High-dimension Tensor Completion via Gradient-based Optimization Under Tensor-train Format

Longhao Yuan\textsuperscript{a,b}, Qibin Zhao\textsuperscript{b,c}, Jianting Cao\textsuperscript{d,a,∗}

\textsuperscript{a}Graduate School of Engineering, Saitama Institute of Technology, Japan
\textsuperscript{b}Tensor Learning Unit, RIKEN Center for Advanced Intelligence Project (AIP), Japan
\textsuperscript{c}School of Automation, Guangdong University of Technology, China
\textsuperscript{d}School of Computer Science and Technology, Hangzhou Dianzi University, China

Abstract

In this paper, we propose a novel approach to recover the missing entries of incomplete data represented by a high-dimension tensor. Tensor-train decomposition, which has powerful tensor representation ability and is free from ‘the curse of dimensionality’, is employed in our approach. By observed entries of incomplete data, we consider to find the factors which can capture the latent features of the data and then reconstruct the missing entries. With low-rank assumption to the original data, tensor completion problem is cast into solving optimization models. Gradient descent methods are applied to optimize the core tensors of tensor-train decomposition. We propose two algorithms: Tensor-train Weighted Optimization (TT-WOPT) and Tensor-train Stochastic Gradient Descent (TT-SGD) to solve tensor completion problems. A high-order tensorization method named visual data tensorization (VDT) is proposed to transform visual data to higher-order forms by which the performance of our algorithms can be improved. The synthetic data experiments and visual data experiments show that our algorithms outperform the state-of-the-art completion algorithms. Especially in high-dimension, high missing rate and large-scale data cases, significant performance can be obtained from our algorithms.

Keywords: tensor completion, data recovery, high-dimension, tensor-train

∗Corresponding author
Email address: cao@sit.ac.jp (Jianting Cao)
decomposition, low-rank approximation, tensorization, gradient-based optimization

1. Introduction

Tensors are high-dimension generalization of vectors and matrices. Representing data by tensor can retain the high dimensional form of data and keep adjacent structure relation information of data. Most of the real world data are more than two dimensions. For example, RGB images are three-dimension tensors \((\text{height} \times \text{width} \times \text{channel})\), videos are four-dimension tensors \((\text{height} \times \text{width} \times \text{channel} \times \text{time})\) and electroencephalography (EEG) signals are three-dimension tensors \((\text{magnitude} \times \text{trails} \times \text{time})\). When facing data with more than two dimensions, traditional methods usually lower data dimension by concatenation and transform data into matrices or vectors, which lead to spatial redundancy and less efficient factorization[1]. In recent years, many theories, algorithms and applications of tensor methodologies have been studied and proposed [2, 3, 4]. Due to the high compression ability and data representation ability of tensor decomposition, many applications related to tensor decomposition have been proposed in variety of fields such as image and video completion [5, 6], signal processing [7, 8], brain computer interface [9], image classification [10], etc.

In practical situations, data missing is ubiquitous due to error and noise in data collecting process which can generate outliers and unwanted data entries. Generally, the main difficulty of estimating missing data is to find the relation between missing entries and observed entries. Tensor decomposition is to decompose tensor data into a sequence of factors which can catch the latent features of the whole data. The basic concept of solving data completion problems by tensor decomposition is that we decompose the incomplete data into factors which contain the latent features of the data, then we use the powerful feature representation ability of the factors to approximate the missing entries. The most studied and standard tensor decomposition models are CAN-
DECOMP/PARAFAC (CP) decomposition [11, 12] and Tucker decomposition [13, 14, 15]. CP decomposition decomposes a tensor into a sum of rank-one tensors, Tucker decomposition represents a tensor as a core tensor and several factor matrices. There are many proposed tensor completion methods which employ the above tensor decomposition models. In [5], CP weighted optimization (CP-WOPT) is proposed. It formulates tensor completion problem as a weighted least squares (WLS) problem and uses optimization method to find the optimal CP factors. Fully Bayesian CP Factorization (FBCP) in [6] employs Bayesian probabilistic model to find the optimal CP factors and CP rank at the same time. Three low-rank approximation algorithms are proposed in [16]. They extend low-rank matrix completion model to tensor and find the low-rank tensor structure by trace norm minimization. In [17], Tucker low-n-rank tensor completion (TLnR) method is proposed, and the experiments show better results than other trace norm minimization methods.

Though CP and Tucker can obtain a relatively high performance in low-order tensors, due to the nature limitation of these two models, when it comes to the situations of high-dimensional tensor, the calculation efficiency will decrease rapidly and the number of model parameters will grow exponentially. In recent years, a matrix product state (MPS) model named tensor-train (TT) is proposed and becomes more and more popular [18, 19, 20]. For an N-th order tensor, if the size of every dimension is I, CP decomposition represents data by $O(NIr)$ parameters, Tucker model needs $O(NIr + r^N)$ parameters, and TT model requires $O(NIr^2)$ parameters. TT decomposition scales linearly to tensor dimension which is the same as CP decomposition. Though CP model is more compact by ranks, it is difficult to find the optimal CP factors especially when the tensor order is high. Tucker model is more flexible and stable, but model parameters will grow exponentially when the tensor dimension increases. Tensor-train is free from ‘the curse of dimensionality’ so it is a better model to process high-dimensional tensor data.

