symmetry

MDPI

Article

Nonparametric Tensor Completion Based on Gradient Descent and Nonconvex Penalty

Kai Xu [1] and Zhi Xiong [1,2,*]

[1] Department of Computer Science and Technology, Shantou University, 243 Daxue Road, Shantou 515063, China; 17kxu@stu.edu.cn

[2] Key Laboratory of Intelligent Manufacturing Technology (Shantou University), Ministry of Education, Shantou University, 243 Daxue Road, Shantou 515063, China

* Correspondence: zxiong@stu.edu.cn

Received: 7 November 2019; Accepted: 10 December 2019; Published: 12 December 2019

Abstract: Existing tensor completion methods all require some hyperparameters. However, these hyperparameters determine the performance of each method, and it is difficult to tune them. In this paper, we propose a novel nonparametric tensor completion method, which formulates tensor completion as an unconstrained optimization problem and designs an efficient iterative method to solve it. In each iteration, we not only calculate the missing entries by the aid of data correlation, but consider the low-rank of tensor and the convergence speed of iteration. Our iteration is based on the gradient descent method, and approximates the gradient descent direction with tensor matricization and singular value decomposition. Considering the symmetry of every dimension of a tensor, the optimal unfolding direction in each iteration may be different. So we select the optimal unfolding direction by scaled latent nuclear norm in each iteration. Moreover, we design formula for the iteration step-size based on the nonconvex penalty. During the iterative process, we store the tensor in sparsity and adopt the power method to compute the maximum singular value quickly. The experiments of image inpainting and link prediction show that our method is competitive with six state-of-the-art methods.

Keywords: tensor completion; iterative solution; nonparametric; gradient descent; nonconvex penalty

1. Introduction

Real-world data are often sparse but rich in structures and can be stored in arrays. Tensors are K-way arrays that can be used to store multimodal data, image/video data, complex relationship network data, etc. At present, tensors have been successfully applied in many fields, such as image restoration [1], recommendation systems [2], signal processing [3], and high-order web link analysis [4]. Moreover, tensors have also been applied in clustering and classification in some recent studies [5,6]. A comprehensive survey of the applications of tensors can be found in [7]. In these applications, a decisive work is to fill in the missing values of the tensor, namely, tensor completion.

For matrix completion, a common method is to decompose the matrix into two-factor matrices and then use them to calculate the missing data [8,9]. Another method is to turn it into a Rank Minimization (RM) problem. Analogous to matrix completion, the methods of tensor completion can also be divided into two categories: tensor decomposition and RM. Tucker decomposition and CANDECOMP/PARAFAC (CP) [10] are two classic methods of tensor decomposition, and they decompose a high-order tensor into a kernel tensor and some factor vectors. Reference [11] proposed tensor Singular Value Decomposition (t-SVD), but it can only be used for small-scale tensors because the tensor will be expanded to a large-scale matrix. In [12], the t-SVD was applied to an image deblurring problem. Reference [13] proposed a low-order tensor decomposition for tagging recommendation,

which is similar to Tucker decomposition. In the second approach, Nuclear Norm Minimization (NNM) is often used to replace RM because RM is NP-hard [14]. Reference [15] directly minimized the tensor nuclear norm for tensor completion. Reference [16] proposed a dual frame for low-rank tensor completion via nuclear norm constraints. Since high-order tensors represent a higher dimensional space, some works [17,18] used the Riemannian manifold for tensor completion, which is still closely linked to RM. In addition, some works [19,20] converted the tensor into matrices and realized tensor completion by means of matrix completion, but they ignored the inner structure and correlation of the data.

The aforementioned methods of tensor completion all require some hyperparameters, such as the upper bound of the rank in the low-rank constraint and the penalty coefficient for norms. However, the selection of these hyperparameters not only consumes a substantial amount of time but also determines the performance of the methods. To address this issue, we propose a novel Nonparametric Tensor Completion (NTC) method based on gradient descent and nonconvex penalty. We use gradient descent to solve the optimization problem of tensor completion and build a gradient tensor with tensor matricizations and Singular Value Decomposition (SVD). We select the optimal direction based on the scaled latent nuclear norm in each iteration. The step-size in gradient descent is regarded as a penalty parameter for the singular value, and we design a nonconvex penalty for it. Furthermore, during the iterative process, we store the tensor in sparsity and adopt the power method to compute the maximum singular value quickly. Experiments of image inpainting and link prediction show that our method is competitive with some state-of-the-art methods. The main contributions of this paper are:

1. Unlike existing methods, our method has no parameters and is easily manipulated.
2. In each iteration, we use tensor matricization and SVD to approximate the gradient descent direction, so the entries outside the observation range can also be updated.
3. Considering the symmetry of every dimension of a universal tensor, we select the optimal gradient tensor via scaled latent nuclear norm in each iteration.
4. We design the formula of iteration step-size elaborately, which makes our iteration able to achieve a higher convergence speed and a lower error.

The rest of the paper is organized as follows. Section 2 introduces the background knowledge. Our method is proposed in Section 3. Section 4 gives the experimental results and analysis. Finally, the conclusions are given in Section 5.

2. Background Knowledge

2.1. Symbols and Formulas

In this paper, vectors and matrices are denoted by lowercase and uppercase bold italic letters, respectively, and tensors are denoted by bold handwriting. The relevant symbols and formulas involved in this paper are as follows:

- $\|\boldsymbol{A}\|_* = \sum_i \sigma_i$ is the nuclear norm of matrix $\boldsymbol{A}$, where σ_i is the ith singular value of $\boldsymbol{A}$.
- $\mathcal{X} \in R^{I_1 \times I_2 \times \cdots \times I_D}$ represents a D-dimensional tensor, where $I_1, I_2, \ldots, I_D$ is the size of each dimension.
- $\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_1=1}^{I_1} \cdots \sum_{i_D=1}^{I_D} \mathcal{X}_{i_1 \cdots i_D} \mathcal{Y}_{i_1 \cdots i_D}$ is the inner product of two tensors $\mathcal{X}$ and $\mathcal{Y}$ of the same dimension, where $\mathcal{X}_{i_1 \cdots i_D}$ and $\mathcal{Y}_{i_1 \cdots i_D}$ are the elements in $\mathcal{X}$ and $\mathcal{Y}$, respectively.
- $\|\mathcal{X}\|_F = \sqrt{\langle \mathcal{X}, \mathcal{X} \rangle}$ is the Frobenius norm of $\mathcal{X}$.
- $\mathcal{X}_{\langle d \rangle}$ represents the mode-d matricization of $\mathcal{X}$, i.e., $\left(\mathcal{X}_{\langle d \rangle}\right)_{i_d j} = \mathcal{X}_{i_1 i_2 \cdots i_D}$, $j = 1 + \sum_{l=1, l \neq d}^{D} (i_l - 1) \prod_{m=1, m \neq d}^{l-1} I_m$.
- $\boldsymbol{A}^{\langle d \rangle}$ represents the mode-d tensorization of $\boldsymbol{A}$, i.e., $\left(\mathcal{X}_{\langle d \rangle}\right)^{\langle d \rangle} = \mathcal{X}$.

