Online Federated Multitask Learning

Rui Li
University at Buffalo
Buffalo, NY, USA
rli35@buffalo.edu

Fenglong Ma
Pennsylvania State University
University Park, PA, USA
fenglong@psu.edu

Wenjun Jiang
University at Buffalo
Buffalo, NY, USA
wenjunji@buffalo.edu

Jing Gao
University at Buffalo
Buffalo, NY, USA
jing@buffalo.edu

***Abstract*—With the popular use of mobile devices, it becomes increasingly important to conduct analysis on distributed data collected from multiple devices. Federated learning is a distributed learning framework which takes advantage of the training data and computational ability of scattered mobile devices to learn prediction models, and multi-task learning infers personalized but shared models among devices. Some recent work has integrated federated and multi-task learning, but such approaches may be impractical and inefficient in the online scenario, e.g., when new mobile devices keep joining the mobile computing system. To address this challenge, we propose OFMTL, an online federated multi-task learning algorithm, which learns the model parameters for the new device without revisiting the data of existing devices. The model parameters are derived by an effective way that combines the information inferred from local data and information borrowed from existing models. Through extensive experiments on three real datasets, we show that the proposed OFMTL framework achieves comparable accuracy to the existing algorithms but with much smaller computation, transmission and storage cost.**

***Index Terms*—Federated learning, multi-task relationship learning, online learning**

I. Introduction

With the rising popularity of mobile devices such as mobile phones and portable sensors, we observe explosive growth of data, which may be scattered on distributed device nodes. Traditional data analysis approaches require that all the data are transmitted to a central server. However, it could be inefficient or infeasible to transmit such decentralized data to the center due to privacy concerns, transmission cost, and limited centralized storage. Then how to conduct analysis on the huge amount of decentralized data without transmitting them to the center becomes an important task.

In this paper, we focus on classification, which is to infer models from training data scattered on different devices and use the models for prediction. To enable distributed classification (i.e., distributed learning), training data should be kept on individual mobile devices and the computation is conducted on local data collected on each device. To learn a global model, intermediate parameters are transmitted between mobile devices and the central server. This general distributed learning framework is called *federated learning* and a variety of algorithms have been proposed [1], [2].

Typically, a federated learning algorithm assumes that a global model is shared by all the local devices [3]. The objective is to infer this global model without transmitting data to the central server. However, in many applications, although there exist similarities among local models, they are not exactly the same. On the other hand, there are usually insufficient training data to derive an accurate local model. Therefore, ideally, we may want to derive local models that share some common characteristics by jointly learning from distributed data. This objective can be naturally captured by federated multi-task learning framework [4].

However, federated multi-task learning does not consider the online scenario, in which case new users or new devices appear and we aim to infer an effective model for the new user timely and update existing models in an efficient manner. As existing algorithms are derived on batch data, we have to rerun federated multi-task learning algorithms whenever new devices join the system, and this may lead to waste in both bandwidth and computational resources as well as long computational time. To overcome this limitation, we propose a novel algorithm that can efficiently and effectively derive model parameters for a new device based on its local data and existing models. We also show that the proposed algorithm achieves comparable accuracy to the existing algorithms but with much smaller computation,transmission and storage cost.

II. Problem Statement

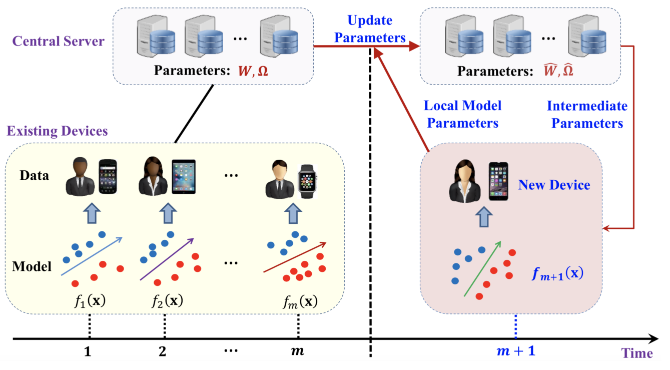


Fig. 1. Overall Framework for Online Federated Multi-Task Learning.

Consider a scenario that a set of users or devices collaborate together to learn personalized classifiers for some purpose, e.g., to detect daily human activity. The data are the sensory data (e.g., accelerometer, gyroscope) collected from users' mobile devices. Each user can manually label a small portion of the data. Here, we introduce a central server to coordinate the training process so that each user only needs to communicate with the central server. During the training process, the users require to keep their data on their own devices to avoid the leakage of private information. Since each user may demonstrate different patterns in their activities, they also require to learn personalized classification models instead of a common one. The problem that we aim to address in this paper is: *When a new device joins the system, without invoking the communication between the server and the current devices in the system, how to learn an accurate personalized classification model for this new device?*

