}^{(t)} \otimes \mathbf{U}^{(t)})^\top(\mathbf{W}^{(t)} \otimes \mathbf{U}^{(t)}){\mathbf{G}_{(2)}^{(t)}}^\top + \alpha_V \mathbf{V}^{(t)} + \lambda_V \mathbf{D}_{\mathbf{V}}^{(t)}\mathbf{V}^{(t)}]_{ij}} \\ =& {\mathbf{V}_{ij}}^{(t)} - {\mathbf{V}_{ij}}^{(t)} \frac{\widehat{\mathcal{J}}'_{ij}({\mathbf{V}_{ij}}^{(t)})}{[\mathbf{V}^{(t)}\mathbf{G}_{(2)}^{(t)}(\mathbf{W}^{(t)} \otimes \mathbf{U}^{(t)})^\top(\mathbf{W}^{(t)} \otimes \mathbf{U}^{(t)}){\mathbf{G}_{(2)}^{(t)}}^\top + \alpha_V \mathbf{V}^{(t)} + \lambda_V \mathbf{D}_{\mathbf{V}}^{(t)}\mathbf{V}^{(t)}]_{ij}}. \end{aligned} \quad (53)$$

The soft thresholding operation ensures convergence as follows: if $\mathbf{S}_{ij}^{(t)} > \frac{\gamma_V}{L}$, the update subtracts a constant value; otherwise, if $\mathbf{S}_{ij}^{(t)} \leq \frac{\gamma_V}{L}$, the value is set to zero. Both cases are consistent with the convergence proof. Because Equation (51) is an auxiliary function for $\widehat{\mathcal{J}}_{ij}$, and according to the soft threshold, $\mathcal{J}_{ij}$ is non-increasing under the updating rule stated in Equation (24). Analogously, we can prove the convergence under the updating rule in Equation (35).

4. Experiments

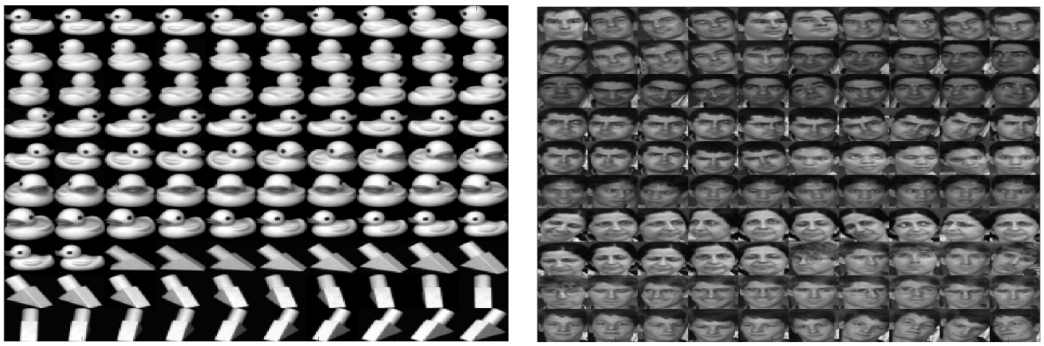
In this section, the experimental setup and analysis of the results are discussed in detail. The descriptions of these datasets are given as follows. First, we introduce the image datasets *Coil20* (http://www.cad.zju.edu.cn/home/dengcai/Data/MLData.html, (accessed on 15 May 2024)), Georgia (https://www.cnblogs.com/kuangqiu/p/7776829.html, (accessed on 10 March 2025)), Iris (http://archive.ics.uci.edu/dataset/53/iris, (accessed 20 November 2024)) [30]. The datasets are summarized in Table 1.

Table 1. Summary of the datasets.

Idx	Datasets	Samples	Features	Classes
D01	Georgia	750	1024	50
D02	Coil20	1440	1024	20
D03	Iris	150	4	3

The details of the public datasets used in the experiments are presented in Figures 1 and 2.

In order to evaluate the effectiveness of our proposed GDSNTD scheme, we compared it with six classical or state-of-the-art clustering and co-clustering methods. All the simulations were performed on a computer with a 2.30-GHz Intel Core i7-11800H CPU and 32 GB memory of 64-bit MATLAB 2016a in Windows 10. Without special specifications, the maximum number of iterations is set to 1000.



(a) (b)

Figure 1. The details of public datasets of *Coil20* (**a**) and *Georgia* (**b**).

Figure 2. Different kinds of *Iris* datasets.

- Non-negative matrix factorization (NMF) [1]: NMF aims to decompose a matrix into two low-dimensional matrices and is now often used as a data processing method in machine learning.

$$F_{NMF} = \min_{\mathbf{U}\geqslant 0,\mathbf{V}\geqslant 0} \left\| \mathbf{X} - \mathbf{U}\mathbf{V}^{\top} \right\|.$$

- Non-negative Tucker decomposition (NTD) [7]: The NTD algorithm is considered as a generalization of NMF.

$$F_{NTD} = \min_{\mathcal{G}\geqslant 0,\mathbf{U},\mathbf{V},\mathbf{W}\geqslant 0} \left\| \mathcal{X} - \mathcal{G} \times_1 \mathbf{U} \times_2 \mathbf{V} \times_3 \mathbf{W} \right\|_F^2.$$

- Graph-regularized NTD (GNTD) [17]:

$$F_{GNTD} = \min_{\mathcal{G}\geqslant 0,\mathbf{U},\mathbf{V},\mathbf{W}\geqslant 0} \left\| \mathcal{X} - \mathcal{G} \times_1 \mathbf{U} \times_2 \mathbf{V} \times_3 \mathbf{W} \right\|_F^2 + \lambda Tr(\mathbf{W}^{\top} L \mathbf{W}).$$

- Graph dual regularized NMF (GDNMF) [12]: GDNMF simultaneously considers the geometric structures of both the data manifold and the feature manifold.

$$F_{GDNMF} = \min_{\mathbf{U}\geqslant 0,\mathbf{V}\geqslant 0} \left\| \mathbf{X} - \mathbf{U}\mathbf{V}^{\top} \right\|_F^2 + \lambda_1 Tr(\mathbf{V}^{\top}\mathbf{L}_{\mathbf{V}}\mathbf{V}) + \lambda_2 Tr(\mathbf{U}^{\top}\mathbf{L}_{\mathbf{U}}\mathbf{U}).$$

- Graph dual regularized non-negative matrix tri-factorization (GDNMTF) [12]: DNMTF is an extension of our DNMF algorithm and simultaneously incorporates

two graph regularizers of both data manifold and feature manifold into its objective function.

$$F_{GDNMTF} = \min_{\mathbf{U}\geqslant 0,\mathbf{S}\geqslant 0,\mathbf{V}\geqslant 0} \left\|\mathbf{X}-\mathbf{U}\mathbf{S}\mathbf{V}^{\top}\right\|_F^2 + \lambda_1 Tr(\mathbf{V}^{\top}\mathbf{L}_{\mathbf{V}}\mathbf{V}) + \lambda_2 Tr(\mathbf{U}^{\top}\mathbf{L}_{\mathbf{U}}\mathbf{U}).$$

- Graph-regularized sparse non-negative matrix trifactorization (GSNMTF) [21]: The GSNMTF model introduces graph regularization and an l_1 norm constraint into the objective function.

$$F_{GSNMTF} = \min_{\mathbf{U}\geqslant 0,\mathbf{S}\geqslant 0,\mathbf{V}\geqslant 0} \left\|\mathbf{X}-\mathbf{U}\mathbf{S}\mathbf{V}^{\top}\right\|_F^2 + \lambda_U Tr(\mathbf{U}^{\top}\mathbf{L}_{\mathbf{U}}\mathbf{U}) + \lambda_V Tr(\mathbf{V}^{\top}\mathbf{L}_{\mathbf{V}}\mathbf{V}) + \gamma_U \left\|\mathbf{U}\right\|_1^1 + \gamma_V \left\|\mathbf{V}\right\|_1^1.$$

4.1. Evaluation Measures

In this section, two widely used metrics, accuracy (AC) and Normalized Mutual Information (NMI), are used to evaluate the clustering performance. Accuracy aims to find a one-to-one relationship between classes and clusters. It is usually used to calculate the proportion of correct samples to the total number of samples. It is a relatively intuitive evaluation index. Accuracy is a clustering evaluation metric that measures the proportion of correctly assigned samples against the total, after a one-to-one mapping between clusters and true classes is established. It is an intuitive and widely used measure [31]. NMI is commonly used in clustering to measure the similarity between two clusterings.

