Auto-weighted Robust Federated Learning with Corrupted Data Sources

Shenghui Li*, Edith Ngai†, Fanghua Ye‡, and Thiemo Voigt*
*Uppsala University, Sweden
†The University of Hong Kong, China
‡University College London, UK
*{shenghui.li, thiemo.voigt}@it.uu.se, †chngai@eee.hku.hk, ‡fanghua.ye.19@ucl.ac.uk

Abstract*—Federated learning provides a communication-efficient and privacy-preserving training process by enabling learning statistical models with massive participants while keeping their data in local clients. However, standard federated learning techniques that naively minimize an average loss function are vulnerable to data corruptions from outliers, systematic mislabeling, or even adversaries. In this paper, we address this challenge by proposing Auto-weighted Robust Federated Learning (ARFL), a novel approach that jointly learns the global model and the weights of local updates to provide robustness against corrupted data sources. We prove a learning bound on the expected risk with respect to the predictor and the weights of clients, which guides the definition of the objective for robust federated learning. We present an objective that minimizes the weighted sum of empirical risk of clients with a regularization term, where the weights can be allocated by comparing the empirical risk of each client with the average empirical risk of the best p clients (p-average). This method can downweight the clients with significantly higher losses, thereby lowering their contributions to the global model. We show that this approach achieves robustness when the data of corrupted clients is distributed differently from the benign ones. To optimize the objective function, we propose a communication-efficient algorithm based on the blockwise minimization paradigm. We conduct extensive experiments on multiple benchmark datasets, including CIFAR-10, FEMNIST, and Shakespeare, considering different neural network models. The results show that our solution is robust against different scenarios including label shuffling, label flipping, and noisy features, and outperforms the state-of-the-art methods in most scenarios.

***Index Terms*—Federated learning, robustness, auto-weighted, distributed learning, neural networks**

I. Introduction

Federated learning [1]–[4] has recently attracted more and more attention due to the increasing concern of user data privacy. In federated learning, the server trains a shared model based on data originating from remote clients such as smartphones and IoT devices, without the need to store and process the data in a centralized server. In this way, federated learning enables joint model training over privacy-sensitive data in a wide range of applications [5], [6], including natural language processing [7], computer vision [8], and speech recognition [9].

However, standard federated learning strategies fail in the presence of data corruption [10], [11]. Data collected from different clients, or data sources (in this paper we use the two terms interchangeably), may vary greatly in data quality and thus reduce the reliability of the learning task. Some of the clients may be unreliable or even malicious. For instance, distributed sensor networks are vulnerable to cybersecurity attacks, such as false data injection attacks [12]. In crowdsourcing scenarios, the problem of noisy data is unignorable due to biased or erroneous workers [13]. As a consequence, the clients can update their local models using corrupted data and send the parameters to the server. After averaging these harmful updates, the accuracy and the convergence of the shared global model can be compromised.

To mitigate this limitation, different robust learning approaches have been proposed in the literature. Some of these techniques rely on robust statistical estimations (e.g. geometric median estimators) to update the aggregated model [14]–[16]. Although these techniques are widely used in traditional distributed learning scenarios with i.i.d. datasets, it is not straightforward to generalize them for non-i.i.d. data settings, i.e., some of the clients have significantly different data distribution than the others. Other approaches require some trusted clients or samples to guide the learning [17]–[19] or detect the updates from corrupted clients [20]–[22]. Unfortunately, the the credibility of these trusted clients and samples are usually not guaranteed. Since the data is stored locally in the clients, the server is insensible to the corruption behaviors and unable to measure the quality of data sources due to privacy and communication constraints.

In this paper, we aim to to improve the robustness of federated learning when the data provided by some of the clients are corrupted. We propose a novel solution, named Auto-weighted Robust Federated Learning (`ARFL`), to jointly learn the global model and the weights of local updates. More specifically, we first prove a learning bound on the expected risk with respect to the predictor and the weights of clients. Based on this theoretical insight, we present our objective that minimizes a weighted sum of the empirical risk of clients with a regularization term. We then theoretically show that the weights in the objective can be allocated by comparing the empirical risk of each client with the best p clients (p-average). When the corrupted clients have significantly higher losses comparing with the p-average loss, their contributions to the global model will be downweighted or even zero-weighted, so as to play less important roles in the global model. Therefore, by using `ARFL`, we can exclude potentially corrupted clients and keep optimizing the global model with the benign clients.

To solve the problem in federated learning settings, we further propose a communication-efficient optimization method based on the blockwise updating paradigm [23]. Through extensive experiments on multiple benchmark datasets (i.e., CIFAR-10, FEMNIST and Shakespeare) with different neural network models[1], we demonstrate the robustness of our approach compared with the state-of-the-art approaches [1], [14], [15], [24], showing up to 30% improvement in model accuracy.

II. Related work

The concept of federated learning has been proposed for collaboratively learning a model without collecting users' data [1]–[4]. The research work on federated learning can be divided into three categories, i.e., horizontal federated learning, vertical federated learning, and federated transfer learning, based on the distribution characteristics of the data. Due to space limits, we refer to Yang et al. [25] for detailed explanations. In this paper we focus on horizontal federated learning where datasets in all clients share the same feature space but different samples. Based on this framework, Federated Averaging (`FedAvg`) has been proposed to update global parameters with a weighted average of the model parameters sent by a subset of clients after several local iterations [1].

Recent attention has also been focused on the provisioning of robust federated learning in the research literature, since previous approaches (e.g., `FedAvg`) are fragile in the presence of malicious or corrupted clients. Among these robust approaches, robust statistical estimations have received much attention in particular. Typical estimation rules include geometric median [26], weighted geometric median [15], and Krum [14]. However, the implementations of [15] and [14] are based on the measurement of Euclidean distance between local updates, which may not be qualitatively meaningful in high dimensional space [27]. Moreover, [26] and [14] rely on the assumption that data are balanced among the clients, i,e, they have the same (or a similar) number of training data points. Hence, those approaches can be inefficient when some of the clients have significantly more data than others.

Others [21], [24], [28], couple the process of teaching and learning based on a few trusted instances to produce a robust prediction model. For example, Sattler et al. [24] separate the client population into different groups (e.g., benign and corrupted groups) based on the pairwise cosine similarities between their parameter updates. Li et al. [28] allow the server to learn to detect and remove the malicious model updates using an encoder-decoder based detection model. These approaches require some trusted clients or samples to guide the learning or detect the updates from corrupted clients. Unfortunately, the credibility of these trusted clients and samples are usually not guaranteed since the data is isolated and stored locally. Thus, the server is insensible to these corruption behaviors without the ability to measure the quality of data at the sources due to privacy and communication constraints. Different from previous studies, we propose a robust approach that can learn both the global model and the weights of clients automatically from a mix of reliable and unreliable clients, without the need of any pre-verified trusted instances.

III. Preliminaries and Motivation

In federated learning tasks, a general assumption is that the target distribution for which the centralized model is learned is a weighted mixture of distributions associated with all clients (or data sources), that is, if we denote by $\mathcal{D}_i$ the distribution associated with the i-th client, the centralized model is trained to minimize the risk with respect to $\mathcal{D}_{\boldsymbol{\alpha}} = \sum_{i=1}^{N} \alpha_i \mathcal{D}_i$, where N is the total number of clients, $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_N)^\top$ is the vector of source-specific weights. We also have $\boldsymbol{\alpha} \in \mathbb{R}_+^n$ and $\mathbf{1}^\top \boldsymbol{\alpha} = 1$ [2], [29], [30].