To make tensor matricization and matrix tensorization easy to understand, a 3-dimensional tensor is provided below as an example. Suppose $\mathcal{X}$ is a tensor of size $3 \times 4 \times 2$ as Figure 1 shows.

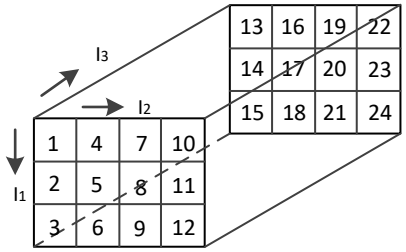

Figure 1. An example of 3-dimensional tensor.

Then the three matricizations of $\mathcal{X}$ are

$$\mathcal{X}_{\langle 1 \rangle} = \begin{bmatrix} 1 & 4 & 7 & 10 & 13 & 16 & 19 & 22 \\ 2 & 5 & 8 & 11 & 14 & 17 & 20 & 23 \\ 3 & 6 & 9 & 12 & 15 & 18 & 21 & 24 \end{bmatrix},$$

$$\mathcal{X}_{\langle 2 \rangle} = \begin{bmatrix} 1 & 2 & 3 & 13 & 14 & 15 \\ 4 & 5 & 6 & 16 & 17 & 18 \\ 7 & 8 & 9 & 19 & 20 & 21 \\ 10 & 11 & 12 & 22 & 23 & 24 \end{bmatrix},$$

$$\mathcal{X}_{\langle 3 \rangle} = \begin{bmatrix} 1 & 2 & 3 & \cdots & 10 & 11 & 12 \\ 13 & 14 & 15 & \cdots & 22 & 23 & 24 \end{bmatrix}.$$

2.2. Related Algorithms

SVD needs to consume a substantial amount of memory and CPU resources. In the real world, data usually have a large scale but are very sparse. For a large sparse matrix, the power method [21] can compute the maximum singular value quickly. Gradient descent is a simple but effective method for solving unconstrained convex optimization problems. It is used to minimize a function by iteratively moving in the direction of the steepest descent as defined by the negative of the gradient. In this paper, we use the gradient descent method to solve the optimization problem of tensor completion and use the power method to improve computational efficiency.

3. Nonparametric Tensor Completion

3.1. Problem Description

The goal of tensor completion is to fill in the missing entries of a partially known tensor. To circumvent this, a usual approach is to find a tensor that is close enough to the original tensor in the positions of the known entries. Suppose $\mathcal{A}$ is a D-order tensor, $\mathcal{A} \in R^{I_1 \times I_2 \times \cdots \times I_D}$; the positions of the observed (namely, known) entries in $\mathcal{A}$ are indicated by Ω. The tensor completion can be formulated by the following optimization problem:

$$\min_{\mathcal{X}} \left\| \mathcal{P}_\Omega(\mathcal{A} - \mathcal{X}) \right\|_F^2, \tag{1}$$

where $[\mathcal{P}_\Omega(\mathcal{X})]_{i_1 i_2 \cdots i_D} = \mathcal{X}_{i_1 i_2 \cdots i_D}$ if $i_1 i_2 \cdots i_D \in \Omega$; otherwise, it is 0. Problem (1) becomes a matrix completion when $D = 2$.

Problem (1) is an unconstrained optimization problem. If $\mathcal{X}$ has identical entries with $\mathcal{A}$ in the range of Ω but has any entries outside the range of Ω, then it makes $\left\| \mathcal{P}_\Omega(\mathcal{X} - \mathcal{A}) \right\|_F^2$ obtain the minimum value 0. However, such $\mathcal{X}$ is meaningless because it ignores the inner structure and correlation of the data. The most common way is to constrain $\mathcal{X}$ with rank or nuclear norm [22,23]. However, it introduces some hyperparameters, e.g., the upper limit for rank or norm. For unsupervised learning, it is difficult to choose the appropriate hyperparameters. To address this issue, we propose an efficient nonparametric iteration method to solve the unconstrained optimization problem (1).

Although we do not add constraints for problem (1), in each iteration, we not only calculate the missing entries by the aid of data correlation, but also consider the low-rank of tensor and the convergence speed of iteration.

3.2. Iterative Calculation Based on Gradient Descent

For simplicity, we convert (1) into the following form:

$$\text{Minimize}: F(\mathcal{X}) = \frac{1}{2}\left\|\mathcal{P}_\Omega(\mathcal{A}-\mathcal{X})\right\|_F^2. \tag{2}$$

$F(\mathcal{X})$ is a continuous differentiable convex function, and its derivative is

$$F\prime(\mathcal{X}) = -\mathcal{P}_\Omega(\mathcal{A}-\mathcal{X}). \tag{3}$$

We use the gradient descent method to solve problem (2), and the iterative formula is as follows:

$$\mathcal{X}^{n+1} = \mathcal{X}^n - \lambda F'(\mathcal{X}^n) = \mathcal{X}^n + \lambda\mathcal{P}_\Omega(\mathcal{A}-\mathcal{X}^n), \tag{4}$$

where n is the number of iterations and λ is the iteration step-size. Note that $\mathcal{P}_\Omega(\mathcal{A}-\mathcal{X}^n)$ only has values in the range of Ω , so (4) cannot update the entries of $\mathcal{X}$ outside the range of Ω . Therefore, we hope to find an approximation of $\mathcal{P}_\Omega(\mathcal{X}^n-\mathcal{A})$ that has values outside the range of Ω so that (4) can update the entries of $\mathcal{X}$ outside the range of Ω.

