IISE Transactions

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/uiie21

Federated data analytics: A study on linear models

Xubo Yue, Raed Al Kontar & Ana María Estrada Gómez

To cite this article: Xubo Yue, Raed Al Kontar & Ana María Estrada Gómez (2023): Federated data analytics: A study on linear models, IISE Transactions, DOI: 10.1080/24725854.2022.2157912

To link to this article: https://doi.org/10.1080/24725854.2022.2157912

IISE TRANSACTIONS
https://doi.org/10.1080/24725854.2022.2157912

Federated data analytics: A study on linear models

Xubo Yue[a], Raed Al Kontar[a], and Ana María Estrada Gómez[b]

[a]Industrial & Operations Engineering, University of Michigan, Ann Arbor, MI, USA; [b]School of Industrial Engineering, Purdue University, West Lafayette, IN, USA

ABSTRACT

As edge devices become increasingly powerful, data analytics are gradually moving from a centralized to a decentralized regime where edge computing resources are exploited to process more of the data locally. This regime of analytics is coined as Federated Data Analytics (FDA). Despite the recent success stories of FDA, most literature focuses exclusively on deep neural networks. In this work, we take a step back to develop an FDA treatment for one of the most fundamental statistical models: linear regression. Our treatment is built upon hierarchical modeling that allows borrowing strength across multiple groups. To this end, we propose two federated hierarchical model structures that provide a shared representation across devices to facilitate information sharing. Notably, our proposed frameworks are capable of providing uncertainty quantification, variable selection, hypothesis testing, and fast adaptation to new unseen data. We validate our methods on a range of real-life applications, including condition monitoring for aircraft engines. The results show that our FDA treatment for linear models can serve as a competing benchmark model for the future development of federated algorithms.

ARTICLE HISTORY
Received 13 June 2022
Accepted 6 December 2022

KEYWORDS
Federated data analytics; linear models; hierarchical model; uncertainty quantification; variable selection; hypothesis testing; fast adaptation; engineering applications

1. Introduction

The sheer amount of data collected nowadays is beginning to overwhelm traditional centralized data analytics regimes where data from the edge is continuously uploaded to a central server to be processed. Excessive communication traffic from data upload, significant central server storage needs, energy expenditures from centralized learning of big data models, and privacy concerns from sharing raw data are becoming critical challenges in centralized systems. Statista, a German company specializing in market and consumer data, has predicted that, by 2024, data produced on edge devices (e.g., cell phone data, self-driving vehicle data) will reach more than hundreds of zettabytes while the global central servers only have 10.4 zettabytes of storage (Morell and Alba, 2022). Transmitting such a vast amount of edge data into a central server is infeasible. Adding to that, training a model with moderately large datasets results in significant budget costs and carbon emissions (Patterson *et al.*, 2021). Furthermore, data-sharing comes with serious privacy concerns. According to Lawson *et al.* (2015), Canadian drivers who refused to enroll in the automotive telematics program demanded that their personal driving data (e.g., behavior, location, web-browsing history) should be respected by vehicle companies and that they be given control over the data collection process. These debates over data protection standards have not faded away over the past decade.

Fortunately, the Internet of Things (IoT) is undergoing a new revolution in which the computing power of edge devices is tremendously increasing (Hassan *et al.*, 2018). AI Chips such as general-purpose chips (GPUs), semi-customized chips (FGPAs), and fully-customized chips (ASICs) are becoming readily available across many applications (Blanco-Filgueira *et al.*, 2019; Rahman and Hossain, 2021; Zhu *et al.*, 2021). Such AI chips are able to process a vast amount of data locally and provide timely responses and decisions (Shi *et al.*, 2016). For instance, the autonomous vehicle company PerceptIn has released a real-time edge computing system, DragonFly+, that is three times more power efficient and delivers three to five times of the computing power of an Nvidia Tx1 and an Intel Core i7 processor (Liu *et al.*, 2019). Another notable example is Tesla's autopilot system that has computing power on the car itself comparable to hundreds of MacBook pros (CleanTechnica, 2021). As a consequence, the traditional IoT is on the verge of shifting to a decentralized framework recently termed the Internet of Federated Things (IoFT) (Kontar *et al.*, 2021) in which some of the data processing is deferred to the edge. In this future, the central server only acts as an orchestrator of the learning process and an integration point of model updates from different devices, rather than the central location where all data is processed. Indeed, IoFT is slowly infiltrating various fields such as manufacturing, transportation, and energy systems (Kontar *et al.*, 2021).

The underlying data analytics framework in IoFT is Federated Data Analytics (FDA), where edge devices exploit their own computation power to collaboratively extract knowledge and build smart analytics while keeping their personal data stored locally. Consequently, edge devices no

CONTACT Raed Al Kontar alkontar@umich.edu
Supplemental data for this article is available online at https://doi.org/10.1080/24725854.2022.2157912

longer need to upload their data to the cloud (or server), and, in turn, the cloud does not need to store that immense amount of data. As such, FDA resolves many of the aforementioned drawbacks of the centralized computing system and sets forth many intrinsic advantages, including privacy-preserving and reducing storage/computation/communication costs, among many others.

In spite of some recent advances in FDA, most, if not all, literature focuses on deep neural networks (Li, Sahu, Talwalkar, *et al.*, 2020) (learned via first-order methods). To date, very few papers have delivered federated treatments of traditional statistical models. Perhaps the closest field where statistical models were investigated is Distributed Learning (DL) (Jordan *et al.*, 2019), yet DL and FDA have several fundamental differences. Despite the terminology "distributed", DL is still a centralized computation approach where different computing nodes operate on all data (Fan *et al.*, 2021). These nodes communicate often, observe each other's data, and can operate on different data partitions. The underlying philosophy for DL is "divide-and-conquer" where data is divided across the nodes (often dynamically), and then the nodes collaborate to "conquer" (learn) a single model. As a notable example, Zhang *et al.* (2015) propose a DL algorithm that solves a ridge regression problem. The basic idea of this paper is as follows: a server first evenly divides a set of data into m disjoint sets and assigns each set to a different node. Each node then solves a ridge regression problem and sends the optimal solutions back to the server. The server then aggregates local estimations. As a result, this approach returns a single global model. In contrast, in FDA, data resides at the edge and cannot be shuffled, randomized, or divided. Therefore, edge devices cannot see each others' data and data partitions for FDA are fixed and often heterogeneous. In addition, devices have datasets with unique features as they correspond to different clients, components or systems (e.g., cars). Therefore, in FDA we cannot divide the data and there is often no single model to conquer, rather, our goal is to borrow strength across edge devices to improve inference and prediction.

In this work, we take a step back and move out of the regime of deep neural networks to study one of the most fundamental statistical models: Linear Regression (LR). Indeed, linear models may facilitate hypothesis testing, uncertainty quantification, variable selection, deriving engineering insight, and establishing a baseline with which to compare other models. Needless to say, in reality, many real applications can be sufficiently characterized by linear models (Liu *et al.*, 2013; Si *et al.*, 2017; Li *et al.*, 2018; Schulz *et al.*, 2020; Arashi *et al.*, 2021; Şahin *et al.*, 2022). In addition, building upon FDA for linear models, one may develop approaches for more complex derivatives such as logistic regression, mixed-effects, and kernel methods.

To this end, we exploit the properties and structure of linear models and develop an FDA treatment for LR with Gaussian noise, entitled `FedLin`. Our treatment is built upon hierarchical models (HMs), which allow borrowing statistical knowledge across groups (i.e., devices or clients in FDA). Specifically, we propose two federated HM structures that provide a shared representation across devices to facilitate information transfer. The first structure establishes a shared representation defined through a structural prior over concatenated device parameters. The second structure is based on the assumption that all device parameters are generated from the same underlying distribution. This allows uncertainty quantification and, consequently, a Bayesian treatment for variable selection in `FedLin`. Our methods are validated on a range of real-life problems, including variable selection and condition monitoring. The results highlight the effective performance of our approaches and their ease of implementation, which may help them serve as benchmark models for many future developments of federated statistical algorithms.

Organization: The remainder of this article is organized as follows. In Section 2, we conduct a literature review, and introduce the general setting and motivation. In Sections 3 and 4, we present our two model structures and their applications. We validate our proposed models on various simulated and real-life datasets in Sections 5 and 6. Finally, we draw conclusions in Section 7. The, codes used in this article, written in R language, can be found at https://github.com/UMDataScienceLab/Federated-Linear-Models.

2. Background

2.1. Literature overview

The idea of FDA was first brought to the forefront of deep learning by McMahan *et al.* (2017). In this work, they proposed an FDA algorithm termed federated averaging (`FedAvg`). The idea of `FedAvg` is simple: a central server distributes initial deep learning model parameters and the network structure to some selected devices, devices perform local Stochastic Gradient Descent (SGD) steps using their data and send their updated parameters back. The server then takes an average of those parameters to update the global model. This process is termed as one communication round and is iterated several times. Although simple, `FedAvg` is currently still one of the most competitive benchmark models (Kairouz *et al.*, 2021). To date, some work has been proposed to improve the performance of federated deep learning algorithms. For instance, Yuan and Ma (2020) and Liu *et al.* (2020) and Yuan and Ma (2020) provided several provable techniques to accelerate `FedAvg` and enable faster convergence. Li, Sanjabi, *et al.* (2019); Du *et al.*, (2021); Yu *et al.* (2020); Yue *et al.* (2021); Du *et al.* (2021) developed variants of `FedAvg` that ensure uniformly good performance across all devices to achieve fairness. Another line of work aims to develop personalized solutions in FDA as excessive heterogeneity can greatly impact the performance of a single global model (Deng *et al.*, 2020; Fallah *et al.*, 2020; Li *et al.*, 2021). Such approaches usually either follow a train-then-personalize philosophy where a trained global model is fine-tuned on local devices or divide the layers of a neural network into shared and individualized ones (Tan *et al.*, 2022), where devices collaborate to learn the common layers using methods such as `FedAvg`. From a theoretical perspective, Stich (2018); and Li, Sahu, Zaheer, *et al.* (2020) prove the convergence of `FedAvg` for convex

functions and homogeneous independent and identically distributed (*i.i.d.*) datasets. Those results are then extended to a non-convex setting by Wang and Joshi (2021). On the other hand, Li, Huang, *et al.* (2019) extend the results of Stich (2018) to the non-*i.i.d.* setting. Furthermore, Shi *et al.* (2021) extend the convergence results to a kernel regime. For a comprehensive overview of the current literature, please refer to Kontar *et al.* (2021).

The major trend of FDA exclusively focuses on neural networks and classification tasks. FDA for statistical models is still scarce. Yue and Kontar (2021) extend the Gaussian process to a federated framework and show that their proposed algorithm can achieve state-of-the-art performance on multi-fidelity modeling problems. Yuan *et al.* (2021) develop a federated composite optimization framework that solves the federated Lasso problem. Tong *et al.* (2020) propose a federated iterative hard thresholding algorithm to tackle the non-convex 0-norm penalized regression problem. The two aforementioned papers mainly formulate penalized regression from a frequentist perspective. In Section 4, we will develop a Bayesian formulation built upon HM for federated penalized regression.