Let $\ell_h(\boldsymbol{z})$ be the loss function that captures the error of a predictor $h \in \mathcal{H}$ (where $\mathcal{H}$ is the hypothesis class) on the training data $\boldsymbol{z} = (\boldsymbol{x}, y)$ (where $(\boldsymbol{x}, y)$ is the input and output pair), and $\mathcal{L}_{\mathcal{D}_{\boldsymbol{\alpha}}}(h)$ be the expected risk of a predictor h on the mixture data distribution $\mathcal{D}_{\boldsymbol{\alpha}}$, we have:

$$\mathcal{L}_{\mathcal{D}_{\boldsymbol{\alpha}}}(h) = \sum_{i=1}^{N} \alpha_i \mathcal{L}_i(h) = \sum_{i=1}^{N} \alpha_i \mathbb{E}_{\boldsymbol{z}\sim\mathcal{D}_i}(\ell_h(\boldsymbol{z})), \tag{1}$$

where $\mathcal{L}_i(h) = \mathbb{E}_{\boldsymbol{z}\sim\mathcal{D}_i}(\ell_h(\boldsymbol{z}))$ is the expected loss of a predictor h on the data distribution $\mathcal{D}_i$ of the i-th client.

Most prior work in federated learning has assumed that all samples are uniformly weighted, where the underlying assumption is that the target distribution is $\overline{\mathcal{U}} = \sum_{i=1}^{N} \frac{m_i}{M} \mathcal{D}_i$ where m_i is the number of samples from client i and $M = \sum_{i=1}^{N} m_i$, thus the expected risk becomes:

$$\mathcal{L}_{\overline{\mathcal{U}}}(h) = \sum_{i=1}^{N} \frac{m_i}{M} \mathbb{E}_{\boldsymbol{z}\sim\mathcal{D}_i}(\ell_h(\boldsymbol{z})) \tag{2}$$

In practice, the goal is to minimize a traditional empirical risk $\hat{\mathcal{L}}_{\overline{\mathcal{U}}}(h)$ as follows:

$$\hat{\mathcal{L}}_{\overline{\mathcal{U}}}(h) = \sum_{i=1}^{N} \frac{m_i}{M} \frac{1}{m_i} \sum_{j=1}^{m_i} (\ell_h(\boldsymbol{z}_{i,j})), \tag{3}$$

which can be minimized by sampling a subset of clients randomly at each round, then running an optimizer such as stochastic gradient descent (SGD) for a variable number of iterations locally on each client. These local updating methods enable flexible and efficient communication compared to traditional mini-batch methods, which would simply calculate a subset of the gradients [31]–[33]. `FedAvg` is one of the most well-known methods to minimize Eq. (3) in non-convex settings [1]. The method runs simply by letting each selected client apply a fixed number of epochs of SGD locally and then averaging the resulted local models.

However, these techniques are not fault tolerant and can be vulnerable to data corruptions on data sources. It has been shown that a few clients with corrupted data could lead to inaccurate models [34]. The problem stems from

[1]Code is available at https://github.com/lishenghui/ARFL

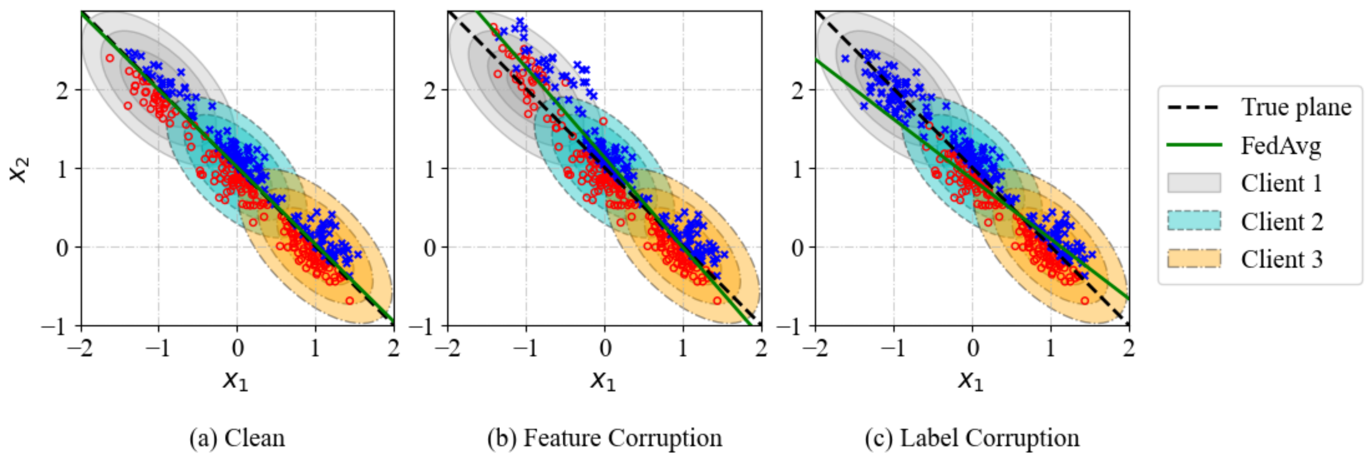


Fig. 1: Illustration of federated learning using the standard `FedAvg` with potential corruptions in local datasets, where red circles represent data of class 0 and blue crosses represent class 1. (a) All samples are clean. (b) The features in Client 1 is shifted by random noise. (c) The labels in Client 1 are forced to class 1.

a mismatch between the target distribution and $\overline{\mathcal{U}}$. That is, in corruption scenarios, the target distribution may be in general quite different from $\overline{\mathcal{U}}$, which includes some corrupted components. More specifically, we model the target distribution with corrupted clients as follows.

$$\mathcal{D}_{\boldsymbol{\alpha}} = \sum_{i=1}^{N} \alpha_i \mathcal{D}_i = \sum_{i=1}^{N} \alpha_i (\eta_i \mathcal{D}_{i,b} + (1-\eta_i)\mathcal{D}_{i,c}),$$

where $\eta_i \in \{0,1\}$ denotes whether the local data distribution $\mathcal{D}_i$ is a benign distribution $\mathcal{D}_{i,b}$ (when $\eta_i = 1$) or a corrupted distribution $\mathcal{D}_{i,c}$ (when $\eta_i = 0$).

When $\sum_{i=1}^{N} \eta_i = N$, all components of the mixture distribution are benign (i.e. $\mathcal{D}_i = \mathcal{D}_{i,b}$). Assuming that the target distribution is uniform $\overline{\mathcal{U}}$, minimizing Eq. (3) can lead to an accurate global model. However, when there are corrupted data sources (i.e. $\sum_{i=1}^{N} \eta_i < N$), the mixture of the distributions will include some corrupted components $\mathcal{D}_{i,c}$. In this case, optimizing the empirical risk with respect to $\overline{\mathcal{U}}$ will not bring an accurate global model.

As an example, we train a binary classification model from three clients, where the data is originally generated from three distributions with linearly separable classes. A Logistic Regression (LR) model is learned using the standard `FedAvg` approach. As shown in Fig. 1(a), the learned separating plane is closed to the true plane when the all datasets are clean. When Client 1 is corrupted on either feature $\boldsymbol{x}$ (Fig. 1(b)) or label y (Fig. 1(c)), the learned plane is driven away from the true one.

One possible solution is to exclude all the corrupted components of the mixture distribution and seek for a new target distribution considering only benign clients, i.e. $\overline{\mathcal{U}}_b = \sum_{i\in\mathcal{B}} \frac{m_i}{M_\mathcal{B}} \mathcal{D}_i$, where $\mathcal{B}$ is the set of all benign clients and $M_\mathcal{B} = \sum_{i\in\mathcal{B}} m_i$. The challenge with this approach is that the centralized server is agnostic to the corruptions on the local clients and hence it is impossible to measure the data quality directly.

IV. Robust Federated Learning

In this section, we describe our Auto-weighted Robust Federated Learning (`ARFL`) in detail. Specifically, we present a novel objective according to a learning bound with respect to both the predictor h and the weights $\boldsymbol{\alpha}$. We analyze the robustness of the proposed objective against corruptions and optimize it with a federated learning based algorithm.