AC is defined as

$$AC = \frac{1}{N}\sum_{i=1}^{N}\delta(gt_i, map(c_i)),$$

where N is the total number of samples, gt_i is the ground-truth label of sample i, c_i is the cluster label assigned by the algorithm, $\delta(\cdot,\cdot)$ is the Dirac delta function, and $map(\cdot)$ is the optimal mapping function [32].

NMI is defined as

$$NMI(B,T) = \frac{2\cdot(I(B;T))}{H(B)+H(T)},$$

where $I(B;T)$ is the mutual information between B and T, and $H(B)$ and $H(T)$ are their entropies. A higher NMI indicates a better alignment between the clustering result and the true labels [33].

4.2. Experimental Setup and Clustering Results Analysis

In this part, the experimental setup is described, and the experimental results are discussed in detail. To ensure the fairness between models, our proposed algorithm and all comparison algorithms use the same random initialization matrix. Subsequently, each experiment is conducted ten times independently on original data, then K-means clustering is performed five times independently on this low-dimensional reduced data. The average and variance of the results are recorded. The standard deviation is set to 0 if it is less than 1×10^{-5}.

Tables 2 and 3 summarize the accuracy and NMI results for each algorithm across all datasets. Table 4 lists the accuracy and standard deviation for each algorithm on the Georgia dataset, while Table 5 presents the corresponding NMI results. Similarly, results for the COIL20 dataset are detailed in Tables 6 (accuracy) and 7 (NMI). From the results, we derive the following main conclusions:

1. From Table 2, we can observe that the value of AC of GDSNTD is better than that of other methods on most datasets. The value of NMI reflects the proportion of correct decisions, which also proves that the proposed model performs better. The performance improvement is evident in the *Georgia* dataset, where GDSNTD improves the accuracy by 48.92% over the NMF co-clustering algorithm. The accuracy of GDSNTD is also improved by 26.1%, 5.15%, 7.02%, 3.71%, and 2.82% compared with other co-clustering algorithms NTD, GNTD, GDNMF, GDNMTF, and GSNMTF, respectively. In the *Coil20* dataset, the accuracy of GDSNTD is also improved by 30.50%, 38.58%, 2.29%, 0.81%, 1.14%, and 0.57%, compared with other co-clustering algorithms NTD, GNTD, GDNMF, GDNMTF, and GSNMTF, respectively. In the *Iris* dataset, the accuracy of GDSNTD is improved by 17.22%, 32.77%, 2.84%, 2.10%, 0.17%, and 0.12% compared with other co-clustering algorithms NMF, NTD, GNTD, GDNMF, GDNMTF, and GSNMTF, respectively.
2. From Table 3, we can observe that our proposed method of GDSNTD can also achieve higher accuracy. The NMI of GDSNTD is as high as 50.16% on the *Iris* dataset, which is 34.95%, 7.46%, 4.27%, 3.49%, and 3.42% better compared with NMF, GNTD, GDNMF, GDNMTF, and GSNMTF, respectively. In the *Georgia* dataset, the accuracy of GDSNTD is also improved by 26.67%, 15.01%, 3.12%, 4.79%, 4.54%, and 4.26% compared with other co-clustering algorithms NMF, NTD, GNTD, GDNMF, GDNMTF, and GSNMTF, respectively. In the *Coil20* dataset, the accuracy of GDSNTD is improved by 21.02%, 23.85%, 2.75%, 0.3%, 0.2%, and 0.32%, compared with other co-clustering algorithms NMF, NTD, GNTD, GDNMF, GDNMTF, and GSNMTF, respectively.
3. Table 4 shows that the average AC of GDSNTD is higher than that of the other methods on most datasets. In the *Georgia* dataset, the accuracy of the co-clustering algorithm GDSNTD is improved by an average of 27.4%, 22.6%, 2.2%, 8.7%, 5.11%, and 3.79% compared with NMF, NTD, GNTD, GDNMF, GDNMTF, and GSNMTF, respectively. From Table 6, it can be observed that on the *Coil20* dataset, the accuracy of GDSNTD is, on average, 23.9%, 30.9%, 1.49%, 4.0%, 5.79%, and 3.18% higher than that of NMF, NTD, GNTD, GDNMF, GDNMTF, and GSNMTF, respectively.
4. From Table 5, it can be observed that on the *Georgia* dataset, the accuracy of GDSNTD is, on average, 23.7%, 19.4%, 1.82%, 7.3%, 5.67%, and 4.63% higher than that of NMF, NTD, GNTD, GDNMF, GDNMTF, and GSNMTF, respectively. From Table 7, it can be observed that on the *Coil20* dataset, the accuracy of GDSNTD is, on average, 20.8%, 31.5%, 1.42%, 3.2%, 6.14%, and 2.76% higher than that of NMF, NTD, GNTD, GDNMF, GDNMTF, and GSNMTF, respectively.

Figure 3 shows the clustering performance on the Georgia, COIL20, and Iris datasets. It can be observed that the proposed GDSNTD method surpasses all other compared methods.

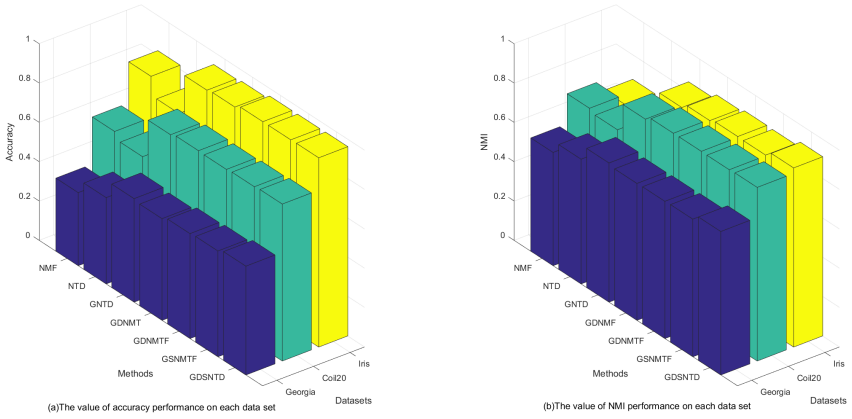


Figure 3. The clustering performance on each dataset.

Table 2. AC (%) of each algorithm on each dataset.

Data	Classes	NMF	NTD	GNTD	GDNMF	GDNMTF	GSNMTF	GDSNTD
D01	50	37.18 ± 2.17	43.91 ± 2.10	52.66 ± 2.98	51.74 ± 2.39	53.39 ± 1.86	53.85 ± 1.79	**55.37 ± 2.39**
D02	20	61.19 ± 4.03	57.62 ± 4.15	78.06 ± 3.63	79.21 ± 3.62	78.95 ± 2.59	79.40 ± 3.05	**79.85 ± 2.79**
D03	3	82.13 ± 9.75	72.51 ± 8.81	93.61 ± 8.90	94.29 ± 1.10	96.11 ± 6.38	96.15 ± 3.05	**96.27 ± 9.01**

Table 3. NMI (%) of each algorithm on each dataset.