Without loss of generality, we consider that there are already m devices in the system as shown in Figure 1. The dataset that the device t owns is denoted as $\{(\boldsymbol{x}_t^1, y_t^1), (\boldsymbol{x}_t^2, y_t^2), \cdots (\boldsymbol{x}_t^{n_t}, y_t^{n_t})\}$ $(t = 1 \cdots m)$, where $\boldsymbol{x}_t^i \in \mathbb{R}^d$ represents the i-th data, $y_t^i \in \{-1, +1\}$ is the corresponding label, and n_t is the number of data stored by this device. Using their own datasets, each device t can learn a local personalized model. The parameters of each local model can be represented by $\boldsymbol{w}_t \in \mathbb{R}^d$. We use $f_t(\boldsymbol{x}) = \boldsymbol{w}_t \cdot \boldsymbol{x}$ to represent the decision boundary. To guarantee the privacy of data, without uploading data to the central server, only local model parameters are sent to the central server. We use the weight matrix $\boldsymbol{W} := (\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_m) \in \mathbb{R}^{d\times m}$ to represent all the model parameters of current devices uploaded in the system and introduce a precision matrix $\boldsymbol{\Omega} \in \mathbb{R}^{m\times m}$ to model the relationships between each pair of devices. Both parameters are stored on the central server.

When a new device joins the system, the goal of **OFMTL** is to learn model parameter $\boldsymbol{w}_{m+1}$ for the new device and update the parameters from weight matrix $\boldsymbol{W}$ and the precision matrix $\boldsymbol{\Omega}$ to $\hat{\boldsymbol{W}} := (\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_m, \boldsymbol{w}_{m+1}) \in \mathbb{R}^{d\times(m+1)}$ and $\hat{\boldsymbol{\Omega}} \in \mathbb{R}^{(m+1)\times(m+1)}$, without the participation of any existing device in the system. As shown in Figure 1, the new device first trains its local model $f_{m+1}(\boldsymbol{x})$, and the parameters of this model is represented by $\boldsymbol{w}_{m+1}$, which is then sent to the central server. Based on the uploaded local parameters $\boldsymbol{w}_{m+1}$, the proposed model can update the parameters $\boldsymbol{W}$ and $\boldsymbol{\Omega}$, without revisiting existing devices. Using the updated parameters $\hat{\boldsymbol{W}}$ and $\hat{\boldsymbol{\Omega}}$, the new device updates its parameters again.

III. Methodology

In this section, we describe the proposed online federated multi-task learning framework OFMTL. We model this problem as an optimization problem where the personalized hyperplane of a new mobile device will be learned based on not only the training data of this device, but also the knowledge "borrowed" from others devices, without explicitly utilizing the raw training data of them.

A. Online Federated Multi-Task Learning

Multi-Task Relationship Learning. The proposed online federated multi-task learning framework inherits the spirit of traditional multi-task relationship learning (MTRL) [5]. The key idea of MTRL is to model the global relationships between the tasks (i.e., the personalized classification models of the devices) and formulate the learning problem as a convex optimization problem.

Mathematically, the MTRL is formulated as follows:

$$\begin{aligned}\min_{\boldsymbol{W},\boldsymbol{\Omega}} \quad & \sum_{t=1}^{m} \frac{1}{n_t} \sum_{i=1}^{n_t} l_t(\boldsymbol{w}_t^\top \boldsymbol{x}_t^i, y_t^i) + \frac{\lambda}{2} tr(\boldsymbol{W}\boldsymbol{\Omega}\boldsymbol{W}^\top),\\ s.t. \quad & tr(\boldsymbol{\Omega}^{-1}) = 1,\\ & \boldsymbol{\Omega}^{-1} \succeq 0.\end{aligned}$$

where l_t is the loss function for device t, which can be an arbitrary convex function. $\boldsymbol{W} := (\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_m) \in \mathbb{R}^{d\times m}$ is the weight matrix, and $\boldsymbol{\Omega} \in \mathbb{R}^{m\times m}$ is the precision matrix.

Assume that one new device tries to join the system with m existing users. We use $\boldsymbol{X}_{m+1}$ to denote the dataset of this new device, and $\boldsymbol{w}_{m+1}$ to denote the model parameter for this new device. After the new device joins the system, the new weight matrix can be expressed as $\hat{\boldsymbol{W}} := (\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_m, \boldsymbol{w}_{m+1})$ and the precision matrix can be expressed $\hat{\boldsymbol{\Omega}} \in \mathbb{R}^{(m+1)\times(m+1)}$. Notice that the first m columns in $\hat{\boldsymbol{W}}$ are fixed, and only $\boldsymbol{w}_{m+1}$ needs to be learned. Therefore, we formulate the proposed framework as a convex optimization problem as follows:

$$\begin{aligned}\min_{\boldsymbol{w}_{m+1},\hat{\boldsymbol{\Omega}}} \quad & \frac{1}{n_{m+1}} \sum_{i=1}^{n_{m+1}} l_{m+1}^i(\boldsymbol{w}_{m+1}^\top \boldsymbol{x}_{m+1}^i, y_{m+1}^i) + \frac{\lambda}{2} tr(\hat{\boldsymbol{W}}\hat{\boldsymbol{\Omega}}\hat{\boldsymbol{W}}^\top)\\ s.t. \quad & tr(\hat{\boldsymbol{\Omega}}^{-1}) = 1,\\ & \hat{\boldsymbol{\Omega}}^{-1} \succeq 0.\end{aligned} \tag{1}$$

