Federated Multilinear Principal Component Analysis (FMPCA) with Applications in Prognostics

Chengyu Zhou, Yuqi Su and Xiaolei Fang
Edward P. Fitts Department of Industrial and Systems Engineering
North Carolina State University

Abstract

Multilinear principal component analysis (MPCA) is an efficient framework for dimension reduction of high-dimensional tensor data. However, MPCA cannot achieve a satisfying performance in real world for two reasons: 1. Data in real world is often distributed in multiple users which yields small sample size in a single user. 2. The privacy protection awareness and corresponding law establishment make it difficult to share the original data of each user which leads to data silo. To address these challenges, this article develops a federated multilinear principal component (FMPCA) framework. The objective of FMPCA is to obtain the same performance as MPCA while preventing data privacy leakage. The goal is achieved by proposing three methodologies towards the data leakage steps of MPCA. Privacy protection of the proposed FMPCA can be guaranteed because high-dimensional tensor data in each user cannot be shared or recovered to other users. LLS regression is utilized to predict the response based on the derived low-dimensional feature tensor from FMPCA and MPCA. Simulated data and a real-world data set are used to validate the performance of the proposed framework.

Keywords: Dimension reduction, tensor, federated learning, privacy protection

1 Introduction

Tensor is a multidimensional array. With the rapid development of science and technology in the past decades, large amount of tensor observations are routinely collected, processed, and stored. For example, the image stream data are naturally 3D tensor objects where each image consists of the 1st and 2nd mode and time is the 3rd mode. However, a typical tensor object commonly resides in high-dimensional tensor space. The high dimensionality poses significant computational challeges and many classification/regression model performs poorly in high dimensional spaces given a small number of training samples. For example, in a image streams containing 40 images which 30×30 pixels, a tensor-coefficient with 36,000 elements needs to be estimated. Thus, dimensional reduction is usually an indispensable preprocessing step for further data analysis.

MPCA (Lu et al. (2008)) is a well-known methodology to reduce the dimension of high-dimensional tensor data and yield a low-dimensional feature tensor. The framework performs feature extraction by determining a multilinear projection that captures most of the original tensor variation. The solution is derived by decomposing the original problem to multiple subproblems and optimizing each subproblem successively. Compared with previous work (Shakhnarovich and Moghaddam (2005); Lee and Elgammal (2005); Ye et al. (2004); Yang et al. (2004); He et al. (2005)) which apply the representation of vector or matrix, MPCA provides a systematic procedure to determine effective representations of tensor objects.

However, MPCA and some other modern machine learning (ML), deep learning (DL) technologies typically require a huge amount of data to reach a satisfying level of performance. The condition is hard to realize because data in real world is usually distributed among multiple users, leading to small sample size in each user. Besides, the modern society is increasingly aware of the data ownership and data governance is more significant which yields law enactment corresponding to data privacy protection such as the California Consumer Privacy Act (CCPA) of the US, China's Cyber Security Las and the General Provisions of Civil Law, e.g.. All the reasons lead to data silo and put a limit on the application of tremendous methodologies in modern industry.

To address the aforementioned challenge, this article proposed a federated multilinear principal component analysis (FMPCA) framework. It is a federated learning-based methodology (McMahan et al. (2017); Konečný et al. (2016); Shokri and Shmatikov (2015)) which is a collaboratively decentralized privacy-preserving technology aiming to overcome data silo. Federated learning aims to build a joint ML model based on the data located at multiple users without exchanging data samples (Yang et al. (2019)). Figure 1 gives an illustration of federated learning. Each user initially trains their model in a distributed fashion, rather than sends their raw data to the server on the cloud. Next, the server on the cloud communicates with each user and aggregates local updates to incorporate a global model. Finally each user downloads the new global updates from the server on the cloud and updates their local models. The iterative training continues updating until convergence is reached or some stopping criterion is met. Federated learning methods have been deployed in practice by major companies (Bonawitz et al. (2019); Sheller et al. (2018)) and play a critical role in supporting privacy-sensitive aplications where training data are distributed at the edge (Brisimi et al. (2018); Huang et al. (2020)).

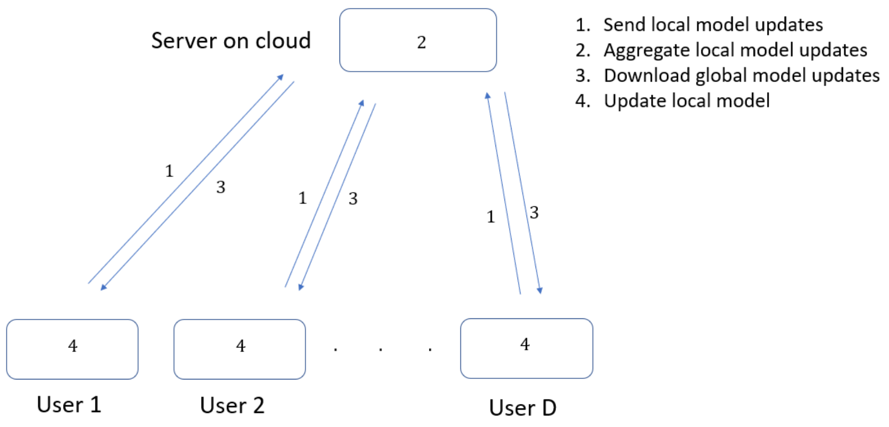

Server on cloud
2
1. Send local model updates
2. Aggregate local model updates
3. Download global model updates
4. Update local model
1
3
1
3
1
3
4
4
4
User 1
User 2
User D

Figure 1: An illustration of Federated Learning.

The objective of FMPCA is to obtain the same performance as multilinear principal compoennt analysis (MPCA) while protecting user privacy and data confidentiality. The motivation is achieved by proposing three algorithms towards the data leakage steps in MPCA - those are - secure centralizaion of input tensor samples, secure derivation of projection matrices in initialization and secure derivation of projection matrices in local optimization. In the 1st algorithm - secure centralization of input tensor samples, each user initially calculates its sample mean and generate perturbations for all the other users. The true sample mean of each user can be masked by adding the perturbations from the other users and then the server on the cloud collects the revised sample mean of each user to calculate the global sample mean. The 2nd and 3rd algorithms - secure derivation of projection matrices in intialization and local optimization are similar. Instead of applying eigen-decomposition to covariance matrix to derive the projection matrices, they conduct singular value decomposition (SVD) to a constructed matrix which concatenates the nth mode matricization of tensor samples from all the users horizontally and prove the left unitary matrix of SVD is equivalent to the projection matrix in MPCA. Next, to preserve user confidentiality, we add user blocks to the constructed matrix and apply an updating algorithm to derive the left unitary matrix securely. Specifically, we conduct SVD to the first user block and update the left unitary matrix based on the information of the following user blocks. The right unitary matrix is not shared between the users in the updating process which helps the raw data in each user cannot be recovered by other users. Data security of the three algorithms can be guaranteed as raw data in each user cannot be shared or recovered by other users and the server on the cloud.

The article is organized as follows. Section 2 introduces our proposed federated multilinear principal component analysis method (FMPCA). In Section 3 and 4, we validate the effectiveness of FMPCA using a simulated dataset and data from a rotating machinery in real world, respectively. Section 5 concludes.

2 The Methodology

Federated multilinear principal component analysis (FMPCA) is a federated learning-based tensor object feature extraction methodology. The objective of FMPCA is to obtain the same performance as multilinear principal component analysis (MPCA) (Lu et al. (2008)) while protecting user privacy and data confidentiality. This is achieved by proposing 3 algorithms corresponding to the privacy leakage steps in MPCA - those are - secure centralization of input tensor samples, secure derivation of projection matrices in initialization and secure derivation of projection matrices in local optimization. Data security is guaranteed because original data in each user cannot be shared or recovered by the central aggregation server and other users.

In Subsection 2.1, we will present some basic tensor notations and definitions. Subsection 2.2 reviews multilinear principal component analysis (MPCA). In Subsection 2.3, we will discuss the challenges of MCPA in privacy-protection. Finally, Subsection 2.4 introduces our proposed federated multilinear principal component analysis (FMPCA).

2.1 Preliminaries

In this section, we will introduce some basic notations and definitions of tensor symbols defined throughout this paper. Vectors are denoted by lowercase boldface letters, e.g., $\boldsymbol{x}$, matrices are denoted by boldface uppercase letters, e.g., $\boldsymbol{X}$, and tensors are denoted by calligraphic letters, e.g., $\mathcal{X}$. The *order* of a tensor is the number of dimensions, also known as *modes* or *ways*. Indices are denoted by lowercase letters and span the range from 1 to the uppercase letter of the index, e.g., $m = 1, 2, \ldots, M$. An Nth-order tensor is denoted as $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, where I_n represents the nth mode of $\mathcal{X}$. The $(i_1, i_2, \ldots, i_N)$th entry of $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is denoted by $x_{i_1,i_2,\ldots,i_n}$. A fiber of $\mathcal{X}$ is a vector defined by fixing every index but one. A matrix column is a mode-1 fiber and a matrix row is a mode-2 fiber. The *vectorization* of $\mathcal{X}$, denoted by $vec(\mathcal{X})$, is obtained by stacking all mode-1 fibers of $\mathcal{X}$. The mode-n *matricization* of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, is a matrix whose columns are mode-n *matricization* of $\mathcal{X}$ in the lexicographical order. The mode-n product of a tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and a matrix $\boldsymbol{U}_n \in \mathbb{R}^{J_n \times I_n}$, denoted by $\mathcal{X} \times_n \boldsymbol{U}_n$, is a tensor whose entry is $(\mathcal{X} \times_n \boldsymbol{U}_n)_{i_1,\ldots,i_{n-1},j_n,i_{n+1},\ldots,i_N} = \Sigma_{I_n=1}^{I_N} x_{i_1,\ldots,i_N} u_{j,i_n}$. The inner product of two tensors $\mathcal{A}, \mathcal{B} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ is denoted by $\langle \mathcal{A}, \mathcal{B} \rangle = \Sigma_{i_1,\ldots,i_N} a_{i_1,\ldots,i_N} b_{i_1,\ldots,i_N}$ and the Frobenius norm of $\mathcal{A}$ is defined as $\|\mathcal{A}\|_F = \sqrt{\langle \mathcal{A}, \mathcal{A} \rangle}$. The *Kronecker product* of two matrices $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ and $\boldsymbol{B} \in \mathbb{R}^{p \times q}$ is an $mp \times nq$ matrix $\boldsymbol{A} \otimes \boldsymbol{B} = [\boldsymbol{a}_1 \otimes \boldsymbol{B} \quad \boldsymbol{a}_2 \otimes \boldsymbol{B} \quad \cdots \quad \boldsymbol{a}_N \otimes \boldsymbol{B}]$. The *Khatri-Rao* product of two matrices whose numbers of columns are the same, $\boldsymbol{A} \in \mathbb{R}^{m \times r}, \boldsymbol{B} \in \mathbb{R}^{p \times r}$, denoted by $\boldsymbol{A} \odot \boldsymbol{B}$, is a $mp \times r$ matrix defined by $[\boldsymbol{a}_1 \otimes \boldsymbol{b}_1 \quad \boldsymbol{a}_2 \otimes \boldsymbol{b}_2 \quad \cdots \quad \boldsymbol{a}_N \otimes \boldsymbol{b}_N]$. If $\boldsymbol{a}$ and $\boldsymbol{b}$ are vectors, then $\boldsymbol{A} \otimes \boldsymbol{B} = \boldsymbol{A} \odot \boldsymbol{B}$.

2.2 Multilinear Principal Component Analysis (MPCA)

Before introducing our proposed federated multilinear principal component analysis (FMPCA), we briefly review the multilinear principal component analysis (MPCA) (Lu et al. (2008)) in this subsection.

Multilinear principal component analysis (MPCA) is feature extraction methodology for tensor objects, such as images and video sequences. The objective is achieved by determining a multilinear projection that captures most of the original tensorial representation. A iterative solution is formulated which proceeds by decomposing the original problem to multiple subproblems. Compared with other state-of-the-art methods, the performance of MPCA-based approach is superior in many applications, such as gait recognition.

We consider a data set consists of M tensor objects $\{\mathcal{X}_m, m = 1, \ldots, M\}$. Each tensor object $\mathcal{X}_m \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ resides in the tensor (multilinear) space $\mathbb{R}^{I_1} \otimes \mathbb{R}^{I_2} \otimes \ldots \otimes \mathbb{R}^{I_N}$, where $\mathbb{R}^{I_1}, \mathbb{R}^{I_2}, \ldots, \mathbb{R}^{I_N}$ are the N vector (linear) spaces. The MPCA objective is to define a multilinear transformaion $\{\tilde{\boldsymbol{U}}^{(n)} \in \mathbb{R}^{I_n \times P_n}, n = 1, \ldots, N\}$ which maps the original tensor space $\mathbb{R}^{I_1} \otimes \mathbb{R}^{I_2} \otimes \ldots \otimes \mathbb{R}^{I_N}$ into a tensor subspace $\mathbb{R}^{P_1} \otimes \mathbb{R}^{P_2} \otimes \ldots \otimes \mathbb{R}^{P_N}$ ($P_n < I_n$ for $n = 1, \ldots, N$): $\mathcal{Y}_m = \mathcal{X}_m \times_1 \tilde{\boldsymbol{U}}^{(1)\top} \times_2 \tilde{\boldsymbol{U}}^{(2)\top} \ldots \times_N \tilde{\boldsymbol{U}}^{(N)\top}, m = 1, \ldots, M$, such that $\{\mathcal{Y}_m \in \mathbb{R}^{P_1} \otimes \mathbb{R}^{P_2} \ldots \otimes \mathbb{R}^{P_N}, m = 1, \ldots, M\}$ can be seen as a feature tensor that resides in the tensor space $\mathbb{R}^{P_1} \otimes \mathbb{R}^{P_2} \ldots \otimes \mathbb{R}^{P_N}$ and captures most of the variations observed in the original tensor objects, assuming that these variations are measured by the total tensor scatter. In other words, the objective of MPCA is to determine the N projection matrices $\{\tilde{\boldsymbol{U}}^{(n)} \in \mathbb{R}^{I_n \times P_n}, n = 1, \ldots, N\}$ that maximize the total tensor scatter $\Psi_{\mathcal{Y}}$:

$$\{\tilde{\boldsymbol{U}}^{(n)}, n = 1, \ldots, N\} = \arg \max_{\tilde{\boldsymbol{U}}^{(1)}, \tilde{\boldsymbol{U}}^{(2)}, \ldots, \tilde{\boldsymbol{U}}^{(N)}} \Psi_{\mathcal{Y}}. \tag{1}$$

where the dimensionality P_n for each mode is assumed to be known or predetermined.

As there is no known optimal solution for the simultaneous optimization of the N projetion matrices, Lu et al. (2008) transfer the simultaneous optimization to N optimization subproblems which can be solved by finding the $\tilde{\boldsymbol{U}}^{(n)}$ that maximizes the scatter in the nth mode vector subspace. This is discussed in Theorem 1 (Lu et al. (2008)):

Theorem 1 *(Lu et al. (2008)) Let $\{\tilde{\boldsymbol{U}}^{(n)}, n = 1, \ldots, N\}$ be the solution to (1). Then, given all the other projection matrices $\tilde{\boldsymbol{U}}^{(1)}, \ldots, \tilde{\boldsymbol{U}}^{(n-1)}, \tilde{\boldsymbol{U}}^{(n+1)}, \ldots, \tilde{\boldsymbol{U}}^{(N)}$, the matrix $\tilde{\boldsymbol{U}}^{(n)}$ consists of the P_n eignevectors corresponding to the largest P_n eigenvalues of the matrix*

$$\boldsymbol{\Phi}^{(n)} = \Sigma_{m=1}^{M} (\boldsymbol{X}_{m(n)} - \bar{\boldsymbol{X}}_{(n)}) \cdot \tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}} \cdot \tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}}^{\top} \cdot (\boldsymbol{X}_{m(n)} - \bar{\boldsymbol{X}}_{(n)})^{\top} \tag{2}$$

where

$$\tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}} = \left(\tilde{\boldsymbol{U}}^{(n+1)} \otimes \tilde{\boldsymbol{U}}^{(n+2)} \otimes \ldots \otimes \tilde{\boldsymbol{U}}^{(N)} \otimes \tilde{\boldsymbol{U}}^{(1)} \otimes \tilde{\boldsymbol{U}}^{(2)} \otimes \ldots \otimes \tilde{\boldsymbol{U}}^{(n-1)}\right).$$

From Theorem 1, the optimization of $\tilde{\boldsymbol{U}}^{(n)}$ depends on the projections in other modes, and thus there is no closed-form solution to (1). Instead, Lu et al. (2008) utilize an iteration procedure to compute the projection matrices one by one with all the others fixed (shown in local optimization of Algorithm 1). The local optimization procedure will be repeated until the result converges or a maximum number K of iterations is reached. MPCA is summarized in Algorithm (1) below.

