ABSTRACT

ZHOU, CHENGYU. High Dimensional Data Analysis for System Condition Monitoring. (Under the direction of Dr. Xiaolei Fang).

This thesis presents new methodologies focusing on the challenges of high dimensional data analysis for condition monitoring in modern manufacture, service sector, and energy area. Chapter 1 introduces the research background, motivation, and challenges, and briefly discusses the research topics in this dissertation.

Chapter 2 proposes a novel convex two-dimensional variable selection method that can inspire both group-wise and element-wise sparsity. It is based on a generalized matrix regression that regresses the quality index from the exponential family against a predictor matrix, in which the rows represent the process variables and columns represent stages. Two types of norms are introduced for regularization. The rows and columns of the regression coefficient matrix are simultaneously penalized using an ℓ_2 norm to inspire group-wise sparsity. In the meantime, all the elements of the coefficient matrix are penalized using an ℓ_1 norm to inspire element-wise sparsity.

Chapter 3 develops a supervised dimension reduction-based prognostic model that uses an asset's incomplete degradation images to predict its TTF. The proposed supervised dimension reduction method works as follows: First, it constructs an optimization criterion that comprises a feature extraction term and a regression term. The first term extracts low-dimensional features from complete/incomplete degradation image streams of training assets, and the second term builds the connection between these assets' TTFs and the extracted features using LLS regression. Solving the optimization criterion yields a set of tensor basis matrices for dimension reduction. The TTFs of the training assets are then regressed against their tensor features using LLS regression, and the parameters are estimated using maximum likelihood estimation. Next, a block updating algorithm is proposed to solve the optimization criterion and closed-form solutions are derived under normal or lognormal distribution no matter the degradation image streams are complete or incomplete.

Chapter 4 proposes a Federated Multilinear Principal Component Analysis (FMPCA) framework to achieve tensor dimension reduction for decentralized data. It is a federated learning-based methodology which is a collaboratively decentralized privacy-preserving technology aiming to overcome data silo. The motivation is achieved by proposing three algorithms for the data leakage steps in Multilinear Principal Component Analysis (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.

Chapter 5 concludes the dissertation.

High Dimensional Data Analysis for System Condition Monitoring

by
Chengyu Zhou

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Industrial Engineering

Raleigh, North Carolina
2023

APPROVED BY:

Dr. Shu-cherng Fang	Dr. Yunan Liu
Dr. Wenbin Lu	Dr. Xiaolei Fang Chair of Advisory Committee

BIOGRAPHY

Chengyu Zhou was born in Tianjin City, People's Republic of China. Before embarking on his Ph.D. journey under the guidance of Dr. Xiaolei Fang, Chengyu earned a B.S. degree in Mechanical Engineering and Automation from Beihang University, China, and an M.S. degree in Integrated Systems Engineering from The Ohio State University. In pursuit of his research and professional growth, Chengyu has also done a master's co-major in the Department of Statistics at NC State University during his Ph.D. program.

Chengyu's research interests primarily lie in industrial data analysis, focusing on high-dimensional and big data applications in signal processing, sensor systems, manufacturing, and service sectors. Throughout his research career, he has developed several diagnostics and prognostics models, employing a wide range of statistical, optimization, and machine learning methodologies.

ACKNOWLEDGEMENTS

I would like to begin by expressing my sincerest and deepest gratitude to my advisor, Dr. Xiaolei Fang, for his unwavering support, guidance, and constructive mentorship throughout my doctoral journey. Dr. Fang's insights have been invaluable, and I have learned a lot from working with him. I appreciate his dedication to my research and career development.

I also want to extend my thanks to my committee members, Dr. Shu-cherng Fang, Dr. Yunan Liu, and Dr. Wenbin Lu, for their valuable feedback and input on my research. Their expertise has greatly improved the quality of this work. Additionally, I am grateful to Dr. Xu Wu for attending my oral exam as the GSR.

Finally, I would like to express my heartfelt gratitude to my parents for their unconditional love and unwavering encouragement. Their belief in me has been the foundation of my success, and I could not have pursued my Ph.D. dream without their ultimate support and faith in me. Last but not least, I want to thank my lovely girlfriend for her continuing companionship, always warm like the sunshine.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER

1

INTRODUCTION

1.1 Background and Motivation

System condition monitoring, a key component of predictive maintenance, is gaining increasing attention due to the potential benefits it offers, including reduced maintenance costs, improved productivity, and increased system availability. To monitor the health of industrial equipment, sensors are used during the condition monitoring process to collect data on significant operating parameters, such as wave-based, vibration-based, and image-based anomalies (Alokita et al. (2019)). Based on the sensor data collected, predictive maintenance then leverages advanced analysis and artificial intelligence to predict system failures before they occur or anticipate maintenance needs.

Complex industrial systems in the real world are monitored by a large number of heterogeneous sensors, which yields high dimensional sensor data (Michau et al. (2020)). High-dimensional data from multiple sensors can provide significantly more comprehensive information for condition monitoring and fault diagnostics than one-dimensional data. An example of a two-dimensional example is Multistage Manufacturing Processes (MMPs), which comprise multiple components, stations, or stages that are involved in

the production of a product. MMPs offer a wealth of information that can be analyzed to detect and diagnose faults, making them a useful tool for quality control and process improvement. Since all the stages in the process are identical, they share the same process variables, including process control parameters and sensing data from condition monitoring sensors. For instance, Figure 1.1 illustrates a seven-stage hot strip mill, the primary function of which is to roll heated steel slabs thinner and longer through seven successive rolling mill stands and then coil up the lengthened steel sheet for transport to the following process (Shi and Zhou (2009)). Figure 1.2 shows that there are 9 process variables at Stage 5: target speed of rollers, the measured speed of rollers, looper value, target force on both side of the rollers, the measured force on the work side of rollers, measured force on the transfer side of rollers, roller gap, looper height, and temperature.

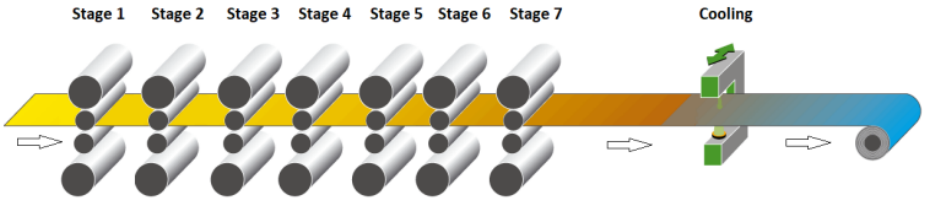


Figure 1.1: A hot strip mill with seven stands.

For 3D or higher dimensional sensor data (also denoted as a high-order tensor), an infrared image-based degradation signal from a rotating machinery is a representative example. The machinery is designed to run degradation experiments for thrust rolling element bearings. Using the machinery, bearing can be run from brand new to failure. During the experiment, an infrared camera is used to monitor the degradation process of the bearing and infrared images are recorded over time, which results in a degradation image stream (Fang et al. (2019)). As illustrated in Figure 1.3, an image stream is a form of 3D sensor data where the first two dimensions capture the structure of a single image, while the third dimension represents the time dimension and captures the progression of degradation over time.

High-dimensional condition monitoring data in modern industries usually has two tasks: **root-cause diagnostics** and **residual useful life (RUL) prognostics**. Root-cause diagnostics is a problem-solving process aimed at identifying the underlying cause of a problem or failure, rather than just addressing its symptoms. The process involves a systematic

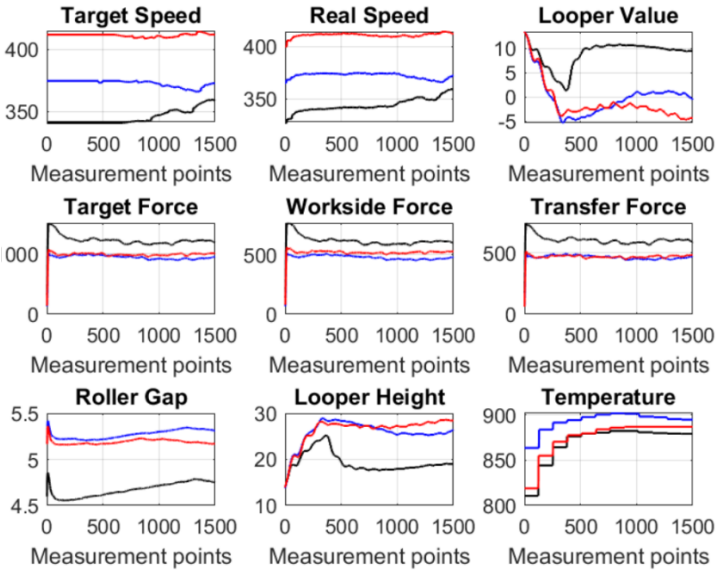


Figure 1.2: Process Variable profiles at Stage 5.

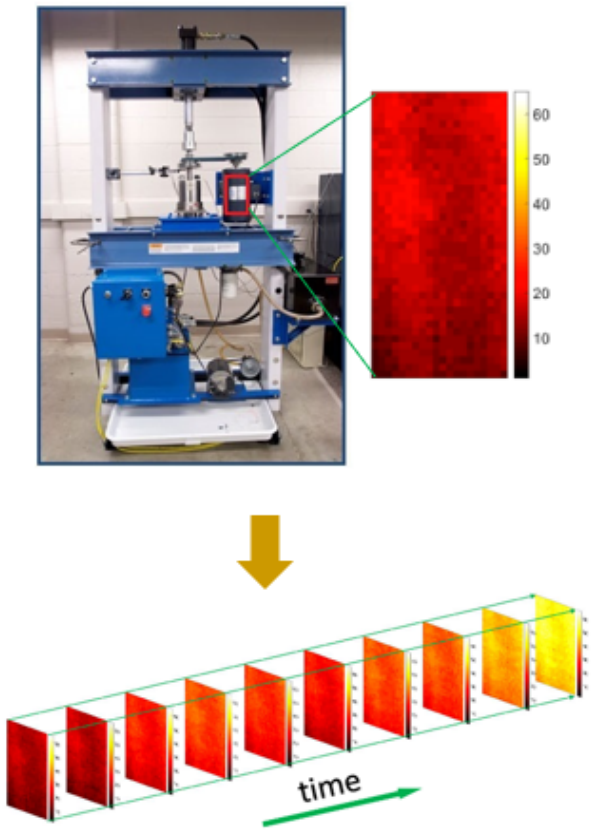


Figure 1.3: An infrared image-based degradation signal from a rotating machinery.

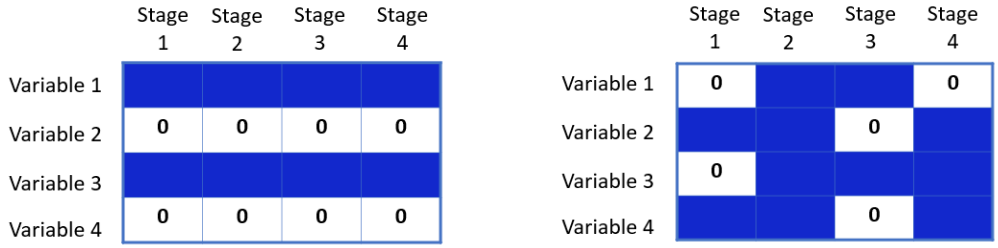
investigation of the problem or failure, data collection and analysis, and the use of tools and techniques to identify the root cause. Residual useful life (RUL) prognostics is a method used to estimate the remaining useful lifespan of a system or component, based on its current condition and performance. It involves analyzing data from sensors and other sources to predict when a system or component is likely to fail, so that preventative maintenance can be scheduled. Both tasks can help organizations reduce maintenance costs, increase system availability, and improve safety (Jardine et al. (2006)).

1.2 Research Challenges

In this section, we discuss the challenges of using high-dimensional condition monitoring data in root-cause diagnostics and residual useful life (RUL) prognostics.

1.2.1 Structured Variable Selection

As discussed earlier, root-cause diagnostics is the process of identifying the underlying factors or variables that contribute to a particular outcome or problem. Therefore, selecting the appropriate variables is a crucial step in root-cause diagnostic as it reduces the dimensionality of the problem and allows us to focus on the most relevant variables that are likely to be driving the outcome.



(a) Structured Variable Selection

(b) Unstructured Variable Selection

Figure 1.4: An illustration of Variable Selection with and without requirement

Unlike one-dimensional data, variable selection for high-dimensional data often has structured variable selection requirement which aims to identify a subset of variables that are most relevant to the outcome of interest. For instance, in the 2D process variable matrix

shown in Figure 1.4, it is clear that variable 2 and variable 4 are not crucial under structured variable selection, whereas there is no clear selection outcome under unstructured variable selection. Therefore, improving selection accuracy under structured variable selection requirements is an important yet challenging task.

1.2.2 Supervised Dimension Reduction

The parameter estimation for high dimensional data analysis is often a challenging issue. For example, assume that the image stream in Figure 1.3 consists of 30 images, each with 50×50 pixels. The corresponding 3D coefficient would require the estimation of 75,000 elements. As the dimensionality increases, the computational cost also increases exponentially. To overcome the challenge, as shown in Figure 1.7, deriving low-dimensional features is essential in high-dimensional data analysis.

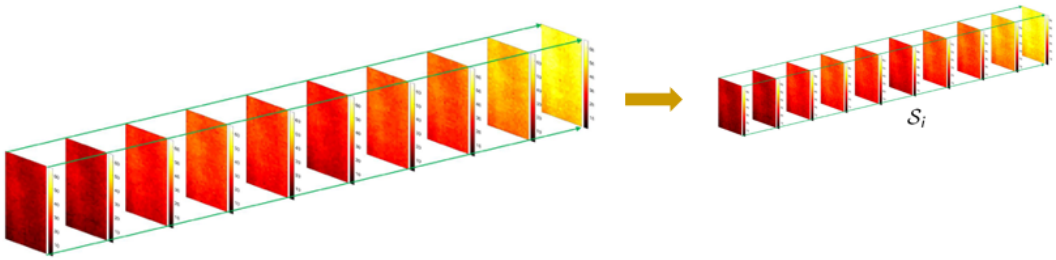


Figure 1.5: Dimension Reduction of 3D infrared image stream.

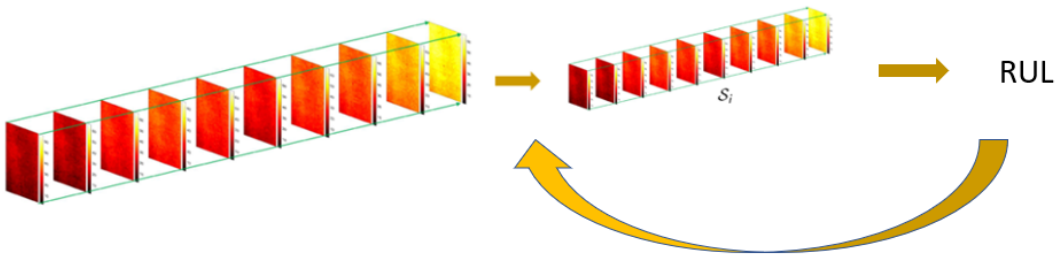


Figure 1.6: Supervised Dimension Reduction of 3D infrared image stream.

However, current high-dimensional dimension reduction techniques used in Remaining Useful Life (RUL) prediction are unsupervised, which means that RUL is not involved in the feature extraction process. This suggests that the dimension reduction and RUL prediction are sequential and separate steps. As a result, the low-dimensional features derived by unsupervised techniques may not be effective in predicting RUL. In order to improve the RUL prediction ability of extracted low-dimensional feature, as shown in Figure 1.6, developing a supervised dimension reduction model that uses RUL to supervise the dimension reduction process should be done carefully.

1.2.3 Handling Missing Data

Many high-dimensional data analysis models assume that sensor data is observed and collected without any errors, generating complete data. However, due to harsh environmental conditions during raw data acquisition and some artificial errors, the resulting sensor data often have different missing levels and corruptions. Therefore, dealing with missing data is essential for modern industrial predictive analysis models.

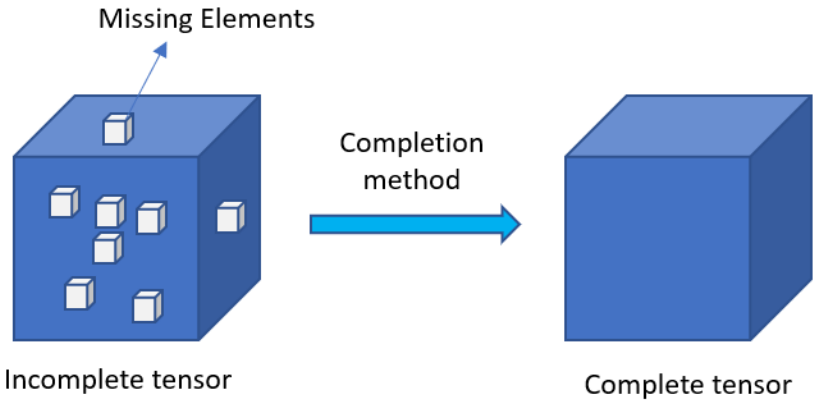


Figure 1.7: An illustration of 3D tensor completion.

However, most high-dimensional RUL prognostic models are only effective when the high-dimensional tensor data is complete. When dealing with high-dimensional data that have missing elements, as shown in Figure 1.7, these models require the use of tensor completion methods to fill in the missing elements before conducting RUL prognostics. While tensor completion methods often achieve relatively good performance, there are two

major disadvantages. First, tensor completion can be computationally expensive for high-dimensional tensors or large datasets because the algorithms involve solving optimization problems or performing iterative computations. Second, the accuracy of tensor completion algorithms can be sensitive to the amount and pattern of missing data, particularly when missing data is concentrated in specific regions or patterns. Therefore, it is worth exploring the development of high-dimensional RUL prognostic models that can handle the missing elements directly, without the need for tensor completion.

1.2.4 Privacy Protection

Modern high-dimensional data analysis models often require a large amount of data to achieve satisfactory performance, given the vast number of sensor data elements involved. However, in practice, data is often distributed among multiple users, resulting in small sample sizes for each user, as illustrated in Figure 1.8. Moreover, data ownership and governance are increasingly significant, and data privacy regulations have been enacted accordingly. Therefore, it is crucial to address privacy concerns when attempting to obtain comparable performance to what would be possible with sufficient sample sizes, while preserving data confidentiality.

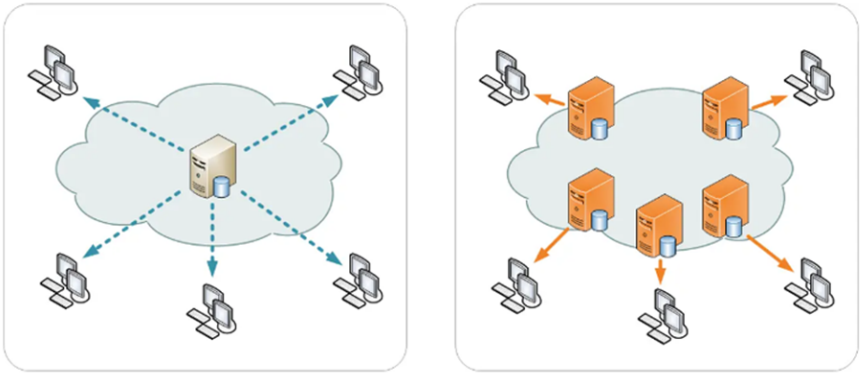


Figure 1.8: Comparison between centralized and decentralized frame

1.3 Overview of dissertation

In this section, we briefly introduce the research topics in the dissertation.

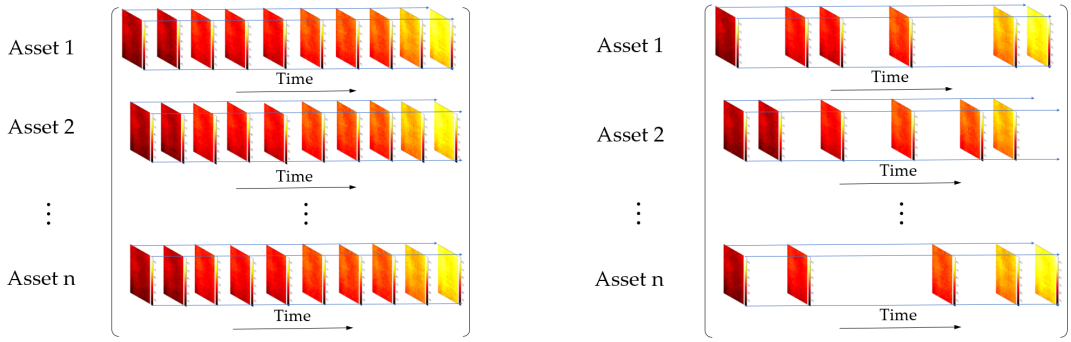
1.3.1 A Convex Two-Dimensional Variable Selection Method for the Root-Cause Diagnostics of Product Defects

Many multistage manufacturing processes consist of multiple identical stages. The root cause diagnostic of the product quality defects of these processes often involves the simultaneous identification of crucial stages and process variables that are related to product anomalies. In the literature, this is typically achieved by using penalized matrix regression that regresses the index of product defect against a matrix whose rows and columns respectively represent the stages and process variables. However, most existing models have some limitations that compromise their applicability and/or performance. For example, some models have an assumption on the rank of the coefficient, which often cannot be satisfied; some others formulate a nonconvex optimization criterion that easily results in a local optimum. Also, most models only provide diagnostics results with group-wise (i.e., stage- and variable-wise) sparsity. To address these challenges, this article proposes a novel convex two-dimensional variable selection method that can inspire both group-wise and element-wise sparsity.

Our methodology is based on a generalized matrix regression that regresses the quality index from the exponential family against a predictor matrix, in which the rows represent the process variables and columns represent stages. Two types of norms are introduced for regularization. The rows and columns of the regression coefficient matrix are simultaneously penalized using an ℓ_2 norm to inspire group-wise sparsity. In the meantime, all the elements of the coefficient matrix are penalized using an ℓ_1 norm to inspire element-wise sparsity. *The proposed method has the following advantages compared with existing methods developed by* (**?**Zhao et al. 2017; Zhao and Leng 2014). *First, it can be applied to a variety of applications since it assumes that the quality index is from the exponential family, which consists of various specific distributions. Second, it is formulated as a convex optimization criterion, which converges to the global optimum and is computationally more efficient. Third, it yields a solution with both group-wise and element-wise sparsity, which provides more insights to guide the revision of the feedback control algorithm.*

1.3.2 A Supervised Tensor Dimension Reduction-Based Prognostic Model for Applications with Incomplete Imaging Data

Recently, prognostic models with imaging-based degradation data have been investigated and attracted more and more attention because imaging data contains richer information of the object being monitored and imaging sensing technologies can easily be deployed. The existing imaging-based prognostic methods include deep learning-based models and statistical learning methods. Although effectiveness of these models have been well investigated, they share two common limitations. First, they assume imaging-based degradation data are complete which means images from all the assets should be collected continuously and regularly with the same sampling time interval (see Figure 1.9 (a) for an example). In reality, however, degradation images often contain significant levels of missing observations because of harsh environment effect on data collection (see Figure 1.9 (b) for an example). Such data incompleteness poses a significant challenge for the parameter estimation of existing statistical learning-based prognostic models. Second, the existing statistical learning-based prognostic models for applications with imaging data is that they employ unsupervised dimension reduction methods for feature extraction, so there is no guarantee that the extracted features are effective for the subsequent TTF prediction.



(a) Complete Image Streams (b) Incomplete Image Streams

Figure 1.9: Degradation stream images with and without missing data.

To address the aforementioned challenges, this topic proposes a supervised dimension reduction-based prognostic model that uses an asset's incomplete degradation images to predict its TTF. Specifically, **unlike the existing models, feature extraction in this article is**

achieved by utilizing historical TTFs to supervise the feature extraction process such that the extracted features are more effective for the subsequent TTF prediction. In addition, **the proposed supervised dimension reduction method works for both complete and incomplete degradation image streams**. The proposed supervised dimension reduction method works as follows: First, it constructs an optimization criterion that comprises a feature extraction term and a regression term. The first term extracts low-dimensional features from complete/incomplete degradation image streams of training assets, and the second term builds the connection between these assets' TTFs and the extracted features using LLS regression. Solving the optimization criterion yields a set of tensor basis matrices for dimension reduction. The TTFs of the training assets are then regressed against their tensor features using LLS regression, and the parameters are estimated using maximum likelihood estimation. Next, a block updating algorithm is proposed to solve the optimization criterion and closed-form solutions are derived under normal or lognormal distribution no matter no matter the degradation image streams are complete or incomplete.

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

In this thesis, chapter 3 concentrates on tensor dimension reduction-based prognostics models for incomplete data, and chapter 4 discusses tensor dimension reduction for decentralized data. Data in the real world is usually distributed among multiple users, leading to a small sample size in each user. Besides, the modern society is increasingly aware that data ownership and data governance are more significant, which yields law enactment corresponding to data privacy protection. All the reasons lead to data silo and put a limit on the application of tremendous machine learning (ML) and deep learning (DL) methodologies in modern industry.

To achieve tensor dimension reduction for decentralized data, a federated multilinear principal component analysis (FMPCA) framework is proposed in this topic. It is a federated learning-based methodology (McMahan et al. (2017); Konečnỳ 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 1.10 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 applications where training data are distributed at the edge (Brisimi et al. (2018); Huang et al. (2020)).

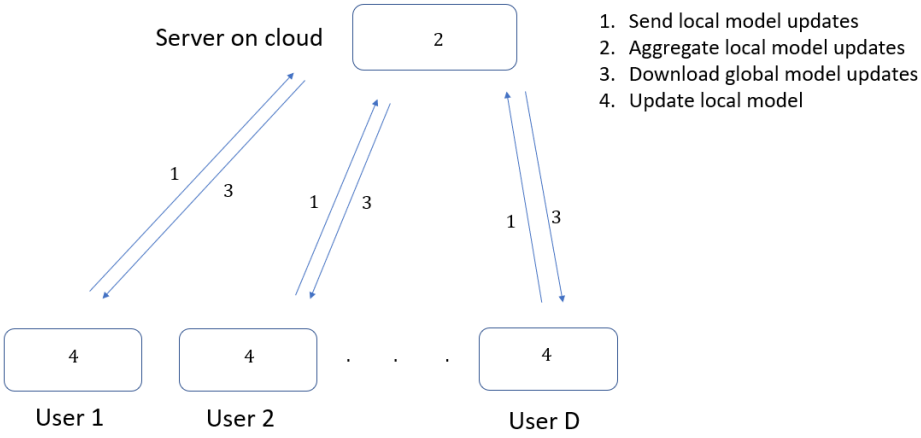


Figure 1.10: An illustration of Federated Learning.

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. 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.

1.4 Outline

The rest of the dissertation is organized as follows. Chapter 2 establishes a convex two-dimensional variable selection method that can inspire both group-wise and element-wise sparsity for the root-cause diagnostics of product defects. Chapter 3 presents a supervised dimension reduction-based prognostic model, which has two advantages over most image-based prognostic models: First, the model does not require tensor data to be complete, which expands its application to incomplete data. Second, it utilizes time-to-failure (TTF) to supervise the extraction of low-dimensional features, which makes the extracted features more effective for the subsequent prognostic. Chapter 4 proposes a federated multilinear principal component analysis (FMPCA) framework, which can achieve the same performance as multilinear principal component analysis while protecting user privacy and data confidentiality. Chapter 5 concludes the dissertation.

CHAPTER

2

A CONVEX TWO-DIMENSIONAL VARIABLE SELECTION METHOD FOR THE ROOT-CAUSE DIAGNOSTICS OF PRODUCT DEFECTS

2.1 Introduction

Many Multistage Manufacturing Processes (MMPs) consist of identical stages, units, stations, or operations. For example, Figure 2.1 demonstrates the schematic layout of a hot strip manufacturing process that is widely used in the steel-making industry. The primary function of the process is to reheat the semi-finished steel slabs to the rolling temperature using furnaces, roll them thinner and longer through a series of roughing and finishing rolling mill stands, and finally coil up the lengthened steel sheet. The finishing mill of the hot strip manufacturing process comprises seven identical stages (denoted as “F1" to “F7" in Figure 2.1), which have the most significant influence on the final thickness and quality

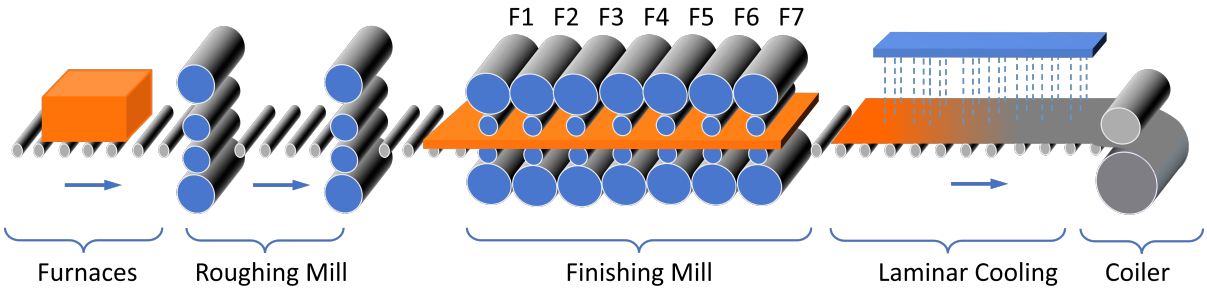
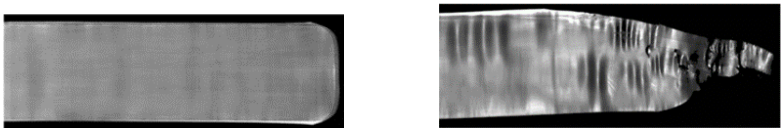


Figure 2.1: The schematic layout of the hot strip mill.

of the product (i.e., steel strip) (Peng et al. 2013). Another example of an MMP process with identical stages is the Additive Manufacturing (AM, also known as 3D printing) that fabricates products layer by layer (Gibson et al. 2021), in which each layer can be seen as one stage. One of the characteristics of these MMPs is that all the stages have the same process variables (including process control parameters and sensing data from condition monitoring sensors) since they are identical. For example, each of the seven stages of the finishing mill in Figure 2.1 has the following process variables: *target speed of rollers, the measured speed of rollers, looper value, target force on both sides of the rollers, the measured force on the work side of rollers, measured force on the transfer side of rollers, roller gap, looper height*, and *temperature*, etc (Jeong and Fang 2022). The values of these process variables usually vary from one product to another since they are either directly adjusted or passively affected by a closed-loop feedback control algorithm that tries to monitor and control the process in real-time for product quality guarantee. The inappropriate values of these process variables may result in defects in product quality. For example, Figure 2.2 illustrates products without and with quality defects from a hot strip manufacturing process. The root cause diagnostics of product quality defects focuses on identifying the crucial process variables (and their stage locations) whose inappropriate values are responsible for the defects of quality. The diagnostic results can be used to guide the revision of the feedback control algorithm to reduce the probability of or even avoid fabricating defective products in the future.

A typical technique for the root cause diagnostics of product defects is the LASSO-based regularization models (Tibshirani 1996), which have also been widely used in a variety of reliability applications such as risk assessment (Adland et al. 2021), failure time prediction (Fang et al. 2017; Rifaai et al. 2022), resilience analysis (Shen et al. 2021), and uncertainty quantification (Schöbi and Sudret 2019). This is achieved by regressing the binary quality



(a) A product without quality defects

(b) A product with quality defects

Figure 2.2: Products with and without quality defects from a hot rolling process (Balmashnova et al. 2012)

	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6	Stage 7
Variable 1	x		x			x	
Variable 2		x		x			x
Variable 3			x		x	x	
Variable 4	x					x	
Variable 5		x		x	x		x

Figure 2.3: Unstructured selection results ("x" represents the variable/stage is selected).

indices of products (the quality index is 0 if a product is defective and 1 otherwise) against the process variables pooled from all the stages via logistic regression and penalizes the regression coefficients using an ℓ_1 norm to inspire sparsity. Any process variables with nonzero coefficients are considered to be responsible for the product defects. However, one of the key limitations of the LASSO is that it cannot provide a structured solution. Specifically, as illustrated in Figure 2.3, a specific process variable might be identified as crucial in some stages and non-crucial in other stages, so it is challenging to determine if this process variable is important or not. Similarly, in a specific stage, some process variables are selected but some others are not, and thus the significance of the stage cannot be decided either. Such an unstructured solution does not provide much useful information to guide the revision of the feedback control algorithm since it usually selects many variables in many stages and does not show the overall importance of each variable and stage. Therefore, the control algorithm experts prefer a more structured solution that identifies a few important stages and process variables such as the ones shown in Figure 2.4, in which stages 2, 4, and 6, and variables 2, 3, and 5 are selected.

To provide a structured result for diagnostics, one straightforward method is to conduct a two-step variable selection. The first step identifies the crucial stages, which can be achieved by applying the group LASSO (Yuan and Lin 2006) on the logistic regression model

	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5	Stage 6	Stage 7
Variable 1							
Variable 2		x		x		x	
Variable 3		x		x		x	
Variable 4							
Variable 5		x		x		x	

Figure 2.4: Structured selection results ("x" represents the variable/stage is selected).

mentioned earlier (i.e., penalizing the regression coefficients corresponding to the variables in each column in Figure 2.3 as a group using an ℓ_2 norm). Next, the non-crucial stages are excluded from the logistic regression model, and the group LASSO is used again to select the crucial process variables (i.e., penalizing the regression coefficients corresponding to the variables in each row in Figure 2.3 as a group using an ℓ_2 norm after removing the non-crucial stages). Of course, the order of the two steps can be switched–that is–we may identify the crucial process variables first and then the important stages. However, one limitation of the two-step selection method is that its selection result is not necessarily reliable since when identifying the crucial stages, the unimportant process variables will compromise the selection accuracy, and vice versa.

Other than the two-step variable selection method, several statistical learning methods have been developed in the literature that can possibly be used to achieve the goal of providing a structured result for diagnostics. Article (Zhao and Leng 2014) proposed a structured LASSO method that models a normally distributed quality index as a bilinear product of the process variable matrix. The sparsity is inspired by penalizing the two unknown vectors in the bilinear product using an ℓ_1 norm. Although the structured LASSO can provide a structured variable selection result, it has two limitations. First, it assumes that the quality index is from a normal distribution, which is often not valid in real-world applications such as the hot strip mill example in Figure 2.1, in which the quality index is binary. Second, it can be proved that the structured LASSO method is equivalent to a penalized matrix regression where the rank of the coefficient matrix is one (see Proposition 1 in (Jeong and Fang 2022) for details). The rank assumption significantly restricts the generalizability of the structured LASSO method. This limitation was later addressed by (Zhao et al. 2017), which developed a trace regression model, which is a penalized matrix regression model that simultaneously penalizes the rows and columns of the regression coefficient matrix using an ℓ_2 norm. However, this model still assumes that the quality

index follows a normal distribution, which limits its applicability.

The latest effort in achieving the goal of providing a structured diagnostic result while addressing the aforementioned challenges is the two-dimensional variable selection method developed by (Jeong and Fang 2022). The method decomposes the unknown regression coefficients matrix as a product of two matrices and simultaneously penalizes the rows of the first matrix and the columns of the second matrix using an ℓ_2 norm. By doing so, they address the rank assumption imposed on the coefficient matrix in the structured LASSO (Zhao and Leng 2014). In addition, the two-dimensional variable selection method assumes that the quality index is from the exponential family, which significantly extends the application generalizability of the model since the exponential family consists of many specific distributions such as normal, Bernoulli, binomial, Poisson, gamma, etc. Although numerical studies have demonstrated the effectiveness of the two-dimensional variable selection method, it has two limitations. First, it was formulated as a non-convex optimization criterion, which implies the solution (i.e., the estimated regression coefficients) are usually not the optimal ones since they can easily get trapped in a local optimum. Second, it only provides solutions with group-wise sparsity, which is also a limitation of the models developed by (Zhao and Leng 2014) and (Zhao et al. 2017). Specifically, since it uses the ℓ_2 norm to inspire sparsity, it only identifies the crucial rows and/or columns of the process variable matrix but cannot tell the importance of the elements in each identified row/column (Hastie et al. 2019). In other words, in the final selection results, if a stage is identified as important, all the process variables in that stage will be identified as crucial. For example, Figure 2.5 summarizes the selected stages and process variables in Figure 2.4, where stages 2, 4, and 6 are identified as important, and process variables 2, 3, and 5 are selected. It can be seen that there is no element-wise sparsity since all the elements in the structured selection result are selected. In reality, control experts prefer a structured selection result with element-wise sparsity. This is because they are interested in knowing not only the overall importance of each stage but also the process variables with the highest priorities in each important stage. Figure 2.6 shows an example of the structured results with element-wise sparsity, in which, for example, stage 6 is chosen as an important stage. In the meantime, only variables 2 and 5 are selected in stage 6, and variable 3 is excluded. We would like to point out that although the result in Figure 2.6 looks similar to that of Figure 2.3, their statistical meanings are essentially different. The result in Figure 2.3 is from a LASSO-based method, and thus it cannot tell the group-wise (i.e., stage-wise and process variable-wise) importance. For example, the process variable 2 in stage 4 is selected in Figure 2.3, which only implies that the element itself (i.e., the variable 2 in stage 4) is im-

	Stage 2	Stage 4	Stage 6
Variable 2	x	x	x
Variable 3	x	x	x
Variable 5	x	x	x

Figure 2.5: Structured selection results without element-wise sparsity (“x" represents the variable/stage is selected).

	Stage 2	Stage 4	Stage 6
Variable 2		x	x
Variable 3	x		
Variable 5	x		x

Figure 2.6: Structured selection results with element-wise sparsity (“x" represents the variable/stage is selected).

portant, and one can neither conclude that overall stage 4 is important nor say that overall variable 2 is important. Unlike Figure 2.3, the results in Figure 2.6 provide two meanings: group-wise and element-wise importance. For example, it first implies that overall stage 6 is important. Second, it suggests that in stage 6, the two important elements are variables 2 and 5.

To address the limitations of the aforementioned models and provide a structured diagnostics result with element-wise sparsity, this chapter develops a convex two-dimensional variable selection method. The method is based on a generalized matrix regression that regresses the quality index from the exponential family against a predictor matrix, in which the rows represent the process variables and columns represent stages. Two types of norms are introduced for regularization. The rows and columns of the regression coefficient matrix are simultaneously penalized using an ℓ_2 norm to inspire group-wise sparsity. In the meantime, all the elements of the coefficient matrix are penalized using an ℓ_1 norm to inspire element-wise sparsity. The proposed method has the following advantages compared with existing methods developed by (Jeong and Fang 2022; Zhao et al. 2017; Zhao and Leng 2014). First, it can be applied to a variety of applications since it assumes that the quality index is from the exponential family, which consists of various specific distributions. Second, it is formulated as a convex optimization criterion, which converges to the global optimum and is computationally more efficient. Third, it yields a solution with both group-wise and

element-wise sparsity, which provides more insights to guide the revision of the feedback control algorithm.

The rest of the chapter is organized as follows. In Section 2.2, we introduce the details of the convex 2D variable selection method. Sections 2.3 and 2.4 validate the effectiveness of the proposed method by comparing its performance with some benchmarks using simulated data and a real-world dataset from a hot strip steel mill. Finally, Section 2.5 concludes.

2.2 The convex 2D variable selection method

This chapter proposes a convex 2D variable selection method for the quality defects diagnostics of multistage manufacturing processes with identical stages. The proposed method is built based on generalized matrix regression, the response variable and predictor of which are the quality index and process variable matrix (see Figure 2.3 for an example), respectively. The quality index is assumed to be from the exponential family, which consists of many specific distributions such as normal, binomial, Poisson, exponential, gamma, and inverse Gaussian, etc. It yields diagnostics results with both group-wise and element-wise sparsity. The group-wise sparsity is achieved by simultaneously penalizing the rows and columns of the unknown regression coefficient matrix using an ℓ_2 norm, which helps select a few crucial stages and process variables that significantly affect the quality of products. The element-wise sparsity is accomplished by penalizing the entries of the coefficient matrix using an ℓ_1 norm. It identifies a few crucial process variables in each selected stage, which suggests that only a subset but not all the process variables in a crucial stage are important. Similarly, it also selects a few crucial stages for each selected process variable, which implies even if a process variable is identified as overall important, it does not affect all but some of the stages.

