Vol.2018-AVM-101 No.11
2018/6/7

欠損有りデータを対象としたテンソル分解に基づくオンライン低ランク部分空間追跡法OLSTEC

笠井 裕之 [†1,a)]

概要：本稿では，欠損データを含む高次元ストリームデータを対象とした低ランク部分空間追跡問題に着目する．当該データが低次元線形空間に位置することを仮定し，本問題をオンライン型・低ランクテンソル補完問題として定義し，CANDECOMP/PARAFAC (CP) テンソル分解に基づく OnLine Low-rank Subspace tracking by TEnsor CP Decomposition (OLSTEC) 法を提案する．特に，着目するデータが時事刻々と入力されるストリームデータで，且つデータの部分空間が緩やかに変化する状況を想定する．提案法の OLSTEC 法は，逐次最小二乗法 (RLS) による速い収束性能を有する勾配法に基づいている．

Online low-rank subspace tracking by tensor decomposition under incomplete data: OLSTEC

Hiroyuki Kasai [†1,a)]

Abstract: This paper considers the problem of online tensor subspace tracking of a partially observed high-dimensional data stream corrupted by noise, where we assume that the data lie in a low-dimensional linear subspace. This problem is cast as an online low-rank tensor completion problem. We propose a novel online tensor subspace tracking algorithm based on the CANDECOMP/PARAFAC (CP) decomposition, dubbed OnLine Low-rank Subspace tracking by TEnsor CP Decomposition (OLSTEC). The proposed algorithm specifically addresses the case in which data of interest are fed into the algorithm over time infinitely, and their subspace are dynamically time-varying. To this end, we build up our proposed algorithm exploiting the recursive least squares (RLS), which is a second-order gradient algorithm.

1. Introduction

The analysis of *big data* characterized by a huge volume of massive data is at the very core of recent machine learning, signal processing, and statistical learning [*1]. The data have a naturally *multi-dimensional* structure, and they are represented by a multi-dimensional array matrix, namely, a *tensor*. When the data are high-dimensional data corrupted by noise, it is very challenging to reveal the underlying latent structure, such as, to obtain meaningful information, to impute missing elements, to remove the noise, or to predict some future behaviors of data of interest. For this purpose, one typical but promising approach exploits the structural assumption that the data of interest have *low-dimensional subspace*, i.e., *low-rank*, in every dimension. Many data analysis tasks are achieved efficiently by considering *singular value decomposition* (SVD), which reveals the latent subspace of the data. However, when the data have missing elements caused by, for example, system error, or communication error, SVD cannot be applied directly. To address this shortcoming, low-rank *tensor completion* has been studied intensively in recent years. A *convex relaxation* [2], [3], [4] approach, which is a popular method, estimates the subspace by minimizing the sum of the nu-

[†1] 現在，電気通信大学 大学院情報理工学研究科
Presently with Graduate School of Informatics and Engineering, The University of Electro-Communications

[a)] kasai@is.uec.ac.jp

[*1] 本稿は [1] の拡張・短縮版であり，MATLAB コードは `https://github.com/hiroyuki-kasai/OLSTEC` から取得可能である．

clear norms of the unfolding matrices of the tensor of interest. This approach extends the successful results in the matrix completion problem [5] accompanied with theoretical performance guarantees. However, because of the high computation cost necessary for the SVD calculation of big matrices every iteration, its scalability is limited on very large-scale data. Instead, a *fixed-rank* non-convex approach with tensor decomposition [6], [7] has gained great attentions recently because of superior performance in practice irrespective of introduction of local minima. This performance also derives from the success of matrix cases [8], [9], [10]. This paper follows the same line as that of the latter approach.

When the data are acquired sequentially from time to time, it is more challenging because of the need for *online-based* analysis without storing all of the past data as well as without reliance on the *batch-based* process. From this perspective, the batch-based SVD approach is inefficient. It cannot be applied for real-time processing. For this problem, *online subspace tracking* plays a fundamentally important role in various data analyses to avoid expensive repetitive computations and high memory consumption.

This present paper particularly addresses three special but realistic situations that arise in the online subspace tracking in practical applications: (i) We consider a *pure online and streaming* setting, where data of interest is fed into the algorithm over time infinitely. For this problem, some existing algorithms in the category of the so-called *streaming* or *online-based* algorithms cannot fully handle such a situation. They actually process new available data only once without storing them in an online manner. However, they update an entire spaces of their model parameters every iteration. This suffers, sooner or later, from the limitation of the computational capacity when data is fed infinitely. (ii) Considering *time-varying* dynamic nature of real-world streaming data, because there might not exist a *unique* and *stationary* subspace over time, we are often required to update such a time-varying subspace from moment to moment. Despite allowing moderate accuracy of subspace estimation, this update makes existing batch-based algorithms useless. In fact, as the experiments described later in the paper reveal clearly, such batch-based approaches do not work well under such a situation. (iii) Furthermore, we consider the situations and applications where *computational speed* is faster than *data acquiring speed.* If the computational complexity per iteration is constant across time and it is affordable in the computational resource, we prefer the algorithm with faster *convergence rate* in terms of iteration rather than the algorithm with faster computational speed.

