FEDERATED ALGORITHM WITH BAYESIAN APPROACH: OMNI-FEDGE

Sai Anuroop Kesanapalli

Department of Computer Science and Engineering
Indian Institute of Technology Dharwad
Karnataka, India - 580011

B. N. Bharath

Department of Electrical Engineering
Indian Institute of Technology Dharwad
Karnataka, India - 580011

ABSTRACT

In this paper, we consider the problem of Federated Learning (FL) under non-i.i.d data setting. We provide an improved estimate of the empirical loss at each node by using a weighted average of losses across nodes with a penalty term. These uneven weights to different nodes are assigned by taking a novel Bayesian approach to the problem where the problem of learning for each device/node is cast as maximizing the likelihood of a joint distribution. This joint distribution is for losses of nodes obtained by using data across devices for a given neural network of a node. We then provide a PAC learning guarantee on the objective function which reveals that the true average risk is no more than the proposed objective and the error term. We leverage this guarantee to propose an algorithm called Omni-Fedge. Using MNIST and Fashion MNIST datasets, we show that the performance of the proposed algorithm is significantly better than existing algorithms.

***Index Terms*—** Federated Learning, Neural Network, Bayesian Approach, Distributed Machine Learning, PAC Learning.

1. INTRODUCTION

A new paradigm in machine learning called *Federated Learning* (FL) enables edge-devices to collaboratively learn a shared prediction model while keeping all the training data securely at the device [1, 2]. There have been numerous research works in FL leading to new applications and algorithms [3, 4, 5]. It is particularly relevant for many wireless applications, especially in the context of fifth generation (5G) networks [3]. Implementing FL in practice imposes several challenges such as *stragglers*, non-i.i.d data residing at heterogeneous edge-devices with varying computation power, privacy concerns, and high communication cost between edge-devices and a Federating Server (FS) [6]. This paper primarily presents a systematic approach backed by theory to address the problem of improving FL performance with non-i.i.d data. A well known algorithm in the field of FL is called Federated Averaging (FedAvg) [7]. This involves exchange of Stochastic Gradient (SG) information computed using local data to a central server, where an average is performed. This average is sent back to all the nodes, where a SG Descent (SGD) is performed to update the neural network weights. Although, simple, it is known to diverge under non-i.i.d setting [2]. Further, this approach returns a single model for all the nodes. The authors in [8] consider the problem of Multi-task Learning (MtL) in terms of finding Pareto optimal solutions. However, the question that arises is how do we know a priori that there exists a trade-off between the tasks? Also, the paper does not provide any theoretical guarantee on the performance. The concept of task-dependent uncertainty in a multi-task setting is considered in [9]. They show that the task uncertainty captures the relative confidence between tasks, reflecting the uncertainty inherent to the regression or classification task. They derive a multi-task loss function based on maximising the Gaussian likelihood with homoscedastic uncertainty. Although, the experimental results are promising, the algorithm is heuristic, and there are no guarantees proved.

In [10], the authors introduce two novel strategies to reduce communication costs arising due to heterogeneity of edge networks viz., lossy compression and Federated Dropout. A new compression framework that is specifically designed to meet the requirements of the federated learning environment using sparse ternary compression is proposed in [11]. Hao et al. [12], propose an efficient and privacy-enhanced FL (PEFL) scheme for industrial artificial intelligence. A large body of existing work assumes that the data is i.i.d, and hence the stochastic gradient obtained using samples from the nodes results in an unbiased estimate. However, this assumption is not true in many applications, and the performance degradation of the algorithms based on i.i.d assumption can be severe [6]. Typically, the co-efficient in the loss function is chosen in proportion to the data residing at the device [13] or the "worst case co-efficient" as in the case of agnostic FL [14]. This may not efficiently handle the non-i.i.d data very well. The authors in [6] proposed a method to overcome this statistical challenge of non-i.i.d data. Another potential solution for this is to use Multi-task Learning (MtL) methods such as MOCHA as proposed in [15]. However, the approach taken in the MtL is heuristic based, and hence

the penalty term, although appropriate for many applications, there are no general guarantees. A systematic approach to MtL is provided in [16]. But the algorithm is centralized, meaning the entire data needs to be communicated, and further, the convergence guarantee is not provided.