Algorithm 1: MPCA algorithm

```
1  Input: A set of tensor samples $\{\mathcal{X}_m \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}, m = 1, \ldots, M\}$.
2  Output: Low-dimensional representations $\{\mathcal{Y}_m \in \mathbb{R}^{P_1 \times P_2 \times \ldots \times P_N}, m = 1, \ldots, M\}$ of
     the input tensor samples with maximum variation captured.
3  Algorithm:
4  Step 1 (Preprocessing): Center the input samples as
     $\{\tilde{\mathcal{X}}_m = \mathcal{X}_m - \bar{\mathcal{X}}, m = 1, \ldots, M\}$, where $\bar{\mathcal{X}} = \frac{1}{M}\Sigma_{m=1}^{M}\mathcal{X}_m$ is the sample mean.
5  Step 2 (Initialization): Calculate the eigen-decomposition of
     $\mathbf{\Phi}^{(n)*} = \Sigma_{m=1}^{M}\tilde{\boldsymbol{X}}_{m(n)} \cdot \tilde{\boldsymbol{X}}_{m(n)}^{T}$ and set $\tilde{\boldsymbol{U}}^{(n)}$ to consist of the eigenvectors
     corresponding to the most significant $P_n$ eigenvalues, for $n = 1, \ldots, N$.
6  Step 3 (Local Optimization):
7  • Calculate $\{\tilde{\mathcal{Y}}_m = \tilde{\mathcal{X}}_m \times_1 \tilde{\boldsymbol{U}}^{(1)^\top} \times_2 \tilde{\boldsymbol{U}}^{(2)^\top} \ldots \times_N \tilde{\boldsymbol{U}}^{(N)^\top}, m = 1, 2, \ldots, M\}$.
8  • Calculate $\Psi_{\mathcal{Y}_0} = \Sigma_{m=1}^{M}\|\tilde{\mathcal{Y}}_m\|_F^2$ (the mean $\bar{\tilde{\mathcal{Y}}}$ is all zero since $\tilde{\mathcal{X}}_m$ is centered).
9  for $k$ = 1 : $K$ do
10     for $n$ = 1 : $N$ do
11         * Set the matrix $\tilde{\boldsymbol{U}}^{(n)}$ to consist of the $P_n$ eigenvectors of the matrix $\mathbf{\Phi}^{(n)}$,
             as defined in Equation (2), corresponding to the largest $P_n$ eigenvalues.
12     end
13     Calculate $\{\tilde{\mathcal{Y}}_m, m = 1, \ldots, M\}$ and $\Psi_{\mathcal{Y}_k}$.
14     If $\Psi_{\mathcal{Y}_k} - \Psi_{\mathcal{Y}_{k-1}} \leq \eta$, break and go to Step 4
15 end
16 Step 4 (Projection): The feature tensor after projection is obtained as
     $\{\mathcal{Y}_m = \mathcal{X}_m \times_1 \tilde{\boldsymbol{U}}^{(1)^\top} \times_2 \tilde{\boldsymbol{U}}^{(2)^\top} \ldots \times_N \tilde{\boldsymbol{U}}^{(N)^\top}, m = 1, 2, \ldots, M\}$.
```

2.3 The Challenges of MPCA in Privacy-Protection

In this subsection, we will introduce three steps of MPCA which have privacy leakage issue.

MPCA (Lu et al. (2008)) and many other modern machine learning (ML), deep learning (DL) technologies typically require a huge amount of data to reach a satisfying level of performance. However, it is hard to realize because data are distributed in multiple users in real world which results in small sample size in a single user. Moreover, the modern society is increasingly paying attention to user privacy and data confidentiality which induces law makers to enact new laws ruling how data should be managed and used. Under theses circumstances, the traditional approaches which collect and share the data to one central location are no longer valid nowadays. For the convenience of discussing the issue, we assume that there exists D users, and each user $d, d = 1, \ldots, D$ has M_d high-dimensional tensor samples - $\{\mathcal{X}^d_{m_d},\ m_d = 1, \ldots, M_d\}$. For example, we use $\mathcal{X}^1_3$ to represent the 3rd high-dimensional tensor sample of user 1.

The first privacy-protection challenge of MPCA occurs in its step 1 of Algorithm (1) - centralization of input tensor samples. Specifically, to accomplish the centralization step in real world, Lu et al. (2008) needs to collect the high-dimensional tensor samples from all the D users at one center aggregation server which generates a combined dataset $\{\mathcal{X}^d_{m_d},\ m_d = 1, \ldots, M_d;\ d = 1, \ldots, D\}$, then calculates the sample mean $\bar{\mathcal{X}}$ as:

$$\bar{\mathcal{X}} = \frac{\sum_{d=1}^{D} \sum_{m_d=1}^{M_d} \mathcal{X}^d_{m_d}}{\sum_{d=1}^{D} M_d}. \tag{3}$$

However, privacy protection for distributed data become impediment for the central aggregation server to collect the data from all the D users. Leakage of users' privacy occurs during the collection because each user $d, d = 1, \ldots, D$ exposes its data to the server and the other users.

The second privacy-protection challenge of MPCA exists in step 2 of Algorithm (1) - derivation of projection matrices in initialization. In reality, to derive the projection matrices $\tilde{\boldsymbol{U}}^{(n)}$ in initialization, it is necessary for Lu et al. (2008) to initially aggregate the high-dimensional tensor samples from all the D users which yields a combined dataset $\{\mathcal{X}^d_{m_d},\ m_d = 1, \ldots, M_d;\ d = 1, \ldots, D\}$. Then centralize the combined dataset as $\{\tilde{\mathcal{X}}^d_{m_d} = \mathcal{X}^d_{m_d} - \bar{\mathcal{X}},\ m_d = 1, \ldots, M_d;\ d = 1, \ldots, D\}$, where $\bar{\mathcal{X}}$ is the sample mean of the combined dataset in Equation (3), and unfold the centralized high-dimensional tensor samples of the combined dataset in the nth mode - $\{\tilde{\boldsymbol{X}}^d_{m_d(n)},\ m_d = 1, \ldots, M_d;\ d = 1, \ldots, D\}$, where $\tilde{\boldsymbol{X}}^d_{m_d(n)}$ is the nth mode matricization of $\tilde{\mathcal{X}}^d_{m_d}$. Next, the projection matrix $\tilde{\boldsymbol{U}}^{(n)}$ can be derived by calculating the eigen-decomposition of the covariance matrix below:

$$\boldsymbol{\Phi}^{(n)*} = \sum_{d=1}^{D} \sum_{m_d=1}^{M_d} \tilde{\boldsymbol{X}}^d_{m_d(n)} \cdot \tilde{\boldsymbol{X}}^{d\top}_{m_d(n)}, \tag{4}$$

where $\tilde{\boldsymbol{X}}^{d\top}_{m_d(n)}$ is the transpose of matrix $\tilde{\boldsymbol{X}}^d_{m_d(n)}$. Thus, the data privation violation occurs in the process of generating the combined dataset and the covariance matrix $\mathbf{\Phi}^{(n)*}$ which leaks original data information of each user to the central aggregation server.

Similar to the second privacy-protection challenge of MPCA, the third data leakage location happens in the derivation of projection matrices in local optimization. To derive the projection matrix $\tilde{\boldsymbol{U}}^{(n)}$ in local optimization, it is still essential to generate the combined dataset $\{\tilde{\mathcal{X}}^d_{m_d},\ m_d = 1,\dots,M_d;\ d = 1,\dots,D\}$ which aggregates the centralized high-dimensional tensor samples from all the D users. Next, Lu et al. (2008) conduct eigen-decomposition to the matrix $\mathbf{\Phi}^{(n)}$ which is shown as follows:

$$\mathbf{\Phi}^{(n)} = \Sigma_{d=1}^{D}\Sigma_{m_d=1}^{M_d}(\boldsymbol{X}^d_{m_d(n)} - \bar{\boldsymbol{X}}_{(n)}) \cdot \tilde{\boldsymbol{U}}_{\mathbf{\Phi}^{(n)}} \cdot \tilde{\boldsymbol{U}}^{\top}_{\mathbf{\Phi}^{(n)}} \cdot (\boldsymbol{X}^d_{m_d(n)} - \bar{\boldsymbol{X}}_{(n)})^{\top}, \tag{5}$$

where

$$\tilde{\boldsymbol{U}}_{\mathbf{\Phi}^{(n)}} = \big(\tilde{\boldsymbol{U}}^{(n+1)} \otimes \tilde{\boldsymbol{U}}^{(n+2)} \otimes \ldots \otimes \tilde{\boldsymbol{U}}^{(N)} \otimes \tilde{\boldsymbol{U}}^{(1)} \otimes \tilde{\boldsymbol{U}}^{(2)} \otimes \ldots \otimes \tilde{\boldsymbol{U}}^{(n-1)}\big), \tag{6}$$

$\boldsymbol{X}^d_{m_d(n)}$ is the nth mode matricization of the m_dth centralized high-dimensional tensor sample in the dth user, $\bar{\boldsymbol{X}}_{(n)}$ is the nth mode matricization of the sample mean $\bar{\mathcal{X}}$ of the combined dataset $\{\tilde{\mathcal{X}}^d_{m_d},\ m_d = 1,\dots,M_d;\ d = 1,\dots,D\}$ in Equation (3).

To address these challenges, we will propose three federated learning based methods which prevent MPCA from privacy leakage in Subsection 2.4 - those are - secure centralization of input tensor samples in 2.4.1, secure derivation of projection matrices in initialization in 2.4.2, secure derivation of projection matrices in local optimization in 2.4.3.

2.4 Federated Multilinear Principal Component Analysis (FMPCA)

Multilinear Principal Component Analysis (MPCA), reviewed in Section 2.2, is to collect all the M high-dimensional tensor samples together at one center aggregation server and derive the low-dimeniosional feature tensors on the server using the combined dataset. However, the problem of data silos mentioned in section 1 becomes impediment for the central aggregation server to collect the data from all the D users. Leakage of users' privacy occurs during the collection because each user $d,\ d = 1,\dots,D$ exposes its data to the server and the other users. Thus, it is natural to seek solutions that can ensure user confidentiality while achieving the same or approximate result as the data are combined.

Subsection 2.4 is organized as follows: we initially discuss 3 steps in MPCA which have concern of data privacy violation - those are - input samples centralization in 2.4.1, projection matrices derivation of initialization in 2.4.2, projection matrices derivation of local optimization in 2.4.3. Next, 2.4.4 will introduce our proposed Federated Multilinear Principal Component Analysis (FMPCA) as a whole.

2.4.1 Secure Centralization of Input Tensor Samples

Referring to the step 1 (Preprocessing) of Algorithm 1, Lu et al. (2008) centralize the input tensor samples as $\{\tilde{\mathcal{X}}_m = \mathcal{X}_m - \bar{\mathcal{X}}, m = 1, \dots, M\}$, where M, as discussed in the background of federated learning, is the number of high-dimensional tensor samples of all the D users and $\bar{\mathcal{X}} = \frac{1}{M}\sum_{m=1}^{M} \mathcal{X}_m$ is the sample mean of all the users. When there are multiple users, the sample mean is computed by:

$$\bar{\mathcal{X}} = \frac{\sum_{d=1}^{D}\sum_{m_d=1}^{M_d} \mathcal{X}_{m_d}^{d}}{\sum_{d=1}^{D} M_d}.$$

where M_d is the number of high-dimensional tensor samples of the dth user and $\mathcal{X}_{m_d}^{d}$ denotes the m_dth high-dimensional tensor sample of the dth user.

To accomplish the centralization, it is necessary for the central aggregation server to collect tensor samples from all the users resulting in the violation of data privacy protection in real world.

For the purpose of preventing privacy leakage, we propose a method for secure centralization of input tensor samples. Specifically, for each user $d, d = 1, \dots, D$, we initially calculate its individual sample mean:

$$\bar{\mathcal{X}}_d = \frac{1}{M_d}\sum_{m_d=1}^{M_d} \mathcal{X}_{m_d}^{d}, \quad d = 1, \dots, D,$$

where M_d denotes the number of high-dimensional tensor samples of user d, $\mathcal{X}_{m_d}^{d}$ denotes the m_dth high-dimensional tensor sample of user d, where $m_d = 1, \dots, M_d$ and $\bar{\mathcal{X}}_d$ represents the sample mean of user d. To mask the true sample mean of each user, we generate tensor perturbations for each pair of users as follows:

$$\mathcal{R}_{d,d'} = \mathcal{S}_{d,d'} - \mathcal{S}_{d',d}, \quad d = 1, \dots, D; \quad d' = 1, \dots, D; \quad d \neq d',$$

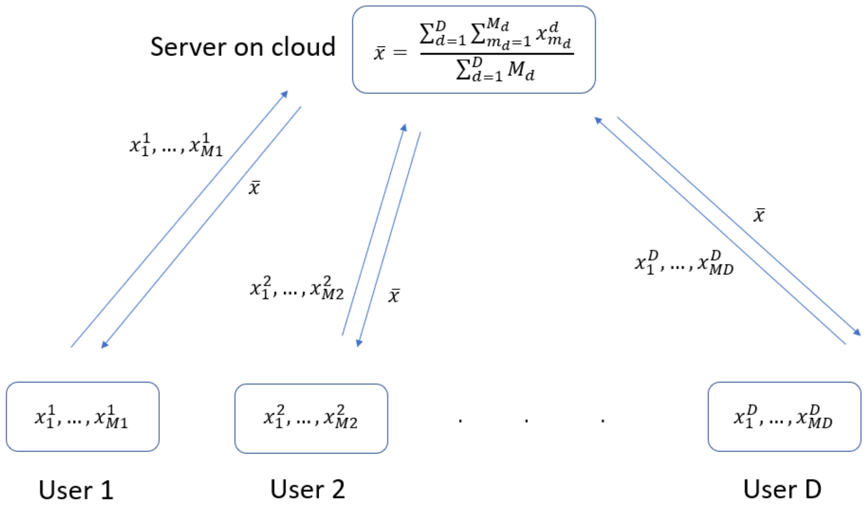
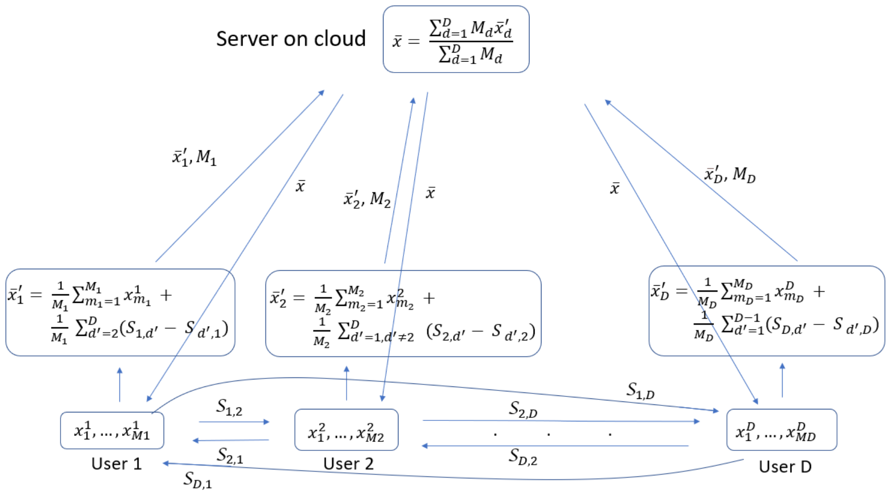


Figure 2: Centralization with privacy leakage.



Server on cloud
$\bar{x} = \frac{\sum_{d=1}^{D} M_d \bar{x}'_d}{\sum_{d=1}^{D} M_d}$
$\bar{x}'_1, M_1$
$\bar{x}$
$\bar{x}'_2, M_2$
$\bar{x}$
$\bar{x}$
$\bar{x}'_D, M_D$
$\bar{x}'_1 = \frac{1}{M_1}\sum_{m_1=1}^{M_1} x_{m_1}^1 + \frac{1}{M_1}\sum_{d'=2}^{D}(S_{1,d'} - S_{d',1})$
$\bar{x}'_2 = \frac{1}{M_2}\sum_{m_2=1}^{M_2} x_{m_2}^2 + \frac{1}{M_2}\sum_{d'=1,d'\neq 2}^{D}(S_{2,d'} - S_{d',2})$
$\bar{x}'_D = \frac{1}{M_D}\sum_{m_D=1}^{M_D} x_{m_D}^D + \frac{1}{M_D}\sum_{d'=1}^{D-1}(S_{D,d'} - S_{d',D})$
$x_1^1, \dots, x_{M1}^1$
$x_1^2, \dots, x_{M2}^2$
$x_1^D, \dots, x_{MD}^D$
$S_{1,2}$
$S_{2,1}$
$S_{2,D}$
$S_{D,2}$
$S_{1,D}$
$S_{D,1}$
User 1
User 2
User D

Figure 3: Centralization without privacy leakage.

where $\mathcal{S}_{d,d'} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ denotes a perturbation to user d' generated by user d and each entry of $\mathcal{S}_{d,d'}$ follows uniform distribution Unif$[0, C)$, C is a known number, $\mathcal{R}_{d,d'}$ represents perturbations for the pair of users d and d'. Then each user add the derived $\mathcal{R}_{d,d'}$ of all the other users to its sample mean:

$$\bar{\mathcal{X}}'_d = \bar{\mathcal{X}}_d + \frac{1}{M_d} \sum_{d'=1, d' \neq d}^{D} \mathcal{R}_{d,d'}, d = 1, \ldots, D$$

where $\bar{\mathcal{X}}'_d$ is the revised sample mean of usr d which adds perturbations. Finally, each user $d, d = 1, \ldots, D$ sends its revised sample mean $\bar{\mathcal{X}}'_d$ and sample size M_d to the server, and the sample mean of all the users can be derived by

$$\bar{\mathcal{X}} = \frac{\sum_{d=1}^{D} M_d \bar{\mathcal{X}}'_d}{\sum_{d=1}^{D} M_d}.$$

where $\bar{\mathcal{X}}$ denotes the sample mean of all the users. After calculating the sample mean of all the users $\bar{\mathcal{X}}$, each user can download it from the server and centralize its tensor samples privately as follows:

$$\tilde{\mathcal{X}}^d_{m_d} = \mathcal{X}^d_{m_d} - \bar{\mathcal{X}},\ m_d = 1, \ldots, M_d;\ d = 1, \ldots, D.$$

where $\tilde{\mathcal{X}}^d_{m_d}$ denotes the m_dth centralized high-dimensional tensor sample of user d.

Algorithm 2 summarizes the method for secure centralization of input tensor samples. The objective of preventing privacy leakage can be achieved because the raw high-dimensional tensor samples as well as the true sample of each user cannot detected by the server and other users. Besides, the correctness of the result in our proposed method can be guaranteed as all the perturbations cancelled out during the summation. The proof for the correctness of the result can be found in the appendix.