In this paper, we mainly focus on applying TT model to data completion problem. Though several tensor completion methods based on TT model have
been proposed recently [21], [22], their applicability and effectiveness are limited. To the best of our knowledge, applying gradient descent method on TT-based tensor completion is firstly proposed in this paper. The main works of this paper are concluded as follows: 1) Based on optimization methodology and tensor-train decomposition, we propose two algorithms named Tensor-train Weighted Optimization (TT-WOPT) and Tensor-train Stochastic Gradient Descent (TT-SGD) which use gradient-based optimization method to find the latent factors. 2) We conduct simulation experiments in different tensor dimensions, and compare our algorithms to the state-of-the-art algorithms. 3) We propose a tensorization method named Visual Data Tensorization (VDT) to transform visual data to higher dimensions, by which the performance of our algorithms is improved. 4) We test the performance of our algorithms on benchmark RGB images, a color video and a hyperspectral image, and compare our algorithms to the state-of-the-art algorithms.

The rest of the paper is organized as follows. In section 2, we state the notations of this paper and introduce tensor-train decomposition. In section 3 we present the two proposed tensor completion algorithms and compare the computational complexity for every iteration. In section 4, the performance of our algorithms is tested by synthetic data and visual data, and compare to the state-of-the-art algorithms. We conclude our work in section 5.

2. Preliminaries and Related works

2.1. Notations

Notations in [2] are adopted in our paper. A scalar is denoted by a normal lowercase letter, e.g., \( x \in \mathbb{R} \), and a vector is denoted by a boldface lowercase letter, e.g., \( \mathbf{x} \in \mathbb{R}^I \). A matrix is denoted by a boldface capital letter, e.g., \( \mathbf{X} \in \mathbb{R}^{I \times J} \). A tensors of order \( N \geq 3 \) is denoted by Euler script letters, e.g., \( \mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N} \).

\( x^{(1)}, x^{(2)}, \cdots, x^{(N)} \) denotes a vector sequence, and \( x^{(n)} \) denotes the \( n \)th vector of the vector sequence. The representations of matrix sequences and tensor
sequences are denoted in the same way. An element of tensor $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ of index $\{i_1, i_2, \cdots, i_N\}$ is denoted by $x_{i_1 i_2 \cdots i_N}$ or $\mathbf{X}(i_1, i_2, \cdots, i_N)$. Mode-$n$ matricization (unfolding) of tensor $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is denoted by $\mathbf{X}^{(n)} \in \mathbb{R}^{I_n \times I_{n-1} I_{n+1} \cdots I_N}$. The inner product of two tensor $\mathbf{X}, \mathbf{Y}$ with the same size $\mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is defined as:
\[
\langle \mathbf{X}, \mathbf{Y} \rangle = \sum_{i_1} \sum_{i_2} \cdots \sum_{i_N} x_{i_1 i_2 \cdots i_N} y_{i_1 i_2 \cdots i_N}.
\]
Furthermore, the Frobenius norm of $\mathbf{X}$ is defined by $\|\mathbf{X}\|_F = \sqrt{\langle \mathbf{X}, \mathbf{X} \rangle}$.

The Hadamard product is denoted by $\ast$. It is an element-wise product of vectors, matrices or tensors of same sizes. For example, given tensor $\mathbf{X}, \mathbf{Y} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, $\mathbf{Z} = \mathbf{X} \ast \mathbf{Y}$, then $\mathbf{Z} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, and $z_{i_1 i_2 \cdots i_N} = x_{i_1 i_2 \cdots i_N} y_{i_1 i_2 \cdots i_N}$. The Kronecker product of two matrices $\mathbf{X} \in \mathbb{R}^{I \times K}$ and $\mathbf{Y} \in \mathbb{R}^{J \times L}$ is $\mathbf{X} \otimes \mathbf{Y} \in \mathbb{R}^{IJ \times KL}$, see details in [2].

2.2. Tensor-train Decomposition

The most significant feature of tensor-train decomposition is that the number of model parameters will not grow exponentially by the increase of tensor dimension. Tensor-train decomposition is to decompose a tensor into a sequence of three-order core tensors (factor tensors): $\mathbf{G}^{(1)}, \mathbf{G}^{(2)}, \cdots, \mathbf{G}^{(N)}$. The relation between the approximated tensor $\mathbf{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and core tensors can be expressed as follow:
\[
\mathbf{X} = \ll \mathbf{G}^{(1)}, \mathbf{G}^{(2)}, \cdots, \mathbf{G}^{(N)} \gg,
\]
where for $n = 1, \cdots, N$, $\mathbf{G}^{(n)} \in \mathbb{R}^{r_{n-1} \times I_n \times r_n}$, $r_0 = r_N = 1$. The notation $\ll \cdot \gg$ is the operation to convert the core tensors to the approximated tensor. For overall expression convenience, $\mathbf{G}^{(1)} \in \mathbb{R}^{I_1 \times I_2 \times r_1}$ and $\mathbf{G}^{(N)} \in \mathbb{R}^{r_{N-1} \times I_N \times 1}$ are considered as two special tensors. The sequence $r_0, r_1, \cdots, r_N$ is named TT-rank which limits the size of every core tensor. Furthermore, the $(i_1, i_2, \cdots, i_N)$th element of tensor $\mathbf{X}$ can be represented by the mode-2 slice of the corresponding core tensors as follow:
\[
x_{i_1 i_2 \cdots i_N} = \prod_{n=1}^{N} G_{i_n}^{(n)},
\]
where $G^{(1)}_{1}, \ldots, G^{(N)}_{N}$ is the sequence of tensor slices. For $n = 1, 2, \ldots, N$, $G^{(n)}_{in} \in \mathbb{R}^{r_n \times r_{n-1}}$ is mode-2 slice extracted from $G^{(n)}$ according to each mode of the index of $x_{i_1i_2\cdots i_N}$. $G^{(1)}_{i_1} \in \mathbb{R}^{1 \times r_1}$ and $G^{(N)}_{i_N} \in \mathbb{R}^{r_{N-1} \times 1}$ are extracted from first core tensor and last core tensor, they are considered as two special matrices for overall expression convenience.