In our work, we consider a federated setup (see Section 2) with N devices/nodes and a central server with a goal of finding device application specific neural network. An improved estimate of the empirical loss at each node is obtained by using a weighted average of losses across nodes with a penalty term. These uneven weights to different nodes are obtained by taking a Bayesian approach to the problem. In particular, we motivate the approach by assuming that the joint distribution of losses of node i, $i = 1, 2, \ldots, N$ obtained by using data across devices for a given neural network of node i follows an independent exponential distribution with certain rate (or weight). The problem of learning amounts to maximizing this joint distribution (see Section 2.1); which results in the objective that is a weighted sum of losses with penalty. We then provide a Probably Approximately Correct (PAC) learning guarantee on the objective function (see Section 3) that we use to design our algorithm called Omni-Fedge (see Section 4). Finally, we present some results on the performance of Omni-Fedge using MNIST and Fashion MNIST data-sets (see Section 5). We show that the proposed algorithm not only outperforms the existing federated algorithms but also requires less communication rounds to achieve a given accuracy.

2. PROBLEM SETTING

We consider a federated system with N edge-devices/nodes that can communicate with one central server. Each edge-device $i = 1, 2, \ldots, N$ is assumed to have collected n_i data points with j-th feature vector denoted by $X_{ij} \in \mathcal{X}$ with the corresponding label $y_{ij} \in \mathcal{Y}$, $j = 1, 2, \ldots, n_i$. Let $z_i := \{z_{ij} = (X_{ij}, y_{ij}) : X_{ij} \in \mathcal{X}, y_{ij} \in \mathcal{Y}, j = 1, 2, \ldots, n_i\}$. We assume that the data points are independent but not necessarily identically distributed across edge-devices. Further, we assume that the data at edge-device $i = 1, 2, \ldots, N$ is sampled from a distribution $\mathcal{D}_i$. The task of each edge-device is to find a neural network denoted by weights $\boldsymbol{\theta} \in \Theta \subseteq \mathcal{R}^d$ that results in the minimum average risk. The risk is defined in terms of a loss function $L_i : \mathcal{Z} \times \boldsymbol{\theta} \to \mathbb{R}^+$, $i = 1, 2, \ldots, N$, where $\mathcal{Z} := \mathcal{X} \times \mathcal{Y}$. Here, the loss function is assumed to be bounded by $l_{max} > 0$. Given this loss as a metric, the edge-device i would like to solve the following optimization problem

$$\min_{\boldsymbol{\theta}} \mathbb{E}_{z_i \sim \mathcal{D}_i} \left\{ L_i(z_i, \boldsymbol{\theta}) \right\}. \tag{1}$$

Since the distribution is unknown, a natural way is to use an estimate of $\mathbb{E}_{z_i \sim \mathcal{D}_i} \{l_i(z_i, \boldsymbol{\theta})\}$ as a proxy in (1). An estimate is given by $\hat{\mathbb{E}} L_i(z_i, \boldsymbol{\theta}) := \frac{1}{n_i} \sum_{j=1}^{n_i} L_i(z_{ij}, \boldsymbol{\theta})$. However, if the data across edge-devices are statistically closer, it is advantageous to exploit it to get a better estimate, and hence a better overall performance. A well known algorithm called FedAvg computes the neural network weights locally using the estimate in (1), and the central server computes an average of the weights, and broadcasts it to all the edge devices. This has number of disadvantages. First, it gives equal weight to all the edge devices, which may result in sub optimal performance. Further, all the edge devices need to use the same neural network, which may not be tuned to its application. A natural way of overcoming this is to put uneven weights for different nodes while computing an estimate of the average loss. A motivation for the weighing scheme proposed in the paper is provided next.