Algorithm 2: Secure Centralization algorithm

```
1 Input: A set of D users where each user has $M_d$ high-dimensional tensor samples
    $\{\mathcal{X}^d_{m_d} \in \mathbb{R}^{I_1\times I_2\times\ldots\times I_N}, m_d = 1,\ldots,M_d; d = 1,\ldots,D\}$, $M$ is the number of
    high-dimensional tensor samples of all the users - $M = \sum_{d=1}^{D} M_d$.
2 Output: The sample mean on the server's end $\bar{\mathcal{X}}$.
3 Algorithm:
4 Step 1: Calculate the mean for each user $d$: $\bar{\mathcal{X}}_d = \frac{1}{M_d}\sum_{m_d=1}^{M_d} \mathcal{X}^d_{m_d}, d = 1,\ldots,D$
5 Step 2: Generate input perturbations for each pair of users
    $\mathcal{R}_{d,d'} = \mathcal{S}_{d,d'} - \mathcal{S}_{d',d}\ \ (d = 1,\ldots,D;\ d' = 1,\ldots,D;\ d \neq d')$, where each entry of
    $\mathcal{S}_{d,d'}$ follows the distribuion - Unif $[0, C)$
6 Step 3: For each user $d, d = 1,\ldots,D$, exchange $\mathcal{R}_{d,d'}$ with all the other users and
    calculate $\bar{\mathcal{X}}'_d = \bar{\mathcal{X}}_d + \frac{1}{M_d}\sum_{d'=1,d'\neq d}^{D} \mathcal{R}_{d,d'}$
7 Step 4: Each user sends $\bar{\mathcal{X}}'_d$ and its sample size $M_d$ to the server, and the server
    calculate the global mean $\bar{\mathcal{X}} = \frac{\sum_{d=1}^{D} M_d\bar{\mathcal{X}}'_d}{\sum_{d=1}^{D} M_d}$
8 Step 5: Each user downloads the global mean $\bar{\mathcal{X}}$ from the server, and centralize its
    individual sample as $\{\tilde{\mathcal{X}}^d_{m_d} = \mathcal{X}^d_{m_d} - \bar{\mathcal{X}},\ m_d = 1,\ldots,M_d; d = 1,\ldots,D\}$
```

2.4.2 Secure Derivation of Projection Matrices in Initialization

In 2.4.2, we will focus on the data privacy leakage in derivation of projection matrices in intialization and how to overcome the challenge.

From step 2 (Initialization) of Algorithm 1, MPCA obtains the projection matrices $\{\tilde{\boldsymbol{U}}^{(n)}, n = 1,\ldots,N\}$ by calculating the eigen-decomposition of $\boldsymbol{\Phi}^{(n)*} = \sum_{m=1}^{M} \tilde{\boldsymbol{X}}_{m(n)} \cdot \tilde{\boldsymbol{X}}^{\top}_{m(n)}$, where $\tilde{\boldsymbol{X}}_{m(n)}, n = 1,\ldots,N$ is the mode-n matricization of centralized high-dimensional tensor $\tilde{\mathcal{X}}_m$ and M is the number of tensor samples of all the users combined. When there are multiple users, the covariance matrix $\boldsymbol{\Phi}^{(n)*}$ can be represented by:

$$\boldsymbol{\Phi}^{(n)*} = \sum_{d=1}^{D}\sum_{m_d=1}^{M_d} \tilde{\boldsymbol{X}}^d_{m_d(n)} \cdot \tilde{\boldsymbol{X}}^{d\top}_{m_d(n)}, \tag{7}$$

where D is the number of users, M_d is the number of high-dimensional tensor samples of the dth user, $\tilde{\boldsymbol{X}}^d_{m_d(n)}$ is the nth mode matricization of the m_dth centralized high-dimensional tensor sample in the dth user, and $\tilde{\boldsymbol{X}}^{d\top}_{m_d(n)}$ is the transpose of matrix $\tilde{\boldsymbol{X}}^d_{m_d(n)}$. To achieve the summation step of deriving $\mathbf{\Phi}^{(n)*}$, it is inevitable for each user $d, d = 1, \ldots, D$ to send their individual high dimensional tensor samples to the central aggregation server which would expose user privacy to the server. Figure 4 shows the procedure flow of projection matrices derivation in initialization of MPCA.

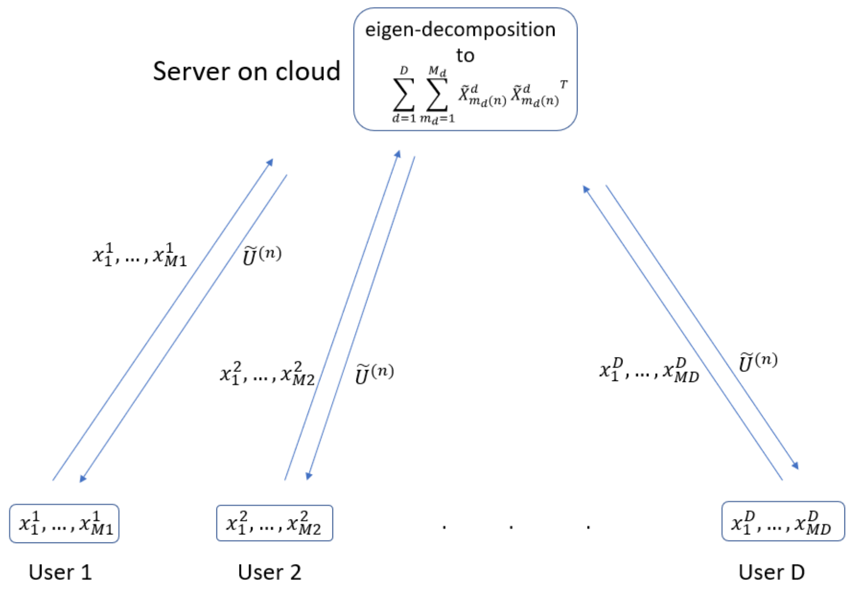

Server on cloud
eigen-decomposition to $\sum_{d=1}^{D}\sum_{m_d=1}^{M_d} \tilde{X}^d_{m_d(n)} \tilde{X}^{d\,T}_{m_d(n)}$
$x_1^1, \ldots, x_{M1}^1$
$\tilde{U}^{(n)}$
$x_1^2, \ldots, x_{M2}^2$
$\tilde{U}^{(n)}$
$x_1^D, \ldots, x_{MD}^D$
$\tilde{U}^{(n)}$
$x_1^1, \ldots, x_{M1}^1$
$x_1^2, \ldots, x_{M2}^2$
$x_1^D, \ldots, x_{MD}^D$
User 1
User 2
User D

Figure 4: Derivation of projection matrices in initialization with privacy leakage.

To address the challenge, we propose a method for secure derivation of projection matrices in initialization. Firstly, due to the fact that calculating eigen-decomposition to $\mathbf{\Phi}^{(n)*}$ in Equation (7) cannot protect data privacy, we consider applying singular value decomposition to a matrix $\boldsymbol{A}_{(n)}$ which is represented as:

$$\boldsymbol{A}_{(n)} = [\tilde{\boldsymbol{X}}^1_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^1_{M_1(n)}, \tilde{\boldsymbol{X}}^2_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^2_{M_2(n)}, \ldots, \tilde{\boldsymbol{X}}^D_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^D_{M_D(n)}]$$

and then show its equivalence with eigen-decomposition in MPCA. Secondly, we develop an alternative approach to securely derive the projection matrix $\tilde{\boldsymbol{U}}^{(n)}$ from $\boldsymbol{A}_{(n)}$ which has the same performance as directly applying SVD to $\boldsymbol{A}_{(n)}$.

To prove the feasibility of using SVD to derive the projection matrix $\tilde{\boldsymbol{U}}^{(n)}$, Proposition 1 is introduced:

Proposition 1 *When there are multiple users in reality, the projection matrix* $\tilde{\boldsymbol{U}}^{(n)}$ *in the initialization of MPCA can be derived by two methods:*

1. Calculate the eigen-decomposition of the covariance matrix $\boldsymbol{\Phi}^{(n)*} = \sum_{d=1}^{D} \sum_{m_d=1}^{M_d} \tilde{\boldsymbol{X}}^{d}_{m_d(n)} \cdot \tilde{\boldsymbol{X}}^{d\top}_{m_d(n)}$ *in Equation (7), and set the projection matrix* $\tilde{\boldsymbol{U}}^{(n)}$ *to comprise all the eigenvectors.*

2. Apply singular value decomposition (SVD) to a matrix $\boldsymbol{A}_{(n)} \in \mathbb{R}^{I_n \times (I_1 \times \ldots \times I_{n-1} \times I_{n+1} \times \ldots \times I_N \times \sum_{d=1}^{D} M_d)}$ *which is represented as:*

$$\boldsymbol{A}_{(n)} = [\tilde{\boldsymbol{X}}^{1}_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^{1}_{M_1(n)}, \tilde{\boldsymbol{X}}^{2}_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^{2}_{M_2(n)}, \ldots, \tilde{\boldsymbol{X}}^{D}_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^{D}_{M_D(n)}]$$

where $\{\tilde{\boldsymbol{X}}^{d}_{m_d(n)},\ m_d = 1, \ldots, M_d;\ d = 1, \ldots, D\}$ *are the nth mode matricization of high-dimensional tensor samples of all the users and concatenated horizontally in* $\boldsymbol{A}_{(n)}$*. And then set the projection matrix* $\tilde{\boldsymbol{U}}^{(n)}$ *to be the left unitary matrix.*

The proof of Proposition 1 can be found in the Appendix.
Based on Proposition 1, the left unitary matrix of applying singular value decomposition (SVD) to $\boldsymbol{A}_{(n)}$, where

$$\boldsymbol{A}_{(n)} = [\tilde{\boldsymbol{X}}^{1}_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^{1}_{M_1(n)}, \tilde{\boldsymbol{X}}^{2}_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^{2}_{M_2(n)}, \ldots, \tilde{\boldsymbol{X}}^{D}_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^{D}_{M_D(n)}],$$

is equivalent to the projection matrix $\tilde{\boldsymbol{U}}^{(n)}$ obtained in the initialization of MPCA.

However, if the projection matrix $\tilde{\boldsymbol{U}}^{(n)}$ is calculated directly by applying singular value decomposition (SVD) to $\boldsymbol{A}_{(n)}$, it is still necessary for the central aggregation server to collect high-dimensional tensor samples from all the D users which yields data privacy leakage. Thus, we develop an alternative method to derive the projection matrix $\boldsymbol{U}^{(n)}$ from $\boldsymbol{A}_{(n)}$ securely. Figure 5 shows the procedure flow of our proposed method. We initially add user blocks to $\boldsymbol{A}_{(n)}$ which is represented as follows:

$$\boldsymbol{A}_{(n)} = [\tilde{\boldsymbol{X}}^{1}_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^{1}_{M_1(n)} \mid \tilde{\boldsymbol{X}}^{2}_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^{2}_{M_2(n)} \mid \ldots \mid \tilde{\boldsymbol{X}}^{D}_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^{D}_{M_D(n)}],$$

and re-express $\boldsymbol{A}_{(n)}$ for simplication:

$$\boldsymbol{A}_{(n)} = [\boldsymbol{A}_{1(n)} | \boldsymbol{A}_{2(n)} | \ldots | \boldsymbol{A}_{D(n)}],$$

where $\boldsymbol{A}_{d(n)} = [\tilde{\boldsymbol{X}}^{d}_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^{d}_{M_d(n)}]$ denotes a matrix which horizontally concatenate the nth mode matricization of high-dimensional tensor samples of the dth user. Instead of directly applying SVD to $\boldsymbol{A}_{(n)}$ in Proposition 1, we conduct SVD to the first user block $\boldsymbol{A}_{1(n)}$ and only share the left unitary matrix $\boldsymbol{U}^{(n)}$, the diagonal matrix with singular values $\boldsymbol{\Sigma}$ to the following users. Next, the left unitary matrix $\boldsymbol{U}^{(n)}$ and the diagonal matrix with singular values $\boldsymbol{\Sigma}$ are updated using the information of other user blocks $\{\boldsymbol{A}_{d(n)}, d = 2, \ldots, D\}$, and then the updating process terminates when all the user blocks are involved in calculation. Finally, the last user D sends the updated left unitary matrix $\boldsymbol{U}^{(n)}$ to the central aggregation server and all the users downloads the left unitary matrix from the central aggregation server. As the right unitary matrix $\boldsymbol{V}$ is not shared between the users, the information of each user cannot be recovered which achieves data privacy protection. To guarantee the performance accuracy of our proposed method, Proposition 2 is introduced.

Proposition 2 *The projection matrix $\boldsymbol{U}^{(n)}$ derived from Algorithm (3) is equivalent to the left unitary matrix of applying singular value decomposition (SVD) to a matrix $\boldsymbol{A}_{(n)}$ which is represented as:*

$$\boldsymbol{A}_{(n)} = [\tilde{\boldsymbol{X}}^1_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^1_{M_1(n)}, \tilde{\boldsymbol{X}}^2_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^2_{M_2(n)}, \ldots, \tilde{\boldsymbol{X}}^D_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^D_{M_D(n)}]$$

where $\{\tilde{\boldsymbol{X}}^d_{m_d(n)},\ m_d = 1, \ldots, M_d,\ d = 1, \ldots, D\}$ are the nth mode matricization of high-dimensional tensor samples of all the users.

The proof of Proposition 2 can be found in the Appendix.

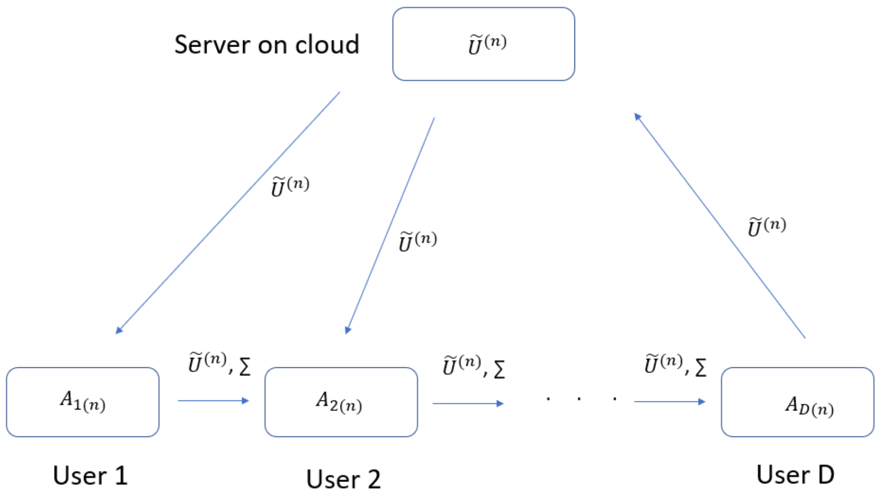


Figure 5: Derivation of projection matrices in initialization without privacy leakage.

Here we explain how to utilize the remaining user blocks $\{\boldsymbol{A}_{d(n)}, d = 2, \ldots, D\}$ to update the left unitary matrix $\tilde{\boldsymbol{U}}^{(n)}$ derived from user 1 in detail. For each column of the

dth user block $\boldsymbol{A}_{d(n)} \in \mathbb{R}^{I_n \times (I_1 \times \ldots \times I_{n-1} \times I_{n+1} \times \ldots \times I_N \times M_d)}$, we initially calculate the $\boldsymbol{e} \in \mathbb{R}^{I_n \times 1}$ by multiplying the covariance matrix $\tilde{\boldsymbol{U}}^{(n)}$ to the column: $\boldsymbol{e} = \boldsymbol{s} - \tilde{\boldsymbol{U}}^{(n)} \tilde{\boldsymbol{U}}^{(n)\top} \boldsymbol{s}$, where $\boldsymbol{s}$ denotes the jth column of $\boldsymbol{A}_{d(n)}, 1 \leq j \leq (I_1 \times \ldots \times I_{n-1} \times I_{n+1} \times \ldots \times I_N \times M_d)$, $\boldsymbol{e}$ is the orthogonal projection of $\boldsymbol{s}$ onto the subspace of orthogonal basis $\tilde{\boldsymbol{U}}^{(n)}$. After deriving the residual $\boldsymbol{e}$, we construct a combined matrix and apply SVD to it:

$$\begin{bmatrix} \Sigma & \boldsymbol{w} \\ \boldsymbol{0} & \|\boldsymbol{e}\| \end{bmatrix} = \hat{\tilde{\boldsymbol{U}}} \hat{\boldsymbol{\Sigma}} \hat{\boldsymbol{V}},$$

where $\boldsymbol{\Sigma} \in \mathbb{R}^{r \times r}$ is the diagonal matrix with positive singular values in the previous iteration, $\boldsymbol{w} = \tilde{\boldsymbol{U}}^{(n)\top} \boldsymbol{s}$ represents the projection of $\boldsymbol{s}$ onto the basis of $\tilde{\boldsymbol{U}}^{(n)}$, $\boldsymbol{0} \in \mathbb{R}^{1 \times r}$ is a $1 \times r$ vector whose elements are all zeros, $\| \cdot \|$ denotes the L2 norm, and $\hat{\tilde{\boldsymbol{U}}} \in \mathbb{R}^{(r+1) \times (r+1)}, \hat{\boldsymbol{\Sigma}} \in \mathbb{R}^{(r+1) \times (r+1)}, \hat{\boldsymbol{V}} \in \mathbb{R}^{(r+1) \times (r+1)}$ are the left unitary matrix, diagonal matrix with positive singular values and right unitary matrix of the constructed matrix, respectively. Then the left singular unitary matrix $\tilde{\boldsymbol{U}}^{(n)}$ and diagonal matrix $\boldsymbol{\Sigma}$ can be updated as follows: $\tilde{\boldsymbol{U}}^{(n)} = \begin{bmatrix} \tilde{\boldsymbol{U}}^{(n)} & \frac{\boldsymbol{e}}{\|\boldsymbol{e}\|} \end{bmatrix} \hat{\tilde{\boldsymbol{U}}}$, $\boldsymbol{\Sigma} = \hat{\boldsymbol{\Sigma}}$, where $\tilde{\boldsymbol{U}}^{(n)} \in \mathbb{R}^{I_n \times (r+1)}$ and $\boldsymbol{\Sigma} \in \mathbb{R}^{(r+1) \times (r+1)}$ are the updated left unitary matrix and the updated diagonal matrix with positive singular values separately. Finally, an automatic truncation is implemented - that is - if $\|\boldsymbol{e}\| \leq \epsilon$, where ϵ is a small number, the updated $\tilde{\boldsymbol{U}}^{(n)}$ and $\boldsymbol{\Sigma}$ will not increase their rank which is represented as follows: $\tilde{\boldsymbol{U}}^{(n)} = \tilde{\boldsymbol{U}}^{(n)}_{:,1:r}$, $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_{1:r,1:r}$, where " : " denotes all the indices in a specific dimension, "$1:r$" represents the indices from 1 to r. Otherwise, the current rank increases by 1.