We assume that there exists a historical dataset for model training (aka. parameter estimation). The dataset consists of the quality index and process variables of n products from the same multistage manufacturing process. We denote the quality index and process variable matrix of product i as $y_i \in \mathbb{R}$ and $\boldsymbol{X}_i \in \mathbb{R}^{s\times t}$, respectively, where s is the number of process variables, t is the number of stages. We assume that y_i follows a distribution from the exponential family. As a result, its probability mass or density function can be expressed as follows (McCullagh and Nelder 1989):

$$\mathbb{P}(y_i|\theta_i,\phi)=\exp\left\{\frac{y_i\theta_i-b(\theta_i)}{a(\phi)}+c(y_i,\phi)\right\}, \tag{2.1}$$

where θ_i is location parameter and $\phi>0$ is scale parameter. $a(\cdot)$, $b(\cdot)$, and $c(\cdot)$ are known functions determined by the specific distribution in the exponential family. For example, if $y_i\sim \boldsymbol{N}(\mu_i,\sigma^2)$, the parameters are $\theta_i=\mu_i$, $\phi=\sigma^2$ and the functions are $a(\phi)=\phi$, $b(\theta_i)=\theta_i^2/2$ and $c(y_i,\phi)=-\frac{1}{2}(y_i^2/\phi+\log(2\pi\phi))$; if y_i follows a Binomial distribution, i.e., $y_i\sim \boldsymbol{B}(m,p_i)$, then $\theta_i=\log(\frac{p_i}{1-p_i})$, $\phi=1$, $a(\phi)=1$, $b(\theta_i)=m\log(1+e^{\theta_i})$, $c(y_i,\phi)=\log\binom{m}{y_i}$; if y_i follows a Poisson distribution, i.e., $y_i\sim \boldsymbol{Poisson}(\lambda_i)$, then $\theta_i=\log(\lambda_i)$, $b(\theta_i)=e^{\theta_i}$, $a(\phi)=1$, and $c(y_i,\phi)=-\log(y_i!)$.

The connection between the quality index and the process variable matrix is established by using the link function. It maps the expectation of product i's quality index (denoted as μ_i) against its process variable matrix $\boldsymbol{X}_i$.

$$g(\mu_i)=\beta+\left\langle \boldsymbol{B},\boldsymbol{X}_i\right\rangle, \tag{2.2}$$

where $\mu_i=\mathbb{E}(y_i|\boldsymbol{X}_i)$ is the expectation of quality index. $g(\cdot)$ is the known link function that depends on the specific distribution that y_i follows. For example, if y_i follows a Normal distribution, then $g(\mu_i)=\mu_i$, and the model is a normal matrix regression model $\mathbb{E}(y_i|\boldsymbol{X}_i)=\beta+\left\langle \boldsymbol{B},\boldsymbol{X}_i\right\rangle$; if y_i follows a binomial distribution, one of the possible link functions is the logit function, $g(\mu_i)=\log(\frac{\mu_i}{1-\mu_i})$, and the model ends up with $\log(\frac{\mu_i}{1-\mu_i})=\beta+\left\langle \boldsymbol{B},\boldsymbol{X}_i\right\rangle$; if y_i follows a Poisson distribution, then $g(\mu_i)=\log(\mu_i)$, and the model is $\log(\mu_i)=\beta+\left\langle \boldsymbol{B},\boldsymbol{X}_i\right\rangle$. β is the intercept, and $\boldsymbol{B}\in\mathbb{R}^{s\times t}$ is the unknown regression coefficient matrix. $\langle\cdot,\cdot\rangle$ is the matrix inner product operator.

The coefficient matrix $\boldsymbol{B}$ in Equation (2.2) can be estimated by using the maximum likelihood estimation (MLE) method, which maximizes the following log-likelihood function:

$$\ell\big(\boldsymbol{B},\beta\big)=\sum_{i=1}^{n}\frac{y_i\theta_i-b(\theta_i)}{a(\phi)}+\sum_{i=1}^{n}c(y_i,\phi), \tag{2.3}$$

where $\theta_i=\beta+\left\langle \boldsymbol{B},\boldsymbol{X}_i\right\rangle$. Let $\mathscr{L}(\boldsymbol{B},\beta)$ be the negative log-likelihood function, i.e., $\mathscr{L}(\boldsymbol{B},\beta)=-\ell(\boldsymbol{B},\beta)$. MLE works by solving the following optimization criterion:

$$\min_{\boldsymbol{B},\beta}\ \mathscr{L}(\boldsymbol{B},\beta). \tag{2.4}$$

To achieve both group-wise (i.e., row- and column-wise) and element-wise selection, we simultaneously penalize each row, each column, and each element of matrix $\boldsymbol{B}$ using

an ℓ_2, ℓ_2, and ℓ_1 norm, respectively. Mathematically, we solve the following optimization criterion:

$$\min_{\boldsymbol{B},\beta} \mathscr{L}(\boldsymbol{B},\beta) + w\mathscr{R}(\boldsymbol{B}), \tag{2.5}$$

where $w > 0$ is the tuning parameter that adjusts the weights of the regularization term, which is defined as follows:

$$\mathscr{R}(\boldsymbol{B}) = \lambda\left(\sum_{i=1}^{s}\sqrt{t}\|\boldsymbol{b}_{i,:}\|_2 + \sum_{j=1}^{t}\sqrt{s}\|\boldsymbol{b}_{:,j}\|_2\right) + (1-\lambda)\sum_{i=1}^{s}\sum_{j=1}^{t}\|b_{i,j}\|_1 \tag{2.6}$$

where $\lambda \in [0,1]$ is a value that controls the weights of the group-wise and element-wise regularization terms; $\|\cdot\|_1$ is the ℓ_1 norm that inspires element-wise sparsity; $\|\cdot\|_2$ is the ℓ_2 norm, which inspires group-wise sparisty; s and t are respectively the number of row and column of $\boldsymbol{B}$; $\boldsymbol{b}_{i,:}$ is the ith row of $\boldsymbol{B}$; $\boldsymbol{b}_{:,j}$ is the jth column of $\boldsymbol{B}$, and $b_{i,j}$ is the (i,j)th element of $\boldsymbol{B}$. $\sqrt{s}$ and $\sqrt{t}$ account for the varying sizes of the penalty term.

The tuning parameters w and λ can be determined using a model selection criterion such as cross-validation (CV), Akaike Information Criterion (AIC), or Bayesian Information Criterion (BIC), etc. The optimization criterion (2.5) is convex, and thus it can be easily solved using existing optimization packages such as the CVX (Boyd et al. 2004).

2.3 Simulation Study

In this section, we validate the performance of our proposed method using simulated datasets with different correlation-structures, multiple sample sizes, and different noise levels.

2.3.1 Data Generation

We first generate the process variable matrix $\boldsymbol{X}_i \in \mathbb{R}^{s\times t}$ by considering two types of entry correlation structures: (i) no correlation (i.e., independent and identically distributed, IID) and (ii) row-wise correlation. For the first correlation structure, all the entries of $\boldsymbol{X}_i$ are generated from an IID standard normal distribution. For the second correlation structure, we set the correlation between the jth row and kth row of matrix $\boldsymbol{X}_i$ as $0.5^{|i-j|}$ to mimic the correlation among process variables, where $i = 1,\ldots,s$, $j = 1,\ldots,s$, and $s = 10$, $t = 10$.

Next, we generate the regression coefficient matrix $\boldsymbol{B} \in \mathbb{R}^{10\times 10}$, each entry of which follows a uniform distribution $\mathscr{U}[-1,1]$. To mimic the group-wise (i.e., row-wise and column-wise) sparsity, we let the 1st, 3rd, 5th, 7th, 9th rows and the 2nd, 4th, 6th, 8th and 10th columns of $\boldsymbol{B}$ be zeros. After setting the zero rows and columns, there are 25 nonzero elements left in the 100 entries of $\boldsymbol{B}$. To further mimic the element-wise sparsity, we randomly choose 12 of the 25 entries and set them to be zeros.

Third, we generate the product quality index y_i's using the following logistic regression model

$$\log(\tfrac{y_i}{1-y_i}) = \beta + \langle \boldsymbol{B}, \boldsymbol{X}_i + \boldsymbol{E}_i \rangle,$$

where $\beta = 0$; $\boldsymbol{E}_i$ is an IID noise/disturbance matrix from normal distribution $\mathscr{N}(0,\sigma^2)$. In this study, we will test three levels of noise: No Noise ($\sigma = 0$), Low Noise ($\sigma = 0.05$), and High Noise ($\sigma = 0.1$).

Finally, the generated data $\{\boldsymbol{X}_i, y_i\}_{i=1}^n$ are used to validate the performance of our model. We will evaluate the performance of our proposed method under two sample sizes: $n = 100$ and $n = 200$. The turning parameter w and weight λ are selected using 10-fold cross-validation.

2.3.2 Benchmark and Evaluation Criteria

The performance of our model will be compared with the 2D variable selection method developed by (Jeong and Fang 2022), which is known to be a state-of-the-art method. As mentioned earlier, it has two limitations compared with the method proposed in this paper. First, it is not convex, which implies its optimization criterion may converge to a local optimum, and thus its variable selection accuracy might be compromised. Second, it provides a solution with only group-wise sparsity but no element-wise sparsity.

The performance of the proposed method and the benchmark is evaluated using two criteria: group-wise (i.e., row-wise and column-wise) selection accuracy and element-wise selection accuracy. The group-wise selection accuracy is defined as follows:

$$\text{Group-Wise Accuracy} = \frac{\text{TP} + \text{TN}}{\text{No. of rows} + \text{No. of columns}}, \tag{2.7}$$

where “TP" represents “True Positive", which is the number of crucial rows and columns that are selected correctly. “TN" represents “True Negative", which is the number of non-crucial rows and columns that are removed correctly. “No. of rows" and “No. of columns" are the number of rows and columns in the process variable matrix, respectively.

Table 2.1: Average selection accuracy and precision for $\boldsymbol{X}_i$ with IID entries (%)

Noise	Method	Group/Element	Sample Size	TP	TN	FP	FN	Accuracy (SD)
No Noise	Proposed Method	Row+Column	100	99.60	90.80	9.20	0.40	95.20(5.31)
			200	100.00	98.00	2.00	0.00	99.00(2.13)
		Element	100	77.46	95.92	4.08	22.54	86.32(4.59)
			200	94.31	98.50	0.50	5.69	96.42(3.10)
	Benchmark	Row+Column	100	90.30	87.40	12.60	9.70	88.85(6.28)
			200	92.40	88.90	11.10	7.60	90.65(6.17)
		Element	100	81.40	28.60	71.40	18.60	55.00(3.02)
			200	82.70	29.48	70.52	17.30	56.10(2.98)
Low Noise	Proposed Method	Row+Column	100	99.80	88.50	11.50	0.20	94.15(5.46)
			200	100.00	97.50	2.50	0.00	98.75(2.29)
		Element	100	81.46	91.50	8.50	18.54	86.28(5.09)
			200	92.08	97.25	2.75	7.92	94.56(3.62)
	Benchmark	Row+Column	100	88.10	85.80	14.20	11.90	86.95(6.43)
			200	88.20	88.20	11.80	11.80	88.20(6.72)
		Element	100	80.54	28.00	72.00	19.46	54.27(3.07)
			200	81.54	28.08	71.92	18.46	54.81(3.18)
High Noise	Proposed Method	Row+Column	100	99.60	86.00	14.00	0.40	92.80(6.25)
			200	100.00	96.80	3.20	0.00	98.40(3.01)
		Element	100	81.38	91.08	8.92	18.62	86.04(5.07)
			200	92.08	96.42	3.58	7.92	94.16(3.57)
	Benchmark	Row+Column	100	87.30	85.30	14.70	12.70	86.30(6.22)
			200	87.70	86.40	13.60	12.30	87.05(6.67)
		Element	100	79.46	28.33	71.67	20.54	53.90(2.89)
			200	80.92	27.75	22.25	19.08	54.34(2.51)

The element-wise selection accuracy is computed using the equation below:

$$\text{Element-Wise Accuracy} = \frac{\text{TP} + \text{TN}}{\text{No. of elements}}, \tag{2.8}$$

where “TP" is the number of crucial entries that are selected correctly. “TN" is the number of non-crucial entries that are removed correctly. The denominator “No. of elements" is the total number of entries in the process variable matrix.

We repeat the evaluation process 100 times. The mean selection accuracy and its standard deviations for the $\boldsymbol{X}_i$'s generated from the first correlation structure (i.e., IID standard normal distribution) are reported in Table 2.1. Table 2.2 reports the mean selection accuracy and its standard deviations for the $\boldsymbol{X}_i$'s generated from the second correlation structure

(i.e., row-wise correlation). In addition to the “TP" and “TN", we also report the “FP" (i.e., the number of non-crucial rows/columns/elements that are falsely identified as crucial) and “FN" (i.e., the number of crucial rows/columns/elements that are falsely selected as non-cricial) in the two tables.

2.3.3 Results and Analysis

Table 2.1 indicates that, when there is no correlation among the entries of the process matrix $\boldsymbol{X}_i$, our proposed convex 2D variable selection method outperforms the benchmark in terms of (group-wise and element-wise) selection accuracy and precision under all different noise levels and sample sizes. For example, when there is no noise added to the quality index, and the sample size is 100, the group-wise selection accuracy (and the standard deviation, SD) of our proposed method is 95.20(5.31), while it is 88.85(6.28) for the benchmark; the element-wise selection accuracy (and SD) of our proposed method is 86.32.00(4.59), while it is 55.00(3.02) for the benchmark. When the noise level is low, and the sample size is 200, the group-wise selection accuracy (and SD) of our proposed method and the baseline model are respectively 98.75(2.29) and 88.20(6.72); the element-wise selection accuracy of the two methods are 94.56(3.62) and 54.81(3.18), respectively. When the noise level is high, and the sample size is 200, the selection accuracy (and SD) of the proposed convex 2D variable selection method and the benchmark are 98.40(3.01 and 86.30(6.22), respectively; the element-wise selection accuracy (and SD) of the two methods are respectively 94.16(3.57) and 54.34(2.51). We believe this is due to two reasons. First, our proposed method is convex while the benchmark is nonconvex, which implies that our proposed method converges to a global optimum while the baseline model might converge to a local optimum. Second and more importantly, our proposed method is able to achieve both group-wise and element-wise selection, while the benchmark can only inspire group sparsity. Since the true response variables (i.e., the quality indices) are generated using a regression coefficient matrix with both group-wise and element-wise sparsity, the group-wise selection accuracy (and precision) of the benchmark is compromised. This is because the benchmark tends to choose a row/column in which all the elements are nonzero. However, this is usually not the case in the generated data, in which even if a row/column is crucial, some of its entries are set to be zeros. In addition, since the benchmark cannot inspire element-wise sparsity, its element-wise selection accuracy is consistently around 50%, which is close to the accuracy of a random guess.

Table 2.1 also illustrates that the selection accuracy and precision of both the proposed

method and the benchmark decrease with the increase of noise. For example, when the sample size is 100, the group-wise selection accuracy (and SD) of our proposed method are 95.20(5.31), 94.15(5.46), and 92.80(6.25), when there is no noise, low noise, and high noise, respectively; the group-wise selection accuracy (and SD) of the benchmark model are respectively 88.85(6.28), 86.95(6.43), and 86.30(6.22). This is reasonable since higher noise added to the response variable will have more compromise on the accuracy of the parameter estimates, and thus results in a lower selection accuracy. In addition, we can observe from Table 2.1 that the selection accuracy and precision increase with the increase of sample size, which is also reasonable.

The accuracy and precision data in Table 2.2 imply that all the conclusions drawn from Table 2.1 still hold: The proposed convex selection method always outperforms the baseline model in terms of both (group-wise and element-wise) selection accuracy and precision; data with smaller noise result in higher selection accuracy and precision; a smaller sample size will yield worse selection accuracy and precision. In addition, comparing Tables 2.1 with 2.2, we notice that both the proposed method and the baseline model have worse selection accuracy and precision when there is a correlation among the rows of the process variable matrices. For example, when the noise level is high, and the sample size is 200, the group-wise selection accuracy (and SD) of the proposed method is 98.40(3.01) in Table 2.1, while it is 85.40(8.12) in Table 2.2. Similarly, the element-wise selection accuracy (and SD) of the proposed method is 94.16(3.57) in Table 2.1, while it is 82.64(3.91 in Table 2.2. Similar performance degradation can also be observed on the baseline model in Table 2.2. This is not something unexpected since it is known that the correlation among the rows/columns of the process variable matrices will compromise the selection accuracy and precision. The correlation between two rows represents that the two process variables contain similar information. If the correlation is high, then the regression model only needs one of them, and the other one is not needed due to the existence of the first one since the information provided by the two variables is overlapped.

2.3.4 Imbalanced Data

We also evaluate the performance of the proposed method and the benchmarks with imbalanced data. Specifically, we generate 100 samples, 90 of which are products with good quality, and the remaining 10 are products with bad quality. This is achieved by generating a large number of samples and choosing 90 whose quality indices (computed by using Equation (2.3.1)) are in the range $(0.5, 0.9]$ and the other 10 whose quality indices are smaller

Table 2.2: Average selection accuracy and precision for $\boldsymbol{X}_i$ with row-wise correlation (%)

Noise	Method	Group/Element	Sample Size	TP	TN	FP	FN	Accuracy (SD)
No Noise	Proposed Method	Row+Column	100	96.80	76.90	23.10	0.00	86.35(8.34)
			200	98.90	92.80	7.20	1.10	95.85(4.77)
		Element	100	71.31	89.58	10.42	28.69	80.08(4.51)
			200	72.62	93.67	6.33	27.38	82.72(3.89)
	Benchmark	Row+Column	100	97.20	67.70	32.30	2.80	82.45(8.39)
			200	93.40	81.40	18.60	6.60	87.40(6.34)
		Element	100	95.62	6.92	93.08	4.38	53.04(1.94)
			200	89.77	16.92	83.08	10.23	54.80(2.64)
Low Noise	Proposed Method	Row+Column	100	97.10	74.10	25.90	2.90	85.60(8.08)
			200	99.00	92.10	7.90	1.00	95.55(5.27)
		Element	100	71.46	89.21	10.79	28.54	80.06(4.86)
			200	72.23	94.00	6.00	27.77	82.68(3.82)
	Benchmark	Row+Column	100	96.70	66.80	33.20	3.30	81.75(8.66)
			200	93.20	80.30	19.70	6.80	86.75(6.57)
		Element	100	94.77	8.08	91.92	5.23	53.16(1.99)
			200	89.54	17.17	82.83	10.46	54.80(2.76)
High Noise	Proposed Method	Row+Column	100	96.80	74.00	26.00	3.20	85.40(8.12)
			200	98.70	92.40	7.60	1.30	95.52(4.97)
		Element	100	71.20	88.90	11.10	28.80	80.02(4.88)
			200	72.62	93.50	6.50	27.38	82.64(3.91)
	Benchmark	Row+Column	100	94.00	66.20	33.80	6.00	80.10(6.35)
			200	86.70	75.70	24.30	13.30	81.20(8.12)
		Element	100	90.46	14.67	85.33	9.54	52.57(2.57)
			200	78.85	30.00	70.00	21.15	55.40(3.03)

than 0.5. Similar to the previous study, we also consider two correlation structures (IID and row-wise correlation) and two noise levels (low and high). The mean selection accuracy and its standard deviation are reported in Tables 2.3 and 2.4.

Tables 2.3 and 2.4 indicate that all the conclusions drawn earlier still hold. In other words, the method proposed in this paper outperforms the benchmark in terms of (group-wise and element-wise) selection accuracy and precision under all noise levels and correlation structures of the process variable matrix. This again shows the importance of simultaneously achieving group-wise and element-wise sparsity. Comparing the results from imbalanced data (Tables 2.1 and 2.2) and balanced data (Tables 2.3 and 2.4), we notice that the data imbalance issue indeed compromises the performance of the proposed method and the

Table 2.3: Average selection accuracy and precision for $\boldsymbol{X}_i$ with IID entries (%) using imbalanced data

Noise	Method	Group/Element	Sample Size	TP	TN	FP	FN	Accuracy (SD)
Low Noise	Proposed Method	Row+Column	100	98.20	85.30	14.70	1.80	91.75(5.78)
		Element	100	79.30	91.25	8.75	20.70	85.28(5.42)
	Benchmark	Row+Column	100	86.33	83.90	16.10	13.67	85.12(6.51)
		Element	100	78.15	26.70	73.30	21.85	52.43(3.09)
High Noise	Proposed Method	Row+Column	100	95.28	82.30	17.70	4.72	88.80(6.45)
		Element	100	78.10	89.20	10.80	21.90	83.65(5.21)
	Benchmark	Row+Column	100	85.10	81.80	18.20	14.90	83.45(6.38)
		Element	100	77.60	25.52	74.48	22.40	51.56(2.57)

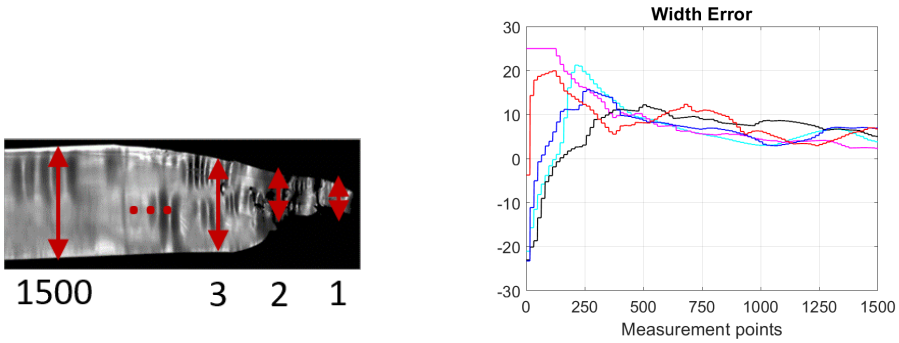
Table 2.4: Average selection accuracy and precision for $\boldsymbol{X}_i$ with row-wise correlation (%) using imbalanced data

Noise	Method	Group/Element	Sample Size	TP	TN	FP	FN	Accuracy (SD)
Low Noise	Proposed Method	Row+Column	100	95.60	72.90	27.10	4.40	84.25(8.12)
		Element	100	69.82	88.70	11.30	30.18	79.26(4.88)
	Benchmark	Row+Column	100	94.90	64.73	35.27	5.10	79.82(8.73)
		Element	100	93.12	7.90	92.10	6.88	50.50(2.04)
High Noise	Proposed Method	Row+Column	100	94.32	70.38	29.62	5.68	82.35(8.17)
		Element	100	70.10	87.30	12.70	29.90	78.70(4.95)
	Benchmark	Row+Column	100	92.75	65.10	34.90	7.25	78.93(8.22)
		Element	100	89.70	13.40	86.60	10.30	51.55(2.58)

benchmark. For example, Table 2.1 indicates that, when using balanced data, the group-wise and element-wise selection accuracy (and SD) of the proposed method are 94.15(5.46) and 86.28(5.09), respectively, when the noise level is low and the correlation structure is IID; while they are respectively 91.75(5.78) and 85.28(5.42) when using imbalanced data (see Table 2.3). Another observation is that although the data imbalance issue compromises the performance, the selection accuracy and SD of the proposed method are significantly better than that of the benchmark.

2.4 Case Study

In this section, we validate our proposed method using data from a real-world application. The dataset is from the hot strip mill shown in Figure 2.1. It consists of the process variables and quality indices of 490 strip steel products from the same rolling mill stand, including 264 good quality products and 226 bad quality products. The product quality is defined based on the percentage of width that is smaller than a predefined width threshold. Specifically, as shown in Figure 2.7(a), the width of each product was measured at 1500 locations uniformly distributed along with the head of the strip. Any measurement point whose width is smaller than the predefined width threshold is considered as a defective point. In other words, as illustrated in Figure 2.7(b), the measurement points with a negative width error are considered to be defective. We divide the number of defective points by the total number of measurement points $(1,500)$ to yield the quality index, whose range is $[0,1]$.



(a) Measurement points for width

(b) Difference between desired and measured width of 5 products.

Figure 2.7: Measurement points of steel strip products and corresponding width error.

Following the suggestion of the engineers who work in the field, we focus on 9 process variables: *target speed of rollers, the measured speed of rollers, looper value, target force on both sides of the rollers, the measured force on the work side of rollers, measured force on the transfer side of rollers, roller gap, looper height*, and *temperature*. At each stage, the data for each process variable is a time series acquired during the production of the product. The time series consists of $1,500$ observations, which are corresponding to the $1,500$ measurement points in Figure 2.7(a). For each process variable, we take the average of the $1,500$ observations from each stage and set it as the value of the process variable

at that stage. By doing so, we construct a 9×7 process variable matrix $\boldsymbol{X}_i$ for product i, $i = 1, \ldots, 490$. Same as what has been done by (Jeong and Fang 2022), we randomly select 400 samples from the dataset to construct a sub dataset. We then apply our method to the sub dataset to identify crucial rows/columns and crucial elements. We repeat this procedure 100 times and then compute the selection accuracy and precision. Any row/columns/elements whose selection rates are higher than 0.5 are considered as important ones. The selection results are reported in Tables 2.5, 2.6, and 2.7.

Table 2.5: Selection rates of the stages (%)

Stage	1	2	3	4	5	6	7
Selection Rate	6	85	23	100	4	0	0

Table 2.5 suggests that *Stage 2* and *Stage 4* are identified as crucial stages that are responsible for the quality defects of the hot strip rolling mill. Table 2.6 indicates that the three crucial process variables are the *Looper Value*, *Roller Gap*, and *Transfer Force*. The method developed by (Jeong and Fang 2022) selected *Stages 3, 4, 6* as crucial stages and *Looper Value*, *Roller Gap*, and *Looper Height* as the important process variables. Our proposed method selects less crucial stages than the model in (Jeong and Fang 2022), and the same stage identified as crucial by both the two methods is *Stage 4*. In addition, both the methods proposed in this article and (Jeong and Fang 2022) identify *Looper Value* and *Roller Gap* as important process variables that might be responsible for the quality defects of the hot strip mill. Both the two process variables are closely related to the performance control of the looper in the hot strip mill, which is known to be the most challenging control component. The process variable *Looper Value* is used to control the tension of the steel strip between two stages. The process variable *Roller Gap* is used to control the thickness of the steel strip, which also significantly affects the real-time looper performance.

In addition to identifying the crucial stages and process variables, the proposed method can also select the important variables in each of the stages identified to be crucial. The results are presented in Table 2.7, which indicates that the only important process variable in *Stage 2* is the *Roller Gap*, and the two important process variables in *Stage 4* are the *Looper Value* and *Transfer Force*. This provides more useful information to guide the control

experts to revise the control algorithm. Unlike our model, the method proposed in (Jeong and Fang 2022) cannot distinguish the importance of the process variables in each crucial stage.

Table 2.6: Selection rates of the process variables (%)

Process variable	Selection Rate
Target speed	0
Measured speed	0
Looper value	100
Both side target force	10
Work side force	1
Transfer force	93
Roller gap	85
Looper height	0
Temperature	0

Table 2.7: Element-wise Selection Rate (%)

Method	2	4
Looper value	0	100
Transfer force	6	93
Roller gap	85	0

2.5 Conclusions

This chapter proposed a generalized matrix regression model that can be used for the quality defect diagnostics of multistage manufacturing processes with multiple identical stages such as hot strip mills in the steel-making industry and 3D printing processes in additive manufacturing. The proposed model is a new generalized matrix regression model with regularization, which is formulated as a convex optimization criterion that consists of a negative log-likelihood function, two ℓ_2 penalization terms, and an ℓ_1 penalization term. By simultaneously using both ℓ_2 and ℓ_1 penalization terms, the proposed method can inspire both group-wise and element-wise sparsity. This implies that, unlike the existing models, the proposed method can provide two levels of information. First, it provides the overall importance of each stage and process variable, which helps identify a few crucial stages and process variables that need to be investigated from the stage-level or process variable-level when revising the control algorithm. Second, it suggests the importance of the process variables in each crucial stage and the stage importance information of each crucial process variable, which provides element-level information that can be used to guide the control algorithm revision as well.

Numerical studies were conducted to validate the effectiveness of the proposed method. The results demonstrated that our proposed method outperformed the benchmark in terms of selection accuracy and precision. We believe this is because the proposed model can achieve both group-wise and element-wise selection, while the benchmark can only inspire group-wise sparsity. A real-world dataset from a hot strip steel mill was also used to evaluate the performance of the proposed method. The results indicated that the proposed method selected less crucial stages and process variables compared with the baseline model. More importantly, the proposed model can provide information regarding the process variables with top priority in each crucial stage, which can provide more information to the control experts to help them revise the feedback control algorithm to reduce the probability of producing defective products in the future.

CHAPTER

3

A SUPERVISED TENSOR DIMENSION REDUCTION-BASED PROGNOSTIC MODEL FOR APPLICATIONS WITH INCOMPLETE IMAGING DATA

3.1 Introduction

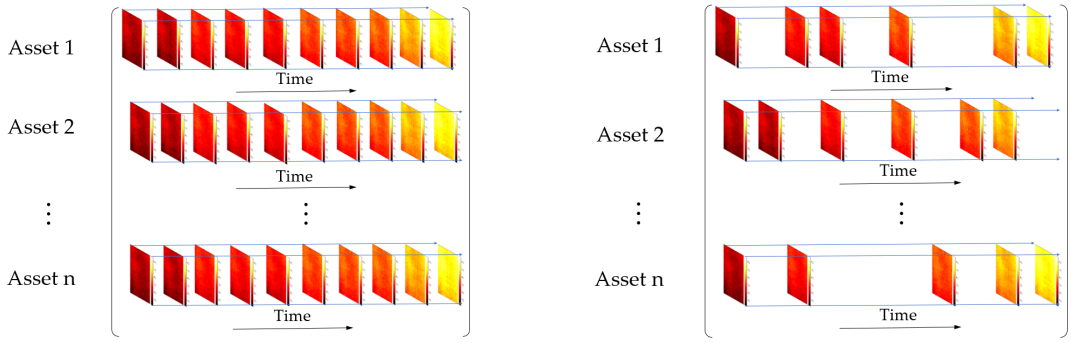
Degradation is an irreversible process of damage accumulation that results in the failure of engineering systems/assets/components (Bogdanoff and Kozin 1985). Although it is usually challenging to observe a physical degradation process, there often are some manifestations associated with degradation processes that can be monitored by sensing technology, which yields data known as degradation data/signals. Degradation data contains the health condition of engineering assets; thus, if modeled properly, they can be used to predict the assets' time-to-failure (TTF) via a process known as prognostic. Many prognostic models have been developed in the literature, most of which focus on using time series-based

degradation data (Ye et al. 2014; Ye and Chen 2014; Hong and Meeker 2010, 2013; Shu et al. 2015; Liu et al. 2013; Gebraeel et al. 2005; Wang et al. 2022). Recently, prognostic models with imaging-based degradation data have been investigated and attracted more and more attention. This is because comparing with time-series data, imaging data usually contains much richer information of the object being monitored, and imaging sensing technologies are noncontact and thus they can usually be easily deployed. One example of imaging-based degradation data is the infrared image stream that measures the change of temperature distribution of a thrust bearing during its degradation process over time (Fang et al. 2019; Aydemir and Paynabar 2019; Dong et al. 2021; Jiang et al. 2022a; Wang et al. 2021). Another example is the images used to measure the performance degradation of infrared systems such as rotary-wing drones (Dong et al. 2021).

The existing imaging-based prognostic methods include deep learning-based models and statistical learning methods. Examples of the deep learning-based models designed for TTF prediction using imaging data include the ones developed by (Aydemir and Paynabar 2019; Yang et al. 2021; Dong et al. 2021; Jiang et al. 2022a,b, 2023). Although these models have worked relatively well, they usually provide point estimations of failure times, and it is challenging for them to quantify the uncertainty of predicted TTFs (e.g., providing a failure time distribution). This limits their applicability since the subsequent decision-making analysis such as maintenance/inventory/logistic optimization requires prognostic models to provide a distribution of the predicted TTF. Also, deep learning-based prognostic models often require a relatively large number of historical samples for model training, which cannot be satisfied by many real-world applications with limited historical data. One example of statistical learning methods for image-based prognostic is the penalized (log)-location-scale (LLS) tensor regression proposed by (Fang et al. 2019). The model first employs multilinear principal component analysis (MPCA) (Lu et al. 2008) to reduce the dimension of high-dimensional imaging-based degradation data, which yields a low-dimensional feature tensor for each asset. Next, it constructs a prognostic model by regressing an asset's TTF against its low-dimensional feature tensor using LLS regression. In the same article, (Fang et al. 2019) have also proposed several benchmarking prognostic models that use imaging-based degradation data for TTF prediction. These models also first employ a dimension reduction method such as functional principal component analysis (FPCA) (Ramsay and Silverman 2005), principal component analysis (PCA) (Abdi and Williams 2010), or B-Spline (Prautzsch et al. 2002), to reduce the dimension of degradation data and then use low-dimensional features to build an LLS regression model for prognostic. Although the aforementioned statistical learning-based prognostic models can provide a

distribution for the predicted TTF, and their effectiveness have been well investigated, they share two common limitations.

The first limitation is that they assume imaging-based degradation data (including historical data for model training and real-time data for model test) are complete, which means images from all the assets should be collected continuously and regularly with the same sampling time interval (see Figure 3.1 (a) for an example). In reality, however, engineering assets often operate in harsh environments that significantly impact the quality of collected data due to errors in data acquisition, communication, read/write operations, etc. As a result, degradation images often contain significant levels of missing observations, which is known as incomplete/missing imaging data (see Figure 3.1 (b) for an example). Such data incompleteness poses a significant challenge for the parameter estimation of existing statistical learning-based prognostic models.



(a) Complete Image Streams (b) Incomplete Image Streams

Figure 3.1: Degradation stream images with and without missing data.

The second common limitation for the existing statistical learning-based prognostic models for applications with imaging data is that they employ unsupervised dimension reduction methods for feature extraction, so there is no guarantee that the extracted features are effective for the subsequent TTF prediction. Specifically, they first use unsupervised dimension reduction methods such as FPCA, PCA, and B Spline to extract features, which are then used to construct prognostic models. Since feature extraction and prognostic model construction are two sequential steps, and no TTF information gets involved in the feature extraction process, it is possible that the extracted features may not be most suitable for predicting TTFs.

To address the aforementioned challenges, this chapter proposes a supervised dimension reduction-based prognostic model that uses an asset's incomplete degradation images to predict its TTF. Similar to the existing statistical learning-based prognostic models, the proposed model also consists of two steps: feature extraction and prognostic model construction. However, unlike the existing models, feature extraction in this article is achieved by developing a new supervised tensor dimension reduction method, which uses historical TTFs to supervise the feature extraction process such that the extracted features are more effective for the subsequent TTF prediction. In addition, unlike the existing unsupervised dimension reduction methods that only work for complete imaging data, the proposed supervised dimension reduction method works for both complete and incomplete degradation image streams.

The proposed supervised dimension reduction method works as follows: First, it detects a low-dimensional tensor subspace in which the high-dimensional degradation image streams are embedded. This is achieved by constructing an optimization criterion that comprises a feature extraction term and a regression term. The first term extracts low-dimensional features from complete/incomplete degradation image streams of training assets, and the second term builds the connection between these assets' TTFs and the extracted features using LLS regression. LLS regression has been widely used in reliability engineering and survival analysis. It includes a variety of TTF distributions, such as (log)normal, (log)logistic, smallest extreme value, and Weibull, etc., which cover most of the TTF distributions in reality (Doray 1994). Since historical TTFs are used to supervise the feature extraction process, it is expected that the extracted features are more effective for the subsequent prognostic. Solving the optimization criterion of the proposed supervised tensor dimension reduction method yields a set of tensor basis matrices that span the low-dimensional tensor subspace for dimension reduction. We then expand both the historical degradation images in the training data set and real-time degradation images from an asset operating in the field (i.e., test data) using the set of tensor basis matrices to extract the low-dimensional tensor features of the training assets and the test asset. The TTFs of the training assets are then regressed against their tensor features using LLS regression, and the parameters are estimated using maximum likelihood estimation. After that, the tensor features of the test asset are fed into the LLS regression model, and its TTF distribution is predicted.

To solve the optimization criterion of the proposed supervised dimension reduction method, we will first transfer the criterion into a block multiconvex problem. Next, we will propose a block updating algorithm, which cyclically optimizes one block parameters while

keeping other blocks fixed until convergence. In addition, we will demonstrate that when TTFs follow normal or lognormal distributions, each sub problem of the block updating algorithm has a closed-form solution, no matter the degradation image streams are complete or incomplete.

The rest of this chapter is orginized as follows. Section 3.2 presents the supervised tensor dimension reduction-based prognostic method. Section 3.3 introduces the block updating algorithm and closed-form solutions when TTFs follow normal/lognormal distributions. Sections 3.5 and 3.6 validate the effectiveness of the proposed prognostic model using a simulated dataset and data from a rotating machinery, respectively. Section 3.7 concludes.

3.2 The Methodology

In this section, we will introduce the proposed supervised tensor dimension reduction-based prognostic model for applications with incomplete imaging data. In Subsection 3.1, we will present some basic tensor notations and definitions. Subsection 3.2 introduces the supervised tensor dimension reduction method. In Subsection 3.3, we will discuss the construction of prognostic model and how to predict the TTF of an asset operating in the field using its real-time degradation imaging data.

3.2.1 Preliminaries

In this section, we introduce some basic notations and definitions of tensor operations that are used throughout the article. The *order* of a tensor is the number of dimensions, also known as ways or modes. Vectors (1 order tensors) are denoted by lowercase boldface letters, e.g., $\boldsymbol{s}$. Matrices (2 order tensors) are denoted by boldface uppercase letters, e.g., $\boldsymbol{S}$. Higher-order tensors (order is 3 or larger) are denoted by calligraphic letters, e.g., $\mathscr{S}$. Indices are denoted by lowercase letters whose range is from 1 to the uppercase letter of the index, e.g., $n = 1, 2, \ldots, N$. An Nth-order tensor is denoted as $\mathscr{S} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, where I_n represents the nth mode of $\mathscr{S}$. The $(i_1, i_2, \ldots, i_N)$th entry of $\mathscr{S} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is denoted by $s_{i_1, i_2, \ldots, i_n}$. A fiber of $\mathscr{S}$ 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 $\mathscr{S}$, denoted by $vec(\mathscr{S})$, stacks all the entries of $\mathscr{S}$ into a column vector. The mode-n *matricization* of a tensor $\mathscr{S} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is denoted by $\boldsymbol{S}_{(n)}$, which arranges the mode-n fibers to be the columns of the resulting matrix. The nth mode product of a tensor $\mathscr{S} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and a matrix $\boldsymbol{U}_n \in \mathbb{R}^{J_n \times I_n}$, denoted by $\mathscr{S} \times_n \boldsymbol{U}_n$, is a tensor whose entry is $(\mathscr{S} \times_n \boldsymbol{U}_n)_{i_1, \ldots, i_{n-1}, j_n, i_{n+1}, \ldots, i_N} = \Sigma_{I_n=1}^{I_N} s_{i_1, \ldots, i_N} u_{j, i_n}$. 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 $mp \times nq$ block matrix:

$$\boldsymbol{A}\otimes\boldsymbol{B} = \begin{bmatrix} \boldsymbol{a}_{11}\boldsymbol{B} & \dots & \boldsymbol{a}_{1n}\boldsymbol{B} \\ \vdots & \ddots & \vdots \\ \boldsymbol{a}_{m1}\boldsymbol{B} & \dots & \boldsymbol{a}_{mn}\boldsymbol{B} \end{bmatrix}.$$

If $\boldsymbol{A}$ and $\boldsymbol{B}$ have the same number of columns $n = q$, then the *Khatri-Rao* product is defined as the $mp \times n$ column-wise *Kronecker* product: $\boldsymbol{A}\odot\boldsymbol{B} = [\boldsymbol{a}_1\otimes\boldsymbol{b}_1 \quad \boldsymbol{a}_2\otimes\boldsymbol{b}_2 \quad \cdots \quad \boldsymbol{a}_n\otimes\boldsymbol{b}_n]$. If $\boldsymbol{a}$ and $\boldsymbol{b}$ are vectors, then $\boldsymbol{A}\otimes\boldsymbol{B} = \boldsymbol{A}\odot\boldsymbol{B}$. More details about tensor notations and operators can be found in Kolda and Bader (2009).

3.2.2 The Supervised Tensor Dimension Reduction Method

We assume that there exists a historical data set for model training. The data set consists of the degradation image streams of M failed assets along with their TTFs, which are denoted as $\mathcal{X}_m \in \mathbb{R}^{I_1\times I_2\times I_3}$ and $y_m \in \mathbb{R}$, respectively, where $m = 1, 2, \dots, M$. For the convenience of introducing the dimension reduction method, we convert the 3D degradation image streams from all the M assets to a 4D tensor $\mathcal{X} \in \mathbb{R}^{I_1\times I_2\times I_3\times M}$, where the sample size M is the 4th mode. Similarly, we let $\boldsymbol{y} = (y_1, \dots, y_M)^\top \in \mathbb{R}^{M\times 1}$ be the vector containing all the TTFs of the M assets.

