Understanding spatial-temporal trends using communication-efficient federated tensor factorization for incomplete data

By

Linghui Zeng

Dr. Joyce Ho
Advisor

Computer Science

Dr. Joyce Ho
Advisor

Dr. Jeremy Jacobson
Committee Member

Dr. Carl Yang
Committee Member

Dr. Liang Zhao
Committee Member

2022

Understanding spatial-temporal trends using communication-efficient federated tensor factorization for incomplete data

By

Linghui Zeng

Dr. Joyce Ho
Advisor

An abstract of
A thesis submitted to the Faculty of the
Emory College of Arts and Sciences of Emory University
in partial fulfillment of the requirements for the degree of
Bachelor of Science with Honors

Computer Science

2022

Abstract

Understanding spatial-temporal trends using communication-efficient federated tensor factorization for incomplete data
By Linghui Zeng

Extracting trends from spatio-temporal data, such as the Google COVID-19 Search Trends Symptoms Dataset or Chicago Crime Dataset, can be used to investigate changes related to health or the environment, respectively. Tensor factorization can naturally capture space and time dependence to identify meaningful patterns.

Recent advances in federated tensor learning have further enabled joint learning across multiple sources in a privacy-preserving manner. Yet, measurements can be erroneous and missing in spatio-temporal data and can negatively impact the current federated tensor factorization approaches. In this paper, we develop a robust federated tensor factorization framework, `FedTefid`, that is not only efficient from a communication and computational perspective but also able to extract temporal patterns from incomplete data via a temporal smoothness constraint. Experiment results show that our proposed method can recover the spatio-temporal patterns even with 90% of the measurements missing.

Understanding spatial-temporal trends using communication-efficient federated tensor factorization for incomplete data

By

Linghui Zeng

Dr. Joyce Ho
Advisor

A thesis submitted to the Faculty of the
Emory College of Arts and Sciences of Emory University
in partial fulfillment of the requirements for the degree of
Bachelor of Science with Honors

Computer Science

2022

Acknowledgments

Many thanks to Emory University for the great four years and for the opportunity of honor program. It was my fortune to have Dr. Joyce Ho as my thesis advisor. Thanks so much to Dr. Joyce Ho for believing in my ability and my potential and giving me the opportunity to participate in the honor program, even though I had not taken her class before. Most importantly, thank Dr. Joyce Ho so much for her guidance, patience and support. Moreover, thank so much to Jing Ma, Dr. Joyce Ho's PhD student, for her helps and supports throughout the entire research project! And finally, thanks so much to my family and all my friends for all the supports!

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Patterns identified in spatio-temporal data can be used to better understand the impacts of changes related to health or the environment. As a motivating example, COVID-19 has greatly impacted people's lives in the last two years. The COVID-19 Search Trends symptoms dataset [8] can be used to extract meaningful trends to gain insight into the impact of the virus on communities as well as to detect outbreaks earlier. With the need to model both time and space simultaneously to better understand the changes across regions and over time, we express as a third-order spatio-temporal tensor [7], or multidimensional array, where each element captures the number of searches for each county for a certain week. Tensor decomposition, such as CANDECOMP/PARAFAC (CP) decomposition, can then be used to analyze the data and extract hidden and meaningful patterns from the data. In particular, CP decomposition factorizes the higher-order tensor as a sum of several rank-one tensors (i.e. outer products of N vectors) and has been widely used in many domains, including psychometrics, computer vision, and computational phenotyping.

In real applications, multidimensional data may not always be stored in one location. For example, consider the scenario where each region of the United States (i.e. West, South West, Mid-West, South East, North East) collects the COVID-19 search data locally and the data cannot be stored at a central server either due to privacy concerns or sheer volume. For such cases, federated tensor factorization can be used to communicate the county-level symptom trends with the Centers for Disease Control and Prevention (CDC). Federated tensor factorization has been proposed to improve the scalability for decentralized data [6, 9, 14]. It also shares the spirit of privacy preserving of the more general federated learning [5], as it avoids communicating the raw tensor and individual-level data.

Unfortunately, existing federated tensor factorization methods cannot readily handle incomplete data, which is ubiquitous across different domains due to various reasons in the data collection process. In the web search setting, data is often anonymized by either not reporting the data or setting the values to be zero. Another factor might be that different regions or sites have different data collection and data processing techniques, which can yield widely different sampling strategies that need to be dealt with. Similarly, the machines collecting the measurements may suddenly fail and not be able to capture the data and results in missing entries.

To solve the missing data problem in federated tensor factorization, we propose `FedTefid`, a **Fed**erated tensor factorization framework with **te**mporal constraints **f**or **i**ncomplete **d**ata. Our model can robustly extract temporal patterns from spatio-temporal data with missing entries by utilizing the temporal smoothness property and only modeling the observed data. We demonstrate the benefit of `FedTefid` on two spatio-temporal datasets, the Chicago Crime dataset and the COVID-19 Search Trends Symptoms dataset. Our experimental results illustrate that our method can recover the patterns even with 90% of the measurements missing.

Chapter 2

Background

2.1 Tensor

A tensor [7] can be defined as a multidimensional array. The *order* of a tensor is the number of its dimensions. A matrix is a second-order tensor. A higher order tensor is denoted by $\boldsymbol{\mathcal{X}}$. And the term *mode* is used to refer to a specific dimension. An order D tensor is denoted by $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_D}$, where $\mathcal{X}_{i_1, i_2, \ldots, i_D}$ indicates its $(i_1, i_2, ..., i_D)$-th element. The index set of all tensor entries is denoted by $\mathcal{I}$, where $|\mathcal{I}| = \prod_{d=1}^{D} I_d$.

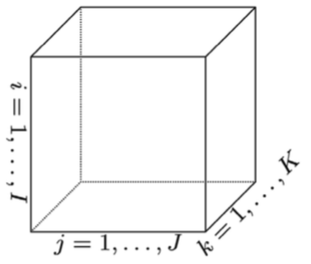

Figure 2.1: A third-order tensor: $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I \times J \times K}$

2.1.1 Rank-One Tensors

A nth-order tensor [7] $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ is rank one if it can be written as the outer product of N vectors, i.e.,

$$\boldsymbol{\mathcal{X}} = a^{(1)} \circ a^{(1)} \circ \ldots \circ a^{(N)}.$$

The symbol "$\circ$" represents the vector outer product, and each element of the tensor is the product of its corresponding vector elements, i.e.,

$$\boldsymbol{\mathcal{X}}_{i_1 i_2 \ldots i_N} = a^{(1)}_{i_1} a^{(2)}_{i_2} \ldots a^{(N)}_{i_N} \text{ for all } 1 \leq i_n \leq I_n.$$

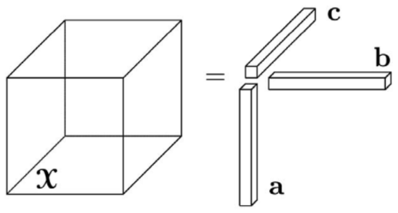

Figure 2.2: A rank-one third-order tensor: $\boldsymbol{\mathcal{X}} = a \circ b \circ c$. The (i, j, k) element of $\boldsymbol{\mathcal{X}}$ is given by $x_{ijk} = a_i b_j c_k$

2.1.2 Tensor Matricization

Tensor matricization [7] is the process to reorder the elements of an nth-order tensor into a matrix. The mode-n matricization of a tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ is denoted by $\mathbf{X}_{(n)}$ and the mode-n fibers are arranged to be the columns of the resulting matrix. The tensor element $(i_1, i_2, \ldots, i_N)$ will be mapped to matrix element (i_n, j) where $j = 1 + \sum_{k=1, k \neq n}^{N} (i_k - 1) J_k$ with $J_k = \prod_{m=1, m \neq n}^{k-1} I_m$.