Algorithm (3) summarizes the method for secure derivation of projection matrices in initiliazation. Due to the fact that the high-dimensional tensor samples in each user cannot be shared or recovered to other users and the centralization aggregation server, data privacy protection of our proposed method can be guaranteed.

Algorithm 3: Secure derivation of projection matrices in initialization

```
Input: A set of D users where each user has centralized high-dimensional tensor samples
    $\{\tilde{\mathcal{X}}^d_{m_d} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N},\ m_d = 1, \ldots, M_d;\ d = 1, \ldots, D\}$.
Output: Projection matrix in the nth mode $\tilde{\boldsymbol{U}}^{(n)} \in \mathbb{R}^{I_n \times P_n}$.
Preprocessing: For each user $d, d = 1, \ldots, D$, unfold its high-dimensional tensor samples
    in the $n$th mode - $\{\tilde{\boldsymbol{X}}^d_{m_d(n)} \in \mathbb{R}^{I_n \times (I_1 \times \ldots \times I_{n-1} \times I_{n+1} \times \ldots \times I_N)},\ m_d = 1, \ldots, M_d\}$ and then
    concatenate the unfolded tensor samples horizontally - $[\tilde{\boldsymbol{X}}^d_{1(n)}, \tilde{\boldsymbol{X}}^d_{2(n)}, \ldots, \tilde{\boldsymbol{X}}^d_{M_d(n)}]$ which is
    denoted as $\boldsymbol{A}_{d(n)} \in \mathbb{R}^{I_n \times (I_1 \times \ldots \times I_{n-1} \times I_{n+1} \times \ldots \times I_N \times M_d)}$
Initialization: Set $\varepsilon = 1 \times e^{-6}$
User 1: $\tilde{\boldsymbol{A}}_{1(n)} = \tilde{\boldsymbol{U}}^{(n)} \boldsymbol{\Sigma} \boldsymbol{V}$, $\tilde{\boldsymbol{U}}^{(n)}$ and $\boldsymbol{\Sigma}$ are shared with user 2.
for User $d = 2 : D$ do
    for Column $j = 1 : (I_1 \times \ldots \times I_{n-1} \times I_{n+1} \times \ldots \times I_N \times M_d)$ do
        $\boldsymbol{s} := \tilde{\boldsymbol{A}}_{d(n)}(:, j)$   % the $j$ th column of $\tilde{\boldsymbol{A}}_{d(n)}$
        $\boldsymbol{w} := \tilde{\boldsymbol{U}}^{(n)\top} \boldsymbol{s}$
        $\boldsymbol{p} := \tilde{\boldsymbol{U}}^{(n)} \boldsymbol{w}$
        $\boldsymbol{e} := \boldsymbol{s} - \boldsymbol{p}$
        $\begin{bmatrix} \boldsymbol{\Sigma} & \boldsymbol{w} \\ \boldsymbol{0} & \| \boldsymbol{e} \| \end{bmatrix} = \hat{\tilde{\boldsymbol{U}}} \hat{\boldsymbol{\Sigma}} \hat{V}^\top$
        $\tilde{\boldsymbol{U}}^{(n)} := \begin{bmatrix} \tilde{\boldsymbol{U}}^{(n)} & \frac{\boldsymbol{e}}{\|\boldsymbol{e}\|} \end{bmatrix} \hat{\tilde{\boldsymbol{U}}}$
        $\boldsymbol{\Sigma} := \hat{\boldsymbol{\Sigma}}$
        if $\| \boldsymbol{e} \| \leq \varepsilon$ then
            $\tilde{\boldsymbol{U}}^{(n)} := \tilde{\boldsymbol{U}}^{(n)}_{1:I_n, 1:r}$   % delete the last column
            $\boldsymbol{\Sigma} := \boldsymbol{\Sigma}_{1:r, 1:r}$   % delete the last row and the last column
        else
            $r := r + 1$
        end
    end
end
Set the projection matrix $\tilde{\boldsymbol{U}}^{(n)} \in \mathbb{R}^{I_n \times r}$ only contain the first $P_n$ columns and then
    generate $\tilde{\boldsymbol{U}}^{(n)} \in \mathbb{R}^{I_n \times P_n}$
```

2.4.3 Secure Derivation of Projection Matrices in Local Optimization

Subsection 2.4.3 discusses the data privacy leakage in derivation of projection matrices in local optimization and introduces a method to overcome the challenge.

In the step 3 (Local Optimization) of Algorithm 1, to derive the projection matrix $\{\tilde{\boldsymbol{U}}^{(n)}, n = 1, \ldots, N\}$, Lu et al. (2008) conduct eigen-decomposition to the matrix $\boldsymbol{\Phi}^{(n)}$ which is shown as follows:

$$\boldsymbol{\Phi}^{(n)} = \Sigma_{m=1}^{M}(\boldsymbol{X}_{m(n)} - \bar{\boldsymbol{X}}_{(n)}) \cdot \tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}} \cdot \tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}}^{\top} \cdot (\boldsymbol{X}_{m(n)} - \bar{\boldsymbol{X}}_{(n)})^{\top} \tag{8}$$

where

$$\tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}} = \big(\tilde{\boldsymbol{U}}^{(n+1)} \otimes \tilde{\boldsymbol{U}}^{(n+2)} \otimes \ldots \otimes \tilde{\boldsymbol{U}}^{(N)} \otimes \tilde{\boldsymbol{U}}^{(1)} \otimes \tilde{\boldsymbol{U}}^{(2)} \otimes \ldots \otimes \tilde{\boldsymbol{U}}^{(n-1)}\big), \tag{9}$$

$\boldsymbol{X}_{m(n)}$ is the nth mode matricization of the mth high-dimensioanl tensor, $\bar{\boldsymbol{X}}_{(n)}$ is the nth mode matricization of the sample mean of high-dimensional tensor samples. When there are multiple users, the matrix $\boldsymbol{\Phi}^{(n)}$ can be expressed as:

$$\boldsymbol{\Phi}^{(n)} = \Sigma_{d=1}^{D}\Sigma_{m_d=1}^{M_d}(\boldsymbol{X}_{m_d(n)}^{d} - \bar{\boldsymbol{X}}_{(n)}) \cdot \tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}} \cdot \tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}}^{\top} \cdot (\boldsymbol{X}_{m_d(n)}^{d} - \bar{\boldsymbol{X}}_{(n)})^{\top}, \tag{10}$$

where

$$\tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}} = \big(\tilde{\boldsymbol{U}}^{(n+1)} \otimes \tilde{\boldsymbol{U}}^{(n+2)} \otimes \ldots \otimes \tilde{\boldsymbol{U}}^{(N)} \otimes \tilde{\boldsymbol{U}}^{(1)} \otimes \tilde{\boldsymbol{U}}^{(2)} \otimes \ldots \otimes \tilde{\boldsymbol{U}}^{(n-1)}\big),$$

$\boldsymbol{X}_{m_d(n)}^{d}$ is the nth mode matricization of the m_dth centralized high-dimensional tensor sample in the dth user, $\bar{\boldsymbol{X}}_{(n)}$ is the nth mode matricization of the sample mean of the combined dataset $\{\tilde{\mathcal{X}}_{m_d}^{d},\ m_d = 1, \ldots, M_d;\ d = 1, \ldots, D\}$. For generating the covariance matrix $\boldsymbol{\Phi}^{(n)}$, it is necessary for the central aggregation server to collect the high-dimensional tensor samples from all the D users which implies data privacy violation. Figure 6 shows the procedure flow of projection matrices derivation in local optimization of MPCA.

To overcome the challenge, we propose a method for secure derivation of projection matrices in local optimization. Firstly, we prove the equivalence of deriving the projection matrix $\tilde{\boldsymbol{U}}^{(n)}$ in local optimization by applying SVD to a matrix $\boldsymbol{A}_{(n)}^{\boldsymbol{\Phi}} \in \mathbb{R}^{I_n \times (P_1 \times \ldots \times P_{n-1} \times P_{n+1} \times \ldots \times P_N \times \sum_{d=1}^{D} M_d)}$ which is represented as:

$$\boldsymbol{A}_{(n)}^{\boldsymbol{\Phi}} = [\tilde{\boldsymbol{X}}_{1(n)}^{1\boldsymbol{\Phi}}, \ldots, \tilde{\boldsymbol{X}}_{M_1(n)}^{1\boldsymbol{\Phi}}, \tilde{\boldsymbol{X}}_{1(n)}^{2\boldsymbol{\Phi}}, \ldots, \tilde{\boldsymbol{X}}_{M_2(n)}^{2\boldsymbol{\Phi}}, \ldots, \tilde{\boldsymbol{X}}_{1(n)}^{D\boldsymbol{\Phi}}, \ldots, \tilde{\boldsymbol{X}}_{M_D(n)}^{D\boldsymbol{\Phi}}] \tag{11}$$

where $\{\tilde{\boldsymbol{X}}_{m_d(n)}^{d\boldsymbol{\Phi}} = \tilde{\boldsymbol{X}}_{m_d(n)}^{d}\tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}},\ m_d = 1, \ldots, M_d;\ d = 1, \ldots, D\}$, $\tilde{\boldsymbol{X}}_{m_d(n)}^{d}$ is the nth mode matricization of the m_dth high-dimensional tensor sample in user d, and $\tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}}$ is calculated

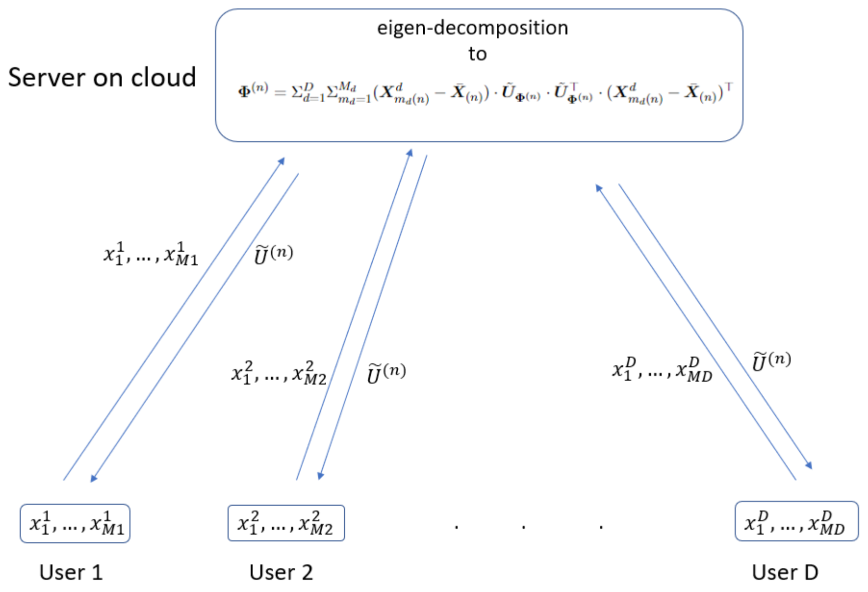

Server on cloud
eigen-decomposition to
$\Phi^{(n)} = \Sigma_{d=1}^{D}\Sigma_{m_d=1}^{M_d}(X_{m_d(n)}^d - \bar{X}_{(n)}) \cdot \tilde{U}_{\Phi^{(n)}} \cdot \tilde{U}_{\Phi^{(n)}}^\top \cdot (X_{m_d(n)}^d - \bar{X}_{(n)})^\top$
$x_1^1, \ldots, x_{M1}^1$
$\tilde{U}^{(n)}$
$x_1^2, \ldots, x_{M2}^2$
$\tilde{U}^{(n)}$
$x_1^D, \ldots, x_{MD}^D$
$\tilde{U}^{(n)}$
$x_1^1, \ldots, x_{M1}^1$
User 1
$x_1^2, \ldots, x_{M2}^2$
User 2
$x_1^D, \ldots, x_{MD}^D$
User D

Figure 6: Derivation of projection matrices in local optimization with privacy leakage.

in Equation (9). Secondly, same as the derivation of projection matrix in initialization, we develop an alternative method to securely derive the projection matrix $\tilde{\boldsymbol{U}}^{(n)}$ from $\boldsymbol{A}^{\boldsymbol{\Phi}}_{(n)}$ which has the same performance as directly applying SVD to $\boldsymbol{A}^{\boldsymbol{\Phi}}_{(n)}$.

To show that the projection matrix $\tilde{\boldsymbol{U}}^{(n)}$ in local optimization of MPCA can be derived by applying SVD to the matrix $\boldsymbol{A}^{\boldsymbol{\Phi}}_{(n)}$ in Equation (11), Proposition 3 is introduced:

Proposition 3 *When there are multiple users in reality, the projection matrix $\tilde{\boldsymbol{U}}^{(n)}$ in the local optimization of MPCA can be derived by two methods:*

1. Calculate the eigen-decomposition of the covariance matrix $\boldsymbol{\Phi}^{(n)} = \Sigma_{d=1}^{D}\Sigma_{m_d=1}^{M_d}(\boldsymbol{X}^d_{m_d(n)} - \bar{\boldsymbol{X}}_{(n)}) \cdot \tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}} \cdot \tilde{\boldsymbol{U}}^\top_{\boldsymbol{\Phi}^{(n)}} \cdot (\boldsymbol{X}^d_{m_d(n)} - \bar{\boldsymbol{X}}_{(n)})^\top$ in Equation (10), and set the projection matrix $\tilde{\boldsymbol{U}}^{(n)}$ to comprise all the eigenvectors.

2. Apply singular value decomposition (SVD) to a matrix $\boldsymbol{A}^{\boldsymbol{\Phi}}_{(n)} \in \mathbb{R}^{I_n \times (P_1 \times \ldots \times P_{n-1} \times P_{n+1} \times \ldots \times P_N \times \sum_{d=1}^{D} M_d)}$ which is represented as:

$$\boldsymbol{A}^{\boldsymbol{\Phi}}_{(n)} = [\tilde{\boldsymbol{X}}^{1\boldsymbol{\Phi}}_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^{1\boldsymbol{\Phi}}_{M_1(n)}, \tilde{\boldsymbol{X}}^{2\boldsymbol{\Phi}}_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^{2\boldsymbol{\Phi}}_{M_2(n)}, \ldots, \tilde{\boldsymbol{X}}^{D\boldsymbol{\Phi}}_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^{D\boldsymbol{\Phi}}_{M_D(n)}],$$

where $\{\tilde{\boldsymbol{X}}^{d\boldsymbol{\Phi}}_{m_d(n)} = \tilde{\boldsymbol{X}}^d_{m_d(n)}\tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}},\ m_d = 1, \ldots, M_d;\ d = 1, \ldots, D\}$ are concatenated horizontally in $\boldsymbol{A}^{\boldsymbol{\Phi}}_{(n)}$, $\tilde{\boldsymbol{X}}^d_{m_d(n)}$ is the nth mode matricization of the m_dth high-dimensional tensor *sample in user d, and $\tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}}$ is calculated in Equation (9). And then set the projection matrix $\tilde{\boldsymbol{U}}^{(n)}$ to be the left unitary matrix.*

Proposition 3 can be found in the Appendix.

Based on Proposition 3, the left unitary matrix of applying singular value decomposition (SVD) to $\boldsymbol{A}^{\boldsymbol{\Phi}}_{(n)}$ in Equation (11) is equivalent to the projection matrix $\tilde{\boldsymbol{U}}^{(n)}$ obtained in the local optimization of MPCA.

However, the data privacy leakage still occurs if we directly conducting SVD to $\boldsymbol{A}^{\boldsymbol{\Phi}}_{(n)}$ in Equation (11) because it is necessary for all the D users to share their high-dimensional tensor samples to the central aggregation server. Therefore, we develop an alternative method to derive the projection matrix $\boldsymbol{U}^{(n)}$ from $\boldsymbol{A}^{\boldsymbol{\Phi}}_{(n)}$ in Equation (11) securely. Figure 7 shows the procedure flow of our proposed method whose steps are similar to the secure derivation of projection matrix in initialization in Subsection 2.4.3. We initially transform $\boldsymbol{A}^{\boldsymbol{\Phi}}_{(n)}$ by adding user blocks:

$$\boldsymbol{A}^{\boldsymbol{\Phi}}_{(n)} = [\boldsymbol{A}^{\boldsymbol{\Phi}}_{1(n)}|\boldsymbol{A}^{\boldsymbol{\Phi}}_{2(n)}|\dots|\boldsymbol{A}^{\boldsymbol{\Phi}}_{D(n)}],$$

where $\boldsymbol{A}^{\boldsymbol{\Phi}}_{d(n)} = [\tilde{\boldsymbol{X}}^{d\boldsymbol{\Phi}}_{1(n)}, \dots, \tilde{\boldsymbol{X}}^{d\boldsymbol{\Phi}}_{M_d(n)}]$, $\{\tilde{\boldsymbol{X}}^{d\boldsymbol{\Phi}}_{m_d(n)} = \tilde{\boldsymbol{X}}^{d}_{m_d(n)}\tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}},\ m_d = 1, \dots, M_d;\ d = 1, \dots, D\}$, $\tilde{\boldsymbol{X}}^{d}_{m_d(n)}$ is the nth mode matricization of the m_dth high-dimensional tensor sample in user d, and $\tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}}$ is calculated in Equation (9). Then we conduct SVD to the first user block $\boldsymbol{A}^{\boldsymbol{\Phi}}_{1(n)}$ and only share the left unitary matrix $\boldsymbol{U}^{(n)}$, the diagonal matrix with singular values $\boldsymbol{\Sigma}$ to the following users. Next, the remaining user blocks $\{\boldsymbol{A}^{\boldsymbol{\Phi}}_{d(n)}, d = 2, \dots, D\}$ are utilized to update the left unitary matrix $\boldsymbol{U}^{(n)}$ and the diagonal matrix with singular values Σ. Finally, the central aggregation server collects the updated projection matrix $\tilde{\boldsymbol{U}}^{(n)}$ from the last user D, and then all the other users download the updated projection matrix $\tilde{\boldsymbol{U}}^{(n)}$ from the server. Privacy protection of the above method is guaranteed because the original data in each user cannot be obtained by other users as well as the central aggregation server. To demonstrate the performance accuracy of our proposed method, Proposition 4 is introduced.