A. Learning Bound

In the following theorem, we present a learning bound on the expected risk on the weighted mixture distribution with respect to the predictor h and the weights $\boldsymbol{\alpha}$. A proof is provided in Appendix A.

Theorem 1. *We denote $\hat{\mathcal{L}}_i(h)$ as the corresponding empirical counterparts of $\mathcal{L}_{\mathcal{D}_i}(h)$. Assume that the loss function is bounded by a constant $\mathcal{M} > 0$. Then, for any $\delta > 0$, with probability at least $1-\delta$ over the data, the following inequality holds:*

$$\mathcal{L}_{\mathcal{D}_{\boldsymbol{\alpha}}}(h) \le \sum_{i=1}^{N} \alpha_i \hat{\mathcal{L}}_i(h) + 2\sum_{i=1}^{N} \alpha_i \mathcal{R}_i(\mathcal{H}) + 3\sqrt{\frac{\log\left(\frac{4}{\delta}\right)\mathcal{M}^2}{2}}\sqrt{\sum_{i=1}^{n} \frac{\alpha_i^2}{m_i}}, \quad (4)$$

where, for each client $i = 1, \ldots, N$,

$$\mathcal{R}_i(\mathcal{H}) = \mathbb{E}_\sigma \left(\sup_{f\in\mathcal{H}} \left(\frac{1}{m_i} \sum_{j=1}^{m_i} \sigma_{i,j} \ell_f(z_{i,j}) \right) \right)$$

and $\sigma_{i,j}$s are independent uniform random variables taking values in $\{-1,+1\}$. These variables are called Rademacher random variables [35]

Although the Rademacher complexities in the bound are functions of both the underlying distribution and the hypothesis class [35], in our setting the hypothesis space $\mathcal{H}$ is fixed and hence these bounds would be identical for all i [22]. Therefore, we expect the $\mathcal{R}_i(\mathcal{H})$ to be of similar order for all clients and the impact of $\boldsymbol{\alpha}$ in the second term to be negligible. Thus we ignore this term during optimization.

The rest of the terms for the learning bound in Theorem 1 thus suggest a trade-off between reducing the weighted sum of the empirical losses and minimizing a weighted norm of the weights $\sqrt{\sum_{i=1}^{n} \frac{\alpha_i^2}{m_i}}$. Note that reducing the weighted sum of the empirical losses will encourage trusting the clients who provide data with the smallest loss but it may lead to large and sparse weights, which will make the model fits only the very few local datasets. On the contrary, minimizing the norm will increase the smoothness of the weight distribution, but it may also increase the weighted sum of empirical losses.

To control this trade-off, we introduce a tuning hyperparameter λ to the weighted norm. For the convenience of derivation, the square root on the sum of weights can be removed by tuning λ.

B. Problem Formulation

We present our problem formulation in the following. The learning bound derived above suggests minimizing the following objective with respect to the model parameters $\boldsymbol{w}$ together with the weights $\boldsymbol{\alpha}$:

$$\begin{aligned} \min_{\boldsymbol{w},\boldsymbol{\alpha}} \quad & \sum_{i=1}^{N} \alpha_i \hat{\mathcal{L}}_i(\boldsymbol{w}) + \frac{\lambda}{2} \sum_{i=1}^{N} \frac{\alpha_i^2}{m_i}, \\ s.t. \quad & \boldsymbol{\alpha} \in \mathbb{R}_+^n, \mathbf{1}^\top \boldsymbol{\alpha} = 1, \end{aligned} \tag{5}$$

where $\boldsymbol{w}$ is a vector of parameters defining a predictor h. Note that here and after we use notation $\hat{\mathcal{L}}_i(\boldsymbol{w})$ to replace $\hat{\mathcal{L}}_i(h)$, representing the empirical risk of hypothesis h (corresponding to w) on client i.

Note that the second term of the objective is small whenever the weights are distributed proportionally to the number of samples of the client. As $\lambda \to \infty$, we have $\alpha_i(\boldsymbol{w}) = \frac{m_i}{M}$, which means that all clients are assigned with weights proportional to their number of training samples and the model minimizes the empirical risk over all the data, regardless of the losses of the clients. Thus, the objective becomes the same as the standard `FedAvg` in Eq. (3). In contrast, as $\lambda \to 0$, the regularization term in Eq. (5) vanishes, so that the client with the lowest empirical risk will dominate the objective by setting its weight to 1. λ thus acts as a form of regularization by encouraging the usage of information from more clients.

C. Robustness of the Objective

We now study the optimal weights $\boldsymbol{\alpha}$ of the problem in Eq. (5) to understand how the objective yields robustness against corrupted data sources. We notice that the objective is a convex quadratic program problem over $\boldsymbol{\alpha}$. Given that $\boldsymbol{\alpha} = N^{-1}\mathbf{1}$ is a strictly feasible point, the problem satisfies Slater's condition, which indicates the strong duality of the problem. Thus, the optimal weights $\boldsymbol{\alpha}$ can be obtained using the Karush-Kuhn-Tucker (KKT) conditions [36]. Here we give the closed-form solution in Theorem 2. The detailed proof is provided in Appendix B.

Theorem 2. *For any $\boldsymbol{w}$, when $\lambda > 0$ and $\{\hat{\mathcal{L}}_i(\boldsymbol{w})\}_{i=1}^{N}$ are sorted in increasing order: $\hat{\mathcal{L}}_1(\boldsymbol{w}) \le \hat{\mathcal{L}}_2(\boldsymbol{w}) \le \ldots \le \hat{\mathcal{L}}_N(\boldsymbol{w})$, by setting:*

$$p = \underset{k}{\operatorname{argmax}}\{1 + \frac{M_k(\overline{\mathcal{L}}_k(\boldsymbol{w}) - \hat{\mathcal{L}}_k(\boldsymbol{w}))}{\lambda} > 0\}, \tag{6}$$

where $M_k = \sum_{i=1}^{k} m_i$,

$$\overline{\mathcal{L}}_k(h) = \frac{\sum_{i=1}^{k} m_i \hat{\mathcal{L}}_i(\boldsymbol{w})}{M_k} \tag{7}$$

is the average loss over the first k clients that have the smallest empirical risks. Then the optimal $\boldsymbol{\alpha}$ to the problem (5) is given by:

$$\alpha_i(\boldsymbol{w}) = \frac{m_i}{M_p}[1 + \frac{M_p(\overline{\mathcal{L}}_p(\boldsymbol{w}) - \hat{\mathcal{L}}_i(\boldsymbol{w}))}{\lambda}]_+, \tag{8}$$

where $[\cdot]_+ = max(0, \cdot)$.

When $\lambda \in (0, \infty)$, plugging $\alpha_i(\boldsymbol{w})$ back into Eq. (5) yields the equivalent concentrated objective

$$\sum_{i=1}^{N}[\frac{m_i}{M_p} + \frac{m_i(\overline{\mathcal{L}}_p(\boldsymbol{w}) - \hat{\mathcal{L}}_i(\boldsymbol{w}))}{\lambda}]_+(\hat{\mathcal{L}}_i(\boldsymbol{w}) + \frac{1}{2}[\frac{\lambda}{M_p} + \overline{\mathcal{L}}_p(\boldsymbol{w}) - \hat{\mathcal{L}}_i(\boldsymbol{w})]_+), \tag{9}$$

which consists of N components and each is related to the empirical risk of the corresponding client and the average loss over the first p clients with smallest losses, which is called the p-average loss. An intuitive interpretation is that the p clients act as a consensus group to reallocate the weights and encourage trusting clients that provide empirical losses that are smaller than the average while downweighting the clients with higher losses. Given a suitable λ, more benign clients will be in the consensus group to dominate the model and exclude the outliers.