Based on tensor matricization and SVD, we have

$$\mathcal{P}_\Omega(\mathcal{A}-\mathcal{X}^n) = \left((\mathcal{P}_\Omega(\mathcal{A}-\mathcal{X}^n))_{\langle d\rangle}\right)^{\langle d\rangle} = \left(\sum\nolimits_{i=1}^{r_d^n}\sigma_{d,i}^n\boldsymbol{u}_{d,i}^n{\boldsymbol{v}_{d,i}^n}^T\right)^{\langle d\rangle}, \tag{5}$$

where $d \in \{1,2,..,D\}$; r_d^n is the rank of the matrix $(\mathcal{P}_\Omega(\mathcal{A}-\mathcal{X}^n))_{\langle d\rangle}$; $\sigma_{d,i}^n$, $\boldsymbol{u}_{d,i}^n$ and $\boldsymbol{v}_{d,i}^n$ are the ith singular value, left singular vector, and right singular vector of $(\mathcal{P}_\Omega(\mathcal{A}-\mathcal{X}^n))_{\langle d\rangle}$, respectively; and $\sigma_{d,1}^n > \sigma_{d,2}^n > \ldots > \sigma_{d,r_d^n}^n$. $\mathcal{P}_\Omega(\mathcal{A}-\mathcal{X}^n)$ can be approximated by selecting the first m ($m < r_d^n$) singular values in (5). In theory, the larger the value of m is, the better the approximation of $\mathcal{P}_\Omega(\mathcal{A}-\mathcal{X}^n)$. However, (i) too many singular values will increase the computational complexity, and (ii) small singular values usually represent noise. Therefore, we only use the largest singular value and corresponding left and right singular vectors to approximate $\mathcal{P}_\Omega(\mathcal{A}-\mathcal{X}^n)$, then

$$\mathcal{P}_\Omega(\mathcal{A}-\mathcal{X}^n) \approx (\sigma_{d,1}^n\boldsymbol{u}_{d,1}^n{\boldsymbol{v}_{d,1}^n}^T)^{\langle d\rangle}. \tag{6}$$

We must emphasize that $\left(\sum_{i=1}^{r_d^n}\sigma_{d,i}^n\boldsymbol{u}_{d,i}^n{\boldsymbol{v}_{d,i}^n}^T\right)^{\langle d\rangle}$ has no values outside the range of Ω, but $(\sigma_{d,1}^n\boldsymbol{u}_{d,1}^n{\boldsymbol{v}_{d,1}^n}^T)^{\langle d\rangle}$ has values outside the range of Ω . That is to say, the values in the range of Ω are extended beyond the range via tensor matricization and SVD. Then, (4) becomes

$$\mathcal{X}^{n+1} = \mathcal{X}^n + \lambda\,\sigma_{d,1}^n(\boldsymbol{u}_{d,1}^n{\boldsymbol{v}_{d,1}^n}^T)^{\langle d\rangle}. \tag{7}$$

Below are some further explanations for (7). $F\prime(\mathcal{X}^n) = \mathcal{P}_\Omega(\mathcal{X}^n-\mathcal{A}) = \left(\sum_{i=1}^{r_d^n}\sigma_{d,i}^n\boldsymbol{u}_{d,i}^n{\boldsymbol{v}_{d,i}^n}^T\right)^{\langle d\rangle}$ only has values in the range of Ω ; and $\sigma_{d,1}^n$, $\boldsymbol{u}_{d,1}^n$ and $\boldsymbol{v}_{d,1}^n$ are the largest singular value, leading left singular vector, and leading right singular vector of $(\mathcal{P}_\Omega(\mathcal{A}-\mathcal{X}^n))_{\langle d\rangle}$, respectively. The reason why we use $\sigma_{d,1}^n$, $\boldsymbol{u}_{d,1}^n$, and $\boldsymbol{v}_{d,1}^n$ to approximate $F\prime(\mathcal{X}^n)$ in each iteration includes four aspects: (i) if noise is eliminated, $\left(\sigma_{d,1}^n\boldsymbol{u}_{d,1}^n{\boldsymbol{v}_{d,1}^n}^T\right)^{\langle d\rangle}$ is close enough to $F\prime(\mathcal{X}^n)$ in the range of Ω; (ii) $\left(\sigma_{d,1}^n\boldsymbol{u}_{d,1}^n{\boldsymbol{v}_{d,1}^n}^T\right)^{\langle d\rangle}$ has values outside the range of Ω ; (iii) the computational cost is effectively reduced; and (iv) SVD nicely considers the correlation between data entries, which ensures the rational of tensor completion.

Here, $(\mathcal{P}_\Omega(\mathcal{A}-\mathcal{X}^n))_{\langle d\rangle}$ is the mode-d unfolding of $\mathcal{P}_\Omega(\mathcal{A}-\mathcal{X}^n)$ and is usually a large-scale matrix, so the traditional SVD cannot be used for efficient calculation. We adopt the power method to calculate the maximum singular value and corresponding singular vectors quickly.

3.3. Proof of Iterative Convergence

For the iterative formula (7), we can prove that if $\lambda \in (0,1)$, then $F\left(\mathcal{X}^{n+1}\right) < F(\mathcal{X}^n)$.

Proof. For a tensor $\mathcal{X}$, we have

$$\begin{array}{c} F(\mathcal{X}) = \frac{1}{2}\left\|\mathcal{P}_\Omega(\mathcal{A}-\mathcal{X})\right\|_F^2 = \frac{1}{2} < \mathcal{P}_\Omega(\mathcal{A}-\mathcal{X}), \mathcal{P}_\Omega(\mathcal{A}-\mathcal{X}) > \\ = \frac{1}{2} < (\mathcal{P}_\Omega(\mathcal{A}-\mathcal{X}))_{\langle d\rangle}, (\mathcal{P}_\Omega(\mathcal{A}-\mathcal{X}))_{\langle d\rangle} >= \frac{1}{2}\left\|(\mathcal{P}_\Omega(\mathcal{A}-\mathcal{X}))\right\|_{\langle d\rangle F}^2. \end{array} \tag{8}$$