For a tensor $\mathcal{X} \in \mathbb{R}^{3\times4\times2}$ with frontal slices:

$$X_1 = \begin{bmatrix} 1 & 4 & 7 & 10 \\ 2 & 5 & 8 & 11 \\ 3 & 6 & 9 & 12 \end{bmatrix}, \ X2 = \begin{bmatrix} 13 & 16 & 19 & 22 \\ 14 & 17 & 20 & 23 \\ 15 & 18 & 21 & 24 \end{bmatrix},$$

the three mode-n unfoldings are

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

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

$$X_{(3)} = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & ... & 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 & 17 & ... & 21 & 22 & 23 & 24 \end{bmatrix}.$$

2.2 CP Decomposition

In 1927, Hitchcock proposed the idea of expressing a tensor as the sum of a finite number of rank-one tensors, and the idea became popular after another introduction as CP (CANDECOMP/PARAFAC) decomposition.

The CP decomposition [7] factorizes a tensor into a sum of component rank-one tensors. For example, a third-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I\times J\times K}$ can be approximated as:

$$\boldsymbol{\mathcal{X}} \approx \sum_{r=1}^{R} a_r \circ b_r \circ c_r,$$

where R is a positive integer and $a_r \in \mathbb{R}^I, b_r \in \mathbb{R}^J, c_r \in \mathbb{R}^K$.

And each element of the tensor can be approximated as

$$x_{ijk} \approx \sum_{r=1}^{R} a_{ir} b_{jr} c_{kr} \text{ for } i = 1, 2, ..., I; j = 1, 2, ..., J; k = 1, 2, ..., K.$$

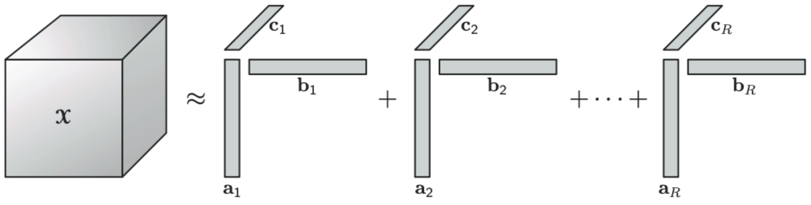

Figure 2.3: CP decomposition of a third-order tensor

The *rank* of a tensor $\boldsymbol{\mathcal{X}}$, denoted as $rank(\boldsymbol{\mathcal{X}})$, is the smallest number of rank-one tensors that can generate $\boldsymbol{\mathcal{X}}$ as their sum. However, there is no finite algorithm to determine the rank of a tensor. In practice, we evaluate different ranks by fitting the various CP decomposition models and pick the "best" one.

For a third-order tensor $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{I \times J \times K}$, the standard computation for a CP decomposition with rank R that best approximates $\boldsymbol{\mathcal{X}}$ is to solve the following objective function:

$$\min_{\hat{x}} ||\boldsymbol{\mathcal{X}} - \hat{\boldsymbol{\mathcal{X}}}||, \text{ with } \hat{\boldsymbol{\mathcal{X}}} = \sum_{r=1}^{R} \lambda_r a_r \circ b_r \circ c_r = [[\lambda; A, B, C]]. \tag{2.1}$$

2.2.1 Generalized CP

The above formulation (Eq. (2.1) assumes the data is numeric and follows a normal distribution. However in practice, data may not satisfy this assumption as is the case with nonnegative, discrete, or boolean data. Thus, a framework to generalize CP [4] to a broad category of loss functions was developed. The objective function of the generalized CP tensor factorization (GTF) can be expressed as

$$\arg\min_{\mathcal{U}} F(\mathcal{U}, \mathcal{X}) = \sum_{i \in \mathcal{I}} f(\mathcal{U}(i), \mathcal{X}(i)), \tag{2.2}$$

where $\mathcal{U} = \sum_{i=1}^{R} \mathbf{U}_{(1)}(:, i) \circ ... \circ \mathbf{U}_{(D)}(:, i)$ is the sum of R rank-one tensors, $\mathbf{U}_{(d)}$ represents the factor matrix of mode d, $\circ$ represents the vector outer product, and $f(\mathcal{U}(i), \mathcal{X}(i))$ denotes the element-wise loss function. The common CP model in Eq. (2.1) uses the least square loss, $f(\mathcal{U}(i), \mathcal{X}(i)) = \frac{1}{2}(\mathcal{U}(i) - \mathcal{X}(i))^2$.

2.2.2 CP Decomposition with Missing Data

Data with missing entries is ubiquitous in various domains, including bibliometrics, social network analysis and computer vision. As the increasing uses of tensor as the representation of data, methods to accurately capture the latent structures within tensor with the existence of missing entries, and possibly reconstructing the missing entries (i.e. tensor completion) began to draw scholars' attention.

CP-WOPT [1] (CP Weighted OPTimization) formulates the CP decomposition for incomplete tensor as a weighted least squares problem that models only the observed entries, and uses a first-order optimization approach to solve the weighted least squares problem. However, CP-WOPT can only be used under a centralized setting, where all the data are stored and computed together, and its generalized for all kinds of tensors without specialized focus on the specialty of spatial-temporal tensors.

2.3 Federated Learning

The performance of a machine learning model highly depends on the quality and quantity of training data. However, some data may not be permitted for use because of privacy issues. Also, training with a huge amount of data may incur a huge amount of money for data storage and computational power. Thus, it becomes natural to seek a way to both store data and train the model in a decentralized fashion.

Federated learning (FL) is a machine learning framework where many clients (e.g. mobile devices or organizations) collaboratively train a model under the arrangement of a central server (e.g. service provider), while the training data are kept decentralized [5]. Under the framework of FL, data are stored locally and not directly shared. Only the intermediate results (e.g. parameter updates) will be communicated with the central server. Thus, FL allow us to train a model with as much data as possible with privacy conservation and efficient storage.

2.3.1 Federated Tensor Factorization

Suppose there are K clients capturing I_1 individual information under the federated setting, and the tensor $\mathcal{X} \in \mathbb{R}^{I1 \times I2 \times \ldots \times I_D}$ is collectively held by the K clients. We consider the horizontally partitioned setting, where each client has the same feature space, but non-overlapping individuals. Thus, the local tensor at each client is denoted as $\mathcal{X}^k \in \mathbb{R}^{I_{1k} \times I_2 \times \ldots \times I_D}$, meaning that each client has information on I_{1k} individuals, and $\sum_{k=1}^{K} I_{1k} = I_1$. The objective function of federated GTF is:

$$\arg\min_{\mathbf{U}_{(d)}} \sum_{k=1}^{K} F(\mathcal{U}, \mathcal{X}^k), s.t. \mathcal{U} = \mathbf{U}_{(1)} \circ \ldots \circ \mathbf{U}_{(D)} \tag{2.3}$$

FedGTF-EF-PC

Recently, FedGTF-EF-PC [11] was proposed to efficiently solve Eq. (2.3). FedGTF-EF-PC proposed a three-step approach for reducing the uplink communication cost of federated generalized tensor factorization without compromising the convergence speed. FedGTF-EF-PC introduced a randomized block update, where only the compressed partial gradient of a sampled block is sent to the central server. The algorithm also reduces the number of communication rounds by only periodically sending updates. At the element level, the algorithm also employs gradient compression and error feedback mechanism to reduce the communication cost without hurting convergences. However, FedGTF-EF-PC assumes that all entries are observed. While a straightforward approach is to assume that all missing entries are 0, this can cause a severe degradation in the results and bias the learned latent factors.

Chapter 3

Proposed Method

We propose FedTefid, an extension of the FedGTF-EF-PC model, that can robustly identify spatio-temporal patterns even from incomplete data. We first formulate a new objective function to explicitly model only the observed entries in the local tensors. Then we incorporate a temporal smoothing technique that can mitigate the effect of erroneous and noisy data to better learn the temporal factors.

The execution of FedTefid is demonstrated in Fig. 3.1.