To show the convexity of the above formula, we first demonstrate the convexity of the regularizer $tr(\hat{\boldsymbol{W}}\hat{\boldsymbol{\Omega}}\hat{\boldsymbol{W}}^\top)$. Actually, the regularizer has been proved to be joint convex w.r.t $\hat{\boldsymbol{W}}$ and $\hat{\boldsymbol{\Omega}}^{-1}$ when $\hat{\boldsymbol{\Omega}}^{-1}$ is a positive semidefinite matrix (satisfied by the second constraint) [5]. Since $\boldsymbol{w}_{m+1}$ is one column of $\hat{\boldsymbol{W}}$, the regularizer is also joint convex w.r.t $\boldsymbol{w}_{m+1}$ and $\hat{\boldsymbol{\Omega}}^{-1}$. Given the loss function and the constraints are both convex w.r.t all the variables, the proposed framework is convex w.r.t $\boldsymbol{w}_{m+1}$ and $\hat{\boldsymbol{\Omega}}^{-1}$.

B. Alternating Optimization Algorithm

Although Eq. (1) is convex w.r.t. $\boldsymbol{w}_{m+1}$ and $\hat{\boldsymbol{\Omega}}^{-1}$ jointly, optimizing the objective function w.r.t. all the variables simultaneously is still not easy. To solve this problem, we propose an alternating optimization algorithm. In each iteration of the algorithm, we first optimize $\boldsymbol{w}_{m+1}$ of the new device with fixed precision matrix $\hat{\boldsymbol{\Omega}}$, and then update the precision matrix $\hat{\boldsymbol{\Omega}}$ based on the model parameter $\boldsymbol{w}_{m+1}$ that we just learned in the first step and the model parameters $\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_m$ of all the previous devices.

Update $\boldsymbol{w}_{m+1}$. We first fix $\hat{\boldsymbol{\Omega}}$ and optimize the optimization problem w.r.t $\boldsymbol{w}_{m+1}$. Since the loss function $l_{m+1}^i(\cdot)$ can be seen as a function of $\boldsymbol{w}_{m+1}$, we can rewrite Eq. (1) as follows:

$$\begin{aligned}\min_{\boldsymbol{w}_{m+1},\boldsymbol{z}_{m+1}} \quad & \frac{1}{n_{m+1}} \sum_{i=1}^{n_{m+1}} l_{m+1}^i(-z_{m+1}^i) + \frac{\lambda}{2} tr(\hat{\boldsymbol{W}}\hat{\boldsymbol{\Omega}}\hat{\boldsymbol{W}}^\top).\\ s.t. \quad & \boldsymbol{w}_{m+1}^\top \cdot \boldsymbol{x}_{m+1}^i + z_{m+1}^i = 0.\end{aligned}$$

The Lagrangian of the above optimization problem is expressed as:

$$\begin{aligned}
&L(\boldsymbol{w}_{m+1},\boldsymbol{z}_{m+1},\boldsymbol{\alpha}_{m+1}) \qquad (2)\\
&=\frac{1}{n_{m+1}}\sum_{i=1}^{n_{m+1}}[l_{m+1}^{i}(-z_{m+1}^{i})-\alpha_{m+1}^{i}z_{m+1}^{i}]\\
&-\frac{1}{n_{m+1}}\sum_{i=1}^{n_{m+1}}\alpha_{m+1}^{i}\boldsymbol{w}_{m+1}^{\top}\boldsymbol{x}_{m+1}^{i}+\frac{\lambda}{2}tr(\hat{\boldsymbol{W}}\hat{\boldsymbol{\Omega}}\hat{\boldsymbol{W}}^{\top}),
\end{aligned}$$

where $-\frac{1}{n_{m+1}}\alpha_{m+1}^{i}$ are introduced as the Lagrange multipliers.

The Lagrangian dual function can be expressed as:

$$\begin{aligned}
&g(\boldsymbol{\alpha}_{m+1})\\
&=\inf_{\boldsymbol{w}_{m+1},\boldsymbol{z}_{m+1}}L(\boldsymbol{w}_{m+1},\boldsymbol{z}_{m+1},\boldsymbol{\alpha}_{m+1})\\
&=-\sup_{\boldsymbol{w}_{m+1}}[\frac{1}{n_{m+1}}\sum_{i=1}^{n_{m+1}}\alpha_{m+1}^{i}\boldsymbol{w}_{m+1}^{\top}\boldsymbol{x}_{m+1}^{i}-\frac{\lambda}{2}tr(\hat{\boldsymbol{W}}\hat{\boldsymbol{\Omega}}\hat{\boldsymbol{W}}^{\top})]\\
&-\frac{1}{n_{m+1}}\sum_{i=1}^{n_{m+1}}l_{m+1}^{i*}(-\alpha_{m+1}^{i})
\end{aligned}$$