3. Tensor-train Completion Algorithms

3.1. Tensor-train Weighted Optimization (TT-WOPT) Algorithm

Define $Y \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is the observed tensor with missing entries and $X \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is the tensor approximated by core tensors of TT decomposition.

For the completion problem of tensors with missing entries, the positions of the missing entries need to be specified. So we define a binary tensor $W \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ named weight tensor in which the indices of missing entries and observed entries of tensor $Y$ can be recorded. Every entry of $W$ meets:

$$w_{i_1i_2\cdots i_N} = \begin{cases} 0 & \text{if } y_{i_1i_2\cdots i_N} \text{ is a missing entry}, \\ 1 & \text{if } y_{i_1i_2\cdots i_N} \text{ is an observed entry}. \end{cases}$$

In this algorithm, the missing entries of $Y$ are filled with zero, so $Y$ is a real number tensor. The problem of finding the decomposition factors of an incomplete tensor can be formulated by a weight least squares (WLS) model. Define $Y_w = W * Y$, and $X_w = W * X$, then WLS model for calculating tensor decomposition factors is formulated by:

$$f(G^{(1)}, G^{(2)}, \ldots, G^{(N)}) = \frac{1}{2} \|Y_w - X_w\|_F^2.$$ 

This is an optimization objective function w.r.t. all the core tensors and it can be solved by optimization method. The relation between observed tensor $Y$ and core tensors can be deduced as the following equation [23]:

$$Y_{(n)} = G^{(n)}_{(2)} (G^{(n)}_{(1)} \otimes G^{(n)}_{(n)}),$$

where for $n = 1, \ldots, N$,

$$G^{(n)} = \ll G^{(n+1)}, \ldots, G^{(N)} \gg \in \mathbb{R}^{r_n \times I_{n+1} \times \cdots \times I_N},$$
\[ G^{\leq n} = \ll G^{(1)}, G^{(2)}, \ldots, G^{(n-1)} \rr \in \mathbb{R}^{I_1 \times \cdots \times I_{n-1} \times R_{n-1}}, \quad (8) \]

and \( G^{>N} = G^{<1} = 1 \).

For \( n = 1, \ldots, N \), the partial derivatives of the objective function w.r.t. the mode-2 matricization of the \( n \)th core tensor \( G^{(n)} \) can be inferred as follow:

\[ \frac{\partial f}{\partial G^{(n)}_{(2)}} = (X_w(n) - Y_w(n))(G^{>n}_{(1)} \otimes G^{<n}_{(1)})^T. \quad (9) \]

After the objective function and gradients are obtained, the core tensors can be optimized by various optimization algorithms. The optimization procedure of the TT-WOPT algorithm is listed in Alg.1.

**Algorithm 1 Tensor-train Weighted Optimization (TT-WOPT)**

1. **Input**: incomplete tensor \( Y \), weight tensor \( W \) and TT-rank \( r \).
2. **Initialization**: core tensors \( G^{(1)}, G^{(2)}, \ldots, G^{(N)} \) of approximated tensor \( X \).
3. **While** the optimization stopping condition is not satisfied
4. Compute \( X_w = W * \ll G^{(1)}, G^{(2)}, \ldots, G^{(N)} \rr \).
5. **For** \( n=1:N \)
6. Compute gradients of every core tensor according to equation (9).
7. **End**
8. Update \( G^{(1)}, G^{(2)}, \ldots, G^{(N)} \) by gradient descend method.
9. **End while**
10. **Output**: \( G^{(1)}, G^{(2)}, \ldots, G^{(N)} \).

### 3.2. Tensor-train Stochastic Gradient Descent (TT-SGD) Algorithm

As seen from equation (5), TT-WOPT uses whole scale of the observed data for every iteration. The computation of the gradients is redundant because the space of missing entries are still used for calculation. If the scale of data is extremely huge and the number of missing entries is high, then only a small amount of observed entries is useful. In this situation, TT-WOPT will waste a lot of computational storage and the computation will become time-consuming. In order to solve the problems of TT-WOPT as mentioned above, we propose
another algorithm, TT-SGD, which only uses one observed entries to compute
the gradients for every iteration is proposed.

Stochastic Gradient Descent (SGD) method has been applied in matrix and
tensor decompositions [24, 25, 26]. For every optimization iteration, we only
use one entry which is randomly sampled from the observed entries to optimize
the according parts of the core tensors. TT-SGD is able to process large-scale
dataset and has more scalability and efficiency. For one observed entry of index
\{i_1, i_2, \cdots, i_N\}, we have \( x = X(i_1, i_2, \cdots, i_N), y = Y(i_1, i_2, \cdots, i_N) \). So by equation
\[ (3), \] the objective function is formulated by:

\[
f(G^{(1)}_{i_1}, G^{(2)}_{i_2}, \cdots, G^{(N)}_{i_N}) = \frac{1}{2} \left\| y - \prod_{k=1}^{N} G^{(k)}_{i_k} \right\|_F^2. \tag{10}\]