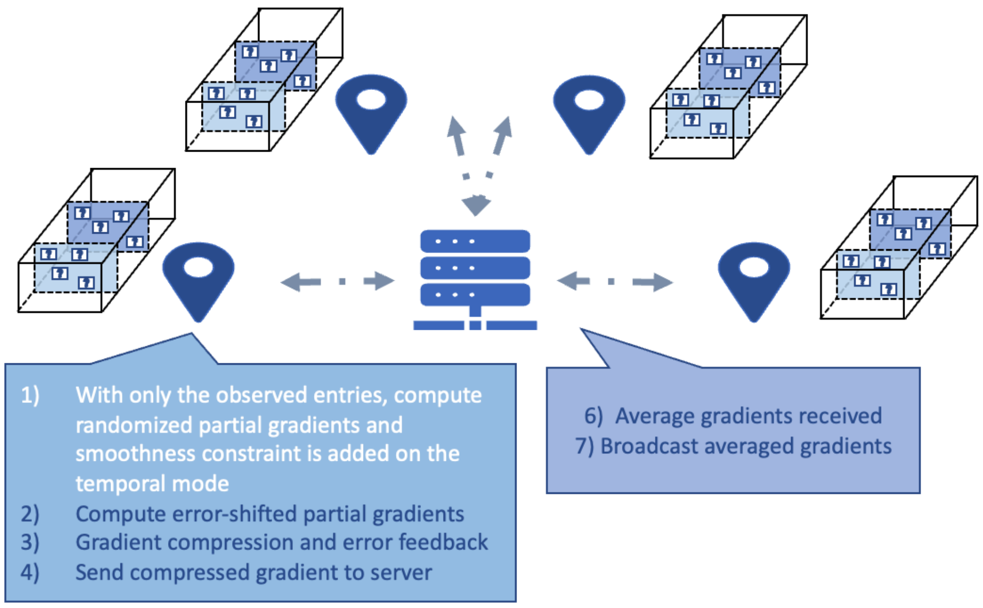
1) With only the observed entries, compute randomized partial gradients and smoothness constraint is added on the temporal mode
2) Compute error-shifted partial gradients
3) Gradient compression and error feedback
4) Send compressed gradient to server
6) Average gradients received
7) Broadcast averaged gradients

Figure 3.1: Illustration of the process of FedTefid.

3.1 Federated GTF with Missing Data

Suppose we have a mask tensor, $\mathcal{M}$, which denotes the missing entries in $\mathcal{X}$, where

$$\mathcal{M}_{i_1,i_2,\ldots i_D} = \begin{cases} 0, & \text{if } \mathcal{X}_{i_1,i_2,\ldots i_D} \text{ is missing} \\ 1, & \text{if } \mathcal{X}_{i_1,i_2,\ldots i_D} \text{ is observed} \end{cases}$$

The GTF objective function in Eq. (2.2) can be expressed as:

$$\arg\min_{\mathcal{U}} F(\mathcal{U}, \mathcal{X}) = \sum_{i \in \mathcal{I}} f(\mathcal{M}(i) * \mathcal{U}(i), \mathcal{M}(i) * \mathcal{X}(i)). \tag{3.1}$$

For example, using the least square loss, the equivalent objective function to Eq. (2.1) is then expressed as:

$$\arg\min_{\mathcal{U}} F(\mathcal{U}, \mathcal{X}) = \sum_{i \in \mathcal{I}} \frac{1}{2}(\mathcal{M}(i) * \mathcal{U}(i) - \mathcal{M}(i) * \mathcal{X}(i))^2 \tag{3.2}$$

as demonstrated in [1].

3.1.1 Gradient updates with observed entries

For the least squares loss, since $\mathcal{M}(i) = 0$ for missing entries at i, it can be shown that Eq. (3.2) is equivalent to the following:

$$\arg\min_{\mathcal{U}} F(\mathcal{U}, \mathcal{X}) = \sum_{i \in \mathcal{I} - \mathcal{P}} \frac{1}{2}(\mathcal{U}(i) - \mathcal{X}(i))^2,$$

where $\mathcal{P}$ denotes the index set of all the missing entries.

Based on the above formulation, we only need to calculate the gradients with the observed entries. During each round of communication, FedGTF-EF-PC calculates the partial stochastic gradients using an efficient fiber sampling technique of the randomized factor. To reflect the changes in Eq. (3.1), we modify the sampling mechanism of the FedGTF-EF-PC algorithm to only sample the indices of the observed entries using the mask tensor, $\mathcal{M}$ for the partial stochastic gradients, thereby ignoring any element in the index set of missing entries, $\mathcal{P}$. We hereby refer to this extension as FedGTFM.

3.2 Temporal smoothing

Although FedGTFM offers some robustness to missing entries by ignoring the gradients associated with the missing entries, the model may still not be able to identify factors in data with large amounts of missing entries. Therefore, we utilize the temporal smoothness property of real-world data, where two successive values along the temporal dimension tend to be closely related to each other [2, 15]. We do so by adding a smoothness regularization along the temporal dimension. Without loss of generality, we assume the t^{th} mode is the temporal dimension. We modify the objective function in Eq. (3.1) to incorporate a regularization function on $\mathbf{U}_{(t)}$:

$$\arg\min_{\mathcal{U}} F(\mathcal{U}, \mathcal{X}) = \sum_{i \in \mathcal{I}} f(\mathcal{M}(i) * \mathcal{U}(i), \mathcal{M}(i) * \mathcal{X}(i)) + \lambda \mathcal{R}\left(\mathbf{U}_{(t)}\right)$$

In particular, we explore both the 2-norm constraint [15] and the 1-norm constraint on the temporal factor to encourage temporal smoothness and further recover the missing entries.

3.2.1 L2 norm

First, we define a smoothness constraint matrix $L \in \mathbb{R}^{(I_t-1)\times I_t}$ as $L_{jj} = 1$ and $L_{j(j+1)} = -1, \forall\ 0 \leq j \leq I_t - 1$ where I_t is the size of the temporal mode. For example,

$$L = \begin{pmatrix} 1 & -1 & & & \\ & 1 & -1 & & \\ & & \ddots & & \\ & & & 1 & -1 \end{pmatrix} \tag{3.3}$$

Thus, if we minimize $||L\mathbf{U}_{(t)}||_F^2 = \sum_{i=1}^{I_n-1} |\mathbf{U}_{(t)}(i) - \mathbf{U}_{(t)}(i+1)|^2$, we are encouraging consecutive values in the temporal factor to not change dramatically. Incorporating the above constraint with the least squares loss yields the following objective function:

$$\arg\min_{\mathbf{U}} F(\mathcal{U}, \mathcal{X}) = \sum_{i \in \mathcal{I}-\mathcal{P}} \frac{1}{2}(\mathcal{U}(i) - \mathcal{X}(i))^2 + \lambda ||L\mathbf{U}_{(t)}||_F^2. \tag{3.4}$$

The regularization parameter λ determines the importance of enforcing the smoothness between two consecutive values. Higher values of λ will penalize solutions that have high temporal variability and therefore may not appropriately reflect the true patterns. However, if there are large amounts of missing data, higher regularization values may help the algorithm recover the original patterns as real-world data tend to be closely related to each other.

Given the 2-norm smoothness constraint, the gradient of $\lambda||L\mathbf{U}_{(t)}||_F^2$ can be calculated as $\lambda L^T * L * \mathbf{U}_{(t)}$. It is important to note that only the partial gradients related to $\mathbf{U}_{(t)}$ will need to be updated to include this term.

3.2.2 L1 norm

We use the same smoothness constraint matrix as shown in Eq. (3.3). For the 1-norm constraint, we define $\mathcal{R}\left(\mathbf{U}_{(t)}\right) = \lambda||L\mathbf{U}_{(t)}||_1$. Note that minimizing $||L\mathbf{U}_{(t)}||_1 = \sum_{i=1}^{I_n-1} |\mathbf{U}_{(t)}(i) - \mathbf{U}_{(t)}(i+1)|$ can also encourage consecutive values in the temporal factor to not change dramatically. The major difference is that the 1-norm will not heavily punish drastic temporal changes should such a pattern truly exist in the data.