In a situation where the majority of clients are benign, $\overline{\mathcal{L}}_p(\boldsymbol{w})$ becomes relatively low as the benign clients achieve the minimum empirical risk. The components with corrupted datasets will be downweighted as they have higher losses. Especially, the i-th component becomes zero when $\hat{\mathcal{L}}_i(\boldsymbol{w}) \ge \frac{\lambda}{M_p} + \overline{\mathcal{L}}_p(\boldsymbol{w})$, which means that a client is considered to be corrupted and does not contribute to the objective if its empirical risk is significantly larger than the p-average loss, where the threshold $\frac{\lambda}{M_p} + \overline{\mathcal{L}}_p(\boldsymbol{w})$ is controlled by λ. From Eq. (6) we can also conclude that the optimal solution has only p non-zero components and the remaining components will be exactly zero.

On the other hand, if the corrupted clients try to bias the model to fit their corrupted datasets, the p-average loss $\overline{\mathcal{L}}_p(\boldsymbol{w})$ becomes higher because the model does not fit the samples from the majority. The threshold $\frac{\lambda}{M_p} + \overline{\mathcal{L}}_p(\boldsymbol{w})$ will also be enlarged, which makes $\alpha_i(\boldsymbol{w})$ fail to downweight the component with high losses. Thus, the optimization problem in Eq. (5) will exceed the minimum.

We expect that the data distributions are more similar among benign clients compared with the corrupted ones even when the data is non-i.i.d., so that we can identify the corrupted clients according to their empirical losses as discussed above. Although this is not always the case in some strongly non-i.i.d. data scenarios, i.e., some local data distributions might be strongly different from the others but they are not corrupted, we argue that it is technically difficult to distinguish between "corrupted" and "just different" (but not corrupted) clients if the data is strongly non-i.i.d., which we leave it for future work. Nevertheless, we show in our experiments (Sec. V) that our approach performs well in some general non-i.i.d. settings.

D. Blockwise Minimization Algorithm

To solve the robust learning problem in federated learning settings, we propose an optimization method based on the blockwise updating paradigm, which is guaranteed to converge to a critical point when the parameter set is closed and convex [37]. The key idea is to divide the problem into two parts. One sub-problem for estimating the model parameters and the other sub-problem for automatically weighting the importance of client updates. Then we minimize the objective iteratively w.r.t. one variable each time while fixing the other one.

The pseudocode of the optimization procedure is given in Algorithm 1. At the beginning of the algorithm, we initialize $\hat{\mathcal{L}} = [\hat{\mathcal{L}}_1, \hat{\mathcal{L}}_2, ..., \hat{\mathcal{L}}_N]^\top$ by broadcasting the initial global model $\boldsymbol{w}_0$ to each client to measure its training loss and return it to the server.

Algorithm 1 Optimization of `ARFL`

```
 Server executes:
 1: Initialize $\boldsymbol{w}_0, \hat{\mathcal{L}}, \boldsymbol{\alpha}$
 2: for each round $t = 1, 2, \dots$ do
 3:     Select a subset $S_t$ from $N$ clients at random
 4:     Broadcast the global model $\boldsymbol{w}_t$ to selected clients $S_t$
 5:     for each client $i \in S_t$ in parallel do
 6:         $\boldsymbol{w}^i_{t+1}, \hat{\mathcal{L}}_i \leftarrow$ ClientUpdate$(i, \boldsymbol{w}_t)$
 7:     end for
 8:     Update $\boldsymbol{w}_{t+1}$ according to Eq. (10)
 9:     Update $\boldsymbol{\alpha}$ according to Theorem 2
10: end for
11:
 ClientUpdate$(i, \boldsymbol{w})$:    // Run on client $i$
12: $\mathcal{L}_i \leftarrow$ (evaluate training loss using training set)
13: $\mathcal{B} \leftarrow$ (split local training set into batches of size $B$)
14: for each local epoch $i$ from 1 to $E$ do
15:     for batch $b \in \mathcal{B}$ do
16:         $\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla \ell(\boldsymbol{w}; b)$
17:     end for
18: end for
19: return $\boldsymbol{w}$ and $\hat{\mathcal{L}}_i$
```

Updating $\boldsymbol{w}$. When $\boldsymbol{\alpha}$ is fixed, similar to the standard `FedAvg` approach, at round t, the server selects a subset S_t of clients at random (Line 3) and broadcasts the global model $\boldsymbol{w}_t$ to the selected clients (Line 4). For each client i, it firstly evaluates its training loss $\mathcal{L}_i$ using its local dataset. Then, the model parameters can be updated by local computation with a few steps of SGD (Line 14-18), after which the client uploads the new model parameters $\boldsymbol{w}$ along with $\mathcal{L}_i$ to the server (Line 6). While at the aggregation step, the server assembles the global model as:

$$\boldsymbol{w}_{t+1} \leftarrow \sum_{i \in S_t} \frac{\alpha_i}{\sum_{i \in S_t} \alpha_i} \boldsymbol{w}^i_{t+1}. \tag{10}$$

Updating $\boldsymbol{\alpha}$. When $\boldsymbol{w}$ is fixed, we update $\boldsymbol{\alpha}$ using Theorem 2. Intuitively, in order to update $\boldsymbol{\alpha}$, the server should broadcast the updated model parameters to all clients to obtain their training loss before updating $\boldsymbol{\alpha}$. Unfortunately, such behavior might significantly increase the burden of the communication network. To improve communication efficiency, we only update the losses from those selected clients while keeping the others unchanged. Thus, the communication protocol becomes the same as `FedAvg`, without extra communication overhead.

V. Experimental Results

A. Experimental Setup

We perform our experimental evaluation on three datasets that are commonly used in previous work [1], [38], [39], including CIFAR-10 [40], FEMNIST [41], [42], and Shakespeare [1], [42]. Their basic information is listed in Table I. For the CIFAR-10 dataset, we consider an i.i.d. partition where each local client has approximately the same amount of samples and in proportion to each of the classes. We use the original test set in CIFAR-10 as our global test set for comparing the performance of all methods. For the Shakespeare and FEMNIST datasets, we treat each speaking role or writer as a client and randomly sample subsets of all clients. We assume that data distributions vary among clients in the raw data, we regard this sampling process as non-i.i.d.. The raw data in FEMNIST and Shakespeare is preprocessed using the popular benchmark LEAF [42], where the data on each local client is partitioned into an 80% training set and a 20% testing set. The details of the network models we use in the experiments are as follows:

- The model for CIFAR-10 is a Convolutional Neural Network (CNN) chosen from Tensorflow's website[2], which consists of three 3x3 convolution layers (the first with 32 channels, the second and third with 64, the first two followed with 2x2 max pooling), a fully connected layer with 64 units and ReLu activation, and a final softmax output layer. To improve the performance, data augmentation (random shift and flips) is used in this dataset [39].
- For the FEMNIST dataset, we train a CNN with two 5x5 convolution layers (the first with 32 channels, the second with 64, each followed by 2x2 max pooling), a fully connected layer with 126 units and ReLu activation, and a final softmax output layer.
- For the Shakespeare dataset, we learn a character-level language model to predict the next character over *the Complete Works of Shakespeare* [43]. The model takes a

[2]https://www.tensorflow.org/tutorials/images/cnn

Dataset	#Classes	#Clients	#Samples	i.i.d.	Model used	l_r	E	Batch size	$\|S_t\|$	#Rounds
CIFAR-10	10	100	60,000	Yes	CNN	0.01	5	64	20	2000
FEMNIST	62	1039	236,500	No	CNN	0.01	20	64	32	2000
Shakespeare	80	71	417,469	No	LSTM	0.6	1	10	16	100