where $l_{m+1}^{i*}(\cdot)$ is the conjugate of $l_{m+1}^{i}(\cdot)$ [6], which is defined as $l_{m+1}^{i*}(-\alpha_{m+1}^{i}) := \sup_{-z_{m+1}^{i}}(z_{m+1}^{i}\alpha_{m+1}^{i} - l_{m+1}^{i}(-z_{m+1}^{i}))$. In the above formula, $tr(\hat{\boldsymbol{W}}\hat{\boldsymbol{\Omega}}\hat{\boldsymbol{W}}^{\top})$ can be extended, and the terms that relate to $\boldsymbol{w}_{m+1}$ are $2\sum_{i=1}^{m}\hat{\Omega}_{i\ m+1}\boldsymbol{w}_{i}^{\top}\boldsymbol{w}_{m+1}+\hat{\Omega}_{m+1\ m+1}\boldsymbol{w}_{m+1}^{\top}\boldsymbol{w}_{m+1}$. If we assume $\frac{\partial L}{\partial \boldsymbol{w}_{m+1}}=0$, we have:

$$\boldsymbol{w}_{m+1}=\frac{\boldsymbol{b}_{m+1}-\sum_{i=1}^{m}\hat{\Omega}_{i\ m+1}\boldsymbol{w}_{i}}{\hat{\Omega}_{m+1\ m+1}} \qquad (3)$$

with

$$\boldsymbol{b}_{m+1}=\frac{1}{\lambda n_{m+1}}\sum_{i=1}^{n_{m+1}}\alpha_{m+1}^{i}\boldsymbol{x}_{m+1}^{i}.$$

In Eq. (3), obviously, $\boldsymbol{b}_{m+1}$ can be computed from the new mobile device directly since it only involves data $\boldsymbol{x}_{m+1}^{i}$ which resides on the new mobile device. While the other two terms $\sum_{i=1}^{m}\hat{\Omega}_{i\ m+1}\boldsymbol{w}_{i}$ and $\hat{\Omega}_{m+1\ m+1}$ can only be obtained from the central server since $\hat{\boldsymbol{\Omega}}$ and $\boldsymbol{w}_{i}$ with $i=1,2,\cdots,m$ only reside on the central server. In this way, instead of transmitting raw data residing on the new mobile device to the central server, we only need to transmit two variables $\sum_{i=1}^{m}\hat{\Omega}_{i\ m+1}\boldsymbol{w}_{i}$ and $\hat{\Omega}_{m+1\ m+1}$ from the central server to the new mobile device, which eliminates the privacy concern and dramatically reduces the communication overhead.

From Eq. (3), in order to compute $\boldsymbol{w}_{m+1}$, we have to compute $\boldsymbol{\alpha}_{m+1}$. Then we go back to $g(\boldsymbol{\alpha}_{m+1})$ and maximize the dual function

$$\max_{\boldsymbol{\alpha}_{m+1}} g(\boldsymbol{\alpha}_{m+1}),$$

which is equivalent to

$$\begin{aligned}
&\min_{\boldsymbol{\alpha}_{m+1}}\{\frac{1}{n_{m+1}}\sum_{i=1}^{n_{m+1}}l_{m+1}^{i*}(-\alpha_{m+1}^{i})+\frac{1}{n_{m+1}}\sum_{i=1}^{n_{m+1}}\alpha_{m+1}^{i}\boldsymbol{w}_{m+1}^{\top}\boldsymbol{x}_{m+1}^{i} \qquad (4)\\
&-\frac{\lambda}{2}tr(\hat{\boldsymbol{W}}\hat{\boldsymbol{\Omega}}\hat{\boldsymbol{W}}^{\top})\}.
\end{aligned}$$

We can plug Eq. (3) into Eq. (4). To accelerate the convergence speed, we use the stochastic dual coordinate ascent method to compute the $\boldsymbol{\alpha}_{m+1}$ [7]. Given that $\Delta\boldsymbol{\alpha}_{m+1}^{\top} := (\Delta\alpha_{m+1}^{1}\Delta\alpha_{m+1}^{2}\cdots\Delta\alpha_{m+1}^{n_{m+1}})\in\mathbb{R}^{n_{m+1}}$ is a small change of $\boldsymbol{\alpha}_{m+1}$, we can approximate the dual problem with a quadratic function and optimize the following optimization problem:

$$\min_{\Delta\boldsymbol{\alpha}_{m+1}} g(\Delta\boldsymbol{\alpha}_{m+1},\boldsymbol{\alpha}_{m+1},\boldsymbol{w}_{m+1}) \qquad (5)$$

with

$$\begin{aligned}
&g(\Delta\boldsymbol{\alpha}_{m+1},\boldsymbol{\alpha}_{m+1},\boldsymbol{w}_{m+1})\\
&=\frac{1}{n_{m+1}}\sum_{i=1}^{n_{m+1}}l_{m+1}^{i*}(-\alpha_{m+1}^{i}-\Delta\alpha_{m+1}^{i})+\frac{1}{n_{m+1}}\boldsymbol{w}_{m+1}^{\top}\boldsymbol{X}_{m+1}\Delta\boldsymbol{\alpha}_{m+1}\\
&+\frac{1}{2\lambda n_{m+1}^{2}\hat{\boldsymbol{\Omega}}_{m+1\ m+1}}\Delta\boldsymbol{\alpha}_{m+1}^{\top}\boldsymbol{X}_{m+1}^{\top}\boldsymbol{X}_{m+1}\Delta\boldsymbol{\alpha}_{m+1}+C(\boldsymbol{\alpha}_{m+1}),
\end{aligned}$$