The objective function with the L1 norm regularization is:

$$\arg\min_{\mathbf{U}} F(\mathcal{U},\mathcal{X}) = \sum_{i\in\mathcal{I}-\mathcal{P}} \frac{1}{2}(\mathcal{U}(i)-\mathcal{X}(i))^2 + \lambda||L\mathbf{U}_{(t)}||_1. \tag{3.5}$$

Similar to above, the regularization parameter λ controls the extent of smoothness along the temporal dimension. While the 1-norm is not a smooth function, for non-zero values of $\mathbf{U}_{(t)}$, the gradient for $\lambda||L\mathbf{U}_{(t)}||_1$ can be calculated as $\lambda sign(L*\mathbf{U}_{(t)})$.

3.3 `FedTefid`

Given the new objective functions in Eq. (3.4) and (3.5), the goal is for each of the clients to achieve consensus on the factor matrices $\mathbf{U}_d, \forall\, 1\leq d\leq D$. `FedTefid` utilizes the same approaches as FedGTF-EF-PC to reduce the uplink communication cost.

In each iteration, for each client (k) in the federated setting, `FedTefid` calculates the gradient on the randomized sampled mode [13] with *only the observed entries*. Sampled entries for gradient calculation can be generated by fiber sampling [3] with $|S|$ fibers. Let O denote the sampled observed entries for gradient calculation, $|O| = ||S|*D| - |\#missing|$ and $\mathbf{H}_d$ denote the Hadamard Product between all the factor matrices except $\mathbf{U}_d$. The corresponding entries for $\mathbf{H}_d$ is then denoted as O'. The partial stochastic gradient with observed data can be calculate as:

$$\mathbf{G}_d^k[j] = \mathbf{Y}_{(d)}^k(O)\mathbf{H}_d^k(O') \tag{3.6}$$

as demonstrated in [11], where $\mathbf{G}_d^k[j]$ denotes the partial stochastic gradient for client k during t iteration on sampled mode d, $\mathbf{Y}_{(d)}$ is the d-unfolding of the element-wise partial gradient $\mathcal{Y}\in\mathbb{R}^{I_1\times I_2\times\ldots\times I_D}$, where $\mathcal{Y}(i) = \frac{\partial f(\mathcal{U}(i),\mathcal{X}(i))}{\partial\mathcal{U}(i)}$, and $\mathbf{H}^k(O') = \mathbf{U}_{(1)}^k(O_1')\circledast\ldots\circledast\mathbf{U}_{(d-1)}^k(O_{(d-1)}')\circledast\mathbf{U}_{(d+1)}^k(O_{(d+1)}')\circledast\ldots\circledast\mathbf{U}_{(D)}^k(O_{(D)}')$ where $\circledast$ represents the Hadamard Product. If mode d is the temporal mode t, a smoothness constraint will be added to the gradient:

$$\mathbf{G}_d^k[j] = \mathbf{G}_d^k[j] + \lambda L^T * L * \mathbf{U}_{(t)} \tag{3.7}$$

or:

$$\mathbf{G}_d^k[j] = \mathbf{G}_d^k[j] + \lambda sign(L * \mathbf{U}_{(t)}) \tag{3.8}$$

Then, the gradient is updated with error feedback as

$$\mathbf{G}_d^k[j] = \mathbf{G}_d^k[j] + \mathbf{E}_d^k[j-1] \tag{3.9}$$

where $\mathbf{E}_d^k[j-1]$ is the stored error. If this is not a communication iteration, each client updates the factor tensor with the computed gradient: $\mathbf{U}_d^k[j] = \mathbf{U}_d^k[j-1] + \gamma \mathbf{G}_d^k[j]$.

If this is a communication iteration, the gradient is compressed as $\mathbf{C}_d^k[j]$, the low-precision representation of $\mathbf{C}_d^k[j]$.The client sends the compressed updated gradient $\mathbf{C}_d^k[j]$ to the central server and updates the error as:

$$\mathbf{E}_d^k[j] = \mathbf{G}_d^k[j] - \mathbf{C}_d^k[j] \tag{3.10}$$

After receiving gradients from each client, the central server averages the received gradients as $\mathbf{G}_d^c[j]$and updates the central factor matrix as:

$$\mathbf{U}_d^c[j] = \mathbf{U}_d^c[j-1] + \gamma \mathbf{G}_d^c[j] \tag{3.11}$$

and broadcast $\mathbf{U}_d^c[j]$ to each client.

Each client updates the factor tensor with the gradient received from the central

Algorithm 1 `FedTefid`

```
Require: Mask tensor $\mathcal{M}$, randomized sampled mode sequences $d_s[0], d_s[1], ..., d_s[J]$, initialized $\mathcal{U}$, and client(k).
 1: for $j = 0, 1, ..., J$ do
 2:     On each client(k) where $k \in K$
 3:     for $d = 1, 2, ..., D$ do
 4:         if $d = d_s[j]$ then
 5:             Each client compute stochastic gradient with observed data only (eq. 3.6, 3.8);
 6:             Conduct error feedback (eq. 3.9, 3.10) and gradient compression steps, and send
                the compressed gradients to server;
 7:             Server updates the factor matrix with the averaged compressed gradients from
                all clients, and broadcast the updated factor matrix to each client(eq. 3.11);
 8:             Clients update local factors $\mathbf{G}$ (eq. 3.12);
 9:         else if $d \neq d_s[j]$ then
10:             Unselected blocks are kept unchanged.
11:         end if
12:     end for
13: end for
```

server as:

$$\mathbf{U}_d^k = \mathbf{U}_d^c[j]. \tag{3.12}$$

Note that `FedTefid` preserves the communication efficiency of FedGTF-EF-PC as it reduces the communication cost both in the mode-level and element-level. Not only is only a single mode updated, but each partial gradient is compressed into the low-precision representation with errors fixed using the error-feedback mechanism. Moreover, `FedTefid` only communicates every few rounds, which further reduces the uplink communication cost. Algorithm 1 summarizes the key steps of `FedTefid`.

Chapter 4

Experiments

We evaluate our model and answer two important questions:

1. How does `FedTefid` compare with other state-of-the-art federated tensor factorization methods on two different datasets?

2. Does direct modeling of missing data entries and temporal smoothness constraints in `FedTefid` help performance?

To achieve these goals, for each experiment, we perform 3 runs with random initializations and summarize them with their means.

4.1 Datasets

We use two real-world, publicly available spatio-temporal datasets.

Chicago Crime A dataset that describes the crime reports in the city of Chicago, ranging from January 1st, 2001 to December 11th, 2017 [12]. We pre-process the data to obtain a 3rd order tensor of size $77 \times 207 \times 32$, representing the community, month, and type-of-crime factor correspondingly. There are 50.51% of non-zero entries in this tensor.

COVID-19 Open Data A dataset that describes the search trends on COVID-related symptoms from each county [8]. We use the data from 2020 and construct a 3rd order tensor of size $3035 \times 52 \times 422$ that represent the county, week, and search-terms respectively. This tensor has 44.01% non-zero entries.

4.2 Baselines

We compare `FedTefid` with the following three baselines:

- **FedGTF**: The FedGTF-EF-PC model run on the incomplete tensor with missing entries treated as 0.
- **FedGTFM**: `FedTefid` without temporal smoothing ($\lambda = 0$) where the partial gradient is calculated with observed entries.
- **LocTefid**: A localized version of `FedTefid` which does not communicate with the central server but models missing data and temporal smoothness on the temporal factor.

4.3 Hyper-parameter tuning

4.3.1 Rank

First, we determine the optimal rank for the Chicago Crime tensor and the COVID-19 tensor. We evaluate R from 2 to 30 and plot the objective without the regularization, see Eq. (3.2). We use 3 runs and summarize with their means to obtain the values here.