TABLE I: Dataset description and parameters

CIFAR-10	**Clean**	**Shuffling**		**Flipping**		**Noisy**	
Corr. Per.	-	30%	50%	30%	50%	30%	50%
FedAvg	$\mathbf{73.63 \pm 0.01}$	61.17 ± 1.81	47.00 ± 7.51	65.01 ± 2.38	51.75 ± 7.75	$\mathbf{73.75 \pm 0.49}$	73.61 ± 0.53
RFA	73.48 ± 0.00	57.86 ± 3.22	40.26 ± 9.14	55.47 ± 5.17	40.91 ± 11.06	73.74 ± 0.52	$\mathbf{73.69 \pm 0.63}$
MKRUM	73.50 ± 0.01	59.27 ± 9.34	52.32 ± 14.90	60.21 ± 5.73	47.96 ± 10.25	73.41 ± 0.69	73.49 ± 0.49
CFL	54.71 ± 0.02	52.54 ± 1.71	50.29 ± 1.95	52.87 ± 1.07	51.67 ± 0.92	54.97 ± 1.14	55.26 ± 1.96
ARFL	73.38 ± 0.01	$\mathbf{71.68 \pm 1.01}$	$\mathbf{69.66 \pm 0.73}$	$\mathbf{71.78 \pm 0.53}$	$\mathbf{70.25 \pm 0.56}$	73.48 ± 0.56	73.29 ± 0.79
FEMNIST	**Clean**	**Shuffling**		**Flipping**		**Noisy**	
Corr. Per.	-	30%	50%	30%	50%	30%	50%
FedAvg	82.09 ± 0.00	61.91 ± 21.33	39.69 ± 20.80	70.19 ± 10.17	48.53 ± 23.49	79.94 ± 0.36	78.27 ± 0.47
RFA	81.98 ± 0.00	74.36 ± 7.52	52.02 ± 22.51	73.80 ± 7.49	50.75 ± 19.91	80.45 ± 0.30	79.21 ± 0.41
MKRUM	81.75 ± 0.01	57.51 ± 21.17	42.40 ± 24.84	78.57 ± 4.83	67.10 ± 7.35	$\mathbf{81.52 \pm 0.53}$	$\mathbf{79.80 \pm 0.22}$
CFL	$\mathbf{82.13 \pm 0.01}$	81.24 ± 0.47	36.03 ± 36.38	81.22 ± 0.36	65.54 ± 26.94	80.13 ± 0.70	79.21 ± 0.64
ARFL	81.76 ± 0.00	$\mathbf{81.60 \pm 0.31}$	$\mathbf{81.35 \pm 0.43}$	$\mathbf{81.87 \pm 0.22}$	$\mathbf{81.30 \pm 0.24}$	80.71 ± 0.28	79.40 ± 0.45
Shakespeare	**Clean**	**Shuffling**		**Flipping**		**Noisy**	
Corr. Per.	-	30%	50%	30%	50%	30%	50%
FedAvg	53.80 ± 0.00	51.98 ± 0.48	47.70 ± 4.96	52.08 ± 0.39	41.85 ± 16.18	51.85 ± 0.56	50.43 ± 1.19
RFA	$\mathbf{54.27 \pm 0.00}$	50.16 ± 1.28	32.49 ± 13.81	50.50 ± 1.02	23.84 ± 21.78	$\mathbf{52.17 \pm 0.50}$	50.69 ± 1.04
MKRUM	50.81 ± 0.00	40.38 ± 7.44	24.46 ± 6.88	44.95 ± 2.43	16.11 ± 15.46	48.19 ± 0.40	45.67 ± 0.46
CFL	54.01 ± 0.00	49.76 ± 4.47	43.68 ± 12.68	51.09 ± 1.36	37.30 ± 19.76	51.98 ± 1.03	50.38 ± 1.39
ARFL	53.52 ± 0.00	$\mathbf{52.85 \pm 0.49}$	$\mathbf{51.61 \pm 0.68}$	$\mathbf{52.82 \pm 0.48}$	$\mathbf{51.74 \pm 0.69}$	52.09 ± 1.27	$\mathbf{50.98 \pm 0.75}$

TABLE II: Averaged test accuracy over five random seeds for `FedAvg`, `RFA`, `MKRUM`, `CFL` and `ARFL` in four different scenarios. In the *shuffling* and *flipping* scenarios, `ARFL` significantly outperforms the others. In the *clean* and *noisy* scenario, `FedAvg`, `RFA`, `MKRUM` and `ARFL` achieve similar accuracy.

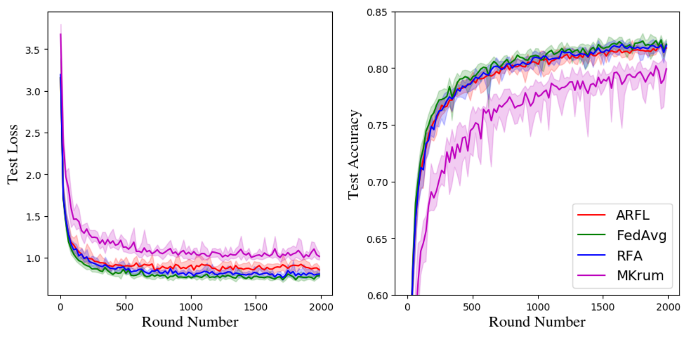
series of characters as input and embeds each of these into an 8-dimensional space. The embedded features are then processed through two stacked Long Short-Term Memory (LSTM) layers, each with 256 nodes and a dropout value of 0.2. Finally, the output of the second LSTM layer is sent to a softmax output layer with one node per character.

For each dataset we consider four different scenarios: 1) **Normal operation** (*clean*): we use the original datasets without any corruption. 2) **Label shuffling** (*shuffling*): the labels of all samples are shuffled randomly in each corrupted client. 3) **Label flipping** (*flipping*): the labels of all samples are switched to a random one in each corrupted client. 4) **Noisy clients** (*noisy*): for CIFAR-10 and FEMNIST datasets, we normalize the inputs to the interval [0, 1]. In this scenario, for the selected noisy clients we add Gaussian noise to all the pixels, so that $x \leftarrow x+\epsilon$, with $\epsilon \sim N(0, 0.7)$. Then we normalize the resulting values again to the interval $[0, 1]$. For the Shakespeare dataset, we randomly select half of the characters and shuffle them so that the input sentence might be disordered. For each corruption scenario, we set 30% and 50% of the clients to be corrupted clients (i.e. providing corrupted data).

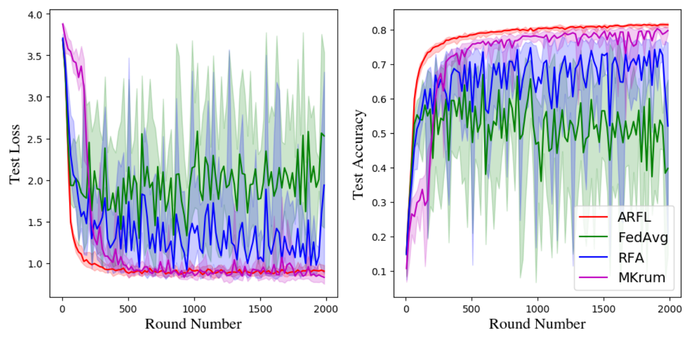
We empirically tune the hyper-parameters on `ARFL` and use the same values in all experiments of each dataset. Following the standard setup, we use SGD and train for E local epochs with local learning rate l_r. A shared global model is trained by all participants, a subset S_t is randomly selected in each round of local training, and $|S_t|$ is the size of S_t. By default, we set $\lambda = M$, where M is the total amount of training samples. We use the parameter setups in Table I, unless specified otherwise. We repeat every experiment five times with different random seeds for data corruption and client selection.