Out of the $I_1 \times I_2 \times I_3 \times M$ entries of $\mathcal{X}$, we use a subset $\Omega \subseteq \{(i_1, i_2, i_3, m), 1 \le i_1 \le I_1, 1 \le i_2 \le I_2, 1 \le i_3 \le I_3, 1 \le m \le M\}$ to denote the indices of the missing ones. To model the missing data, we define a projection operator $\mathcal{P}_\Omega(\cdot)$ as follows:

$$\mathcal{P}_\Omega(\mathcal{X})_{(i_1,i_2,i_3,m)} = \begin{cases} \mathcal{X}_{(i_1,i_2,i_3,m)}, & \text{if } (i_1, i_2, i_3, m) \notin \Omega, \\ 0, & \text{if } (i_1, i_2, i_3, m) \in \Omega, \end{cases} \tag{3.1}$$

where $\mathcal{X}_{(i_1,i_2,i_3,m)}$ is the (i_1, i_2, i_3, m)-th entry of the 4D tensor $\mathcal{X} \in \mathbb{R}^{I_1\times I_2\times I_3\times M}$. To recover the missing entries in tensor $\mathcal{X}$, we may employ the following Tucker decomposition-based tensor completion method (Filipović and Jukić 2015; Liu et al. 2012; Xu et al. 2013):

$$\min_{\mathcal{S},\boldsymbol{U}_1,\boldsymbol{U}_2,\boldsymbol{U}_3} \; ||\mathcal{P}_\Omega(\mathcal{X} - \mathcal{S}\times_1\boldsymbol{U}_1^\top\times_2\boldsymbol{U}_2^\top\times_3\boldsymbol{U}_3^\top)||_F^2. \tag{3.2}$$

where $||\cdot||_F^2$ is the Frobenius norm, $\boldsymbol{U}_1 \in \mathbb{R}^{P_1\times I_1}$, $\boldsymbol{U}_2 \in \mathbb{R}^{P_2\times I_2}$, $\boldsymbol{U}_3 \in \mathbb{R}^{P_3\times I_3}$ are three factor matrices, $\mathcal{S} \in \mathbb{R}^{P_1\times P_2\times P_3\times M}$ is the low-dimensional core tensor, and $\times_n$ is the n-mode product of a tensor with a matrix. The tensor completion criterion (3.2) can be seen as an

unsupervised dimension reduction method for tensor data with missing entries. This is because the degradation image tensor $\mathscr{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times M}$ is a 4-order tensor, which resides in the tensor (multilinear) space $\mathbb{R}^{I_1} \otimes \mathbb{R}^{I_2} \otimes \mathbb{R}^{I_3} \otimes \mathbb{R}^{M}$, where $\mathbb{R}^{I_1}, \mathbb{R}^{I_2}, \mathbb{R}^{I_3}, \mathbb{R}^{M}$ are the 4 vector (linear) spaces; $\mathscr{S} \in \mathbb{R}^{P_1 \times P_2 \times P_3 \times M}$ can be seen as a feature tensor that resides in the tensor space $\mathbb{R}^{P_1} \otimes \mathbb{R}^{P_2} \otimes \mathbb{R}^{P_3} \otimes \mathbb{R}^{M}$. Usually, we have $P_1 \ll I_1$, $P_2 \ll I_2$, and $P_3 \ll I_3$ for degradation imaging data due to the high spatio-temporal correlation among pixels. This implies that the dimension of the image stream from the mth asset is reduced from $\mathbb{R}^{I_1 \times I_2 \times I_3}$ to $\mathbb{R}^{P_1 \times P_2 \times P_3}$, where $m = 1, \ldots, M$.

Although criterion (3.2) can be seen as a dimension reduction method, there is no guarantee that the extracted low-dimensional feature tensor $\mathscr{S}$ is effective for the subsequent TTF prediction. To address this challenge, we propose the following *supervised dimension reduction* method by combining a tensor completion term and an LLS regression term:

$$\min_{\boldsymbol{U}_1, \boldsymbol{U}_2, \boldsymbol{U}_3, \sigma, \boldsymbol{\beta}_1, \beta_0, \mathscr{S}} \alpha ||\mathscr{P}_{\Omega}(\mathscr{X} - \mathscr{S} \times_1 \boldsymbol{U}_1^{\top} \times_2 \boldsymbol{U}_2^{\top} \times_3 \boldsymbol{U}_3^{\top})||_F^2 + (1-\alpha)\ell(\frac{\boldsymbol{y} - \mathbf{1}_M \beta_0 - \boldsymbol{S}_{(4)}\boldsymbol{\beta}_1}{\sigma}), \quad (3.3)$$

where $\boldsymbol{y} \in \mathbb{R}^{M \times 1}$ is the vector containing all the TTFs of the M assets in the training data set. The matrix $\boldsymbol{S}_{(4)} \in \mathbb{R}^{M \times (P_1 \times P_2 \times P_3)}$ is the mode-4 matricization of the low-dimensional feature tensor $\mathscr{S}$, the mth row of which represents the vectorization of the mth asset's feature tensor, $m = 1, \ldots, M$. β_0 is the intercept, and $\boldsymbol{\beta}_1 \in \mathbb{R}^{(P_1 \times P_2 \times P_3) \times 1}$ is the regression coefficient vector. $\mathbf{1}_m \in \mathbb{R}^{M \times 1}$ is an $M \times 1$ vector whose entries are all ones. $\ell(\cdot)$ is the negative log-likelihood function of a location-scale distribution. For example, if TTFs follow normal distributions, then $\ell(\frac{\boldsymbol{y} - \mathbf{1}_M \beta_0 - \boldsymbol{S}_{(4)}\boldsymbol{\beta}_1}{\sigma}) = \frac{M}{2}\log 2\pi + M\log\sigma + \frac{1}{2}\Sigma_{m=1}^{M}\omega_m^2$, where $\omega_m = \frac{y_m - \beta_0 - \boldsymbol{s}_{(4)}^m \boldsymbol{\beta}_1}{\sigma}$, where $\boldsymbol{s}_{(4)}^m$ is the mth row of $\boldsymbol{S}_{(4)}$, and y_m is the TTF of asset m; if TTFs follow logistic distributions, then $\ell(\frac{\boldsymbol{y} - \mathbf{1}_M \beta_0 - \boldsymbol{S}_{(4)}\boldsymbol{\beta}_1}{\sigma}) = M\log\sigma - \Sigma_{m=1}^{M}\omega_m + 2\Sigma_{m=1}^{M}\log(1 + \exp(\omega_m))$; if TTFs follow small extreme value (SEV) distributions, then $\ell(\frac{\boldsymbol{y} - \mathbf{1}_M \beta_0 - \boldsymbol{S}_{(4)}\boldsymbol{\beta}_1}{\sigma}) = n\log\sigma - \Sigma_{m=1}^{M}\omega_m + \Sigma_{m=1}^{M}\exp(\omega_m)$. For assets whose TTFs follow log-location-scale distributions, we may transfer them to the corresponding location-scale distributions by taking their logarithm such that criterion (3.3) can still be used. For example, log-normal, log-logistics, and Weibull distributions can be transferred to normal, logistics, and SEV distributions, respectively. $\alpha \in [0, 1]$ is a weight and $||\cdot||_F^2$ is the Frobenius norm.

In criterion (3.3), the first term $||\mathscr{P}_{\Omega}(\mathscr{X} - \mathscr{S} \times_1 \boldsymbol{U}_1^{\top} \times_2 \boldsymbol{U}_2^{\top} \times_3 \boldsymbol{U}_3^{\top})||_F^2$ is tensor completion from (3.2), which reduces the dimension of high-dimensional incomplete degradation image streams and extracts low-dimensional tensor features. The second term $\ell(\frac{\boldsymbol{y} - \mathbf{1}_M \beta_0 - \boldsymbol{S}_{(4)}\boldsymbol{\beta}_1}{\sigma})$ is LLS regression, which regresses each asset's TTF against its tensor features extracted by the first term. By jointly optimizing the two terms, it is expected that the extracted features are ef-

fective for TTF prediction. However, it is challenging to solve criterion (3.3) since it is neither convex nor block multi-convex. An optimization problem is block multi-convex when its feasible set and objective function are generally non-convex but convex in each block of variables (Xu and Yin 2013). Thus, to simplify the development of optimization algorithms for model parameter estimation, we first transform criterion (3.3) to a block multi-convex one. Specifically, we apply the following re-parameterization: $\tilde{\sigma} = 1/\sigma, \tilde{\beta}_0 = \beta_0/\sigma, \tilde{\boldsymbol{\beta}}_1 = \boldsymbol{\beta}_1/\sigma$. As a result, criterion (3.3) can be re-expressed as follows:

$$\min_{\boldsymbol{U}_1,\boldsymbol{U}_2,\boldsymbol{U}_3,\tilde{\sigma},\tilde{\boldsymbol{\beta}}_1,\tilde{\beta}_0,\boldsymbol{S}_{(4)}} \alpha||\mathscr{P}_\Omega(\mathscr{X} - \mathscr{S} \times_1 \boldsymbol{U}_1^\top \times_2 \boldsymbol{U}_2^\top \times_3 \boldsymbol{U}_3^\top)||_F^2 + (1-\alpha)\ell(\tilde{\sigma}\boldsymbol{y} - \mathbf{1}_M\tilde{\beta}_0 - \boldsymbol{S}_{(4)}\tilde{\boldsymbol{\beta}}_1), \quad (3.4)$$

where, $\ell(\tilde{\sigma}\boldsymbol{y} - \mathbf{1}_M\tilde{\beta}_0 - \boldsymbol{S}_{(4)}\tilde{\boldsymbol{\beta}}_1) = \frac{M}{2}\log 2\pi - M\log\tilde{\sigma} + \frac{1}{2}\Sigma_{m=1}^M \tilde{\omega}_m^2$ for TTFs following normal distributions, and $\tilde{\omega}_m = \tilde{\sigma}\, y_m - \tilde{\beta}_0 - \boldsymbol{s}_{(4)}^m\tilde{\boldsymbol{\beta}}_1$, where $\boldsymbol{s}_{(4)}^m$ is the mth row of $\boldsymbol{S}_{(4)}$ and y_m is the TTF of asset m; $\ell(\tilde{\sigma}\boldsymbol{y} - \mathbf{1}_M\tilde{\beta}_0 - \boldsymbol{S}_{(4)}\tilde{\boldsymbol{\beta}}_1) = -M\log\tilde{\sigma} - \Sigma_{m=1}^M\tilde{\omega}_m + 2\Sigma_{m=1}^M\log(1+\exp(\tilde{\omega}_m))$ for TTFs following logistics distributions, and $\ell(\tilde{\sigma}\boldsymbol{y} - \mathbf{1}_M\tilde{\beta}_0 - \boldsymbol{S}_{(4)}\tilde{\boldsymbol{\beta}}_1) = -n\log\tilde{\sigma} - \Sigma_{m=1}^M\tilde{\omega}_m + \Sigma_{m=1}^M\exp(\tilde{\omega}_m)$ for TTFs following SEV distributions.

The optimization algorithm to solve criterion (3.4), the value of the weight α, and the dimension of the low-dimensional tensor subspace $\{P_1, P_2, P_3\}$ will be discussed in Sections 3.3 and 3.4. Solving the optimization criterion (3.4) using historical training data yields a set of basis matrices $\hat{\boldsymbol{U}}_1 \in \mathbb{R}^{P_1\times I_1}, \hat{\boldsymbol{U}}_2 \in \mathbb{R}^{P_2\times I_2}, \hat{\boldsymbol{U}}_3 \in \mathbb{R}^{P_3\times I_3}$, which contains P_1 basis vectors of the 1-mode linear space $\mathbb{R}^{I_1}$, P_2 basis vectors of the 2-mode linear space $\mathbb{R}^{I_2}$, and P_3 basis vectors of the 3-mode linear space $\mathbb{R}^{I_3}$, respectively. The three linear subspaces form the low-dimensional tensor subspace $\mathbb{R}^{P_1} \otimes \mathbb{R}^{P_2} \otimes \mathbb{R}^{P_3}$ detected by the proposed supervised dimension reduction method.

One of the assumptions of the proposed supervised tensor dimension reduction method in criterion (3.4) is that the TTF of the asset follows a distribution from the LLS family. This is reasonable since the LLS family includes a variety of TTF distributions, such as (log)normal, (log)logistic, smallest extreme value, and Weibull, etc., which cover most of the TTF distributions in engineering applications (Doray 1994). Another assumption is that there is a linear relationship between the location parameter and predictors (i.e., degradation signals or their features in this article). Specifically, the location parameters in criterion (3.3) are expressed as $\mathbf{1}_M\beta_0 + \boldsymbol{S}_{(4)}\boldsymbol{\beta}_1$, which are linear weighted combinations of the rows of $\boldsymbol{S}_{(4)}$). This assumption is widely used in LLS regression (Doray 1994; Fang et al. 2019). However, if a simple linear weighted combination is not adequate to characterize the association between the location parameter and degradation signal features, a high-order polynomial relationship can be constructed (Hastie et al. 2009). By doing so, we

can model a more complex association between the location parameter and features. More importantly, the incorporation of high-order polynomial terms into the proposed supervised tensor dimension reduction method does not affect the effectiveness of the optimization algorithms for parameter estimation to be discussed in Sections 3.3 and 3.4.

3.2.3 Prognostic Model Construction and Real-Time TTF Prediction

In this subsection, we discuss how to build a prognostic model based on the supervised dimension reduction method proposed in Section 3.2.2, and how to predict the TTF distribution of an asset operating in the field using its real-time degradation image data.

Similar to Section 3.2.2, we denote the training data set as $\{\mathcal{X}_m \in \mathbb{R}^{I_1 \times I_2 \times D_m}, y_m\}_{m=1}^M$, where M is the number of failed assets in the training data set. Notice that D_m might not be the same as $D_{m'}$ for two assets m and m', $m = 1,\ldots,M, m' = 1,\ldots,M, m \neq m'$. This is because different asset's failure times (i.e., TTFs) are different, and usually no image data can be collected beyond an asset's failure time since the asset is stopped for maintenance or replace once it is failed. In addition to the training data, we denote the degradation image stream of a test asset by time t as $\mathcal{X}_t \in \mathbb{R}^{I_1 \times I_2 \times I_t}$. The objectives of this subsection include 1) constructing a prognostic model and estimating its parameters using $\{\mathcal{X}_m, y_m\}_{m=1}^M$ in the training data set, and 2) using the estimated prognostic model to predict the TTF (denoted as $\hat{y}_t$) of the test asset based on its degradation image stream $\mathcal{X}_t$.

We first use the proposed supervised dimension reduction method to extract low-dimensional features of both the training and test assets. Specifically, as discussed in Section 3.2.2, we first construct a 4D tensor $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times M}$ using the degradation image streams of the training assets, where $I_3 = \max(\{D_m\}_{m=1}^M)$. Note that $\mathcal{X}$ is an incomplete tensor no matter the image streams from the training assets are complete or incomplete. This is because the TTFs of training assets are different, and thus not all the training assets have I_3 images. To detect the low-dimensional tensor subspace in which the high-dimensional degradation images are embedded, we solve optimization criterion (3.4) by using training data $\{\mathcal{X}, \boldsymbol{y}\}$, where $\boldsymbol{y} = (y_1,\ldots,y_M)^\top$. This yields basis matrices $\hat{\boldsymbol{U}}_1 \in \mathbb{R}^{P_1 \times I_1}, \hat{\boldsymbol{U}}_2 \in \mathbb{R}^{P_2 \times I_2}, \hat{\boldsymbol{U}}_3 \in \mathbb{R}^{P_3 \times I_3}$, which form the low-dimensional tensor subspace $\mathbb{R}^{P_1} \otimes \mathbb{R}^{P_2} \otimes \mathbb{R}^{P_3}$. To extract the low-dimensional features of the training and test assets, we expand the image streams in the low-dimensional tensor subspace $\mathbb{R}^{P_1} \otimes \mathbb{R}^{P_2} \otimes \mathbb{R}^{P_3}$ using the basis matrices $\hat{\boldsymbol{U}}_1, \hat{\boldsymbol{U}}_2, \hat{\boldsymbol{U}}_3$. This is achieved by solving the following optimization criteria:

$$\hat{\mathcal{S}}_m = \arg\min_{\boldsymbol{S}_m} ||\mathcal{P}_\Omega(\mathcal{X}_m - \mathcal{S}_m \times_1 \boldsymbol{U}_1^\top \times_2 \boldsymbol{U}_2^\top \times_3 \boldsymbol{U}_3^\top)||_F^2. \tag{3.5}$$

$$\hat{\mathscr{S}}_t = \arg\min_{\boldsymbol{S}_t} ||\mathscr{P}_\Omega(\mathscr{X}_t - \mathscr{S}_t \times_1 \boldsymbol{U}_1^\top \times_2 \boldsymbol{U}_2^\top \times_3 \boldsymbol{U}_3^\top)||_F^2, \tag{3.6}$$

where $\{\hat{\mathscr{S}}_m\}_{m=1}^M$ are the low-dimensional feature tensors of the M assets in the training data set, and $\hat{\mathscr{S}}_t$ is the low-dimensional feature tensor of the test asset.

Next, we construct a prognostic model using the low-dimensional feature tensors of the M assets in the training data set (i.e., $\{\hat{\mathscr{S}}_m\}_{m=1}^M$) along with their TTFs $\{y_m\}_{m=1}^M$. Specifically, we build the following LLS regression model:

$$y_m = \gamma_0 + vec(\hat{\mathscr{S}}_m)^\top \boldsymbol{\gamma}_1 + \sigma\epsilon_m, \tag{3.7}$$

where $vec(\hat{\mathscr{S}}_m)$ is the vectorization of $\hat{\mathscr{S}}_m$. $\gamma_0 \in \mathbb{R}$ and $\boldsymbol{\gamma}_1 \in \mathbb{R}^{(P_1\times P_2\times P_3)\times 1}$ 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 (3.7) can be estimated by solving the following optimization problem:

$$\min_{\boldsymbol{y},\gamma_0,\boldsymbol{\gamma}_1,\sigma} \ell\left(\frac{\boldsymbol{y} - \mathbf{1}_M\gamma_0 - \hat{\boldsymbol{S}}_{(4)}\boldsymbol{\gamma}_1}{\sigma}\right), \tag{3.8}$$

where $\ell(\cdot)$ is the negative log-likelihood function of a location-scale distribution,$\boldsymbol{y} = (y_1, y_2, \cdots, y_m)^\top$ and $\hat{\boldsymbol{S}}_{(4)} = (vec(\hat{\mathscr{S}}_1)^\top, vec(\hat{\mathscr{S}}_2)^\top, \cdots, vec(\hat{\mathscr{S}}_M)^\top)^\top$, and $\ell(\cdot)$ is the negative log-likelihood function. We conduct the following reparameterization to transform the optimization to be a convex one: $\tilde{\sigma} = 1/\sigma, \tilde{\gamma}_0 = \gamma_0/\sigma, \tilde{\boldsymbol{\gamma}}_1 = \boldsymbol{\gamma}_1/\sigma$:

$$\{\hat{\tilde{\gamma}}_0, \hat{\tilde{\boldsymbol{\gamma}}}_1, \hat{\tilde{\sigma}}\} = \arg\min_{\tilde{\gamma}_0,\tilde{\boldsymbol{\gamma}}_1,\tilde{\sigma}} \ell(\tilde{\sigma}\boldsymbol{y} - \mathbf{1}_M\tilde{\gamma}_0 - \hat{\boldsymbol{S}}_{(4)}\tilde{\boldsymbol{\gamma}}_1). \tag{3.9}$$

Solving (3.9) provides the estimated parameters $\{\hat{\tilde{\gamma}}_0, \hat{\tilde{\boldsymbol{\gamma}}}_1, \hat{\tilde{\sigma}}\}$, which can be transformed 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 is $\hat{y}_m \sim LLS(\hat{\gamma}_0 + vec(\hat{\mathscr{S}}_m)^\top\hat{\boldsymbol{\gamma}}_1, \hat{\sigma})$, where $\hat{\gamma}_0 + vec(\hat{\mathscr{S}}_m)^\top\hat{\boldsymbol{\gamma}}_1$ and $\hat{\sigma}$ are respectively the estimated location and scale parameters.

Finally, we feed the extracted low-dimensional feature tensor of the test asset into the estimated LLS regression model to predict the asset's TTF distribution: $\hat{y}_t \sim LLS(\hat{\gamma}_0 + vec(\hat{\mathscr{S}}_t)^\top\hat{\boldsymbol{\gamma}}_1, \hat{\sigma})$.

3.3 The Optimization Algorithm

In this section, we discuss how to solve the supervised tensor dimension reduction method proposed in section 3.2.2. In subsection 3.3.1, we develop a Block Updating Algorithm to solve criterion (3.4). The algorithm splits the unknown parameters in criterion (3.4) into several blocks, and it cyclically optimizes one block parameter while keeping other blocks fixed until convergence. The sub-optimization problem for each block is convex, so the convergence of the block updating algorithm is guaranteed. In subsection 3.3.2, we discuss the initialization of the proposed algorithm and hyperparameter tuning.

3.3.1 The Block Updating Algorithm

The Block Updating Algorithm first splits the unknown parameters in criterion (3.4) into 5 blocks, i.e., $\boldsymbol{U}_1, \boldsymbol{U}_2, \boldsymbol{U}_3, \mathscr{S}$ and $\{\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}_1, \tilde{\sigma}\}$. It then cyclically optimizes one block of parameters each time while keeping other blocks fixed.

Specifically, at the kth iteration, $\boldsymbol{U}_1$ is updated by solving the following optimization problem while keeping other blocks (i.e., $\boldsymbol{U}_2^{k-1}, \boldsymbol{U}_3^{k-1}, \tilde{\sigma}^{k-1}, \tilde{\beta}_0^{k-1}, \tilde{\boldsymbol{\beta}}_1^{k-1}, \mathscr{S}^{k-1}$) fixed:

$$\begin{aligned}\boldsymbol{U}_1^k &= \arg\min_{\boldsymbol{U}_1} \ \alpha||\mathscr{P}_\Omega(\mathscr{X} - \mathscr{S}^{k-1} \times_1 \boldsymbol{U}_1^\top \times_2 \boldsymbol{U}_2^{k-1^\top} \times_3 \boldsymbol{U}_3^{k-1^\top})||_F^2 + (1-\alpha)\ell(\tilde{\sigma}^{k-1}, \tilde{\beta}_0^{k-1}, \tilde{\boldsymbol{\beta}}_1^{k-1}, \boldsymbol{S}_{(4)}^{k-1}) \\ &= \arg\min_{\boldsymbol{U}_1} \ ||\mathscr{P}_\Omega(\mathscr{X} - \mathscr{S}^{k-1} \times_1 \boldsymbol{U}_1^\top \times_2 \boldsymbol{U}_2^{k-1^\top} \times_3 \boldsymbol{U}_3^{k-1^\top})||_F^2\end{aligned} \tag{3.10}$$

Similarly, the remaining blocks are updated as follows:

$$\begin{aligned}\boldsymbol{U}_2^k &= \arg\min_{\boldsymbol{U}_2} \ \alpha||\mathscr{P}_\Omega(\mathscr{X} - \mathscr{S}^{k-1} \times_1 \boldsymbol{U}_1^{k^\top} \times_2 \boldsymbol{U}_2^\top \times_3 \boldsymbol{U}_3^{k-1^\top})||_F^2 + (1-\alpha)\ell(\tilde{\sigma}^{k-1}, \tilde{\beta}_0^{k-1}, \tilde{\boldsymbol{\beta}}_1^{k-1}, \boldsymbol{S}_{(4)}^{k-1}) \\ &= \arg\min_{\boldsymbol{U}_2} \ ||\mathscr{P}_\Omega(\mathscr{X} - \mathscr{S}^{k-1} \times_1 \boldsymbol{U}_1^{k^\top} \times_2 \boldsymbol{U}_2^\top \times_3 \boldsymbol{U}_3^{k-1^\top})||_F^2\end{aligned} \tag{3.11}$$

$$\begin{aligned}\boldsymbol{U}_3^k &= \arg\min_{\boldsymbol{U}_3} \ \alpha||\mathscr{P}_\Omega(\mathscr{X} - \mathscr{S}^{k-1} \times_1 \boldsymbol{U}_1^{k^\top} \times_2 \boldsymbol{U}_2^{k^\top} \times_3 \boldsymbol{U}_3^\top)||_F^2 + (1-\alpha)\ell(\tilde{\sigma}^{k-1}, \tilde{\beta}_0^{k-1}, \tilde{\boldsymbol{\beta}}_1^{k-1}, \boldsymbol{S}_{(4)}^{k-1}) \\ &= \arg\min_{\boldsymbol{U}_3} \ ||\mathscr{P}_\Omega(\mathscr{X} - \mathscr{S}^{k-1} \times_1 \boldsymbol{U}_1^{k^\top} \times_2 \boldsymbol{U}_2^{k^\top} \times_3 \boldsymbol{U}_3^\top)||_F^2\end{aligned} \tag{3.12}$$

$$\begin{aligned}\{\tilde{\sigma}^k, \tilde{\beta}_0^k, \tilde{\beta}_1^k\} &= \arg\min_{\tilde{\sigma}^k, \tilde{\beta}_0, \tilde{\beta}_1} \quad \alpha||\mathscr{P}_\Omega(\mathscr{X} - \mathscr{S}^{k-1} \times_1 \boldsymbol{U}_1^{k\top} \times_2 \boldsymbol{U}_2^{k\top} \times_3 \boldsymbol{U}_3^{k\top})||_F^2 + (1-\alpha)\ell(\tilde{\sigma}, \tilde{\beta}_0, \tilde{\boldsymbol{\beta}}_1, \boldsymbol{S}_{(4)}^{k-1}) \\ &= \arg\min_{\tilde{\sigma}^k, \tilde{\beta}_0, \tilde{\beta}_1} \quad \ell(\tilde{\sigma}, \tilde{\beta}_0, \tilde{\boldsymbol{\beta}}_1, \boldsymbol{S}_{(4)}^{k-1})\end{aligned} \tag{3.13}$$

$$\mathscr{S}^k = \arg\min_{\mathscr{S}} \ \alpha||\mathscr{P}_\Omega(\mathscr{X} - \mathscr{S} \times_1 \boldsymbol{U}_1^{k\top} \times_2 \boldsymbol{U}_2^{k\top} \times_3 \boldsymbol{U}_3^{k\top})||_F^2 + (1-\alpha)\ell(\tilde{\sigma}^k, \tilde{\beta}_0^k, \tilde{\boldsymbol{\beta}}_1^k, \boldsymbol{S}_{(4)}) \tag{3.14}$$

We summarize the Block Updating Algorithm in Algorithm 1 below. The convergence criterion can be set as $\Psi(\boldsymbol{U}_1^k, \boldsymbol{U}_2^k, \boldsymbol{U}_3^k, \tilde{\sigma}^k, \tilde{\beta}_0^k, \tilde{\boldsymbol{\beta}}_1^k, \mathscr{S}^k) - \Psi(\boldsymbol{U}_1^{k+1}, \boldsymbol{U}_2^{k+1}, \boldsymbol{U}_3^{k+1}, \tilde{\sigma}^{k+1}, \tilde{\beta}_0^{k+1}, \tilde{\boldsymbol{\beta}}_1^{k+1}, \mathscr{S}^{k+1}) < \epsilon$, where Ψ is the value of the objective function in criterion (3.4), and ϵ is a small number. It is easy to show that sub problems (3.10), (3.11), and (3.12) are convex. For normal, logistic, and SEV distributions, their negative log-likelihood functions $\ell(\cdot)$ are also convex, so objective functions (3.13) and (3.14) are convex as well. As a result, the Block Updating Algorithm converges to a stationary point of criterion (3.4).

Algorithm 1: Block Updating Algorithm for solving criterion (3.4)

1. **Input:** Tensor $\mathscr{X}$ constructed from the (incomplete) degradation image streams of M assets and the TTF vector $\boldsymbol{y}$; the dimension of the low-dimensional tensor subspace $\{P_1, P_2, P_3\}$
2. **Initialization:** Initialize $(\boldsymbol{U}_1^0, \boldsymbol{U}_2^0, \boldsymbol{U}_3^0, \tilde{\sigma}^0, \tilde{\beta}_0^0, \tilde{\boldsymbol{\beta}}_1^0, \mathscr{S}^0)$ randomly or heuristically
3. **While** convergence criterion not met **do**
4. $\boldsymbol{U}_1^k \longleftarrow$ (3.10)
5. $\boldsymbol{U}_2^k \longleftarrow$ (3.11)
6. $\boldsymbol{U}_3^k \longleftarrow$ (3.12)
7. $(\tilde{\sigma}^k, \tilde{\beta}_0^k, \tilde{\boldsymbol{\beta}}_1^k) \longleftarrow$ (3.13)
8. $\mathscr{S}^k \longleftarrow$ (3.14)
9. $k = k+1$ **End While**
10. **Output:** Basis matrices of the low-dimensional tensor subspace $\{\boldsymbol{U}_1^k, \boldsymbol{U}_2^k, \boldsymbol{U}_3^k\}$

3.3.2 Initialization and Hyperparameter Tuning

To run Algorithm 1, we need to initialize the parameters $\boldsymbol{U}_1^0, \boldsymbol{U}_2^0, \boldsymbol{U}_3^0, \tilde{\sigma}^0, \tilde{\beta}_0^0, \tilde{\boldsymbol{\beta}}_1^0, \mathscr{S}^0$. The initialization can be accomplished randomly or heuristically. In this article, we propose a heuristic initialization method. Specifically, if tensor $\mathscr{X}$ has no missing entries, MPCA (Lu et al. 2008) is applied to tensor $\mathscr{X}$, which yields $\{\boldsymbol{U}_1^0, \boldsymbol{U}_2^0, \boldsymbol{U}_3^0\}$. Next, we compute $\mathscr{S}^0$ by

solving $\mathscr{S}^0 = \arg\min_{\mathscr{S}} \ ||\mathscr{P}_\Omega(\mathscr{X} - \mathscr{S} \times_1 \boldsymbol{U}_1^{0^\top} \times_2 \boldsymbol{U}_2^{0^\top} \times_3 \boldsymbol{U}_3^{0^\top})||_F^2$. Finally, $\tilde{\beta}_0^0, \tilde{\boldsymbol{\beta}}_1^0, \tilde{\sigma}^0$ are computed by solving $\min_{\tilde{\beta}_0^0, \tilde{\boldsymbol{\beta}}_1^0, \tilde{\sigma}^0} \ell(\tilde{\sigma}^0 \boldsymbol{y} - \mathbf{1}_M \tilde{\beta}_0^0 - \boldsymbol{S}_{(4)}^0 \tilde{\boldsymbol{\beta}}_1^0)$, where $\boldsymbol{S}_{(4)}^0$ is the mode-4 matricization of $\mathscr{S}^0$. If tensor $\mathscr{X}$ has missing values, a tensor completion method (Filipović and Jukić 2015; Liu et al. 2012; Xu et al. 2013) can be conducted before applying MPCA.

In addition to the initialization, the hyperparameter parameters including the weight α and the dimension of tensor subspace (P_1, P_2, P_3) also need to be pre-determined. It is known that α controls the weights of the feature extraction term and the regression term, and $\alpha \in [0,1]$. To select an appropriate weight parameter, we will first split the range $[0,1]$ into $L+1$ intervals equally, which yields $\alpha_0 = 0/L, \alpha_1 = 1/L, \alpha_2 = 2/L, \ldots, \alpha_L = L/L$. Next, we employ cross-validation to select the weight that achieves the highest prediction accuracy. If the weight at the boundary is selected (i.e., $\alpha_0 = 0/L$ or $\alpha_L = L/L$), we further split the interval closest to the boundary and conduct cross-validation again. For example, if $\alpha_0 = 0/L$ is chosen as the best weight, we will split $[0/L, 1/L]$ into $(L+1)$ intervals equally and re-conduct the cross-validation. This process is repeated until a non-boundary weight is selected. Of course, a maximum number of repetitions needs to be set to control the computational time.

The values of $\{P_1, P_2, P_3\}$ can be determined using cross-validation as well. To be specific, we may try a certain number of candidate values for $\{P_1, P_2, P_3\}$ and run Algorithm 1 to extract low-dimensional features, which are then used to build the prognostic model discussed in Section 3.2.3 for TTF prediction. The values which achieve the smallest prediction error will be chosen. It is known that there usually exists high spatio-temporal correlations among degradation image streams (Fang et al. 2019), so the dimension of the tensor subspace is usually low, which helps reduce the computation intensity of model selection. The values of $\{P_1, P_2, P_3\}$ can also be determined heuristically. For example, if MPCA is employed for parameter initialization, then the fraction-of-variance-explained (Lu et al. 2008) can be used to determine the dimension of tensor subspace.

3.4 Analytical Solutions

In this section, we discuss the closed-form solutions of optimization problems (3.10), (3.11), (3.12), (3.13), and (3.14) in Algorithm 1. Specifically, we will discuss the solutions when degradation image streams are complete and incomplete in Sections 3.4.1 and 3.4.2, respectively. For simplicity, we will remove the superscripts k and $k-1$.

3.4.1 Analytical Solutions for Complete Data

3.4.1.1 Solution procedure for $\boldsymbol{U}_1$

When degradation image streams are complete (i.e., the 4D image tensor $\mathscr{X}$ in criterion (3.4) has no missing entries), we have the following proposition, which provides the analytical solution to problem (3.10).

Proposition 1. *If the 4D tensor $\mathscr{X}$ has no missing values, optimization problem (3.10) has the following analytical solution*

$$\boldsymbol{U}_1 = (\boldsymbol{X}_{(1)} \cdot \boldsymbol{S}_{U_1(1)}^{\top} \cdot (\boldsymbol{S}_{U_1(1)} \cdot \boldsymbol{S}_{U_1(1)}^{\top})^{-1})^{\top},$$

where $\boldsymbol{X}_{(1)}$ is the mode-1 matricization of $\mathscr{X}$, $\mathscr{S}_{U_1} = \mathscr{S} \times_2 \boldsymbol{U}_2^{\top} \times_3 \boldsymbol{U}_3^{\top}$, $\boldsymbol{S}_{U_1(1)}$ is the mode-1 matricization of $\mathscr{S}_{U_1}$, and the operator "$\cdot$" represents multiplication.

3.4.1.2 Solution procedure for $\boldsymbol{U}_2$

When degradation image streams are complete, the proposition below gives the analytical solution to problem (3.11).

Proposition 2. *If the 4D tensor $\mathscr{X}$ has no missing values, optimization problem (3.11) has the following analytical solution*

$$\boldsymbol{U}_2 = (\boldsymbol{X}_{(2)} \cdot \boldsymbol{S}_{U_2(2)}^{\top} \cdot (\boldsymbol{S}_{U_2(2)} \cdot \boldsymbol{S}_{U_2(2)}^{\top})^{-1})^{\top},$$

where $\boldsymbol{X}_{(2)}$ is the mode-2 matricization of $\mathscr{X}$, $\mathscr{S}_{U_2} = \mathscr{S} \times_1 \boldsymbol{U}_1^{\top} \times_3 \boldsymbol{U}_3^{\top}$, and $\boldsymbol{S}_{U_2(2)}$ is the mode-2 matricization of $\mathscr{S}_{U_2}$.

3.4.1.3 Solution procedure for $\boldsymbol{U}_3$

When degradation image streams are complete, the proposition below provides the analytical solution to problem (3.12).

Proposition 3. *If the 4D tensor $\mathscr{X}$ has no missing values, optimization problem (3.12) has the following analytical solution*

$$\boldsymbol{U}_3 = (\boldsymbol{X}_{(3)} \cdot \boldsymbol{S}_{U_3(3)}^\top \cdot (\boldsymbol{S}_{U_3(3)} \cdot \boldsymbol{S}_{U_3(3)}^\top)^{-1})^\top,$$

where $\boldsymbol{X}_{(3)}$ is the mode-3 matricization of $\mathscr{X}$, $\mathscr{S}_{U_3} = \mathscr{S} \times_1 \boldsymbol{U}_1^\top \times_2 \boldsymbol{U}_2^\top$, and $\boldsymbol{S}_{U_3(3)}$ is the mode-3 matricization of $\mathscr{S}_{U_3}$.

3.4.1.4 Solution procedure for $\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}_1, \tilde{\sigma}$

For general LLS distributions, there is no closed-form solution for $\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}_1, \tilde{\sigma}$. As a result, we may use existing algorithms (Doray 1994) or convex optimization packages to solve problem (3.13). However, if the TTF follows a Normal (or lognormal) distribution, we may replace the negative log-likelihood term in criterion (3.3) with a mean squared error-based loss function, which results in the following optimization criterion:

$$\min_{\boldsymbol{U}_1, \boldsymbol{U}_2, \boldsymbol{U}_3, \boldsymbol{\beta}_1, \beta_0, \mathscr{S}} \alpha ||\mathscr{P}_\Omega(\mathscr{X} - \mathscr{S} \times_1 \boldsymbol{U}_1^\top \times_2 \boldsymbol{U}_2^\top \times_3 \boldsymbol{U}_3^\top)||_F^2 + (1-\alpha)||\boldsymbol{y} - \boldsymbol{1}_M \beta_0 - \boldsymbol{S}_{(4)} \boldsymbol{\beta}_1||_2^2. \quad (3.15)$$

Since criterion (3.15) is multi-convex, no re-parameterization is needed. As a result, problem (3.13) is equivalent to $\min_{\beta_0, \boldsymbol{\beta}_1} ||\boldsymbol{y} - \boldsymbol{1}_M \beta_0 - \boldsymbol{S}_{(4)} \boldsymbol{\beta}_1||_2^2$, the solution to which can be easily found using least squares: $\boldsymbol{\beta} = (\boldsymbol{S}^\top \boldsymbol{S})^{-1} \boldsymbol{S}^\top \boldsymbol{y}$, where $\boldsymbol{\beta} = (\beta_0, \boldsymbol{\beta}_1^\top)^\top \in \mathbb{R}^{(P_1 \times P_2 \times P_3 + 1) \times 1}$ and $\boldsymbol{S} = (\boldsymbol{1}_M, \boldsymbol{S}_{(4)}) \in \mathbb{R}^{M \times (P_1 \times P_2 \times P_3 + 1)}$. With the estimation of the regression coefficients, the scale parameter can be estimated using $\sigma = \sqrt{||\boldsymbol{y} - \boldsymbol{1}_M \beta_0 - \boldsymbol{S}_{(4)} \boldsymbol{\beta}_1||_2^2 / (M - P_1 \times P_2 \times P_3 - 1)}$.

3.4.1.5 Solution procedure for $\mathscr{S}$

For general LLS distributions, there is no closed-form solution for $\mathscr{S}$ either. Therefore, we may use existing convex optimization packages to solve problem (3.14). However, if the TTF follows a Normal (or lognormal) distribution, according to criterion (3.15), problem (3.14) is equivalent to

$$\mathscr{S}^k = \arg\min_{\mathscr{S}} \ \alpha ||\mathscr{P}_\Omega(\mathscr{X} - \mathscr{S} \times_1 \boldsymbol{U}_1^{k\top} \times_2 \boldsymbol{U}_2^{k\top} \times_3 \boldsymbol{U}_3^{k\top})||_F^2 + (1-\alpha)||\boldsymbol{y} - \boldsymbol{1}_M \beta_0^k - \boldsymbol{S}_{(4)} \boldsymbol{\beta}_1^k||_2^2, \quad (3.16)$$

which has an analytical solution as suggested by Proposition 4.

Proposition 4. *If the 4D tensor $\mathcal{X}$ has no missing values, optimization problem (3.16) has the following analytical solution*

$$\begin{aligned}\boldsymbol{S}_{(4)} = &\left[\alpha \cdot \boldsymbol{X}_{(4)} \cdot (\boldsymbol{U}_3 \otimes \boldsymbol{U}_2 \otimes \boldsymbol{U}_1)^\top + (1-\alpha) \cdot (\boldsymbol{y} - \mathbf{1}_M \cdot \beta_0) \cdot \boldsymbol{\beta}_1^\top\right] \cdot \\ &\left[\alpha \cdot (\boldsymbol{U}_3 \otimes \boldsymbol{U}_2 \otimes \boldsymbol{U}_1) \cdot (\boldsymbol{U}_3 \otimes \boldsymbol{U}_2 \otimes \boldsymbol{U}_1)^\top + (1-\alpha) \cdot \boldsymbol{\beta}_1 \cdot \boldsymbol{\beta}_1^\top\right]^{-1},\end{aligned}$$

where $\boldsymbol{X}_{(4)}$ is the mode-4 matricization of $\mathcal{X}$, $\boldsymbol{S}_{(4)}$ is the mode-4 matricization of $\mathcal{S}$.