In (5), $\boldsymbol{u}_{d,i}^n$ are orthogonal to each other, and $\boldsymbol{v}_{d,i}^n$ are orthogonal to each other. Then, according to (8) and (5), we have

$$\begin{array}{c} F(\mathcal{X}^n) = \frac{1}{2}\left\|\left(\mathcal{P}_\Omega(\mathcal{A}-\mathcal{X}^n)\right)_{\langle d\rangle}\right\|_F^2 = \frac{1}{2}\| \sum_{i=1}^{r_d^n} \sigma_{d,i}^n \boldsymbol{u}_{d,i}^n \boldsymbol{v}_{d,i}^{n\ T}\|_F^2 \\ = \frac{1}{2} < \sum_{i=1}^{r_d^n} \sigma_{d,i}^n \boldsymbol{u}_{d,i}^n \boldsymbol{v}_{d,i}^{n\ T}, \sum_{i=1}^{r_d^n} \sigma_{d,i}^n \boldsymbol{u}_{d,i}^n \boldsymbol{v}_{d,i}^{n\ T} >= \frac{1}{2} \sum_{i=1}^{r_d^n} \left(\sigma_{d,i}^n\right)^2. \end{array} \tag{9}$$

According to (8), (7) and (5), we can deduce

$$\begin{array}{l} F\left(\mathcal{X}^{n+1}\right) = \frac{1}{2}\|\left(\mathcal{P}_\Omega\left(\mathcal{A}-\mathcal{X}^{n+1}\right)\right)_{\langle d\rangle}\|_F^2 = \frac{1}{2}\left\|\left(\mathcal{P}_\Omega\left(\mathcal{A}-\mathcal{X}^n-\lambda\ \sigma_{d,1}^n\left(\boldsymbol{u}_{d,1}^n \boldsymbol{v}_{d,1}^{n\ T}\right)^{\langle d\rangle}\right)\right)_{\langle d\rangle}\right\|_F^2 \\ = \frac{1}{2}\|(\mathcal{P}_\Omega(\mathcal{A}-\mathcal{X}^n))_{\langle d\rangle} - \lambda\sigma_{d,1}^n \boldsymbol{u}_{d,1}^n \boldsymbol{v}_{d,1}^{n\ T}\|_F^2 = \frac{1}{2}\left\|\sum_{i=1}^{r_d^n} \sigma_{d,i}^n \boldsymbol{u}_{d,i}^n \boldsymbol{v}_{d,i}^{n\ T} - \lambda\ \sigma_{d,1}^n \boldsymbol{u}_{d,1}^n \boldsymbol{v}_{d,1}^{n\ T}\right\|_F^2 \\ = \frac{1}{2}\|(1-\lambda)\sigma_{d,1}^n \boldsymbol{u}_{d,1}^n \boldsymbol{v}_{d,1}^{n\ T} + \sum_{i=2}^{r_d^n} \sigma_{d,i}^n \boldsymbol{u}_{d,i}^n \boldsymbol{v}_{d,i}^{n\ T}\|_F^2 = \frac{1}{2}\left((1-\lambda)^2\left(\sigma_{d,1}^n\right)^2 + \sum_{i=2}^{r_d^n}\left(\sigma_{d,i}^n\right)^2\right). \end{array} \tag{10}$$

Since $\lambda \in (0,1)$, $F\left(\mathcal{X}^{n+1}\right) < F(\mathcal{X}^n)$. □

3.4. Selection of the Unfolding Direction

In (7), there exist D different directions when unfolding $\mathcal{P}_\Omega(\mathcal{A}-\mathcal{X}^n)$ since $\mathcal{A}$ is a D-order tensor. When $D = 2$, tensor completion is reduced to matrix completion. For a 2-order tensor $\mathcal{M} \in R^{I_1 \times I_2}$, $\mathcal{M}_{\langle 1\rangle} = M$ and $\mathcal{M}_{\langle 2\rangle} = M^T$. The two unfolding matrices have identical singular values and exchange left and right singular vectors, so the choice of unfolding direction will not affect the final results. When $D > 2$, different unfolding directions may lead to different accuracies and convergence speeds, so we need to select the optimal unfolding direction. Considering the symmetry of every dimension of a tensor, the optimal unfolding direction in each iteration may be different. Below, we discuss how to choose the unfolding direction d in each iteration.

Real-world data tensors often exhibit low-rank structures, and tensor completions usually attempt to recover a low-rank tensor that best approximates a partially observed data tensor [24]. In tensor completion, rank is often surrogated by nuclear norm. Overlapped nuclear norm [25] and scaled latent nuclear norm [26] are two commonly used tensor nuclear norms, but the latter is more appropriate than the former when used in tensor completion [27]. For a D-order tensor $\mathcal{X}$, its scaled latent nuclear norm is defined as follows:

$$\|\mathcal{X}\|_{scaled} = \min_{\sum_{d=1}^D \mathcal{X}_d = \mathcal{X}} \sum_{d=1}^D \frac{1}{\sqrt{I_d}}\left\|\left(\mathcal{X}_d\right)_{\langle d\rangle}\right\|_*. \tag{11}$$

In (11), we let $\mathcal{X}_d = \mathcal{X}, \mathcal{X}_i = 0\ (i \neq d)$; then

$$\|\mathcal{X}\|_{scaled} \leq \frac{1}{\sqrt{I_d}} \left\| (\mathcal{X})_{\langle d \rangle} \right\|_*. \tag{12}$$

Therefore, we have

$$\begin{aligned} \left\| \mathcal{P}_\Omega (\mathcal{A} - \mathcal{X}^n) \right\|_{scaled} &\approx \left\| \left(\sigma_{d,1}^n \boldsymbol{u}_{d,1}^n \boldsymbol{v}_{d,1}^{n\ T} \right)^{\langle d \rangle} \right\|_{scaled} \leq \frac{1}{\sqrt{I_d}} \left\| \left(\left(\sigma_{d,1}^n \boldsymbol{u}_{d,1}^n \boldsymbol{v}_{d,1}^{n\ T} \right)^{\langle d \rangle} \right)_{\langle d \rangle} \right\|_* \\ &= \frac{1}{\sqrt{I_d}} \left\| \sigma_{d,1}^n \boldsymbol{u}_{d,1}^n \boldsymbol{v}_{d,1}^n \right\|_*^T = \frac{\sigma_{d,1}^n}{\sqrt{I_d}}. \end{aligned} \tag{13}$$

The nuclear norm is a convex surrogate for rank [28]. In (4), if $\mathcal{X}^n$ and $\mathcal{P}_\Omega(\mathcal{A} - \mathcal{X}^n)$ are both low-rank tensors, then $\mathcal{X}^{n+1}$ will also tend to be a low-rank tensor. Therefore, we choose the unfolding direction d that minimizes the scaled latent nuclear norm of $\mathcal{P}_\Omega(\mathcal{A} - \mathcal{X}^n)$, i.e.,

$$d = arg \min_d \frac{\sigma_{d,1}^n}{\sqrt{I_d}}. \tag{14}$$

The calculation of $\frac{\sigma_{d,1}^n}{\sqrt{I_d}}$ for each dimension d ($d \in \{1, 2, .., D\}$) is independent and we can perform these calculations in parallel.