For all of these reasons, we particularly address the recursive least squares (RLS) algorithm. Although the RLS does not give higher precision from the viewpoint of the optimization theory [11], it fits the dynamic situation as considered herein because it achieves much faster convergence rate per iteration as a result of the second-order optimization feature.

This paper presents a new online tensor tracking algorithm, dubbed OnLine Low-rank Subspace tracking by TEnsor CP Decomposition (OLSTEC), for a partially observed high-dimensional data stream corrupted by noise. We specifically examine the fixed-rank tensor completion algorithm with the second-order gradient descent based on the CP decomposition exploiting the RLS. The advantage of the proposed algorithm, OLSTEC, is quite robust for dynamically time-varying subspace, which often arises in practical applications. This engenders faster update of sudden change of subspaces of interest. This capability is revealed in the numerical experiments conducted with several benchmarks.

2. Preliminaries and related work

2.1 Preliminaries

Hereinafter, we denote scalars by lower-case letters $(a, b, c, \ldots)$, vectors as bold lower-case letters $(\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}, \ldots)$, and matrices as bold-face capitals $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \ldots)$. An element at (i, j) of a matrix $\mathbf{A}$ is represented as $\mathbf{A}_{i,j}$. It is noteworthy that the transposed column vector of the i-th row vector $\mathbf{A}_{i,:}$ is specially denoted as $\boldsymbol{a}^i$ with superscript to express a row vector explicitly, i.e., a horizontal vector. We designate a multidimensional or multi-*way* (also called *order* or *mode*) array as a *tensor*, which is denoted by $(\boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{B}}, \boldsymbol{\mathcal{C}}, \ldots)$. Tensor *slice* matrices are defined as two-dimensional matrices of a tensor, defined by fixing all but two indices. For example, a *horizontal slice* and a *frontal slices* of a third-order tensor $\boldsymbol{\mathcal{A}}$ are denoted, respectively, as $\boldsymbol{\mathcal{A}}_{i,:,:}$ and $\boldsymbol{\mathcal{A}}_{:,:,k}$. Also, $\boldsymbol{\mathcal{A}}_{:,:,k}$ is used heavily in this article. Therefore, it is simply expressed as $\mathbf{A}_k$ using the bold-face capital font and a single subscript to represent its matrix form explicitly. $\mathrm{rank}(\boldsymbol{\mathcal{X}})$ is the rank of $\boldsymbol{\mathcal{X}}$. Including some above, we basically follow the tensor notation of the review article [12] throughout our article and refer to it for additional details. Finally, $\boldsymbol{a}[t]$ and $\mathbf{A}[t]$ with the *square bracket* represent the computed $\boldsymbol{a}$ and $\mathbf{A}$ after performing t-times updates (iterations) in the online algorithm. The notation $\mathrm{diag}(\boldsymbol{a})$ stands for the diagonal matrix with $\{\boldsymbol{a}_i\}$ as diagonal elements. The symbol $\circledast$ denotes the Hadamard Product, which is the element-wise product. $\|\mathbf{A}\|_F$ represents the Frobenius norm.

2.2 Related work

Representative research of the *matrix-based* online algorithm is the projection approximation subspace tracking (PAST) [13]. GROUSE [14] proposes an incremental gradient descent algorithm performed on the Grassmannian $\mathcal{G}(d,n)$, the space of all d-dimensional subspace of $\mathbb{R}^n$ [15], [16]. The algorithm minimizes on $\ell2$-norm cost function. GRASTA [17] enhances robustness against outliers by exploiting $\ell1$-norm cost function. PETRELS [18] calculates the underlying subspace via a discounted recursive process for each row of the subspace matrix in parallel.

As for the *tensor-based* online algorithm, which is our main focus in this paper, Nion and Sidiropoulos propose an adaptive algorithm to obtain the CP decompositions [19]. Yu et al. also propose an accelerated online tensor learning algorithm (ALTO) based on the Tucker decomposition [20]. However, they do not deal with a missing data presence. Mardani et al. propose an online imputation algorithm based on the CP decomposition under the presence of missing data [21], which is called as TeCPSGD in this paper. This considers the stochastic gradient descent (SGD) for large-scale data. This work bears resemblance to the contribution of the present paper. However, considering situations in which the subspace changes dramatically and the processing speed is sufficiently faster than data acquiring speed, a faster convergence rate algorithm per iteration is crucially important to track this change. Because it is well-known that SGD shows a slow convergence rate as the experiments described later in the paper, it is not suitable for this situation. Kasai and Mishra also propose a novel Riemannian manifold preconditioning approach for the tensor completion problem with multi-linear rank constraint based on the Tucker decomposition [22]. The specific Riemannian metric allows the use of versatile framework of Riemannian optimization on quotient manifolds to develop a preconditioned SGD algorithm (RPTucker). Very recently, Nimishakavi et al. propose a dynamic tensor completion framework called Side Information infused Incremental Tensor Analysis (SIITA), which incorporates side information and works for general incremental tensors based on the Tucker decomposition [23]. However, RPTucker and SIITA do not deal with the pure online subspace tracking scenario. Namely, they update the entire spaces of the factor matrices every iteration, and thus the calculation costs of the factor matrices that grow over time become prohibitively increase.