2.1. Motivation and Problem Statement

The motivation for the objective proposed in this paper comes from the Bayesian approach to the problem. Given the data set z_{jk}, $k = 1, 2, \ldots, n_k$, the loss function at the edge device $j = 1, 2, \ldots, N$ using neural network with weights $(\boldsymbol{\theta}_i, \boldsymbol{\theta}^{(sh)})$ be denoted by $L_{ijk} := L_j(z_{jk}, \boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)})$, $i = 1, 2, \ldots, N$. Here, the neural network weights are divided into two parts, viz, shared and task specific. The shared neural network weights will be learnt in a manner that minimizes the overall error while $\boldsymbol{\theta}_i$ will be fine tuned to meet edge device specific requirements. The two neural networks are concatenated to obtain the full neural network. For a given edge device i, conditioned on weights $\boldsymbol{\omega_i} := (\omega_{i1}, \omega_{i2}, \ldots, \omega_{iN}) \in \mathbb{R}_+^N$, assume that the joint distribution of these losses $f(L_{ijk} : j = 1, 2, \ldots, N, k = 1, 2, \ldots, n_j | \boldsymbol{\omega}_i)$ are drawn independently according to $\Pi_{j=1}^N \Pi_{k=1}^{n_j} f(l_{ijk} | \boldsymbol{\omega}_i)$, where $f(l_{ijk} | \boldsymbol{\omega}_i) = \omega_{ij} \exp\{-\omega_{ij} L_{ijk}\}$. Note that the joint distribution of the losses depends on the neural network weights. For brevity, we use $Q_{\boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)}}$ and $P_{\boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)}}$, $i = 1, 2, \ldots, N$ to denote the probability measure corresponding to the above independent exponential distribution and the true joint distribution, respectively. Note that the distribution $P_{\boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)}}$ is induced by $\mathcal{D}_i$. Using this in the expression for the joint distribution, we get $\Pi_{j=1}^N \Pi_{k=1}^{n_j} \omega_{ij} \exp\left\{ - \sum_{j=1}^N \sum_{k=1}^{n_j} \omega_{ij} L_{ijk} \right\}$. In this case, the problem of learning amounts to finding $\boldsymbol{\theta}^{(i)}$, $\boldsymbol{\theta}^{(sh)}$ and $\boldsymbol{\omega}_i$ for all the devices that maximize the above likelihood. In other words, this results in the following problem

$$\min_{\boldsymbol{\omega_i}, \boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)}} \sum_{j=1}^N \omega_{ij} \hat{\mathbb{E}} L_j(z_j, \boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)}) - \sum_{j=1}^N \log \omega_{ij}, \tag{2}$$

where $\hat{\mathbb{E}} L_j(z_j, \boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)}) := \frac{1}{n_j} \sum_{k=1}^{n_j} L_{ijk}$ is the average loss at the j-th node using the neural network of the i-th node, i.e, $(\boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)})$. The second term above can be thought of as a penalty term. The performance obtained using the neural network obtained by solving the above problem in comparison with the problem in (1) needs to be investigated, which is the essence of the following section.

3. THEORETICAL GUARANTEES

In this section, we provide a Probably Approximately Correct (PAC) learning guarantee on the objective function that will be used to design our algorithm. Towards stating our main result, we need the following definition. A few notations are in order. We use $\tilde{L}_{ijk} \sim Q_{\boldsymbol{\theta}^{(i)},\boldsymbol{\theta}^{(sh)}}$ to denote that the random variable is sampled from the distribution $Q_{\boldsymbol{\theta}^{(i)},\boldsymbol{\theta}^{(sh)}}$.

Definition 3.1 ($\log - \exp$ Complexity)**.** Let $\boldsymbol{\theta}^{(i)}$ and $\boldsymbol{\theta}^{(sh)}$ be a family of weights corresponding to task/edge specific and shared neural networks, respectively. The $\log - \exp$ complexity of the neural network with respect to the distribution $Q_{\boldsymbol{\theta}^{(i)},\boldsymbol{\theta}^{(sh)}}$ (Q for short) for $i = 1, 2, \ldots, N$ is defined as

$$\mathcal{R}_i(\boldsymbol{\theta}) := \log \mathbb{E}_Q \sup_{\boldsymbol{\theta}^{(i)},\boldsymbol{\theta}^{(sh)}} \frac{\exp\left\{\mathbb{E}_{z\sim\mathcal{D}_i} L_i(z, \boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)})\right\}}{\Pi_{j=1}^N \hat{\mathbb{E}} L_j(z_j, \boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)})}. \tag{3}$$

We now present the main result of the paper. Based on the result below, an algorithm for federated learning will be proposed.