Data	Classes	NMF	NTD	GNTD	GDNMF	GDNMTF	GSNMTF	GDSNTD
D01	50	57.63 ± 1.35	63.47 ± 1.37	70.79 ± 1.95	69.66 ± 1.17	69.83 ± 1.09	70.02 ± 0.93	**73.00 ± 1.15**
D01	20	72.99 ± 1.80	71.32 ± 2.22	85.97 ± 1.70	88.07 ± 1.46	88.15 ± 1.06	88.05 ± 1.14	**88.33 ± 1.07**
D03	3	67.53 ± 5.78	60.69 ± 3.10	84.80 ± 6.12	87.40 ± 7.79	88.56 ± 4.59	88.37 ± 5.61	**91.13 ± 7.14**

Table 4. AC (%) of each algorithm on Georgia dataset.

Classes	NMF	NTD	GNTD	GDNMF	GDNMTF	GSNMTF	GDSNTD
3	80.44 ± 11.30	67.56 ± 11.84	87.24 ± 14.38	79.20 ± 12.61	81.29 ± 11.86	79.51 ± 12.52	**91.82 ± 9.18**
7	67.05 ± 8.76	58.51 ± 10.03	66.36 ± 11.23	64.63 ± 7.97	67.75 ± 9.12	**69.22 ± 8.00**	67.03 ± 12.54
11	55.5 ± 7.01	61.73 ± 7.87	70.29 ± 6.97	60.57 ± 6.70	61.26 ± 6.01	66.41 ± 6.58	**70.64 ± 8.10**
15	54.35 ± 7.02	53.5 ± 3.52	63.31 ± 5.65	59.37 ± 6.36	59.50 ± 6.61	58.45 ± 5.04	**64.99 ± 5.49**
19	49.36 ± 3.82	52.32 ± 6.86	60.45 ± 5.13	59.69 ± 5.30	58.81 ± 5.26	60.64 ± 4.40	**61.95 ± 4.77**
23	46.94 ± 4.51	49.01 ± 4.24	**60.99 ± 4.70**	55.91 ± 5.88	54.11 ± 3.79	55.43 ± 4.15	59.16 ± 5.65
27	44.78 ± 3.76	49.64 ± 2.52	58.21 ± 4.38	53.29 ± 4.02	57.12 ± 3.34	59.71 ± 3.01	**60.25 ± 4.51**
31	42.69 ± 3.85	46.65 ± 3.15	57.18 ± 4.11	55.31 ± 3.93	56.64 ± 3.10	56.68 ± 3.00	**58.16 ± 3.82**
35	41.64 ± 2.70	47.14 ± 3.60	55.05 ± 3.72	51.58 ± 4.71	56.45 ± 3.71	56.94 ± 3.70	**57.31 ± 3.07**
39	41.6 ± 2.68	44.74 ± 2.60	53.66 ± 3.22	50.90 ± 2.99	54.81 ± 2.98	54.57 ± 3.29	**55.59 ± 3.07**
43	39.88 ± 2.25	43.62 ± 3.17	53.35 ± 3.02	51.50 ± 2.70	54.13 ± 2.40	54.19 ± 2.42	**54.20 ± 2.66**
47	38.97 ± 2.42	43.81 ± 2.48	53.56 ± 3.22	50.51 ± 2.38	52.89 ± 1.89	53.17 ± 2.18	**54.11 ± 2.30**
49	37.53 ± 1.80	43.46 ± 2.22	52.60 ± 2.26	50.10 ± 2.85	**53.39 ± 2.34**	53.28 ± 2.43	52.88 ± 2.85
50	37.18 ± 2.17	43.91 ± 2.10	52.66 ± 2.98	51.74 ± 2.39	53.39 ± 1.86	53.85 ± 1.79	**55.37 ± 2.39**
Avg.	48.42	50.32	60.35	56.74	58.68	59.43	**61.68**

Table 5. NMI (%) of each algorithm on Georgia dataset.

Classes	NMF	NTD	GNTD	GDNMF	GDNMTF	GSNMTF	GDSNTD
3	57.07 ± 20.27	49.33 ± 16.31	78.08 ± 21.42	65.85 ± 18.07	65.13 ± 15.61	65.00 ± 12.49	**82.78 ± 15.77**
7	60.4 ± 8.54	54.68 ± 10.73	65.48 ± 11.17	63.46 ± 7.16	66.87 ± 10.51	**67.55 ± 10.18**	67.1 ± 12.48
11	60.44 ± 8.04	65.62 ± 6.68	75.79 ± 5.91	65.82 ± 4.88	66.83 ± 4.85	70.14 ± 5.24	**75.93 ± 6.89**
15	60.79 ± 6.47	59.30 ± 2.84	71.45 ± 4.39	68.13 ± 5.05	67.71 ± 5.23	67.87 ± 4.26	**73.05 ± 4.64**
19	59.70 ± 2.31	62.60 ± 5.68	69.98 ± 4.13	70.93 ± 4.49	68.83 ± 3.81	71.29 ± 2.97	**71.64 ± 3.66**
23	58.57 ± 3.36	60.9 ± 3.87	**72.16 ± 2.86**	68.54 ± 4.36	66.93 ± 3.25	67.61 ± 2.85	71.65 ± 4.47
27	58.57 ± 2.85	62.99 ± 2.01	71.60 ± 3.47	66.23 ± 2.93	70.04 ± 1.95	71.88 ± 1.94	**73.61 ± 3.56**
31	58.04 ± 3.56	61.92 ± 2.25	71.02 ± 2.42	69.94 ± 2.43	70.50 ± 2.06	70.55 ± 1.93	**72.31 ± 2.90**
35	58.31 ± 2.31	62.89 ± 2.95	70.63 ± 2.35	67.54 ± 3.01	70.86 ± 2.12	70.93 ± 2.45	**72.14 ± 2.05**
39	58.90 ± 1.93	61.8 ± 1.64	70.65 ± 2.05	67.50 ± 2.18	70.81 ± 1.89	70.73 ± 1.94	**71.69 ± 2.06**
43	58.67 ± 1.53	61.74 ± 2.26	70.67 ± 1.57	68.62 ± 1.36	70.15 ± 1.75	70.31 ± 1.47	**71.27 ± 1.53**
47	58.40 ± 1.79	62.81 ± 1.57	71.43 ± 2.19	68.45 ± 1.11	69.45 ± 1.17	69.52 ± 1.32	**71.40 ± 1.52**
49	57.60 ± 1.33	62.78 ± 1.23	70.49 ± 1.03	68.21 ± 1.55	69.80 ± 1.28	69.85 ± 1.31	**70.78 ± 1.53**
50	57.63 ± 1.35	63.47 ± 1.37	70.79 ± 1.95	69.66 ± 1.17	69.83 ± 1.09	70.02 ± 0.93	**73.00 ± 1.15**
Avg.	58.79	60.92	71.44	67.78	68.84	69.52	**72.74**

Table 6. AC (%) of each algorithm on Coil20 dataset.