All previously described algorithms are first-order algorithms. For that reason and because of their poor curvature approximations in ill-conditioned problems, their convergence rate can be slow. One promising approach in the literature is second-order stochastic gradient algorithms such as stochastic quasi-Newton (QN) methods using Hessian evaluations. Numerous reports of the literature describe studies of stochastic versions of deterministic quasi-Newton methods [24], [25], [26], [27] with higher scalability in the number of variables for large-scale data. AdaGrad, which estimates the diagonal of the squared root of the covariance matrix of the gradients, was proposed [28]. SGD-QN exploits a *diagonal rescaling matrix* based on the *secant condition* with quasi-Newton method [29]. A direct extension of the deterministic BFGS using stochastic gradients and Hessian approximations is known as online BFGS [30]. Its variants include [30], [31], [32]. Overall, they achieve a higher convergence rate by exploiting curvature information of the objective function. Nevertheless, it is unclear whether they are effective under the online tensor subspace tracking applications.

3. Proposed OLSTEC

3.1 Problem formulation

Similarly to the state-of-the-art tensor tacking algorithms [21], [22], [23], this paper assumes that the rank is given or estimated. Without loss of generality, we particularly examine the third order tensor, and its one order increases *over time*. In other words, we address $\boldsymbol{\mathcal{Y}} \in \mathbb{R}^{L\times W\times T}$ of which third order increases infinitely. Assuming that $\boldsymbol{\mathcal{Y}}_{i_1,i_2,i_3}$ are only known for some indices $(i_1,i_2,i_3)\in\Omega$, where Ω is a subset of the complete set of indices (i_1,i_2,i_3), a general *batch-based* fixed-rank tensor completion problem is formulated as

$$\begin{array}{ll} \min\limits_{\boldsymbol{\mathcal{X}}\in\mathbb{R}^{L\times W\times T}} & \frac{1}{2}\|\mathcal{P}_\Omega(\boldsymbol{\mathcal{X}})-\mathcal{P}_\Omega(\boldsymbol{\mathcal{Y}})\|_F^2 \\ \text{subject to} & \text{rank}(\boldsymbol{\mathcal{X}})=R, \end{array} \tag{1}$$

where the operator $\mathcal{P}_\Omega(\boldsymbol{\mathcal{X}})_{i_1,i_2,i_3}=\boldsymbol{\mathcal{X}}_{i_1,i_2,i_3}$ if $(i_1,i_2,i_3)\in\Omega$ and $\mathcal{P}_\Omega(\boldsymbol{\mathcal{X}})_{i_1,i_2,i_3}=0$ otherwise. rank$(\boldsymbol{\mathcal{X}})$ is the rank of $\boldsymbol{\mathcal{X}}$ (see [12] for a details of tensor rank). $R \ll \{L,W,T\}$ enforces a low-rank structure. Then, the problem (1) is reformulated with ℓ_2-norm regularizers as [21]

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \frac{1}{2}\|\mathcal{P}_\Omega(\boldsymbol{\mathcal{Y}})-\mathcal{P}_\Omega(\boldsymbol{\mathcal{X}})\|_F^2+\mu(\|\mathbf{A}\|_F^2+\|\mathbf{B}\|_F^2+\|\mathbf{C}\|_F^2)$$

$$\text{subject to} \quad \mathbf{X}_\tau = \mathbf{A}\mathrm{diag}(\boldsymbol{b}^\tau)\mathbf{C}^T \qquad \text{for } \tau = 1, \ldots, T. \tag{2}$$

where $\mu > 0$ is a regularization parameter. This regularizer suppresses the instability of RLS. Consequently, considering the situation where the partially observed tensor slice $\mathbf{\Omega}_\tau \circledast \mathbf{Y}_\tau$ is acquired sequentially over time, we estimate $\{\mathbf{A}, \mathbf{B}, \mathbf{C}\}$ by minimizing the exponentially weighted least squares;

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} \frac{1}{2}\sum_{\tau=1}^{t}\lambda^{t-\tau}\Bigg[\|\mathbf{\Omega}_\tau \circledast \big[\mathbf{Y}_\tau - \mathbf{A}\mathrm{diag}(\boldsymbol{b}^\tau)\mathbf{C}^T\big]\|_F^2 + \bar{\mu}(\|\mathbf{A}\|_F^2 + \|\mathbf{C}\|_F^2) + \mu[\tau]\|\boldsymbol{b}^\tau\|_2^2\Bigg]. \tag{3}$$

Therein, $\mu[t]$ is the regularizer parameter for $\boldsymbol{b}$, $\bar{\mu} = \mu[\tau]/\sum_{\tau=1}^{t}\lambda^{t-\tau}$, and $0 < \lambda \leq 1$ is the so-called forgetting parameter. The problem (3) with $\lambda = 1$ is equivalent to the batch-based problem (2).