For \( n = 1, 2, \cdots, N \), the partial derivatives of every corresponding slice \( G^{(n)}_{i_n} \)
w.r.t. index \( \{i_1, i_2, \cdots, i_N\} \) of this entry is calculated as:

\[
\frac{\partial f}{\partial G^{(n)}_{i_n}} = (x - y)(\prod_{k=n+1}^{N} G^{(k)}_{i_k}\prod_{k=1}^{n-1} G^{(k)}_{i_k})^T. \tag{11}\]

From the equation we can see, the computational complexity of TT-SGD is
not related to the scale of the observed tensor or the number of observed entries,
so it can process large-scale data by much smaller computational complexity
than TT-WOPT. This algorithm is also suitable for online/real-time learning.
The TT-SGD algorithm is listed below:

3.3. Computational Complexity

For tensor \( X \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N} \), we assume all \( I_1, I_2, \cdots, I_N \) is equal to \( I \), and
\( r_1 = r_2 = \cdots = r_{N-1} = r \). According to equation \[ (9), \] the time complexity and
the space complexity of TT-WOPT are \( O(r^{N-1}I^{N-1}) \) and \( O(IN + r^2I^{N-1}) \)
respectively. According to \[ (11), \] the time complexity and the space complexity
of TT-SGD are \( O(r^{N-1}) \) and \( O(rN) \) respectively. TT-SGD is free from
data dimensionality so it is more suitable to process large-scale data. Though
TT-WOPT has larger computational complexity, it has a steady and fast con-
vergence when processing normal-size data.
Algorithm 2 Tensor-train Stochastic Gradient Descent (TT-SGD)

1: **Input:** incomplete tensor $\mathbf{Y}$ and $TT$ − rank $r$.
2: **Initialization:** core tensors $G^{(1)}, G^{(2)}, \ldots, G^{(N)}$ of approximated tensor $\mathbf{X}$. 
3: **While** the optimization stopping condition is not satisfied
4: Randomly sample one observed entry from $\mathbf{Y}$ w.r.t. index $\{i_1, i_2, \ldots, i_N\}$.
5: **For** $n=1:N$
6: Compute the gradients of the according tensor slices by equation (11).
7: **End**
8: Update $G^{(1)}_{i_1}, G^{(2)}_{i_2}, \ldots, G^{(N)}_{i_N}$ by gradient descent method.
9: **End while**
10: **Output:** $G^{(1)}, G^{(2)}, \ldots, G^{(N)}$.

4. Experiment results

We conduct simulation experiments by synthetic data, and for real world data experiments, we test our algorithms by color images, a video and a hyperspectral image. Our two algorithms are compared with several state-of-the-art algorithms: CP-WOPT [5], FBCP [6], HaLRTC and FaLRTC [16], and TLnR [17]. We apply relative square error (RSE) of approximated tensor $\mathbf{X}$ and observed tensor $\mathbf{Y}$ to evaluate the performance of each algorithm. RSE is defined as:

$$RSE = \frac{\|\mathbf{Y} - \mathbf{X}\|_F}{\|\mathbf{Y}\|_F}.$$  \hspace{1cm} (12)

In addition, to evaluate the performance for only the missing entries, we define a weighted RSE as:

$$RSE_{\bar{w}} = \frac{\|\mathbf{Y}_{\bar{w}} - \mathbf{X}_{\bar{w}}\|_F}{\|\mathbf{Y}_{\bar{w}}\|_F},$$  \hspace{1cm} (13)

where $\mathbf{Y}_{\bar{w}} = \bar{\mathbf{W}} \ast \mathbf{Y}$ and $\mathbf{X}_{\bar{w}} = \bar{\mathbf{W}} \ast \mathbf{X}$. $\bar{\mathbf{W}}$ is a weight tensor in which every entry satisfies:

$$\bar{w}_{i_1i_2\ldots i_N} = \begin{cases} 1 & \text{if } y_{i_1i_2\ldots i_N} \text{ is a missing entry,} \\ 0 & \text{if } y_{i_1i_2\ldots i_N} \text{ is an observed entry.} \end{cases}$$  \hspace{1cm} (14)
For experiments of random missing cases, we randomly remove data points according to different missing rates. Missing rate $m_r$ is defined as:

$$m_r = 1 - \frac{M}{N \prod_{k=1}^{I_k}},$$  \hspace{1cm} (15)$$

where $M$ is the number of observed or sampled entries. To evaluate the quality of approximated visual data, we introduce PSNR (Peak Signal-to-noise Ratio) as an image evaluation index.

For optimization method of TT-WOPT, in order to have a clear comparison with CP-WOPT, we adopt the same optimization method as paper [5]. It uses nonlinear conjugate gradient (NCG) with Hestenes-Stiefel updates [27] and the Moré-Thuente line search method [28]. The optimization method is implemented by an optimization toolbox named Pablano Toolbox [29]. For TT-SGD, we employ an algorithm named Adaptive Moment Estimation (Adam) which has prominent performance on stochastic gradient-based optimization as our gradient descent method [30, 31]. The update rule of Adam is as follow:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t} + \epsilon} m_t,$$  \hspace{1cm} (16)$$

where $t$ is the iteration time of optimization parameter $\theta$, $\eta$ and $\epsilon$ are hyper parameters, $m_t$ and $v_t$ are the first moment estimate and second moment estimate of gradient $g_t$ respectively. $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$, $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$, where $\beta_1$ and $\beta_2$ are hyper parameters. For choosing the hyper parameters in Adam, we adopt the reference values from paper [30]. The values of $\beta_1$, $\beta_2$ and $\epsilon$ are set as 0.9, 0.999 and $10^{-8}$ respectively. In our experiments, we mainly tune the learning rate $\eta$ from values in $\{0.0001, 0.0005, 0.001\}$ to get the best optimization results. In addition, all the data in our experiments are regularized to 0 to 1 to make the algorithms more effective.