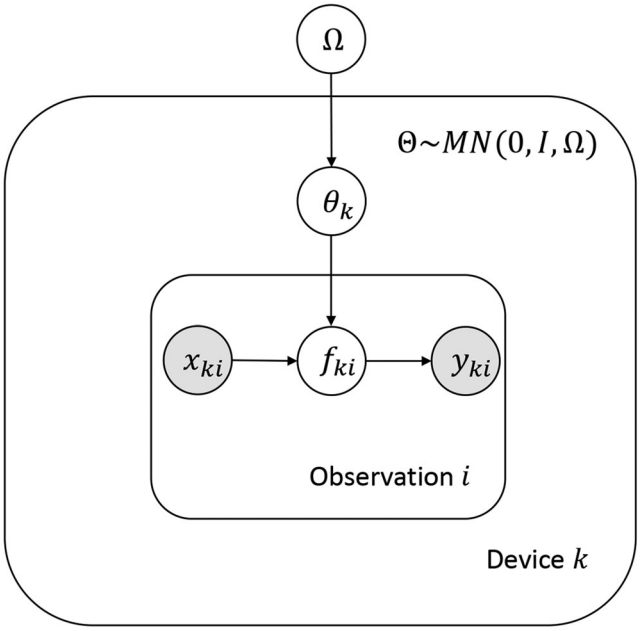

Figure 1. The first HM structure.

2.2. General setting

We start by describing our problem setting. Suppose there exists $K \geq 2$ edge devices. For device $k \in [K] := \{1, ..., K\}$, the dataset is given as $\boldsymbol{D}_k = \{\boldsymbol{X}_k, \boldsymbol{Y}_k\}$ with N_k observations, where $\boldsymbol{Y}_k = [y_{k1}, ..., y_{kN_k}]^\top$ is a $N_k \times 1$ output vector, $\boldsymbol{X}_k = [\boldsymbol{x}_{k1}, ..., \boldsymbol{x}_{kN_k}]$ is a $d \times N_k$ input matrix and $\boldsymbol{x_{ki}} = ([\boldsymbol{x_{ki}}]_1, ..., [\boldsymbol{x_{ki}}]_d)^\top \forall i = 1, ..., N_k$. Here, d is the dimensionality of the input space. In this work, we focus on linear models. More specifically, data on device k is used to learn a linear model parameterized by $\boldsymbol{\theta}_k \in \mathbb{R}^d$. The distribution of y_{ki} is given as

$$y_{ki}|\boldsymbol{x}_{ki}, \boldsymbol{\theta}_k \sim \mathcal{N}(\boldsymbol{x_{ki}}^\top \boldsymbol{\theta}_k, \sigma_k^2), \quad \forall i = 1, ..., N_k, \tag{1}$$

where σ_k^2 is a noise parameter. For the sake of compactness, denote by $\boldsymbol{\Theta} = (\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_K)$ a $d \times K$ matrix concatenating all device parameters.

Furthermore, we assume that a central server is connected to all edge devices and can facilitate the collaborative model learning process. As such, our goal in FDA is to let devices leverage their commonalities to better learn model parameters $\boldsymbol{\Theta}$; all the while distributing the learning efforts and circumventing the need to share raw data.

2.3. FDA and HMs

Since our goal is to borrow strength across devices, the first step is to create a shared representation across individual device models in order to facilitate the inductive transfer of knowledge. Here we adopt the natural hierarchy in FDA where a central server is connected to edge devices and can orchestrate the learning process. Specifically, we assume that individual device parameters $\boldsymbol{\theta}_k$ at the lower hierarchical level are generated from a set of shared parameters at the higher hierarchical level. Through collaboratively learning these shared parameters in a federated manner, devices induce an update on their personalized parameters $\boldsymbol{\theta}_k$ that uses information from all other devices.

Two hierarchical structures are proposed. The first defines a joint prior over $\boldsymbol{\Theta}$ parameterized by a cross-covariance matrix $\boldsymbol{\Omega}$. This allows learning a graph that achieves inductive transfer. Whereas, the second HM structure assumes that the $\boldsymbol{\theta}_k$'s are sampled from a common distribution (e.g., $\boldsymbol{\theta}_k|\boldsymbol{\phi} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$). This allows a Bayesian treatment capable of uncertainty quantification as well as learning a global random variable $\boldsymbol{\phi}$ that can be used to predict on new unseen devices.

We will detail our model formulations, inferences, and applications in the following two sections.

3. A shared representation via correlation

In this section, we present our first hierarchical structure (denoted as `HM1`) that establishes a shared representation by defining a structural prior over $\boldsymbol{\Theta}$ (Figure 1). This prior is parameterized by $\boldsymbol{\Omega}$ - a $K \times K$ cross-covariance matrix. In Figure 1, the matrix $\boldsymbol{\Omega}$ acts as a graph on the central server that encodes a shared representation among the K devices and facilitates information sharing. By learning and exploiting the matrix $\boldsymbol{\Omega}$, devices can borrow information from each other to improve prediction performance.

Mathematically, we impose a structural prior $\mathcal{N}(\boldsymbol{0}, \boldsymbol{\Omega} \otimes \boldsymbol{I})$ on $\text{Vec}(\boldsymbol{\Theta})$, where $\text{Vec}(\cdot)$ is a vectorization operation and $\otimes$ is a Kronecker product. This prior encodes the belief on the underlying distribution that generates components of $\boldsymbol{\Theta}$. More specifically, $\boldsymbol{\Omega}$ is a symmetric matrix whose (i, j)-th component captures the covariance between device i and j. Overall, the aforementioned description translates to the following formulation:

$$\boldsymbol{\Theta} \sim \mathcal{MN}(\boldsymbol{0}, \boldsymbol{I}, \boldsymbol{\Omega}), \tag{2}$$

where $\mathcal{MN}(\boldsymbol{M}, \boldsymbol{A}, \boldsymbol{B})$ denotes a matrix normal distribution with location (mean) parameter $\boldsymbol{M}$, row covariance $\boldsymbol{A}$, and column

covariance $\boldsymbol{B}$. In this prior, the column covariance $\boldsymbol{\Omega}$ captures the covariance across devices. Our prior assumes that the covariance across devices is the same for different parameter components. In fact, this is a common practice in the multitask learning literature (Zhang and Yeung, 2012; Ruder, 2017) for multiple reasons: (i) The goal here is to facilitate information transfer between devices and $\boldsymbol{\Omega}$ achieves exactly this goal. (ii) This is only a prior and, in reality, it is hard to pre-define the within component covariance. (iii) Posterior computations become rather challenging if the prior (basically a regularization) is complex. As we will show in Sections 5-6, a single $\boldsymbol{\Omega}$ is capable of providing excellent performance in several prediction tasks.

By Bayes' rule and incorporating (1) and (2), we can obtain the posterior distribution of $\boldsymbol{\Theta}$ as a product of the prior and the likelihood function. By omitting the constant terms and taking the negative logarithm, we can obtain the negative log-likelihood function:

$$\begin{aligned} -\log p(\boldsymbol{\Theta}|\boldsymbol{\Omega}, \{\boldsymbol{Y}_k\}_{k=1}^K) &\propto -\log p(\{\boldsymbol{Y}_k\}_{k=1}^K|\boldsymbol{\Theta})p(\boldsymbol{\Theta}|\boldsymbol{\Omega}) \\ &= -\log\left(\prod_{k=1}^K p(\boldsymbol{Y}_k|\boldsymbol{\theta}_k)\right) - \log p(\boldsymbol{\Theta}|\boldsymbol{\Omega}) \\ &\propto \sum_{k=1}^K \frac{1}{\sigma_k^2}\|\boldsymbol{Y}_k - \boldsymbol{X}_k^\top\boldsymbol{\theta}_k\|_2^2 + \mathrm{Tr}(\boldsymbol{\Theta}\boldsymbol{\Omega}^{-1}\boldsymbol{\Theta}^\top) \\ &\quad + d\log|\boldsymbol{\Omega}| := L(\boldsymbol{\Theta}, \boldsymbol{\Omega}). \end{aligned}$$

Therefore, our goal is to find the Maximum A Posteriori (MAP) of $(\boldsymbol{\Theta}, \boldsymbol{\Omega})$ that minimizes the negative log-likelihood function, in a federated fashion:

$$(\boldsymbol{\Theta}^*, \boldsymbol{\Omega}^*) = \arg\min_{\boldsymbol{\Theta}, \boldsymbol{\Omega}} L(\boldsymbol{\Theta}, \boldsymbol{\Omega}).$$

To solve this, notice that the derivative with respect to $\boldsymbol{\theta}_k$, for all k, is

$$\frac{\partial L(\boldsymbol{\Theta}, \boldsymbol{\Omega})}{\partial \boldsymbol{\theta}_k} = \frac{-2}{\sigma_k^2}\boldsymbol{X}_k(\boldsymbol{Y}_k - \boldsymbol{X}_k^\top\boldsymbol{\theta}_k) + 2\sum_{i=1}^K \boldsymbol{\theta}_i\boldsymbol{\Omega}_{i,k}^{-1},$$

where $\boldsymbol{\Omega}_{i,k}^{-1}$ is defined as the component located in the ith row and kth column of $\boldsymbol{\Omega}^{-1}$.

In IoFT, the central server does not have access to datasets $\boldsymbol{D} = \{\boldsymbol{D}_1, ..., \boldsymbol{D}_K\}$, nor do devices have access to each other's datasets. Furthermore, the central server cannot share $\boldsymbol{\Omega}$ and $\boldsymbol{\Theta}$ with any device, due to the privacy constraint. Therefore, directly running gradient descent using $\frac{\partial L}{\partial \boldsymbol{\theta}_k}$ is not feasible. Yet, by scrutinizing $\frac{\partial L}{\partial \boldsymbol{\theta}_k}$, one can observe that a gradient update on each $\boldsymbol{\theta}_k$ is split into two gradients. The first term is an update from local data $\boldsymbol{D}_k$ whereas the second is a regularization term from all devices based on $\boldsymbol{\Omega}$. Therefore, the local parameter update can be done via the two local steps below

$$\text{Stage 1}: \text{ Multiple local GD or SGD steps } \boldsymbol{\theta}_k \leftarrow \boldsymbol{\theta}_k + 2\frac{\eta_1}{\sigma_k^2}\boldsymbol{X}_k(\boldsymbol{Y}_k - \boldsymbol{X}_k^\top\boldsymbol{\theta}_k).$$

$$\text{Stage 2}: \text{ Prior Shrinkage } \boldsymbol{\theta}_k \leftarrow \boldsymbol{\theta}_k - 2\eta_2\sum_{i=1}^K \boldsymbol{\theta}_i\boldsymbol{\Omega}_{i,k}^{-1}.$$

At the first stage, device k runs multiple steps of SGD or Gradient Descent (GD) using the local gradient information $\frac{-2}{\sigma_k^2}\boldsymbol{X}_k(\boldsymbol{Y}_k - \boldsymbol{X}_k^\top\boldsymbol{\theta}_k)$. To compute this gradient value, one needs to estimate the local variance parameter σ_k^2. Yet, recall that our main goal is to estimate $\boldsymbol{\theta}_k$ by borrowing information from the covariance matrix $\boldsymbol{\Omega}$. Adding to that, $\boldsymbol{\theta}_k$ and σ_k^2 are independent. Therefore, it is not necessary to estimate σ_k^2 at each local step. Here observe that η_1 is a tunable learning rate parameter and we can thus view $\eta_2 := \frac{\eta_1}{\sigma_k^2}$ as a tuning parameter in stage 1. In other words, we define η_2 as the tunable learning rate during the optimization procedure. This circumvents the need to estimate σ_k^2 locally. Nevertheless, σ_k^2 can be easily estimated from the linear residual term if it is of practitioner's interest. We here note that, since each device k has a different σ_k^2, it is possible to use device-specific learning rates η_{2k} for all k. However, it incurs a heavy tuning cost. In this article, we use the same learning rate for each device. As we will show in Section 5-6, this strategy works well.