From fig. 4.1, we can observe that tensor decomposition for the Chicago Crime tensor with rank 2 gives the smallest loss, meaning that rank 2 is the optimal rank. However, the small rank (rank 2) may not yield interpretable results, especially to

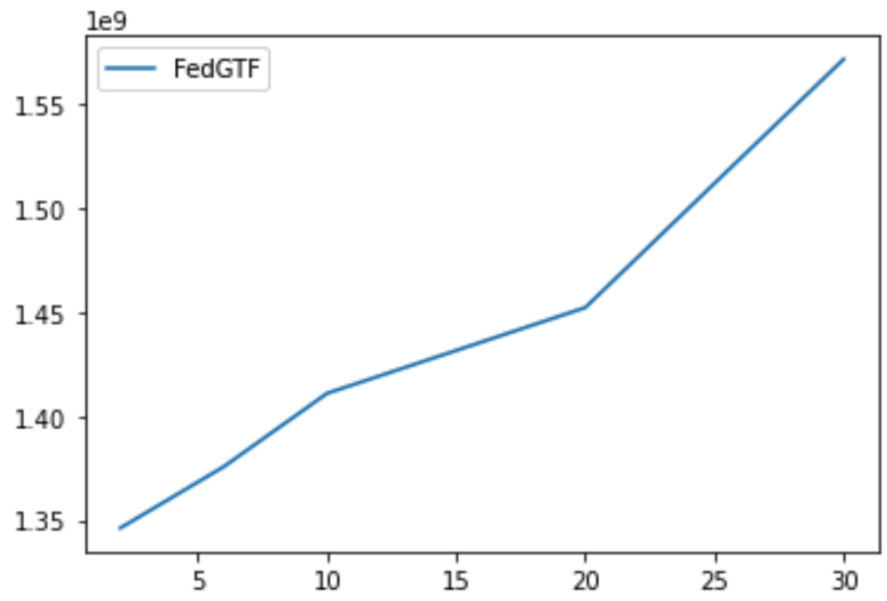

Figure 4.1: Comparison of various rank for Chicago Crime tensor decomposition. The x-axis represents the rank, the y-axis represents the loss.

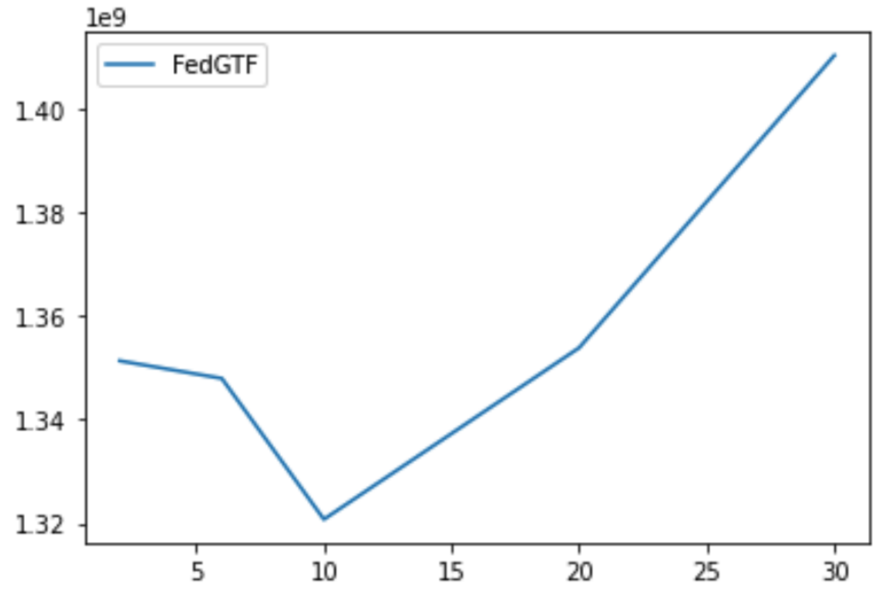

Figure 4.2: Comparison of various rank for COVID-19 tensor decomposition. The x-axis represents the rank, the y-axis represents the loss.

understand the patterns. Thus, we use a slightly larger rank ($R = 6$) which results in a small increase in loss for the rest of the experiments.

From fig. 4.2, we can observe that tensor decomposition for the COVID-19 open

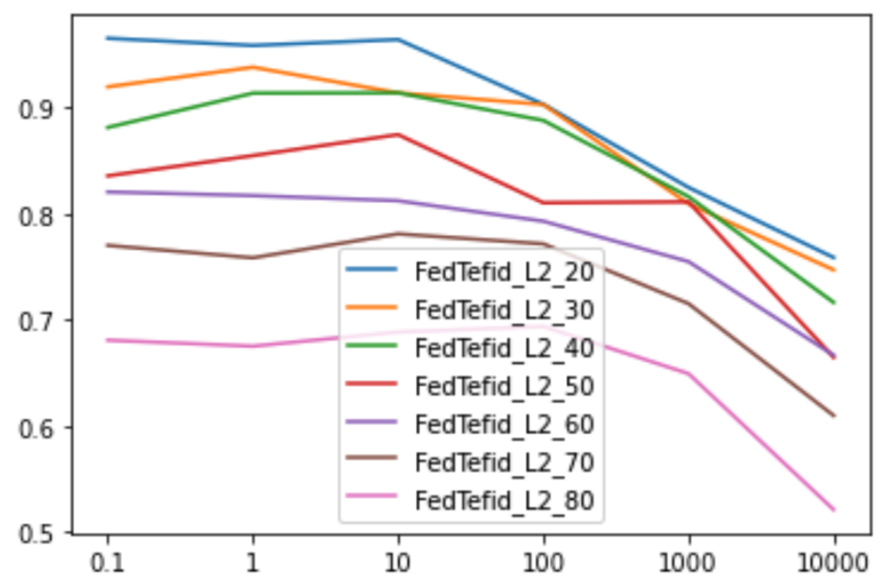

Figure 4.3: Comparison of various penalty λ on the Chicago Crime with different ratios of missing data. The x-axis represents the penalty λ, the y-axis represents the fms score between the resulting FedTefid decomposition and the decomposition without missing entries.

data with rank 10 gives the smallest rank, suggesting that rank 10 is the optimal rank. Thus, we use rank 10 for the case study on the COVID-19 open data.

4.3.2 Smoothness penalty

Then, we explore the impacts of the weights of the smoothness penalty and illustrate with an example of `FedTefid_ L2`. We use the Chicago Crime tensor to search appropriate penalty λ for `FedTefid` with L2 constraint; similar searches are perform for the `FedTefid_L1` and on the COVID-19 tensor.

We evaluate the performance based on the factor match score [7]. We use the factorized tensor learned by FedGTF-EF-PC without any missing data as the gold standard, and compute the factor match score between the four different models and the gold standard. If A and B are single component ktensors that have been normalized so that their weights are λ_a and λ_b, then the score is defined as $score = penalty * (a_1' * b_1) * (a_2' * b_2) * ... * (a_R' * b_R)$, where the $penalty$ is define by $penalty =$

$1 - \frac{|\lambda_a - \lambda_b)}{\max(\lambda_a, \lambda_b)}$. The factor match score quantifies how well the extracted factors match with the original ones, with 1 denoting a perfect match.

Fig. 4.3 summarizes the comparison of various penalty λ on the Chicago Crime with different ratios of missing data. We can observe that, for small missing ratio (less than 50%), small penalty weights like 1 gives better results, while for larger missing ratio (more than 50%), the larger penalty weight like 10 or even 100 gives better results. This illustrates the importance of tuning the regularization parameter appropriately based on the percentage of missing data. For the remainder of the experiments, I use optimal parameter λ depends on the missing ratio. For example, I use $\lambda = 1$ for experiments on the Chicago Crime tensor with 20% missing, and $\lambda = 10$ for that with 50% missing.