We adopt two optimization stopping conditions, one is the error of two adjacent iterations of the objective function value: $f_t - f_{t-1} \leq tol$, where $f_t$ is the objective function value of the $t$th iteration. In our experiment, we set $tol = 1e-4$. The other stopping condition is the maximum iteration time, we
set it according to the scale of data and different algorithms. If one of the two conditions is satisfied, the optimization will be stopped.

4.1. Synthetic Data

We apply synthetic data produced from a highly oscillating function: $f(x) = \sin \frac{1}{4} \cos(x^2)$ [32] in our simulation experiments. The synthetic data is expected to be well approximated by tensor decomposition models. For simplicity, we set all the dimensions and the tt-ranks as a same value, $I_1 = I_2 = \cdots = I_N = I$, $r_0 = r_1 = \cdots = r_N = r$. We sample $I^N$ data from the values generated from the highly oscillating function, then the data is reshaped to the required tensor size. We employ four different tensor structures: $26 \times 26 \times 26$ (3-D), $7 \times 7 \times 7 \times 7 \times 7$ (5-D), $4 \times 4 \times 4 \times 4 \times 4 \times 4 \times 4$ (7-D), $3 \times 3 \times 3 \times 3 \times 3 \times 3 \times 3 \times 3$ (9-D). Then we test TT-WOPT, TT-SGD, CP-WOPT, FBCP, HaLRTC, FaLRTC and TLnR on the synthetic data. The hyper-parameters of each algorithm are tuned to obtain the best performance.

The graphs of Figure 1 show the experiment results. RSE values change w.r.t. different $m_r$ (from 0 to 0.9) of the seven algorithms under the four different synthetic tensors. HaLRTC and FaLRTC cannot test fully observed case, so $m_r$ range of the two algorithms is from 0.2 to 0.9. From the figure we can see that TT-WOPT and TT-SGD perform the best at 5-D, 7-D, and 9-D cases. Though the performance of TT-SGD is a little weaker than TT-WOPT in some cases, the RSE values of TT-SGD always stay stable at around 0.02. In 3-D case, HaLRTC and FBCP perform well under low $m_r$ situations. However, when $m_r$ is over 0.6, performance of these algorithms decrease and our proposed algorithms show better results.

4.2. Visual Data Tensorization (VDT) method

From the simulation results we can see, our proposed algorithms have good and stable performance in high-dimensional tensors. So we provide a Visual Data Tensorization (VDT) method to transform low-order tensor to higher-order tensor. This method is suitable for visual data and can improve the
Figure 1: RSE comparison of seven algorithms under four data dimensions. Missing rate is tested from 0 to 0.9.

The VDT method is derived from an image compression and entanglement methodology [33]. It is a method of transforming a gray-scale image of size $2^l \times 2^l$ into a real ket of a Hilbert space. The method cast the image to a higher-order tensor structure with an appropriate block structured addressing. We generalize this method to a more flexible constraint so that this method can be applied to various data sizes. Naturally, for RGB image, video, hyperspectral image, etc., the data can be represented by a tensor $\mathbf{Y} \in \mathbb{R}^{U \times V \times C_1 \times \cdots \times C_s}$, where the first two dimension of data is visual
image dimensions and the latter dimensions are channels which represent RGB channles, time, trails, etc.. However, 2D representation of image cannot fully exploit the correlation and local structure of the data, our VDT can strengthen the ‘local region correlation’ of visual data. Tensorize visual data by VDT method operates as follow: if the first two dimensions of a visual data tensor is $U \times V$ and can be reshaped to $u_1 \times u_2 \times \cdots \times u_l \times v_1 \times v_2 \times \cdots \times v_l$, then we permute and reshape the data to size $u_1v_1 \times u_2v_2 \times \cdots \times u_lv_l$. This higher-order tensor is a new tensorization structure of the original data. The first dimension of this tensorized data corresponds to a $u_1 \times v_1$ pixel block of the image, and the following dimensions of $u_2v_2, \cdots, u_lv_l$ describe the expanding larger-scale partition of the image. Based on VDT method, TT-based algorithms can efficiently exploit the structure information of visual data and have a better low-rank representation. After the tensorized data is processed by the completion algorithms, a reverse operation of VDT is conducted to get the original image form.

To testify the effectiveness of our VDT method, we choose a benchmark image ‘lena’ with randomly erased 90% entries. We compare the performance of the six algorithms (TT-WOPT, CP-WOPT, CP-WOPT, FBCP, HaLRTC and TLnR) under three different data structures: three-order tensor, nine-order tensor without VTD, nine-order tensor generated by VTD method. The three-order tensor data uses original image data structure of size $256 \times 256 \times 3$. The nine-order tensor without VTD is generated by directly reshaping data to the size $4 \times 4 \times 4 \times 4 \times 4 \times 4 \times 4 \times 4 \times 4 \times 3$. For nine-order tensor with VTD method, firstly the original data is reshaped to a seventeen-way tensor of size $2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 3$ and then it is permuted according to the order of $\{1 \ 9 \ 2 \ 10 \ 3 \ 11 \ 4 \ 12 \ 5 \ 13 \ 6 \ 14 \ 7 \ 15 \ 8 \ 16 \ 17\}$. Finally we reshape the tensor to a nine-order tensor of size $4 \times 4 \times 4 \times 4 \times 4 \times 4 \times 4 \times 4 \times 3$. This nine-order tensor with VTD is considered to be a better structure of the image data. The first dimension of the nine way tensor contains the data of a $2 \times 2$ pixel block of the image and the following orders of the tensor describe the expanding pixel blocks of the image.