Classes	NMF	NTD	GNTD	GDNMF	GDNMTF	GSNMTF	GDSNTD
2	97.92 ± 4.21	76.93 ± 14.22	1 ± 0	1 ± 0	86.49 ± 17.62	99.86 ± 0.28	**1 ± 0**
4	79.26 ± 16.73	79.07 ± 15.15	93.65 ± 11.09	85.87 ± 11.42	74.72 ± 15.61	77.62 ± 16.93	**95.40 ± 8.99**
6	67.24 ± 12.00	65.14 ± 12.03	**86.75 ± 9.76**	78.11 ± 13.43	81.63 ± 11.37	82.63 ± 11.34	83.73 ± 15.22
8	72.17 ± 7.71	65.94 ± 6.14	85.94 ± 7.76	86.87 ± 9.70	88.00 ± 8.82	**88.90 ± 9.13**	86.32 ± 6.78
10	67.52 ± 5.09	69.11 ± 4.78	87.62 ± 8.32	86.64 ± 7.76	85.75 ± 8.30	85.93 ± 7.90	**88.74 ± 9.01**
12	66.43 ± 7.92	64.14 ± 5.07	83.05 ± 7.79	78.69 ± 8.22	78.40 ± 4.71	79.42 ± 6.73	**86.39 ± 7.22**
14	64.58 ± 5.15	63.58 ± 5.98	79.30 ± 5.52	81.04 ± 8.16	82.28 ± 3.47	**82.44 ± 4.95**	81.37 ± 5.39
16	63.86 ± 6.14	60.67 ± 4.48	82.45 ± 4.37	78.67 ± 4.89	80.86 ± 6.45	80.96 ± 7.28	**84.10 ± 3.12**
18	61.71 ± 3.93	61.09 ± 4.28	79.12 ± 5.19	78.58 ± 4.31	79.59 ± 4.15	81.90 ± 4.87	**81.02 ± 4.78**
19	62.06 ± 3.79	59.67 ± 3.78	76.58 ± 4.00	76.32 ± 4.67	77.81 ± 3.44	78.09 ± 3.65	**79.37 ± 2.90**
20	61.19 ± 4.03	57.62 ± 4.15	78.06 ± 3.63	79.21 ± 3.62	78.95 ± 2.59	79.40 ± 3.05	**79.85 ± 2.79**
Avg.	69.45	65.72	84.77	82.73	84.17	83.38	**86.03**

Table 7. NMI (%) of each algorithm on Coil20 dataset.

Classes	NMF	NTD	GNTD	GDNMF	GDNMTF	GSNMTF	GDSNTD
2	91.99 ± 16.19	40.04 ± 30.52	93.35 ± 8.58	1 ± 0	66.48 ± 42.53	98.94 ± 2.14	**1 ± 0**
4	75.51 ± 18.55	75.94 ± 14.84	93.35 ± 9.47	81.75 ± 11.19	76.83 ± 16.39	74.81 ± 20.31	**94.47 ± 8.55**
6	65.27 ± 1.34	64.32 ± 11.79	**86.17 ± 9.08**	78.73 ± 12.42	83.13 ± 8.87	83.88 ± 9.88	85.39 ± 11.34
8	76.67 ± 6.90	70.71 ± 5.58	88.91 ± 4.81	89.55 ± 7.08	91.67 ± 6.04	**92.18 ± 5.70**	89.58 ± 4.03
10	74.23 ± 3.05	74.83 ± 4.02	91.71 ± 5.22	91.48 ± 4.60	90.51 ± 5.36	89.91 ± 5.75	**92.22 ± 5.61**
12	73.35 ± 6.11	71.96 ± 2.98	88.55 ± 5.99	86.72 ± 4.57	85.57 ± 3.70	86.17 ± 4.54	**91.18 ± 5.31**
14	74.63 ± 3.11	72.13 ± 5.24	86.35 ± 3.18	88.58 ± 4.94	**90.34 ± 2.41**	89.84 ± 3.06	87.51 ± 3.13
16	73.87 ± 3.70	71.73 ± 2.72	89.20 ± 2.31	87.10 ± 2.59	88.78 ± 4.06	88.38 ± 4.48	**90.99 ± 1.88**
18	72.77 ± 2.61	72.49 ± 2.84	86.78 ± 3.11	86.97 ± 2.13	89.78 ± 2.51	**89.99 ± 2.87**	89.10 ± 2.45
19	73.43 ± 2.32	72.17 ± 2.58	85.55 ± 1.94	86.44 ± 2.03	87.62 ± 1.39	87.6 ± 1.89	**88.02 ± 1.90**
20	72.99 ± 1.80	71.32 ± 2.22	85.97 ± 1.70	88.07 ± 1.46	88.15 ± 1.06	88.05 ± 1.14	**88.33 ± 1.07**
Avg.	74.97	68.88	89.32	87.76	89.60	88.16	**90.59**

4.3. Interpretation of GDSNTD's Superior Performance

Across all datasets, our proposed method consistently outperforms all competing baselines of NMF, NTD, GNTD, GDNMF, GDNMTF, and GSNMTF in both standard clustering and co-clustering tasks. This empirical evidence strongly suggests that the GDSNTD model with dual graph constraints successfully leads to a more structured and discriminative latent space, thereby improving clustering accuracy.

First, applying manifold constraints to the factor matrices is equivalent to preserving the structure in the "essential features" of the data, resulting in greater accuracy. Second, l_1 regularization counterbalances the Frobenius norm, leading to a more stable optimization process and yielding factors within a more reasonable numerical range. Furthermore, l_1 regularization promotes the learning of "parts" that correspond to local structures within the data, thereby promoting a decomposition with enhanced coherence.

4.4. Parameter Selection

Every parameter is searched over a range of values from 0 to 10^5. The choice of parameters has a pronounced effect on experimental performance; therefore, they were selected systematically through a grid search. Figure 4 shows the process of determining the optimal parameters using a grid search on the *Iris* dataset. From Figure 4, we can see that the accuracy is only 0.7772 when $\alpha_V = \alpha_W = 10^0$ and $\gamma_V = \gamma_W = 10^0$. When the coefficient of the sparse regularization term is $\gamma_V = \gamma_W = 10^2$, the accuracy improves to 0.7980. From Figure 5, it is to be observed that the NMI is 0.8615 when $\alpha_V = \alpha_W = 10^{-3}$

and $\gamma_V = \gamma_W = 10^1$, and when the coefficient of the sparse regularization term is $\gamma_V = \gamma_W = 10^{-2}$, the accuracy is only 0.8776.

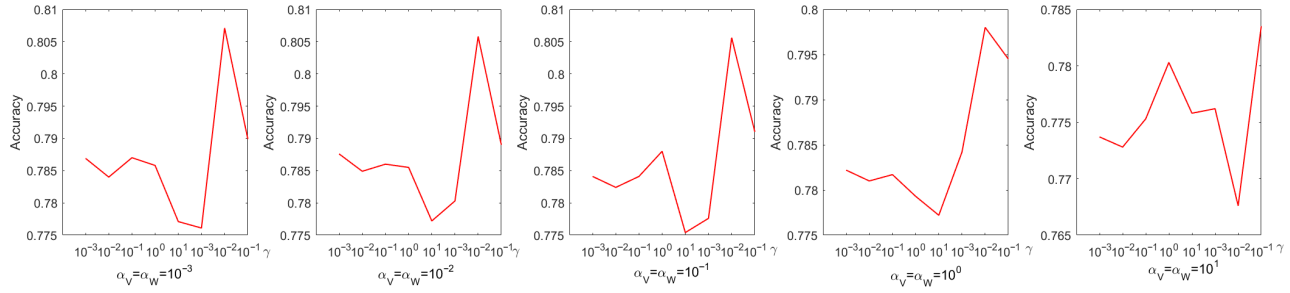


Figure 4. Parameter tuning on dataset *Coil20* regarding accuracy.