4.4 Uniformly missing data

Using the Chicago Crime tensor, we first analyze the performance of the models when data is uniformly missing at random across all the sites. We randomly partition the tensor into 8 different sites (each holds 12.25%) and use the same missing ratio across all the sites.

Fig. 4.4 summarizes the comparison between the various methods under the uniformly missing paradigm. We can observe that `FedTefid_L1`, `FedTefid_ L2` and FedGTFM significantly outperform FedGTF, illustrating the limitations of assuming missing entries are zero. FedGTFM performs reasonably well as it uses only the observed entries for stochastic gradient descent, which allows it to capture the latent information of the original tensor. We note that the temporal information helps `FedTefid` obtain better performance for larger amounts of missing data, as it utilizes the temporal smoothness property to recover the original temporal patterns.

Fig. 4.5 zooms in to compare the performance of `FedTefid_L1`, `FedTefid_ L2` and

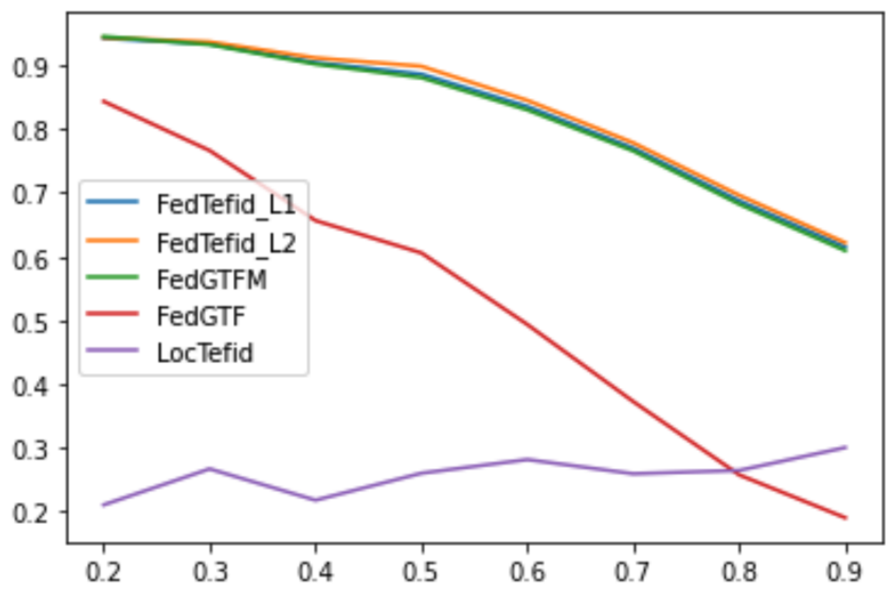

Figure 4.4: Comparison of the various methods on the Chicago Crime tensor with rank 6 on different ratios of missing data.

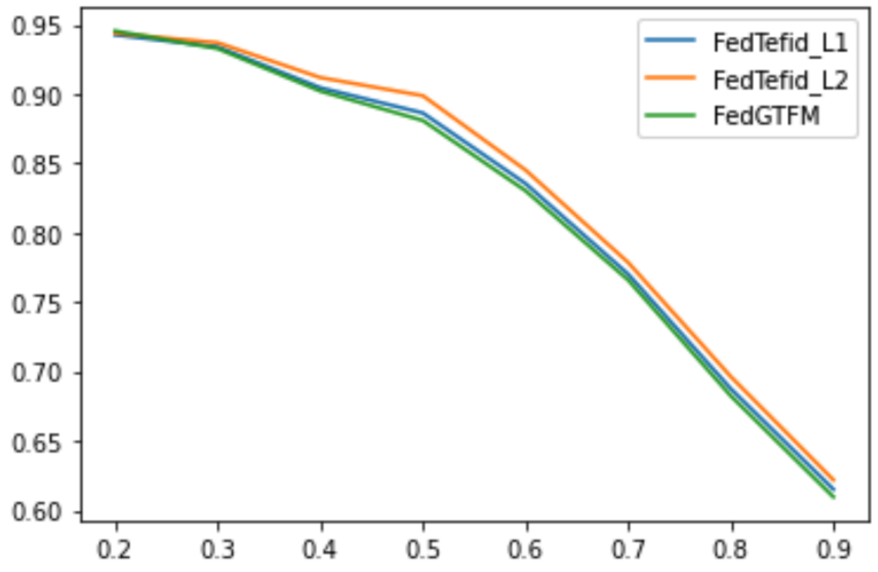

Figure 4.5: Zooming in to see the comparison of the various methods on the Chicago Crime with rank 6 on different ratios of missing data.

FedGTFM. We can observe that, FedGTFM slightly outperform `FedTefid_L1`, `FedTefid_L2` at missing ratio of 0.2. That's legitimate, as masking a small ratio of data leads to a increase in sparsity, which is equivalent to a random drop out on gradient, and that can help converge theoretically. For larger missing ratios, we can observe

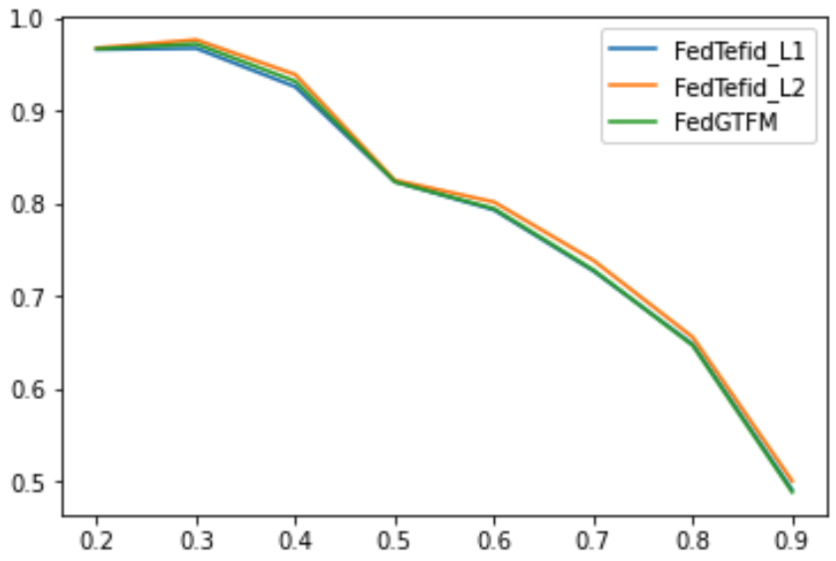

Figure 4.6: The comparison of the various methods on the Chicago Crime with rank 2 on different ratios of missing data.

that `FedTefid_ L2` performs the best, but `FedTefid_L1` also performs better than FedGTFM.

We also use the optimal rank (rank 2) to compare the the performance of `FedTefid_L1`, `FedTefid_ L2` and FedGTFM in fig 4.6.

4.5 Skewed missing data

Next, we assess the methods on the setting where each site has different ratios of missing data. For this setting, we split the Chicago Crime dataset amongst 5 different sites, where two of the sites had higher ratios of missing data (i.e., $> 70\%$). Tab. 4.1 presents the results of the four methods on four different settings of missing data. We can observe that `FedTefid`and FedGTFM are robust when different sites have different ratios of missing data and outperform both FedGTF and LocTefid. We also notice that there is a sizeable gap between the federated methods and the non-federated setting. This illustrates the importance of allowing different sites to share information with the central server. Moreover, it highlights that sites with more

Table 4.1: Performance on the methods on Chicago Crime data with skewed missing data across the five sites. The mask ratios across four settings are {10, 30, 50, 70, 90}, {10, 10, 50, 70, 90}, {10, 30, 40, 80, 90}, {30, 20, 30, 80, 90} for setting 1, 2, 3, and 4 respectively.

Setting	`FedTefid`	FedGTFM	FedGTF	LocTefid
1	**0.894**	0.872	0.523	0.280
2	**0.821**	0.802	0.549	0.391
3	**0.822**	0.821	0.607	0.304
4	**0.905**	0.881	0.643	0.261

missing data can leverage information from sites with less missing data to identify the latent factors.