Theorem 1 (PAC bound)**.** *For a given neural network $\boldsymbol{\theta}$, and the $\log - \exp$ complexity, the following bound holds with a probability of at least $1 - \delta$, ($\delta > 0$)*

$$\inf_{\boldsymbol{\theta}} \mathbb{E}_{z_i\sim\mathcal{D}_i}\{L_i(z_i, \boldsymbol{\theta})\} \leq \inf_{\boldsymbol{\theta}^{(sh)}} \Big[\texttt{Obj}_i(\boldsymbol{\theta}^{(sh)}) + \mathcal{R}_i(\boldsymbol{\theta}) + \sup_{\boldsymbol{\theta}^{(i)},\boldsymbol{\theta}^{(sh)},\boldsymbol{\omega}_i} KL(Q||P) + l_{max}\sqrt{\sum_{j=1}^N \frac{\omega_{ij}^2}{2n_j^2}\log\left(\frac{1}{\delta}\right)} - N\Big],$$

where $KL(Q||P)$ is the KL-divergence between two joint distributions Q and P, $\texttt{Obj}_i(\boldsymbol{\theta}^{(sh)}) :=$

$$\inf_{\boldsymbol{\omega}_i} \sum_{j=1}^N \Big[\omega_{ij} \inf_{\boldsymbol{\theta}^{(i)}} \hat{\mathbb{E}} L_j(z_j, \boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)}) - \log \omega_{ij}\Big]. \tag{4}$$

Proof: See Appendix (section 6). □

The above theorem suggests that the true average risk is no more than the proposed objective and the error term. The error term is in terms of KL divergence between the exponential distribution presented in the previous section, and the true distribution of the losses for a fixed (worst case) neural network weights. In other words, if the true distribution is close to the independent exponential distribution, then the error in terms of KL is small, as expected. Further, the error also depends on the $\log - \exp$ complexity. The last term in the above equation indicates that higher weights correspond to higher error. This will be used as a regularizer while designing the algorithm, which is explained in the next section.

4. PROPOSED FEDERATED ALGORITHM (OMNI-FEDGE)

The main result suggests that the average loss can be minimized by minimizing $\texttt{Obj}_i(\boldsymbol{\theta}^{(sh)})$ with respect to $\boldsymbol{\theta}^{(sh)}$ while keeping the regularizer term under control. Optimizing over both shared and task specific neural network weights can be split into the following two steps

1. Given a shared neural network weights $\boldsymbol{\theta}^{(sh)}$, each device can use its data to solve $\min_{\boldsymbol{\theta}^{(i)}} \hat{\mathbb{E}} L_j(z_j, \boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)})$ to obtain the task specific neural network weights $\boldsymbol{\theta}^{(i)}$, $i = 1, 2, \ldots, N$.
2. Given the task specific weights, the first step is to solve the optimization problem in (4). This results in $\texttt{Obj}_i(\boldsymbol{\theta}^{(sh)})$. Minimizing this over $\boldsymbol{\theta}^{(sh)}$ results in the shared neural network weights.

The above two steps need to be iterated until convergence. However, the implementation needs to be done in a distributed fashion. A pseudo code for the same is provided in Algorithm 1. The first step above corresponds to step 5 of the algorithm. A standard gradient descent algorithm can be used to implement this step. The second step corresponds to steps 6 to 11 of the algorithm. It is important to note that steps 10 and 11 compute a sum of gradients and broadcast rather than sending each gradient separately. This saves in terms of computation as well as communication complexity. In the following section, we present the experimental results.