At the second stage, device k will then use the aggregated information from all devices $\sum_{i=1}^K \boldsymbol{\theta}_i\boldsymbol{\Omega}_{i,k}^{-1}$ broadcasted from the central server to update $\boldsymbol{\theta}_k$ by exploiting the covariance matrix $\boldsymbol{\Omega}$. One key notable feature of this updating framework is that the central server only needs to share an aggregated metric $\sum_{i=1}^K \boldsymbol{\theta}_i\boldsymbol{\Omega}_{i,k}^{-1}$. This operation is indeed reminiscent of federated averaging and can preserve privacy while allowing devices to borrow strength from each other.

Finally, we will discuss the updating rule of $\boldsymbol{\Omega}$ on the central server. The most straightforward way is to take the derivative of $L(\boldsymbol{\Theta}, \boldsymbol{\Omega})$ with respect to $\boldsymbol{\Omega}$. By doing so, we obtain

$$\frac{\partial L(\boldsymbol{\Theta}, \boldsymbol{\Omega})}{\partial \boldsymbol{\Omega}} = -\boldsymbol{\Omega}^{-1}\boldsymbol{\Theta}^\top\boldsymbol{\Theta}\boldsymbol{\Omega}^{-1} + d\boldsymbol{\Omega}^{-1} = 0 \Rightarrow \boldsymbol{\Omega} = \frac{\boldsymbol{\Theta}^\top\boldsymbol{\Theta}}{d}.$$

As a result, it is natural to update $\boldsymbol{\Omega}$ using this closed-form expression. Here one can view that $\boldsymbol{\Theta}^\top\boldsymbol{\Theta}$ encodes the information of device covariance. Unfortunately, this approach typically faces singularity issues when $\boldsymbol{\Theta}^\top\boldsymbol{\Theta}$ is not a positive definite matrix (e.g., contains zero elements). To resolve this, we propose an updating procedure that prevents an abrupt change in $\boldsymbol{\Omega}$ to safeguard against singularity. More specifically, we express the updated $\boldsymbol{\Omega}$ as a convex combination between $\boldsymbol{\Omega}$ from the previous communication round and the exact updating equation $\frac{\boldsymbol{\Theta}^\top\boldsymbol{\Theta}}{d}$. That being said, we have

$$\boldsymbol{\Omega} \leftarrow (1-\alpha)\boldsymbol{\Omega} + \frac{\alpha}{d}\boldsymbol{\Theta}^\top\boldsymbol{\Theta}.$$

Here, α is a parameter that controls the change in $\boldsymbol{\Omega}$ and $\frac{\boldsymbol{\Theta}^\top\boldsymbol{\Theta}}{d}$ encodes the devices' covariance. A small α renders a conservative updating rule while a large α under-weights the importance of the covariance matrix from the previous communication round. When $\alpha = 1$, we recover the closed-form updating equation.

Here note that, in the aforementioned framework, the central server selects all devices at each communication round. This scheme is known as full device participation. In reality, however, some local devices are often offline or unwilling to respond due to various reasons. To accommodate this situation, one can sample a subset of devices at each communication round. We term

this scenario as partial device participation. We summarize the detailed algorithm in Algorithm 1.

Algorithm 1: Improving Device Performance by Exploiting Structural Covariance

Data: Number of devices K, number of local iterations T, set $\mathcal{S}$ that contains indices of the selected devices, number of communication rounds C, randomly initialized model parameter $\boldsymbol{\Theta}$, initial matrix $\boldsymbol{\Omega} = \boldsymbol{I}$, learning rate η_2 (selected by grid-search or other tuning methods), proportion $\alpha = 0.1$ (default), dimension d.

for $c = 0 : (C-1)$ **do**

Server broadcasts column of $\boldsymbol{\Theta}$ (i.e., $\boldsymbol{\theta}_k$) and $\sum_{i=1}^{K} \boldsymbol{\theta}_i \boldsymbol{\Omega}_{i,k}^{-1}$ for all selected k;

for $k \in \mathcal{S}$ **do**

for $t = 0 : (T-1)$ **do**

Device-side: (Sampling Batch) Sample a subset of data $(\boldsymbol{X}_k^b, \boldsymbol{Y}_k^b)$ from $\boldsymbol{D}_k$, where superscript b means batch;

Device-side: (SGD or GD) $\boldsymbol{\theta}_k^{(t+1)} = \boldsymbol{\theta}_k^{(t)} + 2\eta_2 \boldsymbol{X}_k^b (\boldsymbol{Y}_k^b - \boldsymbol{X}_k^{b\top} \boldsymbol{\theta}_k^{(t)})$;

end

Device-side: (Prior Shrinkage) $\boldsymbol{\theta}_k^{new} \leftarrow \boldsymbol{\theta}_k^{(T)} - 2\eta_2 \sum_{i=1}^{K} \boldsymbol{\theta}_i \boldsymbol{\Omega}_{i,k}^{-1}$;

end

Server-side: Combine all $\boldsymbol{\theta}_k$, for $k \notin \mathcal{S}$, and all $\boldsymbol{\theta}_k^{new}$, for $k \in \mathcal{S}$, to create a new matrix $\boldsymbol{\Theta}$;

Server-side: $\boldsymbol{\Omega} \leftarrow (1-\alpha)\boldsymbol{\Omega} + \frac{\alpha}{d} \boldsymbol{\Theta}^\top \boldsymbol{\Theta}$.

end

Return $\boldsymbol{\Theta}, \boldsymbol{\Omega}$.

As we will show in our numerical studies, this simple-to-implement algorithm requires very few communication rounds to recover the true parameters and excels at leveraging knowledge across all devices.

4. An HM based on the distribution assumption

So far, we have presented HM1, which exploits the relationship among devices to improve prediction performance. One drawback of Algorithm 1 is that it only returns a point estimate of $\boldsymbol{\Theta}$. In practice, it is also desirable to quantify the uncertainty in the parameter estimates. Additionally, the estimated $\boldsymbol{\Theta}$ and $\boldsymbol{\Omega}$ cannot provide any borrowable information for new devices, yet the idea of fast adaptation to new unseen data is crucial in many fields such as meta-learning (Vanschoren, 2019). In this section, we will present an alternative model structure that is formulated from a Bayesian perspective to tackle the aforementioned issues.

4.1. Structure and formulation

Our second structure (denoted as HM2) assumes all device parameters are generated from the same underlying distribution. To give a simple example, one can assume $\boldsymbol{\theta}_k | \boldsymbol{\phi} \sim \mathcal{N}(\boldsymbol{\mu}, \tau \boldsymbol{I})$ (Figure 2) where $\boldsymbol{\phi} = (\boldsymbol{\mu}, \tau)$ is a set of a global hyper-parameters

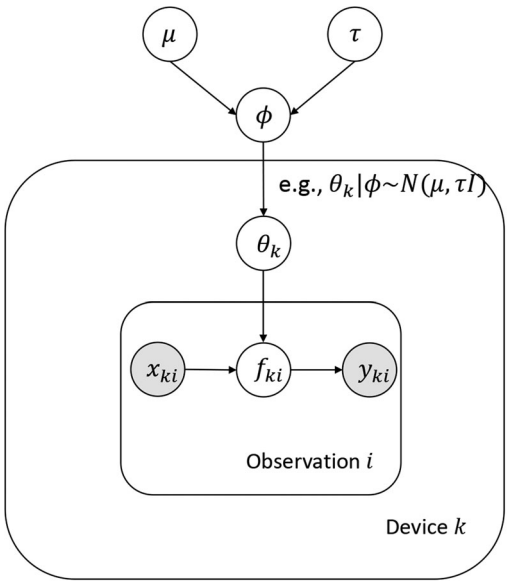

Figure 2. The second HM structure.

on the central server. Here it is critical to note that $\tau \boldsymbol{I} \in \mathbb{R}^{d \times d}$ is a within covariance matrix and does not denote covariances across K devices as all $\boldsymbol{\theta_k}$'s come from the same underlying distribution. This assignment indicates that $\{\boldsymbol{\theta}_k\}_{k=1}^K$ are related and generated from the same distribution, yet the degree of model similarity is controlled by the variance parameter τ. A small $|\tau|$ implies all model parameters are similar (i.e., homogeneous) and the hierarchical model is closely related to learning a single common parameter $\boldsymbol{\theta}$ that fits all devices' data. On the other hand, a large variance $|\tau|$ incurs more heterogeneity among devices. In the extreme case when $|\tau| \to \infty$, the hierarchical model is equivalent to a separate modeling approach where each device's data is fitted separately (Albert and Hu, 2019).

Now, we formally define the HM2 formulation. From our hierarchical definition and taking a fully Bayesian treatment by placing a prior $p(\boldsymbol{\phi})$ on the global hyper-parameters, the joint posterior of $\{\boldsymbol{\theta}_k\}_{k=1}^K$ and $\boldsymbol{\phi}$ for HM2 can be written as:

$$\begin{aligned} &p(\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_K, \boldsymbol{\phi} | \boldsymbol{Y}_1, ..., \boldsymbol{Y}_K) \\ &\quad \propto p(\boldsymbol{Y}_1, ..., \boldsymbol{Y}_K | \boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_K, \boldsymbol{\phi}) p(\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_K, \boldsymbol{\phi}) \\ &= p(\boldsymbol{\phi}) \prod_{k=1}^{K} p(\boldsymbol{Y}_k | \boldsymbol{\theta}_k, \boldsymbol{\phi}) p(\boldsymbol{\theta}_k | \boldsymbol{\phi}) \ . \end{aligned} \tag{3}$$

To further contextualize HM2, we provide an example of a possible formulation.