3.2 Algorithm of OLSTEC

The unknown variables in (3) are $\mathbf{A}, \mathbf{C}$, and $\boldsymbol{b}$. Also, $\mathbf{A}$ and $\mathbf{C}$ are a *non-convex* set. Therefore, this function results in non-convex. The proposed OLSTEC algorithm, as summarized by Algorithm 1, alternates between least-squares estimation of $\boldsymbol{b}[t]$ for fixed $\mathbf{A}[t-1]$ and $\mathbf{C}[t-1]$, and second-order stochastic gradient steps using the RLS algorithm on $\mathbf{A}[t-1]$ and $\mathbf{C}[t-1]$ for fixed $\boldsymbol{b}[t]$. It is noteworthy that $\mathbf{W}[t]$ with the square bracket represents the calculated $\mathbf{W}$ after performing t-times updates.

3.2.1 Calculation of $\boldsymbol{b}[t]$

The estimate $\boldsymbol{b}[t]$ is obtained in a closed form by minimizing the residual by fixing $\{\mathbf{A}[t-1], \mathbf{C}[t-1]\}$ derived at time $t-1$. Then, we obtain $\boldsymbol{b}[t]$ as

$$\boldsymbol{b}[t] = \left[\mu[t]\mathbf{I}_R + \sum_{l=1}^{L}\sum_{w=1}^{W}[\mathbf{\Omega}_t]_{l,w}\boldsymbol{g}_{l,w}[t](\boldsymbol{g}_{l,w}[t])^T\right]^{-1} \left[\sum_{l=1}^{L}\sum_{w=1}^{W}[\mathbf{\Omega}_t]_{l,w}\mathbf{Y}[t]_{l,w}\boldsymbol{g}_{l,w}[t]\right]. \tag{4}$$

Therein, $\boldsymbol{g}_{l,w}[t] = \boldsymbol{a}^l[t-1] \circledast \boldsymbol{c}^w[t-1] \in \mathbb{R}^R$.

3.2.2 Calculation of $\mathbf{A}[t]$ and $\mathbf{C}[t]$ based on RLS

The calculation of $\mathbf{C}[t]$ uses $\mathbf{A}[t-1]$, and the calculation of $\mathbf{A}[t]$ uses $\mathbf{C}[t-1]$. This paper addresses a second-order stochastic gradient based on the RLS algorithm with the forgetting parameters. As for $\mathbf{A}[t]$, defining

$$\mathbf{RA}_l[t] = \lambda\mathbf{RA}_l[t-1] + \sum_{w=1}^{W}[\mathbf{\Omega}_t]_{l,w}\boldsymbol{\alpha}_w[t](\boldsymbol{\alpha}_w[t])^T + (\mu[t] - \lambda\mu[t-1])\mathbf{I}_R, \tag{5}$$

$\boldsymbol{a}^l[t]$ is obtained as presented below.

Algorithm 1 OLSTEC algorithm

```
Require: ${\mathbf{Y}_t \text{ and } \mathbf{\Omega}_t\}_{t=1}^{\infty}$, $\lambda$, $\mu[t]$ for $t = 1, 2, \cdots$.
Initialize $\{\mathbf{A}[0], \boldsymbol{b}[0], \mathbf{C}[0]\}$, $\mathbf{Y}[0] = \mathbf{0}$, $(\mathbf{RA}_l[0])^{-1} = (\mathbf{RC}_w[0])^{-1} = \gamma\mathbf{I}_R, \gamma > 0$.
for $t = 1, 2, \cdots$ do
    Calculate $\boldsymbol{b}[t]$
    for $l = 1, 2, \cdots, L$ do
        Calculate $\mathbf{RA}_l[t]$ and $\boldsymbol{a}^l[t]$
    end for
    for $w = 1, 2, \cdots, W$ do
        Calculate $\mathbf{RC}_l[t]$ and $\boldsymbol{c}^w[t]$
    end for
end for
return $\mathbf{X}_t = \mathbf{A}[t]\mathrm{diag}(\boldsymbol{b}[t])(\mathbf{C}[t])^T$
```

$$\boldsymbol{a}^l[t] = \boldsymbol{a}^l[t-1] - (\mu[t] - \lambda\mu[t-1])(\mathbf{RA}_l[t])^{-1}\boldsymbol{a}^l[t-1] + \sum_{w=1}^{W}[\mathbf{\Omega}_t]_{l,w}\left([\mathbf{Y}_t]_{l,w} - (\boldsymbol{\alpha}_w[t])^T\boldsymbol{a}^l[t-1]\right) \cdot(\mathbf{RA}_l[t])^{-1}\boldsymbol{\alpha}_w[t]. \tag{6}$$