3.5. Design of the Iteration Step-Size

Another part of (7) is determining the step-size λ of each iteration. We consider the following.

1. In the gradient descent method, $-F\prime(\mathcal{X})$ is the fastest descent direction, and we use the maximum singular value (and corresponding singular vectors) of its unfolding matrix to calculate it approximately. If the maximum singular value is very large, we may ignore some larger singular values, and the approximation of the fastest descent direction may be unsatisfactory. Therefore, we should adopt a small step-size to avoid excessive errors. Conversely, if the maximum singular value is very small, the approximation may be more accurate, and we can adopt a large step-size. In other words, the larger the maximum singular value is, the smaller the step-size.
2. In the gradient descent method, $-F\prime(\mathcal{X})$ will become increasingly smaller during the iterative process; thus, the maximum singular value σ_d^n of each iteration presents a downtrend as a whole. Then, according to Point 1, the step-size should show an upward trend during the iterative process. However, the traditional approach and some related approaches [29,30] all make the step-size increasingly smaller during the iterative process, which does not meet our requirements.
3. The step-size λ can also be viewed as a penalty for the singular value. In matrix completion, the nonconvex function is used to penalize the singular value and achieves a better effect than the direct use of the nuclear norm [31,32]. Reference [32] penalizes the singular value by the $f(x) = log(x + 1)$ function for matrix completion and does not introduce additional parameters.

Based on the above three points, the formula of the iteration step-size is designed as follows:

$$\lambda = \frac{log\left(\sigma_{d,1}^n + 1\right)}{\sigma_{d,1}^n}. \tag{15}$$

In (15), the larger $\sigma_{d,1}^n$ is, the smaller λ, and the singular value after penalty is $log\left(\sigma_{d,1}^n + 1\right)$, which is a nonconvex function. In addition, it easy to prove $\lambda \in (0, 1)$.

3.6. Optimization of Calculation

If we directly use (7) to compute $\mathcal{X}^{n+1}$, we need to store and update each entry of $\mathcal{X}^n$ during the iterative process. The time and space complexities are tremendous. We split (7) into the following two parts:

$$\mathcal{P}_{\Omega}\left(\mathcal{X}^{n+1}\right) = \mathcal{P}_{\Omega}(\mathcal{X}^n) + \mathcal{P}_{\Omega}\left(\lambda\sigma_{d,1}^n\left(\boldsymbol{u}_{d,1}^n \boldsymbol{v}_{d,1}^{n\,T}\right)^{\langle d\rangle}\right), \tag{16}$$

$$\mathcal{P}_{\overline{\Omega}}\left(\mathcal{X}^{n+1}\right) = \mathcal{P}_{\overline{\Omega}}(\mathcal{X}^n) + \mathcal{P}_{\overline{\Omega}}\left(\lambda\sigma_{d,1}^n\left(\boldsymbol{u}_{d,1}^n \boldsymbol{v}_{d,1}^{n\,T}\right)^{\langle d\rangle}\right), \tag{17}$$

where $\overline{\Omega}$ denotes the positions of the missing entries in tensor $\mathcal{A}$. The goal of tensor completion is to calculate the entries in the range of $\overline{\Omega}$. To achieve this, we need to calculate the values of $\sigma_{d,1}^n$, $\boldsymbol{u}_{d,1}^n$, and $\boldsymbol{v}_{d,1}^n$ in each iteration. Because

$$\mathcal{P}_{\Omega}(\mathcal{A} - \mathcal{X}^n) = \mathcal{P}_{\Omega}(\mathcal{A}) -_{\Omega} (\mathcal{X}^n), \tag{18}$$

the values of $\sigma_{d,1}^n$, $\boldsymbol{u}_d^n$, and $\boldsymbol{v}_{d,1}^n$ depend only on $\mathcal{P}_{\Omega}(\mathcal{X}^n)$ but not on $\mathcal{P}_{\overline{\Omega}}(\mathcal{X}^n)$. That is, we just need to store and update the entries of $\mathcal{X}^n$ in the range of Ω (namely, $\mathcal{P}_{\Omega}(\mathcal{X}^n)$) during the iterative process. Furthermore, $\mathcal{P}_{\Omega}(\mathcal{X}^n)$ can be stored in a sparse tensor.

3.7. Analysis of Time Complexity

According to (14), we need to calculate the maximum singular of $(\mathcal{P}_{\Omega}(\mathcal{A} - \mathcal{X}^n))_{\langle d\rangle}$ for each dimension d ($d \in \{1, 2, .., D\}$) in each iteration. The power method is used to calculate the maximum singular of matrix, which itself is also an iterative method. In each iteration of the power method, we need to calculate a matrix-vector multiplication, where the size of the matrix is $I_d \times \frac{I_1 \times I_2 \times \cdots \times I_D}{I_d}$. Thus, the time complexity of the matrix-vector multiplication is at most $I_d \times \frac{I_1 \times I_2 \times \cdots \times I_D}{I_d} = I_1 \times I_2 \times \cdots \times I_D$.

Based on the above analysis, the time complexity of our method is $\mathrm{O}(N \times D \times P \times I_1 \times I_2 \times \cdots \times I_D)$, where N is the number of iterations in the gradient descent method and P is the number of iterations in the power method.