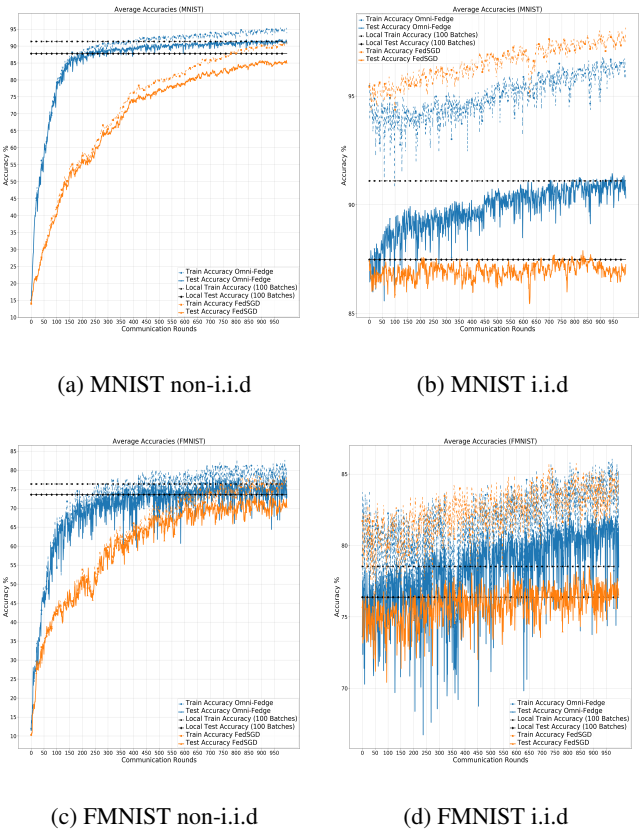
Algorithm 1: Omni-Fedge

1 `Omni-Fedge()`**:**
2 INITIALIZE $\boldsymbol{\theta}^{sh}$ and BROADCAST (BC) to all nodes
3 **for** $t \in \{1, 2, \ldots\ldots\}$ **do**
4 **for** $i = 1, 2, \ldots, N$ **do**
5 $\boldsymbol{\theta}_t^{(i)} = \arg\min_{\boldsymbol{\theta}^{(i)}} \hat{\mathbb{E}} L_i(z_i, \boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)})$
6 Each device i BCs $\boldsymbol{\theta}_t^{(i)}$ to all other nodes $l = 1, 2, \ldots, N$ through FS.
7 COMPUTE AND SEND $\hat{\mathbb{E}} L_i(z_i, \boldsymbol{\theta}^{(j)}, \boldsymbol{\theta}^{(sh)})$ to all nodes.
8 `Minimize-Objective()`
9 to get $\boldsymbol{\omega}_i$ for all i.
10 At each node, COMPUTE $\sum_{j=1}^N \omega_{ji}^* \nabla_{\boldsymbol{\theta}_t^{(sh)}} \hat{\mathbb{E}} L_i(z_i, \boldsymbol{\theta}^{(j)}, \boldsymbol{\theta}^{(sh)})$ and BC it to all nodes through FS.
11 Perform GRADIENT UPDATE $\boldsymbol{\theta}_{t+1}^{(sh)} := \boldsymbol{\theta}_t^{(sh)} - \eta^{com}\gamma_t^{(i)}$, where $\gamma_t^{(i)} := \frac{1}{N}\Big(\sum_{l=1}^N \sum_{j=1}^N \omega_{jl}^* \nabla_{\boldsymbol{\theta}_t^{(sh)}} \hat{\mathbb{E}} L_l(z_l, \boldsymbol{\theta}^{(j)}, \boldsymbol{\theta}^{(sh)})\Big)$
12 GO TO step 3.
13 `Minimize-Objective()`**:**
14 COMPUTE $\boldsymbol{\omega}_i^* = \arg\min_{\boldsymbol{\omega}_i} \Big(\sum_{j=1}^N \omega_{ij} \hat{\mathbb{E}} L_j(z_j, \boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)}) - \log \prod_{j=1}^N \omega_{ij}\Big)$