As in the $\mathbf{A}[t]$ case, $\mathbf{C}[t]$ is also obtainable

3.3 Accelerated OLSTEC (OLSTEC-A)

Addressing that the calculation cost of the inversion of $\mathbf{RA}[t]$, this extension is to reduce the calculation cost while keeping the approximation quality reasonably higher. The calculation cost of the inversion of $\mathbf{RA}[t]$ is the most expensive parts in (6). Therefore, we execute an diagonal approximation of $\mathbf{RA}[t]$ to reduce the calculation costs, which ignores the *off-diagonal* part of them. More specifically, we calculate $\boldsymbol{a}^l[t]$ instead of (6) as presented below.

$$\boldsymbol{a}^l[t] = \boldsymbol{a}^l[t-1] - (\mu[t] - \lambda\mu[t-1])(\mathbf{DA}_l[t])^{-1}\boldsymbol{a}^l[t-1] + \sum_{w=1}^{W}[\mathbf{\Omega}_t]_{l,w}\left([\mathbf{Y}_t]_{l,w} - (\boldsymbol{\alpha}_w[t])^T\boldsymbol{a}^l[t-1]\right) \cdot (\mathbf{DA}_l[t])^{-1}\boldsymbol{\alpha}_w[t].$$

Therein, $\mathbf{DA}_l[t]$ is defined by reformulating (5) as

$$\mathbf{DA}_l[t] = \lambda\mathbf{DA}_l[t-1] + \mathrm{diag}\left(\sum_{w=1}^{W}[\mathbf{\Omega}_t]_{l,w}\boldsymbol{\alpha}_w[t](\boldsymbol{\alpha}_w[t])^T\right) + (\mu[t] - \lambda\mu[t-1])\mathbf{I}_R. \tag{7}$$

The calculation of $(\mathbf{DA}_l[t])^{-1}$ is light because $\mathbf{DA}_l[t]$ is a diagonal matrix. Similarly, $\boldsymbol{c}^w[t]$ can be lightly solvable.

4. Theoretical analysis

4.1 Convergence analysis

This subsection describes a convergence analysis of the proposed OLSTEC. Since the problem at hand is non-convex, the target of the convergence analysis is to provide a convergence to a stationary point of the function.

A convergence analysis of the online subspace tracking on matrices based on the RLS algorithm has been well studied in [33]. A analysis for a tensor case with stochastic gradient is provided in [21]. They are inspired by the work in [34]. Although our case is slightly different from them, the fundamental proof strategy is the same. Therefore, the complete proof is omitted. However, the basic strategy is given below in brief by following [21], [33], [34].

We first define $g_t(\mathbf{A}, \mathbf{C}, \boldsymbol{b})$ as $g_t(\mathbf{A}, \mathbf{C}, \boldsymbol{b}) = \frac{1}{2}\|\boldsymbol{\Omega}_t \circledast \left[\mathbf{Y}_t - \mathbf{A}\mathrm{diag}(\boldsymbol{b}^t)\mathbf{C}^T\right]\|_F^2 + \mu[t]/2\|\boldsymbol{b}^t\|_2^2$, The proposed algorithm amounts to seek the minimization of the following cost with the fixed forgetting parameter $\lambda = 1$ as $F_t(\mathbf{A}, \mathbf{C}) = \sum_{\tau=1}^{t} \mathrm{argmin}_{\boldsymbol{b}} g_\tau(\mathbf{A}, \mathbf{C}, \boldsymbol{b}) + \frac{\bar{\mu}}{2}(\|\mathbf{A}\|_F^2 + \|\mathbf{C}\|_F^2)$. When calculating $\boldsymbol{b}[t] = \mathrm{argmin}_{\boldsymbol{b}}\ g_t(\mathbf{A}[t-1], \mathbf{C}[t-1], \boldsymbol{b})$ as (4) and fixing $\mathbf{C} = \mathbf{C}[t-1]$, we define the surrogate function $\hat{F}_t$ of F_t as $\hat{F}_t(\mathbf{A}) = \sum_{\tau=1}^{t} g_\tau(\mathbf{A}, \mathbf{C}[t-1], \boldsymbol{b}[t]) + \frac{\bar{\mu}}{2}(\|\mathbf{A}\|_F^2 + \|\mathbf{C}[t-1]\|_F^2)$, Hence, we define the t-th summand in the above equation for $t = 1, 2, \ldots$ as $\hat{f}_t(\mathbf{A}) = g_t(\mathbf{A}, \mathbf{C}[t-1], \boldsymbol{b}[t]) + \mu[t]/(2t)(\|\mathbf{A}\|_F^2 + \|\mathbf{C}[t-1]\|_F^2)$. Then, we consider the calculation of $\mathbf{A}[t] = \mathrm{argmin}_{\mathbf{A}} \hat{f}_t(\mathbf{A})$. The minimization problem of $\hat{f}_t(\mathbf{A})$ boils down a smooth convex quadratic optimization problem, and the minimizer $\mathbf{A}[t]$ is the solution of linear equation of $\nabla \hat{f}_t(\mathbf{A}) = \mathbf{0}$ as (6). Alternatively, we consider $\tilde{F}_t(\mathbf{C})$ from $F_t(\mathbf{A}, \mathbf{C})$ with setting $\mathbf{A} = \mathbf{A}[t-1]$ derived above as $\tilde{F}_t(\mathbf{C}) = g_\tau(\mathbf{A}[t-1], \mathbf{C}, \boldsymbol{b}[t]) + \bar{\mu}/2(\|\mathbf{A}[t]\|_F^2 + \|\mathbf{C}\|_F^2)$, Here, we similarly define the t-th summand of the earlier equation for $t = 1, 2, \ldots$ as $\tilde{f}_t(\mathbf{C}) = g_t(\mathbf{A}[t-1], \mathbf{C}, \boldsymbol{b}[t]) + \mu[t]/(2t)(\|\mathbf{A}[t-1]\|_F^2 + \|\mathbf{C}\|_F^2)$. In an analogous manner to $\mathbf{A}$, considering $\mathbf{C}[t] = \mathrm{argmin}_{\mathbf{C}}\ \tilde{f}_t(\mathbf{C})$, we obtain the minimizer $\mathbf{C}[t]$ as the solution of linear equation of $\nabla \tilde{f}_t(\mathbf{C}) = \mathbf{0}$. Now, we provide the convergence result for Algorithm 1 below;