4. Experiments

In this section, we first compare the performance of our NTC against some recently proposed methods and then demonstrate the effectiveness of the step-size design in NTC. Experiments are performed on a PC with Intel i7 CPU and 32 GB RAM. The software environment is MATLAB R2017a on Windows 10. We use the Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) as performance indicators,

$$\mathrm{RMSE} = \sqrt{\frac{1}{\left|\overline{\Omega}_T\right|}\sum\nolimits_{i_1 i_2 \cdots i_D \in \overline{\Omega}_T}\left(\mathcal{X}_{i_1 i_2 \cdots i_D} - \mathcal{A}_{i_1 i_2 \cdots i_D}\right)^2}, \tag{19}$$

$$\mathrm{MAE} = \frac{1}{\left|\overline{\Omega}_T\right|}\sum\nolimits_{i_1 i_2 \cdots i_D \in \overline{\Omega}_T}\left|\mathcal{X}_{i_1 i_2 \cdots i_D} - \mathcal{A}_{i_1 i_2 \cdots i_D}\right|, \tag{20}$$

where $\overline{\Omega}_T$ represents the entries in the testing set and $\left|\overline{\Omega}_T\right|$ is the number of entries in $\overline{\Omega}_T$.

4.1. Performance Comparison

We use image inpainting and link prediction to perform the experiments for performance comparison. Our competitors include (i) SPC [33], which applies the total variation constraint to the low-rank model for tensor completion; (ii) TR-LS [16], which adopts a dual framework to solve the low-rank tensor completion; (iii) Rprecon [17], which is a Riemannian manifold preconditioning approach for tensor completion; (iv) GeomCG [18], which applies Riemannian optimization to the

completion of fixed-rank tensor; (v) FFW [28], which uses the Frank-Wolfe algorithm and scaled latent nuclear norm for tensor completion; and (vi) CTD-S [34], which accelerates tensor decomposition by removing redundancy. They all need to tune several parameters.

The experimental data are divided into two parts: the training set and testing set. For the methods that need to tune some parameters, we take out a subset of data from the training set, and this portion is called the verification set, which is used to learn the optimal parameters. Our method has no parameters, so it does not need a verification set. Each experiment is repeated five times, and the results are averaged. As mentioned, in (14), we can calculate $\frac{\sigma^n_{d,1}}{\sqrt{I_d}}$ for each dimension d ($d \in \{1, 2, .., D\}$) in parallel. However, considering (i) 3-dimensional tensors are used for our tests; (ii) the power method is used to calculate the maximum singular value ($\sigma^n_{d,1}$) quickly; and (iii) parallelization itself also needs a certain amount of overhead, we do not parallelize these calculations in our tests.

Figure 2 shows three RGB images, which are with 720×1280, 600×960, and 1024×1024 pixels, respectively. Each image can be stored in a tensor of size "height" × "width" × 3. For each image, we experiment with two training-test ratios: 25%:75% and 40%:60%. For the methods with parameters, we select 20% of the training set as the verification set (for parameter tuning). The test results are shown in Tables 1 and 2, where we highlight the best result(s) in bold and underline the second-best result(s) in each column.

(**a**) Mountain (**b**) Rice (**c**) Stockton

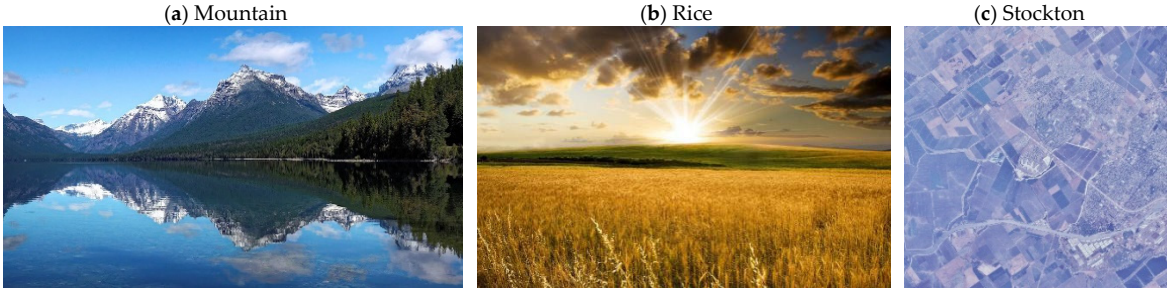

Figure 2. Test images.

Table 1. Test Results of Image Inpainting (Training set is 25%).

Methods	Mountain		Rice		Stockton	
	RMSE	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
NTC	**0.224**	**0.136**	**0.258**	**0.176**	0.502	0.360
SPC	0.307	0.224	0.647	0.541	0.664	0.520
TR-LS	0.266	0.169	0.265	**0.176**	**0.462**	**0.319**
Rprecon	0.281	0.180	0.296	0.208	0.504	0.353
GeomCG	0.402	0.291	0.304	0.210	0.568	0.402
FFW	0.241	0.150	0.272	0.188	0.520	0.376
CTD-S	0.843	0.713	0.867	0.730	0.915	0.782

Table 2. Test Results of Image Inpainting (Training set is 40%).

Methods	Mountain		Rice		Stockton	
	RMSE	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**
NTC	**0.190**	**0.129**	**0.218**	**0.145**	0.458	0.324
SPC	0.254	0.162	0.537	0.439	0.515	0.367
TR-LS	0.236	0.149	0.220	0.147	**0.433**	**0.291**
Rprecon	0.237	0.145	0.290	0.204	0.476	0.332
GeomCG	0.238	0.146	0.290	0.204	0.513	0.363
FFW	0.211	0.132	0.239	0.162	0.481	0.344
CTD-S	0.748	0.628	0.759	0.635	0.858	0.725

It can be seen from Tables 1 and 2 that, on the whole, our NTC has the smallest errors and performs best. TR-LS and FFW can also achieve good results. CTD-S is a noniterative tensor completion method, and its performance is not satisfactory.

Next, we use the latest dataset on MovieLens [35] to build three three-dimensional tensors of "user-movie-category" to perform the experiments. Table 3 presents the related information of the three datasets. The known entries in each tensor (dataset) are randomly divided into a training set and a testing set. For each dataset, we experiment with two training-test ratios: 80%:20% and 60%:40%. Similarly, we select 20% of the training set as a verification set for the methods with parameters. Tables 4 and 5 give the results.

Table 3. Three Datasets on MovieLens.

Name	Size	Number of Entries	Number of Known Entries
300T	$300 \times 4667 \times 19$	26,601,900	66,887
2400T	$2400 \times 11{,}292 \times 19$	514,915,200	611,198
4000T	$4000 \times 13{,}370 \times 19$	1,016,120,000	1,028,536

Table 4. Test Results of Link Prediction (Training set is 80%).