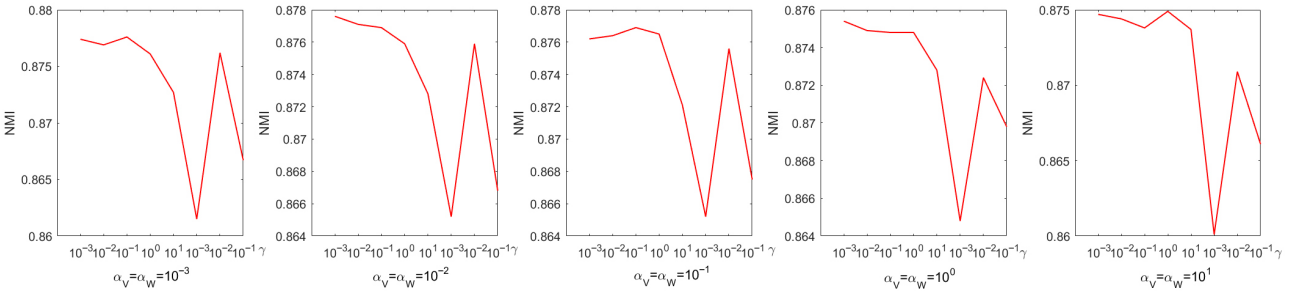


Figure 5. Parameter tuning on dataset *Coil20* regarding NMI.

4.5. Convergence Study

As described in Section 3, the convergence of the proposed algorithms has been theoretically proved. In this subsection, we experimentally analyze the convergence of the proposed algorithms by examining the relationship between the number of iterations and the value of the objective function. This relationship is visualized in Figures 6–8. The convergence behavior of GSNTD on the *Georgia* dataset is illustrated in Figure 6. The convergence behavior of GSNTD on the *Coil20* dataset is illustrated in Figure 7. The convergence behavior of GSNTD on the *Iris* dataset is illustrated in Figure 8. The observed monotonic decrease in the objective function value demonstrates that the algorithm converges effectively under the multiplicative update rules. This result provides empirical support for the convergence proof given in Theorem 1. We can find that GDSNTD is usually able to reach convergence within 1000 iterations.

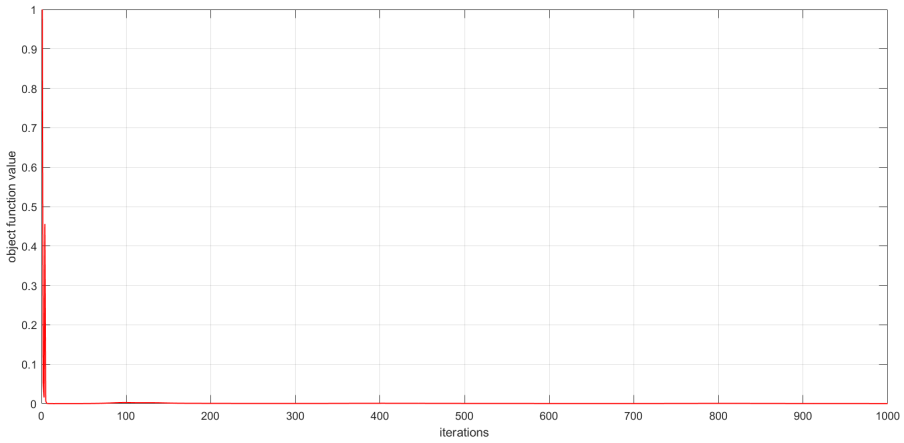


Figure 6. Convergence curves of GDSNTD on *Georgia* dataset.

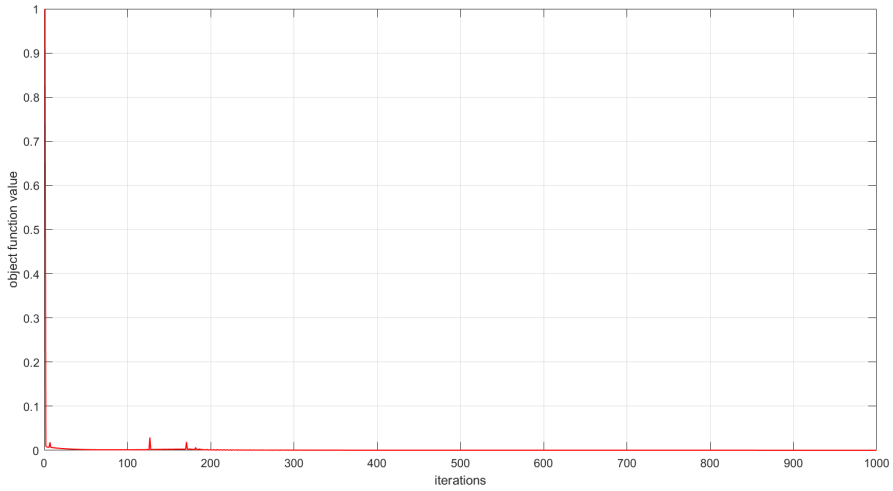


Figure 7. Convergence curves of GDSNTD on *Coil20* dataset.

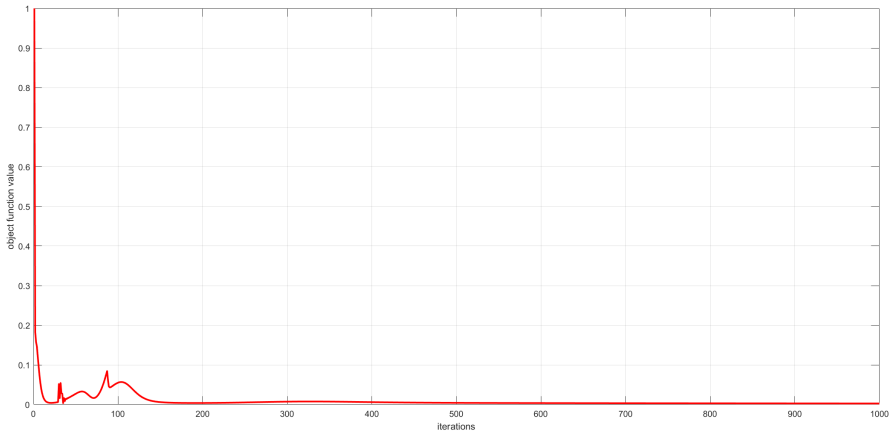


Figure 8. Convergence curves of GDSNTD on *Iris* dataset.

4.6. Complexity Analysis

In this subsection, we analyze the computational complexity of GDSNTD. Note that $\mathcal{X}$ is a third-order $I_1 \times I_2 \times I_3$-dimensional tensor, and $\mathcal{G}$ is a third-order $J_1 \times J_2 \times J_3$-dimensional core tensor. Factor matrices are $\mathbf{U} \in \mathbb{R}_+^{I_1 \times J_1}$, $\mathbf{V} \in \mathbb{R}_+^{I_2 \times J_2}$, and $\mathbf{W} \in \mathbb{R}_+^{I_3 \times J_3}$.