Example 4.1. *Assume each device k fits a linear regression parameterized by $\boldsymbol{\theta}_k$. The local dataset is given as $(\boldsymbol{X}_k, \boldsymbol{Y}_k)$ for all k. Then one possible hierarchical formulation is:*

$$\begin{aligned} \boldsymbol{Y}_k | \boldsymbol{\theta}_k, \boldsymbol{\phi} &\sim \mathcal{N}(\boldsymbol{X}_k^\top \boldsymbol{\theta}_k, \sigma_k^2 \boldsymbol{I}) \\ \boldsymbol{\theta}_k | \boldsymbol{\phi} &\sim \mathcal{N}(\boldsymbol{\mu}, \tau I) \\ \boldsymbol{\mu} &\sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \\ \tau &\sim \log \mathcal{N}(0, 1) \\ \boldsymbol{\phi} &= (\boldsymbol{\mu}, \tau). \end{aligned}$$

Clearly, inferring the joint posterior above is very challenging in a federated setting. Yet, in a hierarchical model, if we

know the posterior over the upper hierarchical level $p(\boldsymbol{\phi}|\{\boldsymbol{Y}_k\}_{k=1}^K)$ (i.e., over global hyper-parameters $\boldsymbol{\phi}$), we can directly use this posterior as a prior to infer the lower level $p(\boldsymbol{\theta}_k|\{\boldsymbol{Y}_k\}_{k=1}^K)$ parameters locally.

More specifically, by integrating out device parameters from (3), we can derive that

$$\int p(\boldsymbol{\theta}_1,...,\boldsymbol{\theta}_K,\boldsymbol{\phi}|\boldsymbol{Y}_1,...,\boldsymbol{Y}_K)d\boldsymbol{\theta}_1...d\boldsymbol{\theta}_K$$
$$= p(\boldsymbol{\phi}|\boldsymbol{Y}_1,...,\boldsymbol{Y}_K) = p(\boldsymbol{\phi})\prod_{k=1}^K \int p(\boldsymbol{Y}_k|\boldsymbol{\theta}_k,\boldsymbol{\phi})p(\boldsymbol{\theta}_k|\boldsymbol{\phi})d\boldsymbol{\theta}_k = p(\boldsymbol{\phi})\prod_{k=1}^K f_k(\boldsymbol{\phi}),$$

where $f_k(\boldsymbol{\phi}) = \int p(\boldsymbol{Y}_k|\boldsymbol{\theta}_k,\boldsymbol{\phi})p(\boldsymbol{\theta}_k|\boldsymbol{\phi})d\boldsymbol{\theta}_k$. As a consequence, our main goal is to collaboratively learn $p(\boldsymbol{\phi}|\boldsymbol{Y}_1,...,\boldsymbol{Y}_K)$ in a federated setting. One key challenge, however, is that the central server does not have any access to any edge dataset $\boldsymbol{D}_k$ and therefore directly computing $p(\boldsymbol{\phi}|\boldsymbol{Y}_1,...,\boldsymbol{Y}_K)$ is infeasible. For this reason, we resort to a trick based on the approximate inference methods to learn this posterior distribution.

4.2. Federated Bayesian inference - expectation propagation

In this section, we will present the federated inference framework for `HM2`. Specifically, our goal is to learn the posterior density $p(\boldsymbol{\phi}|\boldsymbol{Y}_1,...,\boldsymbol{Y}_K)$. In statistics, the most straightforward and popular approaches to do so are Markov Chain Monte Carlo (MCMC) methods. Yet, as will be clear shortly, we argue that sampling methods are not practical in the federated hierarchical setting, due to their sequential nature. Take Gibbs sampling as an example, device 1 needs to sample $\boldsymbol{\theta}_1$ from the density $p(\boldsymbol{\theta}_1|\boldsymbol{\theta}_2,...,\boldsymbol{\theta}_K,\boldsymbol{\phi},\boldsymbol{Y}_1)$ then pass those samples to the central server. The central server then needs to transmit those sampled $\boldsymbol{\theta}_1$ to device 2, and device 2 will sample from $p(\boldsymbol{\theta}_2|\boldsymbol{\theta}_1,...,\boldsymbol{\theta}_K,\boldsymbol{\phi},\boldsymbol{Y}_2)$. It can be seen that this sequential nature of MCMC significantly increases the communication cost and also slows down the federated optimization process when the total number of devices is large. Even if one can smartly parallelize the sampling process, the number of MCMC samples obtained locally will be large if the dimension is high, due to the curse of dimensionality (Jordan *et al.*, 2019).

To resolve the aforementioned issues, we resort to Expectation Propagation (EP) (Minka, 2001) to approximate the posterior distribution. EP is one of the most widely-used algorithms for computing an approximate posterior distribution (Minka, 2013; Vehtari *et al.*, 2020). Here, we first briefly introduce the idea of EP in a centralized regime. Consider a posterior distribution with independent data points

$$\pi(\boldsymbol{\phi}) := p(\boldsymbol{\phi}|\boldsymbol{Y}) \propto p(\boldsymbol{\phi})\prod_{i=1}^N p(y_i|\boldsymbol{\phi})$$

where $\boldsymbol{Y} = (y_1,...,y_N)^\top$ is the data vector. EP approximates $\pi(\boldsymbol{\phi})$ by a density $q(\boldsymbol{\phi})$ such that

$$q(\boldsymbol{\phi}) = p(\boldsymbol{\phi})\prod_{i=1}^N q_i(\boldsymbol{\phi}).$$

Intuitively, EP uses $q_i(\boldsymbol{\phi})$ to approximate $p(y_i|\boldsymbol{\phi})$, for all i. The most common choice is the normal density (Vehtari *et al.*, 2020). To achieve this goal, at each iteration, EP first takes an approximation factor $q_i(\boldsymbol{\phi})$ out from the current $q(\boldsymbol{\phi})$ and replaces it with the true factor $p(y_i|\boldsymbol{\phi})$. This step yields a new density $q^{new}(\boldsymbol{\phi})$. This resulting new density can be used as an updated approximated posterior. This step is iterated over all i till convergence. Please, refer to Barthelme (2016) for a comprehensive summary of EP. It can be seen that EP can be naturally extended to FDA where each device can be viewed as an independent "data point". In the following paragraphs, we will detail the federated extension of EP.

The main idea is to approximate terms $f_k(\boldsymbol{\phi})$ by a local device approximation function $q_k(\boldsymbol{\phi})$ for all $k = 1,...,K$. More specifically, we have

$$p(\boldsymbol{\phi}|\boldsymbol{Y}_1,...,\boldsymbol{Y}_K) \approx p(\boldsymbol{\phi})\prod_{k=1}^K q_k(\boldsymbol{\phi}) := q(\boldsymbol{\phi}). \tag{4}$$

Using the framework of EP, we gradually update $q(\boldsymbol{\phi})$ by iteratively renewing $q_k(\boldsymbol{\phi})$ at each device k. During each communication round, given the estimated $q(\boldsymbol{\phi})$ broadcasted from the server, device k first computes the cavity distribution

$$q_{-k}(\boldsymbol{\phi}) \propto \frac{q(\boldsymbol{\phi})}{q_k(\boldsymbol{\phi})} \tag{5}$$

and the tilted distribution

$$q_{\backslash k}(\boldsymbol{\phi}) \propto f_k(\boldsymbol{\phi})q_{-k}(\boldsymbol{\phi}). \tag{6}$$

It then computes the updated posterior approximation such that

$$q^{new}(\boldsymbol{\phi}) \approx q_{\backslash k}(\boldsymbol{\phi}). \tag{7}$$

Intuitively, the cavity distribution $q_{-k}(\boldsymbol{\phi})$ removes the impact of the old $q_k(\boldsymbol{\phi})$ from the approximated posterior density $q(\boldsymbol{\phi})$ and the tilted distribution adds the true target density $f_k(\boldsymbol{\phi})$ to $q_{-k}(\boldsymbol{\phi})$. As a result, we use the tilted distribution as an updated approximation to the posterior density of $\boldsymbol{\phi}$. This step is typically done through a sampling method and is distribution-dependent. We will detail this approximation procedure in Section 4.2.2.

Afterward, device k calculates the change in its local approximation by

$$\Delta q_k(\boldsymbol{\phi}) = \frac{q^{new}(\boldsymbol{\phi})}{q(\boldsymbol{\phi})}. \tag{8}$$

Instead of sending q^{new} back to the server, we calculate the change in the global posterior imposed by device k via (8) and sends $\Delta q_k(\boldsymbol{\phi})$ to the central server. The server aggregates all device approximations by

$$q(\boldsymbol{\phi}) \leftarrow q(\boldsymbol{\phi})\prod_{k=1}^K \Delta q_k(\boldsymbol{\phi}). \tag{9}$$

We summarize the EP algorithm in Algorithm 2.

Algorithm 2: The Federated Expectation Propagation Algorithm

Data: number of devices K, set $\mathcal{S}$ that contains indices of the selected devices, number of communication rounds C, initial approximation $\{q_k(\boldsymbol{\phi})\}_{k=1}^K$, prior $p(\boldsymbol{\phi})$.
for $c = 0 : (C-1)$ **do**
  Server broadcasts $q(\boldsymbol{\phi})$;
  for $k \in \mathcal{S}$ **do**
    Device-side: Calculate the cavity distribution $q_{-k}(\boldsymbol{\phi})$ using Eq. (5);
    Device-side: Calculate the tilted distribution $q_{\backslash k}(\boldsymbol{\phi})$ using Eq. (6);
    Device-side: Get new $q(\boldsymbol{\phi})$ from the tilted distribution using Eq. (7);
    Device-side: Calculate $\Delta q_k(\boldsymbol{\phi})$ using Eq. (8) and update local $q_k(\boldsymbol{\phi})$;
    Device-side: Send $\Delta q_k(\boldsymbol{\phi})$ to the central server;
  end
  Server-side: Update $q(\boldsymbol{\phi})$ using Eq. (9);
end
Return $q(\boldsymbol{\phi})$.

4.2.1. Posterior of device parameters

Once we obtain $q(\boldsymbol{\phi})$ that approximates $p(\boldsymbol{\phi}|\boldsymbol{Y}_1, ..., \boldsymbol{Y}_k)$, we can further estimate the posterior of device parameters. Specifically, given a device k,

$$\begin{aligned} p(\boldsymbol{\theta}_k|\{\boldsymbol{Y}_k\}_{k=1}^K) &= \int_{\boldsymbol{\phi}} \int_{\boldsymbol{\theta}_j, j\neq k} p(\boldsymbol{\theta}_k, \boldsymbol{\phi}|\{\boldsymbol{Y}_k\}_{k=1}^K) d\boldsymbol{\theta}_j d\boldsymbol{\phi} \\ &\propto \int p(\boldsymbol{\phi}) p(\boldsymbol{Y}_k|\boldsymbol{\theta}_k, \boldsymbol{\phi}) p(\boldsymbol{\theta}_k|\boldsymbol{\phi}) \\ &\quad \prod_{j\neq k} \int p(\boldsymbol{Y}_j|\boldsymbol{\theta}_j, \boldsymbol{\phi}) p(\boldsymbol{\theta}_j|\boldsymbol{\phi}) d\boldsymbol{\theta}_j d\boldsymbol{\phi} \\ &\approx \int q_{-k}(\boldsymbol{\phi}) p(\boldsymbol{Y}_k|\boldsymbol{\theta}_k, \boldsymbol{\phi}) p(\boldsymbol{\theta}_k|\boldsymbol{\phi}) d\boldsymbol{\phi}. \end{aligned}$$

As a consequence, we can use the posterior of $\boldsymbol{\theta}_k$ to quantify uncertainties or conduct hypothesis testing. The posterior samples from $p(\boldsymbol{\theta}_k|\{\boldsymbol{Y}_k\}_{k=1}^K)$ can be obtained by off-the-shelf posterior sampling techniques (see `mcmc` package in R or `NumPyro` library in Python). Here we provide a simpler sampling trick. In the above equation, if we ignore the integral, we can obtain the joint posterior

$$p(\boldsymbol{\theta}_k, \boldsymbol{\phi}|\{\boldsymbol{Y}_k\}_{k=1}^K) \approx q_{-k}(\boldsymbol{\phi}) p(\boldsymbol{Y}_k|\boldsymbol{\theta}_k, \boldsymbol{\phi}) p(\boldsymbol{\theta}_k|\boldsymbol{\phi}). \tag{10}$$

As a result, one can ignore the integration and use sampling methods to jointly sample $(\boldsymbol{\theta}_k, \boldsymbol{\phi})$ from (10) and discard $\boldsymbol{\phi}$. Now, given M samples $\{\boldsymbol{\theta}_{ki}\}_{i=1}^M$, we can readily use the samples to estimate moments, coverage probability, and do hypothesis tests.