Methods	300T		2400T		4000T	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
NTC	0.919	0.726	**0.853**	0.654	**0.843**	0.647
SPC	0.962	0.776	0.996	0.795	——	——
TR-LS	0.936	0.742	0.858	**0.651**	0.851	0.654
Rprecon	0.963	0.738	0.870	0.657	0.856	0.647
GeomCG	0.944	0.735	0.870	0.661	0.854	**0.641**
FFW	**0.915**	**0.721**	0.860	0.768	0.858	0.662
CTD-S	0.987	0.806	0.969	0.663	0.974	0.773

Table 5. Test Results of Link Prediction (Training set is 60%).

Methods	300T		2400T		4000T	
	RMSE	MAE	RMSE	MAE	RMSE	MAE
NTC	**0.926**	**0.724**	**0.877**	0.677	**0.860**	0.663
SPC	0.968	0.768	0.998	0.798	——	——
TR-LS	0.934	0.735	0.971	0.763	0.867	0.670
Rprecon	0.970	0.738	0.887	**0.664**	0.872	**0.657**
GeomCG	0.975	0.772	0.886	0.672	0.872	0.659
FFW	**0.926**	0.728	0.882	0.684	0.881	0.684
CTD-S	0.993	0.803	0.979	0.778	0.987	0.786

In Tables 4 and 5, each method yields a total of 12 error values. Overall, NTC performs best, as its six errors are the smallest and four errors are the second-smallest. In particular, for the 2400T and 4000T datasets, our NTC achieves the smallest RMSE. TR-LS, Rprecon, GeomCG, and FFW can achieve better results on some datasets, and CTD-S has the highest RMSE and MAE. Unfortunately, 32 GB RAM cannot support SPC working on the 4000T dataset. In a word, our method is competitive with other methods.

4.2. Effectiveness of Our Step-Size Design

We choose 10,532 users and 22,157 movies from the latest dataset on MovieLens to build a two-dimensional tensor. The tensor contains 1,048,575 known entries, where 80% of the data is selected as the training set and the rest is selected as the testing set. We compare the performance of our step-size design against the performances of the other three designs. Our design is denoted as D0,

and the other three designs are denoted as D1, D2, and D3. The four designs make the singular value after penalization (namely, $\lambda \times \sigma_{d,1}^n$) equal to $log(\sigma_{d,1}^n + 1)$, 1, $\sigma_{d,1}^n$, and $\sigma_{d,1}^n/2^n$, respectively. Figure 3 demonstrates their RMSE curves versus CPU time.

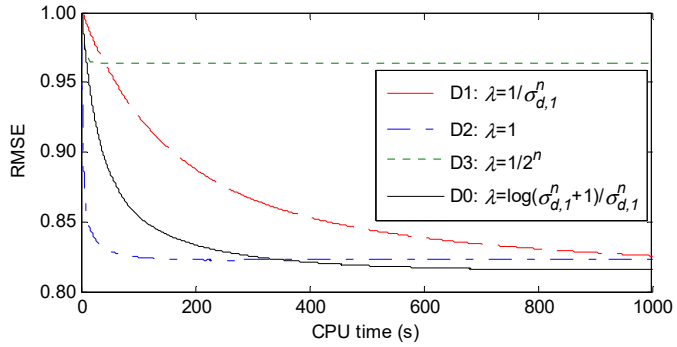

Figure 3. The performances of different step-size designs.

From Figure 3 we can see that (i) D3 performs the worst, and its RMSE is significantly higher than that of other designs; (ii) D0 and D2 have better performances than D1; and (iii) D2 converges very fast in the first 300 s, but then is overtaken by D0. Overall, our design (D0) can achieve a higher convergence speed and a lower error.

5. Conclusions

This paper proposes a new Nonparametric Tensor Completion (NTC) method based on gradient descent and nonconvex penalty. Our method formulates tensor completion as an unconstrained optimization problem and designs an efficient iterative method to solve it. We use gradient descent to solve the optimization problem of tensor completion and build a gradient tensor with tensor matricizations and SVD. We select the optimal direction based on the scaled latent nuclear norm and design the formula of iteration step-size elaborately. Furthermore, during the iterative process, we store the tensor in sparsity and adopt the power method to compute the maximum singular value quickly. Unlike existing methods, our method has no parameters and is easily manipulated. We use image inpainting and link prediction to compare NTC against six state-of-the-art methods, and the test results demonstrate that NTC is competitive with them. In addition, an experiment of two-dimensional tensor completion shows the effectiveness of our step-size design.

In this paper, we select the optimal unfolding direction by scaled latent nuclear norm to construct an iterative formula. Next, we will try to select more than one unfolding directions and weight these directions so as to achieve lower error and faster convergence speed.

Author Contributions: The authors contributed equally to the theoretical framing and algorithms and the corresponding author took principle responsibility for writing the article.

Funding: This research was funded by the National Natural Science Foundation of China, grant number 61202366; and the Natural Science Foundation of Guangdong Province, grant number 2018A030313438.

Acknowledgments: The authors are grateful to the editors and anonymous referees for their helpful comments and suggestions.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hu, Y.; Zhang, D.; Ye, J.; Li, X.; He, X. Fast and accurate matrix completion via truncated nuclear norm regularization. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 2117–2130. [CrossRef] [PubMed]
2. Symeonidis, P. ClustHOSVD: Item recommendation by combining semantically enhanced tag clustering with tensor HOSVD. *IEEE Trans. Syst. Man Cybern. Syst.* **2016**, *46*, 1240–1251. [CrossRef]