Proposition 4 *The projection matrix $\boldsymbol{U}^{(n)}$ derived from Algorithm (4) is equivalent to the left unitary matrix of applying singular value decomposition (SVD) to a matrix $\boldsymbol{A}^{\boldsymbol{\Phi}}_{(n)}$ in Equation (11).*

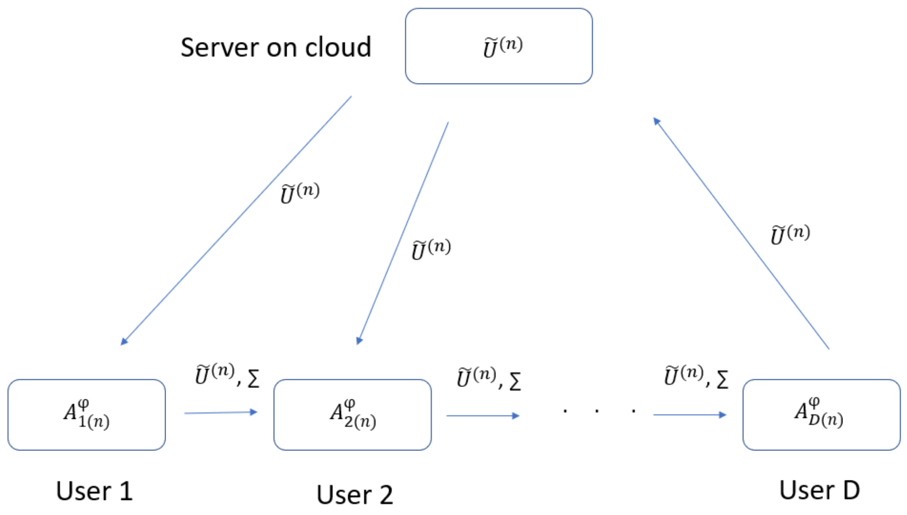


Figure 7: Derivation of projection matrices in local optimization without privacy leakage.

The proof of Proposition 4 can be found in the Appendix.

Algorithm 4 summarizes the method for secure derivation of projection matrices in local optimization of MPCA. As the updating process in Algorithm (4) are the same as the secure derivation of projection matrix in initialization in Subsection 2.4.2, we will not introduce it in detail. The goal of preventing privacy leakage would be realized by our proposed method because the high-dimensional tensor samples in each user cannot be shared or recovered to other users and the centralization aggregation server.

Algorithm 4: Secure derivation of projection matrices in local optimization

```
1  Input: A set of D users where each user has centralized high-dimensional tensor samples
     $\{\tilde{\mathcal{X}}^d_{m_d} \in \mathbb{R}^{I_1\times I_2\times\ldots\times I_N},\ m_d=1,\ldots,M_d;\ d=1,\ldots,D\}$, all the projection matrices except
     the $n$th mode $\{\tilde{\boldsymbol{U}}^{(1)},\ldots,\tilde{\boldsymbol{U}}^{(n-1)},\tilde{\boldsymbol{U}}^{(n+1)},\ldots,\tilde{\boldsymbol{U}}^{(N)}\}$
2  Output: Projection matrix in the $n$th mode $\tilde{\boldsymbol{U}}^{(n)} \in \mathbb{R}^{I_n\times P_n}$.
3  Preprocessing: The server calculates $\tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}} = \tilde{\boldsymbol{U}}^{(n+1)} \otimes\ldots\otimes \tilde{\boldsymbol{U}}^{(N)} \otimes \tilde{\boldsymbol{U}}^{(1)} \otimes\ldots\otimes \tilde{\boldsymbol{U}}^{(n-1)}$.
     For each user $d,\ d=1,\ldots,D$, downloads $\tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}}$ from the server and unfold its centralized
     high-dimensional tensor in the $n$th mode which generates $\{\tilde{\boldsymbol{X}}^d_{m_d(n)}, m_d=1,\ldots,M_d\}$.
     Then each user calculates $\{\tilde{\boldsymbol{X}}^{d\boldsymbol{\Phi}}_{m_d(n)} = \tilde{\boldsymbol{X}}^d_{m_d(n)}\tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}},\ m_d=1,\ldots,M_d\}$ and concatenate
     $\{\tilde{\boldsymbol{X}}^{d\boldsymbol{\Phi}}_{m(n)},\ m_d=1,\ldots,M_d\}$ horizontally - $[\tilde{\boldsymbol{X}}^{d\boldsymbol{\Phi}}_{1(n)}, \tilde{\boldsymbol{X}}^{d\boldsymbol{\Phi}}_{2(n)},\ldots,\tilde{\boldsymbol{X}}^{d\boldsymbol{\Phi}}_{M_d(n)}]$ which is denoted as
     $\boldsymbol{A}^{\boldsymbol{\Phi}}_{d(n)} \in \mathbb{R}^{P_1\times\ldots\times P_{n-1}\times I_n\times P_{n+1}\times\ldots\times P_N\times M_d}$.
4  Initialization: Set $\varepsilon = 1\times e^{-6}$
5  User 1: $\tilde{\boldsymbol{A}}^{\boldsymbol{\Phi}}_{d(n)} = \tilde{\boldsymbol{U}}^{(n)}\boldsymbol{\Sigma}\tilde{\boldsymbol{V}},\ \tilde{\boldsymbol{U}}^{(n)}$ and $\boldsymbol{\Sigma}$ are shared with user 2.
6  for User $d = 2:D$ do
7      for Column $j = 1:(P_1\times\ldots\times P_{n-1}\times P_{n+1}\times\ldots P_N\times M_d)$ do
8          $\boldsymbol{s} := \tilde{\boldsymbol{A}}^{\Phi}_{d(n)}(:,j)$  % the $j$th column of $\tilde{\boldsymbol{A}}^{\boldsymbol{\Phi}}_{d(n)}$
9          $\boldsymbol{w} := \tilde{\boldsymbol{U}}^{(n)\top}\boldsymbol{s}$
10         $\boldsymbol{p} := \tilde{\boldsymbol{U}}^{(n)}\boldsymbol{w}$
11         $\boldsymbol{e} := \boldsymbol{s} - \boldsymbol{p}$
12         $\begin{bmatrix}\boldsymbol{\Sigma} & \boldsymbol{w}\\ \boldsymbol{0} & \|\boldsymbol{e}\|\end{bmatrix} = \hat{\tilde{\boldsymbol{U}}}\hat{\boldsymbol{\Sigma}}\hat{\tilde{V}}^\top$
13         $\tilde{\boldsymbol{U}}^{(n)} = \begin{bmatrix}\tilde{\boldsymbol{U}}^{(n)} & \frac{\boldsymbol{e}}{\|\boldsymbol{e}\|}\end{bmatrix}\hat{\tilde{\boldsymbol{U}}}$
14         $\boldsymbol{\Sigma} := \hat{\boldsymbol{\Sigma}}$
15         if $\|\boldsymbol{e}\| \le \varepsilon$ then
16             $\tilde{\boldsymbol{U}}^{(n)} := \tilde{\boldsymbol{U}}^{(n)}_{1:I_n,1:r}$  % delete the last column
17             $\boldsymbol{\Sigma} := \boldsymbol{\Sigma}_{1:r,1:r}$  % delete both the last row and last column
18         else
19             $r := r+1$
20         end
21     end
22 end
23 Set the projection matrix $\tilde{\boldsymbol{U}}^{(n)} \in \mathbb{R}^{I_n\times d}$ only contain the first $P_n$ columns and generate
     $\tilde{\boldsymbol{U}}^{(n)} \in \mathbb{R}^{I_n\times P_n}$.
```

2.4.4 Overview of Federated Multilinear Principal Component Analysis (FMPCA)

In this subsection, our proposed federated multilinear principal component analysis (FMPCA) is introduced as a whole.

To overcome the three privacy-protection challenges of MPCA discussed in Subsection 2.3, FMPCA revises MPCA by three proposed methods - those are - secure centralization of input tensor samples in 2.4.1, secure derivation of projection matrices in initialization in 2.4.2 and secure derivation of projection matrices in local optimization in 2.4.3.

Algorithm (5) summarizes the whole procedures of FMPCA. Given a set of D users where each user has M_d high-dimensional tensor samples $\{\mathcal{X}^d_{m_d} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}, m_d = 1, \ldots, M_d; d = 1, \ldots, D\}$, the server initially applies the secure aggregation in algorithm (2) to derive the mean tensor $\bar{\mathcal{X}}$ of all the users. Then each user $d,\ d = 1, \ldots, D$ downloads $\bar{\mathcal{X}}$ from the server and centralizes its tensor samples as $\{\tilde{\mathcal{X}}^d_{m_d} = \mathcal{X}^d_{m_d} - \bar{\mathcal{X}},\ m_d = 1, \ldots, M_d;\ d = 1, \ldots, D\}$. After centralization, we initialize the projection matrices $\{\tilde{\boldsymbol{U}}^{(n)} \in \mathbb{R}^{I_n \times P_n}, n = 1, \ldots, N\}$ based on Algorithm (3) which is discussed in 2.4.2 and send the projection matrices to the central server.

To map the original tensor space $\mathbb{R}^{I_1} \otimes \mathbb{R}^{I_2} \otimes \ldots \otimes \mathbb{R}^{I_N}$ into a tensor subspace $\mathbb{R}^{P_1} \otimes \mathbb{R}^{P_2} \otimes \ldots \otimes \mathbb{R}^{P_N}$, each user downloads the initialized projection matrices $\{\tilde{\boldsymbol{U}}^{(n)} \in \mathbb{R}^{I_n \times P_n}, n = 1, \ldots, N\}$ from the server and calculates its low-dimensional tensor representation privately:

$$\tilde{\mathcal{Y}}^d_{m_d} = \tilde{\mathcal{X}}^d_{m_d} \times_1 \tilde{\boldsymbol{U}}^{(1)\top} \times_2 \tilde{\boldsymbol{U}}^{(2)\top} \ldots \times_N \tilde{\boldsymbol{U}}^{(N)\top},\ m_d = 1, \ldots, M_d;\ d = 1, \ldots, D. \tag{12}$$

where $\mathcal{Y}^d_{m_d}$ denotes the low-dimensional feature tensor of the m_dth sample in user d. Then each user individually calculates its sum of square Frobenius norm of low-dimensional tensors which is denoted as $\{\Sigma^{M_d}_{m_d=1} \|\tilde{\mathcal{Y}}^d_{m_d}\|^2_F, d = 1, \ldots, D\}$ and sends it to the server. After receiving the sum of square Frobenius norms of low-dimensional feature tensors, the server calculates the initial total tensor scatter $\Psi_{\mathcal{Y}_0}$ by operating a summation step:

$$\Psi_{\mathcal{Y}_0} = \Sigma^D_{d=1} \Sigma^{M_d}_{m_d=1} \|\tilde{\mathcal{Y}}^d_{m_d}\|^2_F \tag{13}$$

Please note that each user should not send its low-dimensional feature tensor $\tilde{\mathcal{Y}}^d_{m_d}$ directly to the server because the original high-dimensional tensor can be recovered by the following equation:

$$\tilde{\mathcal{X}}^d_{m_d} = \tilde{\mathcal{Y}}^d_{m_d} \times_1 \tilde{\boldsymbol{U}}^{(1)} \times_2 \tilde{\boldsymbol{U}}^{(2)} \ldots \times_N \tilde{\boldsymbol{U}}^{(N)},\ m_d = 1, \ldots, M_d;\ d = 1, \ldots, D.$$

On the opposite, if each user d only sends the sum of squared Frobenius norm of low-dimensional tensors $\Sigma_{m_d=1}^{M_d} \|\tilde{\mathcal{Y}}^d_{m_d}\|_F^2$ to the server, its original high-dimensional tensor samples cannot be recovered and data privacy protection is guaranteed.

The iteration step to update projection matrices $\{\tilde{\boldsymbol{U}}^{(n)}, n = 1, \ldots, N\}$ is similar to MPCA in subsection 2.2. Specifically, in each iteration K, we derive $\{\tilde{\boldsymbol{U}}^{(n)}, n = 1, \ldots, N\}$ from algorithm (4) in 2.4.3. The total scatter variation of low-dimensional tensor $\Psi_{\mathcal{Y}_k}$ can be derived following the same procedure as the generation of $\Psi_{\mathcal{Y}_0}$.

Algorithm 5: FMPCA updating algorithm

```
1  Input: A set of D users where each user has $M_d$ high-dimensional tensor samples
     $\{\mathcal{X}^d_{m_d} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}, m_d = 1, \ldots, M_d;\ d = 1, \ldots, D\}$.
2  Output: A set of D users where each user has $M_d$ low-dimensional tensor samples
     after projection $\{\mathcal{Y}^d_{m_d} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}, m_d = 1, \ldots, M_d;\ d = 1, \ldots, D\}$.
3  Step 1 (Preprocessing): Centralize $\{\mathcal{X}^d_{m_d}, m_d = 1, \ldots, M_d;\ d = 1, \ldots, D\}$ to
     $\{\tilde{\mathcal{X}}^d_{m_d}, m_d = 1, \ldots, M_d;\ d = 1, \ldots, D\}$ by Algorithm (2).
4  Step 2 (Initialization): Initialize $\{\tilde{\boldsymbol{U}}^{(n)}, n = 1, \ldots, N\}$ by Algorithm (3).
5  Step 3 (Local Optimization):
6  Each user $d, d = 1, \ldots, D$, downloads initialized projection matrices
     $\{\tilde{\boldsymbol{U}}^{(n)}, n = 1, \ldots, N\}$ from the central aggregation Server and calculates its
     low-dimensional feature tensor privately -
     $\{\tilde{\mathcal{Y}}^d_{m_d} = \tilde{\mathcal{X}}^d_{m_d} \times_1 \tilde{\boldsymbol{U}}^{(1)^\top} \times_2 \tilde{\boldsymbol{U}}^{(2)^\top} \ldots \times_N \tilde{\boldsymbol{U}}^{(N)^\top}, m_d = 1, \ldots, M_d;\ d = 1, \ldots, D\}$
7  Each user $d, d = 1, \ldots, D$, calculates its sum of squared Frobenius norm of
     low-dimensional tensors $\Sigma_{m_d=1}^{M_d} \parallel \tilde{\mathcal{Y}}^d_{m_d} \parallel_F^2$ privately and send it to the server.
8  The server calculates $\Sigma_{d=1}^{D} \Sigma_{m_d=1}^{M_d} \parallel \tilde{\mathcal{Y}}^d_{m_d} \parallel_F^2$ and denote it as $\boldsymbol{\Psi}_{\mathcal{Y}_0}$.
9  for $k = 1 : K$ do
10     for $n = 1 : N$ do
11         Calculate $\tilde{\boldsymbol{U}}^{(n)}$ by Algorithm (3)
12     end
13     Follow the generation of $\Psi_{\mathcal{Y}_0}$ in the line 6 - 8, calculate $\Psi_{\mathcal{Y}_k}$
14     If $\Psi_{\mathcal{Y}_k} - \Psi_{\mathcal{Y}_{k-1}} \leq \eta$, break and go to Step 4
15 end
16 Step 4 (Projection):
17 For each user $d, d = 1, \ldots, D$, the low-dimensional feature tensor is obtained as
     $\{\mathcal{Y}_{m_d} = \mathcal{X}^d_{m_d} \times_1 \tilde{\boldsymbol{U}}^{(1)^\top} \times_2 \tilde{\boldsymbol{U}}^{(2)^\top} \ldots \times_N \boldsymbol{U}^{(N)^\top}, m_d = 1, 2, \ldots, M_d\}$
```

3 Prognostics Model Construcion and Real-Time TTF Prediction

In this section, we will give a real world example for the high-dimensional tensor sample and the response in MPCA (Lu et al. (2008)), then discuss how to build a prognostic model to predict the response based on the high-dimensional tensor samples.

In this article, we treat the degradation image streams and the time-to-failure (TTF) as the high-dimensional tensor samples and the response, respectively. Specifically, degradation is an irreversible process of damage accumulation and can be monitored by sensing technology which yields data known as degradation data/signals. It is useful to predict the assets' time-to-failure (TTF) via a process known as prognostic. The reason we choose imaging-based degradation data is that it contains rich information of the object being monitored, are generally non-contact and do not require permanent installation or fixturing.

To connect the response (TTF) and low-dimensional feature tensor, we construct a (log)-location-scale (LLS) tensor regression model. LLS regression has been widely used in reliability and survival analysis (Doray (1994)) because it includes a variety of TTF distributions, such as (log)normal, (log)logistics, smallest extreme value (SEV), and Weibull, etc.

Similar to subsection 2.4.4, we denote the training high-dimensional tensor sample set as $\{\mathcal{X}^d_{m_d} \in \mathbb{R}^{I_1 \times I_2 \times \ldots I_N}, m_d = 1, \ldots, M_d; d = 1, \ldots, D\}$, where D is the number of users, M_d is the number of high-dimensional tensor samples in user d and use $\{z^d_{m_d} \in \mathbb{R}, m_d = 1, \ldots, M_d; d = 1, \ldots, D\}$ to represent the TTFs corresponding to the high-dimensional tensor samples of all the users. In addition to the training data, we denote the test high-dimensional tensor samples as $\{\mathcal{X}_t \in \mathcal{R}^{I_1 \times \ldots \times I_n}, t = 1, \ldots, T\}$, where T is the number of test samples.