Here we should note that the estimated posterior density $q(\boldsymbol{\phi})$ encodes crucial information across all devices (recall the explanation in Section 4.1). One can exploit this information for a new device to achieve fast adaptation. For example, we can treat the posterior mean of $\boldsymbol{\phi}$ as an initial model parameter for a new device. This idea is similar to meta-learning (Vanschoren, 2019), where one tries to learn a global model that can quickly adapt to a new task.

4.2.2. Normal approximation

In practice, it is common to model $p(\boldsymbol{\phi})$ and $q_k(\boldsymbol{\phi}), \forall k$ as normal densities. This is due to a very useful property of normal random variables.

Lemma 1. *(Williams and Rasmussen, 2006) Suppose there are two normal random variables (with the same dimension) such that* $\boldsymbol{\theta}_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ *and* $\boldsymbol{\theta}_2 \sim \mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$. *Let* $\boldsymbol{r}_i = \boldsymbol{\Sigma}_i^{-1}\boldsymbol{\mu}_i$, $\boldsymbol{Q}_i = \boldsymbol{\Sigma}_i^{-1}$ *for* $i = 1, 2$. *Define* $p(\boldsymbol{\theta}_+) = p(\boldsymbol{\theta}_1)p(\boldsymbol{\theta}_2)$ *and* $p(\boldsymbol{\theta}_-) = \frac{p(\boldsymbol{\theta}_1)}{p(\boldsymbol{\theta}_2)}$. *We have that*

$$\begin{aligned} \boldsymbol{\theta}_+ &\sim \mathcal{N}(\boldsymbol{r}_1 + \boldsymbol{r}_2, \boldsymbol{Q}_1 + \boldsymbol{Q}_2) \\ \boldsymbol{\theta}_- &\sim \mathcal{N}(\boldsymbol{r}_1 - \boldsymbol{r}_2, \boldsymbol{Q}_1 - \boldsymbol{Q}_2). \end{aligned}$$

Lemma 1 states that the product of two Gaussian densities gives another unnormalized Gaussian density. We use θ_+ to represent this new Gaussian random variable. Similarly, the quotient of two Gaussian densities gives an unnormalized Gaussian density and we use θ_- to represent this new random variable.

Using Lemma 1, one can efficiently implement the EP algorithm, as all components in Algorithm 2 can be computed in closed forms. Here, we detail the implementation technique. We model the prior of $\boldsymbol{\phi}$ as a multivariate normal random variable with mean $\boldsymbol{\mu}_0$ and variance $\boldsymbol{\Sigma}_0$. We also assume $q_k(\boldsymbol{\phi})$ has a normal density parameterized by $\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k$, for all k. If the support of some components in $\boldsymbol{\phi}$ does not lie in $\mathbb{R}$, one can always perform a logarithmic or logistic transformation to those components. By Gaussian properties, (4) can be computed in closed-form such that $q(\boldsymbol{\phi})$ has a normal density parametrized by mean $\boldsymbol{r}_0 + \sum_{k=1}^K \boldsymbol{r}_k$ and variance $\boldsymbol{Q}_0 + \sum_{k=1}^K \boldsymbol{Q}_k$, where $\boldsymbol{r}_j = \boldsymbol{\Sigma}_j^{-1}\boldsymbol{\mu}_j$ and $\boldsymbol{Q}_j = \boldsymbol{\Sigma}_j^{-1}$, for all $j = 0, 1, ..., K$. Similarly, by Lemma 1, the cavity distribution in (5) can be computed in a closed-form by a subtraction operation. This results in $\boldsymbol{r}_{-k} := \boldsymbol{r} - \boldsymbol{r}_k, \boldsymbol{Q}_{-k} := \boldsymbol{Q} - \boldsymbol{Q}_k$. Compared to sampling approaches, one key advantage of EP is that the computation and communication steps are simple and efficient. The central server and devices only need to transmit the mean vector and variance matrix to perform model updating and aggregation. Notably, during each communication round, there is no need to estimate or transmit normalizing constants. We detail this idea in Algorithm 3. In Algorithm 3, one needs to compute the tilted distribution and obtain $\boldsymbol{r}_k^{new}, \boldsymbol{Q}_k^{new}$. These steps correspond to (6) and (7). However, as $f_k(\boldsymbol{\phi})$ may not be a normal density, the resulting tilted distribution is not normal. Therefore, we need to use a normal distribution to approximate the tilted distribution. To do so, we can run a set of simulation draws (using any sampling method coded in R or Python libraries) from $f_k(\boldsymbol{\phi})q_{-k}(\boldsymbol{\phi})$ and estimate the mean and covariance of those draws. We set the resulting mean to be $\boldsymbol{\mu}_{\backslash k}$ and the resulting covariance to be $\boldsymbol{\Sigma}_{\backslash k}$.

Algorithm 3: The Federated Expectation Propagation Algorithm using Normal Approximation

Data: number of devices K, set $\mathcal{S}$ that contains indices of the selected devices, number of communication rounds C, initial approximation $\{\boldsymbol{r}_k, \boldsymbol{Q}_k\}_{k=1}^K$, prior $\boldsymbol{r}_0, \boldsymbol{Q}_0$, initial posterior parameters $\boldsymbol{r} = \boldsymbol{r}_0 + \sum_{k=1}^K \boldsymbol{r}_k, \boldsymbol{Q} = \boldsymbol{Q}_0 + \sum_{k=1}^K \boldsymbol{Q}_k$.

for $c = 0 : (C-1)$ **do**
  Server broadcasts $\boldsymbol{r}, \boldsymbol{Q}$;
  for $k \in \mathcal{S}$ **do**
    Device-side: Calculate the cavity distribution with parameters $\boldsymbol{r}_{-k} := \boldsymbol{r} - \boldsymbol{r}_k, \boldsymbol{Q}_{-k} := \boldsymbol{Q} - \boldsymbol{Q}_k$;
    Device-side: Calculate the tilted distribution $\mathbf{r}_{\backslash k}, \mathbf{Q}_{\backslash k}$;
    Device-side: Obtain new $\boldsymbol{r}_k^{new}, \boldsymbol{Q}_k^{new}$;
    Device-side: Calculate $\Delta \boldsymbol{r}_k = \boldsymbol{r}_k^{new} - \boldsymbol{r}_k, \Delta \boldsymbol{Q}_k = \boldsymbol{Q}_k^{new} - \boldsymbol{Q}_k$;
    Device-side: Send $\Delta \boldsymbol{r}_k, \Delta \boldsymbol{Q}_k$ to the central server;
  end
  Server-side: Update $\boldsymbol{r} = \boldsymbol{r} + \sum_{k \in \mathcal{S}} \Delta \boldsymbol{r}_k, \boldsymbol{Q} = \boldsymbol{Q} + \sum_{k \in \mathcal{S}} \Delta \boldsymbol{Q}_k$;
end
Return $\boldsymbol{r}, \boldsymbol{Q}$.

4.3. Federated and penalized regression for variable selection

To move a step further, we ask "is it possible to let devices exploit the shared representation structure to perform variable selection?" Indeed, there are some attempts to tackle this question from a frequentist perspective. Yuan *et al.* (2021) develop a federated composite optimization framework that solves the federated Lasso problem. Tong *et al.* (2020) propose a federated iterative hard thresholding algorithm to tackle non-convex penalized regression. Despite these few efforts in exploring variable selection from a frequentist perspective, no literature exists in the Bayesian setting.

Tibshirani (1996) has shown that a Lasso estimate can be achieved when the regression parameters have *i.i.d.* Laplace priors. Since then, researchers have started to build Bayesian priors for many other penalized regressions such as the elastic net and fussed Lasso. Please refer to the work of Van Erp *et al.* (2019) for a detailed literature review. Inspired by the Bayesian interpretation of penalized regressions, we develop a hierarchical structure, based upon `HM2`, to perform federated variable selection. To proceed, we impose priors on $\boldsymbol{\theta}_k$, for all k, such that

$$\theta_{ki} | \boldsymbol{\phi} \sim \pi(\lambda, \sigma^2), \forall i = 1, ..., d$$

where $\boldsymbol{\phi} = (\lambda, \sigma^2)$, λ is a regularization parameter and σ^2 is a variance parameter. Here, $\pi(\lambda, \sigma^2)$ is a distribution parameterized by λ, σ^2. For instance, if we set $\pi(\lambda, \sigma^2)$ to be a Laplace distribution with zero mean and $\frac{\sigma}{\lambda}$ diversity, then we recover the Bayesian counterpart of Lasso regression (Tibshirani, 1996). Another example is if we set $\pi(\lambda, \sigma^2)$ to be $\mathcal{N}\left(0, \frac{\sigma}{\lambda}\right)$, then we recover Ridge regression. There are many possible choices of prior beliefs on σ^2 and λ. In this work, we impose log-normal priors on σ^2 and λ (Van Erp *et al.*, 2019) and we set $\boldsymbol{\phi} = (\log \lambda, \log \sigma^2)$. Our framework can flexibly incorporate other priors such as a non-informative prior on σ^2 or a half-Cauchy prior on λ. In this work, we will use a log-normal prior as an illustrative example.

The posterior distribution of $\boldsymbol{\theta}_k$, for all k, and $\boldsymbol{\phi}$ can be learned by Algorithm 2. Here, one caveat is that, unlike frequentist penalized regressions, the Bayesian methods do not shrink regression coefficients to be exactly zero. As a result, we will calculate the Credible Interval (CI) for each parameter. If the CI of a parameter, say θ_{ki}, covers zero, we will exclude this predictor.