5. EXPERIMENTAL RESULTS AND CONCLUSIONS

The experiments are carried out using MNIST [17] handwritten data set and FMNIST [18] fashion data set, with 5 nodes and a central FS. Both MNIST and FMNIST have $60,000$ training examples and a test set of $10,000$ examples each. Each example is a 28×28 grayscale image, associated with labels from 10 classes. The table below shows the split between training and test data. The non-i.i.d case is emulated using a non-uniform sampling of data from the MNIST and FMNIST data sets. The data samples assigned to each node are further divided into 100 batches, and batch-wise training is performed. The proposed algorithm is compared with (i) local training; training using local data only, and (ii) FedSGD. In FedSGD, the gradient is averaged, and one neural network is used across all the devices. Fig. 1 shows the performance of the proposed algorithm compared with the above mentioned algorithms/methods for both training and test data. It is clear from the figures that the proposed outperforms local and FedSGD for both i.i.d and non-i.i.d scenarios. It was also observed during the experiments that the algorithm results in higher weights for more relevant nodes, and under i.i.d case, the proposed algorithm allocates equal weights across all the nodes corroborating our intuition. To conclude, we considered the problem of FL under a non-i.i.d data setting and provided a sound theoretical framework for the proposed algorithm based on a novel Bayesian approach. We also introduced a new complexity measure as a consequence of the PAC bound. As evident from the experimental results, performance of the proposed algorithm is significantly better than existing algorithms with better generalization.

Type of Data	Training samples per node	Testing samples per node
MNIST i.i.d	622	778
MNIST non-i.i.d	673	842
FMNIST i.i.d	622	778
FMNIST non-i.i.d	331	414



(a) MNIST non-i.i.d (b) MNIST i.i.d

(c) FMNIST non-i.i.d (d) FMNIST i.i.d

Fig. 1. Plots of Average Accuracies vs Communication Rounds for Omni-Fedge and FedSGD

6. APPENDIX

Proof. Consider the following difference for $i = 1, 2, \ldots, N$ $\Phi_i(\mathcal{Z}) := \sup_{\boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)}, \boldsymbol{\omega}_i} \Big[\mathbb{E}_{z\sim\mathcal{D}_i} L_i(z, \boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)}) - \sum_{j=1}^N \omega_{ij}\hat{\mathbb{E}}L_j(z_j, \boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)}) + \sum_{j=1}^N \log\omega_{ij}\Big]$, where $\mathcal{Z}$ is the data at all the nodes. Let $\mathcal{Z}'$ be such that it differs from $\mathcal{Z}$ at only one place and $|\Phi_i(\mathcal{Z}) - \Phi_i(\mathcal{Z}')| \leq \frac{\omega_{ij} l_{max}}{n_j}$. By applying McDiarmid's inequality, with a probability of at least $1-\delta$ ($\delta > 0$), the following holds for all $\boldsymbol{\theta}^{(sh)}$ and $\boldsymbol{\theta}^{(i)}$ $\sup_{\boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)}, \boldsymbol{\omega}_i} \Big[\mathbb{E}_{z\sim\mathcal{D}_i} L_i(z, \boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)}) - \sum_{j=1}^N \omega_{ij}\hat{\mathbb{E}}L_j(z_j, \boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)}) + \sum_{j=1}^N \log\omega_{ij}\Big] \leq \mathbb{E}\Big[\sup_{\boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)}, \boldsymbol{\omega}_i} \Big[\mathbb{E}_{z\sim\mathcal{D}_i} L_i(z, \boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)}) - G(\omega)\Big]\Big] + l_{max}\sqrt{\sum_{j=1}^N \frac{\omega_{ij}^2}{2n_j^2}\log\frac{1}{\delta}}$, where $G(\omega) := \sum_{j=1}^N(\omega_{ij}\hat{\mathbb{E}}L_j(z_j, \boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)}) - \log\omega_{ij})$. By optimizing over ω_{ij}, $\sup_{\boldsymbol{\omega}_i} G(\omega) = N + \sum_{j=1}^N \log\hat{\mathbb{E}}L_j(z_j, \boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)})$. Applying Fenchel-Young inequality to $\mathbb{E}\Big[\sup_{\bar{\boldsymbol{\theta}}^{(i)}}\Big[\Delta(\bar{\boldsymbol{\theta}}^{(i)})\Big]\Big]$ (see [19]), where $\Delta(\bar{\boldsymbol{\theta}}^{(i)}) := \mathbb{E}_{z\sim\mathcal{D}_i} L_i(z, \boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)}) - \sum_{j=1}^N \log\hat{\mathbb{E}}L_j(z_j, \boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)})$ and $\bar{\boldsymbol{\theta}}^{(i)} := (\boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)})$. Now, pulling the supremum inside the summation, we get $\mathbb{E}_{z\sim\mathcal{D}_i} L_i(z, \boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)}) \leq \sum_{j=1}^N \omega_{ij}\hat{\mathbb{E}}L_j(z_j, \boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)}) - \sum_{j=1}^N \log\omega_{ij} + \sup_{\boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)}}\Big[\mathrm{KL}(Q||P)\Big] + \log\mathbb{E}\Big[\sup_{\boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)}} \frac{\exp\{\mathbb{E}_{z\sim\mathcal{D}_i} L_i(z, \boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)})\}}{\Pi_{j=1}^N \hat{\mathbb{E}}L_j(z_j, \boldsymbol{\theta}^{(i)}, \boldsymbol{\theta}^{(sh)})}\Big] + l_{max}\sqrt{\sum_{j=1}^N \frac{\omega_{ij}^2}{2n_j^2}\log\frac{1}{\delta}} - N$. Since this holds for all $\boldsymbol{\theta}^{(sh)}$, $\boldsymbol{\theta}^{(i)}$ and $\boldsymbol{\omega}_i$, taking infimum over these proves the theorem. □