where $C(\boldsymbol{\alpha}_{m+1})$ represents a function which is only related to $\boldsymbol{\alpha}_{m+1}$. When we compute $\Delta\boldsymbol{\alpha}_{m+1}$, it can be removed from $g(\Delta\boldsymbol{\alpha}_{m+1},\boldsymbol{\alpha}_{m+1},\boldsymbol{w}_{m+1})$. From Eq. (5), it is obvious to see that $\Delta\boldsymbol{\alpha}_{m+1}$ can be computed on mobile device $m+1$ since $\boldsymbol{w}_{m+1}$ and $\boldsymbol{X}_{m+1}$ all reside on mobile device $m+1$. The details about the local SDCA (stochastic dual coordinate ascent) algorithm will be presented in Algorithm 2.

Update $\hat{\boldsymbol{\Omega}}$. After updating $\boldsymbol{w}_{m+1}$, we fix it and then update $\hat{\boldsymbol{\Omega}}$. Assume that there are m devices existed in the system, and the corresponding precision matrix is $\boldsymbol{\Omega}\in\mathbb{R}^{m\times m}$. When a new device joins the system, we initialize the new precision matrix $\hat{\boldsymbol{\Omega}}$ as

$$\hat{\boldsymbol{\Omega}} := \begin{pmatrix} \frac{m}{m+1}\boldsymbol{\Omega} & \boldsymbol{0} \\ \boldsymbol{0}^{\top} & \frac{1}{m+1} \end{pmatrix}. \qquad (6)$$

In this way, $\hat{\boldsymbol{\Omega}}$ satisfies the two constraints in Eq. (1). To update $\hat{\boldsymbol{\Omega}}$, we need to solve the convex optimization problem (i.e., Eq. (1)) over $\hat{\boldsymbol{\Omega}}$ with fixed $\boldsymbol{w}_{m+1}$ on the central server.

C. Model Parameters Retraining

Since the join of new devices brings more training data, which may potentially benefit the early devices. Retraining the model parameters may help the early devices to learn a more accurate decision boundary. Assume that at the beginning, there are m_0 devices in the system ($m_0 << n$). When a new device joins the system, we conduct the proposed OFMTL to learn the model parameter for it, which takes $O(1)$ communication overhead. We do not conduct the model parameter retraining until the number of the newly added devices in the system reaches a constant ratio (denoted as η, $\eta > 0$) of m, where m is the number of devices in the current system when we retrain the model parameters. m is initialized with m_0. The model parameters retraining can be done with a multi-task relationship learning algorithm, such as MOCHA [4]. After the model parameter retraining, we update m to be $(1+\eta)m$ and continue from the first procedure. When the number of users in the system reaches n, the total communication overhead is still $O(n)$.

D. Algorithm

The pseudo code of the proposed OFMTL is demonstrated in Algorithm 1. In Algorithm 1, the derivation of $\Delta\boldsymbol{\alpha}_{m+1}$ is based on the stochastic dual coordinate ascent algorithm [7]. Here, we summarize the stochastic dual coordinate ascent algorithm on the mobile device $m+1$ in Algorithm 2 in which $\boldsymbol{e}_i \in \mathbb{R}^{n_{m+1}}$ is a one-hot basis vector whose the i-th element is 1, and the other elements are 0.

Algorithm 1 Online Federated multi-task Learning (OFMTL)

```
Input:
    Existing devices in the system: Data $\boldsymbol{X}_t$ and fixed model parameter $\boldsymbol{w}_t$ for $t = 1, 2, \cdots, m$, precision matrix $\boldsymbol{\Omega} \in \mathbb{R}^{m \times m}$;
    New Device: Data $\boldsymbol{X}_{m+1}$;
Initialize:
    $\boldsymbol{\alpha}_{m+1}^{(0)} := \boldsymbol{0} \in \mathbb{R}^{n_{m+1}}$, $\hat{\boldsymbol{\Omega}}^{(0)} := \begin{pmatrix} \frac{m}{m+1}\boldsymbol{\Omega} & \boldsymbol{0} \\ \boldsymbol{0}^\top & \frac{1}{m+1} \end{pmatrix} \in \mathbb{R}^{(m+1)\times(m+1)}$, $\boldsymbol{w}_{m+1}^{(0)} := \boldsymbol{0} \in \mathbb{R}^d$;
1:  for $i = 0, 1, \cdots$ do
2:      for $t = 0, 1, \cdots$ do
3:          // Solving local subproblem;
4:          Derive $\Delta\boldsymbol{\alpha}_{m+1}^{(t)}$ from Eq. (5);
5:          // Local update;
6:          $\boldsymbol{\alpha}_{m+1}^{(t)} \leftarrow \boldsymbol{\alpha}_{m+1}^{(t-1)} + \Delta\boldsymbol{\alpha}_{m+1}^{(t)}$;
7:          Update $\boldsymbol{w}_{m+1}^{(t)}$ according to Eq. (3);
8:      end for
9:      The new device sends $\boldsymbol{w}_{m+1}$ to the server;
10:     $\hat{\boldsymbol{\Omega}}^{(p)} \leftarrow$ solve problem on server for fixed $\hat{\boldsymbol{W}}$, update $\hat{\boldsymbol{\Sigma}} = \hat{\boldsymbol{\Omega}}^{-1}$, and server sends $\sum_{i=1}^{m} \hat{\Omega}_{i\ m+1}\boldsymbol{w}_i$ and $\hat{\Omega}_{m+1\ m+1}$ to the new device.
11: end for
```