4.6 Case study on COVID-19 Open Data

We illustrate `FedTefid` to identify temporal search trends using the COVID-19 Search Trends symptoms dataset. We split the original tensor into the 5 regions, West, South West, Mid-West, South East, North East, to mimic the scenario where each region collects the data locally and communicates with a central server (e.g., CDC) to monitor the trends of COVID-19 and its variants. This setting has several benefits as it can preserve the privacy of the county-level data as only the gradients are sent while also allowing data collection and processing to be localized and have different missing percentages as demonstrated in Tab. 4.1.

We use the optimal rank 10 for the CP decomposition for the COVID-19 tensor. Fig. 4.7 shows the extracted temporal trends of 10 topics using FedGTF-EF-PC with complete data. And Fig. 4.8 shows the extracted temporal trends of 10 topics using `FedTefid` on incomplete data with 50% missing entries. We can observe that `FedTefid` captures the trends well even with 50% data missing, and the trends in Fig. 4.8 seems to be more robust.

For illustrative purposes, we highlight the top 3 highest-weighted search term topics ($\mathbf{U}_3$) and the temporal trends. Fig. 4.9 showcases the weekly trends over the

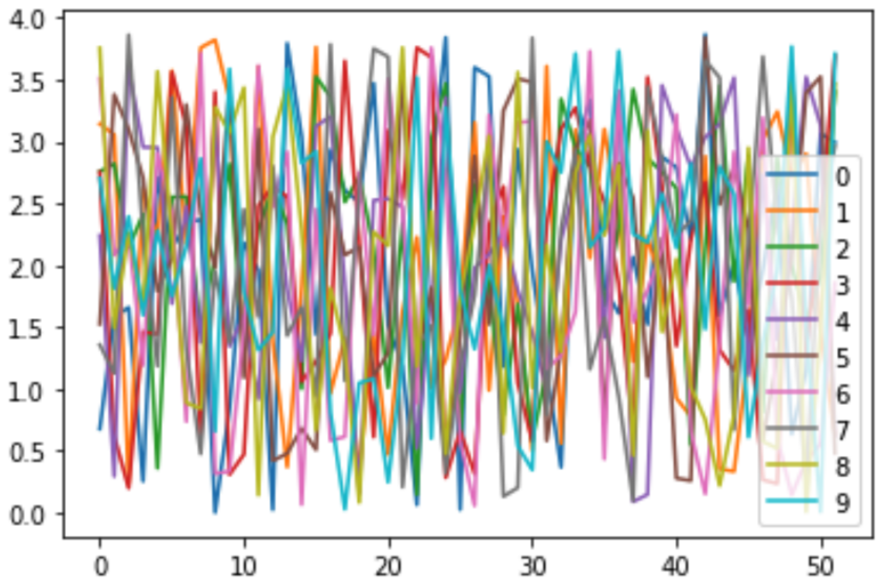

Figure 4.7: Temporal trends of the 10 extracted topics using FedGTF-EF-PC on complete data. The x-axis represents the timestamps (weeks in 2020) and the y-axis represents the value of the topics.

last 22 weeks in 2020 (July 20th to December 31st). For example, we can see that the three topics have similar trends from week 30 to week 35, then the orange topic (i.e., pain, low back pain, cough) has the opposite temporal trend as the green topic (i.e., abdominal obesity, lightheadedness, developmental disability).

Besides temporal trends of the topics, we can also explore the relationships between terms and extracted topics as in Fig. 4.10, and between counties and extracted topics as in Fig. 4.11.

We are also able to find similar counties with `FedTefid`. For example, we can normalize the county factor matrix ($\mathbf{U}_{(1)}$) of the 10 extracted topics and calculate the closest counties to Union County in Georgia using Euclidean distance. We find that the following counties share similar trends on COVID-related symptoms: Benton (WA), Wyandot (OH), Santa Barbara (CA), Kittitas (WA), Cumberland (ME), Miami (OH), Lubbock (TX), McPherson (KS), Lancaster (NE), and Adams (MS). Researchers can then use these similar counties to further analyze the factors that

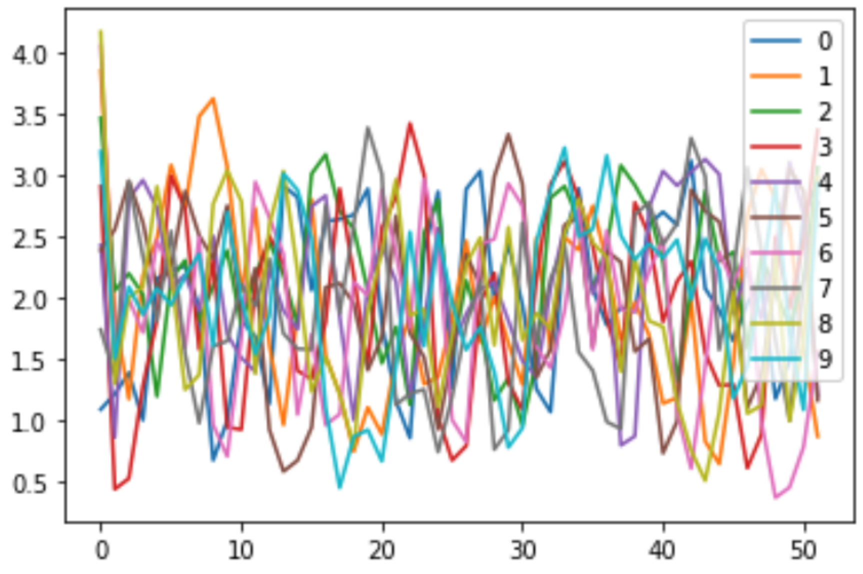

Figure 4.8: Temporal trends of the 10 extracted topics using FedTefid on incomplete data with 50% missing entries. The x-axis represents the timestamps (weeks in 2020) and the y-axis represents the value of the topics.

influence the spreading of COVID-19 within communities and the variants with different symptoms.

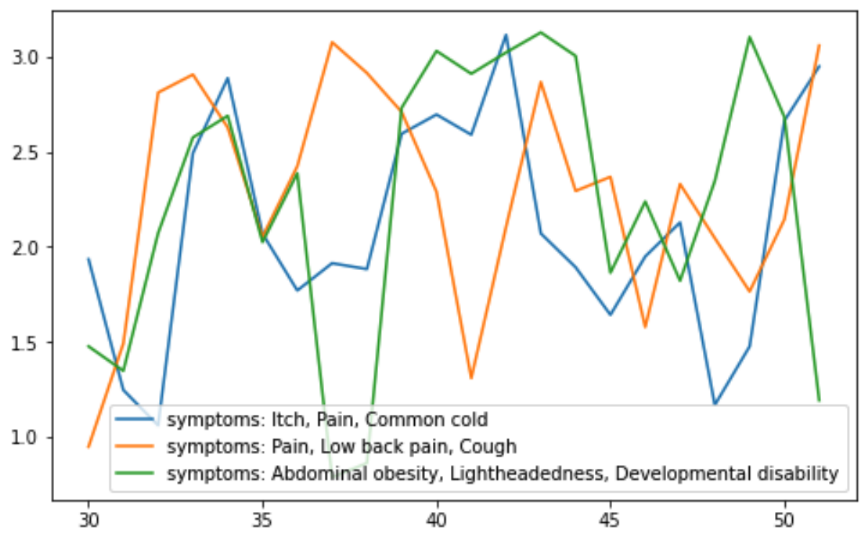

Figure 4.9: Temporal trends of top 3 extracted topics. The x-axis represents the timestamps (week 30 - 52 in year 2020) and the y-axis represents the value of the topics. The legends show the top 3 most frequent search terms for each topic.