To detect the low-dimensional tensor subspace in which the high-dimensional degradation images are embedded, we apply our proposed federated multilinear princial component analysis (FMPCA) to the training high-dimensional tensor samples $\{\mathcal{X}^d_{m_d} \in \mathbb{R}^{I_1 \times I_2 \times \ldots I_N}, m_d = 1, \ldots, M_d; d = 1, \ldots, D\}$. Then the projection matrices $\{\tilde{\boldsymbol{U}}^{(n)} \in \mathbb{R}^{I_n \times P_n}\}$ which form the low-dimensional tensor subspace $\mathbb{R}^{P_1} \otimes \mathbb{R}^{P_2} \otimes \ldots \otimes \mathbb{R}^{P_n}$ can be generated. To extract the low-dimensional features of the training and test assets, we expand the image streams in the low-dimensional tensor subspace $\mathbb{R}^{P_1} \otimes \mathbb{R}^{P_2} \otimes \ldots \otimes \mathbb{R}^{P_N}$ using the projection matrices $\{\tilde{\boldsymbol{U}}^{(n)}, n = 1, \ldots, N\}$. This is achieved by the following equations:

$$\hat{\mathcal{Y}}^d_{m_d} = \mathcal{X}^d_{m_d} \times_1 \boldsymbol{U}_1^\top \times_2 \boldsymbol{U}_2^\top \ldots \times_N \boldsymbol{U}_N^\top,\ m_d = 1, \ldots, M_d;\ d = 1, \ldots, D. \tag{14}$$

$$\hat{\mathcal{Y}}_t = \mathcal{X}_t \times_1 \boldsymbol{U}_1^\top \times_2 \boldsymbol{U}_2^\top \ldots \times_N \boldsymbol{U}_N^\top,\ t = 1, \ldots, T. \tag{15}$$

where $\{\hat{\mathcal{Y}}^d_{m_d},\ m_d = 1, \ldots, M_d;\ d = 1, \ldots, D\}$ are the low-dimensional feature tensors of all the users in the training assets, and $\{\hat{\mathcal{Y}}_t,\ t = 1, \ldots, T\}$ is the low-dimensional feature tensor of the test asset.

Next, we construct a prognostic model using the low-dimensional feature tensors of all the users in the training data set $\{\hat{\mathcal{Y}}^d_{m_d},\ m_d = 1, \ldots, M_d;\ d = 1, \ldots, D\}$ and their TTFs $\{z_{m_d},\ m_d = 1, \ldots, M_d;\ d = 1, \ldots, D\}$. The (log)-location-scale distribution regression model is shown as follows:

$$z^d_{m_d} = \beta_0 + vec(\hat{\mathcal{Y}}^d_{m_d})^\top \boldsymbol{\beta}_1 + \sigma\epsilon_m, \tag{16}$$

where $vec(\hat{\mathcal{Y}}^d_{m_d})$ is the vectorization of $\hat{\mathcal{Y}}_m$, $\beta_0 \in \mathbb{R}$ and $\boldsymbol{\beta}_1 \in \mathbb{R}^{(P_1 \times P_2 \times \ldots \times P_N)\times 1}$ are the regression coefficients, σ is the scale parameter, and ϵ_m is the random noise term with a standard location-scale probability density function $f(\epsilon)$. For example, $f(\epsilon) = 1/\sqrt{2\pi}\exp(-\epsilon^2/2)$ for a normal distribution and $f(\epsilon) = \exp(\epsilon - \exp(\epsilon))$ for an SEV distribution. The parameters in criterion (16) can be estimated by solving the following optimization problem:

$$\min_{\boldsymbol{z}, \beta_0, \boldsymbol{\beta}_1, \sigma} \ \ell\left(\frac{\boldsymbol{z} - \boldsymbol{1}_M\beta_0 - \hat{\boldsymbol{Y}}\boldsymbol{\beta}_1}{\sigma}\right), \tag{17}$$

where $M = \Sigma_{d=1}^D M_d$ represents combining the high-dimensional tensor samples of D users together, $\hat{\boldsymbol{Y}} = (vec(\hat{\mathcal{Y}}_1)^\top, vec(\hat{\mathcal{Y}}_2)^\top, \ldots, vec(\hat{\mathcal{Y}}_M)^\top)^\top$, $\ell(\cdot)$ is the negative log-likelihood function of a location-scale distribution. For example, if TTFs follow normal distributions, then $\ell(\frac{\boldsymbol{z} - \boldsymbol{1}_M\beta_0 - \hat{\boldsymbol{Y}}\boldsymbol{\beta}_1}{\sigma}) = \frac{M}{2}\log 2\pi + M\log\sigma + \frac{1}{2}\Sigma_{m=1}^M \omega_m^2$, where $\omega_m = \frac{y_m - \beta_0 - \hat{\boldsymbol{y}}^m}{\sigma}$, where $\hat{\boldsymbol{y}}^m$ is the mth row of $\hat{\boldsymbol{Y}}$, and z_m is the TTF of asset m; if TTFs follow logistic distributions, then

$\ell(\frac{\boldsymbol{z}-\boldsymbol{1}_M\beta_0-\hat{\boldsymbol{Y}}\boldsymbol{\beta}_1}{\sigma}) = M\log\sigma - \Sigma_{m=1}^{M}\omega_m + 2\Sigma_{m=1}^{M}\log(1+\exp(\omega_m))$; if TTFs follow small extreme value (SEV) distributions, then $\ell(\frac{\boldsymbol{z}-\boldsymbol{1}_M\beta_0-\hat{\boldsymbol{Y}}\boldsymbol{\beta}_1}{\sigma}) = n\log\sigma - \Sigma_{m=1}^{M}\omega_m + \Sigma_{m=1}^{M}\exp(\omega_m)$.

However, it is challenging to solve criterion (17) since it is not convex. Thus, to simplify the development of optimization algorithms for model parameter estimation, we transorm criterion (17) to a convex one where we apply the following re-parameterazation: $\tilde{\sigma} = 1/\sigma, \tilde{\beta}_0 = \beta_0/\sigma, \tilde{\boldsymbol{\beta}}_1 = \boldsymbol{\beta}_1/\sigma$:

$$\{\hat{\tilde{\beta}}_0, \hat{\tilde{\boldsymbol{\beta}}}_1, \hat{\tilde{\sigma}}\} = \arg\min_{\tilde{\beta}_0,\tilde{\boldsymbol{\beta}}_1,\tilde{\sigma}} \ell(\tilde{\sigma}\boldsymbol{z} - \boldsymbol{1}_M\tilde{\beta}_0 - \hat{\boldsymbol{Y}}\tilde{\boldsymbol{\beta}}_1). \tag{18}$$

where $\ell(\tilde{\sigma}\boldsymbol{z} - \boldsymbol{1}_M\tilde{\beta}_0 - \hat{\boldsymbol{Y}}\tilde{\boldsymbol{\beta}}_1) = \frac{M}{2}\log 2\pi - M\log\tilde{\sigma} + \frac{1}{2}\Sigma_{m=1}^{M}\tilde{\omega}_m^2$ for TTFs following normal distributions, and $\tilde{\omega}_m = \tilde{\sigma}z_m - \tilde{\beta}_0 - \hat{\boldsymbol{y}}^m$, where $\hat{\boldsymbol{y}}^m$ is the mth row of $\hat{\boldsymbol{Y}}$ and z_m is the TTF of asset m; $\ell(\tilde{\sigma}\boldsymbol{z} - \boldsymbol{1}_M\tilde{\beta}_0 - \hat{\boldsymbol{Y}}\tilde{\boldsymbol{\beta}}_1) = -M\log\tilde{\sigma} - \Sigma_{m=1}^{M}\tilde{\omega}_m + 2\Sigma_{m=1}^{M}\log(1+\exp(\tilde{\omega}_m))$ for TTFs following logistics distributions, and $\ell(\tilde{\sigma}\boldsymbol{z} - \boldsymbol{1}_M\tilde{\beta}_0 - \hat{\boldsymbol{Y}}\tilde{\boldsymbol{\beta}}_1) = -n\log\tilde{\sigma} - \Sigma_{m=1}^{M}\tilde{\omega}_m + \Sigma_{m=1}^{M}\exp(\tilde{\omega}_m)$ for TTFs following SEV distributions.

After deriving the estimated parameters $\{\hat{\tilde{\beta}}_0, \hat{\tilde{\boldsymbol{\beta}}}_1, \hat{\tilde{\sigma}}\}$ from criterion (18), we can transform them back to the estimation of the parameters in the LLS regression model: $\hat{\gamma}_0 = \hat{\tilde{\gamma}}_0/\hat{\tilde{\sigma}}$, $\hat{\boldsymbol{\gamma}}_1 = \hat{\tilde{\boldsymbol{\gamma}}}_1/\hat{\tilde{\sigma}}$ and $\hat{\sigma} = 1/\hat{\tilde{\sigma}}$. As a result, the fitted LLS regression model can be constructed as $\hat{z}_m \sim LLS(\hat{\beta}_0 + vec(\hat{\mathcal{Y}}_m)^\top\hat{\boldsymbol{\beta}}_1, \hat{\sigma})$, where $\hat{\beta}_0 + vec(\hat{\mathcal{Y}}_m)^\top\hat{\boldsymbol{\beta}}_1$ and $\hat{\sigma}$ are respectively the estimated location and scale parameters.

Similar to the prognostic model for training set, the extracted low-dimensional feature tensor of the test asset are fed into the regression model established earlier to predict the asset's real-time TTF distribution: $\hat{z}_t \sim LLS(\hat{\beta}_0 + vec(\hat{\mathcal{Y}}_t)^\top\hat{\boldsymbol{\beta}}_1, \hat{\sigma})$.

Please note that there is data privacy leakage when we combine the low-dimensional feature tensor of all the users. Similar to section 2.4.4, the orignial high-dimensional tensor can be recovered based on the following equation:

$$\tilde{\mathcal{X}}_{m_d}^d = \tilde{\mathcal{Y}}_{m_d}^d \times_1 \tilde{\boldsymbol{U}}^{(1)} \times_2 \tilde{\boldsymbol{U}}^{(2)} \ldots \times_N \tilde{\boldsymbol{U}}^{(N)}, m_d = 1,\ldots,M_d; d = 1,\ldots,D.$$

.

where $\tilde{\mathcal{X}}_{m_d}^d$ is the m_dth high-dimensional tensor sample in the dth user and $\tilde{\mathcal{Y}}_{m_d}^d$ is the m_dth low-dimensioanl feature tensor of the dth user. To securely estimate the parameters in the LLS regression model, please refer to Federated LLS.

4 Numerical Study

In this section, we validate the effectiveness of our proposed FMPCA. The simulated data generation is based on a heat transfer process.

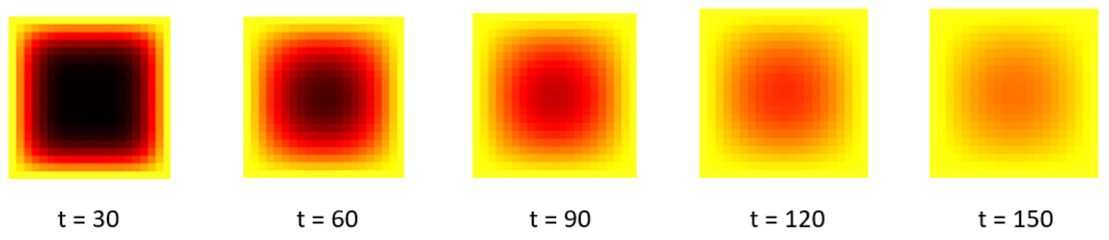
4.1 Data Generation

We generate degradation image streams for 500 assets. Assume for the image stream from asset m, which is denoted by $\mathcal{X}_m(x, y, t), m = 1, 2, \ldots, 500$, is generated from the following heat transfer process:

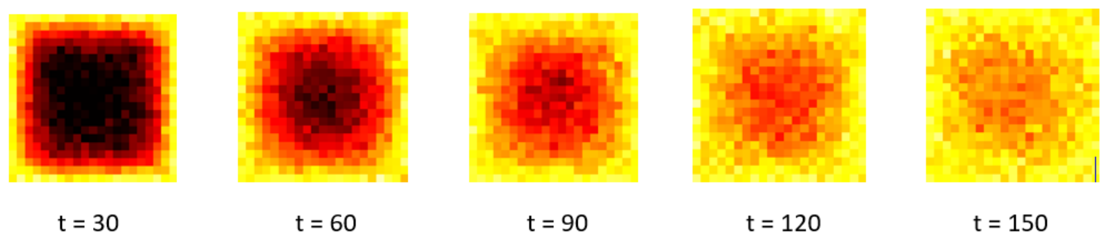
$$\frac{\partial \mathcal{X}_m(x, y, t)}{\partial t} = \alpha_m \Big(\frac{\partial^2 \mathcal{X}_m}{\partial x^2} + \frac{\partial^2 \mathcal{X}_m}{\partial y^2} \Big),$$

where $(x, y), 0 \leq x, y \leq 0.2$, represents the location of each image pixel. α_m is the thermal diffusivity coefficient for asset m, and is randomly generated from a uniform distribution $\mathcal{U}(0.5 \times 10^{-4}, 1 \times 10^{-4})$. t is the time frame. The initial and boundary conditions are set such that $\mathcal{X}|_{t=1} = 0$ and $\mathcal{X}_m|_{x=0} = \mathcal{X}_m|_{x=0.2} = \mathcal{X}_m|_{y=0} = \mathcal{X}_m|_{y=0.2} = 30$. At each time t, the image is recorded at locations $x = \frac{j}{n+1}, y = \frac{k}{n+1}, j, k = 1, \ldots, n$, resulting in an $n \times n$ matrix. Here, we set $n = 21$ and $t = 1, 2, \ldots, 150$, which generates 150 images of size 21×21 for each asset. For the convenience of computation complexity, we select the images whose indices are $15, 30, \ldots, 150$ which leads to 10 images of size 21×21 for each asset represented by $21 \times 21 \times 10$ tensor. Next, an independent and identically noise $\epsilon \sim N(0, 0.1)$ are added to each pixel. Figure 8 shows the degradation images with and without noise observed at time $t = 30, 60, 90, 120, 150$.

To simulate the TTF of each asset, we mimic the real-time TTF prediction process. Firstly, we apply MPCA to determine the low rank $\{P_n, n = 1, 2, 3\}$ where the variation kept in each mode is set to 97% as suggested. Next, each entry of projection matrices $\{\boldsymbol{U}_{(n)} \in \mathbb{R}^{I_n \times P_n}, n = 1, 2, 3\}$ follows IID standard normal distribution. The elements of coefficients $\beta_0 \in \mathbb{R}$ and $\boldsymbol{\beta}_1 \in \mathbb{R}^{(P_1 \times P_2 \times P_3) \times 1}$ are also generated by standard normal distribution and then divided by 100 to compromise the low-dimeneisional tensor $\{\mathcal{Y}_m, m = 1, \ldots, 500\}$. Similar to the generation of degradation stream images, an iid random noise $\epsilon_m \sim \boldsymbol{N}(0, 0.1)$ is added to each TTF. The TTF generation process are shown as follow:



(a) Without noise



(b) With noise

Figure 8: Simulated degradation images based on heat transfer process.

$$\mathcal{Y}_m = \mathcal{X}_m \times_1 \boldsymbol{U}^{(1)\top} \times_2 \boldsymbol{U}^{(2)\top} \times_3 \boldsymbol{U}^{(3)\top}, m = 1, \ldots, M.$$

$$z_m = e^{vec(\mathcal{Y}_m) \times \boldsymbol{\beta}_1 + \beta_0 + \epsilon_m}, m = 1, \ldots, M.$$

where $\mathcal{Y}_m$ and z_m denote the low-dimensional feature tensor and the TTF of the mth asset, respectively.

4.2 The Benchmark and Performance Comparison

We randomly select 400 assets as a training data set and the remaining 100 assets as a test data set. For the 400 assets in the training data set, 250 assets are randomly assigned to User 1, 100 assets to User 2 and the remaining 50 assets to User 3.

We compare the performance of our proposed method with 4 benchmarks. The first benchmark, denoted as "User Combination", utilizes the total 400 assets in the training data set for deriving the projection matrices $\{\boldsymbol{U}^{(n)}, n = 1, \ldots, 3\}$. The second benchmark, designated as "User 1", only uses the 250 assets in User 1 as the training data for generating projection matrices. Similar to "User 1", the third and fourth benchmark, which denoted as “User 2” and “User 3”, apply the 100 assets in User 2 and the 50 assets in User 3, respectively. All the 4 benchmarks employ MPCA (Lu et al. (2008)) to derive the projection matrices and extract low-dimensional features by the LLS-based prognostics model in Section 3. The dimension for the tensor subspace $\{P_n, n = 1, 2, 3\}$ is determined by cross-validation (CV). Specifically, we use the training data to conduct a 10-fold CV for various combination of (P_1, P_2, P_3), where $P_1 = 1, \dots, 5$, $P_2 = 1, \dots, 5$ and $P_3 = 1, \dots, 5$.

For our proposed method, referred to as “FMPCA”, we also use 10-fold CV to determine the dimension for the tensor subspace $\{P_n, n = 1, 2, 3\}$ and apply the LLS-based prognostics model in Section 3 to extract low-dimensional feature tensors. In this section, we use lognormal regression to build the prognostic model. The prediction errors of our proposed method and 4 benchmarks are calculated from the following equation:

$$\text{Prediction Error} = \frac{|\text{Estimated TTF} - \text{True TTF}|}{\text{True TTF}}. \tag{19}$$

To test the robustness of results, we repeat the above procedure (From splitting the 500 assets for training and test data set to calculating prediction error) 10 times and combine the prediction errors together.

4.3 Results and analysis

Figure 9 reports the prediction errors of our proposed method and the four benchmarks.

Figure 9 illustrates that the performance of our proposed method is the same as the 1st benchmark “User Combination”. We can see that the median absolute prediction errors (and min, 1st quantile, 3rd quantile, max) are both 0.13 (0, 0.03, 0.21, 0.41). It demonstrates that the projection matrices $\{\tilde{\boldsymbol{U}} \in \mathbb{R}^{I_n \times P_n}, n = 1, 2, 3\}$ derived by our proposed method FMPCA is exactly the same as MPCA (Lu et al. (2008)). Thus, our proposed FMPCA does not have the loss of performance accuracy while ensures user privacy and data security.