We implement all the code in Tensorflow, simulating a federated learning system with one server and N clients, where N is the total number of clients in the dataset (see Table I). We compare the performance of `ARFL` with the following state-of-the-art approaches:

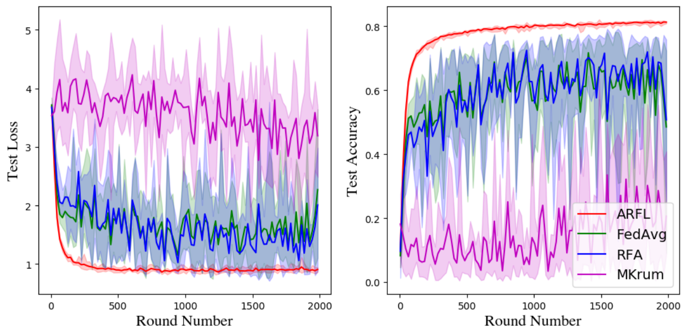
- **FedAvg** [1]. The standard Federated Averaging aggregation approach that just calculates the weighted average of the parameters from local clients.
- **RFA** [15]. A robust aggregation approach that minimizes the weighted Geometric Median (GM) of the parameters from local clients. A smoothed Weiszfeld's algorithm is used to compute the approximate GM.
- **MKrum (Multi-Krum)** [14]. A Byzantine tolerant aggregation rule. Note that this approach tolerates some Byzantine failures such as completely arbitrary behaviors from local updates.
- **CFL** [24]. A Clustered Federated Learning (CFL) ap-



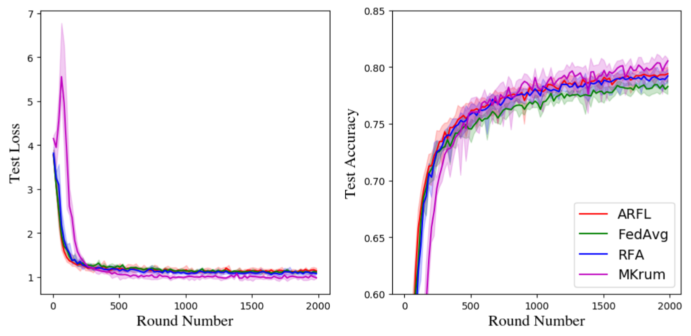
(a) Clean



(b) Shuffling



(c) Flipping



(d) Noisy

Fig. 2: Test loss and accuracy vs. round number for `FedAvg`, `RFA`, `MKRUM` and `ARFL` on the FEMNIST dataset with 50% clients with different corruption scenarios.

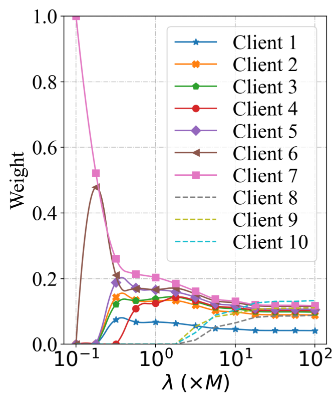
proach that separates the client population into different groups based on the pairwise cosine similarities between their parameter updates, where the clients are partitioned into two groups, i.e., benign clients and corrupted clients.

B. Robustness and Convergence

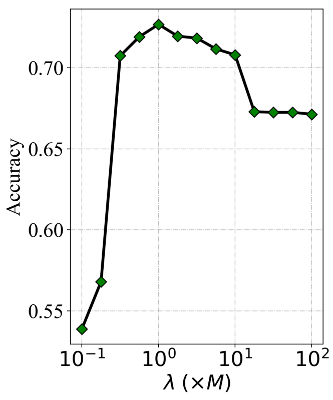
Firstly, we show that `ARFL` is a more robust solution by comparing the average test accuracy in Table II. The results show that `ARFL` is robust in all data corruption scenarios and corruption levels. It achieves the highest test accuracy in most scenarios compared with the other approaches. `ARFL` achieves significantly higher test accuracy in the *shuffling* and *flipping* corruption scenarios compared with all the existing methods, which is especially noticeable in the case of *flipping* corruption with the level of 50%, where the test accuracy for the CIFAR-10, FEMNIST and Shakespeare datasets are $70.25\%, 81.30\%$ and 51.74%, which are $18.5\%, 14.2\%$, and 9.89% higher than that of the best of existing methods, respectively. In the *clean* and *noisy* scenarios, `ARFL` achieves test accuracy very close to the best method in the comparison. In the *clean* scenario, `FedAvg` and `RFA` show marginally higher test accuracy than `ARFL` in digits after the decimal point. This may be due to the rounding errors in the estimation of the weights of clients in `ARFL`.

As expected, `FedAvg`'s performance is significantly affected by the presence of corrupted clients, especially in *shuffling* and *flipping* scenarios. Furthermore, `MKRUM` also shows poor performance in *shuffling* and *flipping* scenarios of all datasets. `RFA` works well for the FEMNIST dataset, but worse than `FedAvg` in the *shuffling* and *flipping* scenarios for the CIFAR-10 and Shakespere datasets. It is also interesting to observe that `CFL` works well for the FEMNIST and Shakespeare datasets under 30% corruption level, but the accuracy decreases significantly when the corruption level is 50%. The reason is that when half of the clients are corrupted, it fails to identify which group of clients are corrupted. These results support the benefits of `ARFL`, which, in general, offers better performance than the existing approaches across the corruption scenarios we consider. Note that our approach can handle even higher corruption rates in those scenarios. For example, using the FEMNIST data set with 70% corrupted clients we still achieve an accuracy above 79%. However, it is also noticeable that if multiple clients try to bias their data to the same distribution (i.e., colluding corruption), our approach is unable to handle such a high corruption rate.

Next, we study the convergence of the approaches by comparing the test loss and accuracy of the global model versus the number of training rounds in Figure 2, where 50% of the clients are corrupted. As discussed before, `CFL` is unable to handle such a high corruption level. Therefore we only compare the remaining four approaches. The shaded areas denote the minimum and maximum values over five repeated runs. The figure shows that all approaches converge to a good solution in the *clean* and *noisy* scenarios. However, in the *clean* scenario, the test loss of `ARFL` is slightly higher than the



(a) Weights v.s. λ



(b) Accuracy v.s. λ

Fig. 3: Effects of λ on weights and model accuracy on the CIFAR-10 dataset. The corrupted clients (dashed lines) are zero-weighted when $\lambda \leq 2.5 \times M$, and the accuracy reaches its peak when $\lambda = 1 \times M$.

others. The reason is that the global model could bias towards some of the local updates, which can be avoided by increasing λ. Furthermore, all the RFA, FedAvg and MKRUM approaches diverge in the *shuffling* and *flipping* corruption scenarios, which indicates that the local data corruption harms the global model during their training process. On the contrary, our ARFL method is able to converge with high accuracy, since it is able to lower the contribution of the corrupted clients. In the *clean* and *noisy* scenarios, we observe that MKRUM converges slower, as it only uses a subset of selected updates to aggregate the global model.

C. Effects of λ

As mentioned before, λ in the objective should be tuned in order to provide a trade-off between robustness and average accuracy. Here we conduct further experiments on the CIFAR-10 dataset to study the effects of λ. We simulate a non-i.i.d. partition for which the number of data points and class proportions are unbalanced. The dataset is partitioned into $N = 10$ clients by sampling $\boldsymbol{P}_k \sim Dir_N(0.5)$ and allocating a $\boldsymbol{P}_{k,i}$ proportion of the training instances of class k to local client i, where three clients are corrupted by shuffling their labels randomly (Client 8-10). We use the original test set in CIFAR-10 as the global test set. We train the model for 1000 rounds where each client runs one epoch of SGD on their training set before each aggregation, where λ is set in the range of $[10^{-1}, 10^2]$. All the other settings are the same as in Sec. V-A.