5. Simulation studies

5.1. `HM1`: Proof of concept using Algorithm 1

Case I: We assume that $K = 2$ and generate $(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$ from a matrix-variate normal distribution with zero mean and

$$\boldsymbol{I} \otimes \boldsymbol{\Omega} = \boldsymbol{I} \otimes \begin{bmatrix} 1 & 0.7 \\ 0.7 & 1 \end{bmatrix}$$

covariance. The input space dimension is $d = 5$. Data on each device are generated from linear models using the generated parameters. We set noise to be 0.05. To demonstrate the benefits of our correlation-based construction in `HM1`, we create imbalanced sample sizes on the devices. Specifically, device 1 only has $N_1 = 20$ data points whereas device 2 has $N_2 = 200$ data points. We train Algorithm 1 with $C = 30, \eta_2 = 0.01, \alpha = 0.1$ and we set the number of local steps T to be 20. We compare the performance of Algorithm 1 with a separate modeling approach where each device fits its own model without communication. Specifically, each device runs 600 local SGD steps with learning rate 0.01.

Case II: We set $K = 100$ and generate a 100×100 positive definite matrix $\boldsymbol{\Omega}$ using the R package `clusterGeneration`. We then generate true device parameters based on the matrix $\boldsymbol{\Omega}$. We set $d = 8$ and generate data using the linear models with noise $\sigma^2 = 0.1$. For the first 30 devices, we assign 40 data points and for the remaining 70 devices, we assign 275 data points. Overall, we create an imbalanced data generation scenario. Case II can be viewed as a generalization of Case I with more devices. Again, we train models using the same hyperparameters as those in Case I.

Case III: We use the same setting as the one in Case II, but all devices have 20 data points (i.e., balanced data generation).

Case IV: We increase the sample size on each device to 200 and use the same setting as the one in Case III.

Performance Evaluation: Denote by $(\boldsymbol{X}_k^*, \boldsymbol{Y}_k^*)$ the testing dataset on device k where $\boldsymbol{X}_k^* = [x_{k1}^*, ..., x_{kN_k^*}^*]^\top$ and $\boldsymbol{Y}_k^* = [y_{k1}^*, ..., y_{kN_k^*}^*]$. The averaged Root-mean-square error (A-

RMSE) across all devices is defined as

$$\text{A}-\text{RMSE}=\frac{1}{K}\sum_{k=1}^{K}\sqrt{\frac{\sum_{i=1}^{N_k^*}(f(x_{ki}^*)-y_{ki}^*)^2}{N_k^*}},$$

where $f_k(x^*)=x^{*\top}\boldsymbol{\theta}_k$. On each device, we generate 1000 data points using the true device parameters for testing. In Case I, the RMSE is calculated on device 1. In Case II, the A-RMSE is calculated using devices $k\in\{1,\ldots,30\}$. Here our goal is to assess the prediction accuracy on devices with scant data. In Cases III/IV, we calculate A-RMSE using all 100 devices. We report our results in Table 1. It can be seen that Algorithm 1 yields much smaller A-RMSE under the imbalanced data scenario. This conveys the importance of borrowing strength from other devices under the FDA framework. In Case III, the local sample size is insufficient to allow each model alone to perform well. However, our FDA can still benefit devices' training by borrowing information from other devices. In case IV, Algorithm 1 does not offer a major improvement, as all local devices have enough data. In this case, doing local training without collaboration should be sufficient.

Accuracy of Parameter Estimation: The negative log-likelihood function $L(\boldsymbol{\Theta},\boldsymbol{\Omega})$ is a non-convex function of $(\boldsymbol{\Theta},\boldsymbol{\Omega})$. To test the impact of the initialization, we conduct a sensitivity analysis below. Denote by $\hat{\boldsymbol{\Theta}}$ the concatenated estimated device parameters and $\boldsymbol{\Theta}^*$ the concatenated true data-generating parameters. In Figure 3, we plot $\frac{\|\hat{\boldsymbol{\Theta}}-\boldsymbol{\Theta}^*\|}{\sqrt{K}}$ versus communication round for Case II and III over 30 independent runs. Each independent run used a different initialized $\boldsymbol{\Theta}$. More specifically, we first generate $d*K$ random numbers from a standard normal distribution. We then create a $d*K$ matrix $\boldsymbol{\Theta}$ using these random numbers. It can be seen that Algorithm 1 accurately recovers the true underlying model parameters. Furthermore, it can be observed that Algorithm 1 typically converges within 30–40 communication rounds. We observed that, in all simulations, Algorithm 1 could be trained within 5 seconds on a standard laptop. In conclusion, our proposed algorithm is easy to implement and optimize.

Table 1. The A-RMSE of our proposed model and the separate model over 30 independent runs. We report standard deviations of A-RMSEs in brackets.

Case	*Algorithm 1*	*Separate*
I	0.081(±0.001)	0.094(±0.001)
II	0.050(±0.000)	0.056(±0.001)
III	0.044(±0.002)	0.072(±0.004)
IV	0.035(±0.000)	0.035(±0.000)

5.2. HM2: *Proof of concept*

In this section, we test the variable selection performance of HM2. Following the examples in Van Erp *et al.* (2019), we create several simulation cases below:

Case I: We set $K=10$, $\boldsymbol{\theta}_{true}=(3,1.5,0,0,2,0,0,0)^\top$ and generate all columns of $\{\boldsymbol{X}_k\}_{k=1}^K$ from a standard multivariate normal distribution. We then generate $\{\boldsymbol{Y}_k\}_{k=1}^K$ using $\boldsymbol{\theta}_{true}$ and $\{\boldsymbol{X}_k\}_{k=1}^K$ for all k. We set the noise to 0.05. Each device has 100 data points for training and 1000 data points for testing.

Case II: We use the same setting as the one in Case I. The difference is that the first two devices only have 20 data points each, whereas the other devices have 200 data points each. The number of testing data points is 1000.

Case III: We set $K=20$, $\boldsymbol{\theta}_{true}=(\underbrace{3,\ldots,3}_{10},\underbrace{0,\ldots,0}_{10},\underbrace{3,\ldots,3}_{10})^\top$. Each device has 40 observations for training and 400 observations for testing.

We evaluate the performance of our model based on prediction and variable selection accuracy. The prediction accuracy is evaluated by A-RMSE. Variable selection accuracy is based on the averaged correct and false inclusion rates. To decide whether to include a variable or not, we first calculate a

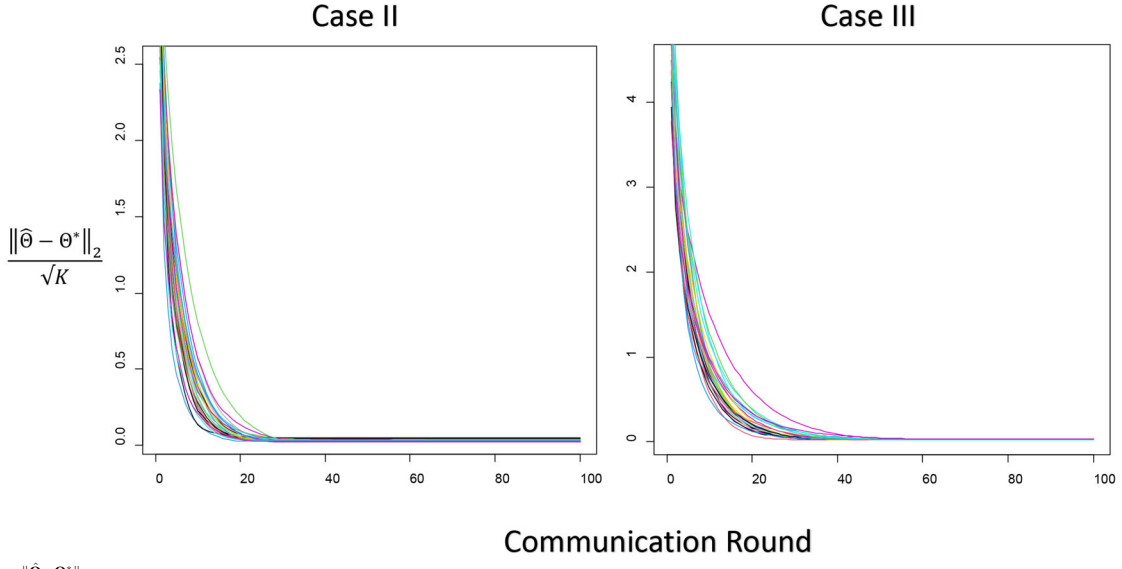

Figure 3. Plot of $\frac{\|\hat{\boldsymbol{\Theta}}-\boldsymbol{\Theta}^*\|}{\sqrt{K}}$ versus communication round. Each color represents one independent run.

Table 2. The A-RMSE and averaged correct/false inclusion rates for different federated variable selection methods over 30 experimental runs.

Methods	*A-RMSE*	*Averaged Correct Inclusion Rate*	*Averaged False Inclusion Rate*
Lasso (HM2, Case I)	0.055($\pm$0.001)	0.880($\pm$0.003)	0.095($\pm$0.001)
Lasso (HM2, Case II)	0.062($\pm$0.002)	0.875($\pm$0.004)	0.101($\pm$0.001)
Lasso (HM2, Case III)	0.088($\pm$0.001)	0.891($\pm$0.005)	0.115($\pm$0.001)

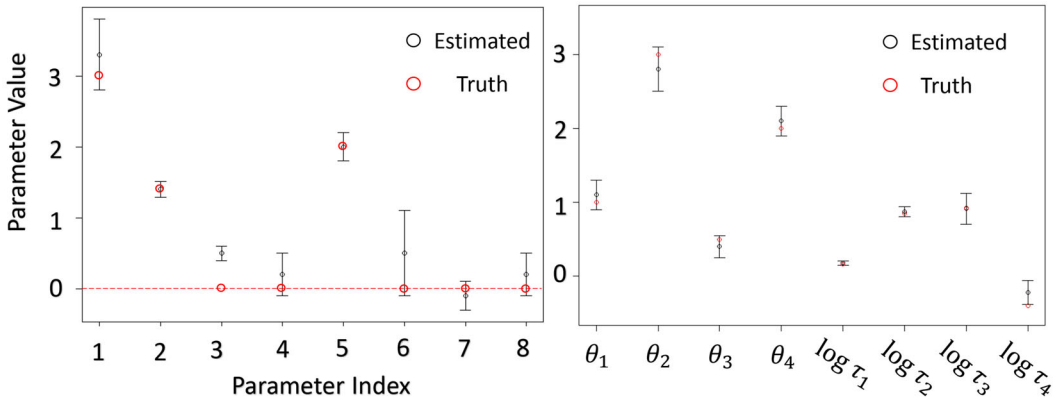

Figure 4. Plot of parameter estimation and CI.

90% CI for each parameter. If the CI covers zero, we will exclude this predictor. Results are reported in Table 2. It can be seen that our proposed federated variable selection methods can correctly identify more than 85% of effective predictors while maintaining low false inclusion rates.