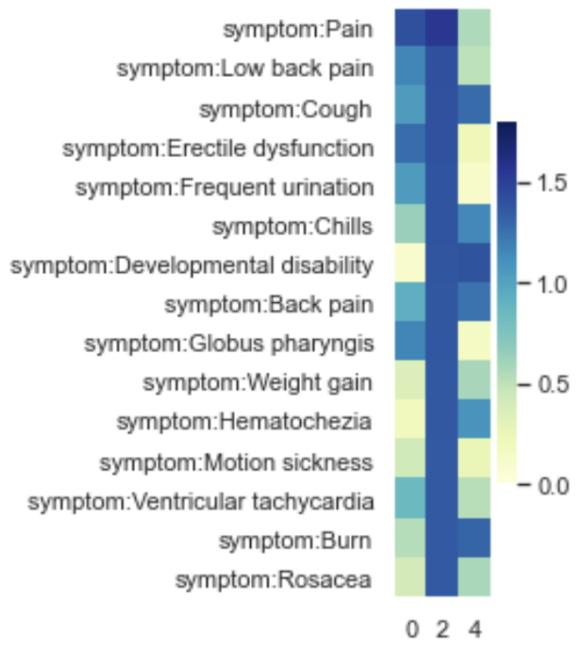

Figure 4.10: The heatmap of top 3 extracted topics. The x-axis represents the top 3 extracted topics (0 as Itch, Pain, Common cold; 2 as Pain, Low back pain, Cough; 4 as Abdominal obesity, Lightheadedness, Developmental disability) and the y-axis represents the terms that weighted the most in topic 2. The colors show the weights of terms on each topic.

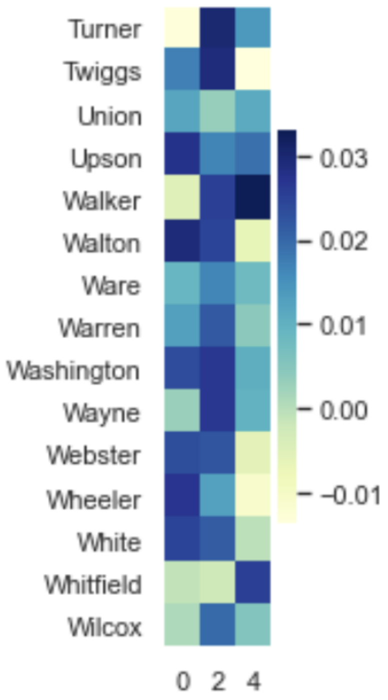

Figure 4.11: The heatmap of some Georgia counties. The x-axis represents the top 3 extracted topics (0 as Itch, Pain, Common cold; 2 as Pain, Low back pain, Cough; 4 as Abdominal obesity, Lightheadedness, Developmental disability) and the y-axis represents some Georgia counties. The colors show the weights of counties on each topic.

Chapter 5

Conclusion

In this paper, we study the missing data problem in federated tensor factorization. We propose a communication-efficient federated tensor factorization for incomplete data, `FedTefid`, which builds on the framework of FedGTF-EF-PC and extends it to model only the observed entries and incorporate a temporal smoothness constraint. We demonstrate the benefits of using `FedTefid` on uniformly missing data and data with skewed missing ratio, and we demonstrate a sample use case of `FedTefid` with the COVID-19 Open Data.

There are some future directions to be consider after this study. One is to explore more about the smoothness constraints. For example, we can consider using different weights, except 1 and -1, for the smoothness constraint matrix. For example, we can try run ARIMA model on the dataset and use the resulting coefficients for the smoothness constraint matrix[10]. Besides that, it's worthy to explore how to extend the methods for data types other than spatio-temporal tensors.

Bibliography

[1] Evrim Acar, Daniel M. Dunlavy, Tamara G. Kolda, and Morten Mørup. Scalable tensor factorizations for incomplete data. *Chemometrics and Intelligent Laboratory Systems*, 106(1):41–56, Mar 2011. ISSN 0169-7439. doi: 10.1016/j.chemolab.2010.08.004. URL `http://dx.doi.org/10.1016/j.chemolab.2010.08.004`.

[2] Dawon Ahn, Jun-Gi Jang, and U Kang. Time-aware tensor decomposition for missing entry prediction, 2020.

[3] Casey Battaglino, Grey Ballard, and Tamara G. Kolda. A practical randomized cp tensor decomposition. *SIAM J. Matrix Anal. Appl.*, 39:876–901, 2018.

[4] David Hong, Tamara G. Kolda, and Jed A. Duersch. Generalized canonical polyadic tensor decomposition. *SIAM Review*, 62(1):133–163, 2020. doi: 10.1137/18M1203626. URL `https://doi.org/10.1137/18M1203626`.

[5] Peter Kairouz, H. Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, Rafael G. L. D'Oliveira, Hubert Eichner, Salim El Rouayheb, David Evans, Josh Gardner, Zachary Garrett, Adrià Gascón, Badih Ghazi, Phillip B. Gibbons, Marco Gruteser, Zaid Harchaoui, Chaoyang He, Lie He, Zhouyuan Huo, Ben Hutchinson, Justin Hsu, Martin Jaggi, Tara Javidi, Gauri Joshi, Mikhail Khodak, Jakub Konečný, Aleksandra Korolova, Farinaz Koushanfar, Sanmi Koyejo, Tancrède Lepoint, Yang Liu, Prateek Mittal,

Mehryar Mohri, Richard Nock, Ayfer Özgür, Rasmus Pagh, Mariana Raykova, Hang Qi, Daniel Ramage, Ramesh Raskar, Dawn Song, Weikang Song, Sebastian U. Stich, Ziteng Sun, Ananda Theertha Suresh, Florian Tramèr, Praneeth Vepakomma, Jianyu Wang, Li Xiong, Zheng Xu, Qiang Yang, Felix X. Yu, Han Yu, and Sen Zhao. Advances and open problems in federated learning, 2021.

[6] Yejin Kim, Jimeng Sun, Hwanjo Yu, and Xiaoqian Jiang. Federated tensor factorization for computational phenotyping. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 887–895, 2017.

[7] Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3):455–500, 2009.

[8] Google LLC. Google covid-19 search trends symptoms dataset. `http://goo.gle/covid19symptomdataset`. Accessed: ¡2022-2-11¿.

[9] Jing Ma, Qiuchen Zhang, Jian Lou, Joyce C. Ho, Li Xiong, and Xiaoqian Jiang. Privacy-preserving tensor factorization for collaborative health data analysis. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, page 1291–1300, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450369763. doi: 10.1145/3357384.3357878. URL `https://doi.org/10.1145/3357384.3357878`.

[10] Jing Ma, Qiuchen Zhang, Joyce C. Ho, and Li Xiong. Spatio-temporal tensor sketching via adaptive sampling, 2020. URL `https://arxiv.org/abs/2006.11943`.

[11] Jing Ma, Qiuchen Zhang, Jian Lou, Li Xiong, and Joyce C Ho. Communication efficient federated generalized tensor factorization for collaborative health data analytics. In *Proceedings of the Web Conference 2021*, pages 171–182, 2021.

[12] Shaden Smith, Jee W. Choi, Jiajia Li, Richard Vuduc, Jongsoo Park, Xing Liu, and George Karypis. FROSTT: The formidable repository of open sparse tensors and tools, 2017. URL `http://frostt.io/`.

[13] Qingquan Song, Hancheng Ge, James Caverlee, and Xia Hu. Tensor completion algorithms in big data analytics. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 13(1):1–48, 2019.

[14] Jia Yang, Cai Fu, Xiao-Yang Liu, and Anwar Walid. Recommendations in smart devices using federated tensor learning. *IEEE Internet of Things Journal*, pages 1–1, 2021. doi: 10.1109/JIOT.2021.3116505.

[15] Tatsuya Yokota, Qibin Zhao, and Andrzej Cichocki. Smooth parafac decomposition for tensor completion. *IEEE Transactions on Signal Processing*, 64(20): 5423–5436, 2016.