Figure 2 and table 1 show the visual results and numerical results of the six
algorithms under the three different data structure. We can see that in three-order tensor case, the results among the algorithms are similar. However, for nine-order cases, other four algorithms fail the completion task while TT-WOPT and TT-SGD performs well. Furthermore, when the image is transformed to nine-order tensor by VDT method, we see the distinct improvement of our two algorithms.

Figure 2: Simulation results for random missing ( \( m_r = 0.9 \) ) of ‘lena’ image under six algorithms. The first row uses original three-order tensor data, the second row uses nine-order tensor data without VDT, and the third row uses nine-order tensor data with proposed VDT method.

Table 1: Comparison of the inpainting performance (RSE and PSNR) of six algorithms under three tensor form of image ‘lena’.

|                | TT-WOPT | TT-SGD | CP-WOPT | FBCP | HaLRTC | TLnR |
|----------------|---------|--------|---------|------|--------|------|
| three-order    |         |        |         |      |        |      |
| RSE            | 0.2822  | 0.2604 | 0.3392  | **0.1942** | 0.1981 | 0.6552 |
| PSNR           | 16.12   | 16.84  | 14.53   | **19.36** | 19.18  | 8.802 |
| nine-order     |         |        |         |      |        |      |
| RSE            | **0.1558** | 0.1793 | 0.2562  | 0.2682 | 0.9310 | 1.207 |
| PSNR           | **21.31** | 20.06  | 16.95   | 16.57 | 5.746  | 3.486 |
| nine-order VDT |         |        |         |      |        |      |
| RSE            | **0.1262** | 0.1493 | 0.2573  | 0.2687 | 0.9301 | 0.7114 |
| PSNR           | **23.21** | 21.77  | 16.97   | 16.57 | 5.751  | 10.84 |
4.3. Benchmark Image Completion

Significant improvement of our algorithms can be seen when we use higher-order image data processed by VDT method. However for algorithms which are based on CP decomposition and Tucker decomposition, tensorization to higher-order will exponentially increase the model parameters and make it harder to compute optimal tensor factors, so the performance will decrease. In later experiments, we only apply VDT method to TT-WOPT and TT-SGD. For other compared algorithms, we keep the original data structure to get better results.

In this section, we consider several irregular missing cases (scratch missing, whole row missing and block missing) and a 99% missing case. For TT-WOPT and TT-SGD, we transform image data to nine-order tensor by VDT method. For CP-WOPT, FBCP, HaLRTC and TLnR, we use the original image size $256 \times 256 \times 3$. From Figure 3 and Table. 2 we can see, for irregular missing experiments, low RSE values can often be obtained from HaLRTC, this is because HaLRTC algorithm puts observed entries to the final result and only approximate observed entries. However, RSE values of HaLRTC are higher than our proposed algorithms, so our algorithms can better approximate the missing entries of data. 99% random missing experiment is a challenge task among all the image completion algorithms. Our two proposed algorithms with VDT method can achieve a high performance under this situation while other four algorithms totally fail.

4.4. Video Completion

For color video completion task, we only test our TT-SGD because TT-SGD is better for large-scale data than TT-WOPT. In addition, when large-scale data is tested, many algorithms which work well on benchmark images become inefficiently or ineffectively. In the following experiments, we only compare CP-WOPT, FBCP and HaLRTC which are functional for large-scale data with our proposed TT-SGD. For applying VDT method on a video data of size $256 \times 320 \times 3 \times 100$, we first reshape the data to size $4 \times 2 \times 2 \times 2 \times 2 \times 5 \times 2 \times 2 \times 2 \times 2 \times 2 \times 3 \times 100$, then permute it by index $\{1 8 2 9 3 10 4 11 5 12 6 13 7 14 15 16\}$,
then we reshape it to size $20 \times 4 \times 4 \times 4 \times 3 \times 100$. We compare three random missing cases ($m_r = 0.7$, $m_r = 0.9$ and $m_r = 0.99$) in this experiment. Figure 4 is the visual results of TT-SGD. We select two frames (frame 1 and frame 75) from the video and show the completion results of three missing rates in this figure. Table 3 shows the numerical results of TT-SGD, CP-WOPT, FBCP and HaLRTC. We can see that TT-SGD outperforms other algorithms and can recover the video well even there is only 1% sampled entries. It should also be noted that TT-SGD converges much faster than the other compared algorithms.