The proof of Propositions 1, 2, 3, and 4 can be found in the Appendix

3.4.2 Analytical Solutions for Incomplete Data

In this subsection, we discuss the closed-form solutions for optimization problems (3.10), (3.11), (3.12), (3.13), and (3.14) when the degradation tensor $\mathcal{X}$ in criterion (3.4) is incomplete. We consider the most general missing pattern, *entry-wise missing*, which means that any entry of $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \times M}$ can be missing. Thus, the indices of missing entries can be denoted as a subset $\Omega \subseteq \{(i_1, i_2, i_3, m), 1 \le i_1 \le I_1, 1 \le i_2 \le I_2, 1 \le i_3 \le I_3, 1 \le m \le M\}$.

3.4.2.1 Solution procedure for $\boldsymbol{U}_1$

When tensor $\mathcal{X}$ has a general entry-wise missing structure, there is no closed-form solution for $\boldsymbol{U}_1$ in optimization criterion (3.10). However, we may decompose criterion (3.10) into multiple sub-optimization problems, each of which has an analytical solution. To be specific, we first give the following lemma.

Lemma 1. *Let $\boldsymbol{A} \in \mathbb{R}^{M \times N}$, $\boldsymbol{C} \in \mathbb{R}^{P \times N}$, and $\boldsymbol{B} \in \mathbb{R}^{M \times P}$, the solution to criterion $\arg\min_B \|\boldsymbol{A} - \boldsymbol{B}\boldsymbol{C}\|_F^2$ can be found by solving each row of $\boldsymbol{B}$ separately–that is–solving $\{\boldsymbol{b}_m\}_{m=1}^M$ as follows:*

$$\arg\min_{b_m} \|\boldsymbol{a}_m - \boldsymbol{b}_m \boldsymbol{C}\|_F^2, \quad m = 1, \ldots, M$$

where $\boldsymbol{a}_m \in \mathbb{R}^{1 \times N}$ is the mth row of $\boldsymbol{A}$, $\boldsymbol{b}_m \in \mathbb{R}^{1 \times P}$ is the mth row of $\boldsymbol{B}$.

The proof of Lemma 1 can be found in the Appendix. Lemma 1 enables us to solve each column of matrices $\boldsymbol{U}_1$ separately. Denote the i_1th column of matrix $\boldsymbol{U}_1 \in \mathbb{R}^{P_1 \times I_1}$ as $\boldsymbol{u}_1^{i_1} \in \mathbb{R}^{P_1 \times 1}, i_1 = 1, \ldots, I_1$, we replace optimization problem (3.10) with I_1 sub problems by separately optimizing $\boldsymbol{u}_1^1, \boldsymbol{u}_1^2, \ldots, \boldsymbol{u}_1^{I_1}$. Proposition 5 suggests that there is an analytical solution when optimizing $\boldsymbol{u}_1^{i_1}$.

Proposition 5. *When optimizing the i_1th column of $\boldsymbol{U}_1$ in problem* (3.10)*, we have the following analytical solution*

$$\boldsymbol{u}_1^{i_1} = (\boldsymbol{x}_{(1)}^{i_1,\pi_{i_1}} \cdot \boldsymbol{S}_{U_1(1)}^{\pi_{i_1}}{}^{\top} \cdot (\boldsymbol{S}_{U_1(1)}^{\pi_{i_1}} \cdot \boldsymbol{S}_{U_1(1)}^{\pi_{i_1}}{}^{\top})^{-1})^{\top},$$

where $\boldsymbol{x}_{(1)}^{i_1}$ denotes the i_1th row of $\boldsymbol{X}_{(1)}$, π_{i_1} is a set consisting of the indices of available entries of $\boldsymbol{x}_{(1)}^{i_1}$, $\boldsymbol{x}_{(1)}^{i_1,\pi_{i_1}}$ denotes a vector consisting of the available entries in the i_1th row of $\boldsymbol{X}_{(1)}$, $\mathcal{S}_{U_1} = \mathcal{S} \times_2 \boldsymbol{U}_2^{\top} \times_3 \boldsymbol{U}_3^{\top}$, $\boldsymbol{S}_{U_1(1)}$ is the mode-1 matricization of $\mathcal{S}_{U_1}$, and $\boldsymbol{S}_{U_1(1)}^{\pi_{i_1}}$ denotes a matrix comprises the π_{i_1} columns of $\boldsymbol{S}_{U_1(1)}$.

As mentioned earlier, when tensor $\mathcal{X}$ has a general entry-wise missing structure, there is no closed-form solution for $\boldsymbol{U}_1$ in optimization criterion (3.10). Therefore, we have to optimize each column of matrix $\boldsymbol{U}_1$ separately using the analytical solution provided in Proposition 5. However, for image-based applications, the missing data in tensor $\mathcal{X}$ are images but not entries, as illustrated in Figure 4.8(b). This yields an *image-wise missing* structure, which is a special case of the general *entry-wise missing* structure. For image streams with an *image-wise missing* structure, Proposition 6 suggests that $\boldsymbol{U}_1$ in optimization criterion (3.10) can be estimated analytically as a whole, which means we do not have to optimize each of its columns separately.

Proposition 6. *If the indices of tensor $\mathcal{X}$'s missing entries can be denoted as $\Omega \subseteq \{(:,:,i_3,m), 1 \leq i_3 \leq I_3, 1 \leq m \leq M\}$, where ":" denotes all the indices in a dimension, then $\mathcal{X}$'s mode-1 matricization $\boldsymbol{X}_{(1)}$ has missing columns. Let π be the set consisting of the indices of available columns in $\boldsymbol{X}_{(1)}$, then optimization problem (3.10) has the following analytical solution*

$$\boldsymbol{U}_1 = (\boldsymbol{X}_{(1)}^{\pi} \cdot \boldsymbol{S}_{U_1(1)}^{\pi}{}^{\top} \cdot (\boldsymbol{S}_{U_1(1)}^{\pi} \cdot \boldsymbol{S}_{U_1(1)}^{\pi}{}^{\top})^{-1})^{\top},$$

where $\boldsymbol{X}_{(1)}^{\pi}$ is a matrix consisting of the π columns of $\boldsymbol{X}_{(1)}$, $\mathcal{S}_{U_1} = \mathcal{S} \times_2 \boldsymbol{U}_2^{\top} \times_3 \boldsymbol{U}_3^{\top}$, $\boldsymbol{S}_{U_1(1)}$ is the mode-1 matricization of $\mathcal{S}_{U_1}$, and $\boldsymbol{S}_{U_1(1)}^{\pi}$ denotes a matrix constituting the π columns of $\boldsymbol{S}_{U_1(1)}$.

3.4.2.2 Solution procedure for $\boldsymbol{U}_2$

Similar to $\boldsymbol{U}_1$, there is no closed-form solution for $\boldsymbol{U}_2$ in optimization criterion (3.11) when tensor $\mathcal{X}$ has a general entry-wise missing structure. However, Lemma 1 implies that we may also decompose optimization problem (3.11) into multiple sub-criteria, each of

which has a closed-form solution. Specifically, denote the i_2th column of matrix $\boldsymbol{U}_2 \in \mathbb{R}^{P_2 \times I_2}$ as $\boldsymbol{u}_2^{i_2} \in \mathbb{R}^{P_2 \times 1}$, $i_2 = 1, \ldots, I_2$, we can replace optimization problem (3.11) with I_2 sub problems by separately optimizing $\boldsymbol{u}_2^1, \boldsymbol{u}_2^2, \ldots, \boldsymbol{u}_2^{I_2}$. Proposition 7 shows that there is an analytical solution for $\boldsymbol{u}_2^{i_2}$.

Proposition 7. *When optimizing the i_2th column of $\boldsymbol{U}_2$ in problem* (3.11)*, we have the following analytical solution*

$$\boldsymbol{u}_2^{i_2} = (\boldsymbol{x}_{(2)}^{i_2,\pi_{i_2}} \cdot \boldsymbol{S}_{U_2(2)}^{\pi_{i_2}}{}^{\top} \cdot (\boldsymbol{S}_{U_2(2)}^{\pi_{i_2}} \cdot \boldsymbol{S}_{U_2(2)}^{\pi_{i_2}}{}^{\top})^{-1})^{\top}$$

where $\boldsymbol{x}_{(2)}^{i_2}$ denotes the i_2th row of $\boldsymbol{X}_{(2)}$, π_{i_2} is a set consisting of the indices of available entries of $\boldsymbol{x}_{(2)}^{i_2}$, $\boldsymbol{x}_{(2)}^{i_2,\pi_{i_2}}$ is a vector consisting of the available entries in the i_2th column of $\boldsymbol{X}_{(2)}$, $\mathscr{S}_{U_2} = \mathscr{S} \times_1 \boldsymbol{U}_1^{\top} \times_3 \boldsymbol{U}_3^{\top}$, $\boldsymbol{S}_{U_2(2)}$ is the mode-2 matricization of $\mathscr{S}_{U_2}$, and $\boldsymbol{S}_{U_2(2)}^{\pi_{i_2}}$ denotes a matrix comprises the π_{i_2} columns of $\boldsymbol{S}_{U_2(2)}$.

Similar to $\boldsymbol{U}_1$, when tensor $\mathscr{X}$ has the *image-wise missing* structure, we do not have to optimize each of the columns of $\boldsymbol{U}_2$ separately. Proposition 8 below gives an analytical solution to $\boldsymbol{U}_2$ when tensor $\mathscr{X}$ has missing images.

Proposition 8. *If the indices of tensor $\mathscr{X}$'s missing entries can be denoted as $\Omega \subseteq \{(:,:,i_3,m), 1 \le i_3 \le I_3, 1 \le m \le M\}$, where ":" denotes all the indices in a dimension, then $\mathscr{X}$'s mode-2 matricization $\boldsymbol{X}_{(2)}$ has missing columns. Let π be the set consisting of the indices of available columns in $\boldsymbol{X}_{(2)}$, then optimization problem (3.11) has the following analytical solution*

$$\boldsymbol{U}_2 = (\boldsymbol{X}_{(2)}^{\pi} \cdot \boldsymbol{S}_{U_2(2)}^{\pi}{}^{\top} \cdot (\boldsymbol{S}_{U_2(2)}^{\pi} \cdot \boldsymbol{S}_{U_2(2)}^{\pi}{}^{\top})^{-1})^{\top},$$

where $\boldsymbol{X}_{(2)}^{\pi}$ is a matrix consisting of the π columns of $\boldsymbol{X}_{(2)}$, $\boldsymbol{S}_{U_2(2)}$ is the mode-2 matricization of $\mathscr{S}_{U_2}$, and $\boldsymbol{S}_{U_2(2)}^{\pi}$ denotes a matrix constituting the π columns of $\boldsymbol{S}_{U_2(2)}$.

3.4.2.3 Solution procedure for $\boldsymbol{U}_3$

There is no closed-form solution for $\boldsymbol{U}_3$ in optimization criterion (3.12) when tensor $\mathscr{X}$ has missing entries. Based on Lemma 1, we decompose optimization problem (3.12) into multiple sub-criteria, each of which has a closed-form solution. Denote the i_3th column of matrix $\boldsymbol{U}_3 \in \mathbb{R}^{P_3 \times I_3}$ as $\boldsymbol{u}_3^{i_3} \in \mathbb{R}^{P_3 \times 1}$, $i_3 = 1, \ldots, I_3$, we replace optimization problem (3.12) with I_3 sub problems by separately optimizing $\boldsymbol{u}_3^1, \boldsymbol{u}_3^2, \ldots, \boldsymbol{u}_3^{I_3}$ respectively. Proposition 9 suggests that there is an analytical solution when optimizing $\boldsymbol{u}_3^{i_3}$.

Proposition 9. *When optimizing the i_3th column of $\boldsymbol{U}_3$ in problem* (3.12)*, we have the following analytical solution*

$$\boldsymbol{u}_3^{i_3} = (\boldsymbol{x}_{(3)}^{i_3,\pi_{i_3}} \cdot \boldsymbol{S}_{U_3(3)}^{\pi_{i_3}\top} \cdot (\boldsymbol{S}_{U_3(3)}^{\pi_{i_3}} \cdot \boldsymbol{S}_{U_3(3)}^{\pi_{i_3}\top})^{-1})^\top,$$

where $\boldsymbol{x}_{(3)}^{i_3}$ denotes the i_3th row of $\boldsymbol{X}_{(3)}$, π_{i_3} is a set consisting of the indices of available entries of $\boldsymbol{x}_{(3)}^{i_3}$, $\boldsymbol{x}_{(3)}^{i_3,\pi_{i_3}}$ is a vector consisting of the available entries in the i_3th row of $\boldsymbol{X}_{(3)}$, $\mathscr{S}_{U_3} = \mathscr{S} \times_1 \boldsymbol{U}_1^\top \times_2 \boldsymbol{U}_2^\top$, $\boldsymbol{S}_{U_3(3)}$ is the mode-3 matricization of $\mathscr{S}_{U_3}$, and $\boldsymbol{S}_{U_3(3)}^{\pi_{i_3}}$ denotes a matrix comprises the π_{i_3} columns of $\boldsymbol{S}_{U_3(3)}$.

3.4.2.4 Solution procedure for $\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}_1, \tilde{\sigma}$

Whether tensor $\mathscr{X}$ contains missing entries does not affect the methods for $\tilde{\beta}_0, \tilde{\boldsymbol{\beta}}_1$, and $\tilde{\sigma}$ estimation. Therefore, the estimation methods discussed in Section 3.4.1.4 can still be used.

3.4.2.5 Solution procedure for $\mathscr{S}$

For general LLS distributions, there is no closed-form solution for $\mathscr{S}$. Therefore, we may use existing convex optimization packages to solve problem (3.14). However, if the TTF follows a Normal (or lognormal) distribution, problem (3.14) is equivalent to (3.16) (see Section 3.4.1.5 for details). Based on Lemma 1, we may optimize each row of $\boldsymbol{S}_{(4)}$ separately. We denote the mth row of matrix $\boldsymbol{S}_{(4)} \in \mathbb{R}^{M\times(P_1\times P_2\times P_3)}$ as $\boldsymbol{s}_{(4)}^m \in \mathbb{R}^{1\times(P_1\times P_2\times P_3)}$, $m = 1,\ldots,M$, and replace optimization problem (3.16) with M sub problems–that is–separately optimizing $\boldsymbol{s}_{(4)}^1, \boldsymbol{s}_{(4)}^2, \ldots, \boldsymbol{s}_{(4)}^M$. Proposition 10 suggests that there is an analytical solution when optimizing $\boldsymbol{s}_{(4)}^m$.

Proposition 10. *When optimizing the mth row of matrix $\boldsymbol{S}_{(4)}$ in problem (3.16), we have the following analytical solution*

$$\begin{aligned}\boldsymbol{s}_{(4)}^m = &\left[\alpha \cdot \boldsymbol{x}_{(4)}^{m,\pi_m} \cdot (\boldsymbol{U}_3 \otimes \boldsymbol{U}_2 \otimes \boldsymbol{U}_1)^{\pi_m\top} + (1-\alpha)\cdot(y_m - \tilde{\beta}_0)\cdot\tilde{\boldsymbol{\beta}}_1^\top\right]\cdot \\ &\left[\alpha\cdot(\boldsymbol{U}_3 \otimes \boldsymbol{U}_2 \otimes \boldsymbol{U}_1)^{\pi_m} \cdot (\boldsymbol{U}_3 \otimes \boldsymbol{U}_2 \otimes \boldsymbol{U}_1)^{\pi_m\top} + (1-\alpha)\cdot\tilde{\boldsymbol{\beta}}_1\cdot\tilde{\boldsymbol{\beta}}_1^\top\right]^{-1},\end{aligned}$$

where $\boldsymbol{x}_{(4)}^m$ represents the mth row of $\boldsymbol{X}_{(4)}$, π_m denotes the set consisting of the indices of available entries in $\boldsymbol{x}_{(4)}^m$, $\boldsymbol{x}_{(4)}^{m,\pi_m}$ is a vector consisting of the available entries in the mth row of $\boldsymbol{X}_{(4)}$, and $(\boldsymbol{U}_3\otimes\boldsymbol{U}_2\otimes\boldsymbol{U}_1)^{\pi_m}$ denotes a matrix comprising the π_m columns of matrix $\boldsymbol{U}_3\otimes\boldsymbol{U}_2\otimes\boldsymbol{U}_1$.

The proof of all Propositions 5, 6, 7, 8, 9, and 10 can be found in the Appendix.

3.5 Numerical Studies

In this section, we validate the effectiveness of our proposed supervised tensor dimension reduction-based prognostic model using simulated data.

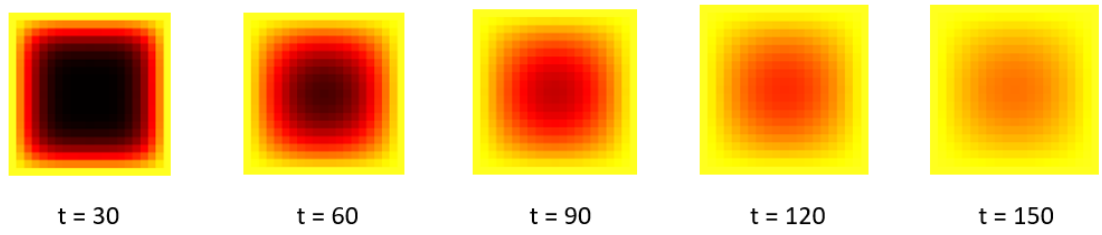
3.5.1 Data Generation

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

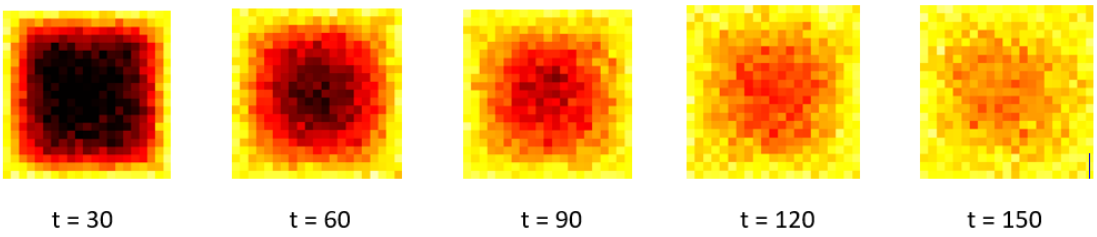
$$\frac{\partial \mathscr{X}_m(x,y,t)}{\partial t} = \alpha_m\Big(\frac{\partial^2 \mathscr{X}_m}{\partial x^2} + \frac{\partial^2 \mathscr{X}_m}{\partial y^2}\Big), \tag{3.17}$$

where $(x,y), 0 \le x, y \le 0.2$, represents the location of each image pixel. α_m is the thermal diffusivity coefficient, which is randomly generated from a uniform distribution $\mathscr{U}(0.5\times 10^{-4}, 1\times 10^{-4})$. t is the time index. The initial and boundary conditions are set such that $\mathscr{X}|_{t=1} = 0$ and $\mathscr{X}_m|_{x=0} = \mathscr{X}_m|_{x=0.2} = \mathscr{X}_m|_{y=0} = \mathscr{X}_m|_{y=0.2} = 30$. 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 yields 150 images of size 21×21 for each asset. This implies that the degradation image stream of each asset can be represented by a $21\times 21\times 150$ tensor. In addition, an independent and identically distributed random noise $\epsilon \sim N(0,0.1)$ is added to each pixel. Figure 4.8 demonstrates an example of some images with and without noise from one of the assets simulated in this study.

To determine the TTF of an asset, we first transform the asset's $21\times 21\times 150$ tensor to a 1×150 time series by taking the average pixel intensity of each image. The time series signal indicates how the average heat of the asset involves over time. Next, we let the TTF of the asset be the time point where the amplitude of the time series signal crosses a pre-defined soft failure threshold, which is set as 23 in this study. Since the images of different assets are generated with different thermal diffusivity coefficients, the time points where their time series signals go beyond the threshold may be different. Thus, the TTF of different asset may also be different. To mimic reality, we truncate the image stream of each asset by keeping only the images observed before its TTF. In other words, any images observed after an asset's TTF are removed from the image tensor of the asset. Such a truncation is normal in reality since an asset usually gets maintained or replaced once its degradation



(a) Without noise



(b) With noise

Figure 3.2: Simulated degradation images based on heat transfer process.

signal crosses the soft failure threshold. Consequently, the third dimension of the tensor of different assets might be different. In addition, to reduce the computation load, we keep one of every 10 images in the truncated image stream of each asset.

3.5.2 The Benchmark and Performance Comparison

We randomly split the generated data into a training data set consisting of 400 assets and a test data set consisting of the remaining 100 assets. To test the robustness of the proposed method, we consider four levels of data incompleteness: (1) 0% missing, (2) 10% missing, (3) 50% missing, (4) 90% missing. For the first scenario, (1) 0% missing, we use all the generated data for model training and testing. Please notice that even though all the available images are used, the image tensor $\mathscr{X}$ is still incomplete due to failure time truncation–that is– different assets may have different TTFs and thus different number of images (see the discussion in the second paragraph of Section 3.2.3). For the remaining scenarios, we randomly remove some images from each asset's image stream. For example, when 10% missing, we randomly remove 10% of the images (rounding to the nearest integer) from the image stream of each asset.

We compare the performance of our proposed method with an unsupervised tensor dimension reduction-based benchmark. Considering image streams are incomplete, the

baseline model first applies a tensor completion method known as TMac developed by

We use the heuristic method discussed in Section 3.3.1 to initialize the block updating algorithm. In this study, we use lognormal regression to build the prognostic model. The proposed method is denoted as “Proposed_CV". The prediction errors of our proposed method and two benchmarks are calculated by using the equation below and reported in Figures 3.3, 3.4, 3.5, and 3.6.

$$\text{Prediction Error} = \frac{|\text{Estimated TTF} - \text{True TTF}|}{\text{True TTF}}. \tag{3.18}$$

3.5.3 Results and analysis

Figure 3.3 reports the prediction errors of the two benchmarks and our proposed method when data is complete, which means no image is removed on purpose. Figure 3.4 shows the prediction errors when 10% entries in the 3rd mode (time) of degradation image streams are missing, while Figures 3.5 and 3.6 demonstrate the errors when 50% and 90% images are missing, respectively.

Figures 3.3, 3.4, 3.5, and 3.6 illustrate that our proposed method outperforms the benchmarks under all data missing rates. For example, when the degradation image signals are complete, the median absolute prediction errors (and the Interquartile Ranges, i.e., IQRs) of the proposed method and the two benchmarks are 0.003 (0.003), 0.027 (0.035), and 0.025 (0.033), respectively; when 10% images are missing, the median absolute prediction errors (and IQRs) of the three methods are respectively 0.019 (0.017), 0.058 (0.067), and 0.053 (0.063); when 50% of images are missing, they are 0.052 (0.084), 0.302 (0.405), and 0.104 (0.168). We believe this is because our proposed method applies historical TTFs to supervise the low-dimensional tensor dimension reduction, and thus the extracted features are more effective for failure time prediction. Unlike our method, the two baseline models use MPCA, an unsupervised tensor dimension reduction method, for feature extraction. Since the extracted features are only determined by the image streams, and no TTF gets involved, they are not as effective as the features extracted by our proposed method, and thus their failure time prediction accuracy and precision are compromised.

The figures 3.3, 3.4, 3.5, and 3.6 also suggest that the performances of all the three models deteriorate, and the superiority of our proposed method over the two benchmarks decreases, with the increase of data missing rate. For example, when data is complete, the median absolute prediction errors (and IQRs) of “Proposed_CV" and “MPCA_CV" are 0.003 (0.003) and 0.025 (0.033), respectively; when the missing rate increases to 90%, they are

respectively 0.13 (0.21) and 0.16 (0.19), which are almost comparable. This is reasonable since the performance of all the models are compromised more when more data are missing. In addition, no model will perform well if a high percentage (say more than 90%) of data are missing since it implies that very limited useful degradation information is available for modeling.

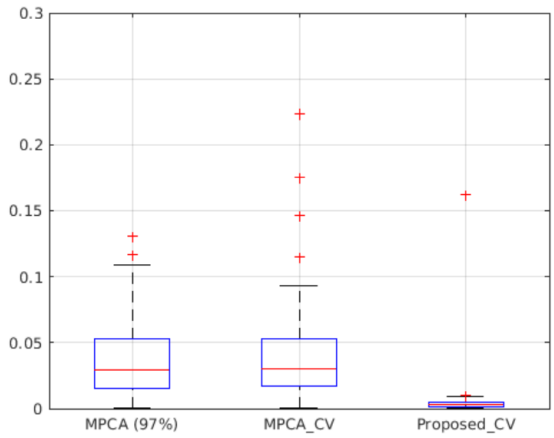


Figure 3.3: Prediction errors when data is complete in Numerical Study.

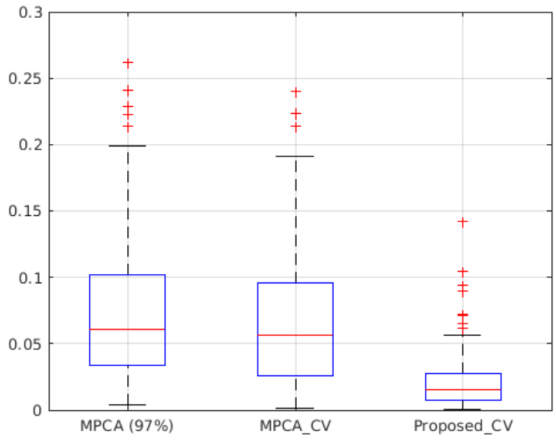


Figure 3.4: Prediction errors when 10% data is missing in Numerical Study.

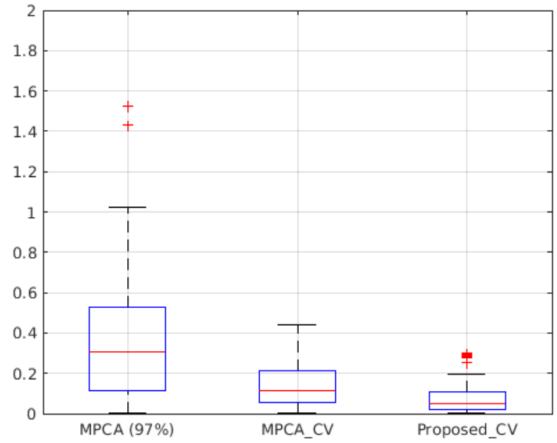


Figure 3.5: Prediction errors when 50% data is missing in Numerical Study.

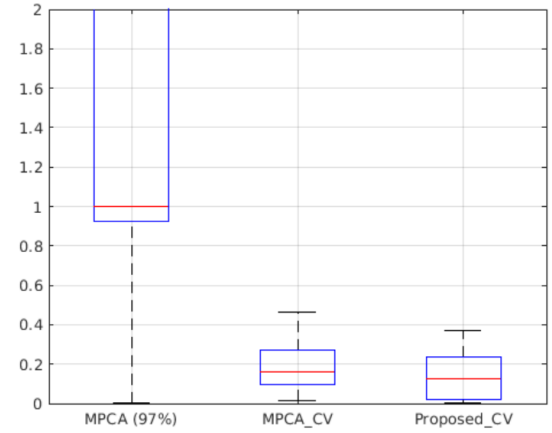


Figure 3.6: Prediction errors when 90% data is missing in Numerical Study.

The Figures 3.3-3.6 also demonstrate that “MPCA_CV" always outperforms “MPCA (97%)", and the superiority of “MPCA_CV" is augmented with the increase of data missing rate. For instance, when 10% images are missing, the median absolute prediction errors (and IQR) of “MPCA (97%)" and “MPCA_CV" are 0.058 (0.067) and 0.053 (0.063), respectively; when the missing rate is 50%, they are 0.302 (0.405) and 0.104 (0.168). One of the possible reasons is that “MPCA (97%)" determines the dimension of the tensor subspace by setting the “FVE" as 97%, which usually results in relatively high-dimensional features, although the dimension is smaller than that of the original image tensor. Relatively high-dimensional features implies an insufficient dimension reduction. In addition, it means the number of parameters in the subsequent LLS-based prognostic model is relatively large, which poses estimation challenges given that the number of samples (assets) for model training is limited.

3.6 Case Study

In this section, we use degradation image streams obtained from a rotating machinery test bed to validate the effectiveness of our proposed method. The test bed is designed to perform accelerated degradation tests on rolling entry thrust bearings. Specifically, bearings were run from brand new to failure. An FLIR T300 infrared camera was used to monitor the degradation process and collect degradation images over time. In the meanwhile, an accelerometer was mounted on the test bed to monitor the vibration of the bearing, and the failure time is defined as the time point where the amplitude of defective vibration frequencies crosses a threshold based on ISO standards for machine vibration. The data set consists of 284 degradation image streams and their corresponding TTFs, and each image has 40×20 pixels. As an illustration, a sequence of images obtained at different (ordered) time periods of one of the bearing are shown in Figure 4.10. More details about the experimental setup and the data set can be found in

We use 5-fold cross validation to evaluate the performance of our proposed model and the two benchmarks discussed in Section 3.5. Similar to the simulation study, we conduct 10-fold cross validation to determine the optimal weight parameter in criterion (3.4) and the most appropriate dimension of the tensor subspace. In addition, we also consider four levels of data incompleteness: (1) 0% missing (i.e., complete), (2) 10% missing, (3) 50% missing, and (4) 90% missing. Figure 3.8 illustrates the absolute prediction errors when degradation image streams are complete. Figure 3.9 shows prediction errors when 10% of

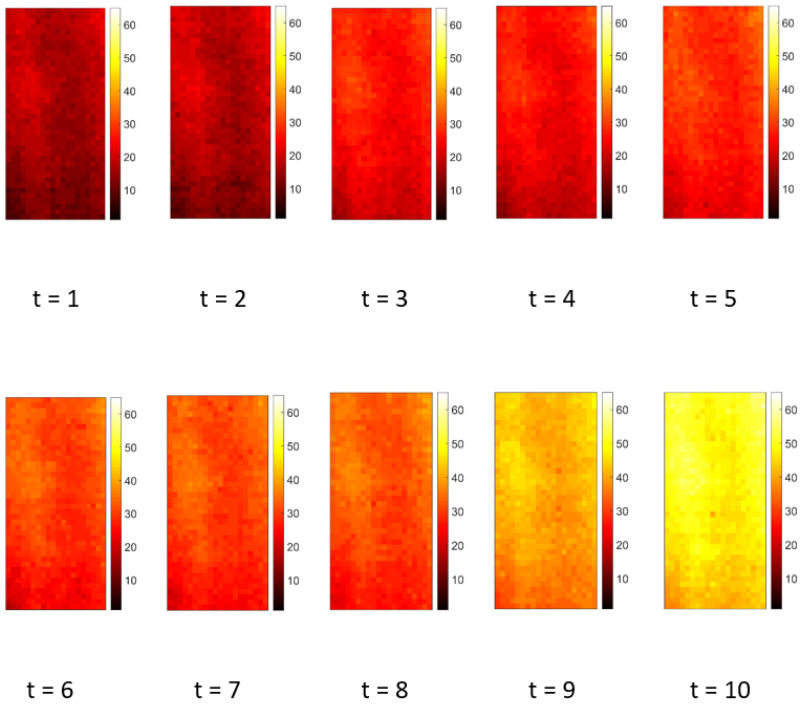


Figure 3.7: An illustration of one infrared degradation image stream.

the images of each bearing are missing. Figures 3.10 and 3.11 demonstrate the absolute prediction errors when the missing rate are 50% and 90%, respectively.

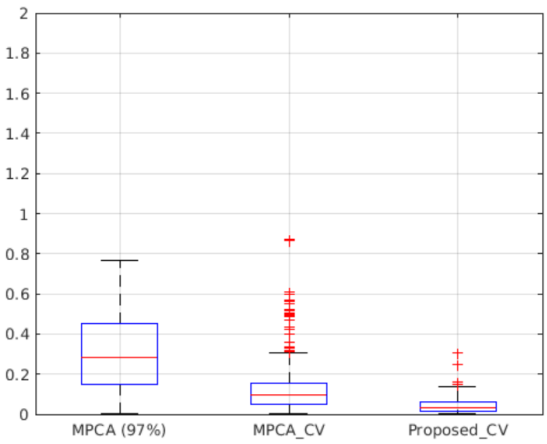


Figure 3.8: Prediction errors when data is complete in Case Study.

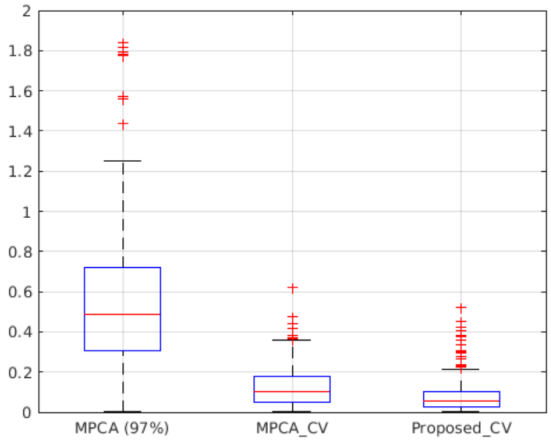


Figure 3.9: Prediction errors when 10% data is missing in Case Study.

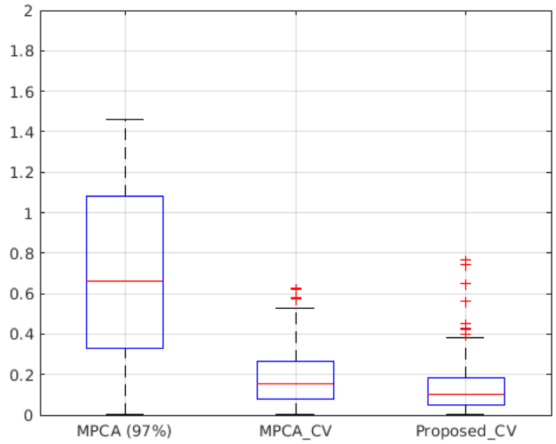


Figure 3.10: Prediction errors when 50% data is missing in Case Study.

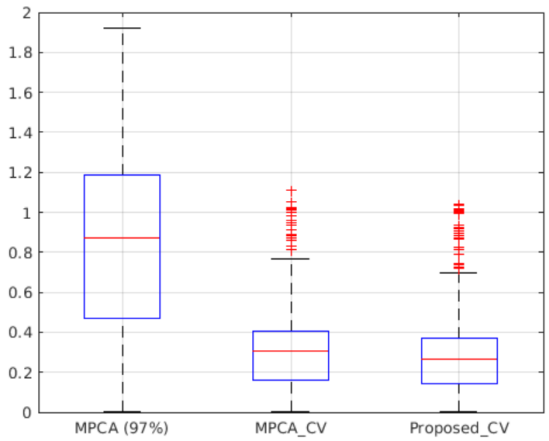


Figure 3.11: Prediction errors when 90% data is missing in Case Study.

Similar to the discovery in the numerical study in Section 3.5, Figures 3.8, 3.9, 3.10, and 3.11 indicate that our proposed method constantly works better than the two benchmarks under all the 4 data missing rates. For example, the median absolute prediction errors (and IQRs) of our proposed method and the two benchmarks are 0.03 (0.04), 0.3 (0.24), and 0.1 (0.16), respectively, when the degradation image streams are complete. When 50% of the images are missing, the median absolute prediction errors (and IQR) of are respectively 0.09 (0.17), 0.62 (0.65), and 0.15 (0.19). We believe this is because our proposed model is a supervised dimension reduction-based method, which uses TTF information to supervise the defection of the low-dimensional tensor subspace, while the benchmarks are unsupervised dimension reduction-based methods without TTF information involved. Since our method considers TTF information when detecting the tensor subspace, the extracted features are more effective for failure time prediction.

Figure 3.8, 3.9, 3.10, 3.11 also show that the prediction errors of all the 3 methods increase with the increase of data missing rates. For example, when the missing rates are 0%, 10%, 50%, and 90%, the median absolute prediction errors (and IQRs) of “MPCA (97%)" are 0.3 (0.24), 0.49 (0.42), 0.62 (0.65), and 0.83 (0.78), respectively, while they are respectively 0.09 (0.11), 0.1 (0.16), 0.15 (0.19), and 0.31 (0.22) for “MPCA_CV", and 0.03 (0.04), 0.05 (0.08), 0.09 (0.17), and 0.29 (0.21) for our proposed method. This is reasonable since a higher data missing rate means less useful degradation information and thus a worse model performance. In addition, we observe that the superiority of our proposed method over the two benchmarks decrease with the increase of data missing rates. For

example, the prediction accuracy of our method and “MPCA (97%)" are comparable when the data missing rate is 90%. We again believe this is because not much useful information is available when data is highly incomplete and none of the two models perform well with such limited data.

We also observe that “MPCA_CV" always outperforms “MPCA (97%)", and the superiority of “MPCA_CV" is augmented with the increase of data missing rate. For example, when 10% images are missing, the median absolute prediction errors (and IQR) of “MPCA (97%)" and MPCA_CV are 0.49 (0.42) and 0.1 (0.16), respectively; when the missing rate is 50%, the they are 0.62 (0.65) and 0.15 (0.19). We again believe this is because “MPCA (97%)" determines the dimension of the tensor subspace by setting the “FVE" as 97%, which results in relatively high-dimensional features due to the insufficient dimension reduction. Also, it results in a parameter estimation challenge since the number of parameters to be estimated in the prognostic model is relatively large comparing to the limited number of historical samples for model training. This suggests that cross validation is a better method to determine the dimension of the tensor subspace, especially when we do not have enough number of samples for model training.

3.7 Conclusions

This chapter proposed a supervised tensor dimension reduction-based prognostic model for applications with incomplete degradation imaging data. This is achieved by first developing a new supervised tensor dimension reduction method that reduces the dimension of incomplete high-dimensional degradation image streams and provides low-dimensional tensor features, which are then used to build a prognostic model based on (log)-location-scale regression.

The supervised tensor dimension reduction method uses historical TTFs to supervise the detection of a low-dimensional tensor subspace to reduce the dimension of incomplete high-dimensional image streams. Mathematically, it is formulated as an optimization criterion that combines a feature extraction term and a regression term. The feature extraction term focuses on identifying a tensor space to extract low-dimensional tensor features from high-dimensional image streams. The regression term regresses failure times against the features extracted by the first term using LLS regression. By jointly optimizing the two terms, it is expected to detect an appropriate tensor subspace such that the extracted features are effective for TTF prediction. To estimate the parameters of the supervised dimension

reduction method, we developed a Block Updating Algorithm for applications where TTFs follow distributions in the (log)-location-scale family. The algorithm works by splitting the parameters into several blocks and cyclically optimizing one block of parameters while keeping other blocks fixed until convergence. In addition, we showed that if TTFs follow normal or lognormal distributions, there is a closed-form solution when optimizing each block of the parameters, no matter the imaging data is complete or incomplete.

Simulated data as well as a data set from rotating machinery were used to validate the effectiveness of our proposed method. The results showed that our proposed prognostic method consistently outperforms the unsupervised tensor reduction-based benchmarks under various data missing rates. This validated the benefits and importance of using failure time information to supervise the dimension reduction of high-dimensional degradation image streams when building prognostic models.

The proposed prognostic model assumes that the TTFs of assets in the training dataset are known. In many real-world applications, the historical failure times might be right censored. This is because a component might be replaced before failure, so the exact TTF is unknown, and we only know that it is larger than the replacement time. How to incorporate censored TTFs into the proposed method can be an interesting future research topic.

CHAPTER

4

FEDERATED MULTILINEAR PRINCIPAL COMPONENT ANALYSIS (FMPCA) WITH APPLICATIONS IN PROGNOSTICS

4.1 Introduction

Over the past few decades, tensors, which are multidimensional arrays, have gained increasing prominence in scientific and technological advancements. With the exponential growth in data collection, processing, and storage, tensors have emerged as a common representation for various types of observations. For instance, image streams naturally lend themselves to a 3D tensor representation, where each image corresponds to the first and second modes, while time is represented as the third mode. However, working with tensors in high-dimensional spaces presents significant computational challenges, particularly when dealing with limited training samples. For example, consider an image stream comprising 40 images of size 30×30 pixels, where estimating a tensor-coefficient requires handling 36,000 elements. Consequently, dimensional reduction techniques often serve as

crucial preprocessing steps for further data analysis.