Fig. 3(a) shows the optimized weights as a function of λ on the CIFAR-10 dataset. It is readily apparent that $\boldsymbol{\alpha}$ has only one non-zero element (Client 7) for small λ and all elements of $\boldsymbol{\alpha}$ come to certain non-zero values for large λ. In between these two extremes, we obtain sparse solutions of $\boldsymbol{\alpha}$ in which only a part of elements have non-zero values. It is also noticeable that all the corrupted clients (dashed lines) are zero-weighted when

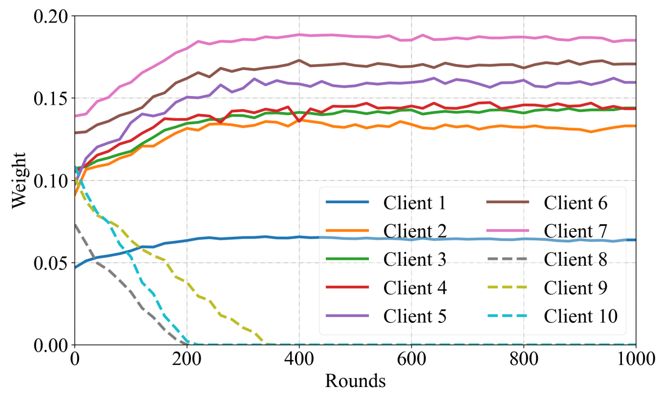


Fig. 4: Visualization of the weights v.s. round number of ARFL. All corrupted clients (dashed lines) are zero-weighted after 350 rounds of training.

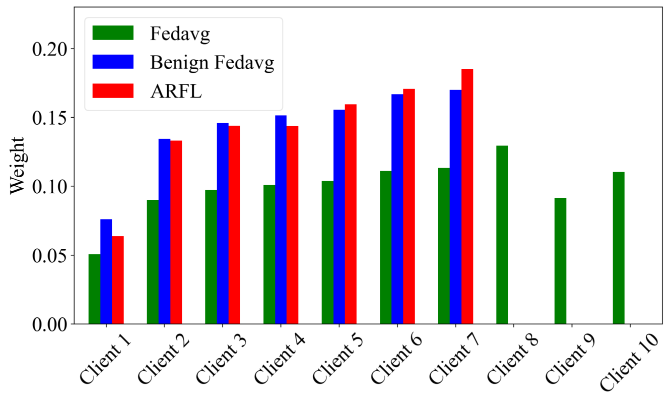


Fig. 5: Weights comparison between ARFL and FedAvg. ARFL's weights distributed similar to those of FedAvg with only benign clients.

$\lambda \leq 2.5 \times M$, which means that they make no contribution when the server aggregates the updated local models.

In Fig. 3(b), we plot the model accuracy as a function of λ, showing that the model achieves relatively low accuracy for small λ, which demonstrates that extremely sparse weights are not favorable under this non-i.i.d. data setting. The reason behind this is that the model finally fits only one local dataset without considering data from other clients. The accuracy increases as more benign clients are upweighted and contribute their local updates to the global model. The optimal value for λ is $1.0 \times M$ in this experiment, where all the benign clients have non-zero weights, while all the corrupted clients have zero weights. However, as λ further increases, the global model is harmed by the corrupted clients as they gain adequate weights, which leads to lower accuracy.

D. Auto-weighting Analysis

In this part, we investigate the auto-weighting process during training in ARFL when $\lambda = 1.0 \times M$, where we keep the setup the same as the previous subsection. As shown in Fig. 4, our approach successfully downweights all the three corrupted clients, for which the weights become zero after 350 rounds of training.

We next compare the reallocated weights with the standard FedAvg, where the weights are fixed as $\alpha_i = \frac{m_i}{M}$, and FedAvg with only benign clients (Benign FedAvg), where $\alpha_i = \frac{m_i}{M_{\mathcal{B}}}$ for benign clients, and $\alpha_i = 0$ for corrupted clients. Fig. 5 shows that the learned weight distribution (in red) of ARFL is approximated to the distribution considering only benign clients (in blue). We conclude that our approach can automatically reweight the clients and approximate the mixture distribution to the benign uniform distribution during the model training process, even when the centralized server is agnostic about the local corruptions.

VI. Conclusions and Future work

In this paper, we proposed Auto-weighted Robust Federated Learning (ARFL), a novel approach that automatically reweights the local updates to lower the contribution of corrupted clients who provide low-quality updates to the global model. Experimental results on benchmark datasets corroborate the competitive performance of ARFL compared to the state-of-the-art methods under different data corruption scenarios. Our future work will focus on robust aggregation without the losses provided by clients since evaluating training loss from local clients could also be a potential overhead. Furthermore, we plan to extend ARFL to a more general federated learning approach that is robust against both model poisoning and data corruption.

References

[1] 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*, pp. 1273–1282, 2017.

[2] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[3] 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.

[4] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," *arXiv preprint arXiv:1610.05492*, 2016.

[5] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied federated learning: Improving google keyboard query suggestions," *arXiv preprint arXiv:1812.02903*, 2018.

[6] Q. Yang, Y. Liu, Y. Cheng, Y. Kang, T. Chen, and H. Yu, "Federated learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 13, no. 3, pp. 1–207, 2019.

[7] M. Chen, A. T. Suresh, R. Mathews, A. Wong, C. Allauzen, F. Beaufays, and M. Riley, "Federated learning of n-gram language models," in *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pp. 121–130, 2019.

[8] J. Luo, X. Wu, Y. Luo, A. Huang, Y. Huang, Y. Liu, and Q. Yang, "Real-world image datasets for federated learning," *arXiv preprint arXiv:1910.11089*, 2019.

[9] D. Guliani, F. Beaufays, and G. Motta, "Training speech recognition models with federated learning: A quality/cost framework," *arXiv preprint arXiv:2010.15965*, 2020.

[10] N. Rodríguez-Barroso, E. Martínez-Cámara, M. Luzón, G. G. Seco, M. Á. Veganzones, and F. Herrera, "Dynamic federated learning model for identifying adversarial clients," *arXiv preprint arXiv:2007.15030*, 2020.

[11] S. Mahloujifar, M. Mahmoody, and A. Mohammed, "Data poisoning attacks in multi-party learning," in *International Conference on Machine Learning*, pp. 4274–4283, 2019.

[12] X. Liu and E. Ngai, "Gaussian process learning for distributed sensor networks under false data injection attacks," in *2019 IEEE Conference on Dependable and Secure Computing (DSC)*, pp. 1–6, IEEE, 2019.

[13] P. Wais, S. Lingamneni, D. Cook, J. Fennell, B. Goldenberg, D. Lubarov, D. Marin, and H. Simons, "Towards building a high-quality workforce with mechanical turk," in *In Proc. NIPS Workshop on Computational Social Science and the Wisdom of Crowds*, Citeseer, 2010.

[14] P. Blanchard, R. Guerraoui, J. Stainer, *et al.*, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems*, pp. 119–129, 2017.

[15] K. Pillutla, S. M. Kakade, and Z. Harchaoui, "Robust aggregation for federated learning," *arXiv preprint arXiv:1912.13445*, 2019.

[16] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning*, pp. 5650–5659, 2018.

[17] X. Li, N. Liu, C. Chen, Z. Zheng, H. Li, and Q. Yan, "Communication-efficient collaborative learning of geo-distributed jointcloud from heterogeneous datasets," in *2020 IEEE International Conference on Joint Cloud Computing*, pp. 22–29, IEEE, 2020.

[18] X. Peng, Z. Huang, Y. Zhu, and K. Saenko, "Federated adversarial domain adaptation," in *International Conference on Learning Representations*, 2019.

[19] D. Sui, Y. Chen, J. Zhao, Y. Jia, Y. Xie, and W. Sun, "Feded: Federated learning via ensemble distillation for medical relation extraction," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2118–2128, 2020.

[20] Y. Han and X. Zhang, "Robust federated training via collaborative machine teaching using trusted instances," *arXiv preprint arXiv:1905.02941*, 2019.

[21] Y. Han and X. Zhang, "Robust federated learning via collaborative machine teaching.," in *AAAI*, pp. 4075–4082, 2020.