Consider the updating rule in (8), the operation $(\mathbf{W} \otimes \mathbf{V} \otimes \mathbf{U})^\top vec(\mathcal{X})$ corresponds to the tensor mode product $\mathcal{X} \times_1 \mathbf{U}^\top \times_2 \mathbf{V}^\top \times_3 \mathbf{W}^\top$. Its computational complexity is $\mathcal{O}(I_1 I_2 I_3 J_1 + I_2 I_3 J_1 J_2 + I_3 J_1 J_2 J_3)$. $(\mathbf{W} \otimes \mathbf{V} \otimes \mathbf{U})^\top (\mathbf{W} \otimes \mathbf{V} \otimes \mathbf{U}) vec(\mathcal{G})$ corresponds to $vec(\mathcal{G} \times_1 \mathbf{U}^\top \mathbf{U} \times_2 \mathbf{V}^\top \mathbf{V} \times_3 \mathbf{W}^\top \mathbf{W})$. The computational complexity is $J_1 J_2 J_3 (J_1 + J_2 + J_3)$. The total computational complexity of computing the update of $\mathcal{G}$ is bounded by $\mathcal{O}(I_1 I_2 I_3 J_1 + I_2 I_3 J_1 J_2 + I_3 J_1 J_2 J_3 + J_1 J_2 J_3 (J_1 + J_2 + J_3))$.

Consider the updating rule in (13), $\mathbf{X}_{(1)} (\mathbf{W} \otimes \mathbf{V}) \mathbf{G}_{(1)}^\top$ corresponds to $(\mathcal{X} \times_2 \mathbf{V}^\top \times_3 \mathbf{W}^\top)_{(1)} \mathbf{G}_{(1)}^\top$, which needs $\mathcal{O}(I_1 I_2 I_3 J_2 + I_1 I_3 J_2 J_3 + I_1 J_1 J_2 J_3)$ operations. $\mathbf{U} \mathbf{G}_{(1)} ((\mathbf{W} \otimes \mathbf{V})^\top (\mathbf{W} \otimes \mathbf{V})) \mathbf{G}_{(1)}^\top$ corresponds to $(\mathcal{G} \times_1 \mathbf{U} \times_2 \mathbf{V}^\top \mathbf{V} \times_3 \mathbf{W}^\top \mathbf{W})_{(1)} \mathbf{G}_{(1)}^\top$, which needs $\mathcal{O}(J_1^2 J_2 J_3 + J_1 J_2^2 J_3 + J_1 J_2 J_3^2 + I_1 J_1^2)$ operations. The total computational complexity of computing the update of $\mathbf{U}$ is bounded by $\mathcal{O}(I_1 I_2 I_3 J_2 + I_1 I_3 J_2 J_3 + I_1 J_1 J_2 J_3 + J_1^2 J_2 J_3 + J_1 J_2^2 J_3 + J_1 J_2 J_3^2 + I_1 J_1^2)$.

Consider the updating rule in (23), $\mathbf{X}_{(2)}{}^{(t)} (\mathbf{W}^{(t)} \otimes \mathbf{U}^{(t)}) \mathbf{G}_{(2)}^{(t)\top} + \lambda_V \mathbf{C}_\mathbf{V}^{(t)} \mathbf{V}^{(t)}$ corresponds to $(\mathcal{X} \times_1 \mathbf{U}^\top \times_3 \mathbf{W}^\top)_{(2)} \mathbf{G}_{(2)}^\top + \lambda_V \mathbf{C}_\mathbf{V} \mathbf{V}$, and it takes $\mathcal{O}(I_1 I_2 I_3 J_1 + I_2 I_3 J_1 J_3 + I_2 J_1 J_2 J_3 + I_2^2 J_2)$ operations. $\mathbf{V}^{(t)} \mathbf{G}_{(2)}^{(t)} (\mathbf{W}^{(t)} \otimes \mathbf{U}^{(t)})^\top (\mathbf{W}^{(t)} \otimes \mathbf{U}^{(t)}) \mathbf{G}_{(2)}^{(t)\top} + \alpha_V \mathbf{V}^{(t)} + \lambda_V \mathbf{D}_\mathbf{V}^{(t)} \mathbf{V}^{(t)}$ corresponds to $(\mathcal{G} \times_1 \mathbf{U}^\top \mathbf{U} \times_2 \mathbf{V} \times_3 \mathbf{W}^\top \mathbf{W})_{(2)} \mathbf{G}_{(2)}^\top + \alpha_V \mathbf{V} + \lambda_V \mathbf{D}_\mathbf{V} \mathbf{V}$, and it takes

$\mathcal{O}(J_1^2J_2J_3 + J_1J_2^2J_3 + J_1J_2J_3^2 + I_2J_2^2 + I_2^2J_2 + I_2J_2)$ operations. The total computational complexity of computing the update of $\mathbf{V}$ is bounded by $\mathcal{O}(I_1I_2I_3J_1 + I_2I_3J_1J_3 + I_2J_1J_2J_3 + 2I_2^2J_2 + J_1^2J_2J_3 + J_1J_2^2J_3 + J_1J_2J_3^2 + I_2J_2^2 + I_2J_2)$.

Similarly, consider the updating rule in (34), the total computational complexity of computing the update of $\mathbf{W}$ is bounded by $\mathcal{O}(I_1I_2I_3J_1 + I_2I_3J_1J_2 + I_3J_1J_2J_3 + 2I_3^2J_3 + J_1^2J_2J_3 + J_1J_2^2J_3 + J_1J_2J_3^2 + I_3J_3^2 + I_3J_3)$.

Therefore, the total computational complexity of the proposed method is $\mathcal{O}(I_1I_2I_3(3J_1 + J_2) + I_2I_3(2J_2 + J_3) + I_1I_3J_2J_3 + J_1J_2J_3(2I_2 + I_1 + I_2 + 4(J_1 + J_2 + J_3)) + 2I_2^2J_2 + 2I_3^2J_3 + I_1J_1^2 + I_2J_2^2 + I_3J_3^2 + I_2J_2 + I_3J_3)$. Compared with GNTD, the total computational complexity of the GDSNTD is increased by $2I_2^2J_2 + 2I_3^2J_3 + I_2J_2 + I_3J_3$.

Figure 9 shows the relative performance in terms of clustering ACC and time consumption among NTD, GNTD, GDNMTF, GSNMTF, and GDSNTD on the *Georgia* dataset. Figure 10 shows the relative performance in terms of clustering NMI and time consumption among NTD, GNTD, GDNMTF, GSNMTF, and GDSNTD on the *Coil20* dataset. In the figure, CPU(s) is the running time in seconds.

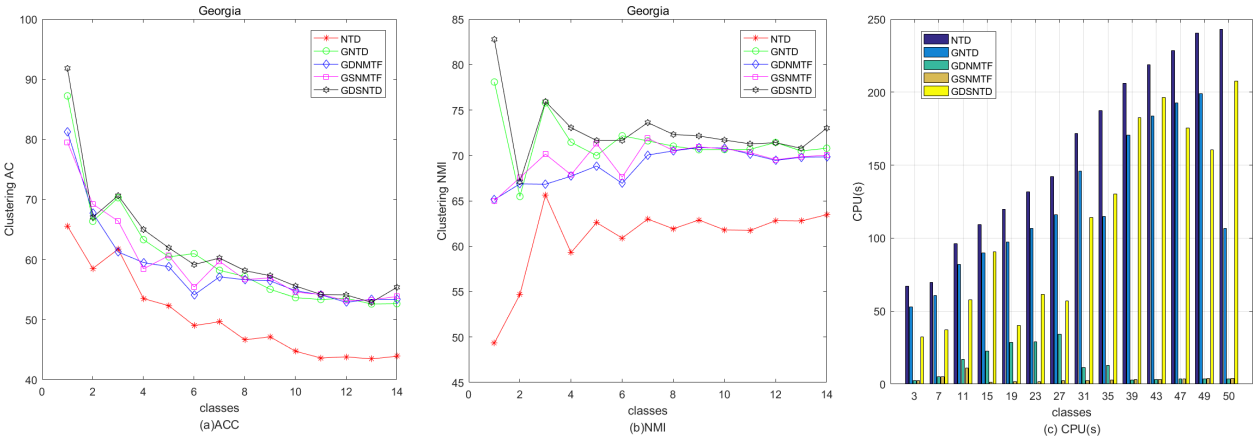


Figure 9. Comparison of clustering performance and running time among NTD, GNTD, GDNMTF, GSNMTF, and GDSNTD on the *Coil20* dataset.