7. REFERENCES

[1] Jakub Konečnỳ, H Brendan McMahan, Daniel Ramage, and Peter Richtárik, "Federated optimization: Distributed machine learning for on-device intelligence," *arXiv preprint arXiv:1610.02527*, 2016.

[2] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[3] Solmaz Niknam, Harpreet S Dhillon, and Jeffrey H Reed, "Federated learning for wireless communications: Motivation, opportunities, and challenges," *IEEE Communications Magazine*, vol. 58, no. 6, pp. 46–51, 2020.

[4] Sebastian Ruder, "An overview of multi-task learning in deep neural networks," *arXiv preprint arXiv:1706.05098*, 2017.

[5] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays, "Applied federated learning: Improving google keyboard query suggestions," *arXiv preprint arXiv:1812.02903*, 2018.

[6] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582*, 2018.

[7] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.

[8] Ozan Sener and Vladlen Koltun, "Multi-task learning as multi-objective optimization," in *Advances in Neural Information Processing Systems*, 2018, pp. 527–538.

[9] Alex Kendall, Yarin Gal, and Roberto Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7482–7491.

[10] Sebastian Caldas, Jakub Konečny, H Brendan McMahan, and Ameet Talwalkar, "Expanding the reach of federated learning by reducing client resource requirements," *arXiv preprint arXiv:1812.07210*, 2018.

[11] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek, "Robust and communication-efficient federated learning from non-iid data," *IEEE transactions on neural networks and learning systems*, 2019.

[12] Meng Hao, Hongwei Li, Xizhao Luo, Guowen Xu, Haomiao Yang, and Sen Liu, "Efficient and privacy-enhanced federated learning for industrial artificial intelligence," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 10, pp. 6532–6542, 2019.

[13] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith, "Federated learning: Challenges, methods, and future directions," *arXiv preprint arXiv:1908.07873*, 2019.

[14] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh, "Agnostic federated learning," *arXiv preprint arXiv:1902.00146*, 2019.

[15] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, and Ameet S Talwalkar, "Federated multi-task learning," in *Advances in Neural Information Processing Systems*, 2017, pp. 4424–4434.

[16] Changjian Shui, Mahdieh Abbasi, Louis-Émile Robitaille, Boyu Wang, and Christian Gagné, "A principled approach for learning task similarity in multitask learning," *arXiv preprint arXiv:1903.09109*, 2019.

[17] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[18] Han Xiao, Kashif Rasul, and Roland Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," 2017.

[19] Michael M Wolf, "Mathematical foundations of supervised learning," 2018.