Theorem 4.1. *Consider Algorithm 1 with $\lambda = 1$ supposing that: (a1) $\{\boldsymbol{\Omega}_t\}_{t=1}^{\infty}$ and $\{\mathbf{Y}_t\}_{t=1}^{\infty}$ are independent and identically distributed (i.i.d.) random processes; (a2) $\|\boldsymbol{\Omega}_t \circledast \mathbf{Y}_t\|_\infty$ is uniformly bounded; (a3) $\{\mathbf{A}[t], \mathbf{C}[t]\}_{t=1}^{\infty}$ are in a compact set; (a4) $\lambda_{min}[\hat{F}_t(\mathbf{A})] \geq c1$ for some $c1 > 0$ and $\lambda_{min}[\tilde{F}_t(\mathbf{C})] \geq c2$ for some $c2 > 0$. Then, $\lim_{t\to\infty} F_t(\mathbf{A}[t], \mathbf{C}[t]) = \mathbf{0}$ almost surely (a.s.), i.e., the subspace $\{\mathbf{A}[t], \mathbf{C}[t]\}_{t=1}^{\infty}$ asymptotically approaches the stationary point set of the problem as $t \to \infty$.*

The proof sketch is as follows: We first prove that $\hat{F}_t(\mathbf{A}), \tilde{F}_t(\mathbf{C})$ and $F_t(\mathbf{A}, \mathbf{C})$ are a quasi-martingale sequence, and thus convergent a.s. This can be proven by exploiting the strong convexity assumption (a4) on $\hat{F}_t(\mathbf{A})$ and $\tilde{F}_t(\mathbf{C})$. Next, the cost sequence $\{\hat{F}_t(\mathbf{A}[t]) - F_t(\mathbf{A}[t], \mathbf{C}[t])\} \to 0$ yields the convergence of the gradient $\{\nabla_{\mathbf{A}} \hat{F}_t(\mathbf{A}[t]) - \nabla_{\mathbf{A}} F_t(\mathbf{A}[t], \mathbf{C}[t])\} \to 0$. Similarly, the cost sequence $\{\tilde{F}_t(\mathbf{C}[t]) - F_t(\mathbf{A}[t], \mathbf{C}[t])\} \to 0$ yields the convergence of the gradient $\{\nabla_{\mathbf{C}} \tilde{F}_t(\mathbf{C}[t]) - \nabla_{\mathbf{C}} F_t(\mathbf{A}[t], \mathbf{C}[t])\} \to 0$. This provides the desired claim.

表 1 Computational complexity comparison.