Figure 9 also suggests that “FMPCA” outperforms the other 3 benchmarks - “User 1”, “User 2” and “ User 3”. Besides, among the 3 decentralized benchmarks, “User 1” works the best while “User 3” has the worst prediction accuracy. For example, the median absolute prediction errors (and the Interquartile Range, i.e., IQRs) of “FMPCA” and “User 1”, “User 2”, “User 3” are 0.13 (0.17), 0.19 (0.28), 0.37 (0.39), 0.4 (0.59), separately. It is

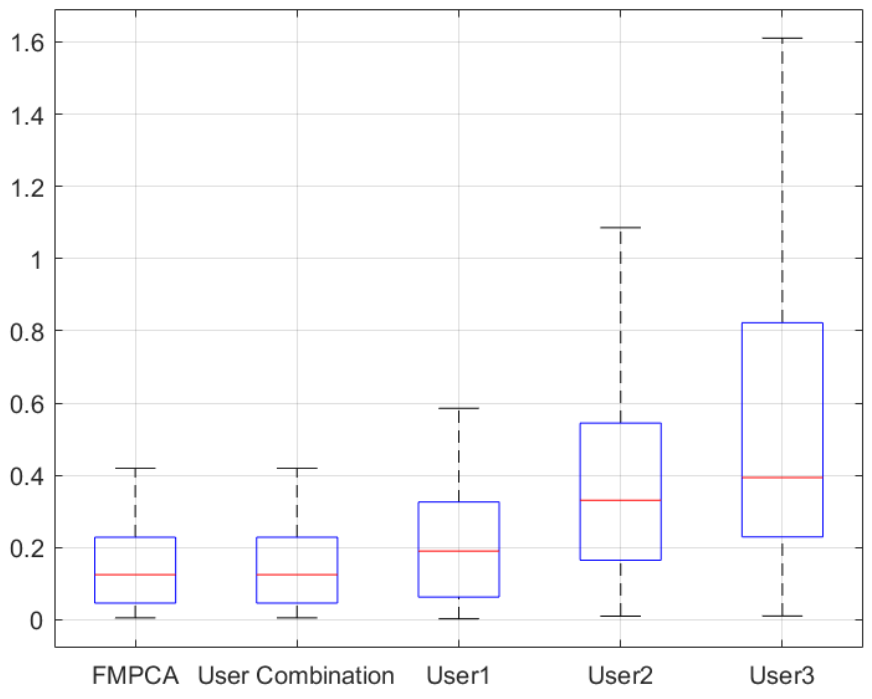


Figure 9: Numerical Study.

reasonable because federated learning enables several users to collaboratively train a model and data silo puts a limit on the training data size which results in a worse performance.

5 Case Study

In this section, we validate the effectiveness of our proposed method by degradation image streams obtained from a rotating machinery test bed. The experimental test bed is designed to perform accelerated degradation tests on rolling element thrust bearings. The test bearings are run from brand new to failure. Degradation images are collected by an FLIR T300 infrared camera. Meanwhile, an accelerometer was mounted on the test bed to moniter the vibration of the bearing. Failure time is defined once the amplitude of defective vibration frequencies crosses a threshold based on ISO standards for machine vibration. Each infrared image has 40×20 pixels. And totally 284 degradation image streams and their corresponding TTFs are generated. More detail about the data set can be found in Gebraeel et al. (2009) and Fang et al. (2019).

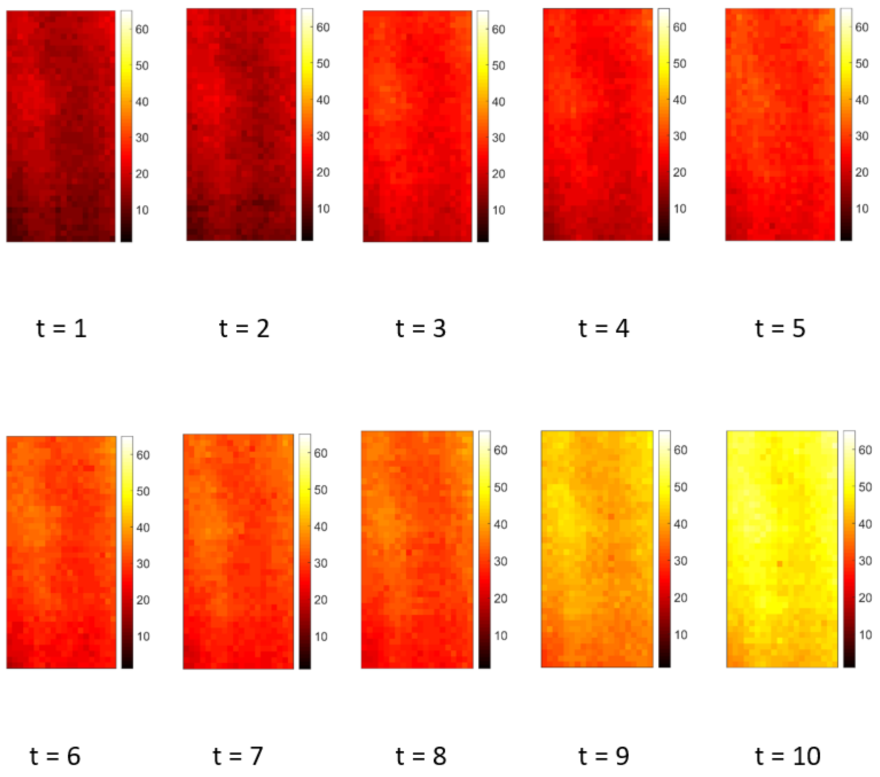


Figure 10: An illustration of one infrared degradation image stream.

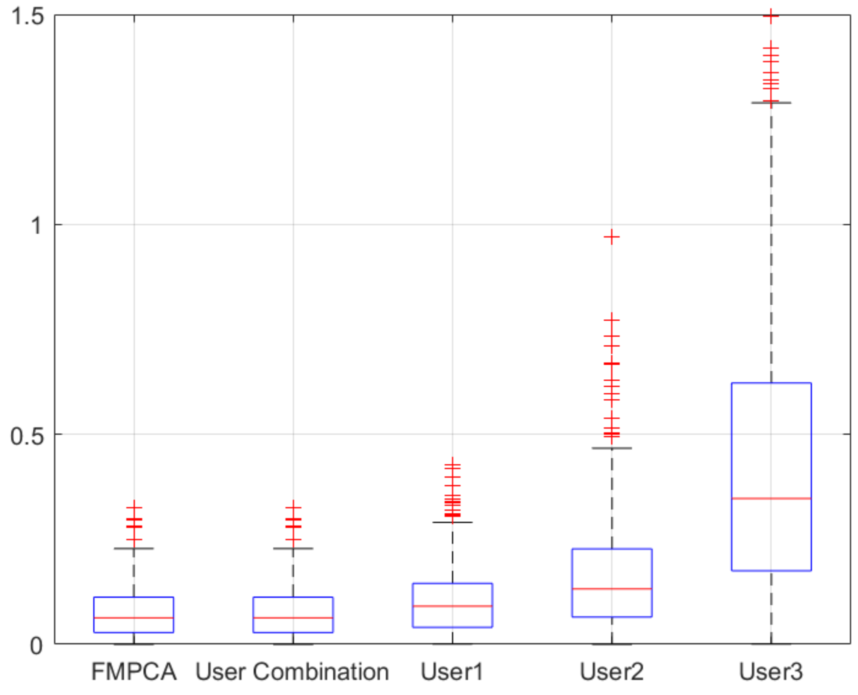


Figure 11: Case Study.

Due to failure time truncation, different assets may have different number of images which yields incomplete image streams. Thus, we first apply a tensor completion method known as TMac developed by Xu et al. (2013). Next, similar to numerical study, we randonly split the 284 image streams into a training data set containing 227 assets and a test data set consisting of the remaining 57 assets. The ratio is roughly 80% for training and 20% for testing. In the training data set, 140 assets are randomly assigned to User 1, 57 assets to User 2 and the last 30 assets to User 3. Similar to the numerical study, We use 10-fold cross-validation to determine the optimal combination of the rank of tensor subspace (P_1, P_2, P_3) from the range $1, \dots, 5$. In addition, same as the numerical study, we repeat the whole process 10 times and combine the prediction errors together to test the result robustness. Figure 11 shows the prediction errors of our proposed method and 4 benchmarks in case study.

Similar to the discovery in the numerical study in Section 4, Figure 11 indicates that our proposed method can achieve the same prediction accuracy as the 1st benchmark "User Combination" because the median absolute prediction errors (and min, 1st quantile, 3rd quantile, max) are both 0.06 (0, 0.02, 0.1, 0.22). It is verified again that our proposed method FMPCA can derive the same projection matrices $\{\tilde{\boldsymbol{U}} \in \mathbb{R}^{I_n \times P_n}, n = 1, 2, 3\}$ as MPCA (Lu et al. (2008). It also demonstrates that there is no performance loss in FMPCA for preventing privacy violation.

Figure 11 also shows that the prediction accuracy of the our proposed "FMPCA" is better than the other 3 benchmarks - "User 1", "User 2" and "User 3". And among the 3 benchmarks, "User 1" performs the best, "User 2" follows behind, and "User 3" has the worst prediction accuracy. For example, the median absolute prediction errors (and IQR) of "User Combination" and "User 1", "User 2", "User 3" are 0.06 (0.08), 0.1 (0.11), 0.15 (0.2), 0.31 (0.47). We again believe that federated learning enables collaboration between different users and achieve a better performance. And more samples in a dataset means more useful prediction information available which yields a better prediction performance.

6 Conclusion

This paper developed a federated learning-based multilinear principal component analysis (FMPCA) which aims to preserve user privacy and data security for multilinear princial component analysis (MPCA) Lu et al. (2008). This is achieved by proposing 3 algorithms towards data leakage locations in MPCA - those are - secure centralization of input tensor samples, secure derivation of projection matrices in Initialization, secure derivation of projection matrices in local optimization. In secure centralization of input tensor samples, each user calculates its individual sample mean, then generate perturbations for each pair of users and exchange it with all the other users. In this way, sample mean of each user cannot be not shared with the central server and other users which realize data privacy protection. The other 2 algorithms - secure derivation of projection matrices in initialization and local optimization, construct a new matrix with the tensor samples of all the users. We proved that the left singular vectors by applying SVD to the new matrix is equivalent to the projection matrices in MPCA. An updating procedure to derive the left singular vectors from the matrix is presented to preserve data confidentiality.

Referring to a simulated dataset as well as a data set from rotating machinery, we evaluated the effectiveness of our proposed FMPCA methodology. The result shows that our proposed method can derive exactly the same projection matrices as MPCA and indicates there is no performance loss while preventing privacy violation. It also verifies that federated learning enables multiple users to collaboratively train a model which yields a superior performance compared with data silo.

Acknowledgements

The authors acknowledge the generous support from the funding agency of XYZ.

Appendix

1 Proof of Proposition 1

Given $\boldsymbol{A}_{(n)} = [\tilde{\boldsymbol{X}}^1_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^1_{M_1(n)}, \tilde{\boldsymbol{X}}^2_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^2_{M_2(n)}, \ldots, \tilde{\boldsymbol{X}}^D_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^D_{M_D(n)}]$, where $\{\tilde{\boldsymbol{X}}^d_{m_d(n)} \in \mathbb{R}^{I_n \times (I_1 \times \ldots\ldots \times I_{n-1} \times I_{n+1} \times \ldots \times I_N)},\ m_d = 1, \ldots, M_d;\ d = 1, \ldots, D\}$ are concatenated horizontally in $\boldsymbol{A}_{(n)}$, we have $\boldsymbol{A}_{(n)} \boldsymbol{A}^\top_{(n)} = [\tilde{\boldsymbol{X}}^1_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^1_{M_1(n)}, \tilde{\boldsymbol{X}}^2_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^2_{M_2(n)}, \ldots, \tilde{\boldsymbol{X}}^D_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^D_{M_D(n)}] \cdot [\tilde{\boldsymbol{X}}^1_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^1_{M_1(n)}, \tilde{\boldsymbol{X}}^2_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^2_{M_2(n)}, \ldots, \tilde{\boldsymbol{X}}^D_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^D_{M_D(n)}]^\top$ which yields $\boldsymbol{A}_{(n)} \boldsymbol{A}^\top_{(n)} = \sum_{d=1}^{D} \sum_{m_d=1}^{M_d} \tilde{\boldsymbol{X}}^d_{m_d} \tilde{\boldsymbol{X}}^{d\top}_{m_d}$. Based on $\boldsymbol{\Phi}^{(n)*} = \Sigma_{d=1}^{D} \Sigma_{m_d=1}^{M_d} \tilde{\boldsymbol{X}}^d_{d(n)} \cdot \tilde{\boldsymbol{X}}^{d\top}_{d(n)}$ in Equation (7), we have $\boldsymbol{A}_{(n)} \boldsymbol{A}^\top_{(n)} = \boldsymbol{\Phi}^{(n)*}$.

If we apply singular value decomposition to $\boldsymbol{A}_{(n)}$ which generates $\boldsymbol{A}_{(n)} = \boldsymbol{U}^{(n)} \boldsymbol{\Sigma} \boldsymbol{V}^\top$, where $\boldsymbol{U}^{(n)} \in \mathbb{R}^{I_n \times r}$ is the left unitary matrix of $\boldsymbol{A}_{(n)}$, $\boldsymbol{\Sigma} \in \mathbb{R}^{r \times r}$ is diagonal matrix with positive singular values of $\boldsymbol{A}_{(n)}$, and $\boldsymbol{V} \in \mathbb{R}^{r \times (I_1 \times \ldots \times I_{n-1} \times I_{n+1} \times \ldots \times I_N \times \sum_{d=1}^{D} M_d)}$ is right unitary matrix of $\boldsymbol{A}_{(n)}$, $\boldsymbol{A}_{(n)} \boldsymbol{A}^\top_{(n)}$ can be expressed as $\boldsymbol{A}_{(n)} \boldsymbol{A}^\top_{(n)} = \boldsymbol{U}^{(n)} \boldsymbol{\Sigma} \boldsymbol{V}^\top \cdot (\boldsymbol{U}^{(n)} \boldsymbol{\Sigma} \boldsymbol{V}^\top)^\top$. This implies $\boldsymbol{A}_{(n)} \boldsymbol{A}^\top_{(n)} = \boldsymbol{U}^{(n)} \boldsymbol{\Sigma} \boldsymbol{V}^\top \boldsymbol{V} \boldsymbol{\Sigma} \boldsymbol{U}^{(n)\top}$. Furthermore, $\boldsymbol{A}_{(n)} \boldsymbol{A}^\top_{(n)} = \boldsymbol{U}_{(n)} \boldsymbol{\Sigma}^2 \boldsymbol{U}^{(n)\top}$ since $\boldsymbol{V}$ is orthogonal matrix.

Therefore, we have $\boldsymbol{A}_{(n)} \boldsymbol{A}^\top_{(n)} = \sum_{d=1}^{D} \sum_{m_d=1}^{M_d} \tilde{\boldsymbol{X}}^d_{m_d} \tilde{\boldsymbol{X}}^{d\top}_{m_d} = \boldsymbol{\Phi}^{(n)*} = \boldsymbol{U}^{(n)} \boldsymbol{\Sigma}^2 \boldsymbol{U}^{(n)\top}$. According to the definition of eigen-decomposition, $\boldsymbol{U}^{(n)}$ comprise all the eigenvectors of $\boldsymbol{\Phi}^{(n)*}$.

2 Proof of Proposition 2

Because all the iterations of the updating process are the same, we take the first iteration as an example.

When the first column of the second user block s is concatenated horizontally to the first user block, we have a new matrix $[\boldsymbol{A}_{1(n)}, \quad \boldsymbol{s}]$. And then we can apply SVD to $[\boldsymbol{A}_{1(n)}, \quad \boldsymbol{s}]$ which yields $[\boldsymbol{A}_{1(n)}, \quad \boldsymbol{s}] = \boldsymbol{U}_{direct} \boldsymbol{\Sigma}_{direct} \boldsymbol{V}_{direct}$, where $\boldsymbol{U}_{direct}$,$\boldsymbol{\Sigma}_{direct}$, $\boldsymbol{V}_{direct}$ denote the left unitary matrix, diagonal matrix with singular values and the right unitary matrix directly derively from $[\boldsymbol{A}_{1(n)}, \quad \boldsymbol{s}]$, respectively.