Multilinear Principal Component Analysis (MPCA) has gained popularity as a method for reducing the dimensionality of high-dimensional tensor data and obtaining a low-dimensional feature tensor (Lu et al. 2008). This framework leverages multilinear projection techniques to extract features that capture the majority of the original tensor's variation. MPCA tackles the original problem by decomposing it into multiple subproblems and iteratively optimizing each subproblem to derive the solution. In comparison to previous approaches that rely on vector or matrix representations (Shakhnarovich and Moghaddam 2005; Lee and Elgammal 2005; Ye et al. 2004; Yang et al. 2004; He et al. 2005), MPCA offers a systematic approach for determining effective representations of tensor objects. Due to its effectiveness in handling high-dimensional tensor data, the MPCA framework has become widely adopted in various domains. By utilizing MPCA to extract features, researchers can obtain a low-dimensional representation that facilitates further analysis and modeling of the original data. This low-dimensional representation captures the essential information from the high-dimensional tensor, enabling efficient data exploration and interpretation.

Although MPCA and other modern machine learning (ML) and deep learning (DL) technologies hold great promise across various applications, they typically demand a substantial amount of data to achieve satisfactory performance. However, meeting this requirement becomes challenging in practice due to data distribution among multiple users, resulting in limited sample sizes for each user. Additionally, growing awareness regarding data ownership and governance in modern society has led to the enactment of data privacy protection laws such as the California Consumer Privacy Act (CCPA) in the US and China's Cyber Security Law and General Provisions of Civil Law. Consequently, data silos have become more prevalent, limiting the application of these powerful methodologies across different industries.

To tackle the challenges previously mentioned, this chapter proposes a framework called Federated Multilinear Principal Component Analysis (FMPCA). FMPCA is built upon the principles of federated learning, which is a collaboratively decentralized privacy-preserving technology designed to address data silos (McMahan et al. 2017; Konečnỳ et al. 2016; Shokri and Shmatikov 2015). Federated learning enables the creation of a joint machine learning model using data distributed among multiple users, without the need to exchange data samples (Yang et al. 2019). Figure 4.1 provides an illustration of how federated learning operates. Initially, each user trains their model in a distributed manner, avoiding the need to transmit raw data to a central server in the cloud. The server then communicates with each user and aggregates local updates to construct a global model. Subsequently, each user

downloads the updated global model from the server and incorporates it into their local model. This iterative training process continues until convergence or a defined stopping criterion is met. Major companies have successfully deployed federated learning methods (Bonawitz et al. 2019; Sheller et al. 2018), playing a critical role in supporting privacy-sensitive applications where training data is distributed at the edge (Brisimi et al. 2018; Huang et al. 2020).

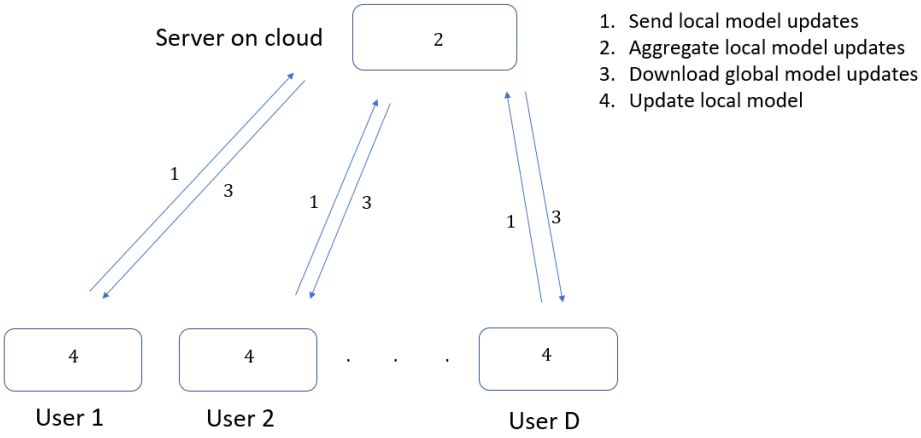


Figure 4.1: An illustration of Federated Learning.

The aim of FMPCA is to achieve performance on par with multilinear principal component analysis (MPCA) while preserving user privacy and data confidentiality. To achieve this objective, the framework proposes three algorithms that address the potential leakage of user data in MPCA. These algorithms include secure centralization of input tensor samples, secure derivation of projection matrices during initialization, and secure derivation of projection matrices during local optimization. In the first algorithm, secure centralization of input tensor samples, each user calculates their sample mean and generates perturbations for other users. The true sample mean of each user is masked by adding the perturbations from other users, and the server collects the revised sample mean of each user to calculate the global sample mean. The second and third algorithms, secure derivation of projection matrices in initialization and local optimization, utilize singular value decomposition (SVD) on a constructed matrix instead of eigen-decomposition of the covariance matrix. By incorporating user blocks into the constructed matrix and applying an updating algorithm,

the left unitary matrix of SVD is securely derived while preserving user confidentiality. The right unitary matrix is not shared during the updating process, ensuring that raw data in each user cannot be recovered by other users. All three algorithms collectively ensure the security of user data, preventing its sharing or recovery by other users or the cloud server.

The chapter 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.

4.2 The Methodology

Federated Multilinear Principal Component Analysis (FMPCA) is a methodology for extracting tensor object features based on federated learning. The primary objective of FMPCA is to achieve performance comparable to Multilinear Principal Component Analysis (MPCA) (Lu et al. 2008) while ensuring user privacy and data confidentiality. This objective is accomplished through the proposal of three algorithms that address potential privacy leakage steps encountered in MPCA: secure centralization of input tensor samples, secure derivation of projection matrices during initialization, and secure derivation of projection matrices during local optimization. By implementing these algorithms, FMPCA guarantees data security by preventing the sharing or recovery of original data in each user by the central aggregation server or other users.

In Subsection 4.2.1, we will introduce the basic tensor notations and definitions necessary for understanding the methodology. Subsection 4.2.2 provides a review of Multilinear Principal Component Analysis (MPCA). We then discuss the challenges faced by MPCA in privacy protection in Subsection 4.2.3. Finally, Subsection 4.2.4 introduces our proposed framework, Federated Multilinear Principal Component Analysis (FMPCA), which addresses the privacy concerns associated with MPCA while preserving its effectiveness in feature extraction.

4.2.1 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., $\mathscr{X}$. 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 $\mathscr{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, where I_n represents the nth mode of $\mathscr{X}$. The $(i_1, i_2, \ldots, i_N)$th entry of $\mathscr{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ is denoted by $x_{i_1, i_2, \ldots, i_n}$. A fiber of $\mathscr{X}$ 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 $\mathscr{X}$, denoted by $vec(\mathscr{X})$, is obtained by stacking all mode-1 fibers of $\mathscr{X}$. The mode-n *matricization* of a tensor $\mathscr{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$, is a matrix whose columns are mode-n *matricization* of $\mathscr{X}$ in the lexicographical order. The mode-n product of a tensor $\mathscr{X} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_N}$ and a matrix $\boldsymbol{U}_n \in \mathbb{R}^{J_n \times I_n}$, denoted by $\mathscr{X} \times_n \boldsymbol{U}_n$, is a tensor whose entry is $(\mathscr{X} \times_n \boldsymbol{U}_n)_{i_1, \ldots, i_{n-1}, j_n, i_{n+1}, \ldots, i_N} = \Sigma_{I_n=1}^{I_N} x_{i_1, \ldots, i_N} u_{j, i_n}$. The inner product of two tensors $\mathscr{A}, \mathscr{B} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ is denoted by $\langle \mathscr{A}, \mathscr{B} \rangle = \Sigma_{i_1, \ldots, i_N} a_{i_1, \ldots, i_N} b_{i_1, \ldots, i_N}$ and the Frobenius norm of $\mathscr{A}$ is defined as $\|\mathscr{A}\|_F = \sqrt{\langle \mathscr{A}, \mathscr{A} \rangle}$. 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 $\boldsymbol{A} \otimes \boldsymbol{B} = \boldsymbol{A} \odot \boldsymbol{B}$.

4.2.2 Multilinear Principal Component Analysis (MPCA)

In this subsection, we briefly review multilinear principal component analysis (MPCA) (Lu et al. 2008) before introducing our proposed federated multilinear principal component analysis (FMPCA) method.

MPCA is a feature extraction technique used for tensor objects, such as images and video sequences. Its objective is to determine a multilinear projection that captures most of the original tensorial representation. MPCA achieves this by formulating an iterative solution that decomposes the original problem into multiple subproblems. Compared to other state-of-the-art methods, MPCA-based approaches exhibit superior performance in various applications, such as gait recognition.

We consider a data set consists of M tensor objects $\{\mathscr{X}_m, m = 1, \ldots, M\}$. Each tensor object $\mathscr{X}_m \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ 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 $n = 1, \ldots, N$): $\mathscr{Y}_m = \mathscr{X}_m \times_1 \tilde{\boldsymbol{U}}^{(1)\top} \times_2 \tilde{\boldsymbol{U}}^{(2)\top} \ldots \times_N \tilde{\boldsymbol{U}}^{(N)\top}, m = 1, \ldots, M$, such that $\{\mathscr{Y}_m \in \mathbb{R}^{P_1} \otimes \mathbb{R}^{P_2} \ldots \otimes \mathbb{R}^{P_N}, m = 1, \ldots, M\}$ 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 $\Psi_{\mathscr{Y}}$:

$$\{\tilde{\boldsymbol{U}}^{(n)}, n = 1, \ldots, N\} = \arg \max_{\tilde{\boldsymbol{U}}^{(1)}, \tilde{\boldsymbol{U}}^{(2)}, \ldots, \tilde{\boldsymbol{U}}^{(N)}} \Psi_{\mathscr{Y}}. \tag{4.1}$$

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) transfers 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 (4.1). 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*

$$\boldsymbol{\Phi}^{(n)} = \Sigma_{m=1}^{M} (\boldsymbol{X}_{m(n)} - \bar{\boldsymbol{X}}_{(n)}) \cdot \tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}} \cdot \tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}}^{\top} \cdot (\boldsymbol{X}_{m(n)} - \bar{\boldsymbol{X}}_{(n)})^{\top} \tag{4.2}$$

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 (4.1). Instead, (Lu et al. 2008) utilizes an iteration procedure to compute the projection matrices one by one with all the others fixed (shown in local optimization of Algorithm 2). 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 (2) below.

Algorithm 2: MPCA algorithm

Input: A set of tensor samples $\{\mathcal{X}_m \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}, m = 1, \ldots, M\}$.

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.

Algorithm:

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.

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 $n = 1, \ldots, N$.

Step 3 (Local Optimization):

• Calculate $\{\tilde{\mathcal{Y}}_m = \tilde{\mathcal{X}}_m \times_1 \tilde{\boldsymbol{U}}^{(1)^\top} \times_2 \tilde{\boldsymbol{U}}^{(2)^\top} \ldots \times_N \tilde{\boldsymbol{U}}^{(N)^\top}, m = 1, 2, \ldots, M\}$.

• 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).

for $k = 1 : K$ **do**

  for $n = 1 : N$ **do**

    * Set the matrix $\tilde{\boldsymbol{U}}^{(n)}$ to consist of the P_n eigenvectors of the matrix $\mathbf{\Phi}^{(n)}$, as defined in Equation (4.2), corresponding to the largest P_n eigenvalues.

  Calculate $\{\tilde{\mathcal{Y}}_m, m = 1, \ldots, M\}$ and $\Psi_{\mathcal{Y}_k}$.

  If $\Psi_{\mathcal{Y}_k} - \Psi_{\mathcal{Y}_{k-1}} \leq \eta$, break and go to Step 4

Step 4 (Projection): The feature tensor after projection is obtained as $\{\mathcal{Y}_m = \mathcal{X}_m \times_1 \tilde{\boldsymbol{U}}^{(1)^\top} \times_2 \tilde{\boldsymbol{U}}^{(2)^\top} \ldots \times_N \tilde{\boldsymbol{U}}^{(N)^\top}, m = 1, 2, \ldots, M\}$.

4.2.3 The Challenges of MPCA in Privacy-Protection

In this subsection, we will discuss three steps of MPCA that raise privacy leakage concerns.

MPCA (Lu et al. 2008) and many other modern machine learning (ML) and deep learning (DL) technologies typically require a large amount of data to achieve satisfactory performance. However, in the real world, data is distributed among multiple users, resulting in small sample sizes for individual users. Additionally, there is an increasing societal focus on user privacy and data confidentiality, which has led to new laws governing how data should be managed and used. In this context, traditional approaches that involve collecting and sharing data to a central location are no longer appropriate. 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 - $\{\mathcal{X}_{m_d}^{d},\ m_d = 1, \ldots, M_d\}$. For example, we use $\mathcal{X}_3^1$ to

represent the 3rd high-dimensional tensor sample of user 1.

The first privacy-protection challenge of MPCA occurs in the step 1 of Algorithm 2 - 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:

$$\bar{\mathcal{X}} = \frac{\sum_{d=1}^{D}\sum_{m_d=1}^{M_d}\mathcal{X}^{d}_{m_d}}{\sum_{d=1}^{D} M_d}. \tag{4.3}$$

However, privacy protection for distributed data 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 $d, d = 1,\ldots,D$ exposes its data to the server and the other users.

The second privacy-protection challenge of MPCA exists in the step 2 of Algorithm 2 - 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 (4.3), 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:

$$\boldsymbol{\Phi}^{(n)*} = \sum_{d=1}^{D}\sum_{m_d=1}^{M_d}\tilde{\boldsymbol{X}}^{d}_{m_d(n)}\cdot\tilde{\boldsymbol{X}}^{d\top}_{m_d(n)}, \tag{4.4}$$

where $\tilde{\boldsymbol{X}}^{d\top}_{m_d(n)}$ is the transpose of matrix $\tilde{\boldsymbol{X}}^{d}_{m_d(n)}$. Therefore, the process of generating the combined dataset and the covariance matrix $\boldsymbol{\Phi}^{(n)*}$ results in data privacy violations, as it leaks original data information from 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 $\{\tilde{\mathcal{X}}^{d}_{m_d},\ m_d = 1,\ldots,M_d;\ d = 1,\ldots,D\}$ 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:

$$\mathbf{\Phi}^{(n)} = \Sigma_{d=1}^{D} \Sigma_{m_d=1}^{M_d} (\boldsymbol{X}_{m_d(n)}^{d} - \bar{\boldsymbol{X}}_{(n)}) \cdot \tilde{\boldsymbol{U}}_{\mathbf{\Phi}^{(n)}} \cdot \tilde{\boldsymbol{U}}_{\mathbf{\Phi}^{(n)}}^{\top} \cdot (\boldsymbol{X}_{m_d(n)}^{d} - \bar{\boldsymbol{X}}_{(n)})^{\top}, \tag{4.5}$$

where

$$\tilde{\boldsymbol{U}}_{\mathbf{\Phi}^{(n)}} = \left(\tilde{\boldsymbol{U}}^{(n+1)} \otimes \tilde{\boldsymbol{U}}^{(n+2)} \otimes \ldots \otimes \tilde{\boldsymbol{U}}^{(N)} \otimes \tilde{\boldsymbol{U}}^{(1)} \otimes \tilde{\boldsymbol{U}}^{(2)} \otimes \ldots \otimes \tilde{\boldsymbol{U}}^{(n-1)}\right), \tag{4.6}$$

$\boldsymbol{X}_{m_d(n)}^{d}$ 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 $\bar{\mathcal{X}}$ of the combined dataset $\{\tilde{\mathcal{X}}_{m_d}^{d},\ m_d = 1,\ldots,M_d;\ d = 1,\ldots,D\}$ in Equation (4.3).

To address these challenges, we will propose three federated learning based methods which prevent MPCA from privacy leakage in Subsection 4.2.4 - those are - secure centralization of input tensor samples in 4.2.4.1, secure derivation of projection matrices in initialization in 4.2.4.2, secure derivation of projection matrices in local optimization in 4.2.4.3.

4.2.4 Federated Multilinear Principal Component Analysis (FMPCA)

As discussed in Section 4.2.2, Multilinear Principal Component Analysis (MPCA) involves collecting all the M high-dimensional tensor samples at a central aggregation server to derive low-dimensional feature tensors using the combined dataset. However, as mentioned in Section 4.1, the issue of data silos can make it challenging for the central aggregation server to collect data from all D users. This poses a risk to user privacy as each user d, $d = 1,\ldots,D$ exposes their data to the server and to other users during the collection process. Therefore, it is essential to seek solutions that ensure user confidentiality while achieving the same or approximate results through data combination.

Subsection 4.2.4 is organized as follows: we initially discuss 3 steps in MPCA which have concern of data privacy violation - those are - input samples centralization in 4.2.4.1, projection matrices derivation of initialization in 4.2.4.2, projection matrices derivation of local optimization in 4.2.4.3. Next, 4.2.4.4 will introduce our proposed Federated Multilinear Principal Component Analysis (FMPCA) as a whole.

4.2.4.1 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 $\{\tilde{\mathcal{X}}_m = \mathcal{X}_m - \bar{\mathcal{X}}, m = 1, \ldots, M\}$, 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:

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

where M_d is the number of high-dimensional tensor samples of the dth user and $\mathcal{X}^d_{m_d}$ 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.

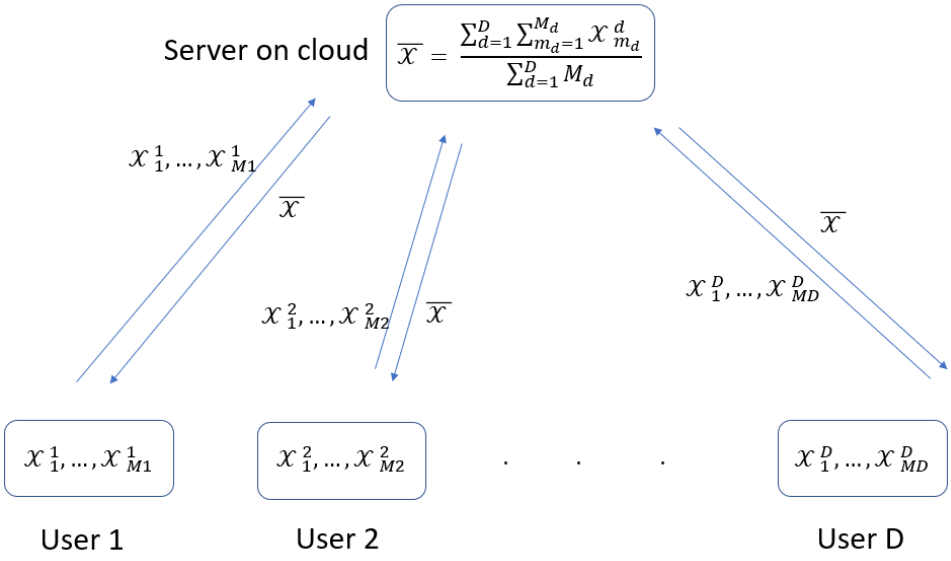


Figure 4.2: Centralization with privacy leakage.

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

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

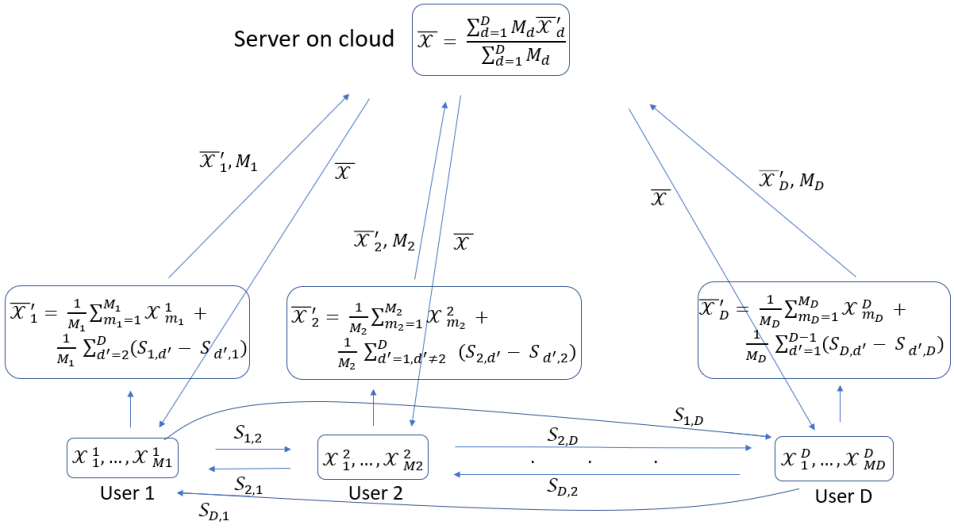


Figure 4.3: Centralization without privacy leakage.

where M_d denotes the number of high-dimensional tensor samples of user d, $\mathcal{X}^d_{m_d}$ denotes the m_dth high-dimensional tensor sample of user d, where $m_d = 1, \ldots, M_d$ 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:

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

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:

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

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

Algorithm 3 outlines our proposed method for the secure centralization of input tensor samples. Our approach aims to prevent privacy leakage by ensuring that the raw high-dimensional tensor samples and the true sample of each user cannot be detected by the server or other users. Moreover, our method guarantees the correctness of the result as all perturbations are cancelled out during summation. A detailed proof of the correctness of our approach can be found in the appendix.

Algorithm 3: Secure centralization algorithm

1 **Input**: A set of D users where each user has M_d high-dimensional tensor samples $\{\mathcal{X}_{m_d}^{d} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}, m_d = 1,\ldots,M_d; d = 1,\ldots,D\}$, 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: $\bar{\mathcal{X}}_d = \frac{1}{M_d}\sum_{m_d=1}^{M_d} \mathcal{X}_{m_d}^{d}, d = 1,\ldots,D$

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

4.2.4.2 Secure Derivation of Projection Matrices in Initialization

In 4.2.4.2, 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 2, MPCA obtains the projection matrices $\{\tilde{\boldsymbol{U}}^{(n)}, n = 1,\ldots,N\}$ by calculating the eigen-decomposition of $\boldsymbol{\Phi}^{(n)*} = \sum_{m=1}^{M} \tilde{\boldsymbol{X}}_{m(n)} \cdot \tilde{\boldsymbol{X}}_{m(n)}^{\top}$,

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:

$$\boldsymbol{\Phi}^{(n)*} = \sum_{d=1}^{D} \sum_{m_d=1}^{M_d} \tilde{\boldsymbol{X}}^{d}_{m_d(n)} \cdot \tilde{\boldsymbol{X}}^{d\top}_{m_d(n)}, \tag{4.7}$$

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 $\boldsymbol{\Phi}^{(n)*}$, 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 4.4 shows the procedure flow of projection matrices derivation in the initialization of MPCA.

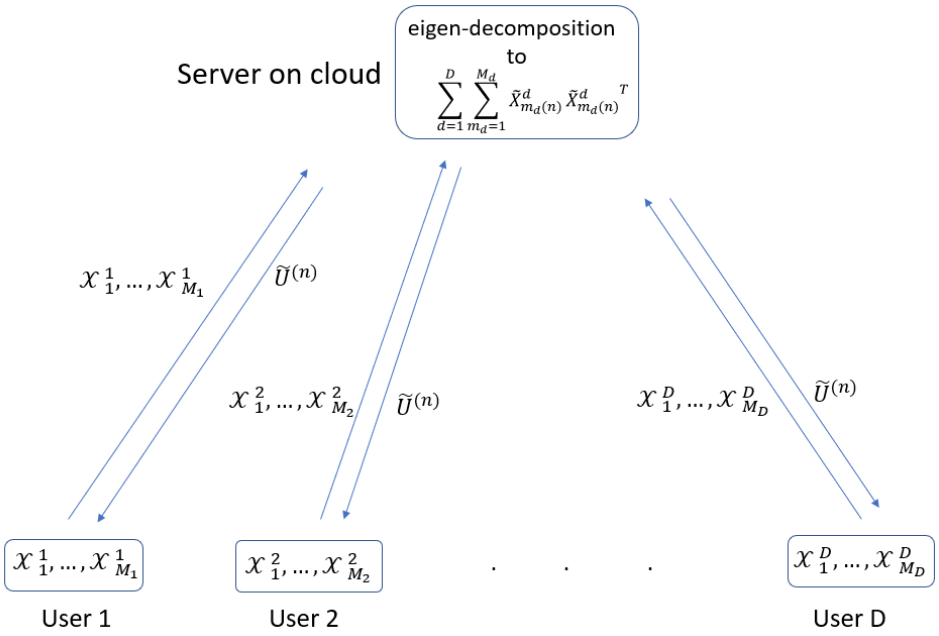


Figure 4.4: Derivation of projection matrices in initialization with privacy leakage.

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 $\boldsymbol{\Phi}^{(n)*}$ in Equation (4.7) 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 11 is introduced:

Proposition 11. *When there are multiple users, the projection matrix $\tilde{\boldsymbol{U}}^{(n)}$ in the initialization of MPCA can be derived by two equivalent 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 (4.7), 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 $\boldsymbol{A}_{(n)}$. The projection matrix $\tilde{\boldsymbol{U}}^{(n)}$ to be the left unitary matrix correspondingly.

The proof of Proposition 11 can be found in the Appendix.
Based on Proposition 11, 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 4.5 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 11, 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 12 is introduced.

Proposition 12. *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 $\boldsymbol{A}_{(n)}$ 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 12 can be found in the Appendix.

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 $\boldsymbol{A}_{d(n)}, 1 \le j \le (I_1 \times \ldots \times I_{n-1} \times I_{n+1} \times \ldots \times I_N \times M_d)$, $\boldsymbol{e}$ 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:

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

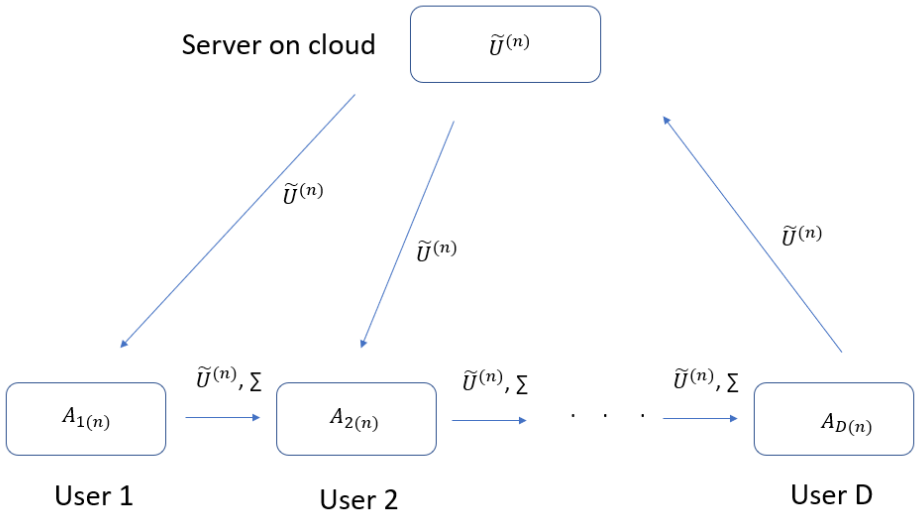


Figure 4.5: Derivation of projection matrices in initialization without privacy leakage.

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 $\tilde{\boldsymbol{U}}^{(n)}$, $\mathbf{0} \in \mathbb{R}^{1 \times r}$ 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: $\tilde{\boldsymbol{U}}^{(n)} = \left[\tilde{\boldsymbol{U}}^{(n)} \quad \frac{\boldsymbol{e}}{\|\boldsymbol{e}\|}\right] \hat{\tilde{\boldsymbol{U}}}$, $\boldsymbol{\Sigma} = \hat{\boldsymbol{\Sigma}}$, 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 4 summarizes our proposed method for the secure derivation of projection matrices during initialization. Our approach ensures data privacy protection as the high-dimensional tensor samples in each user cannot be shared or recovered by other users or the central aggregation server.

Algorithm 4: Secure derivation of projection matrices in initialization

Input: A set of D users where each user has centralized high-dimensional tensor samples $\{\tilde{\mathcal{X}}_{m_d}^{d} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N},\ m_d = 1,\ldots,M_d;\ d = 1,\ldots,D\}$.

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 nth mode - $\{\tilde{\boldsymbol{X}}_{m_d(n)}^{d} \in \mathbb{R}^{I_n \times (I_1 \times \ldots \times I_{n-1} \times I_{n+1} \times \ldots \times I_N)},\ m_d = 1,\ldots,M_d\}$ and then concatenate the unfolded tensor samples horizontally - $[\tilde{\boldsymbol{X}}_{1(n)}^{d}, \tilde{\boldsymbol{X}}_{2(n)}^{d}, \ldots, \tilde{\boldsymbol{X}}_{M_d(n)}^{d}]$ 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**

    $\boldsymbol{s} \coloneqq \tilde{\boldsymbol{A}}_{d(n)}(:, j)$ % the j th column of $\tilde{\boldsymbol{A}}_{d(n)}$

    $\boldsymbol{w} \coloneqq \tilde{\boldsymbol{U}}^{(n)^{\top}} \boldsymbol{s}$

    $\boldsymbol{p} \coloneqq \tilde{\boldsymbol{U}}^{(n)} \boldsymbol{w}$

    $\boldsymbol{e} \coloneqq \boldsymbol{s} - \boldsymbol{p}$

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

    $\tilde{\boldsymbol{U}}^{(n)} \coloneqq \left[\tilde{\boldsymbol{U}}^{(n)} \quad \frac{\boldsymbol{e}}{\|\boldsymbol{e}\|}\right] \hat{\tilde{\boldsymbol{U}}}$

    $\boldsymbol{\Sigma} \coloneqq \hat{\boldsymbol{\Sigma}}$

    if $\| \boldsymbol{e} \| \leq \varepsilon$ **then**

      $\tilde{\boldsymbol{U}}^{(n)} \coloneqq \tilde{\boldsymbol{U}}^{(n)}_{1:I_n, 1:r}$ % delete the last column

      $\boldsymbol{\Sigma} \coloneqq \boldsymbol{\Sigma}_{1:r, 1:r}$ % delete the last row and the last column

    else

      $r \coloneqq r + 1$

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 $\tilde{\boldsymbol{U}}^{(n)} \in \mathbb{R}^{I_n \times P_n}$

4.2.4.3 Secure Derivation of Projection Matrices in Local Optimization

Subsection 4.2.4.3 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 2, to derive the projection matrix $\{\tilde{\boldsymbol{U}}^{(n)}, n = 1,\ldots,N\}$, (Lu et al. 2008) conducts eigen-decomposition to the matrix $\boldsymbol{\Phi}^{(n)}$ which is shown as follows:

$$\boldsymbol{\Phi}^{(n)} = \Sigma_{m=1}^{M} (\boldsymbol{X}_{m(n)} - \bar{\boldsymbol{X}}_{(n)}) \cdot \tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}} \cdot \tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}}^{\top} \cdot (\boldsymbol{X}_{m(n)} - \bar{\boldsymbol{X}}_{(n)})^{\top} \tag{4.8}$$

where

$$\tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}} = \left(\tilde{\boldsymbol{U}}^{(n+1)} \otimes \tilde{\boldsymbol{U}}^{(n+2)} \otimes \ldots \otimes \tilde{\boldsymbol{U}}^{(N)} \otimes \tilde{\boldsymbol{U}}^{(1)} \otimes \tilde{\boldsymbol{U}}^{(2)} \otimes \ldots \otimes \tilde{\boldsymbol{U}}^{(n-1)}\right), \tag{4.9}$$

$\boldsymbol{X}_{m(n)}$ 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:

$$\boldsymbol{\Phi}^{(n)} = \Sigma_{d=1}^{D} \Sigma_{m_d=1}^{M_d} (\boldsymbol{X}_{m_d(n)}^{d} - \bar{\boldsymbol{X}}_{(n)}) \cdot \tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}} \cdot \tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}}^{\top} \cdot (\boldsymbol{X}_{m_d(n)}^{d} - \bar{\boldsymbol{X}}_{(n)})^{\top}, \tag{4.10}$$

where

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

$\boldsymbol{X}_{m_d(n)}^{d}$ 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 $\{\tilde{\mathscr{X}}_{m_d}^{d},\ m_d = 1,\ldots,M_d;\ d = 1,\ldots,D\}$. 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 4.6 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:

$$\boldsymbol{A}_{(n)}^{\boldsymbol{\Phi}} = [\tilde{\boldsymbol{X}}_{1(n)}^{1\boldsymbol{\Phi}}, \ldots, \tilde{\boldsymbol{X}}_{M_1(n)}^{1\boldsymbol{\Phi}}, \tilde{\boldsymbol{X}}_{1(n)}^{2\boldsymbol{\Phi}}, \ldots, \tilde{\boldsymbol{X}}_{M_2(n)}^{2\boldsymbol{\Phi}}, \ldots, \tilde{\boldsymbol{X}}_{1(n)}^{D\boldsymbol{\Phi}}, \ldots, \tilde{\boldsymbol{X}}_{M_D(n)}^{D\boldsymbol{\Phi}}] \tag{4.11}$$

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 in Equation (4.9). 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 $\boldsymbol{A}_{(n)}^{\boldsymbol{\Phi}}$ which has the same performance as directly applying SVD to $\boldsymbol{A}_{(n)}^{\boldsymbol{\Phi}}$.

To show that the projection matrix $\tilde{\boldsymbol{U}}^{(n)}$ in local optimization of MPCA can be derived by applying SVD to the matrix $\boldsymbol{A}_{(n)}^{\boldsymbol{\Phi}}$ in Equation (4.11), Proposition 13 is introduced:

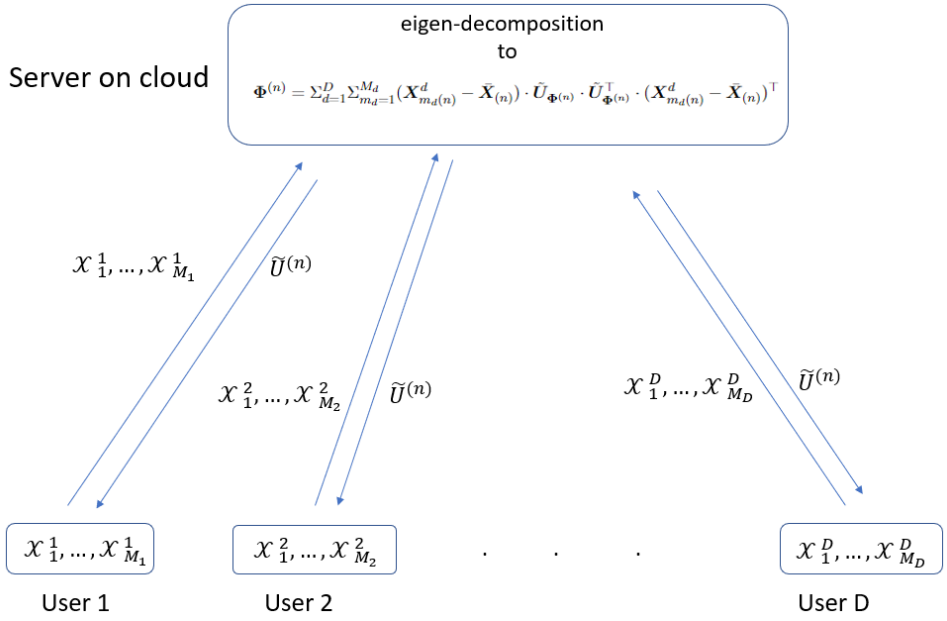


Figure 4.6: Derivation of projection matrices in local optimization with privacy leakage.

Proposition 13. *When there are multiple users, the projection matrix* $\tilde{\boldsymbol{U}}^{(n)}$ *in the local optimization of MPCA can be derived by two equivalent methods:*

1. Calculate the eigen-decomposition of the covariance matrix $\mathbf{\Phi}^{(n)} = \Sigma_{d=1}^{D}\Sigma_{m_d=1}^{M_d}(\boldsymbol{X}^{d}_{m_d(n)} - \bar{\boldsymbol{X}}_{(n)}) \cdot \tilde{\boldsymbol{U}}_{\mathbf{\Phi}^{(n)}} \cdot \tilde{\boldsymbol{U}}^{\top}_{\mathbf{\Phi}^{(n)}} \cdot (\boldsymbol{X}^{d}_{m_d(n)} - \bar{\boldsymbol{X}}_{(n)})^{\top}$ *in Equation (4.10), and set the projection matrix* $\tilde{\boldsymbol{U}}^{(n)}$ *to comprise all the eigenvectors.*

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

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

where $\{\tilde{\boldsymbol{X}}^{d\mathbf{\Phi}}_{m_d(n)} = \tilde{\boldsymbol{X}}^{d}_{m_d(n)}\tilde{\boldsymbol{U}}_{\mathbf{\Phi}^{(n)}},\ m_d = 1, \ldots, M_d;\ d = 1, \ldots, D\}$ *are concatenated horizontally in* $\boldsymbol{A}^{\mathbf{\Phi}}_{(n)}$, $\tilde{\boldsymbol{X}}^{d}_{m_d(n)}$ *is the* n*th mode matricization of the* m_d*th high-dimensional tensor sample in user* d*, and* $\tilde{\boldsymbol{U}}_{\mathbf{\Phi}^{(n)}}$ *is calculated in Equation (4.9). The projection matrix* $\tilde{\boldsymbol{U}}^{(n)}$ *is set to be the left unitary matrix correspondingly.*

Proposition 13 can be found in the Appendix.

Based on Proposition 13, the left unitary matrix of applying singular value decomposition (SVD) to $\boldsymbol{A}^{\mathbf{\Phi}}_{(n)}$ in Equation (4.11) 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 $\boldsymbol{A}^{\mathbf{\Phi}}_{(n)}$ in

Equation (4.11) 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 (4.11) securely. Figure 4.7 shows the procedure flow of our proposed method whose steps are similar to the secure derivation of projection matrix in initialization in Subsection 20. We initially transform $\boldsymbol{A}^{\boldsymbol{\Phi}}_{(n)}$ by adding user blocks:

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

where $\boldsymbol{A}^{\boldsymbol{\Phi}}_{d(n)} = [\tilde{\boldsymbol{X}}^{d\boldsymbol{\Phi}}_{1(n)}, \ldots, \tilde{\boldsymbol{X}}^{d\boldsymbol{\Phi}}_{M_d(n)}]$, $\{\tilde{\boldsymbol{X}}^{d\boldsymbol{\Phi}}_{m_d(n)} = \tilde{\boldsymbol{X}}^{d}_{m_d(n)} \tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}},\ m_d = 1, \ldots, M_d;\ d = 1, \ldots, D\}$, $\tilde{\boldsymbol{X}}^{d}_{m_d(n)}$ 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 (4.9). 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 $\{\boldsymbol{A}^{\boldsymbol{\Phi}}_{d(n)}, d = 2, \ldots, D\}$ 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 14. *The projection matrix $\boldsymbol{U}^{(n)}$ derived from Algorithm (5) is equivalent to the left unitary matrix of applying singular value decomposition (SVD) to a matrix $\boldsymbol{A}^{\boldsymbol{\Phi}}_{(n)}$ in Equation (4.11).*

The proof of Proposition 14 can be found in the Appendix.

Algorithm 4 5 summarizes our proposed method for the secure derivation of projection matrices during the local optimization step of MPCA. As the updating process in Algorithm 5 is the same as the secure derivation of projection matrix during initialization in Subsection 8, we will not provide a detailed introduction. Our approach ensures privacy protection by preventing the sharing or recovery of high-dimensional tensor samples in each user by other users or the central aggregation server.

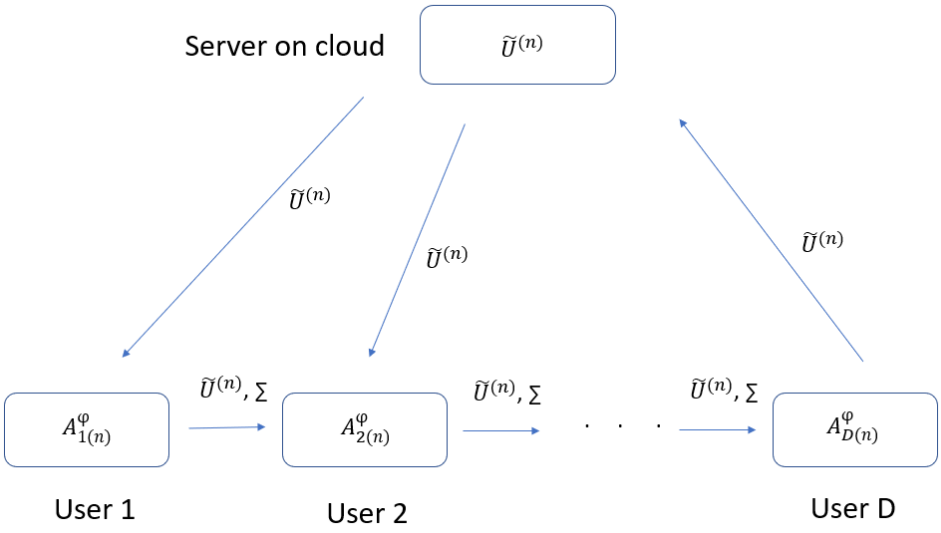


Figure 4.7: Derivation of projection matrices in local optimization without privacy leakage.