Algorithm	Complexity per iteration
TeCPSGD [21]	$\mathcal{O}(\lvert\boldsymbol{\Omega}_t\rvert R^2)$
RPTucker [22]	$\mathcal{O}(\lvert\boldsymbol{\Omega}_t\rvert R^2 + (L + W + T)R^2)$
SIITA [23]	$\mathcal{O}(\lvert\boldsymbol{\Omega}_t\rvert R^3 + (L^2 + W^2 + T^2)R)$
OLSTEC (Proposed)	$\mathcal{O}(\lvert\boldsymbol{\Omega}_t\rvert R^2 + (L + W)R^3)$
OLSTEC-A (Proposed)	$\mathcal{O}(\lvert\boldsymbol{\Omega}_t\rvert R^2 + (L + W)R)$

4.2 Computational complexity and memory consumption analysis

With respect to computational complexity per iteration, OLSTEC requires $\mathcal{O}(|\boldsymbol{\Omega}_t|R^2 + (L+W)R^3)$ because of $\mathcal{O}(|\boldsymbol{\Omega}_t|R^2)$ for $\boldsymbol{b}[t]$ in (4) and $\mathcal{O}((L+W)R^3)$ for the inversion of $\mathbf{RA}_l$ in (6) and $\mathbf{RC}_w$, respectively. The accelerated OLSTEC (OLSTEC-A) requires $\mathcal{O}(|\boldsymbol{\Omega}_t|R^2 + (L + W)R)$, where the second term is achieved by the diagonal approximation $\mathbf{DA}_l$ of $\mathbf{RA}_l$ such as (7). Meanwhile, TeCPSGD requires $\mathcal{O}(|\boldsymbol{\Omega}_t|R)$ for updating two factor matrices $\mathbf{A}[t]$ and $\mathbf{C}[t]$, and $\mathcal{O}(|\boldsymbol{\Omega}_t|R^2)$ for $\boldsymbol{b}[t]$. Thus, the total complexity per iteration is $\mathcal{O}(|\boldsymbol{\Omega}_t|R^2)$. As for SIITA assuming without side information and all ranks of the multilinear rank are fixed to R, SIITA requires $\mathcal{O}(|\boldsymbol{\Omega}_t|R^3 + (L^2 + W^2 + T^2)R)$ because it maintains its entire factor matrices. Due to the same reason as SIITA, RPTucker requires $\mathcal{O}(|\boldsymbol{\Omega}_t|R^2 + (L + W + T)R^2)$. Even if R is much smaller than $\{L, W, |\boldsymbol{\Omega}_t|\}$, the calculation complexities of T^2R in SIITA and TR^2 in RPTucker become dominant in the all computations when the streaming data size T becomes huge. Consequently, OLSTEC(-A) requires much less computing than SIITA and TeCPSGD does. In addition, while OLSTEC needs higher computations than TeCPSGD does, OLSTEC-A requires the same order complexity as TeCPSGD when $\{L, W\} \ll |\boldsymbol{\Omega}_t|$. The overall results are summarized in Table 1.

As for memory consumption, $\mathcal{O}((L+W)R^2)$ is required in OLSTEC, respectively, for $\mathbf{RA}_l$ and $\mathbf{RC}_w$. OLSTEC-A reduces the memory consumption to $\mathcal{O}((L + W)R)$.

5. Conclusion

We have proposed a new online tensor subspace tracking algorithm, designated as OLSTEC, for a partially observed high-dimensional data stream corrupted by noise. Numerical comparisons will be given at the presentation.

参考文献

[1] Kasai, H.: Online Low-Rank Tensor Subspace Tracking from Incomplete Data by CP Decomposition using Recursive Least Squares, *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2016).

[2] Liu, J., Musialski, P., Wonka, P. and Ye, J.: Tensor Completion for Estimating Missing Values in Visual Data, *IEEE Trans. Pattern Anal. Mach. Intell.*, Vol. 35, No. 1, pp. 208–220 (2013).

[3] Tomioka, R., Hayashi, K. and Kashima, H.: Estimation of low-rank tensors via convex optimization, *arXiv:1010.0789* (2011).

[4] Signoretto, M., Dinh, Q. T., Lathauwer, L. D. and Suykens, J. A.: Learning with tensors: a framework based on convex optimization and spectral regularization, *Mach. Learn.*, Vol. 94, No. 3, pp. 303–351 (2014).

[5] Candès, E. J. and Recht, B.: Exact Matrix Completion via Convex Optimization, *Foundations of Computational Mathematics*, Vol. 9, No. 6, pp. 717–772 (2009).

[6] Filipović, M. and Jukić, A.: Tucker factorization with missing data with application to low-n-rank tensor completion, *Multidim. Syst. Sign. P.* (2013).

[7] Kressner, D., Steinlechner, M. and Vandereycken, B.: Low-rank tensor completion by Riemannian optimization, *BIT Numer. Math.*, Vol. 54, No. 2, pp. 447–468 (2014).

[8] Boumal, N. and Absil, P.-A.: RTRMC : A Riemannian trust-region method for low-rank matrix completion,, *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)* (2011).

[9] Mishra, B., Meyer, G., Bach, F. and Sepulchre, R.: Low-rank optimization with trace norm penalty, *SIAM Journal on Optimization*, Vol. 23, No. 4, pp. 2124–2149 (2013).

[10] Ngo, T. and Saad, Y.: Scaled Gradients on Grassmann Manifolds for Matrix Completion, *NIPS*, pp. 1421–1429 (2012).