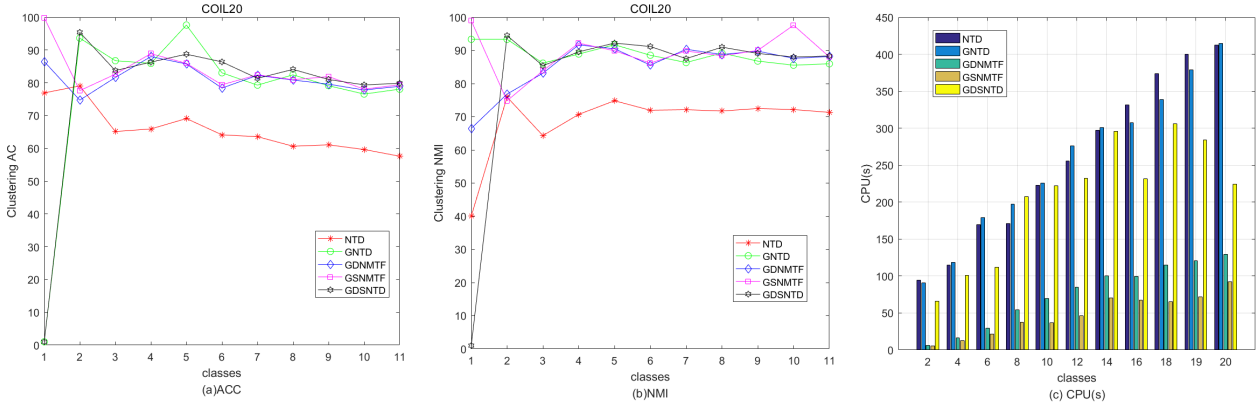


Figure 10. Comparison of clustering performance and running time among NTD, GNTD, GDNMTF, GSNMTF, and GDSNTD on the *Georgia* dataset.

5. Conclusions

In this paper, we propose a co-clustering method via a dual graph-regularized sparse non-negative Tucker decomposition (GDSNTD) framework. It incorporates the Frobenius norm, graph regularization, and an l_1 norm constraint. The graph regularization term is used to maintain the intrinsic geometric structure of the data. This approach allows

the model to represent the internal structure of high-dimensional data more accurately. The model not only adds the l_1 norm to achieve the adjustment of data eigenvalues but also embeds the Frobenius norm to improve the generalization ability of the model. We then present a detailed derivation of the solution, design the accompanying algorithm for GDSNTD, and provide a convergence proof. The experimental results show that selecting appropriate parameters can improve the model's classification accuracy. This indicates that the selected rules have practical significance. Experiments on public datasets demonstrate that our proposed model achieves superior image clustering performance. The proposed method outperforms the state-of-the-art method GSNMTF by an average of 3.79% in clustering accuracy and 4.63% in Normalized Mutual Information on the Georgia dataset.

6. Future Work

Although the proposed GSNTTD method has shown good clustering performance, it operates on the assumption that the data resides in a well-formed latent vector space [34]. In recent years, deep learning-based clustering has experienced rapid advancement. The current popular deep learning-based methods are dominated by several key approaches, such as Graph Autoencoders (GAEs), graph convolutional network-based clustering (GCN-based clustering), and Deep Tensor Factorization.

GAEs aim to learn low-dimensional latent representations (embeddings) of graph-structured data in an unsupervised manner. It learns to compress input data into a lower-dimensional latent representation (encoding) and then reconstruct the original input from this representation (decoding). The critical distinction is that the input data is a graph, characterized by its topology (structure) and node attributes (features). Kipf et al. in [35] formally introduced the GAE and its variational counterpart, the variational graph autoencoder (VGAE). The field of GAE has evolved rapidly from foundational models to highly sophisticated and specialized architectures. Recent advancements have focused on developing more sophisticated architectures to address specific challenges. Zhou K et al. proposed a new causal representation method based on a graph autoencoder embedded autoencoder (GeAE). The GeAE employs a causal structure learning module to account for non-linear causal relationships present in the data [36].

Kipf and Welling in [35] introduced the variational graph autoencoder (VGAE), a framework for unsupervised learning on graph-structured data based on the variational autoencoder (VAE). In this work, they demonstrated this VGAE model using a GCN encoder and a simple inner product decoder. In [37], Kipf and Welling first introduced the efficient, first-order approximation-based graph convolutional layer. This formulation has laid the groundwork for numerous subsequent GCN studies. The core idea of a GCN encoder is to learn low-dimensional vector representations (i.e., node embeddings) for nodes by propagating and transforming information across the graph structure. The node's embedding is determined not only by its own features but also jointly by the features of its neighbor nodes and the local graph structure. The core principle of GCN-based clustering is to unify node representation learning and cluster assignment into an end-to-end, jointly optimized framework and perform these tasks simultaneously.

The principle of Deep Tensor Factorization can be understood as using a deep neural network to perform the factorization and reconstruction process. Wu et al. proposed a Neural Tensor Factorization model, which incorporates a multi-layer perceptron structure to learn the non-linearities between different latent factors [38]. In [39], Jiang et al. proposed a generic architecture of deep transfer tensor factorization (DTTF), where the side information is embedded to provide effective compensation for the tensor sparsity.

In [40], Ballard et al. categorized feedforward neural networks, graph convolutional neural networks, and autoencoders into non-generative deep learning-based multi-omics

integration methods. They found that deep learning-based approaches build off of previous statistical methods to integrate multi-omics data by enabling the modeling of complex and non-linear interactions between data types. Therefore, deep learning-based clustering has emerged as a rapidly advancing field. It integrates conventional cluster analysis with deep learning, leveraging its powerful feature representation and non-linear mapping capabilities to demonstrate superior performance in unsupervised clustering tasks. In the future, we intend to incorporate recent deep learning-based methods into our subsequent research to explore more up-to-date clustering methodologies.

Author Contributions: Conceptualization, J.H. and L.L.; methodology, L.L.; writing—original draft preparation J.H. and L.L. All authors have read and agreed to the published version of the manuscript.

Funding: The authors acknowledge the support from the National Natural Science Foundation of China under Grants (Nos. 12361044, 12161020, 12061025), Basic Research Project of Science and Technology Plan of Guizhou of China under Grants Qian Ke he foundation ZK[2023] General 022.