3. Shlezinger, N.; Dabora, R.; Eldar, Y.C. Using mutual information for designing the measurement matrix in phase retrieval problems. In Proceedings of the IEEE International Symposium on Information Theory, Aachen, Germany, 25–30 June 2017; pp. 2343–2347.
4. Li, S.; Shao, M.; Fu, Y. Multi-view low-rank analysis with applications to outlier detection. *ACM Trans. Knowl. Discov. Data* **2018**, *12*, 32. [CrossRef]
5. Cheng, M.; Jing, L.; Ng, M.K. Tensor-based low-dimensional representation learning for multi-view clustering. *IEEE Trans. Image Process.* **2019**, *28*, 2399–2414. [CrossRef]
6. Zhao, G.; Tu, B.; Fei, H.; Li, N.; Yang, X. Spatial-spectral classification of hyperspectral image via group tensor decomposition. *Neurocomputing* **2018**, *316*, 68–77. [CrossRef]
7. Cichocki, A.; Phan, A.H.; Zhao, Q.B.; Lee, N.; Oseledets, I.; Sugiyama, M.; Mandic, D.P. Tensor networks for dimensionality reduction and large-scale optimizations part 2 applications and future perspectives. *Found. Trends Mach. Learn.* **2017**, *9*, 431–673. [CrossRef]
8. Koren, Y.; Bell, R.; Volinsky, C. Matrix factorization techniques for recommender systems. *Computer* **2009**, *42*, 30–37. [CrossRef]
9. Wen, Z.; Yin, W.; Zhang, Y. Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm. *Math. Program. Comput.* **2012**, *4*, 333–361. [CrossRef]
10. Rahmani, M.; Atia, G.K. High dimensional low rank plus sparse matrix decomposition. *IEEE Trans. Signal Process.* **2017**, *65*, 2004–2019. [CrossRef]
11. Kilmer, M.E.; Martin, C.D. Factorization strategies for third-order tensors. *Linear Algebra Appl.* **2011**, *435*, 641–658. [CrossRef]
12. Kilmer, M.E.; Braman, K.; Hao, N.; Hoover, R.C. Third-order tensors as operators on matrices: A theoretical and computational framework with applications in imaging. *SIAM J. Matrix Anal. Appl.* **2013**, *34*, 148–172. [CrossRef]
13. Cai, Y.; Zhang, M.; Luo, D.; Ding, C.; Chakravarthy, S. Low-order tensor decompositions for social tagging recommendation. In Proceedings of the 4th ACM international conference on Web search and data mining, Hong Kong, China, 9–12 February 2011; pp. 695–704.
14. Lee, H.; Yoo, J.; Choi, S. Semi-supervised nonnegative matrix factorization. *IEEE Signal Process. Lett.* **2009**, *17*, 4–7.
15. Yuan, M.; Zhang, C.H. On tensor completion via nuclear norm minimization. *Found. Comput. Math.* **2016**, *16*, 1031–1068. [CrossRef]
16. Nimishakavi, M.; Jawanpuria, P.; Mishra, B. A dual framework for low-rank tensor completion. In Proceedings of the 32nd Conference on Neural Information Processing Systems, Montréal, QC, Canada, 3–8 December 2018.
17. Kasai, H.; Mishra, B. Low-rank tensor completion: A Riemannian manifold preconditioning approach. In Proceedings of the 33rd International Conference on Machine Learning, New York, NY, USA, 19–24 June 2016; pp. 1012–1021.
18. Kressner, D.; Steinlechner, M.; Vandereycken, B. Low-rank tensor completion by Riemannian optimization. *BIT Numer. Math.* **2014**, *54*, 447–468. [CrossRef]
19. Xu, Y.; Hao, R.; Yin, W.; Su, Z. Parallel matrix factorization for low-rank tensor completion. *Inverse Probl. Imaging* **2015**, *9*, 601–624. [CrossRef]
20. Yu, W.; Zhang, H.; He, X.; Chen, X.; Xiong, L.; Qin, Z. Aesthetic-based clothing recommendation. In Proceedings of the 2018 World Wide Web Conference, Lyon, France, 23–27 April 2018; pp. 649–658.
21. Halko, N.; Martinsson, P.G.; Tropp, J.A. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Rev.* **2011**, *53*, 217–288. [CrossRef]
22. Ji, T.Y.; Huang, T.Z.; Zhao, X.L.; Ma, T.H.; Deng, L.J. A non-convex tensor rank approximation for tensor completion. *Appl. Math. Model.* **2017**, *48*, 410–422. [CrossRef]
23. Derksen, H. On the nuclear norm and the singular value decomposition of tensor. *Found. Comput. Math.* **2016**, *16*, 779–811. [CrossRef]
24. Yao, Q.; Kwok, J.T. Accelerated and inexact soft-impute for large-scale matrix and tensor completion. *IEEE Trans. Knowl. Data Eng.* **2019**, *31*, 1665–1679. [CrossRef]
25. Liu, J.; Musialski, P.; Wonka, P.; Ye, J.P. Tensor completion for estimating missing values in visual data. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 208–220. [CrossRef]

26. Wimalawarne, K.; Sugiyama, M.; Tomioka, R. Multitask learning meets tensor factorization: Task imputation via convex optimization. In Proceedings of the International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 2825–2833.
27. Tomioka, R.; Suzuki, T. Convex tensor decomposition via structured schatten norm regularization. In Proceedings of the 26th International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013; pp. 1331–1339.
28. Guo, X.; Yao, Q.; Kwok, J.T. Efficient sparse low-rank tensor completion using the Frank-Wolfe algorithm. In Proceedings of the 31st AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; pp. 1948–1954.
29. Lu, S.; Hong, M.; Wang, Z. A nonconvex splitting method for symmetric nonnegative matrix factorization: Convergence analysis and optimality. In Proceedings of the 2017 IEEE International Conference on Acoustics, Speech and Signal Processing, New Orleans, LA, USA, 5–9 March 2017; pp. 2572–2576.
30. Kuang, D.; Yun, S.; Park, H. SymNMF: Nonnegative low-rank approximation of a similarity matrix for graph clustering. *J. Glob. Optim.* **2015**, *62*, 545–574. [CrossRef]
31. Oh, T.H.; Matsushita, Y.; Tai, Y.W.; Kweon, I.S. Fast randomized singular value thresholding for nuclear norm minimization. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition, Boston, MA, USA, 7–12 June 2015; pp. 4484–4493.
32. Nie, F.; Hu, Z.; Li, X. Matrix completion based on non-convex low-rank approximation. *IEEE Trans. Image Process.* **2019**, *28*, 2378–2388. [CrossRef] [PubMed]
33. Yokota, T.; Zhao, Q.; Cichocki, A. Smooth PARAFAC decomposition for tensor completion. *IEEE Trans. Signal Process.* **2016**, *64*, 5423–5436. [CrossRef]
34. Lee, J.; Choi, D.; Sael, L. CTD: Fast, accurate, and interpretable method for static and dynamic tensor decompositions. *PLoS ONE* **2018**, *13*, e0200579. [CrossRef]
35. Harper, F.M.; Konstan, J.A. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.* **2016**, *5*, 19. [CrossRef]