Algorithm 2 Stochastic dual coordinate ascent to derive $\Delta\boldsymbol{\alpha}_{m+1}$

```
Input:
    $N \geqslant 1$, $\boldsymbol{\alpha}_{m+1}$, $\boldsymbol{w}_{m+1}$, and data $\boldsymbol{X}_{m+1}$
Initialize:
    $\Delta\boldsymbol{\alpha}_{m+1} := \boldsymbol{0} \in \mathbb{R}^{n_{m+1}}$
1:  for $n = 0, 1, \cdots, N$ do
2:      choose $i \in \{1, 2, \cdots, N\}$ randomly
3:      $\Delta\alpha_{m+1}^i \leftarrow \min_{\Delta\alpha_{m+1}^i} (\boldsymbol{\Delta\alpha}_{m+1} + \Delta\alpha_{m+1}^i \boldsymbol{e}_i, \boldsymbol{\alpha}_{m+1}, \boldsymbol{w}_{m+1})$
4:      $\boldsymbol{\Delta\alpha}_{m+1} \leftarrow \boldsymbol{\Delta\alpha}_{m+1} + \Delta\alpha_{m+1}^i \boldsymbol{e}_i$
5:  end for
```

The proposed algorithm (Algorithm 1) is composed of the following three parts: (1) Computing the dual variable $\boldsymbol{\alpha}_{m+1}$ on the new mobile device $m+1$ using the local SDCA algorithm with objective function (5), which is in the pseudo code line 4; (2) Updating $\boldsymbol{w}_{m+1}$ with fixed precision matrix $\hat{\boldsymbol{\Omega}}$ and utilizing the dual variable $\boldsymbol{\alpha}_{m+1}$ computed in the first part, which is in the pseudo code line 7; (3) Updating the precision matrix $\hat{\boldsymbol{\Omega}}$ on the central server based on the model parameter $\boldsymbol{w}_{m+1}$ computed on the second part and all the previous model parameters $\boldsymbol{w}_1, \boldsymbol{w}_2, \cdots, \boldsymbol{w}_m$ saved on the server, which is in the pseudo code line 10. Among theses three parts, (1) and (2) are implemented on the new mobile device, while (3) is implemented on the central server.

IV. Experiments

In this section, we will introduce the experimental settings and then demonstrate the performance of the proposed algorithm on three real-world datasets.

A. Experimental Settings

There are three **datasets** used to validate the performance of the proposed algorithm.

(1) Human Activity Recognition (HAR) [8]. This dataset contains the readings of accelerometer and gyroscope of smartphones when each participant performs six activities. And we try to separate the activities of sitting and standing. The total number of features is 561, and the number of participants is 30. Each participant is regarded as a separate task.

(2) Eating Recognition via Google Glass (GLEAM) [9]. This dataset consists of the data collected from Google Glass spanning eating, brief walks and other activities from 38 participants for two hours. We generate a 144-element vector for each data, including 5 statistical and 3 spectral features for 6 sensors with 3 axes. Every participant is considered as a task, and we make predictions between eating and other activities.

(3) Eating Habits Monitoring (EHM) [10]. This dataset consists of acoustic signals collected from wearable necklace-like devices that recognize the eating habits of human beings. There are 4 participants performing activities: eating chips and drinking water. Each participant can be regarded as a separate task. We extract 20 features in the time domain and frequency domain as mentioned in [10].

Baselines. To fairly evaluate the performance of the proposed model OFMTL, we adopt three baselines.

(1) Global SVM. In this setting, we assume that all the raw data storing on different mobile devices are transmitted to the server. We then train a classifier using SVM (Support Vector Machines) on the server, i.e., the global model. All the mobile devices (i.e., local servers) share the same parameters.

(2) Local SVM. For the local model, we assume that there is no central server, and each mobile device only utilizes the data residing on itself to train its local SVM. For the online setting, if a new mobile device joins the system, the new device simply utilizes its own local data to train the local model.

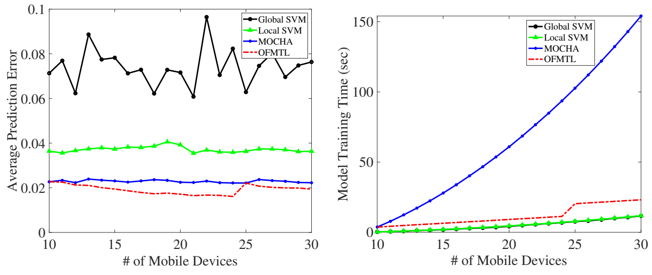
(3) MOCHA [4]. MOCHA is the state-of-the-art federated multi-task learning algorithm. When learning the classifier, MOCHA considers both the server and each local mobile device at the same time. Each mobile device is considered as a task, and intermediate parameters are transmitted between all the local mobile devices and the central server. However, MOCHA cannot deal with the online setting. In the experi-

ments, when a new mobile device is added to the system, we rerun the MOCHA algorithm on all the data.