Algorithm 5: Secure derivation of projection matrices in local optimization

Input: A set of D users where each user has centralized high-dimensional tensor samples $\{\tilde{\mathcal{X}}_{m_d}^{d} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N},\ m_d = 1,\ldots,M_d;\ d = 1,\ldots,D\}$, all the projection matrices except the nth mode $\{\tilde{\boldsymbol{U}}^{(1)},\ldots,\tilde{\boldsymbol{U}}^{(n-1)},\tilde{\boldsymbol{U}}^{(n+1)},\ldots,\tilde{\boldsymbol{U}}^{(N)}\}$

Output: Projection matrix in the nth mode $\tilde{\boldsymbol{U}}^{(n)} \in \mathbb{R}^{I_n \times P_n}$.

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 nth mode which generates $\{\tilde{\boldsymbol{X}}_{m_d(n)}^{d}, m_d = 1,\ldots,M_d\}$. Then each user calculates $\{\tilde{\boldsymbol{X}}_{m_d(n)}^{d\boldsymbol{\Phi}} = \tilde{\boldsymbol{X}}_{m_d(n)}^{d}\tilde{\boldsymbol{U}}_{\boldsymbol{\Phi}^{(n)}},\ m_d = 1,\ldots,M_d\}$ and concatenate $\{\tilde{\boldsymbol{X}}_{m(n)}^{d\boldsymbol{\Phi}},\ m_d = 1,\ldots,M_d\}$ horizontally - $[\tilde{\boldsymbol{X}}_{1(n)}^{d\boldsymbol{\Phi}}, \tilde{\boldsymbol{X}}_{2(n)}^{d\boldsymbol{\Phi}},\ldots,\tilde{\boldsymbol{X}}_{M_d(n)}^{d\boldsymbol{\Phi}}]$ which is denoted as $\boldsymbol{A}_{d(n)}^{\boldsymbol{\Phi}} \in \mathbb{R}^{P_1 \times \ldots \times P_{n-1} \times I_n \times P_{n+1} \times \ldots \times P_N \times M_d}$.

Initialization: Set $\varepsilon = 1 \times e^{-6}$

User 1: $\tilde{\boldsymbol{A}}_{d(n)}^{\boldsymbol{\Phi}} = \tilde{\boldsymbol{U}}^{(n)}\boldsymbol{\Sigma}\tilde{\boldsymbol{V}}$, $\tilde{\boldsymbol{U}}^{(n)}$ and $\boldsymbol{\Sigma}$ are shared with user 2.

for ***User*** $d = 2 : D$ **do**

  for ***Column*** $j = 1 : (P_1 \times \ldots \times P_{n-1} \times P_{n+1} \times \ldots P_N \times M_d)$ **do**

    $\boldsymbol{s} \coloneqq \tilde{\boldsymbol{A}}_{d(n)}^{\Phi}(:, j)$ % the jth column of $\tilde{\boldsymbol{A}}_{d(n)}^{\boldsymbol{\Phi}}$

    $\boldsymbol{w} \coloneqq \tilde{\boldsymbol{U}}^{(n)^\top}\boldsymbol{s}$

    $\boldsymbol{p} \coloneqq \tilde{\boldsymbol{U}}^{(n)}\boldsymbol{w}$

    $\boldsymbol{e} \coloneqq \boldsymbol{s} - \boldsymbol{p}$

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

    $\tilde{\boldsymbol{U}}^{(n)} = \left[\tilde{\boldsymbol{U}}^{(n)} \quad \frac{\boldsymbol{e}}{\|\boldsymbol{e}\|}\right]\hat{\tilde{\boldsymbol{U}}}$

    $\boldsymbol{\Sigma} \coloneqq \hat{\boldsymbol{\Sigma}}$

    if $\| \boldsymbol{e} \| \leq \varepsilon$ **then**

      $\tilde{\boldsymbol{U}}^{(n)} \coloneqq \tilde{\boldsymbol{U}}_{1:I_n,1:r}^{(n)}$ % delete the last column

      $\boldsymbol{\Sigma} \coloneqq \boldsymbol{\Sigma}_{1:r,1:r}$ % delete both the last row and last column

    else

      $r \coloneqq r + 1$

Set the projection matrix $\tilde{\boldsymbol{U}}^{(n)} \in \mathbb{R}^{I_n \times d}$ only contain the first P_n columns and generate $\tilde{\boldsymbol{U}}^{(n)} \in \mathbb{R}^{I_n \times P_n}$.

4.2.4.4 Overview of Federated Multilinear Principal Component Analysis (FMPCA)

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

To overcome the three privacy-protection challenges of MPCA discussed in Subsection

4.2.3, FMPCA revises MPCA by three proposed methods - those are - secure centralization of input tensor samples in 4.2.4.1, secure derivation of projection matrices in initialization in 4.2.4.2 and secure derivation of projection matrices in local optimization in 4.2.4.3.

Algorithm 6 summarizes the whole procedures of FMPCA. Given a set of D users where each user has M_d high-dimensional tensor samples $\{\mathscr{X}_{m_d}^d \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}, m_d = 1,\ldots,M_d; d = 1,\ldots,D\}$, the server initially applies the secure aggregation in Algorithm 3 to derive the mean tensor $\bar{\mathscr{X}}$ of all the users. Then each user $d,\ d = 1,\ldots,D$ downloads $\bar{\mathscr{X}}$ from the server and centralizes its tensor samples as $\{\tilde{\mathscr{X}}_{m_d}^d = \mathscr{X}_{m_d}^d - \bar{\mathscr{X}},\ m_d = 1,\ldots,M_d;\ d = 1,\ldots,D\}$. 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 4 which is discussed in 4.2.4.2 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:

$$\tilde{\mathscr{Y}}_{m_d}^d = \tilde{\mathscr{X}}_{m_d}^d \times_1 \tilde{\boldsymbol{U}}^{(1)\top} \times_2 \tilde{\boldsymbol{U}}^{(2)\top} \ldots \times_N \tilde{\boldsymbol{U}}^{(N)\top},\ m_d = 1,\ldots,M_d;\ d = 1,\ldots,D. \tag{4.12}$$

where $\mathscr{Y}_{m_d}^d$ 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 $\{\Sigma_{m_d=1}^{M_d} \|\tilde{\mathscr{Y}}_{m_d}^d\|_F^2, d = 1,\ldots,D\}$ 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 $\Psi_{\mathscr{Y}_0}$ by operating a summation step:

$$\Psi_{\mathscr{Y}_0} = \Sigma_{d=1}^{D} \Sigma_{m_d=1}^{M_d} \|\tilde{\mathscr{Y}}_{m_d}^d\|_F^2 \tag{4.13}$$

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

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

On the opposite, if each user d only sends the sum of squared Frobenius norm of low-dimensional tensors $\Sigma_{m_d=1}^{M_d} \|\tilde{\mathscr{Y}}_{m_d}^d\|_F^2$ 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 4.2.2. Specifically, in each iteration K, we derive $\{\tilde{\boldsymbol{U}}^{(n)}, n = 1,\ldots,N\}$ from algorithm 5 in 4.2.4.3. The total scatter variation of low-dimensional tensor $\Psi_{\mathscr{Y}_k}$ can be derived following the same procedure as the generation of $\Psi_{\mathscr{Y}_0}$.

Algorithm 6: FMPCA updating algorithm

1 **Input:** A set of D users where each user has M_d high-dimensional tensor samples $\{\mathscr{X}_{m_d}^d \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}, m_d = 1,\ldots,M_d;\ d = 1,\ldots,D\}$.

2 **Output:** A set of D users where each user has M_d low-dimensional tensor samples after projection $\{\mathscr{Y}_{m_d}^d \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}, m_d = 1,\ldots,M_d;\ d = 1,\ldots,D\}$.

3 **Step 1 (Preprocessing):** Centralize $\{\mathscr{X}_{m_d}^d, m_d = 1,\ldots,M_d;\ d = 1,\ldots,D\}$ to $\{\tilde{\mathscr{X}}_{m_d}^d, m_d = 1,\ldots,M_d;\ d = 1,\ldots,D\}$ by Algorithm 3.

4 **Step 2 (Initialization):** Initialize $\{\tilde{\boldsymbol{U}}^{(n)}, n = 1,\ldots,N\}$ by Algorithm 4.

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

7 Each user $d, d = 1,\ldots,D$, calculates its sum of squared Frobenius norm of low-dimensional tensors $\Sigma_{m_d=1}^{M_d} \parallel \tilde{\mathscr{Y}}_{m_d}^d \parallel_F^2$ privately and send it to the server.

8 The server calculates $\Sigma_{d=1}^{D}\Sigma_{m_d=1}^{M_d} \parallel \tilde{\mathscr{Y}}_{m_d}^d \parallel_F^2$ and denote it as $\boldsymbol{\Psi}_{\mathscr{Y}_0}$.

9 **for** $k = 1 : K$ **do**

10 **for** $n = 1 : N$ **do**

11 Calculate $\tilde{\boldsymbol{U}}^{(n)}$ by Algorithm 5

12 Follow the generation of $\Psi_{\mathscr{Y}_0}$ in the line 6 - 8, calculate $\Psi_{\mathscr{Y}_k}$

13 If $\Psi_{\mathscr{Y}_k} - \Psi_{\mathscr{Y}_{k-1}} \leq \eta$, break and go to Step 4

14 **Step 4 (Projection):**

15 For each user $d, d = 1,\ldots,D$, the low-dimensional feature tensor is obtained as $\{\mathscr{Y}_{m_d} = \mathscr{X}_{m_d}^d \times_1 \tilde{\boldsymbol{U}}^{(1)^\top} \times_2 \tilde{\boldsymbol{U}}^{(2)^\top} \ldots \times_N \boldsymbol{U}^{(N)^\top}, m_d = 1,2,\ldots,M_d\}$

4.3 Prognostics Model Construcion and Real-Time TTF Prediction

In this section, we provide a real-world example of the high-dimensional tensor samples and response used in MPCA (Lu et al. 2008). We will then discuss the development of a

prognostic model that can predict the response based on these high-dimensional tensor samples.

In this chapter, we consider degradation image streams and time-to-failure (TTF) as the high-dimensional tensor samples and response, respectively. Degradation refers to the process of damage accumulation, which can be monitored by sensing technology, resulting in data known as degradation data/signals. Prognostics is a process used to predict the TTF of assets. We have chosen imaging-based degradation data because it provides a wealth of information about the monitored object, is usually non-contact, and does not require permanent installation or fixturing.

To relate the response (TTF) and the low-dimensional feature tensor, we use a (log)-location-scale (LLS) tensor regression model. LLS regression has been commonly used in reliability and survival analysis (Doray 1994), as it allows for a range of TTF distributions to be considered, such as (log)normal, (log)logistic, smallest extreme value (SEV), and Weibull.

Similar to subsection 20, we denote the training high-dimensional tensor sample set as $\{\mathcal{X}_{m_d}^{d} \in \mathbb{R}^{I_1 \times I_2 \times \ldots I_N}, m_d = 1,\ldots,M_d; d = 1,\ldots,D\}$, where D is the number of users, M_d is the number of high-dimensional tensor samples in user d and use $\{z_{m_d}^{d} \in \mathbb{R}, m_d = 1,\ldots,M_d; d = 1,\ldots,D\}$ 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 $\{\mathcal{X}_t \in \mathscr{R}^{I_1 \times \ldots \times I_n}, t = 1,\ldots,T\}$, 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 $\{\mathcal{X}_{m_d}^{d} \in \mathbb{R}^{I_1 \times I_2 \times \ldots I_N}, m_d = 1,\ldots,M_d; d = 1,\ldots,D\}$. 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:

$$\hat{\mathcal{Y}}_{m_d}^{d} = \mathcal{X}_{m_d}^{d} \times_1 \boldsymbol{U}_1^{\top} \times_2 \boldsymbol{U}_2^{\top} \ldots \times_N \boldsymbol{U}_N^{\top},\ m_d = 1,\ldots,M_d;\ d = 1,\ldots,D. \tag{4.14}$$

$$\hat{\mathcal{Y}}_t = \mathcal{X}_t \times_1 \boldsymbol{U}_1^{\top} \times_2 \boldsymbol{U}_2^{\top} \ldots \times_N \boldsymbol{U}_N^{\top},\ t = 1,\ldots,T. \tag{4.15}$$

where $\{\hat{\mathcal{Y}}_{m_d}^{d},\ m_d = 1,\ldots,M_d;\ d = 1,\ldots,D\}$ 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 $\{\hat{\mathscr{Y}}_{m_d}^d,\ m_d = 1,\ldots,M_d;\ d = 1,\ldots,D\}$ 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:

$$z_{m_d}^d = \beta_0 + vec(\hat{\mathscr{Y}}_{m_d}^d)^\top \boldsymbol{\beta}_1 + \sigma\epsilon_m, \tag{4.16}$$

where $vec(\hat{\mathscr{Y}}_{m_d}^d)$ is the vectorization of $\hat{\mathscr{Y}}_m$, $\beta_0 \in \mathbb{R}$ 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 (4.16) can be estimated by solving the following optimization problem:

$$\min_{\boldsymbol{z},\beta_0,\boldsymbol{\beta}_1,\sigma} \ell\left(\frac{\boldsymbol{z} - \mathbf{1}_M\beta_0 - \hat{\boldsymbol{Y}}\boldsymbol{\beta}_1}{\sigma}\right), \tag{4.17}$$

where $M = \Sigma_{d=1}^D M_d$ represents combining the high-dimensional tensor samples of D users together, $\hat{\boldsymbol{Y}} = (vec(\hat{\mathscr{Y}}_1)^\top, vec(\hat{\mathscr{Y}}_2)^\top, \ldots, vec(\hat{\mathscr{Y}}_M)^\top)^\top$, $\ell(\cdot)$ is the negative log-likelihood function of a location-scale distribution. For example, if TTFs follow normal distributions, then $\ell(\frac{\boldsymbol{z}-\mathbf{1}_M\beta_0-\hat{\boldsymbol{Y}}\boldsymbol{\beta}_1}{\sigma}) = \frac{M}{2}\log 2\pi + M\log\sigma + \frac{1}{2}\Sigma_{m=1}^M\omega_m^2$, 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 $\ell(\frac{\boldsymbol{z}-\mathbf{1}_M\beta_0-\hat{\boldsymbol{Y}}\boldsymbol{\beta}_1}{\sigma}) = M\log\sigma - \Sigma_{m=1}^M\omega_m + 2\Sigma_{m=1}^M\log(1+\exp(\omega_m))$; if TTFs follow small extreme value (SEV) distributions, then $\ell(\frac{\boldsymbol{z}-\mathbf{1}_M\beta_0-\hat{\boldsymbol{Y}}\boldsymbol{\beta}_1}{\sigma}) = n\log\sigma - \Sigma_{m=1}^M\omega_m + \Sigma_{m=1}^M\exp(\omega_m)$.

However, it is challenging to solve criterion (4.17) since it is not convex. Thus, to simplify the development of optimization algorithms for model parameter estimation, we transorm criterion (4.17) to a convex one where we apply the following re-parameterazation: $\tilde{\sigma} = 1/\sigma, \tilde{\beta}_0 = \beta_0/\sigma, \tilde{\boldsymbol{\beta}}_1 = \boldsymbol{\beta}_1/\sigma$:

$$\{\hat{\tilde{\beta}}_0, \hat{\tilde{\boldsymbol{\beta}}}_1, \hat{\tilde{\sigma}}\} = \arg\min_{\tilde{\beta}_0,\tilde{\boldsymbol{\beta}}_1,\tilde{\sigma}} \ell(\tilde{\sigma}\boldsymbol{z} - \mathbf{1}_M\tilde{\beta}_0 - \hat{\boldsymbol{Y}}\tilde{\boldsymbol{\beta}}_1). \tag{4.18}$$

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

After deriving the estimated parameters $\{\hat{\tilde{\beta}}_0, \hat{\tilde{\boldsymbol{\beta}}}_1, \hat{\tilde{\sigma}}\}$ from criterion (4.18), we can transform them back to the estimation of the parameters in the LLS regression model: $\hat{\gamma}_0 = \hat{\tilde{\gamma}}_0/\hat{\tilde{\sigma}}$,

$\hat{\gamma}_1 = \hat{\tilde{\gamma}}_1/\hat{\tilde{\sigma}}$ and $\hat{\sigma} = 1/\hat{\tilde{\sigma}}$. As a result, the fitted LLS regression model can be constructed as $\hat{z}_m \sim LLS(\hat{\beta}_0 + vec(\hat{\mathscr{Y}}_m)^\top \hat{\beta}_1, \hat{\sigma})$, where $\hat{\beta}_0 + vec(\hat{\mathscr{Y}}_m)^\top \hat{\beta}_1$ 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: $\hat{z}_t \sim LLS(\hat{\beta}_0 + vec(\hat{\mathscr{Y}}_t)^\top \hat{\beta}_1, \hat{\sigma})$.

Please note that there is data privacy leakage when we combine the low-dimensional feature tensor of all the users. Similar to section 20, the orignial high-dimensional tensor can be recovered based on the following equation:

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

.

where $\tilde{\mathscr{X}}^d_{m_d}$ is the m_dth high-dimensional tensor sample in the dth user and $\tilde{\mathscr{Y}}^d_{m_d}$ 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 (Su and Fang. (2023)).

4.4 Numerical Study

In this section, we evaluate the effectiveness of our proposed Federated Multilinear Principal Component Analysis (FMPCA) using simulated data generated from a heat transfer process.

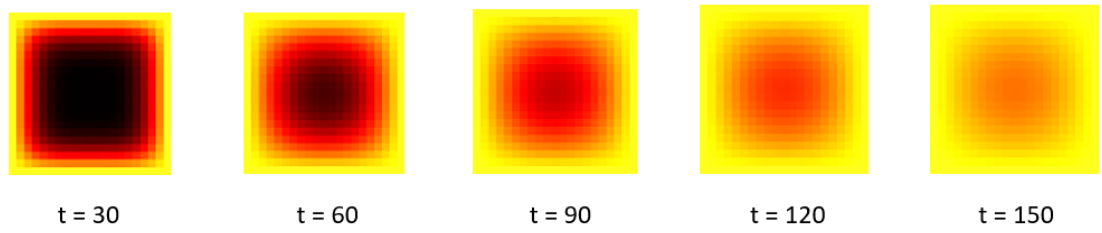
4.4.1 Data Generation

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

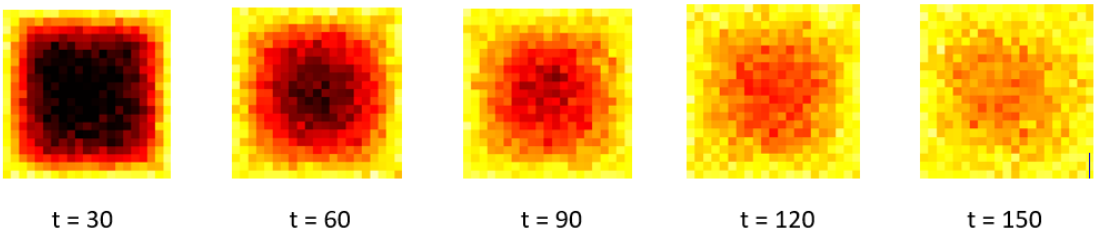
$$\frac{\partial \mathscr{X}_m(x, y, t)}{\partial t} = \alpha_m \Big(\frac{\partial^2 \mathscr{X}_m}{\partial x^2} + \frac{\partial^2 \mathscr{X}_m}{\partial y^2} \Big),$$

where $(x, y), 0 \le x, y \le 0.2$, 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 $\mathscr{U}(0.5 \times 10^{-4}, 1 \times 10^{-4})$. t is the time frame. The initial and boundary conditions are set such that $\mathscr{X}|_{t=1} = 0$ and $\mathscr{X}_m|_{x=0} = \mathscr{X}_m|_{x=0.2} = \mathscr{X}_m|_{y=0} = \mathscr{X}_m|_{y=0.2} = 30$. 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 4.8 shows the degradation images with and without noise observed at time $t = 30, 60, 90, 120, 150$.



(a) Without noise



(b) With noise

Figure 4.8: Simulated degradation images based on heat transfer process.

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:

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

$$z_m = e^{vec(\mathcal{Y}_m)\times\beta_1+\beta_0+\epsilon_m}, m = 1,\ldots,M.$$

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

4.4.2 The Benchmark and Performance Comparison

We divided the dataset into a training set and a test set, comprising 400 and 100 assets, respectively. Within the training set, we randomly assigned 250 assets to User 1, 100 assets to User 2, and the remaining 50 assets to User 3.

To evaluate the performance of our proposed method, we compared it against four 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 4.3. 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 $P_1 = 1,\ldots,5$, $P_2 = 1,\ldots,5$ and $P_3 = 1,\ldots,5$.

In our proposed method, referred to as “FMPCA", we employed a 10-fold cross-validation approach to determine the dimension for the tensor subspace $\{P_n, n = 1,2,3\}$. We utilized the LLS-based prognostics model from Section 4.3 to extract low-dimensional feature tensors, employing lognormal regression for building the prognostic model. Prediction errors for our method and the four benchmarks were calculated using the following equation:

$$\text{Prediction Error} = \frac{|\text{Estimated TTF} - \text{True TTF}|}{\text{True TTF}}. \tag{4.19}$$

To assess result robustness, we repeated the entire procedure, including dataset splitting, prediction error calculation, and cross-validation, 10 times. The resulting prediction errors were then combined for analysis.

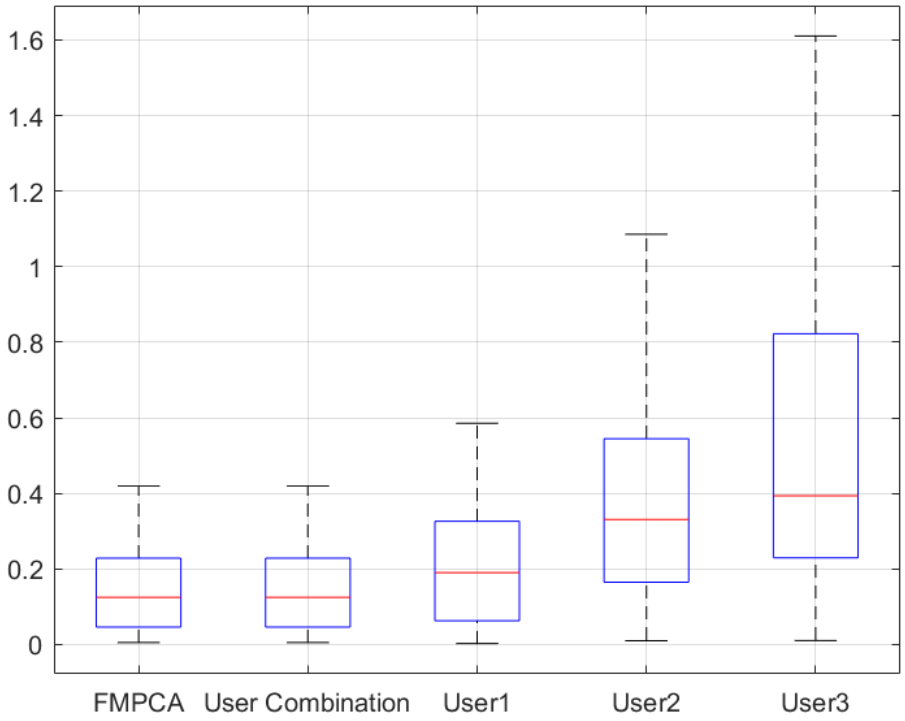


Figure 4.9: Numerical Study.

4.4.3 Results and analysis

Figure 4.9 presents the prediction errors of our proposed method, FMPCA, as well as the four benchmarks.

The figure demonstrates that the performance of our proposed method is equivalent to the 1st benchmark, “User Combination". Both FMPCA and "User Combination" exhibit median absolute prediction errors of 0.13 (with minimum, 1st quartile, 3rd quartile, and maximum values of 0, 0.03, 0.21, and 0.41, respectively). This similarity indicates that the projection matrices $\{\tilde{\boldsymbol{U}} \in \mathbb{R}^{I_n \times P_n}, n = 1,2,3\}$ derived by FMPCA are identical to those of MPCA (Lu et al. 2008). Consequently, FMPCA preserves performance accuracy while ensuring user privacy and data security.

Furthermore, Figure 4.9 illustrates that FMPCA outperforms the other three benchmarks (“User 1", “User 2" and “ User 3"). Among the three decentralized benchmarks, “User 1" demonstrates the best prediction accuracy, while “ User 3" exhibits the worst. For instance, the median absolute prediction errors (along with the Interquartile Ranges, or IQRs) for FMPCA, “User 1", “User 2" and “ User 3" are 0.13 (0.17), 0.19 (0.28), 0.37 (0.39), and 0.4

(0.59), respectively. This observation is reasonable as federated learning allows multiple users to collaboratively train a model, while data silos impose limitations on the size of the training data, thereby affecting performance.

4.5 Case Study

In this section, we assess the effectiveness of our proposed method using degradation image streams acquired from a specialized test bed that conducts accelerated degradation tests on rolling element thrust bearings. To capture the degradation images, we employ the FLIR T300 infrared camera. Concurrently, an accelerometer is installed on the test bed to monitor the vibration of the bearing. Failure time is defined as the point at which the amplitude of defective vibration frequencies surpasses a threshold determined by ISO standards for machine vibration. The degradation images have dimensions of 40×20 pixels. We generate a total of 284 degradation image streams along with their corresponding time-to-failure (TTF) data. Further information regarding the dataset can be found in the works of (Gebraeel et al. 2009) and (Fang et al. 2019).

Due to the truncation of failure time, different assets in our dataset may have varying numbers of images, leading to incomplete image streams. To address this, we employ a tensor completion method called TMac, developed by (Xu et al. 2013). Following the numerical study, we randomly split the 284 image streams into a training dataset comprising 227 assets and a test dataset with the remaining 57 assets. The ratio of training to testing data is approximately 80% and 20%, respectively. Within the training dataset, we assign 140 assets to User 1, 57 assets to User 2, and the remaining 30 assets to User 3. Similar to the numerical study, we utilize 10-fold cross-validation to determine the optimal combination of tensor subspace ranks (P_1, P_2, P_3) ranging from $1, \ldots, 5$. Additionally, we repeat the entire process 10 times and aggregate the prediction errors to assess result robustness. Figure 4.11 depicts the prediction errors of our proposed method and the four benchmarks for the case study.

Similar to the findings in the numerical study (Section 4.4), Figure 4.11 demonstrates that our proposed method, FMPCA, achieves comparable prediction accuracy to the first benchmark, “User Combination". The median absolute prediction errors (with minimum, 1st quartile, 3rd quartile, and maximum values) for both FMPCA and “User Combination" are 0.06 (with values of 0, 0.02, 0.1, and 0.22, respectively). This further verifies that FMPCA derives 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)

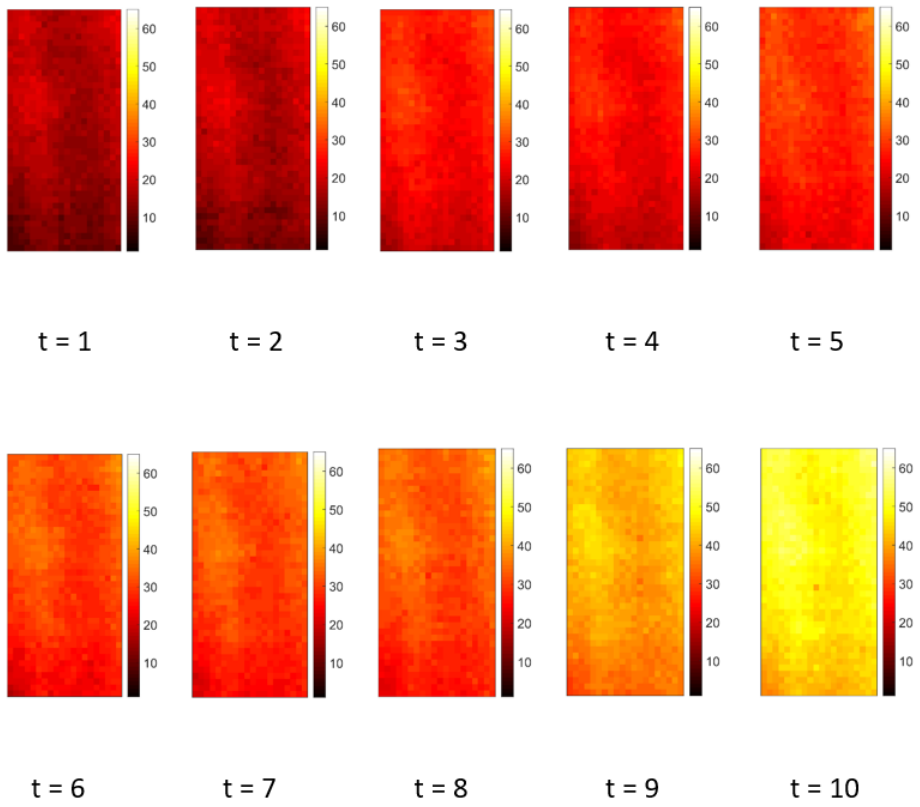


Figure 4.10: An illustration of one infrared degradation image stream.

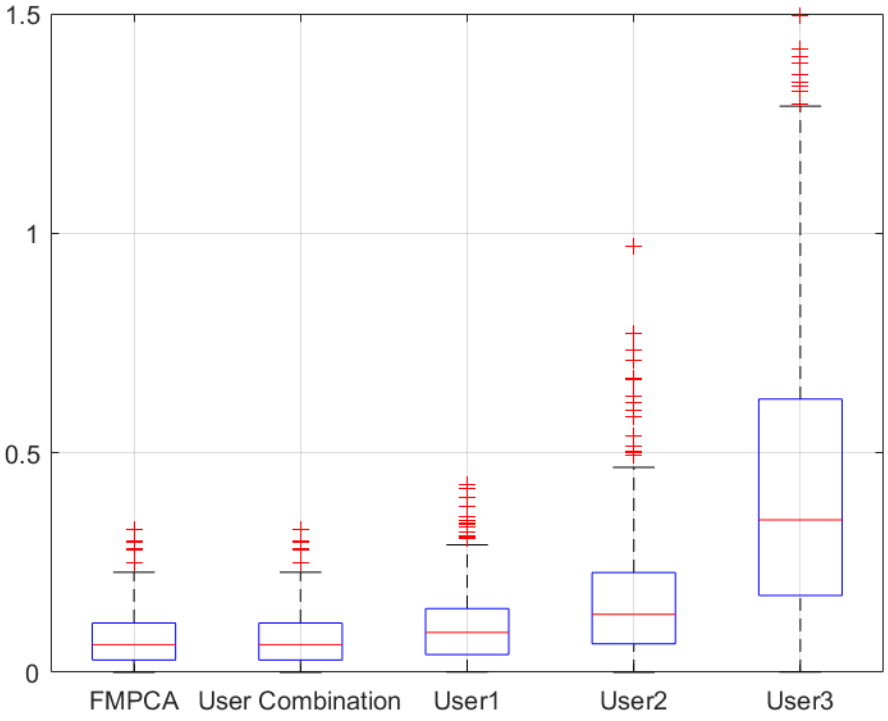


Figure 4.11: Case Study.

without compromising performance.

Furthermore, Figure 4.11 reveals that FMPCA outperforms the other three benchmarks ("User 1", "User 2" and "User 3"). Among the benchmarks, "User 1" exhibits the highest prediction accuracy, followed by "User 2", while "User 3" demonstrates the lowest accuracy. For instance, the median absolute prediction errors (with the Interquartile Range, or IQR) for "User Combination", "User 1", "User 2" and "User 3" are 0.06 (0.08), 0.1 (0.11), 0.15 (0.2), and 0.31 (0.47), respectively. These results affirm that federated learning encourages collaboration among users, leading to improved performance. Moreover, a larger dataset provides more valuable predictive information, resulting in enhanced prediction performance.

4.6 Conclusions

This chapter introduces a privacy-preserving federated learning-based approach called Federated Multilinear Principal Component Analysis (FMPCA) to address data leakage concerns in multilinear principal component analysis (MPCA) (Lu et al. 2008). FMPCA incorporates three algorithms to ensure data security at various stages of MPCA: secure cen-

tralization of input tensor samples, secure derivation of projection matrices in initialization, and secure derivation of projection matrices in local optimization. In the secure centralization of input tensor samples, each user calculates their sample mean and generates perturbations that are shared with other users, preventing the disclosure of sample means to the central server or other users and safeguarding data privacy. The other two algorithms, secure derivation of projection matrices in initialization and local optimization, involve constructing a new matrix by combining tensor samples from all users. We demonstrate that the left singular vectors obtained through singular value decomposition (SVD) of this matrix are equivalent to the projection matrices used in MPCA. An updating procedure is proposed to derive the left singular vectors while preserving the confidentiality of user data.

We conducted evaluations of our proposed FMPCA methodology using both simulated and real-world datasets from rotating machinery. The results confirm that FMPCA can derive the same projection matrices as MPCA, maintaining performance accuracy, while ensuring data privacy. Moreover, our findings highlight the advantages of federated learning, wherein multiple users collaborate to train a model, leading to superior performance compared to traditional data silos.

CHAPTER

5

CONCLUSIONS

This thesis introduces new methodologies that address the challenges of analyzing high-dimensional data for condition monitoring in modern manufacturing, service sectors, and the energy industry. The following sections provide a summary of the main research results and novel contributions of this dissertation.

(1) *A novel convex two-dimensional variable selection method that can inspire both group-wise and element-wise sparsity was proposed.* It can be used for the quality defect diagnostics of multistage manufacturing processes with multiple identical stages such as hot strip mills in the steel-making industry and 3D printing processes in additive manufacturing. The proposed model is a new generalized matrix regression model with regularization, which is formulated as a convex optimization criterion that consists of a negative log-likelihood function, two ℓ_2 penalization terms, and an ℓ_1 penalization term. By simultaneously using both ℓ_2 and ℓ_1 penalization terms, the proposed method can inspire both group-wise and element-wise sparsity. This implies that, unlike the existing models, the proposed method can provide two levels of information. First, it provides the overall importance of each stage and process variable, which helps identify a few crucial stages and process variables that need to be investigated from the stage-level or process variable-level when revising the control algorithm. Second, it suggests the importance of the process variables in each

crucial stage and the stage importance information of each crucial process variable, which provides element-level information that can be used to guide the control algorithm revision as well. A simulated dataset and a real-world dataset from a hot strip steel mill were also used to evaluate the performance of the proposed method. The results demonstrated that our proposed method outperformed the benchmark in terms of selection accuracy and precision. We believe this is because the proposed model can achieve both group-wise and element-wise selection, while the benchmark can only inspire group-wise sparsity.

(2) *A supervised dimension reduction-based prognostic model was proposed to use an asset's incomplete degradation images to predict its TTF*. This is achieved by first developing a new supervised tensor dimension reduction method that reduces the dimension of incomplete high-dimensional degradation image streams and provides low-dimensional tensor features, which are then used to build a prognostic model based on (log)-location-scale regression. The supervised tensor dimension reduction method uses historical TTFs to supervise the detection of a low-dimensional tensor subspace to reduce the dimension of incomplete high-dimensional image streams. Mathematically, it is formulated as an optimization criterion that combines a feature extraction term and a regression term. The feature extraction term focuses on identifying a tensor space to extract low-dimensional tensor features from high-dimensional image streams. The regression term regresses failure times against the features extracted by the first term using LLS regression. By jointly optimizing the two terms, it is expected to detect an appropriate tensor subspace such that the extracted features are effective for TTF prediction. To estimate the parameters of the supervised dimension reduction method, we developed a Block Updating Algorithm for applications where TTFs follow distributions in the (log)-location-scale family. The algorithm works by splitting the parameters into several blocks and cyclically optimizing one block of parameters while keeping other blocks fixed until convergence. In addition, we showed that if TTFs follow normal or lognormal distributions, there is a closed-form solution when optimizing each block of the parameters, no matter the imaging data is complete or incomplete. Simulated data as well as a data set from rotating machinery were used to validate the effectiveness of our proposed method. The results showed that our proposed prognostic method consistently outperforms the unsupervised tensor reduction-based benchmarks under various data missing rates. This validated the benefits and importance of using failure time information to supervise the dimension reduction of high-dimensional degradation image streams when building prognostic models.

(3) *A federated learning-based multilinear principal component analysis (FMPCA) was proposed to 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. A simulated dataset as well as a data set from rotating machinery 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.

REFERENCES

Abdi, H. and Williams, L. J. (2010). Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459.

Adland, R., Jia, H., Lode, T., and Skontorp, J. (2021). The value of meteorological data in marine risk assessment. *Reliability Engineering & System Safety*, 209:107480.

Alokita, S., Rahul, V., Jayakrishna, K., Kar, V., Rajesh, M., Thirumalini, S., and Manikandan, M. (2019). Recent advances and trends in structural health monitoring. *Structural health monitoring of biocomposites, fibre-reinforced composites and hybrid composites*, pages 53–73.

Aydemir, G. and Paynabar, K. (2019). Image-based prognostics using deep learning approach. *IEEE Transactions on Industrial Informatics*, 16(9):5956–5964.

Balmashnova, E., Bruurmijn, M., Dissanayake, R., Duits, R., Kampmeijer, L., and van Noorden, T. (2012). Image recognition of shape defects in hot steel rolling. *Proceedings of the 84th European Study Group Mathematics with Industry (SWI 2012)*, pages 22–38.

Bogdanoff, J. L. and Kozin, F. (1985). Probabilistic models of cumulative damage(book). *New York, Wiley-Interscience, 1985, 350 p.*

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

Boyd, S., Boyd, S. P., and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

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

Dong, Y., Xia, T., Wang, D., Fang, X., and Xi, L. (2021). Infrared image stream based regressors for contactless machine prognostics. *Mechanical Systems and Signal Processing*, 154:107592.

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

Fang, X., Paynabar, K., and Gebraeel, N. (2017). Multistream sensor fusion-based prognostics model for systems with single failure modes. *Reliability Engineering & System Safety*, 159:322–331.

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

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

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

Gebraeel, N. Z., Lawley, M. A., Li, R., and Ryan, J. K. (2005). Residual-life distributions from component degradation signals: A bayesian approach. *IiE Transactions*, 37(6):543–557.

Gibson, I., Rosen, D. W., Stucker, B., and Khorasani, M. (2021). *Additive manufacturing technologies*, volume 17. Springer.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). The elements of statistical learnin. *Cited on*, 33.

Hastie, T., Tibshirani, R., and Wainwright, M. (2019). *Statistical learning with sparsity: the lasso and generalizations*. Chapman and Hall/CRC.

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

Hong, Y. and Meeker, W. Q. (2010). Field-failure and warranty prediction based on auxiliary use-rate information. *Technometrics*, 52(2):148–159.

Hong, Y. and Meeker, W. Q. (2013). Field-failure predictions based on failure-time data with dynamic covariate information. *Technometrics*, 55(2):135–149.

Huang, L., Yin, Y., Fu, Z., Zhang, S., Deng, H., and Liu, D. (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.

Jardine, A. K., Lin, D., and Banjevic, D. (2006). A review on machinery diagnostics and prognostics implementing condition-based maintenance. *Mechanical systems and signal processing*, 20(7):1483–1510.

Jeong, C. and Fang, X. (2022). Two-dimensional variable selection and its applications in the diagnostics of product quality defects. *IISE Transactions*, 54(7):619–629.

Jiang, Y., Xia, T., Fang, X., Wang, D., Pan, E., and Xi, L. (2023). Sparse hierarchical parallel residual networks ensemble for infrared image stream-based remaining useful life prediction. *IEEE Transactions on Industrial Informatics*.

Jiang, Y., Xia, T., Wang, D., Fang, X., and Xi, L. (2022a). Adversarial regressive domain adaptation approach for infrared thermography-based unsupervised remaining useful life prediction. *IEEE Transactions on Industrial Informatics*, 18(10):7219–7229.

Jiang, Y., Xia, T., Wang, D., Fang, X., and Xi, L. (2022b). Spatiotemporal denoising wavelet network for infrared thermography-based machine prognostics integrating ensemble uncertainty. *Mechanical Systems and Signal Processing*, 173:109014.

Kolda, T. G. (2006). Multilinear operators for higher-order decompositions. Technical report, Citeseer.

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

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

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

Liu, J., Musialski, P., Wonka, P., and Ye, J. (2012). Tensor completion for estimating missing values in visual data. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):208–220.