As mentioned in Section 4, one advantage of HM2 is that it can provide Uncertainty Quantification (UQ) for parameter estimation. We will provide two examples to demonstrate the UQ capability of HM2:

1. We collect the estimated posterior for parameters $\boldsymbol{\theta}_1$ from an independent run in case I and calculate the mean and 90% credible interval. The resulting plot is presented in Figure 4 (Left). It can be seen that the true parameters are included in the CI generated by HM2.
2. We create $K = 100$ devices and generate device parameter

$$\begin{aligned}\boldsymbol{\theta}_k|\boldsymbol{\phi} &\sim \mathcal{N}(\boldsymbol{\mu}_{true}, \boldsymbol{\Sigma}_{true}) \\ &:= \mathcal{N}\left(\begin{bmatrix}1\\3\\0.5\\2\end{bmatrix}, \begin{bmatrix}1.17 & 0 & 0 & 0\\ 0 & 2.35 & 0 & 0\\ 0 & 0 & 2.52 & 0\\ 0 & 0 & 0 & 0.67\end{bmatrix}\right)\end{aligned}$$

for $k \in \{1, ..., 100\}$. We then use $\boldsymbol{\theta}_k$ to generate 100 data points for each device k. Our HM structure can be summarized as follows:

$$\begin{aligned}\boldsymbol{Y}_k|\boldsymbol{\theta}_k, \boldsymbol{\phi} &\sim \mathcal{N}(\boldsymbol{X}_k^\top \boldsymbol{\theta}_k, \sigma_k^2 \boldsymbol{I}) \\ \boldsymbol{\theta}_k|\boldsymbol{\phi} &\sim \mathcal{N}(\boldsymbol{\mu}, \mathrm{diag}(\tau_1, ..., \tau_4)) \\ \boldsymbol{\mu} &\sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}) \\ \tau_i &\sim \log\mathcal{N}(0, 1), \forall i \\ \boldsymbol{\phi} &= (\boldsymbol{\mu}, \log\tau_1, ..., \log\tau_4).\end{aligned}$$

We calculate the posterior distribution of $\boldsymbol{\phi}$ using Algorithm 3 and plot the mean and 90% CI for each component in Figure 4 (Right). It can be seen that the mean of the posterior of $\boldsymbol{\phi}$ is close to the truth and the 90% CI also covers the true parameter. This estimated $q(\boldsymbol{\phi})$ can be used as an initialization for new devices to achieve fast adaption.

6. Real-world case studies

6.1. Student performance dataset

This is a public dataset that can be found at https://archive.ics.uci.edu/ml/index.php. The dataset contains information on student performance (measured by exam scores) in secondary education of two Portuguese schools, namely, Gabriel Pereira and Mousinho da Silveira. It includes 29 predictors covering gender, grades, demographic, and many other social/school-related features. Detailed information can be found in Cortez and Silva (2008). We treat each school as a "device" (i.e., $K = 2$). On each device, we randomly pick 60% of the students as the training dataset and another 40% of the students as the testing dataset. We create dummy variables for all nominal variables, such as job and guardian. All other numeric variables are standardized to a zero mean and one standard deviation, following the guide in Cortez and Silva (2008). This data processing yields 38 predictors.

Our first goal is to select relevant predictors using our federated penalized regression technique (See Section 4.3). We then use the selected predictors to predict the final exam grades of students. We consider the two most widely-used variable selection methods: Lasso and Ridge. Results are reported in Table 3. The model performance is evaluated based on the RMSE and the number of included predictors.

It can be seen that the variable selection performance of HM2 is consistent with the centralized variable selection method such as Lasso and Ridge regressions. This implies that our framework can serve as a new paradigm for decentralized variable selection problems. Additionally, HM2 also

Table 3. The RMSE and number of included predictors for different federated variable selection methods.

Methods	*A-RMSE*	*# included predictors (School 1)*	*# included predictors (School 2)*
Lasso (HM2)	0.825	21	23
Ridge (HM2)	0.817	21	22
Methods	*RMSE*	*# included predictors*	
Lasso (Centralized)	0.820	21	
Ridge (Centralized)	0.815	21	

Table 4. The A-RMSE of all models over 30 independent runs. We report the standard deviation in the brackets.

Sensor	HM1 ($\alpha = 0.9$)	Separate	FedAvg	Ditto	Dis-Ridge
Sensor 2	**0.270**($\pm$**0.001**)	0.299($\pm$0.003)	0.450($\pm$0.013)	0.281($\pm$0.001)	0.552($\pm$0.002)
Sensor 3	**0.218**($\pm$**0.002**)	0.223($\pm$0.001)	0.303($\pm$0.009)	0.220($\pm$0.001)	0.288($\pm$0.002)
Sensor 7	**0.369**($\pm$**0.004**)	0.405($\pm$0.003)	0.628($\pm$0.011)	0.388($\pm$0.005)	0.605($\pm$0.001)
Sensor 8	**0.267**($\pm$**0.001**)	0.307($\pm$0.001)	0.395($\pm$0.008)	0.289($\pm$0.001)	0.390($\pm$0.003)

yields comparable A-RMSEs compared to centralized methods. This demonstrates the advantage of borrowing strength from other devices. However, please note that, in terms of prediction accuracy, federated variable selection can rarely beat the centralized approach as the latter uses more data.

6.2. NASA aircraft gas turbine engines

In this case study, we consider condition monitoring data generated from aircraft gas turbine engines using the NASA commercial Modular Aero-Propulsion System Simulation (C-MAPSS) tools. The dataset is available at https://ti.arc.nasa.gov/tech/dash/groups/pcoe/. This dataset contains 100 engines. In each engine, 24 sensors are installed to collect time-series degradation signals. For each engine, we treat the first 60% of the time-series observations as the training dataset and the remaining 40% of the signals as the testing dataset. Within the training dataset, we sample 20% of the data as a validation dataset. Our goal is therefore to predict the sensor signal trajectory on each gas turbine engine by training Algorithm 1 using the training dataset. In this scenario, each engine can be viewed as a device (i.e., $K = 100$).

It can be observed that all signal trajectories exhibit polynomial patterns and therefore, many existing works resort to polynomial regression to analyze this dataset (Liu *et al.*, 2013; Song and Liu, 2018). Here we detail the modeling procedure. Given a specific sensor, for all $k \in \{1, ..., K\}$, device k fits a $d = 6$th-order polynomial regression in the form of

$$\boldsymbol{Y}_k = \boldsymbol{X}_k^\top \boldsymbol{\theta}_k + \text{noise},$$

where the $(d+1) \times N_k$ design matrix $\boldsymbol{X}_k$ is in the form of

$$\boldsymbol{X}_k^\top = \begin{bmatrix} 1 & [x_{k1}]_1 & [x_{k1}]_2^2 & \dots & [x_{k1}]_d^d \\ 1 & [x_{k2}]_1 & [x_{k2}]_2^2 & \dots & [x_{k2}]_d^d \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & [x_{kN_k}]_1 & [x_{kN_k}]_2^2 & \dots & [x_{kN_k}]_d^d \end{bmatrix}.$$

In the above expression, $\boldsymbol{Y}_k$ represents the signal trajectory for device k and $\boldsymbol{x}_{k1}$ represents time. In HM1, device parameters $\{\boldsymbol{\theta}_k\}_k$ are estimated using Algorithm 1.

We benchmark our proposed model with the following algorithms:

- FedAvg: FedAvg is one of the most fundamental and competing benchmark models in the FDA. During each communication round, device k, for all selected k, first runs several steps of local SGD and then sends updated parameters $\boldsymbol{\theta}_k$ back to the server. The server the aggregates those parameters by calculating $\bar{\boldsymbol{\theta}} = \frac{1}{|\mathcal{S}|}\sum_{k\in\mathcal{S}} \boldsymbol{\theta}_k$. This step is repeated several times and ultimately, each device will use the global parameter $\bar{\boldsymbol{\theta}}$ to perform prediction.
- Dis-Ridge (Zhang *et al.*, 2015): Dis-Ridge is a distributed ridge regression method. The server first evenly divides a set of data into m disjoint sets and assigns each set to a different node. Each node then solves a ridge regression problem, by optimizing $\frac{1}{N_k}\sum_{i=1}^{N_k}(y_{ki} - \boldsymbol{x}_{ki}^\top \boldsymbol{\theta}_k)^2 + \lambda\|\boldsymbol{\theta}_k\|^2$, and sends the optimal solution back to the server. The server then aggregates local estimations.
- Ditto: Ditto is a personalized FL algorithm. The first stage of Ditto is the same as FedAvg and generates a global parameter $\bar{\boldsymbol{\theta}}$. Afterwards, each device k derives the personalized solution $\boldsymbol{v}_k$ by solving a constrained optimization problem $F_k(\boldsymbol{v}_k) + \frac{\lambda_{Ditto}}{2}\|\boldsymbol{v}_k - \bar{\boldsymbol{\theta}}\|_2^2$ where $F_k(\cdot)$ is the local loss function and λ_{Ditto} is a tuning parameter. The intuition is that each device can run several updating procedures to collect personalized solutions while this solution stays in the vicinity of the shared global model to retain useful information from a global model.
- Separate: In Separate, each device simply fits its own linear model without communication.

For all models, we set $T = 20, C = 100$, and use grid-search to tune the learning rate (and other model hyper-parameters). In Algorithm 1, we set $\alpha = 0.9$. We report the A-RMSE across all 100 devices in Table 4.

From Table 4, it can be seen that FedAvg and Dis-Ridge consistently yield the worst prediction accuracy as one shared global parameter $\bar{\boldsymbol{\theta}}$ does not suit all devices, especially in a heterogeneous setting. Personalized approaches such as Ditto circumvent this disadvantage of global models and generate personalized solutions for each

device. Those personalized methods, however, ignore related information amongst devices. Our method, on the other hand, improves the prediction accuracy by exploiting a joint structure for inductive transfer.

7. Conclusion

This article proposes a federated treatment for linear regression by adopting a HM approach. We test our proposed framework on a range of simulated and real-world datasets. Despite the simplicity of our linear model framework, it can outperform many state-of-the-art federated algorithms and we argue that it can serve as a competing benchmark model for the future development of federated algorithms. One possible future direction is to extend our framework to generalized linear models such as linear mixed-effect models or to more complicated models such as Gaussian processes and tensor regression. We hope our work will help inspire continued exploration into the world of federated data analytics and its engineering applications.

Funding

The authors acknowledge the generous support from the NSF CAREER AWARD 2144147.

Notes on contributors

Xubo Yue is a PhD candidate in the Department of Industrial & Operations Engineering at the University of Michigan. His research focuses on federated and distributed data analytics. Currently, he is developing federated data analytics methods that rethink how both prescriptive and predictive analytics are achieved within IoT-enabled systems, specifically manufacturing and renewable energy. He has received several best paper awards from the Institute for Operations Research and the Management Sciences (INFORMS), the Institute of Industrial and Systems Engineers (IISE), and other renowned organizations.

Raed Kontar is an assistant professor in the Industrial & Operations engineering department at the University of Michigan and an affiliate with the Michigan Institute for Data Science. Raed's research focuses on distributed and federated probabilistic modeling. Raed obtained an undergraduate degree in civil & environmental engineering from the American University of Beirut in 2014, a master's degree in statistics in 2017 and a PhD degree in industrial & systems engineering in 2018, both from the University of Wisconsin-Madison. Raed received the NSF CAREER award in 2022. His research is currently supported by both NSF and NIH.