Implementation Details. We assume that there are m_0 users already in the current system, and new users join the system one by one. For the aforementioned three datasets, we choose different values of m_0, as the number of participants is different. For the HAR and GLEAM datasets, we set m_0 as 10, and for the EHM dataset, $m_0 = 1$. The proposed OFMTL needs to update the overall model when the number of new devices exceeds a threshold. In the experiments, this threshold is set as 1.5 times of the number of current devices.

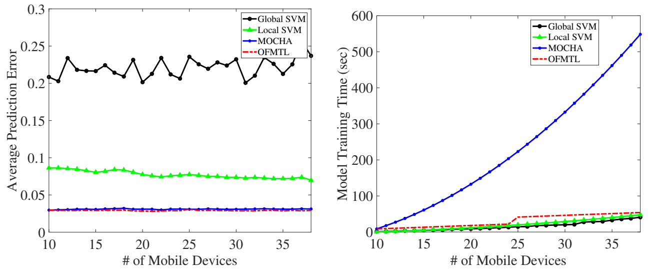
Evaluation Metric. We use average prediction error and model training time as the evaluation metric. Average prediction error refers to the average error rate of the classifiers based on their predictions. Meanwhile, model training time refers to the computing time our system takes for model updating when the number of devices gradually increases from m_0 to m. Each time when a new device joins the system, for our proposed OFMTL, the increment of model training time equals the time we perform Algorithm 1. While for MOCHA, it equals the time we retrain the model from scratch on all existing devices. The model training time is measured by seconds.

B. Performance Evaluation



(a) HAR Average Prediction Error (b) HAR Model Training Time

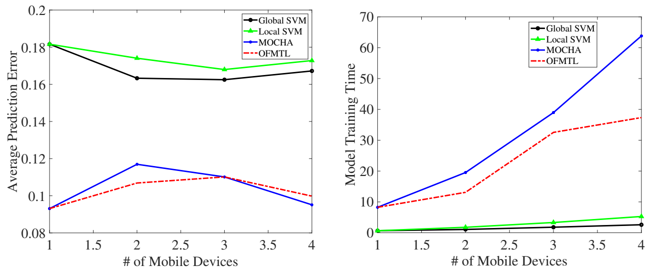
Fig. 2. Average Prediction Error and Model Training Time V.S. the Number of Mobile Devices in the system on HAR Dataset.



(a) GLEAM Average Prediction Error (b) GLEAM Model Training Time

Fig. 3. Average Prediction Error and Model Training Time V.S. the Number of Mobile Devices in the system on GLEAM Dataset.

Figure 2-4 shows the average prediction error and model training time with respect to the number of mobile devices in the system on three datasets. In (a), X-axis represents the number of mobile devices in the current system, and Y-axis denotes the average prediction error. We can observe that the proposed OFMTL achieves comparable or even better performance compared with the state-of-art federated multi-task learning approach MOCHA, which is the batch algorithm using all the data.



(a) EHM Average Prediction Error (b) EHM Model Training Time

Fig. 4. Average Prediction Error and Model Training Time V.S. the Number of Mobile Devices in the system on EHM Dataset.

On the HAR and GLEAM datasets, the global model (Global SVM) achieves the worst performance. This is reasonable because the data stored on different local mobile devices may have different characteristics or patterns. Without considering these differences, just putting all the data together to train a global model and then testing on each local mobile device may lead to bad overall performance. The local model (Local SVM) performs better than the global model, which shows that Local SVM can capture the characteristics of data residing on each mobile device.

On the EHM dataset, the local model performs slightly better than the global model. This is reasonable because the data collected from different users are the sound of drinking water and eating chips, and even for different participants, the acoustic signals may share similar patterns. When training Global SVM, the amount of data is much larger than that of data used for training Local SVM. Thus, the global model achieves better performance.

In Figure 2(a), 3(a) and 4(a), we can observe that the proposed online federated multi-task learning algorithm OFMTL achieves similar performance as MOCHA. On the HAR dataset, it is even better than MOCHA when there are 12 to 25 devices in the system. This is because MOCHA can not guarantee to reach the global optimal point. When a new device joins the system, OFMTL only learns a classifier for the new device, the classifiers of previous devices will be fixed. Meanwhile, MOCHA retrains the model with all mobile devices. For those devices with fixed classifiers in OFMTL, retraining the models on these devices with MOCHA may even impair the performance. This demonstrates the effectiveness of the proposed OFMTL.

On the HAR and GLEAM datasets, the performance of OFMTL equals to that of MOCHA when there are 15 extra mobile devices added into the system, i,e, the total number of mobile devices is 25. That is because with 10 initial mobile devices, when 15 new mobile devices join the system one by one, which is up to the threshold (i.e., 1.5 times of the number of original mobile devices), the proposed OFMTL will update the overall model. This is the same as running MOCHA. Thus, the performance of them is the same. The same observation can been found on the EHM dataset when the number of mobile devices is 3.