Liu, K., Gebraeel, N. Z., and Shi, J. (2013). A data-level fusion model for developing composite health indices for degradation modeling and prognostic analysis. *IEEE Transactions on Automation Science and Engineering*, 10(3):652–664.

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

McCullagh, P. and Nelder, J. (1989). Generalized linear models ii.

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

Michau, G., Hu, Y., Palmé, T., and Fink, O. (2020). Feature learning for fault detection in high-dimensional condition monitoring signals. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 234(1):104–115.

Peng, K., Zhang, K., and Li, G. (2013). Quality-related process monitoring based on total kernel pls model and its industrial application. *Mathematical Problems in Engineering*, 2013.

Prautzsch, H., Boehm, W., and Paluszny, M. (2002). *Bézier and B-spline techniques*, volume 6. Springer.

Ramsay, J. and Silverman, B. (2005). Principal components analysis for functional data. *Functional data analysis*, pages 147–172.

Rifaai, T. M., Abokifa, A. A., and Sela, L. (2022). Integrated approach for pipe failure prediction and condition scoring in water infrastructure systems. *Reliability Engineering & System Safety*, 220:108271.

Schöbi, R. and Sudret, B. (2019). Global sensitivity analysis in the context of imprecise probabilities (p-boxes) using sparse polynomial chaos expansions. *Reliability Engineering & System Safety*, 187:129–141.

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

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

Shen, L., Tang, Y., and Tang, L. C. (2021). Understanding key factors affecting power systems resilience. *Reliability Engineering & System Safety*, 212:107621.

Shi, J. and Zhou, S. (2009). Quality control and improvement for multistage systems: A survey. *Iie Transactions*, 41(9):744–753.

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

Shu, Y., Feng, Q., and Coit, D. W. (2015). Life distribution analysis based on l évy subordinators for degradation with random jumps. *Naval Research Logistics (NRL)*, 62(6):483–492.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288.

Wang, F., Gahrooei, M. R., Zhong, Z., Tang, T., and Shi, J. (2021). An augmented regression model for tensors with missing values. *IEEE Transactions on Automation Science and Engineering*, 19(4):2968–2984.

Wang, F., Wang, A., Tang, T., and Shi, J. (2022). Sgl-pca: Health index construction with sensor sparsity and temporal monotonicity for mixed high-dimensional signals. *IEEE Transactions on Automation Science and Engineering*.

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

Xu, Y. and Yin, W. (2013). A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on imaging sciences*, 6(3):1758–1789.

Yang, J., Zhang, D., Frangi, A. F., and Yang, J.-y. (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., Liu, Y., Cheng, Y., Kang, Y., Chen, T., and Yu, H. (2019). Federated learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 13(3):1–207.

Yang, Z., Baraldi, P., and Zio, E. (2021). A multi-branch deep neural network model for failure prognostics based on multimodal data. *Journal of Manufacturing Systems*, 59:42–50.

Ye, J., Janardan, R., and Li, Q. (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*, pages 354–363.

Ye, Z.-S. and Chen, N. (2014). The inverse gaussian process as a degradation model. *Technometrics*, 56(3):302–311.

Ye, Z.-S., Xie, M., Tang, L.-C., and Chen, N. (2014). Semiparametric estimation of gamma processes for deteriorating products. *Technometrics*, 56(4):504–513.

Yuan, M. and Lin, Y. (2006). Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67.

Zhao, J. and Leng, C. (2014). Structured lasso for regression with matrix covariates. *Statistica Sinica*, pages 799–814.

Zhao, J., Niu, L., and Zhan, S. (2017). Trace regression model with simultaneously low rank and row (column) sparse parameter. *Computational Statistics & Data Analysis*, 116:1–18.

APPENDICES

APPENDIX

A

PROOF IN CHAPTER 3

A.1 Proof of Proposition 1

The original optimization problem is

$$\arg\min_{\boldsymbol{U}_1} \ \alpha||\mathscr{P}_\Omega(\mathscr{X}-\mathscr{S}\times_1\boldsymbol{U}_1^\top\times_2\boldsymbol{U}_2^\top\times_3\boldsymbol{U}_3^\top)||_F^2+(1-\alpha)||\boldsymbol{y}-\mathbf{1}_M\cdot\tilde{\boldsymbol{\beta}}_0-\boldsymbol{S}_{(4)}\cdot\tilde{\boldsymbol{\beta}}_1||_F^2,$$

which is equivalent to the following problem when data is complete:

$$\arg\min_{\boldsymbol{U}_1} \ \alpha||\mathscr{X}-\mathscr{S}\times_1\boldsymbol{U}_1^\top\times_2\boldsymbol{U}_2^\top\times_3\boldsymbol{U}_3^\top||_F^2+(1-\alpha)||\boldsymbol{y}-\mathbf{1}_M\cdot\tilde{\boldsymbol{\beta}}_0-\boldsymbol{S}_{(4)}\cdot\tilde{\boldsymbol{\beta}}_1||_F^2,$$

which is convex. Thus, it can be solved by setting the derivatives to be zeros–that is $\frac{d\Psi}{d\boldsymbol{U}_1}=0$, where $\Psi=\alpha||\mathscr{X}-\mathscr{S}\times_1\boldsymbol{U}_1^\top\times_2\boldsymbol{U}_2^\top\times_3\boldsymbol{U}_3^\top||_F^2+(1-\alpha)||\boldsymbol{y}-\mathbf{1}_M\cdot\tilde{\boldsymbol{\beta}}_0-\boldsymbol{S}_{(4)}\cdot\tilde{\boldsymbol{\beta}}_1||_F^2$. This implies $\frac{d}{d\boldsymbol{U}_1}(||\mathscr{X}-\mathscr{S}\times_1\boldsymbol{U}_1^\top\times_2\boldsymbol{U}_2^\top\times_3\boldsymbol{U}_3^\top||_F^2)=0$. According to the communication law of tensor mode multiplication, we have $\frac{d}{d\boldsymbol{U}_1}(||\mathscr{X}-(\mathscr{S}\times_2\boldsymbol{U}_2^\top\times_3\boldsymbol{U}_3^\top)\times_1\boldsymbol{U}_1^\top||_F^2)=0$. Thus, $\frac{d}{d\boldsymbol{U}_1}(||\mathscr{X}-\mathscr{S}_{U_1}\times_1\boldsymbol{U}_1^\top||_F^2)=0$, where $\mathscr{S}_{U_1}=\mathscr{S}\times_2\boldsymbol{U}_2^\top\times_3\boldsymbol{U}_3^\top$. Furthermore, we have $\frac{d}{d\boldsymbol{U}_1}(||\boldsymbol{X}_{(1)}-\boldsymbol{U}_1^\top\cdot\boldsymbol{S}_{U_1(1)}||_F^2)=0$ due to the fact that $||\mathscr{S}||_F^2=||\boldsymbol{S}_{(n)}||_F^2$ and the property of tensor mode multiplication $\mathscr{S}\times_n\boldsymbol{U}=\boldsymbol{U}\cdot\boldsymbol{S}_{(n)}$. By taking the derivative of the Frobenius norm, we have $2(\boldsymbol{X}_{(1)}-\boldsymbol{U}_1^\top\cdot$

$\boldsymbol{S}_{U_1(1)}) \cdot (-\boldsymbol{S}_{U_1(1)}^\top) = \boldsymbol{0}$. Thus, $\boldsymbol{U}_1^\top \cdot \boldsymbol{S}_{U_1(1)} \cdot \boldsymbol{S}_{U_1(1)}^\top = \boldsymbol{X}_{(1)} \cdot \boldsymbol{S}_{U_1(1)}^\top$, which gives that $\boldsymbol{U}_1^\top = \boldsymbol{X}_{(1)} \cdot \boldsymbol{S}_{U_1(1)}^\top \cdot (\boldsymbol{S}_{U_1(1)} \cdot \boldsymbol{S}_{U_1(1)}^\top)^{-1}$. Finally, we have $\boldsymbol{U}_1 = (\boldsymbol{X}_{(1)} \cdot \boldsymbol{S}_{U_1(1)}^\top \cdot (\boldsymbol{S}_{U_1(1)} \cdot \boldsymbol{S}_{U_1(1)}^\top)^{-1})^\top$.

A.2 Proof of Proposition 2

The original optimization problem is

$$\arg\min_{\boldsymbol{U}_2} \ \alpha ||\mathscr{P}_\Omega(\mathscr{X} - \mathscr{S} \times_1 \boldsymbol{U}_1^\top \times_2 \boldsymbol{U}_2^\top \times_3 \boldsymbol{U}_3^\top)||_F^2 + (1-\alpha)||\boldsymbol{y} - \boldsymbol{1}_M \cdot \tilde{\boldsymbol{\beta}}_0 - \boldsymbol{S}_{(4)} \cdot \tilde{\boldsymbol{\beta}}_1||_F^2,$$

which is equivalent to the following problem when data is complete:

$$\arg\min_{\boldsymbol{U}_2} \ \alpha ||\mathscr{X} - \mathscr{S} \times_1 \boldsymbol{U}_1^\top \times_2 \boldsymbol{U}_2^\top \times_3 \boldsymbol{U}_3^\top||_F^2 + (1-\alpha)||\boldsymbol{y} - \boldsymbol{1}_M \cdot \tilde{\boldsymbol{\beta}}_0 - \boldsymbol{S}_{(4)} \cdot \tilde{\boldsymbol{\beta}}_1||_F^2$$

which is convex. Therefore, it can be solved by setting the derivatives to be zeros–that is $\frac{d\Psi}{d\boldsymbol{U}_2} = \boldsymbol{0}$, where $\Psi = \alpha||\mathscr{X} - \mathscr{S} \times_1 \boldsymbol{U}_1^\top \times_2 \boldsymbol{U}_2^\top \times_3 \boldsymbol{U}_3^\top||_F^2 + (1-\alpha)||\boldsymbol{y} - \boldsymbol{1}_M \cdot \tilde{\boldsymbol{\beta}}_0 - \boldsymbol{S}_{(4)} \cdot \tilde{\boldsymbol{\beta}}_1||_F^2$. This implies $\frac{d}{d\boldsymbol{U}_2}(||\mathscr{X} - \mathscr{S} \times_1 \boldsymbol{U}_1^\top \times_2 \boldsymbol{U}_2^\top \times_3 \boldsymbol{U}_3^\top||_F^2) = \boldsymbol{0}$. According to the communication law of tensor mode multiplication, we have $\frac{d}{d\boldsymbol{U}_2}(||\mathscr{X} - (\mathscr{S} \times_1 \boldsymbol{U}_1^\top \times_3 \boldsymbol{U}_3^\top) \times_2 \boldsymbol{U}_2^\top||_F^2) = \boldsymbol{0}$. Thus, $\frac{d}{d\boldsymbol{U}_2}(||\mathscr{X} - \mathscr{S}_{U_2} \times_2 \boldsymbol{U}_2^\top||_F^2) = \boldsymbol{0}$, where $\mathscr{S}_{U_2} = \mathscr{S} \times_1 \boldsymbol{U}_1^\top \times_3 \boldsymbol{U}_3^\top$. Furthermore, we have $\frac{d}{d\boldsymbol{U}_2}(||\boldsymbol{X}_{(2)} - \boldsymbol{U}_2^\top \cdot \boldsymbol{S}_{U_2(2)}||_F^2) = 0$ due to the fact that $||\mathscr{S}||_F^2 = ||\boldsymbol{S}_{(n)}||_F^2$ and the property of tensor mode multiplicaion $\mathscr{S} \times_n \boldsymbol{U} = \boldsymbol{U} \cdot \boldsymbol{S}_{(n)}$. By taking the derivative of the Frobenius norm, we have $2(\boldsymbol{X}_{(2)} - \boldsymbol{U}_2^\top \cdot \boldsymbol{S}_{U_2(2)}) \cdot (-\boldsymbol{S}_{U_2(2)}^\top) = \boldsymbol{0}$. Thus, $\boldsymbol{U}_2^\top \cdot \boldsymbol{S}_{U_2(2)} \cdot \boldsymbol{S}_{U_2(2)}^\top = \boldsymbol{X}_{(2)} \cdot \boldsymbol{S}_{U_2(2)}^\top$ which gives that $\boldsymbol{U}_2^\top = \boldsymbol{X}_{(2)} \cdot \boldsymbol{S}_{U_2(2)}^\top \cdot (\boldsymbol{S}_{U_2(2)} \cdot \boldsymbol{S}_{U_2(2)}^\top)^{-1}$. Finally we have $\boldsymbol{U}_2 = (\boldsymbol{X}_{(2)} \cdot \boldsymbol{S}_{U_2(2)}^\top \cdot (\boldsymbol{S}_{U_2(2)} \cdot \boldsymbol{S}_{U_2(2)}^\top)^{-1})^\top$.

A.3 Proof of Proposition 3

The original optimization problem is

$$\arg\min_{\boldsymbol{U}_3} \ \alpha ||\mathscr{P}_\Omega(\mathscr{X} - \mathscr{S} \times_1 \boldsymbol{U}_1^\top \times_2 \boldsymbol{U}_2^\top \times_3 \boldsymbol{U}_3^\top)||_F^2 + (1-\alpha)||\boldsymbol{y} - \boldsymbol{1}_M \cdot \tilde{\boldsymbol{\beta}}_0 - \boldsymbol{S}_{(4)} \cdot \tilde{\boldsymbol{\beta}}_1||_F^2,$$

which is equivalent to the following problem when data is complete:

$$\arg\min_{\boldsymbol{U}_3} \ \alpha ||\mathscr{X} - \mathscr{S} \times_1 \boldsymbol{U}_1^\top \times_2 \boldsymbol{U}_2^\top \times_3 \boldsymbol{U}_3^\top||_F^2 + (1-\alpha)||\boldsymbol{y} - \boldsymbol{1}_M \cdot \tilde{\boldsymbol{\beta}}_0 - \boldsymbol{S}_{(4)} \cdot \tilde{\boldsymbol{\beta}}_1||_F^2$$

which is convex. Therefore, it can be solved by setting the derivatives to be zeros–that is $\frac{d\Psi}{d\boldsymbol{U}_3} = \boldsymbol{0}$, where $\Psi = \alpha||\mathscr{X} - \mathscr{S} \times_1 \boldsymbol{U}_1^\top \times_2 \boldsymbol{U}_2^\top \times_3 \boldsymbol{U}_3^\top||_F^2 + (1-\alpha)||\boldsymbol{y} - \boldsymbol{1}_M \cdot \tilde{\boldsymbol{\beta}}_0 - \boldsymbol{S}_{(4)} \cdot \tilde{\boldsymbol{\beta}}_1||_F^2$.

This implies $\frac{d}{d\boldsymbol{U}_3}(||\mathscr{X}-\mathscr{S}\times_1\boldsymbol{U}_1^\top\times_2\boldsymbol{U}_2^\top\times_3\boldsymbol{U}_3^\top||_F^2)=\boldsymbol{0}$. According to the communication law of tensor mode multiplication, we have $\frac{d}{d\boldsymbol{U}_3}(||\mathscr{X}-(\mathscr{S}\times_1\boldsymbol{U}_1^\top\times_2\boldsymbol{U}_2^\top)\times_3\boldsymbol{U}_3^\top||_F^2)=\boldsymbol{0}$. Thus, $\frac{d}{d\boldsymbol{U}_3}(||\mathscr{X}-\mathscr{S}_{U_3}\times_3\boldsymbol{U}_3^\top||_F^2)=\boldsymbol{0}$, where $\mathscr{S}_{U_3}=\mathscr{S}\times_1\boldsymbol{U}_1^\top\times_2\boldsymbol{U}_2^\top$. Furthermore, we have $\frac{d}{d\boldsymbol{U}_3}(||\boldsymbol{X}_{(3)}-\boldsymbol{U}_3^\top\cdot\boldsymbol{S}_{U_3(3)}||_F^2)=\boldsymbol{0}$ due to the fact that $||\mathscr{S}||_F^2=||\boldsymbol{S}_{(n)}||_F^2$ and the property of tensor mode multiplicaion $\mathscr{S}\times_n\boldsymbol{U}=\boldsymbol{U}\cdot\boldsymbol{S}_{(n)}$. By taking the derivative of the Frobenius norm, we have $2(\boldsymbol{X}_{(3)}-\boldsymbol{U}_3^\top\cdot\boldsymbol{S}_{U_3(3)})\cdot(-\boldsymbol{S}_{U_3(3)}^\top)=\boldsymbol{0}$. Thus, $\boldsymbol{U}_3^\top\cdot\boldsymbol{S}_{U_3(3)}\cdot\boldsymbol{S}_{U_3(3)}^\top=\boldsymbol{X}_{(3)}\cdot\boldsymbol{S}_{U_3(3)}^\top$, which gives that $\boldsymbol{U}_3^\top=\boldsymbol{X}_{(3)}\cdot\boldsymbol{S}_{U_3(3)}^\top\cdot(\boldsymbol{S}_{U_3(3)}\cdot\boldsymbol{S}_{U_3(3)}^\top)^{-1}$. Finally, we have $\boldsymbol{U}_3=(\boldsymbol{X}_{(3)}\cdot\boldsymbol{S}_{U_3(3)}^\top\cdot(\boldsymbol{S}_{U_3(3)}\cdot\boldsymbol{S}_{U_3(3)}^\top)^{-1})^\top$.

A.4 Proof of Proposition 4

The original optimization problem is

$$\arg\min_{S}\ \alpha||\mathscr{P}_\Omega(\mathscr{X}-\mathscr{S}\times_1\boldsymbol{U}_1^\top\times_2\boldsymbol{U}_2^\top\times_3\boldsymbol{U}_3^\top)||_F^2+(1-\alpha)||\boldsymbol{y}-\boldsymbol{1}_M\cdot\boldsymbol{\beta}_0-\boldsymbol{S}_{(4)}\cdot\boldsymbol{\beta}_1||_F^2,$$

which is equivalent to the following problem when data is complete:

$$\arg\min_{\hat{S}}\ \alpha||\mathscr{X}-\mathscr{S}\times_1\boldsymbol{U}_1^\top\times_2\boldsymbol{U}_2^\top\times_3\boldsymbol{U}_3^\top||_F^2+(1-\alpha)||\boldsymbol{y}-\boldsymbol{1}_M\cdot\boldsymbol{\beta}_0-\boldsymbol{S}_{(4)}\cdot\boldsymbol{\beta}_1||_F^2,$$

which is convex. Thus, it can be solved by setting the derivatives to be zeros–that is $\frac{d\Psi}{d\boldsymbol{S}}=\boldsymbol{0}$, where $\Psi=\alpha||\mathscr{X}-\mathscr{S}\times_1\boldsymbol{U}_1^\top\times_2\boldsymbol{U}_2^\top\times_3\boldsymbol{U}_3^\top||_F^2+(1-\alpha)||\boldsymbol{y}-\boldsymbol{1}_M\cdot\boldsymbol{\beta}_0-\boldsymbol{S}_{(4)}\cdot\boldsymbol{\beta}_1||_F^2$. According to the connection between Kronecker product and tensor mode multiplication (Kolda 2006), we have $\frac{d}{d\boldsymbol{S}}(\alpha||\boldsymbol{X}_{(4)}-\boldsymbol{S}_{(4)}\cdot(\boldsymbol{U}_3\otimes\boldsymbol{U}_2\otimes\boldsymbol{U}_1)||_F^2+(1-\alpha)||\boldsymbol{y}-\boldsymbol{1}_M\cdot\boldsymbol{\beta}_0-\boldsymbol{S}_{(4)}\cdot\boldsymbol{\beta}_1||_F^2)=\boldsymbol{0}$. By taking the derivative of the Frobenius norm, we have $2\alpha\cdot[\boldsymbol{X}_{(4)}-\boldsymbol{S}_{(4)}\cdot(\boldsymbol{U}_3\otimes\boldsymbol{U}_2\otimes\boldsymbol{U}_1)]\cdot[-(\boldsymbol{U}_3\otimes\boldsymbol{U}_2\otimes\boldsymbol{U}_1)^\top]+2(1-\alpha)\cdot[\boldsymbol{y}-\boldsymbol{1}_M\cdot\boldsymbol{\beta}_0-\boldsymbol{S}_{(4)}\cdot\boldsymbol{\beta}_1]\cdot(-\boldsymbol{\beta}_1^\top)=\boldsymbol{0}$. Thus, $-2\alpha\cdot\boldsymbol{X}_{(4)}\cdot(\boldsymbol{U}_3\otimes\boldsymbol{U}_2\otimes\boldsymbol{U}_1)^\top+2\alpha\cdot\boldsymbol{S}_{(4)}\cdot(\boldsymbol{U}_3\otimes\boldsymbol{U}_2\otimes\boldsymbol{U}_1)\cdot(\boldsymbol{U}_3\otimes\boldsymbol{U}_2\otimes\boldsymbol{U}_1)^\top+2(1-\alpha)(\boldsymbol{y}-\boldsymbol{1}_M\cdot\boldsymbol{\beta}_0)\cdot(-\boldsymbol{\beta}_1^\top)+2(1-\alpha)\cdot(\boldsymbol{S}_{(4)}\cdot\boldsymbol{\beta}_1\cdot\boldsymbol{\beta}_1^\top)=\boldsymbol{0}$. Finally, we have $\boldsymbol{S}_{(4)}=\big[\alpha\cdot\boldsymbol{X}_{(4)}\cdot(\boldsymbol{U}_3\otimes\boldsymbol{U}_2\otimes\boldsymbol{U}_1)^\top+(1-\alpha)\cdot(\boldsymbol{y}-\boldsymbol{1}_M\cdot\boldsymbol{\beta}_0)\cdot\boldsymbol{\beta}_1^\top\big]\cdot\big[\alpha\cdot(\boldsymbol{U}_3\otimes\boldsymbol{U}_2\otimes\boldsymbol{U}_1)\cdot(\boldsymbol{U}_3\otimes\boldsymbol{U}_2\otimes\boldsymbol{U}_1)^\top+(1-\alpha)\cdot\boldsymbol{\beta}_1\cdot\boldsymbol{\beta}_1^\top\big]^{-1}$.

A.5 Proof of Lemma 1

Let $\boldsymbol{a}_m\in\mathbb{R}^{1\times N}$ denotes the mth row of matrix $\boldsymbol{A}\in\mathbb{R}^{M\times N}$ and $\boldsymbol{b}_m\in\mathbb{R}^{1\times P}$ denotes the mth row of matrix $\boldsymbol{B}\in\mathbb{R}^{M\times P}$, $m=1,\ldots,M$, then we have

$$\boldsymbol{A}-\boldsymbol{B}\boldsymbol{C}=[(\boldsymbol{a}_1-\boldsymbol{b}_1\boldsymbol{C})^\top,\ldots,(\boldsymbol{a}_M-\boldsymbol{b}_M\boldsymbol{C})^\top]^\top.$$

Based on the definition of Frobenius norm, the original objective function in Lemma 1 can be transformed as follows:

$$\|\boldsymbol{A}-\boldsymbol{B}\boldsymbol{C}\|_F^2 = \|[(\boldsymbol{a}_1-\boldsymbol{b}_1\boldsymbol{C})^\top,\ldots,(\boldsymbol{a}_M-\boldsymbol{b}_M\boldsymbol{C})^\top]^\top\|_F^2 = \sum_{m=1}^{M}\|\boldsymbol{a}_m-\boldsymbol{b}_m\boldsymbol{C}\|_F^2.$$

Therefore, we have the following:

$$\arg\min_{\boldsymbol{B}}\|\boldsymbol{A}-\boldsymbol{B}\boldsymbol{C}\|_F^2 = \arg\min_{\{\boldsymbol{b}_m\}_{m=1}^M}\sum_{m=1}^{M}\|\boldsymbol{a}_m-\boldsymbol{b}_m\boldsymbol{C}\|_F^2.$$

where $\boldsymbol{B}=[\boldsymbol{b}_1^\top,\ldots,\boldsymbol{b}_M^\top]^\top$. Therefore, to solve the original objective function, we can simply solve the following M sub problems:

$$\arg\min_{\boldsymbol{b}_m}\|\boldsymbol{a}_m-\boldsymbol{b}_m\boldsymbol{C}\|_F^2, \quad m=1,\ldots,M.$$

A.6 Proof of Proposition 5

The original optimization problem is

$$\arg\min_{\boldsymbol{U}_1}\ \alpha\|\mathcal{P}_\Omega(\mathcal{X}-\mathcal{S}\times_1\boldsymbol{U}_1^\top\times_2\boldsymbol{U}_2^\top\times_3\boldsymbol{U}_3^\top)\|_F^2+(1-\alpha)\|\boldsymbol{y}-\boldsymbol{1}_M\cdot\tilde{\boldsymbol{\beta}}_0-\boldsymbol{S}_{(4)}\cdot\tilde{\boldsymbol{\beta}}_1\|_F^2,$$

which is equivalent to the following problem when data is missing:

$$\arg\min_{\boldsymbol{U}_1}\ \alpha\|\mathcal{X}-(\mathcal{S}\times_1\boldsymbol{U}_1^\top\times_2\boldsymbol{U}_2^\top\times_3\boldsymbol{U}_3^\top)\odot\text{logic}(\mathcal{X})\|_F^2+(1-\alpha)\|\boldsymbol{y}-\boldsymbol{1}_M\cdot\tilde{\boldsymbol{\beta}}_0-\boldsymbol{S}_{(4)}\cdot\tilde{\boldsymbol{\beta}}_1\|_F^2,$$

where $\odot$ is the inner product, logic($\mathcal{X}$) denotes the logical value of $\mathcal{X}$–that is–if an entry is observed, its logical value is 1; otherwise, it is 0. Since the problem is convex, it can be solved by setting the derivatives to be zeros, i.e., $\frac{d\Psi}{d\boldsymbol{U}_1}=\boldsymbol{0}$, where $\Psi=\alpha\|\mathcal{X}-(\mathcal{S}\times_1\boldsymbol{U}_1^\top\times_2\boldsymbol{U}_2^\top\times_3\boldsymbol{U}_3^\top)\odot\text{logic}(\mathcal{X})\|_F^2+(1-\alpha)\|\boldsymbol{y}-\boldsymbol{1}_M\cdot\tilde{\boldsymbol{\beta}}_0-\boldsymbol{S}_{(4)}\cdot\tilde{\boldsymbol{\beta}}_1\|_F^2$. This implies $\frac{d}{d\boldsymbol{U}_1}(\|\mathcal{X}-(\mathcal{S}\times_1\boldsymbol{U}_1^\top\times_2\boldsymbol{U}_2^\top\times_3\boldsymbol{U}_3^\top)\odot\text{logic}(\mathcal{X})\|_F^2)=\boldsymbol{0}$. According to the communication law of tensor mode multiplication, we have $\frac{d}{d\boldsymbol{U}_1}(\|\mathcal{X}-[(\mathcal{S}\times_1\boldsymbol{U}_1^\top\times_2\boldsymbol{U}_2^\top)\times_3\boldsymbol{U}_3^\top]\odot\text{logic}(\mathcal{X})\|_F^2)=\boldsymbol{0}$. Thus, $\frac{d}{d\boldsymbol{U}_1}(\|\mathcal{X}-(\mathcal{S}_{U_1}\times_1\boldsymbol{U}_1^\top)\odot\text{logic}(\mathcal{X})\|_F^2)=\boldsymbol{0}$, where $\mathcal{S}_{U_1}=\mathcal{S}\times_2\boldsymbol{U}_2^\top\times_3\boldsymbol{U}_3^\top$. Furthermore, we have $\frac{d}{d\boldsymbol{U}_1}(\|\boldsymbol{X}_{(1)}-(\boldsymbol{U}_1^\top\cdot\boldsymbol{S}_{U_1(1)})\odot\text{logic}(\boldsymbol{X}_{(1)})\|_F^2)=\boldsymbol{0}$ since $\|\mathcal{S}\|_F^2=\|\boldsymbol{S}_{(n)}\|_F^2$ and $\mathcal{S}\times_n\boldsymbol{U}=\boldsymbol{U}\cdot\boldsymbol{S}_{(n)}$.

Figure A.1 shows the pattern of mode-1 matricization of the 4D tensor $\mathcal{X}$ when it has missing entries whose indices can be denoted by a set $\Omega\subseteq\{(i_1,i_2,i_3,m),1\le i_1\le I_1,1\le i_2\le I_2,1\le i_3\le I_3,1\le m\le M\}$. Based on Lemma 1, we can sequentially optimize each column

of U_1.

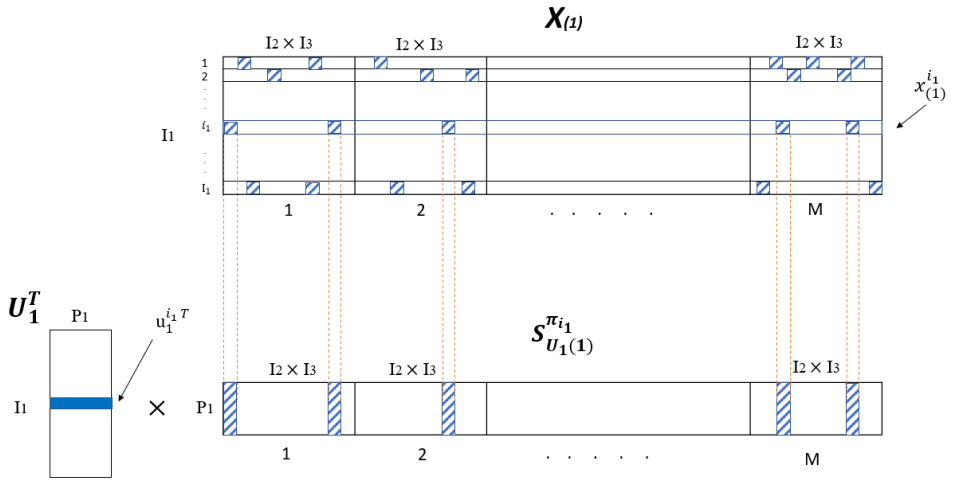


Figure A.1: An illustration of the data missing pattern in Proposition 5 (stripes representing available entries).

Specifically, we denote the i_1th row in $\boldsymbol{U}_1^\top$ as $\boldsymbol{u}_1^{i_1\top}$ (blue solid row of $\boldsymbol{U}_1^\top$ in Figure A.1). The available entries in the i_1th row of $\boldsymbol{X}_{(1)}$ are denoted as $\boldsymbol{x}_{(1)}^{i_1,\pi_{i_1}}$ (blue striped squares of $\boldsymbol{x}_{(1)}^{i_1}$ in Figure A.1). In $\boldsymbol{S}_{U_1(1)}^{\pi_{i_1}}$, we choose the columns whose indices are the same as those of the available entries of $\boldsymbol{x}_{(1)}^{i_1}$ (blue striped columns of $\boldsymbol{S}_{U_1(1)}^{\pi_{i_1}}$ in Figure A.1). As a result, we have $\frac{d}{d\boldsymbol{u}_1^{i_1}}(\Sigma_{i_1=1}^{I_1}||\boldsymbol{x}_{(1)}^{i_1,\pi_{i_1}}-(\boldsymbol{u}_1^{i_1\top}\cdot\boldsymbol{S}_{U_1(1)}^{\pi_{i_1}})||_F^2)=\boldsymbol{0}$. Because we only take the derivative of $\boldsymbol{u}_1^{i_1}$, we have $\frac{d}{d\boldsymbol{u}_1^{i_1}}(||\boldsymbol{x}_{(1)}^{i_1,\pi_{i_1}}-(\boldsymbol{u}_1^{i_1\top}\cdot\boldsymbol{S}_{U_1(1)}^{\pi_{i_1}})||_F^2)=\boldsymbol{0}$. By taking the derivative of the Frobenius norm, we have $2(\boldsymbol{x}_{(1)}^{i_1,\pi_{i_1}}-\boldsymbol{u}_1^{i_1\top}\cdot\boldsymbol{S}_{U_1(1)}^{\pi_{i_1}})\cdot(-\boldsymbol{S}_{U_1(1)}^{\pi_{i_1}\top})=\boldsymbol{0}$. Thus, $\boldsymbol{u}_1^{i_1\top}\cdot\boldsymbol{S}_{U_1(1)}^{\pi_{i_1}}\cdot\boldsymbol{S}_{U_1(1)}^{\pi_{i_1}\top}=\boldsymbol{x}_{(1)}^{i_1,\pi_{i_1}}\cdot\boldsymbol{S}_{U_1(1)}^{\pi_{i_1}\top}$, which gives that $\boldsymbol{u}_1^{i_1}=(\boldsymbol{x}_{(1)}^{i_1,\pi_{i_1}}\cdot\boldsymbol{S}_{U_1(1)}^{\pi_{i_1}\top}\cdot(\boldsymbol{S}_{U_1(1)}^{\pi_{i_1}}\cdot\boldsymbol{S}_{U_1(1)}^{\pi_{i_1}\top})^{-1})^\top$.

A.7 Proof of Proposition 6

The original optimization problem is

$$\arg\min_{\boldsymbol{U}_1}\ \alpha||\mathscr{P}_\Omega(\mathscr{X}-\mathscr{S}\times_1\boldsymbol{U}_1^\top\times_2\boldsymbol{U}_2^\top\times_3\boldsymbol{U}_3^\top)||_F^2+(1-\alpha)||\boldsymbol{y}-\boldsymbol{1}_M\cdot\tilde{\boldsymbol{\beta}}_0-\boldsymbol{S}_{(4)}\cdot\tilde{\boldsymbol{\beta}}_1||_F^2,$$

which is equivalent to the following problem when data is incomplete:

$$\arg\min_{\boldsymbol{U}_1} \ \alpha||\mathscr{X}-(\mathscr{S}\times_1\boldsymbol{U}_1^\top\times_2\boldsymbol{U}_2^\top\times_3\boldsymbol{U}_3^\top)\odot\text{logic}(\mathscr{X})||_F^2+(1-\alpha)||\boldsymbol{y}-\boldsymbol{1}_M\cdot\tilde{\boldsymbol{\beta}}_0-\boldsymbol{S}_{(4)}\cdot\tilde{\boldsymbol{\beta}}_1||_F^2,$$

where $\odot$ is the inner product, logic($\mathscr{X}$) denotes the logical value of $\mathscr{X}$. Since the problem is convex, it can be solved by setting the derivatives to be zeros, i.e., $\frac{d\Psi}{d\boldsymbol{U}_1}=\boldsymbol{0}$, where $\Psi=\alpha||\mathscr{X}-(\mathscr{S}\times_1\boldsymbol{U}_1^\top\times_2\boldsymbol{U}_2^\top\times_3\boldsymbol{U}_3^\top)\odot\text{logic}(\mathscr{X})||_F^2+(1-\alpha)||\boldsymbol{y}-\boldsymbol{1}_M\cdot\tilde{\boldsymbol{\beta}}_0-\boldsymbol{S}_{(4)}\cdot\tilde{\boldsymbol{\beta}}_1||_F^2$. This implies $\frac{d}{d\boldsymbol{U}_1}(||\mathscr{X}-(\mathscr{S}\times_1\boldsymbol{U}_1^\top\times_2\boldsymbol{U}_2^\top\times_3\boldsymbol{U}_3^\top)\odot\text{logic}(\mathscr{X})||_F^2)=\boldsymbol{0}$. According to the communication law of tensor mode multiplication, we have $\frac{d}{d\boldsymbol{U}_1}(||\mathscr{X}-[(\mathscr{S}\times_2\boldsymbol{U}_2^\top\times_3\boldsymbol{U}_3^\top)\times_1\boldsymbol{U}_1^\top]\odot\text{logic}(\mathscr{X})||_F^2)=\boldsymbol{0}$. Thus, $\frac{d}{d\boldsymbol{U}_1}(||\mathscr{X}-(\mathscr{S}_{U_1}\times_1\boldsymbol{U}_1^\top)\odot\text{logic}(\mathscr{X})||_F^2)=\boldsymbol{0}$, where $\mathscr{S}_{U_1}=\mathscr{S}\times_2\boldsymbol{U}_2^\top\times_3\boldsymbol{U}_3^\top$. Furthermore, we have $\frac{d}{d\boldsymbol{U}_1}(||\boldsymbol{X}_{(1)}-(\boldsymbol{U}_1^\top\cdot\boldsymbol{S}_{U_1(1)})\odot\text{logic}(\boldsymbol{X}_{(1)})||_F^2)=\boldsymbol{0}$ due to the fact that $||\mathscr{S}||_F^2=||\boldsymbol{S}_{(n)}||_F^2$ and $\mathscr{S}\times_n\boldsymbol{U}=\boldsymbol{U}\cdot\boldsymbol{S}_{(n)}$ (a property of tensor mode multiplication).

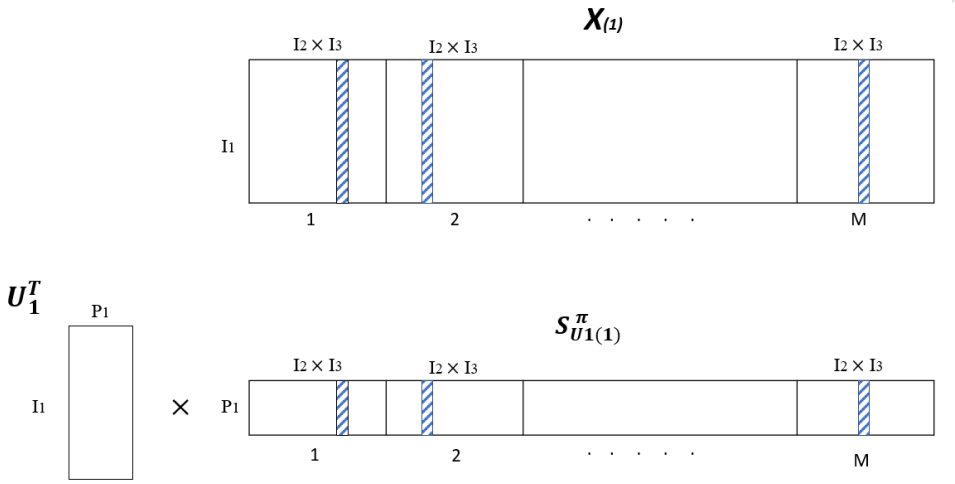


Figure A.2: An illustration of the data missing pattern in Proposition 6 (stripes representing available columns).

As discussed earlier, for applications with missing images, the indices of tensor $\mathscr{X}$'s missing entries can be denoted as $\Omega\subseteq\{(:,:,i_3,m),1\le i_3\le I_3,1\le m\le M\}$, where ":" denotes all the indices in a dimension. As a result, it can be easily shown that $\mathscr{X}$'s mode-1 matricization $\boldsymbol{X}_{(1)}$ has missing columns (see Figure A.2 for an illustration). Let π be the set consisting of the indices of available columns in $\boldsymbol{X}_{(1)}$, then we need to solve $\frac{d}{d\boldsymbol{U}_1}(||\boldsymbol{X}_{(1)}^\pi-\boldsymbol{U}_1^\top\cdot\boldsymbol{S}_{U_1(1)}^\pi||_F^2)=\boldsymbol{0}$, where $\boldsymbol{S}_{U_1(1)}^\pi$ denotes a matrix constituting the π columns of $\boldsymbol{S}_{U_1(1)}$. Thus, we have $2(\boldsymbol{X}_{(1)}^\pi-$

$\boldsymbol{U}_1^\top \cdot \boldsymbol{S}^\pi_{U_1(1)}) \cdot (-\boldsymbol{S}^\pi_{U_1(1)}{}^\top) = \boldsymbol{0}$. Thus, $\boldsymbol{U}_1^\top \cdot \boldsymbol{S}^\pi_{U_1(1)} \cdot \boldsymbol{S}^\pi_{U_1(1)}{}^\top = \boldsymbol{X}^\pi_{(1)} \cdot \boldsymbol{S}^\pi_{U_1(1)}{}^\top$, which gives that $\boldsymbol{U}_1^\top = \boldsymbol{X}^\pi_{(1)} \cdot \boldsymbol{S}^\pi_{U_1(1)}{}^\top \cdot (\boldsymbol{S}^\pi_{U_1(1)} \cdot \boldsymbol{S}^\pi_{U_1(1)}{}^\top)^{-1}$. This yields the solution $\boldsymbol{U}_1 = (\boldsymbol{X}^\pi_{(1)} \cdot \boldsymbol{S}^\pi_{U_1(1)}{}^\top \cdot (\boldsymbol{S}^\pi_{U_1(1)} \cdot \boldsymbol{S}^\pi_{U_1(1)}{}^\top)^{-1})^\top$.