Ana María Estrada Gómez is an assistant professor in the School of Industrial Engineering at Purdue University. She received a BSc in industrial engineering and a B.Sc. in mathematics from la Universidad de los Andes in 2013 and 2015, respectively. She also holds a M.Sc. in industrial engineering from la Universidad de los Andes (2015), and a M.Sc. in statistics from Georgia Tech (2018). In 2021, she received her PhD in industrial engineering with a specialization in statistics from Georgia Tech. Her research interests lie in developing efficient methodologies and algorithms for modeling, monitoring, and diagnosing complex systems collecting high-dimensional data, using statistics and machine learning tools. The methods that she has developed have been applied in the manufacturing, environmental, and healthcare sectors. She is the recipient of the SPES + Q&P Best Student Paper Award from ASA, the QSR Best Poster Award from INFORMS, and the IISE Doctoral Colloquium Best Poster Award. At Georgia Tech, she was recognized with the Graduate Teaching Fellowship, granted by the Center for Teaching and Learning, and with the Stewart Fellowship, awarded by the School of Industrial and Systems Engineering. She has also been appointed as a Latina Trailblazer in Engineering Fellow by Purdue's College of Engineering.

References

Albert, J. and Hu, J. (2019) *Probability and Bayesian Modeling*, CRC Press, Milton Park, Oxfordshire.

Arashi, M., Roozbeh, M., Hamzah, N.A. and Gasparini, M. (2021) Ridge regression and its applications in genetic studies. *Plos one*, **16**(4), e0245376.

Barthelme, S. (2016) Simon Barthelme: The expectation-propagation algorithm: A tutorial - part 1. https://www.youtube.com/watch?v=0tomU1q3AdY&t=2373s

Blanco-Filgueira, B., Garcia-Lesta, D., Fernández-Sanjurjo, M., Brea, V.M. and López, P. (2019) Deep learning-based multiple object visual tracking on embedded system for iot and mobile edge computing applications. *IEEE Internet of Things Journal*, **6**(3), 5423–5431.

CleanTechnica (2021) Tesla fsd hardware has 150 million times more computer power than Apollo 11 computer. https://cleantechnica.com/2021/05/24/tesla-fsd-hardware-has-150-million-times-more-computer-power-than-apollo-11-computer/#:~:text=Tesla's%20twin%20chips%20and%20combined,and%20Collins%20to%20the%20moon (accessed 24 May 2021).

Cortez, P. and Silva, A.M.G. (2008) Using data mining to predict secondary school student performance, in A. Brito and J. Teixeira (eds.), *Proceedings of 5th Annual Future Business Technology Conference*, EUROSIS-ETI, Porto, pp. 5–12.

Deng, Y., Kamani, M.M. and Mahdavi, M. (2020) Adaptive personalized federated learning. *arXiv preprint arXiv:2003.13461*.

Du, W., Xu, D., Wu, X. and Tong, H. (2021) Fairness-aware agnostic federated learning, In *Proceedings of the 2021 SIAM International Conference on Data Mining (SDM)*, SIAM, Philadelphia, PA, pp. 181–189.

Fallah, A., Mokhtari, A. and Ozdaglar, A. (2020) Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*.

Fan, J., Guo, Y. and Wang, K. (2021) Communication-efficient accurate statistical estimation. *Journal of the American Statistical Association*, **116**, 1–11.

Hassan, N., Gillani, S., Ahmed, E., Yaqoob, I. and Imran, M. (2018) The role of edge computing in internet of things. *IEEE Communications Magazine*, **56**(11), 110–115.

Jordan, M.I., Lee, J.D. and Yang, Y. (2019) Communication-efficient distributed statistical inference. *Journal of the American Statistical Association*, **114**(526), 668–681.

Kairouz, P., McMahan, H.B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A.N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R. *et al.* (2021) Advances and open problems in federated learning. *Foundations and Trends[textregistered] in Machine Learning*, **14**(1–2), 1–210.

Kontar, R., Shi, N., Yue, X., Chung, S., Byon, E., Chowdhury, M., Jin, J., Kontar, W., Masoud, N., Nouiehed, M. *et al.* (2021) The internet of federated things (ioft). *IEEE Access*, **9**, 156071–156113.

Lawson, P., McPhail, B. and Lawton, E. (2015) The connected car: Who is in the driver's seat? A study on privacy and onboard vehicle telematics technology. https://trid.trb.org/view/1348911

Li, J., Xu, J. and Zhou, Q. (2018) Monitoring serially dependent categorical processes with ordinal information. *IISE Transactions*, **50**(7), 596–605.

Li, T., Hu, S., Beirami, A. and Smith, V. (2021) Ditto: Fair and robust federated learning through personalization, in *International Conference on Machine Learning*, Virtual Conference, International Conference on Machine Learning, Vienna, Austria, pp. 6357–6368.

Li, T., Sahu, A.K., Talwalkar, A. and Smith, V. (2020) Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, **37**(3), 50–60.

Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A. and Smith, V. (2020) Federated optimization in heterogeneous networks. *Proceedings of Machine Learning and Systems*, **2**, 429–450.

Li, T., Sanjabi, M., Beirami, A. and Smith, V. (2019) Fair resource allocation in federated learning. *arXiv preprint arXiv:1905.10497*.

Li, X., Huang, K., Yang, W., Wang, S. and Zhang, Z. (2019) On the convergence of fedavg on non-iid data. *arXiv preprint arXiv: 1907.02189*.

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.

Liu, S., Liu, L., Tang, J., Yu, B., Wang, Y. and Shi, W. (2019) Edge computing for autonomous driving: Opportunities and challenges. *Proceedings of the IEEE*, **107**(8), 1697–1716.

Liu, W., Chen, L., Chen, Y. and Zhang, W. (2020) Accelerating federated learning via momentum gradient descent. *IEEE Transactions on Parallel and Distributed Systems*, **31**(8), 1754–1766.

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*, The 20th International Conference on Artificial Intelligence and Statistics, Ft. Lauderdale, FL, pp. 1273–1282.

Minka, T.P. (2001) A family of algorithms for approximate Bayesian inference. Ph.D. thesis, Massachusetts Institute of Technology, Cambridge, MA.

Minka, T.P. (2013) Expectation propagation for approximate Bayesian inference. *arXiv preprint arXiv:1301.2294*.

Morell, J.Á. and Alba, E. (2022) Dynamic and adaptive fault-tolerant asynchronous federated learning using volunteer edge devices. *Future Generation Computer Systems*, **133**, 53–67.

Patterson, D., Gonzalez, J., Le, Q., Liang, C., Munguia, L.-M., Rothchild, D., So, D., Texier, M. and Dean, J. (2021) Carbon emissions and large neural network training. *arXiv preprint arXiv: 2104.10350*.

Rahman, M.A. and Hossain, M.S. (2021) An internet-of-medical-things-enabled edge computing framework for tackling covid-19. *IEEE Internet of Things Journal*, **8**(21), 15847–15854.

Ruder, S. (2017) An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*.

Şahin, D.Ö., Akleylek, S. and Kiliç, E. (2022) Linregdroid: Detection of android malware using multiple linear regression models-based classifiers. *IEEE Access*, **10**, 14246–14259.

Schulz, M.-A., Yeo, B., Vogelstein, J.T., Mourao-Miranada, J., Kather, J.N., Kording, K., Richards, B. and Bzdok, D. (2020) Different scaling of linear models and deep learning in ukbiobank brain images versus machine-learning datasets. *Nature Communications*, **11**(1), 1–15.

Shi, N., Lai, F., Kontar, R.A. and Chowdhury, M. (2021) Fed-ensemble: Improving generalization through model ensembling in federated learning. *arXiv preprint arXiv:2107.10663*.

Shi, W., Cao, J., Zhang, Q., Li, Y. and Xu, L. (2016) Edge computing: Vision and challenges. *IEEE Internet of Things Journal*, **3**(5), 637–646.

Si, W., Yang, Q. and Wu, X. (2017) A distribution-based functional linear model for reliability analysis of advanced high-strength dual-phase steels by utilizing material microstructure images. *IISE Transactions*, **49**(9), 863–873.

Song, C. and Liu, K. (2018) Statistical degradation modeling and prognostics of multiple sensor signals via data fusion: A composite health index approach. *IISE Transactions*, **50**(10), 853–867.

Stich, S.U. (2018) Local SGD converges fast and communicates little. *arXiv preprint arXiv:1805.09767*.

Tan, A.Z., Yu, H., Cui, L. and Yang, Q. (2022) Towards personalized federated learning. *IEEE Transactions on Neural Networks and Learning Systems*, **33**, 1–17.

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

Tong, Q., Liang, G., Zhu, T. and Bi, J. (2020) Federated nonconvex sparse learning. *arXiv preprint arXiv:2101.00052*.

Van Erp, S., Oberski, D.L. and Mulder, J. (2019) Shrinkage priors for Bayesian penalized regression. *Journal of Mathematical Psychology*, **89**, 31–50.

Vanschoren, J. (2019) Meta-learning, in *Automated Machine Learning*, Springer Nature, Switzerland, pp. 35–61.

Vehtari, A., Gelman, A., Sivula, T., Jylänki, P., Tran, D., Sahai, S., Blomstedt, P., Cunningham, J.P., Schiminovich, D. and Robert, C.P. (2020) Expectation propagation as a way of life: A framework for Bayesian inference on partitioned data. *The Journal of Machine Learning Research*, **21**(17), 1–53.

Wang, J. and Joshi, G. (2021) Cooperative SGD: A unified framework for the design and analysis of local-update sgd algorithms. *Journal of Machine Learning Research*, **22**(213), 1–50.

Williams, C.K. and Rasmussen, C.E. (2006) *Gaussian Processes for Machine Learning*, Volume 2. MIT press Cambridge, MA.

Yu, H., Liu, Z., Liu, Y., Chen, T., Cong, M., Weng, X., Niyato, D. and Yang, Q. (2020) A fairness-aware incentive scheme for federated learning, in *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, Association for Computing Machinery, New York, NY, pp. 393–399.

Yuan, H. and Ma, T. (2020) Federated accelerated stochastic gradient descent. *Advances in Neural Information Processing Systems*, **33**, 5332–5344.

Yuan, H., Zaheer, M. and Reddi, S. (2021) Federated composite optimization, in *International Conference on Machine Learning*, Curran Associates Inc., Red Hook, NY, pp. 12253–12266.

Yue, X. and Kontar, R.A. (2021) Federated Gaussian process: Convergence, automatic personalization and multi-fidelity modeling. *arXiv preprint arXiv:2111.14008*.

Yue, X., Nouiehed, M. and Kontar, R.A. (2021) Gifair-fl: An approach for group and individual fairness in federated learning. *arXiv preprint arXiv:2108.02741*.

Zhang, Y., Duchi, J. and Wainwright, M. (2015) Divide and conquer kernel ridge regression: A distributed algorithm with minimax optimal rates. *The Journal of Machine Learning Research*, **16**(1), 3299–3340.

Zhang, Y. and Yeung, D.-Y. (2012) A convex formulation for learning task relationships in multi-task learning. *arXiv preprint arXiv: 1203.3536*.

Zhu, S., Ota, K. and Dong, M. (2021) Green AI for IIOT: Energy efficient intelligent edge computing for industrial internet of things. *IEEE Transactions on Green Communications and Networking*, **6**, 79–88.