Figure 2(b), 3(b) and 4(b) shows the model training time with respect to the number of devices in the system. We can observe that MOCHA takes the most training time, local and global SVM take the least, and the model training time of OFMTL is in the middle. The model training time of MOCHA grows fastest. That's because when a new mobile device joins the system, MOCHA retrains the model with all the mobile devices. Each local device learns a local model, and then sends it to the central server. The central server updates the parameters and sends them back to each local server. This procedure needs to repeat until MOCHA converges. While for OFMTL, the training time grows linearly and slowly as more devices join the system. This is consistent with the analysis that the total communication overhead is $O(n)$ in part C of Section III. And it is because that OFMTL only learns a classifier for the new device with its data and the stored parameters, instead of learning classifiers for all previously joined mobile devices. Meanwhile, we can also note that there is a jump when the 25th device joins the system for HAR and GLEAM dataset. This is because OFMTL reaches the threshold and it invokes MOCHA. The same phenomenon is also observed at the 3rd device for EHM dataset.

V. Related Work

Federated learning was first proposed in [11], which enables the mobile devices to learn a shared model without transmitting the raw data residing on separate mobile devices to the server [1]–[3], [12]. The state-of-the-art federated learning algorithm is FederatedAverging [3], in which case all devices share a same model and ignore the difference between users' habits. In contrast, our algorithm designs personalized model for every device in the system.

Multi-Task Learning is to learn individual models for multiple related tasks. The most relevant categories are online multi-task learning [13], [14] and distributed multi-task learning [15], [16]. However, all previous online algorithms focus on online data, which is different from the online task setting studied in this paper. Distributed multi-task learning algorithm DMTRL proposed in [15] assumes all devices solve the sub-problem at the same scale, which contradicts the large variability of data and system in federated learning scenario.

Federated Multi-Task Learning was firstly proposed in [4]. Smith et al. found that distributed multi-task learning is suitable to handle the federated learning problem. They proposed MOCHA algorithm which provides a personalized model for every mobile device by only transmitting intermediate variables between mobile devices and the server. And MOCHA allows devices to solve the sub-problems at different accuracy. However, MOCHA is unable to handle the ubiquitous setting that new devices keep joining the system.

VI. Conclusions

In this paper, we studied an ubiquitous problem found in many intelligent mobile computing scenarios, which is online federated multitask learning. In this task, we aim to infer classification models for individual users or devices who join the mobile computing system sequentially. We proposed an effective algorithm named OFMTL, which infers a personalized model for each new user or device without revisiting the data of existing devices. The relationships among data residing on different devices are captured. The proposed model updates the relationship matrices and derives the new model efficiently. We conducted experiments on three real-world datasets, and results show that OFMTL achieves compatible results comparing to the batch federated multitask learning algorithm with much less resource and computation costs.

Acknowledgements

This work is sponsored by NSF IIS-1553411. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] B. McMahan and D. Ramage, "Federated learning: Collaborative machine learning without centralized training data," *Google Research Blog*, vol. 3, 2017.

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

[3] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial Intelligence and Statistics*, 2017, pp. 1273–1282.

[4] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4424–4434.

[5] Y. Zhang and D.-Y. Yeung, "A convex formulation for learning task relationships in multi-task learning," in *Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence*, 2010, pp. 733–742.

[6] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[7] S. Shalev-Shwartz and T. Zhang, "Stochastic dual coordinate ascent methods for regularized loss minimization," *Journal of Machine Learning Research*, vol. 14, no. Feb, pp. 567–599, 2013.

[8] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. Reyes-Ortiz, "A public domain dataset for human activity recognition using smartphones," in *21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2013, pp. 437–442.

[9] S. A. Rahman, C. Merck, Y. Huang, and S. Kleinberg, "Unintrusive eating recognition using google glass," in *2015 9th International Conference on Pervasive Computing Technologies for Healthcare*, 2015, pp. 108–111.

[10] Y. Bi, W. Xu, N. Guan, Y. Wei, and W. Yi, "Pervasive eating habits monitoring and recognition through a wearable acoustic sensor," in *Proceedings of the 8th International Conference on Pervasive Computing Technologies for Healthcare*, 2014, pp. 174–177.

[11] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *Private Multi-Party Machine Learning, NIPS*, 2016.

[12] K. Bonawitz, H. Eichner, W. Grieskamp, D. Huba, A. Ingerman, V. Ivanov, C. M. Kiddon, J. Konečnỳ, S. Mazzocchi, B. McMahan *et al.*, "Towards federated learning at scale: System design," in *SysML Conference*, 2019.

[13] A. Saha, P. Rai, H. DaumÃ, S. Venkatasubramanian *et al.*, "Online learning of multiple tasks and their relationships," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 643–651.

[14] G. Li, S. C. Hoi, K. Chang, W. Liu, and R. Jain, "Collaborative online multitask learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 8, pp. 1866–1876, 2013.

[15] S. Liu, S. J. Pan, and Q. Ho, "Distributed multi-task relationship learning," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 937–946.

[16] J. Wang, M. Kolar, and N. Srerbo, "Distributed multi-task learning," in *Artificial Intelligence and Statistics*, 2016, pp. 751–760.