4.5. Hyperspectral Image Completion

In this section, we test TT-SGD, CP-WOPT, FBCP and HaLRTC on a hyperspectral image (HSI) of size $256 \times 256 \times 191$ recorded by a satellite. Due to the bad working condition of satellite sensors, the collected data often has Gaussian noise, impulse noise, dead lines, and stripes. In this experiment, we firstly consider the situation when the HSI has dead lines. It is a common missing case
Table 2: Comparison of the inpainting performance (RSE, $\textit{RSE}_w$, and PSNR) of six algorithms under four missing conditions.

|               | TT-WOPT | TT-SGD | CP-WOPT | FBCP   | HaLRTC | TLnR |
|---------------|---------|--------|---------|--------|--------|------|
| **Scratch**   |         |        |         |        |        |      |
| RSE           | 0.1584  | 0.2052 | 0.1944  | 0.1942 | **0.1554** | 0.3780 |
| $\textit{RSE}_w$ | **0.187** | 0.216  | 0.2422  | 0.2457 | 0.2033 | 0.4955 |
| PSNR          | 21.04   | 18.66  | 19.14   | 19.12  | **21.07** | 13.32 |
| **Row**       |         |        |         |        |        |      |
| RSE           | **0.1683** | 0.2009 | 0.5554  | 0.5483 | 0.539  | 0.540 |
| $\textit{RSE}_w$ | **0.1967** | 0.2138 | 1.0181  | 1.000  | 1.000  | 1.000 |
| PSNR          | **20.76** | 19.23  | 10.32   | 10.43  | 10.58  | 10.55 |
| **Block**     |         |        |         |        |        |      |
| RSE           | 0.1680  | 0.1849 | 0.1794  | 0.1679 | **0.1051** | 0.2182 |
| $\textit{RSE}_w$ | 0.2473  | **0.2286** | 0.3254  | 0.2572 | 0.2295 | 0.3761 |
| PSNR          | 20.90   | 20.07  | 20.30   | 20.90  | **24.98** | 18.58 |
| **99% random**|         |        |         |        |        |      |
| RSE           | 0.3285  | **0.3165** | 1.130   | 0.4080 | 0.9231 | 1.533 |
| $\textit{RSE}_w$ | 0.3294  | **0.3176** | 1.144   | 0.4092 | 0.9282 | 1.542 |
| PSNR          | 16.05   | **16.29** | 5.175   | 14.02  | 6.926  | 2.527 |

Table 3: Comparison of the inpainting performance (RSE and PSNR) of the four algorithms under three random missing cases.

| Algorithm | $m_r = 0.7$ | $m_r = 0.9$ | $m_r = 0.99$ |
|-----------|--------------|--------------|--------------|
|           | RSE PSNR     | RSE PSNR     | RSE PSNR     |
| **TT-SGD** | **0.0878 27.61** | **0.1342 23.93** | **0.1394 23.60** |
| CP-WOPT   | 0.2108 20.00  | 0.2455 18.67  | 0.2671 17.94  |
| FBCP      | 0.1777 21.48  | 0.2174 19.73  | 0.2972 17.02  |
| HaLRTC    | 0.1000 26.50  | 0.1676 21.99  | 0.7737 8.704  |

In recording the HSI by satellite. Then we consider the case when only 1% of the data is obtained, which is useful in data compression and transformation. For data preprocessing, we transform the HSI data to $16 \times 16 \times 16 \times 16 \times 191$ by VDT method for TT-SGD, and apply original three-order tensor form as the input of the other compared algorithms. We set tt-ranks of deadline missing case as 48 and 99% missing case as 24. Figure 5 shows the visual results of the four algorithms at first channel of the HSI.

In the experiment, TT-SGD performs best among the algorithms at both dead line missing case and 99% random missing case. In 99% random missing
case, HaLRTC totally fail, CP-WOPT and FBCP obtain lower performance than TT-SGD. In addition, it should be noted that the volume of data is about $1.25 \times 10^7$, and when the iteration reaches $1 \times 10^6$ (16% of the total data), the optimization of TT-SGD is converged. This indicates that TT-SGD has much faster computation and it is more efficient than other compared algorithms.
5. Conclusion

In this paper, in order to solve tensor completion problem, based on tensor-train decomposition, we propose two algorithms named TT-WOPT and TT-SGD. We first cast the problem to optimization models, then we use gradient descent methods to find the optimal core tensors of TT decomposition. Finally the core tensors are used to approximate the missing entries of incomplete data. We conduct simulation experiments and visual data experiments, and compare our algorithms to the state-of-the-art algorithms. From the simulation experiments we can see, the performance of our algorithms stays stable when the tensor dimension increases, while the performance of other algorithms drops quickly. To tensorize visual data to a higher-order tensor, we propose VDT method. Visual data experiments show that after high-order tensorization by VDT method, the performance of our two algorithms improves. Furthermore, we conduct several visual data experiments. Our algorithms outperform the
state-of-the-art algorithms in various missing cases, particularly when the tensor dimension is high and the missing rate is high, our algorithms with VDT method can process extreme data missing situation (i.e., 99% missing) well while other algorithms fail. Besides, our proposed TT-SGD has low computational complexity and shows distinct performance in large-scale data.

Significant performance of TT-based tensor completion algorithms proves that it is a promising aspect. It should be noted that the selection of tt-rank is important to obtain better experiment results. So we will find a better scheme to automatically choose tt-rank properly in our future work.

References

[1] A. Shashua, T. Hazan, Non-negative tensor factorization with applications to statistics and computer vision, in: Proceedings of the 22nd international conference on Machine learning, ACM, 2005, pp. 792–799.

[2] T. G. Kolda, B. W. Bader, Tensor decompositions and applications, SIAM review 51 (3) (2009) 455–500.

[3] M. A. O. Vasilescu, D. Terzopoulos, Multilinear subspace analysis of image ensembles, in: Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on, Vol. 2, IEEE, 2003, pp. II–93.

[4] T. Franz, A. Schultz, S. Sizov, S. Staab, Triplerank: Ranking semantic web data by tensor decomposition, The Semantic Web-ISWC 2009 (2009) 213–228.