[11] Haykin, S.: *Adaptive Filter Theory*, Prentice Hall (2002).

[12] Kolda, T. G. and Bader, B. W.: Tensor Decompositions and Applications, *SIAM Review*, Vol. 51, No. 3, pp. 455–500 (2009).

[13] Yang, B.: Projection approximation subspace tracking, *IEEE Trans. on Signal Processing*, Vol. 43, No. 1, pp. 95–107 (1995).

[14] Balzano, L., Nowak, R. and Recht, B.: Online Identification and Tracking of Subspaces from Highly Incomplete Information, *arXiv:1006.4046* (2010).

[15] Edelman, A., Arias, T. and Smith, S.: The geometry of algorithms with orthogonality constraints, *SIAM J. Matrix Anal. Appl.*, Vol. 20, No. 2, pp. 303–353 (1998).

[16] Absil, P.-A., Mahony, R. and Sepulchre, R.: *Optimization Algorithms on Matrix Manifolds*, Princeton University Press (2008).

[17] He, J. H., Balzano, L. and Szlam, A.: Incremental gradient on the grassmannian for online foreground and background separation in subsampled video, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2012).

[18] Chi, Y., Eldar, Y. C. and Calderbank, R.: PETRELS: Parallel Subspace Estimation and Tracking using Recursive Least Squares from Partial Observations, *IEEE Trans. on Signal Processing*, Vol. 61, No. 23, pp. 5947–5959 (2013).

[19] Nion, D. and Sidiropoulos, N.: Adaptive Algorithms to Track the PARAFAC Decomposition of a Third-Order Tensor, *IEEE Transactions on Signal Processing*, Vol. 57, No. 6, pp. 2299–2310 (2009).

[20] Yu, R., Cheng, D. and Liu, Y.: Accelerated Online Low-Rank Tensor Learning for Multivariate Spatio-Temporal Streams, *International Conference on Machine Learning (ICML)* (2015).

[21] Mardani, M., Mateos, G. and Giannakis, G.: Subspace Learning and Imputation for Streaming Big Data Matrices and Tensors, *IEEE Transactions on Signal Processing*, Vol. 63, No. 10, pp. 266–2677 (2015).

[22] Kasai, H. and Mishra, B.: Low-rank tensor completion: a Riemannian manifold preconditioning approach, *The 33rd International Conference on Machine Learning (ICML)* (2016).

[23] Nimishakavi, M., Mishra, B., Gupta, M. and Talukdar.P.: Inductive Framework for Multi-Aspect Streaming Tensor Completion with Side Information, *arXiv preprint arXiv:1802.06371* (2018).

[24] Dennis, J. E. and Moré, J. J.: A Characterization of Superlinear Convergence and Its Application to Quasi-Newton Methods, *Mathematics of Computation*, Vol. 28, No. 126, pp. 549–560 (1974).

[25] Powell, M.: Some global convergence properties of a variable metric algorithm for minimization without exact line search, *SIAM-AMS Proceedings in Nonlinear Programming*, Vol. IX, pp. 53–72 (1976).

[26] Byrd, R. H., Nocedal, J. and Yuan, Y.-X.: Global Convergence of a Cass of Quasi-Newton Methods on Convex Problems, *SIAM Journal on Numerical Analysis*, Vol. 24, No. 5, pp. 1171–1190 (1987).

[27] Nocedal, J. and S.J., W.: *Numerical Optimization*, Springer, New York, USA (2006).

[28] Duchi, J., Hazan, E. and Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization, *Journal of Machine Learning Research*, Vol. 12, pp. 2121–2159 (2011).

[29] Bordes, A., Bottou, L. and Callinari, P.: SGD-QN: Careful Quasi-Newton Stochastic Gradient Descent, *Journal of Machine Learning Research*, Vol. 10, pp. 1737–1754 (2009).

[30] Schraudolph, N. N., Yu, J. and Gunter, S.: A stochastic quasi-Newton method for online convex optimization, *International Conference on Artificial Intelligence and Statistics (AISTATS)* (2007).

[31] Mokhtari, A. and Ribeiro, A.: Global convergence of online limited memory BFGS, *Journal of Machine Learning Research*, Vol. 16, pp. 3151–3181 (2015).

[32] Byrd, R. H., Hansen, S. L., Nocedal, J. and Singer, Y.: A stochastic quasi-Newton method for large-scale optimization, *SIAM J. Optim.*, Vol. 26, No. 2 (2016).

[33] Mardani, M., Mateos, G. and Giannakis, G.: Dynamic Anomalography: Tracking Network Anomalies Via Sparsity and Low Rank, *IEEE Journal of Selected Topics in Signal Processing*, Vol. 7, No. 1, pp. 50–66 (2013).

[34] Mairal, J., Bach, F., Ponce, J. and Sapiro, G.: Online Learning for Matrix Factorization and Sparse Coding, *JMLR*, Vol. 11, pp. 19–60 (2010).