Data Availability Statement: The original contributions presented in this study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflict of interest. The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

1. Lee, D.; Seung, H.S. Learning the parts of objects by non-negative matrix factorization. *Nature* **1999**, *401*, 788–791. [CrossRef] [PubMed]
2. Friedlander, M.P.; Hatz, K. Computing non-negative tensor factorizations. *Optim. Methods Softw.* **2008**, *23*, 631–647. [CrossRef]
3. Shcherbakova, E.M.; Matveev, S.A.; Smirnov, A.P.; Tyrtyshnikov, E.E. Study of performance of low-rank nonnegative tensor factorization methods. *Russ. J. Numer. Anal. Math. Model.* **2023**, *38*, 231–239. [CrossRef]
4. De Lathauwer, L. A survey of tensor methods. In Proceedings of the 2009 IEEE International Symposium on Circuits and Systems, Taipei, Taiwan, 24–27 May 2009; pp. 2773–2776. [CrossRef]
5. Zhou, G.; Cichocki, A.; Zhao, Q.; Xie, S. Nonnegative matrix and tensor factorizations: An algorithmic perspective. *IEEE Signal Process. Mag.* **2014**, *31*, 54–65. [CrossRef]
6. Kolda, T.G.; Bader, B.W. Tensor decompositions and applications. *SIAM Rev.* **2009**, *51*, 455–500. [CrossRef]
7. Kim, Y.D.; Choi, S. Nonnegative Tucker decomposition. In Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, MN, USA, 17–22 June 2007; IEEE: New York, NY, USA, 2007; pp. 1–8. [CrossRef]
8. Cai, D.; He, X.; Han, J.; Huang, T.S. Graph regularized nonnegative matrix factorization for data representation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *33*, 1548–1560. [CrossRef]
9. Sun, F.; Xu, M.; Hu, X.; Jiang, X. Graph regularized and sparse nonnegative matrix factorization with hard constraints for data representation. *Neurocomputing* **2016**, *173*, 233–244. [CrossRef]
10. Sun, J.; Wang, Z.H.; Sun, F.; Li, H.J. Sparse dual graph-regularized NMF for image co-clustering. *Neurocomputing* **2018**, *316*, 156–165. [CrossRef]
11. Shang, F.; Jiao, L.C.; Wang, J. Graph dual regularization non-negative matrix factorization for co-clustering. *Pattern Recognit.* **2012**, *45*, 2237–2250. [CrossRef]
12. Long, X.; Lu, H.; Peng, Y.; Li, W. Graph regularized discriminative non-negative matrix factorization for face recognition. *Multimed. Tools Appl.* **2014**, *72*, 2679–2699. [CrossRef]
13. Saberi-Movahed, F.; Berahm, K.; Sheikhpour, R.; Li, Y.; Pan, S. Nonnegative matrix factorization in dimensionality reduction: A survey. *arXiv* **2024**, arXiv:2405.03615. [CrossRef]
14. Jing, W.J.; Lu, L.; Ou, W. Semi-supervised non-negative matrix factorization with structure preserving for image clustering. *Neural Netw.* **2025**, *187*, 107340. [CrossRef]
15. Li, X.; Ng, M.K.; Cong, G.; Ye, Y.; Wu, Q. MR-NTD: Manifold Regularization Nonnegative Tucker Decomposition for Tensor Data Dimension Reduction and Representation. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *28*, 1787–1800. [CrossRef]
16. Yin, W.; Ma, Z. LE & LLE Regularized Nonnegative Tucker Decomposition for clustering of high dimensional datasets. *Neurocomputing* **2019**, *364*, 77–94. [CrossRef]

17. Qiu, Y.; Zhou, G.; Zhang, Y.; Xie, S. Graph regularized nonnegative Tucker decomposition for tensor data representation. In Proceedings of the ICASSP 2019—2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 12–17 May 2019; IEEE: New York, NY, USA, 2019; pp. 8613–8617. [CrossRef]
18. Dhillon, I.S.; Mallela, S.; Modha, D.S. Information-theoretic co-clustering. In Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, 24–27 August 2003; pp. 89–98. [CrossRef]
19. Whang, J.J.; Dhillo, I.S. Non-exhaustive, Overlapping co-clustering. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, Singapore, 6–10 November 2017; pp. 2367–2370. [CrossRef]
20. Del Buono, N.; Pio, G. Non-negative matrix tri-factorization for co-clustering: An analysis of the block matrix. *Inf. Sci.* **2015**, *301*, 13–26. [CrossRef]
21. Deng, P.; Li, T.R.; Wang, H.; Wang, D.; Horng, S.J.; Liu, R. Graph Regularized Sparse Non-Negative Matrix Factorization for Clustering. *IEEE Trans. Comput. Soc. Syst.* **2023**, *10*, 910–921. [CrossRef]
22. Chachlakis, D.G.; Dhanaraj, M.; Prater-Bennette, A.; Markopoulos, P.P. Dynamic L_1-norm Tucker tensor decomposition. *IEEE J. Sel. Top. Signal Process.* **2021**, *15*, 587–602. [CrossRef]
23. Peng, X.; Lu, C.Y.; Yi, Z.; Tang, H.J. Connections Between Nuclear-Norm af Frobenius-Norm-Based Representations. *IEEE Trans. Neural Netw. Learn. Syst.* **2016**, *29*, 218–224. [CrossRef]
24. Ahmed, T.; Raja, H.; Bajwa, W.U. Tensor regression using low-rank and sparse Tucker decompositions. *SIAM J. Math. Data Sci.* **2020**, *2*, 944–966. [CrossRef]
25. Chachlakis, D.G.; Prater-Bennette, A.; Markopoulos, P.P. L_1-norm Tucker tensor decomposition. *IEEE Access* **2019**, *7*, 178454–178465. [CrossRef]
26. Rockafellar, R.T. Lagrange multipliers and optimality. *SIAM Rev.* **1993**, *35*, 183–238. [CrossRef]
27. Stephen, B.; Lieven, V. *Convex Optimization*; Cambridge University Press: Cambridge, UK, 2004. [CrossRef]
28. Patrick, L.C.; Valérie, R.W. Signal recovery by proximal forward-backward splitting. *Multiscale Model. Simul.* **2005**, *4*, 1168–1200. [CrossRef]
29. Kim, H.; Park, H. Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. *SIAM J. Matrix Anal. Appl.* **2008**, *30*, 713–730. [CrossRef]
30. Asuncion, A.; Newman, D. *Iris*; UCI Machine Learning Repository: Irvine, CA, USA, 1936. [CrossRef]
31. Cai, D.; He, X.; Han, J. Document clustering using locality preserving indexing. *IEEE Trans. Knowl. Data Eng.* **2005**, *17*, 1624–1637. [CrossRef]
32. Yang, C.; Yi, Z. Document clustering using locality preserving indexing and support vector machines. *Soft Comput.* **2008**, *17*, 677–683. [CrossRef]
33. Estévez, P.A.; Tesmer, M.; Perez, C.A.; Zurada, J.M. Normalized mutual information feature selection. *IEEE Trans. Neural Netw.* **2009**, *20*, 189–201. [CrossRef]
34. Dong, Y.; Deng, Y.; Dong, Y.; Wang, J. A survey of clustering based on deep learning. *Comput. Appl.* **2022**, *42*, 1021–1028. [CrossRef]
35. Kipf, T.N.; Welling, M. Variational Graph Auto-Encoders. Neural Information Processing Systems (NeurIPS) Workshop on Bayesian Deep Learning. *arXiv* **2016**, arXiv:1611.07308. [CrossRef]
36. Zhou, K.; Jiang, M.; Gabrys, B.; Xu, Y. Learning causal representations based on a GAE embedded autoencoder. *IEEE Trans. Knowl. Data Eng.* **2025**, *37*, 3472–3484. [CrossRef]
37. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv* **2016**, arXiv:1609.02907v4. [CrossRef]
38. Wu, X.; Shi, B.; Dong, Y.; Huang, C.; Chawla, N.V. Neural tensor factorization. *arXiv* **2018**, arXiv:1802.04416. [CrossRef]
39. Jiang, P.; Xin, K.; Li, C. Deep Transfer Tensor Factorization for Multi-View Learning. *IEEE Int. Conf. Data Min. Work.* **2022**, 459–466. [CrossRef]
40. Ballard, J.L.; Wang, Z.; Li, W.; Shen, L.; Long, Q. Deep learning-based approaches for multi-omics data integration and analysis. *Biodata Min.* **2024**, *17*, 38. [CrossRef] [PubMed]