For the matrix $[\boldsymbol{A}_{1(n)}, \quad \boldsymbol{s}]$, if we only apply SVD to $\boldsymbol{A}_{1(n)}$ which yields $[\tilde{\boldsymbol{U}}^{(n)} \boldsymbol{\Sigma} \boldsymbol{V}, \quad \boldsymbol{s}]$. Then a transformation can be made to separate the right unitary matrix $\boldsymbol{V}$ apart:

$$[\tilde{\boldsymbol{U}}^{(n)}\boldsymbol{\Sigma}\boldsymbol{V}, \quad \boldsymbol{s}] = [\tilde{\boldsymbol{U}}^{(n)}, \quad \frac{\boldsymbol{e}}{\| \boldsymbol{e} \|}] \begin{bmatrix} \Sigma & \boldsymbol{w} \\ \boldsymbol{0} & \|\boldsymbol{e}\| \end{bmatrix} \begin{bmatrix} \boldsymbol{V} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{1} \end{bmatrix}^\top$$

where $\boldsymbol{e} = (\boldsymbol{I} - \tilde{\boldsymbol{U}}^{(n)}\tilde{\boldsymbol{U}}^{(n)\top})\boldsymbol{s}$ denotes the orthogonal projection of $\boldsymbol{s}$ onto the subspace which is orthogonal to $\tilde{\boldsymbol{U}}^{(n)}$, $\boldsymbol{w} = \tilde{\boldsymbol{U}}^{(n)\top}\boldsymbol{s}$ is the projection of $\boldsymbol{s}$ onto the basis of $\tilde{\boldsymbol{U}}^{(n)}$. Next, SVD can be performed to the middle matrix which is shown as follows:

$$\begin{bmatrix} \boldsymbol{\Sigma} & \boldsymbol{w} \\ \boldsymbol{0} & \|\boldsymbol{e}\| \end{bmatrix} = \hat{\boldsymbol{U}}\hat{\boldsymbol{\Sigma}}\hat{\boldsymbol{V}}^\top$$

then $[\boldsymbol{A}_{1(n)}, \quad \boldsymbol{s}]$ can be described as:

$$[\boldsymbol{A}_{1(n)}, \quad \boldsymbol{s}] = [\tilde{\boldsymbol{U}}^{(n)}, \quad \frac{\boldsymbol{e}}{\| \boldsymbol{e} \|}]\hat{\boldsymbol{U}}\hat{\boldsymbol{\Sigma}}\hat{\boldsymbol{V}}^\top \begin{bmatrix} \boldsymbol{V} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{1} \end{bmatrix}^\top$$

Thus, compared with the result of directly applying SVD to $[\boldsymbol{A}_{1(n)}, \quad \boldsymbol{s}]$, we have

$$\boldsymbol{U}_{direct} = [\tilde{\boldsymbol{U}}^{(n)}, \quad \frac{\boldsymbol{e}}{\| \boldsymbol{e} \|}]\hat{\boldsymbol{U}}$$

3 Proof of Proposition 3

From Equation (10), the covariance matrix $\boldsymbol{\Phi}^{(n)}$ is represented as $\boldsymbol{\Phi}^{(n)} = \Sigma_{d=1}^{D}\Sigma_{m_d=1}^{M_d}(\boldsymbol{X}_{m_d(n)}^{d} - \bar{\boldsymbol{X}}_{(n)}) \cdot \tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}} \cdot \tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}}^\top \cdot (\boldsymbol{X}_{m_d(n)}^{d} - \bar{\boldsymbol{X}}_{(n)})^\top$. Based on multiplication propery of matrix transpose $(\boldsymbol{A} \cdot \boldsymbol{B})^\top = \boldsymbol{B}^\top \cdot \boldsymbol{A}^\top$, we have $\boldsymbol{\Phi}^{(n)} = \Sigma_{d=1}^{D}\Sigma_{m_d=1}^{M_d}[(\boldsymbol{X}_{m_d(n)}^{d} - \bar{\boldsymbol{X}}_{(n)}) \cdot \tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}}] \cdot [(\boldsymbol{X}_{m_d(n)}^{d} - \bar{\boldsymbol{X}}_{(n)}) \cdot \tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}}]^\top$. For simplifying expression, we use $\tilde{\boldsymbol{X}}_{d(n)}^{d\boldsymbol{\Phi}}$ to denote $(\boldsymbol{X}_{m_d(n)}^{d} - \bar{\boldsymbol{X}}_{(n)}) \cdot \tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}}$ which yields $\boldsymbol{\Phi}^{(n)} = \Sigma_{d=1}^{D}\Sigma_{m_d=1}^{M_d}\tilde{\boldsymbol{X}}_{d(n)}^{d\boldsymbol{\Phi}} \cdot \tilde{\boldsymbol{X}}_{d(n)}^{d\boldsymbol{\Phi}\top}$.

Given $\boldsymbol{A}_{(n)}^{\boldsymbol{\Phi}} = [\tilde{\boldsymbol{X}}_{1(n)}^{1\boldsymbol{\Phi}}, \ldots, \tilde{\boldsymbol{X}}_{M_1(n)}^{1\boldsymbol{\Phi}}, \tilde{\boldsymbol{X}}_{1(n)}^{2\boldsymbol{\Phi}}, \ldots, \tilde{\boldsymbol{X}}_{M_2(n)}^{2\boldsymbol{\Phi}}, \ldots, \tilde{\boldsymbol{X}}_{1(n)}^{D\boldsymbol{\Phi}}, \ldots, \tilde{\boldsymbol{X}}_{M_D(n)}^{D\boldsymbol{\Phi}}]$, where $\{\tilde{\boldsymbol{X}}_{m_d(n)}^{d\boldsymbol{\Phi}} \in \mathbb{R}^{I_n \times (P_1 \times \ldots\ldots \times P_{n-1} \times P_{n+1} \times \ldots \times P_N)},\ m_d = 1, \ldots, M_d;\ d = 1, \ldots, D\}$ are concatenated horizontally in $\boldsymbol{A}_{(n)}^{\boldsymbol{\Phi}}$, we have $\boldsymbol{A}_{(n)}^{\boldsymbol{\Phi}}\boldsymbol{A}_{(n)}^{\boldsymbol{\Phi}\top} = [\tilde{\boldsymbol{X}}_{1(n)}^{1\boldsymbol{\Phi}}, \ldots, \tilde{\boldsymbol{X}}_{M_1(n)}^{1\boldsymbol{\Phi}}, \tilde{\boldsymbol{X}}_{1(n)}^{2\boldsymbol{\Phi}}, \ldots, \tilde{\boldsymbol{X}}_{M_2(n)}^{2\boldsymbol{\Phi}}, \ldots, \tilde{\boldsymbol{X}}_{1(n)}^{D\boldsymbol{\Phi}}, \ldots, \tilde{\boldsymbol{X}}_{M_D(n)}^{D\boldsymbol{\Phi}}] \cdot [\tilde{\boldsymbol{X}}_{1(n)}^{1\boldsymbol{\Phi}}, \ldots, \tilde{\boldsymbol{X}}_{M_1(n)}^{1\boldsymbol{\Phi}}, \tilde{\boldsymbol{X}}_{1(n)}^{2\boldsymbol{\Phi}}, \ldots, \tilde{\boldsymbol{X}}_{M_2(n)}^{2\boldsymbol{\Phi}}, \ldots, \tilde{\boldsymbol{X}}_{1(n)}^{D\boldsymbol{\Phi}}, \ldots, \tilde{\boldsymbol{X}}_{M_D(n)}^{D\boldsymbol{\Phi}}]^\top$ which yields $\boldsymbol{A}_{(n)}^{\boldsymbol{\Phi}}\boldsymbol{A}_{(n)}^{\boldsymbol{\Phi}\top} = \sum_{d=1}^{D}\sum_{m_d=1}^{M_d}\tilde{\boldsymbol{X}}_{m_d}^{d\boldsymbol{\Phi}}\tilde{\boldsymbol{X}}_{m_d}^{d\boldsymbol{\Phi}\top}$. Based on the derived equation $\boldsymbol{\Phi}^{(n)} = \Sigma_{d=1}^{D}\Sigma_{m_d=1}^{M_d}\tilde{\boldsymbol{X}}_{d(n)}^{d\boldsymbol{\Phi}} \cdot \tilde{\boldsymbol{X}}_{d(n)}^{d\boldsymbol{\Phi}\top}$ in the last paragraph, we have $\boldsymbol{A}_{(n)}^{\boldsymbol{\Phi}}\boldsymbol{A}_{(n)}^{\boldsymbol{\Phi}\top} = \boldsymbol{\Phi}^{(n)}$.

If we apply singular value decomposition to $\boldsymbol{A}_{(n)}^{\boldsymbol{\Phi}}$ which generates $\boldsymbol{A}_{(n)}^{\boldsymbol{\Phi}} = \boldsymbol{U}^{(n)}\boldsymbol{\Sigma}\boldsymbol{V}^\top$, where $\boldsymbol{U}^{(n)} \in \mathbb{R}^{I_n \times r}$ is the left unitary matrix of $\boldsymbol{A}_{(n)}^{\boldsymbol{\Phi}}$, $\boldsymbol{\Sigma} \in \mathbb{R}^{r \times r}$ is diagonal matrix with positive singular values of $\boldsymbol{A}^{\boldsymbol{\Phi}}_{(n)}$, and $\boldsymbol{V} \in \mathbb{R}^{r\times(P_1\times\ldots\times P_{n-1}\times P_{n+1}\times\ldots\times P_N\times\sum_{d=1}^{D} M_d)}$ is right unitary matrix of $\boldsymbol{A}^{\boldsymbol{\Phi}}_{(n)}$, $\boldsymbol{A}^{\boldsymbol{\Phi}}_{(n)}\boldsymbol{A}^{\boldsymbol{\Phi}\top}_{(n)}$ can be expressed as $\boldsymbol{A}^{\boldsymbol{\Phi}}_{(n)}\boldsymbol{A}^{\boldsymbol{\Phi}\top}_{(n)} = \boldsymbol{U}^{(n)}\boldsymbol{\Sigma}\boldsymbol{V}^{\top}\cdot(\boldsymbol{U}^{(n)}\boldsymbol{\Sigma}\boldsymbol{V}^{\top})^{\top}$. This implies $\boldsymbol{A}^{\boldsymbol{\Phi}}_{(n)}\boldsymbol{A}^{\boldsymbol{\Phi}\top}_{(n)} = \boldsymbol{U}^{(n)}\boldsymbol{\Sigma}\boldsymbol{V}^{\top}\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^{(n)\top}$. Furthermore, $\boldsymbol{A}_{(n)}\boldsymbol{A}^{\top}_{(n)} = \boldsymbol{U}^{(n)}\boldsymbol{\Sigma}^2\boldsymbol{U}^{(n)\top}$ since $\boldsymbol{V}$ is orthogonal matrix.

Therefore, we have $\boldsymbol{A}^{\boldsymbol{\Phi}}_{(n)}\boldsymbol{A}^{\boldsymbol{\Phi}\top}_{(n)} = \sum_{d=1}^{D}\sum_{m_d=1}^{M_d}\tilde{\boldsymbol{X}}^{d\boldsymbol{\Phi}}_{m_d}\tilde{\boldsymbol{X}}^{d\boldsymbol{\Phi}\top}_{m_d} = \boldsymbol{\Phi}^{(n)} = \boldsymbol{U}^{(n)}\boldsymbol{\Sigma}^2\boldsymbol{U}^{(n)\top}$. According to the definition of eigen-decomposition, $\boldsymbol{U}^{(n)}$ comprises all the eigenvectors of $\boldsymbol{\Phi}^{(n)}$ in equation (10).

4 Proof of Proposition 4

Similar to the proof of proposition 2, we take the first iteration in algorithm (4) as an example.

A new matrix $[\boldsymbol{A}^{\boldsymbol{\Phi}}_{1(n)}, \quad \boldsymbol{s}]$ is constructed by concatenating the first column of the second user block $\boldsymbol{s}$ horizontally to the first user block. If we directly apply SVD to $[\boldsymbol{A}^{\boldsymbol{\Phi}}_{1(n)}, \quad \boldsymbol{s}]$, then we have $[\boldsymbol{A}^{\boldsymbol{\Phi}}_{1(n)}, \quad \boldsymbol{s}] = \boldsymbol{U}^{\boldsymbol{\Phi}}_{direct}\boldsymbol{\Sigma}^{\boldsymbol{\Phi}}_{direct}\boldsymbol{V}^{\boldsymbol{\Phi}}_{direct}$, where $\boldsymbol{U}^{\boldsymbol{\Phi}}_{direct}$,$\boldsymbol{\Sigma}^{\boldsymbol{\Phi}}_{direct}$, $\boldsymbol{V}^{\boldsymbol{\Phi}}_{direct}$ denote the left unitary matrix, diagonal matrix with singular values and the right unitary matrix of $[\boldsymbol{A}^{\boldsymbol{\Phi}}_{1(n)}, \quad \boldsymbol{s}]$, respectively.

Based on algorithm (4), we conduct SVD to the first user block $\boldsymbol{A}^{\boldsymbol{\Phi}}_{1(n)}$ which results in $[\tilde{\boldsymbol{U}}^{(n)}\boldsymbol{\Sigma}\boldsymbol{V}, \quad \boldsymbol{s}]$. Then we can separate the right unitary matrix $\boldsymbol{V}$ by the following equation:

$$[\tilde{\boldsymbol{U}}^{(n)}\boldsymbol{\Sigma}\boldsymbol{V}, \quad \boldsymbol{s}] = [\tilde{\boldsymbol{U}}^{(n)}, \quad \frac{\boldsymbol{e}}{\parallel \boldsymbol{e} \parallel}]\begin{bmatrix}\boldsymbol{\Sigma} & \boldsymbol{w}\\ \boldsymbol{0} & \|\boldsymbol{e}\|\end{bmatrix}\begin{bmatrix}\boldsymbol{V} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{1}\end{bmatrix}^{\top}$$

where $\boldsymbol{e} = (\boldsymbol{I} - \tilde{\boldsymbol{U}}^{(n)}\tilde{\boldsymbol{U}}^{(n)\top})\boldsymbol{s}$ denotes the orthogonal projection of $\boldsymbol{s}$ onto the subspace which is orthogonal to $\tilde{\boldsymbol{U}}^{(n)}$, $\boldsymbol{w} = \tilde{\boldsymbol{U}}^{(n)\top}\boldsymbol{s}$ is the projection of $\boldsymbol{s}$ onto the basis of $\tilde{\boldsymbol{U}}^{(n)}$. Next, we apply an additional SVD to the middle matrix in the above equation:

$$\begin{bmatrix}\boldsymbol{\Sigma} & \boldsymbol{w}\\ \boldsymbol{0} & \|\boldsymbol{e}\|\end{bmatrix} = \hat{\boldsymbol{U}}\hat{\boldsymbol{\Sigma}}\hat{\boldsymbol{V}}^{\top}$$

which transform $[\boldsymbol{A}^{\boldsymbol{\Phi}}_{1(n)}, \quad \boldsymbol{s}]$ as follows:

$$[\boldsymbol{A}^{\boldsymbol{\Phi}}_{1(n)}, \quad s] = [\tilde{\boldsymbol{U}}^{(n)}, \quad \frac{\boldsymbol{e}}{\parallel \boldsymbol{e} \parallel}]\hat{\boldsymbol{U}}\hat{\boldsymbol{\Sigma}}\hat{\boldsymbol{V}}^{\top}\begin{bmatrix}\boldsymbol{V} & \boldsymbol{0}\\ \boldsymbol{0} & \boldsymbol{1}\end{bmatrix}^{\top}$$

Therefore, compared with the left unitary matrix of applying SVD to $[\boldsymbol{A}^{\boldsymbol{\Phi}}_{1(n)}, \quad \boldsymbol{s}]$ directly, we have

$$\boldsymbol{U}^{\boldsymbol{\Phi}}_{direct} = [\tilde{\boldsymbol{U}}^{(n)}, \quad \frac{\boldsymbol{e}}{\| \boldsymbol{e} \|}]\hat{\boldsymbol{U}}$$

5 Proof in 2.4.1

$$\begin{aligned}
\frac{\sum_{d=1}^{D} M_d \bar{\mathcal{X}}'_d}{\sum_{d=1}^{D} M_d} &= \frac{\sum_{d=1}^{D} M_d(\bar{\mathcal{X}}_d + \frac{1}{M_d}\sum_{d'=1,d'\neq d}^{D} \mathcal{R}_{d,d'})}{\sum_{d=1}^{D} M_d} \\
&= \frac{\sum_{d=1}^{D} M_d \bar{\mathcal{X}}_d + \sum_{d=1}^{D}\sum_{d'=1,d'\neq d}^{D} \mathcal{R}_{d,d'}}{\sum_{d=1}^{D} M_d} \\
&= \frac{\sum_{d=1}^{D} M_d \bar{\mathcal{X}}_d + \sum_{d=1}^{D}\sum_{d'=1,d'\neq d}^{D} \mathcal{S}_{d,d'} - \sum_{d=1}^{D}\sum_{d'=1,d'\neq d}^{D} \mathcal{S}_{d',d}}{\sum_{d=1}^{D} M_d} \\
&= \frac{\sum_{d=1}^{D} M_d \bar{\mathcal{X}}_d}{\sum_{d=1}^{D} M_d} \\
&= \bar{\mathcal{X}}
\end{aligned}$$

References

Bonawitz, K., H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. Kiddon, J. Konečnỳ, S. Mazzocchi, B. McMahan, et al. (2019). Towards federated learning at scale: System design. *Proceedings of Machine Learning and Systems 1*, 374–388.

Brisimi, T. S., R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi (2018). Federated learning of predictive models from federated electronic health records. *International journal of medical informatics 112*, 59–67.

Doray, L. (1994). Ibnr reserve under a loglinear location-scale regression model. In *Casualty Actuarial Society Forum*, Volume 2, pp. 607–652. Citeseer.

Fang, X., K. Paynabar, and N. Gebraeel (2019). Image-based prognostics using penalized tensor regression. *Technometrics 61*(3), 369–384.

Gebraeel, N., A. Elwany, and J. Pan (2009). Residual life predictions in the absence of prior degradation knowledge. *IEEE Transactions on Reliability 58*(1), 106–117.

He, X., D. Cai, and P. Niyogi (2005). Tensor subspace analysis. *Advances in neural information processing systems 18.*

Huang, L., Y. Yin, Z. Fu, S. Zhang, H. Deng, and D. Liu (2020). Loadaboost: Loss-based adaboost federated machine learning with reduced computational complexity on iid and non-iid intensive care data. *Plos one 15*(4), e0230706.

Konečnỳ, J., H. B. McMahan, D. Ramage, and P. Richtárik (2016). Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*.

Lee, C.-S. and A. Elgammal (2005). Towards scalable view-invariant gait recognition: Multilinear analysis for gait. In *International Conference on Audio-and Video-Based Biometric Person Authentication*, pp. 395–405. Springer.

Lu, H., K. N. Plataniotis, and A. N. Venetsanopoulos (2008). Mpca: Multilinear principal component analysis of tensor objects. *IEEE transactions on Neural Networks 19*(1), 18–39.

McMahan, B., E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas (2017). Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. PMLR.

Shakhnarovich, G. and B. Moghaddam (2005). Face recognition in subspaces. In *Handbook of face recognition*, pp. 141–168. Springer.

Sheller, M. J., G. A. Reina, B. Edwards, J. Martin, and S. Bakas (2018). Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation. In *International MICCAI Brainlesion Workshop*, pp. 92–104. Springer.

Shokri, R. and V. Shmatikov (2015). Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pp. 1310–1321.

Xu, Y., R. Hao, W. Yin, and Z. Su (2013). Parallel matrix factorization for low-rank tensor completion. *arXiv preprint arXiv:1312.1254*.

Yang, J., D. Zhang, A. F. Frangi, and J.-y. Yang (2004). Two-dimensional pca: a new approach to appearance-based face representation and recognition. *IEEE transactions on pattern analysis and machine intelligence 26*(1), 131–137.

Yang, Q., Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu (2019). Federated learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning 13*(3), 1–207.

Ye, J., R. Janardan, and Q. Li (2004). Gpca: An efficient dimension reduction scheme for image compression and retrieval. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 354–363.