A.8 Proof of Proposition 7

The original optimization problem is

$$\arg\min_{\boldsymbol{U}_2} \ \alpha||\mathscr{P}_\Omega(\mathscr{X} - \mathscr{S} \times_1 \boldsymbol{U}_1^\top \times_2 \boldsymbol{U}_2^\top \times_3 \boldsymbol{U}_3^\top)||_F^2 + (1-\alpha)||\boldsymbol{y} - \boldsymbol{1}_M \cdot \tilde{\boldsymbol{\beta}}_0 - \boldsymbol{S}_{(4)} \cdot \tilde{\boldsymbol{\beta}}_1||_F^2,$$

which is equivalent to the following problem when data is missing:

$$\arg\min_{\boldsymbol{U}_2} \ \alpha||\mathscr{X} - (\mathscr{S} \times_1 \boldsymbol{U}_1^\top \times_2 \boldsymbol{U}_2^\top \times_3 \boldsymbol{U}_3^\top) \odot \text{logic}(\mathscr{X})||_F^2 + (1-\alpha)||\boldsymbol{y} - \boldsymbol{1}_M \cdot \tilde{\boldsymbol{\beta}}_0 - \boldsymbol{S}_{(4)} \cdot \tilde{\boldsymbol{\beta}}_1||_F^2,$$

where $\odot$ is the inner product, logic($\mathscr{X}$) denotes the logical value of $\mathscr{X}$. Since the problem is convex, it can be solved by setting the derivatives to be zeros, i.e., $\frac{d\Psi}{d\boldsymbol{U}_2} = \boldsymbol{0}$, where $\Psi = \alpha||\mathscr{X} - (\mathscr{S} \times_1 \boldsymbol{U}_1^\top \times_2 \boldsymbol{U}_2^\top \times_3 \boldsymbol{U}_3^\top) \odot \text{logic}(\mathscr{X})||_F^2 + (1-\alpha)||\boldsymbol{y} - \boldsymbol{1}_M \cdot \tilde{\boldsymbol{\beta}}_0 - \boldsymbol{S}_{(4)} \cdot \tilde{\boldsymbol{\beta}}_1||_F^2$. This implies $\frac{d}{d\boldsymbol{U}_2}(||\mathscr{X} - (\mathscr{S} \times_1 \boldsymbol{U}_1^\top \times_2 \boldsymbol{U}_2^\top \times_3 \boldsymbol{U}_3^\top) \odot \text{logic}(\mathscr{X})||_F^2) = \boldsymbol{0}$. According to the communication law of tensor mode multiplication, we have $\frac{d}{d\boldsymbol{U}_2}(||\mathscr{X} - [(\mathscr{S} \times_1 \boldsymbol{U}_1^\top \times_3 \boldsymbol{U}_3^\top) \times_2 \boldsymbol{U}_2^\top] \odot \text{logic}(\mathscr{X})||_F^2) = \boldsymbol{0}$. Thus, $\frac{d}{d\boldsymbol{U}_2}(||\mathscr{X} - (\mathscr{S}_{U_2} \times_2 \boldsymbol{U}_2^\top) \odot \text{logic}(\mathscr{X})||_F^2) = \boldsymbol{0}$, where $\mathscr{S}_{U_2} = \mathscr{S} \times_1 \boldsymbol{U}_1^\top \times_3 \boldsymbol{U}_3^\top$. Furthermore, we have $\frac{d}{d\boldsymbol{U}_2}(||\boldsymbol{X}_{(2)} - (\boldsymbol{U}_2^\top \cdot \boldsymbol{S}_{U_2(2)}) \odot \text{logic}(\boldsymbol{X}_{(2)})||_F^2) = \boldsymbol{0}$ since $||\mathscr{S}||_F^2 = ||\boldsymbol{S}_{(n)}||_F^2$ and $\mathscr{S} \times_n \boldsymbol{U} = \boldsymbol{U} \cdot \boldsymbol{S}_{(n)}$.

Figure A.3 shows the pattern of mode-2 matricization of the 4D tensor $\mathscr{X}$ when it has missing entries whose indices can be denoted by a set $\Omega \subseteq \{(i_1, i_2, i_3, m), 1 \le i_1 \le I_1, 1 \le i_2 \le I_2, 1 \le i_3 \le I_3, 1 \le m \le M\}$. Based on Lemma 1, we can sequentially optimize each column of $\boldsymbol{U}_2$.

Specifically, we denote the i_2th row in $\boldsymbol{U}_2^\top$ as $\boldsymbol{u}_2^{i_2\top}$ (blue solid row of $\boldsymbol{U}_2^\top$ in Figure A.3). The available entries in the i_2th row of $\boldsymbol{X}_{(2)}$ are denoted as $\boldsymbol{x}_{(2)}^{i_2,\pi_{i_2}}$ (blue striped squares of $\boldsymbol{x}_{(2)}^{i_2}$ in Figure A.3). In $\boldsymbol{S}^{\pi_{i_2}}_{U_2(2)}$, we choose the columns whose indices are the same as those of the available entries in $\boldsymbol{x}_{(2)}^{i_2}$ (blue striped columns of $\boldsymbol{S}^{\pi_{i_2}}_{U_2(2)}$ in Figure A.3). As a result, we have $\frac{d}{d\boldsymbol{u}_2^{i_2}}(\sum_{i_2=1}^{I_2} ||\boldsymbol{x}_{(2)}^{i_2,\pi_{i_2}} - (\boldsymbol{u}_2^{i_2\top} \cdot \boldsymbol{S}^{\pi_{i_2}}_{U_2(2)})||_F^2) = \boldsymbol{0}$. Since on the derivative of $\boldsymbol{u}_2^{i_2}$ is taken, we have $\frac{d}{d\boldsymbol{u}_2^{i_2}}(||\boldsymbol{x}_{(2)}^{i_2,\pi_{i_2}} - (\boldsymbol{u}_2^{i_2\top} \cdot \boldsymbol{S}^{\pi_{i_2}}_{U_2(2)})||_F^2) = \boldsymbol{0}$. By taking the derivative of the Frobenius norm, we have $2(\boldsymbol{x}_{(2)}^{i_2,\pi_{i_2}} - \boldsymbol{u}_2^{i_2\top} \cdot \boldsymbol{S}^{\pi_{i_2}}_{U_2(2)}) \cdot (-\boldsymbol{S}^{\pi_{i_2}}_{U_2(2)}{}^\top) = \boldsymbol{0}$. Thus, $\boldsymbol{u}_2^{i_2\top} \cdot \boldsymbol{S}^{\pi_{i_2}}_{U_2(2)} \cdot \boldsymbol{S}^{\pi_{i_2}}_{U_2(2)}{}^\top = \boldsymbol{x}_{(2)}^{i_2,\pi_{i_2}} \cdot \boldsymbol{S}^{\pi_{i_2}}_{U_2(2)}{}^\top$, which gives that $\boldsymbol{u}_2^{i_2} = (\boldsymbol{x}_{(2)}^{i_2,\pi_{i_2}} \cdot \boldsymbol{S}^{\pi_{i_2}}_{U_2(2)}{}^\top \cdot (\boldsymbol{S}^{\pi_{i_2}}_{U_2(2)} \cdot \boldsymbol{S}^{\pi_{i_2}}_{U_2(2)}{}^\top)^{-1})^\top$.

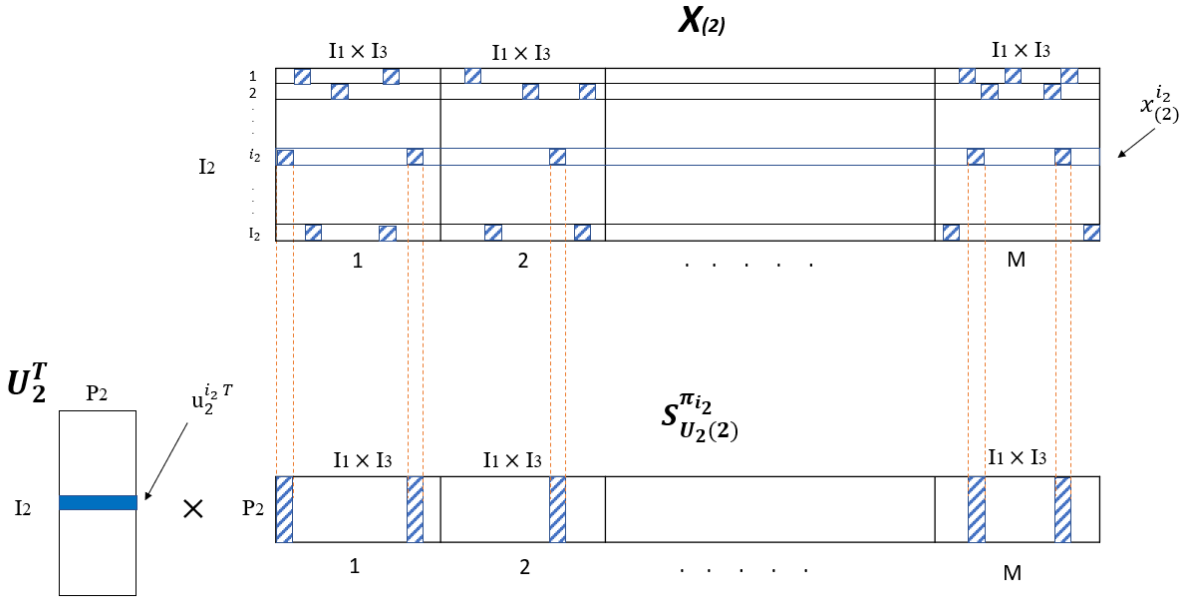


Figure A.3: An illustration of the data missing pattern in Proposition 7 (stripes representing available entries).

A.9 Proof of Proposition 8

The original optimization problem is

$$\arg\min_{\boldsymbol{U}_2} \ \alpha||\mathscr{P}_\Omega(\mathscr{X}-\mathscr{S}\times_1\boldsymbol{U}_1^\top\times_2\boldsymbol{U}_2^\top\times_3\boldsymbol{U}_3^\top)||_F^2+(1-\alpha)||\boldsymbol{y}-\boldsymbol{1}_M\cdot\tilde{\boldsymbol{\beta}}_0-\boldsymbol{S}_{(4)}\cdot\tilde{\boldsymbol{\beta}}_1||_F^2,$$

which is equivalent to the following problem when data is incomplete:

$$\arg\min_{\boldsymbol{U}_2} \ \alpha||\mathscr{X}-(\mathscr{S}\times_1\boldsymbol{U}_1^\top\times_2\boldsymbol{U}_2^\top\times_3\boldsymbol{U}_3^\top)\odot\text{logic}(\mathscr{X})||_F^2+(1-\alpha)||\boldsymbol{y}-\boldsymbol{1}_M\cdot\tilde{\boldsymbol{\beta}}_0-\boldsymbol{S}_{(4)}\cdot\tilde{\boldsymbol{\beta}}_1||_F^2,$$

where $\odot$ is the inner product, $\text{logic}(\mathscr{X})$ denotes the logical value of $\mathscr{X}$. Since the optimization criterion is convex, it can be solved by setting the derivatives to be zeros, i.e., $\frac{d\Psi}{d\boldsymbol{U}_2}=\boldsymbol{0}$, where $\Psi=\alpha||\mathscr{X}-(\mathscr{S}\times_1\boldsymbol{U}_1^\top\times_2\boldsymbol{U}_2^\top\times_3\boldsymbol{U}_3^\top)\odot\text{logic}(\mathscr{X})||_F^2+(1-\alpha)||\boldsymbol{y}-\boldsymbol{1}_M\cdot\tilde{\boldsymbol{\beta}}_0-\boldsymbol{S}_{(4)}\cdot\tilde{\boldsymbol{\beta}}_1||_F^2$. This implies $\frac{d}{d\boldsymbol{U}_2}(||\mathscr{X}-(\mathscr{S}\times_1\boldsymbol{U}_1^\top\times_2\boldsymbol{U}_2^\top\times_3\boldsymbol{U}_3^\top)\odot\text{logic}(\mathscr{X})||_F^2)=\boldsymbol{0}$. According to the communication law of tensor mode multiplication, we have $\frac{d}{d\boldsymbol{U}_2}(||\mathscr{X}-[(\mathscr{S}\times_1\boldsymbol{U}_1^\top\times_3\boldsymbol{U}_3^\top)\times_2\boldsymbol{U}_2^\top]\odot\text{logic}(\mathscr{X})||_F^2)=\boldsymbol{0}$. As a result, $\frac{d}{d\boldsymbol{U}_2}(||\mathscr{X}-(\mathscr{S}_{\boldsymbol{U}_2}\times_2\boldsymbol{U}_2^\top)\odot\text{logic}(\mathscr{X})||_F^2)=\boldsymbol{0}$, where $\mathscr{S}_{\boldsymbol{U}_2}=\mathscr{S}\times_1\boldsymbol{U}_1^\top\times_3\boldsymbol{U}_3^\top$. Furthermore, we have $\frac{d}{d\boldsymbol{U}_2}(||\boldsymbol{X}_{(2)}-(\boldsymbol{U}_2^\top\cdot\boldsymbol{S}_{U_2(2)})\odot\text{logic}(\boldsymbol{X}_{(2)})||_F^2)=\boldsymbol{0}$ since $||\mathscr{S}||_F^2=||\boldsymbol{S}_{(n)}||_F^2$ and $\mathscr{S}\times_n\boldsymbol{U}=\boldsymbol{U}\cdot\boldsymbol{S}_{(n)}$.

It can be easily shown that $\mathscr{X}$'s mode-2 matricization $\boldsymbol{X}_{(2)}$ has missing columns as

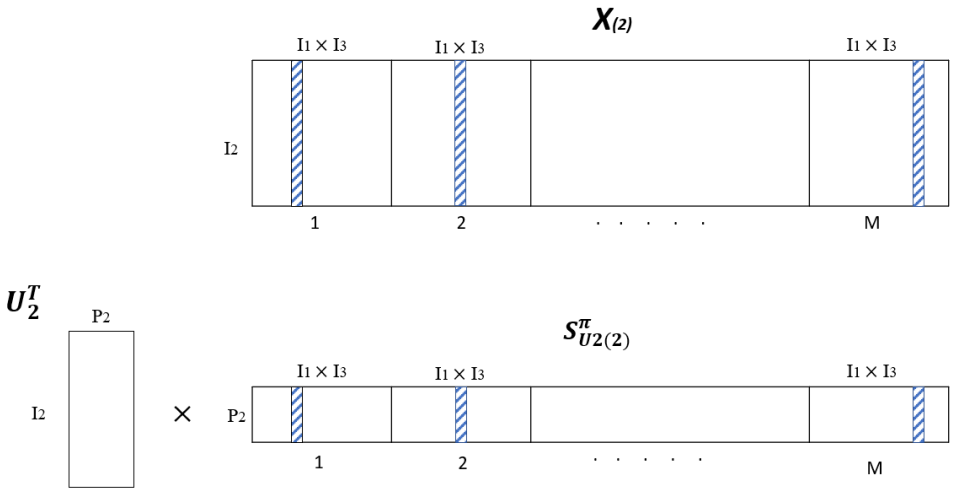


Figure A.4: An illustration of the data missing pattern in Proposition 8 (stripes representing available columns).

well (see Figure A.4 for an illustration). Therefore, similar to the proof of Proposition 9, we have $\frac{d}{d\boldsymbol{U}_2}(||\boldsymbol{X}^{\pi}_{(2)}-\boldsymbol{U}_2^{\top}\cdot\boldsymbol{S}^{\pi}_{U_2(2)}||^2_F)=0$, where π denotes the indices of available columns in $\boldsymbol{X}_{(2)}$, $\boldsymbol{S}^{\pi}_{U_2(2)}$ denotes a matrix constituting the π columns of $\boldsymbol{S}_{U_2(2)}$. As a result, we have $2(\boldsymbol{X}^{\pi}_{(2)}-\boldsymbol{U}_2^{\top}\cdot\boldsymbol{S}^{\pi}_{U_2(2)})\cdot(-\boldsymbol{S}^{\pi}_{U_2(2)}{}^{\top})=0$. Thus, $\boldsymbol{U}_2^{\top}\cdot\boldsymbol{S}^{\pi}_{U_2(2)}\cdot\boldsymbol{S}^{\pi}_{U_2(2)}{}^{\top}=\boldsymbol{X}^{\pi}_{(2)}\cdot\boldsymbol{S}^{\pi}_{U_2(2)}{}^{\top}$, which gives that $\boldsymbol{U}_2^{\top}=\boldsymbol{X}^{\pi}_{(2)}\cdot\boldsymbol{S}^{\pi}_{U_2(2)}{}^{\top}\cdot(\boldsymbol{S}^{\pi}_{U_2(2)}\cdot\boldsymbol{S}^{\pi}_{U_2(2)}{}^{\top})^{-1}$. This yields the analytical solution $\boldsymbol{U}_2=(\boldsymbol{X}^{\pi}_{(2)}\cdot\boldsymbol{S}^{\pi}_{U_2(2)}{}^{\top}\cdot(\boldsymbol{S}^{\pi}_{U_2(2)}\cdot\boldsymbol{S}^{\pi}_{U_2(2)}{}^{\top})^{-1})^{\top}$.

A.10 Proof of Proposition 9

The original optimization problem is

$$\arg\min_{\boldsymbol{U}_3}\ \alpha||\mathscr{P}_{\Omega}(\mathscr{X}-\mathscr{S}\times_1\boldsymbol{U}_1^{\top}\times_2\boldsymbol{U}_2^{\top}\times_3\boldsymbol{U}_3^{\top})||^2_F+(1-\alpha)||\boldsymbol{y}-\boldsymbol{1}_M\cdot\tilde{\boldsymbol{\beta}}_0-\boldsymbol{S}_{(4)}\cdot\tilde{\boldsymbol{\beta}}_1||^2_F,$$

which is equivalent to the following problem when data is missing:

$$\arg\min_{\boldsymbol{U}_3}\ \alpha||\mathscr{X}-(\mathscr{S}\times_1\boldsymbol{U}_1^{\top}\times_2\boldsymbol{U}_2^{\top}\times_3\boldsymbol{U}_3^{\top})\odot\text{logic}(\mathscr{X})||^2_F+(1-\alpha)||\boldsymbol{y}-\boldsymbol{1}_M\cdot\tilde{\boldsymbol{\beta}}_0-\boldsymbol{S}_{(4)}\cdot\tilde{\boldsymbol{\beta}}_1||^2_F,$$

where $\odot$ is the inner product, logic$(\mathscr{X})$ denotes the logical value of $\mathscr{X}$. Since the problem is convex, it can be solved by setting the derivatives to be zeros, i.e., $\frac{d\Psi}{d\boldsymbol{U}_3}=0$, where $\Psi=\alpha||\mathscr{X}-(\mathscr{S}\times_1\boldsymbol{U}_1^{\top}\times_2\boldsymbol{U}_2^{\top}\times_3\boldsymbol{U}_3^{\top})\odot\text{logic}(\mathscr{X})||^2_F+(1-\alpha)||\boldsymbol{y}-\boldsymbol{1}_M\cdot\tilde{\boldsymbol{\beta}}_0-\boldsymbol{S}_{(4)}\cdot\tilde{\boldsymbol{\beta}}_1||^2_F$. This implies

$\frac{d}{d\boldsymbol{U}_3}(\|\mathcal{X}-(\mathcal{S}\times_1\boldsymbol{U}_1^\top\times_2\boldsymbol{U}_2^\top\times_3\boldsymbol{U}_3^\top)\odot\text{logic}(\mathcal{X})\|_F^2)=\mathbf{0}$. According to the communication law of tensor mode multiplication, we have $\frac{d}{d\boldsymbol{U}_3}(\|\mathcal{X}-[(\mathcal{S}\times_1\boldsymbol{U}_1^\top\times_2\boldsymbol{U}_2^\top)\times_3\boldsymbol{U}_3^\top]\odot\text{logic}(\mathcal{X})\|_F^2)=\mathbf{0}$. Thus, $\frac{d}{d\boldsymbol{U}_3}(\|\mathcal{X}-(\mathcal{S}_{U_3}\times_3\boldsymbol{U}_3^\top)\odot\text{logic}(\mathcal{X})\|_F^2)=\mathbf{0}$, where $\mathcal{S}_{U_3}=\mathcal{S}\times_1\boldsymbol{U}_1^\top\times_2\boldsymbol{U}_2^\top$. Furthermore, we have $\frac{d}{d\boldsymbol{U}_3}(\|\boldsymbol{X}_{(3)}-(\boldsymbol{U}_3^\top\cdot\boldsymbol{S}_{U_3(3)})\odot\text{logic}(\boldsymbol{X}_{(3)})\|_F^2)=\mathbf{0}$ since $\|\mathcal{S}\|_F^2=\|\boldsymbol{S}_{(n)}\|_F^2$ and $\mathcal{S}\times_n\boldsymbol{U}=\boldsymbol{U}\cdot\boldsymbol{S}_{(n)}$.

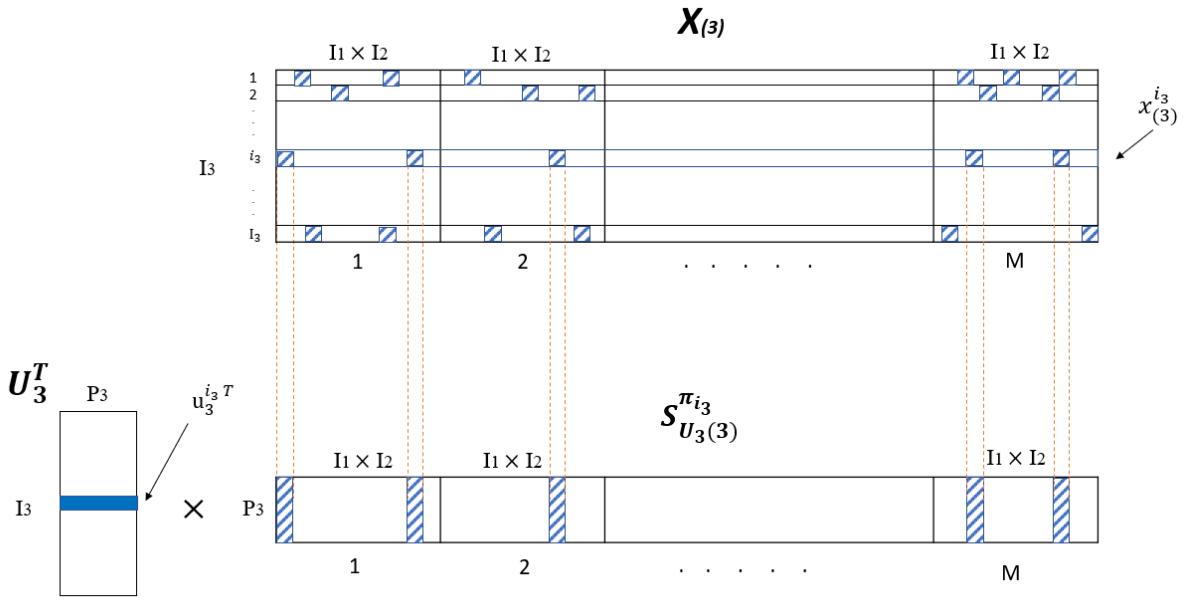


Figure A.5: An illustration of the data missing pattern in Proposition 9 (stripes representing available entries).

Figure A.5 shows the pattern of mode-3 matricization of the 4D tensor $\mathcal{X}$ when it has missing entries whose indices can be denoted by a set $\Omega\subseteq\{(i_1,i_2,i_3,m),1\le i_1\le I_1,1\le i_2\le I_2,1\le i_3\le I_3,1\le m\le M\}$. Based on Lemma 1, we can sequentially optimize each column of $\boldsymbol{U}_3$. The i_3th row in $\boldsymbol{U}_3^\top$ is denoted as $\boldsymbol{u}_3^{i_3\top}$ (blue solid row of $\boldsymbol{U}_3^\top$ in Figure A.5). The available entries in the i_3th row of $\boldsymbol{X}_{(3)}$ are denoted as $\boldsymbol{x}_{(3)}^{i_3,\pi_{i_3}}$ (blue striped squares of $\boldsymbol{x}_{(3)}^{i_3}$ in Figure A.5). In $\boldsymbol{S}_{U_3(3)}^{\pi_{i_3}}$, we choose the columns whose indices are the same as those of the available entries of $\boldsymbol{x}_{(3)}^{i_3}$ (blue striped columns of $\boldsymbol{S}_{U_3(3)}^{\pi_{i_3}}$ in Figure A.5). Thus, we have $\frac{d}{d\boldsymbol{u}_3^{i_3}}(\Sigma_{i_3=1}^{I_3}\|\boldsymbol{x}_{(3)}^{i_3,\pi_{i_3}}-(\boldsymbol{u}_3^{i_3\top}\cdot\boldsymbol{S}_{U_3(3)}^{\pi_{i_3}})\|_F^2)=\mathbf{0}$, which yields $\frac{d}{d\boldsymbol{u}_3^{i_3}}(\|\boldsymbol{x}_{(3)}^{i_3,\pi_{i_3}}-(\boldsymbol{u}_3^{i_3\top}\cdot\boldsymbol{S}_{U_3(3)}^{\pi_{i_3}})\|_F^2)=\mathbf{0}$. By taking the derivative of the Frobenius norm, we have $2(\boldsymbol{x}_{(3)}^{i_3,\pi_{i_3}}-\boldsymbol{u}_3^{i_3\top}\cdot\boldsymbol{S}_{U_3(3)}^{\pi_{i_3}})\cdot(-\boldsymbol{S}_{U_3(3)}^{\pi_{i_3}\top})=\mathbf{0}$. Thus, $\boldsymbol{u}_3^{i_3\top}\cdot\boldsymbol{S}_{U_3(3)}^{\pi_{i_3}}\cdot\boldsymbol{S}_{U_3(3)}^{\pi_{i_3}\top}=\boldsymbol{x}_{(3)}^{i_3,\pi_{i_3}}\cdot\boldsymbol{S}_{U_3(3)}^{\pi_{i_3}\top}$, which gives that $\boldsymbol{u}_3^{i_3}=(\boldsymbol{x}_{(3)}^{i_3,\pi_{i_3}}\cdot\boldsymbol{S}_{U_3(3)}^{\pi_{i_3}\top}\cdot(\boldsymbol{S}_{U_3(3)}^{\pi_{i_3}}\cdot\boldsymbol{S}_{U_3(3)}^{\pi_{i_3}\top})^{-1})^\top$.

A.11 Proof of Proposition 10

The original optimization problem is

$$\arg\min_{S} \ \alpha||\mathscr{P}_{\Omega}(\mathscr{X}-\mathscr{S}\times_1 \boldsymbol{U}_1^{\top}\times_2 \boldsymbol{U}_2^{\top}\times_3 \boldsymbol{U}_3^{\top})||_F^2+(1-\alpha)||\boldsymbol{y}-\boldsymbol{1}_M\cdot\beta_0-\boldsymbol{S}_{(4)}\cdot\boldsymbol{\beta}_1||_F^2$$

which is equivalent to the following problem when data is missing:

$$\arg\min_{S} \ \alpha||\mathscr{X}-(\mathscr{S}\times_1 \boldsymbol{U}_1^{\top}\times_2 \boldsymbol{U}_2^{\top}\times_3 \boldsymbol{U}_3^{\top})\odot \text{logic}(\mathscr{X})||_F^2+(1-\alpha)||\boldsymbol{y}-\boldsymbol{1}_M\cdot\beta_0-\boldsymbol{S}_{(4)}\cdot\boldsymbol{\beta}_1||_F^2$$

where $\odot$ is the inner product, $\text{logic}(\mathscr{X})$ denotes the logical value of $\mathscr{X}$. Since the problem is convex, it can be solved by setting the derivative to be zeros–that is $\frac{d\Psi}{d\boldsymbol{S}}=\boldsymbol{0}$, where $\Psi=\alpha||\mathscr{X}-(\mathscr{S}\times_1 \boldsymbol{U}_1^{\top}\times_2 \boldsymbol{U}_2^{\top}\times_3 \boldsymbol{U}_3^{\top})\odot \text{logic}(\mathscr{X})||_F^2+(1-\alpha)||\boldsymbol{y}-\boldsymbol{1}_M\cdot\beta_0-\boldsymbol{S}_{(4)}\cdot\boldsymbol{\beta}_1||_F^2$. According to the connection between Kronecker product and tensor mode multiplication, we have $\frac{d}{d\boldsymbol{S}}(\alpha||\boldsymbol{X}_{(4)}-[\boldsymbol{S}_{(4)}\cdot(\boldsymbol{U}_3\otimes\boldsymbol{U}_2\otimes\boldsymbol{U}_1)]\odot \text{logic}(\boldsymbol{X}_{(4)})||_F^2+(1-\alpha)||\boldsymbol{y}-\boldsymbol{1}_M\odot\beta_0-\boldsymbol{S}_{(4)}\odot\boldsymbol{\beta}_1||_F^2)=\boldsymbol{0}$.

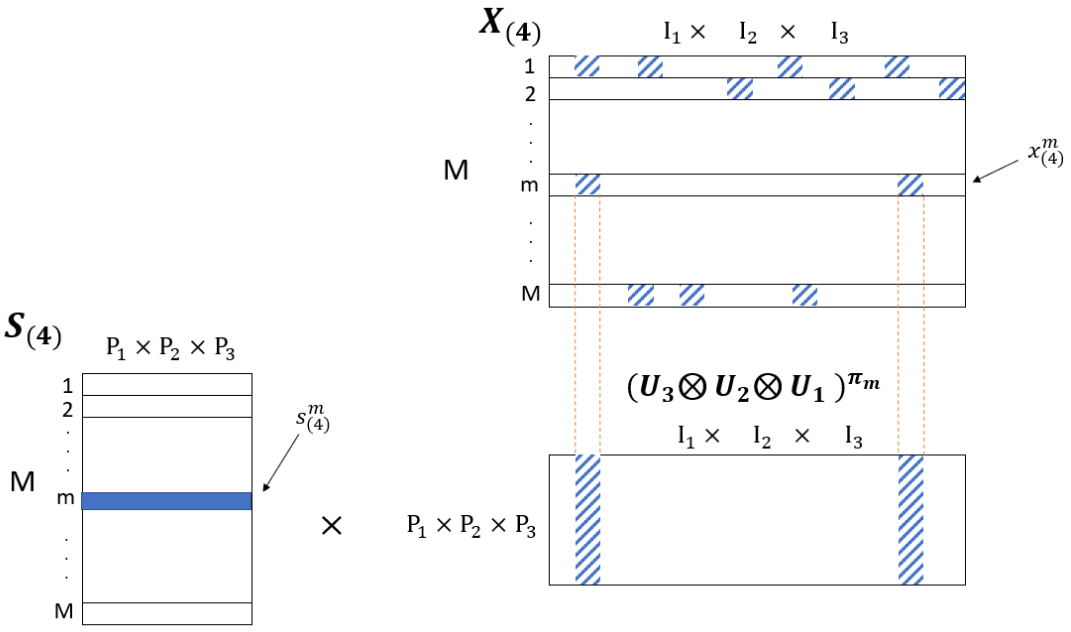


Figure A.6: An illustration of the data missing pattern in Proposition 10 (stripes representing available entries).

Figure A.6 shows the pattern of mode-4 matricization of the 4D tensor $\mathscr{X}$ when it has

missing entries whose indices can be denoted by a set $\Omega \subseteq \{(i_1, i_2, i_3, m), 1 \leq i_1 \leq I_1, 1 \leq i_2 \leq I_2, 1 \leq i_3 \leq I_3, 1 \leq m \leq M\}$. Based on Lemma 1, we can sequentially optimize each column of $\boldsymbol{U}_3$.

The mth row in $\boldsymbol{S}_{(4)}$ is denoted as $\boldsymbol{s}_{(4)}^m$ (blue solid row of $\boldsymbol{S}_{(4)}$ in Figure A.6). The available entries in the mth row of $\boldsymbol{X}_{(4)}$ are denoted as $\boldsymbol{x}_{(4)}^{m,\pi_m}$ (blue striped squares of $\boldsymbol{x}_{(4)}^m$ in Figure A.6). In $(\boldsymbol{U}_3 \otimes \boldsymbol{U}_2 \otimes \boldsymbol{U}_1)$, we choose the columns whose indices are the same as those of the available entries of $\boldsymbol{x}_{(4)}^m$ (blue striped columns of $(\boldsymbol{U}_3 \otimes \boldsymbol{U}_2 \otimes \boldsymbol{U}_1)^{\pi_m}$ in Figure A.6). Thus, we have $\frac{d}{d\boldsymbol{s}_{(4)}^m}(\alpha\{\Sigma_{m=1}^M ||\boldsymbol{x}_{(4)}^{m,\pi_m} - [\boldsymbol{s}_{(4)}^m \cdot (\boldsymbol{U}_3 \otimes \boldsymbol{U}_2 \otimes \boldsymbol{U}_1)^{\pi_m}]\}||_F^2 + (1-\alpha)\Sigma_{m=1}^M ||y_m - \beta_0 - \boldsymbol{s}_{(4)}^m \cdot \boldsymbol{\beta}_1||_F^2) = \boldsymbol{0}$, which yields $\frac{d}{d\boldsymbol{s}_{(4)}^m}(\alpha\{||\boldsymbol{x}_{(4)}^{m,\pi_m} - [\boldsymbol{s}_{(4)}^m \cdot (\boldsymbol{U}_3 \otimes \boldsymbol{U}_2 \otimes \boldsymbol{U}_1)^{\pi_m}]\}||_F^2 + (1-\alpha)||y_m - \beta_0 - \boldsymbol{s}_{(4)}^m \cdot \boldsymbol{\beta}_1||_F^2) = \boldsymbol{0}$. By taking the derivative of Frobenius norm, we have $2\alpha \cdot [\boldsymbol{x}_{(4)}^{m,\pi_m} - \boldsymbol{s}_{(4)}^m \cdot (\boldsymbol{U}_3 \otimes \boldsymbol{U}_2 \otimes \boldsymbol{U}_1)^{\pi_m}] \cdot [-(\boldsymbol{U}_3 \otimes \boldsymbol{U}_2 \otimes \boldsymbol{U}_1)^{\pi_m \top}] + 2(1-\alpha) \cdot [y_m - \beta_0 - \boldsymbol{s}_{(4)}^m \cdot \boldsymbol{\beta}_1] \cdot (-\boldsymbol{\beta}_1^\top) = \boldsymbol{0}$. Thus, $-2\alpha \cdot \boldsymbol{x}_{(4)}^{m,\pi_m} \cdot (\boldsymbol{U}_3 \otimes \boldsymbol{U}_2 \otimes \boldsymbol{U}_1)^{\pi_m \top} + 2\alpha \cdot \boldsymbol{s}_{(4)}^m \cdot (\boldsymbol{U}_3 \otimes \boldsymbol{U}_2 \otimes \boldsymbol{U}_1)^{\pi_m} \cdot (\boldsymbol{U}_3 \otimes \boldsymbol{U}_2 \otimes \boldsymbol{U}_1)^{\pi_m \top} + 2(1-\alpha)(y_m - \beta_0) \cdot (-\boldsymbol{\beta}_1^\top) + 2(1-\alpha) \cdot (\boldsymbol{s}_{(4)}^m \cdot \boldsymbol{\beta}_1 \cdot \boldsymbol{\beta}_1^\top) = \boldsymbol{0}$, which gives that $\boldsymbol{s}_{(4)}^m = \left[\alpha \cdot \boldsymbol{x}_{(4)}^{m,\pi_m} \cdot (\boldsymbol{U}_3 \otimes \boldsymbol{U}_2 \otimes \boldsymbol{U}_1)^{\pi_m \top} + (1-\alpha) \cdot (y_m - \beta_0) \cdot \boldsymbol{\beta}_1^\top\right] \cdot \left[\alpha \cdot (\boldsymbol{U}_3 \otimes \boldsymbol{U}_2 \otimes \boldsymbol{U}_1)^{\pi_m} \cdot (\boldsymbol{U}_3 \otimes \boldsymbol{U}_2 \otimes \boldsymbol{U}_1)^{\pi_m \top} + (1-\alpha) \cdot \boldsymbol{\beta}_1 \cdot \boldsymbol{\beta}_1^\top\right]^{-1}$.

APPENDIX

B

PROOF IN CHAPTER 4

B.1 Proof of Proposition 11

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 $\boldsymbol{\Phi}^{(n)*} = \sum_{d=1}^{D} \sum_{m_d=1}^{M_d} \tilde{\boldsymbol{X}}^d_{d(n)} \cdot \tilde{\boldsymbol{X}}^{d^\top}_{d(n)}$ in Equation (4.7), 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 $\boldsymbol{\Phi}^{(n)*}$.

B.2 Proof of Proposition 12

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

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

B.3 Proof of Proposition 13

From Equation (4.10), 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 $\{\tilde{\boldsymbol{X}}_{m_d(n)}^{d\boldsymbol{\Phi}}\in\mathbb{R}^{I_n\times(P_1\times.....\times P_{n-1}\times P_{n+1}\times...\times P_N)},\ m_d=1,\ldots,M_d;\ d=1,\ldots,D\}$ 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 $\boldsymbol{A}_{(n)}^{\boldsymbol{\Phi}}$, and $\boldsymbol{V}\in\mathbb{R}^{r\times(P_1\times...\times P_{n-1}\times P_{n+1}\times...\times P_N\times\sum_{d=1}^{D}M_d)}$ is right unitary matrix of $\boldsymbol{A}_{(n)}^{\boldsymbol{\Phi}}$, $\boldsymbol{A}_{(n)}^{\boldsymbol{\Phi}}\boldsymbol{A}_{(n)}^{\boldsymbol{\Phi}\top}$ can be expressed as $\boldsymbol{A}_{(n)}^{\boldsymbol{\Phi}}\boldsymbol{A}_{(n)}^{\boldsymbol{\Phi}\top}=\boldsymbol{U}^{(n)}\boldsymbol{\Sigma}\boldsymbol{V}^\top\cdot(\boldsymbol{U}^{(n)}\boldsymbol{\Sigma}\boldsymbol{V}^\top)^\top$. This implies $\boldsymbol{A}_{(n)}^{\boldsymbol{\Phi}}\boldsymbol{A}_{(n)}^{\boldsymbol{\Phi}\top}=\boldsymbol{U}^{(n)}\boldsymbol{\Sigma}\boldsymbol{V}^\top\boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{U}^{(n)\top}$. Furthermore, $\boldsymbol{A}_{(n)}\boldsymbol{A}_{(n)}^\top=\boldsymbol{U}^{(n)}\boldsymbol{\Sigma}^2\boldsymbol{U}^{(n)\top}$ since $\boldsymbol{V}$ is orthogonal matrix.

Therefore, we have $\boldsymbol{A}_{(n)}^{\boldsymbol{\Phi}}\boldsymbol{A}_{(n)}^{\boldsymbol{\Phi}\top}=\sum_{d=1}^{D}\sum_{m_d=1}^{M_d}\tilde{\boldsymbol{X}}_{m_d}^{d\boldsymbol{\Phi}}\tilde{\boldsymbol{X}}_{m_d}^{d\boldsymbol{\Phi}\top}=\boldsymbol{\Phi}^{(n)}=\boldsymbol{U}^{(n)}\boldsymbol{\Sigma}^2\boldsymbol{U}^{(n)\top}$. According to the definition of eigen-decomposition, $\boldsymbol{U}^{(n)}$ comprises all the eigenvectors of $\boldsymbol{\Phi}^{(n)}$ in equation (4.10).

B.4 Proof of Proposition 14

Similar to the proof of proposition 2, we take the first iteration in algorithm (5) as an example.

A new matrix $[\boldsymbol{A}_{1(n)}^{\boldsymbol{\Phi}},\quad \boldsymbol{s}]$ 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 $[\boldsymbol{A}_{1(n)}^{\boldsymbol{\Phi}},\quad \boldsymbol{s}]$, then we have $[\boldsymbol{A}_{1(n)}^{\boldsymbol{\Phi}},\quad \boldsymbol{s}]=\boldsymbol{U}_{direct}^{\boldsymbol{\Phi}}\boldsymbol{\Sigma}_{direct}^{\boldsymbol{\Phi}}\boldsymbol{V}_{direct}^{\boldsymbol{\Phi}}$, where $\boldsymbol{U}_{direct}^{\boldsymbol{\Phi}}$,$\boldsymbol{\Sigma}_{direct}^{\boldsymbol{\Phi}}$, $\boldsymbol{V}_{direct}^{\boldsymbol{\Phi}}$ denote the left unitary matrix, diagonal matrix with singular values and the right unitary matrix of $[\boldsymbol{A}_{1(n)}^{\boldsymbol{\Phi}},\quad \boldsymbol{s}]$, respectively.

Based on algorithm (5), we conduct SVD to the first user block $\boldsymbol{A}_{1(n)}^{\boldsymbol{\Phi}}$ 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:

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

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:

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

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

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

B.5 Proof in 4.2.4.1

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