[5] E. Acar, D. M. Dunlavy, T. G. Kolda, M. Mørup, Scalable tensor factorizations for incomplete data, Chemometrics and Intelligent Laboratory Systems 106 (1) (2011) 41–56.
[6] Q. Zhao, L. Zhang, A. Cichocki, Bayesian cp factorization of incomplete tensors with automatic rank determination, IEEE transactions on pattern analysis and machine intelligence 37 (9) (2015) 1751–1763.

[7] L. De Lathauwer, J. Castaing, Blind identification of underdetermined mixtures by simultaneous matrix diagonalization, IEEE Transactions on Signal Processing 56 (3) (2008) 1096–1105.

[8] D. Muti, S. Bourennane, Multidimensional filtering based on a tensor approach, Signal Processing 85 (12) (2005) 2338–2353.

[9] J. Mocks, Topographic components model for event-related potentials and some biophysical considerations, IEEE transactions on biomedical engineering 35 (6) (1988) 482–484.

[10] A. Shashua, A. Levin, Linear image coding for regression and classification using the tensor-rank principle, in: Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, Vol. 1, IEEE, 2001, pp. I–I.

[11] L. Sorber, M. Van Barel, L. De Lathauwer, Optimization-based algorithms for tensor decompositions: Canonical polyadic decomposition, decomposition in rank-(l1,r1,l2,r2,...,ln) terms, and a new generalization, SIAM Journal on Optimization 23 (2) (2013) 695–720.

[12] J. H. d. M. Goulart, M. Boizard, R. Boyer, G. Favier, P. Comon, Tensor cp decomposition with structured factor matrices: Algorithms and performance, IEEE Journal of Selected Topics in Signal Processing 10 (4) (2016) 757–769.

[13] L. R. Tucker, Some mathematical notes on three-mode factor analysis, Psychometrika 31 (3) (1966) 279–311.

[14] L. De Lathauwer, B. De Moor, J. Vandewalle, On the best rank-1 and rank-(r 1, r 2,..., r n) approximation of higher-order tensors, SIAM journal on Matrix Analysis and Applications 21 (4) (2000) 1324–1342.
[15] C.-Y. Tsai, A. M. Saxe, D. Cox, Tensor switching networks, in: Advances in Neural Information Processing Systems, 2016, pp. 2038–2046.

[16] J. Liu, P. Musialski, P. Wonka, J. Ye, Tensor completion for estimating missing values in visual data, IEEE Transactions on Pattern Analysis and Machine Intelligence 35 (1) (2013) 208–220.

[17] M. Filipović, A. Jukić, Tucker factorization with missing data with application to low-n-rank tensor completion, Multidimensional systems and signal processing 26 (3) (2015) 677–692.

[18] I. V. Oseledets, Tensor-train decomposition, SIAM Journal on Scientific Computing 33 (5) (2011) 2295–2317.

[19] H. N. Phien, H. D. Tuan, J. A. Bengua, M. N. Do, Efficient tensor completion: Low-rank tensor train, arXiv preprint arXiv:1601.01083.

[20] Y. Yang, D. Krompass, V. Tresp, Tensor-train recurrent neural networks for video classification, in: International Conference on Machine Learning, 2017, pp. 3891–3900.

[21] W. Wang, V. Aggarwal, S. Aeron, Tensor completion by alternating minimization under the tensor train (tt) model, arXiv preprint arXiv:1609.05587.

[22] J. A. Bengua, H. N. Phien, H. D. Tuan, M. N. Do, Efficient tensor completion for color image and video recovery: Low-rank tensor train, IEEE Transactions on Image Processing 26 (5) (2017) 2466–2479.

[23] A. Cichocki, N. Lee, I. Oseledets, A.-H. Phan, Q. Zhao, D. P. Mandic, et al., Tensor networks for dimensionality reduction and large-scale optimization: Part 1 low-rank tensor decompositions, Foundations and Trends® in Machine Learning 9 (4-5) (2016) 249–429.

[24] R. Gemulla, E. Nijkamp, P. J. Haas, Y. Sismanis, Large-scale matrix factorization with distributed stochastic gradient descent, in: Proceedings of
the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2011, pp. 69–77.

[25] T. Maehara, K. Hayashi, K.-i. Kawarabayashi, Expected tensor decomposition with stochastic gradient descent., in: AAAI, 2016, pp. 1919–1925.

[26] Y. Wang, A. Anandkumar, Online and differentially-private tensor decomposition, in: Advances in Neural Information Processing Systems, 2016, pp. 3531–3539.

[27] S. Wright, J. Nocedal, Numerical optimization, Springer Science 35 (67-68) (1999) 7.

[28] J. J. Moré, D. J. Thuente, Line search algorithms with guaranteed sufficient decrease, ACM Transactions on Mathematical Software (TOMS) 20 (3) (1994) 286–307.

[29] D. M. Dunlavy, T. G. Kolda, E. Acar, Poblano v1.0: A matlab toolbox for gradient-based optimization, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, Tech. Rep. SAND2010-1422.

[30] D. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980.

[31] S. Ruder, An overview of gradient descent optimization algorithms, arXiv preprint arXiv:1609.04747.

[32] B. N. Khoromskij, Tensor numerical methods for multidimensional pdes: theoretical analysis and initial applications, ESAIM: Proceedings and Surveys 48 (2015) 1–28.

[33] J. I. Latorre, Image compression and entanglement, arXiv preprint quant-ph/0510031.

[34] H. Zhang, W. He, L. Zhang, H. Shen, Q. Yuan, Hyperspectral image restoration using low-rank matrix recovery, IEEE Transactions on Geoscience and Remote Sensing 52 (8) (2014) 4729–4743.