[22] N. Konstantinov and C. Lampert, "Robust learning from untrusted sources," in *International Conference on Machine Learning*, pp. 3488–3498, 2019.

[23] S. Zheng, Z. Huang, and J. Kwok, "Communication-efficient distributed blockwise momentum sgd with error-feedback," in *Advances in Neural Information Processing Systems*, pp. 11450–11460, 2019.

[24] F. Sattler, K.-R. Müller, T. Wiegand, and W. Samek, "On the byzantine robustness of clustered federated learning," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8861–8865, IEEE, 2020.

[25] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 10, no. 2, pp. 1–19, 2019.

[26] Y. Chen, L. Su, and J. Xu, "Distributed statistical machine learning in adversarial settings: Byzantine gradient descent," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 1, no. 2, pp. 1–25, 2017.

[27] C. C. Aggarwal, A. Hinneburg, and D. A. Keim, "On the surprising behavior of distance metrics in high dimensional space," in *International conference on database theory*, pp. 420–434, Springer, 2001.

[28] S. Li, Y. Cheng, W. Wang, Y. Liu, and T. Chen, "Learning to detect malicious clients for robust federated learning," *arXiv preprint arXiv:2002.00211*, 2020.

[29] M. Mohri, G. Sivek, and A. T. Suresh, "Agnostic federated learning," in *International Conference on Machine Learning*, pp. 4615–4625, 2019.

[30] J. Hamer, M. Mohri, and A. T. Suresh, "Fedboost: A communication-efficient algorithm for federated learning," in *International Conference on Machine Learning*, pp. 3973–3983, PMLR, 2020.

[31] J. Wang and G. Joshi, "Cooperative sgd: A unified framework for the design and analysis of communication-efficient sgd algorithms," in *ICML Workshop on Coding Theory for Machine Learning*, 2019.

[32] S. U. Stich, "Local sgd converges fast and communicates little," in *International Conference on Learning Representations*, 2018.

[33] H. Yu, S. Yang, and S. Zhu, "Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 5693–5700, 2019.

[34] R. Guerraoui, S. Rouault, *et al.*, "The hidden vulnerability of distributed learning in byzantium," in *International Conference on Machine Learning*, pp. 3521–3530, 2018.

[35] D. Yin, R. Kannan, and P. Bartlett, “Rademacher complexity for adversarially robust generalization,” in *International Conference on Machine Learning*, pp. 7085–7094, PMLR, 2019.
[36] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.
[37] L. Grippo and M. Sciandrone, “On the convergence of the block nonlinear gauss–seidel method under convex constraints,” *Operations research letters*, vol. 26, no. 3, pp. 127–136, 2000.
[38] T. Li, M. Sanjabi, A. Beirami, and V. Smith, “Fair resource allocation in federated learning,” in *International Conference on Learning Representations*, 2019.
[39] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, “Federated learning with matched averaging,” in *International Conference on Learning Representations*, 2019.
[40] D. A. van Dyk and X.-L. Meng, “The art of data augmentation,” *Journal of Computational and Graphical Statistics*, pp. 1–50, 2001.
[41] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, “Emnist: Extending mnist to handwritten letters,” in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2921–2926, IEEE, 2017.
[42] S. Caldas, P. Wu, T. Li, J. Konečnỳ, H. B. McMahan, V. Smith, and A. Talwalkar, “Leaf: A benchmark for federated settings,” *arXiv preprint arXiv:1812.01097*, 2018.
[43] W. Shakespeare, *The complete works of William Shakespeare*. Publicly available at http://www.gutenberg.org/ebooks/100/.

Appendix A
Proof of Theorem 1

Theorem 1. *We denote $\hat{\mathcal{L}}_i(h)$ as the corresponding empirical counterparts of $\mathcal{L}_{\mathcal{D}_i}(h)$. Assume that the loss function is bounded by a constant $\mathcal{M} > 0$. Then, for any $\delta > 0$, with probability at least $1-\delta$ over the data, the following inequality holds:*

$$\mathcal{L}_{\mathcal{D}_{\boldsymbol{\alpha}}}(h) \le \sum_{i=1}^{N} \alpha_i \hat{\mathcal{L}}_i(h) + 2\sum_{i=1}^{N} \alpha_i \mathcal{R}_i(\mathcal{H}) + 3\sqrt{\frac{\log\left(\frac{4}{\delta}\right)\mathcal{M}^2}{2}}\sqrt{\sum_{i=1}^{n}\frac{\alpha_i^2}{m_i}}, \tag{4}$$

where, for each client $i = 1, \dots, N$,

$$\mathcal{R}_i(\mathcal{H}) = \mathbb{E}_{\sigma}\left(\sup_{f\in\mathcal{H}}\left(\frac{1}{m_i}\sum_{j=1}^{m_i}\sigma_{i,j}\ell_f(z_{i,j})\right)\right)$$

and $\sigma_{i,j}$s are independent uniform random variables taking values in $\{-1,+1\}$. These variables are called Rademacher random variables [35]

Proof. Write:

$$\mathcal{L}_{\mathcal{D}_{\boldsymbol{\alpha}}}(h) \le \hat{\mathcal{L}}_{\mathcal{D}_{\boldsymbol{\alpha}}}(h) + \sup_{f\in\mathcal{H}}\left(\mathcal{L}_{\mathcal{D}_{\boldsymbol{\alpha}}}(f) - \hat{\mathcal{L}}_{\mathcal{D}_{\boldsymbol{\alpha}}}(f)\right) \tag{11}$$

To link the second term to its expectation, we prove the following:

Lemma 1. *Define the function $\phi : (\mathcal{X}\times\mathcal{Y})^m \to \mathbb{R}$ by:*

$$\phi(\{x_{1,1},y_{1,1}\},\dots,\{x_{N,m_N},y_{N,m_N}\}) = \sup_{f\in\mathcal{H}}\left(\mathcal{L}_{\mathcal{D}_{\boldsymbol{\alpha}}}(f) - \hat{\mathcal{L}}_{\mathcal{D}_{\boldsymbol{\alpha}}}(f)\right).$$

Denote for brevity $z_{i,j} = \{x_{i,j}, y_{i,j}\}$. Then, for any $i \in \{1,2,\dots,N\}, j \in \{1,2,\dots,m_i\}$:

$$\sup_{z_{1,1},\dots,z_{N,m_N},z'_{i,j}} |\phi(z_{1,1},\dots,z_{i,j},\dots,z_{N,m_N}) - \phi\left(z_{1,1},\dots,z'_{i,j},\dots,z_{N,m_N}\right)| \le \frac{\alpha_i}{m_i}\mathcal{M} \tag{12}$$

Proof. Fix any i, j and any $z_{1,1},\dots,z_{N,m_N},z'_{i,j}$. Denote the α-weighted empirical average of the loss with respect to the sample $z_{1,1},\dots,z'_{i,j},\dots,z_{N,m_N}$ by $\mathcal{L}'_{\mathcal{D}_{\boldsymbol{\alpha}}}$. Then we have that:

$$\begin{aligned}
&|\phi(\dots,z_{i,j},\dots) - \phi(\dots,z'_{i,j},\dots)| \\
&= |\sup_{f\in\mathcal{H}}\left(\mathcal{L}_{\mathcal{D}_{\boldsymbol{\alpha}}}(f) - \hat{\mathcal{L}}_{\mathcal{D}_{\boldsymbol{\alpha}}}(f)\right) - \sup_{f\in\mathcal{H}}(\mathcal{L}_{\mathcal{D}_{\boldsymbol{\alpha}}}(f) - \hat{\mathcal{L}}'_{\mathcal{D}_{\boldsymbol{\alpha}}}(f))| \\
&\le |\sup_{f\in\mathcal{H}}(\hat{\mathcal{L}}'_{\mathcal{D}_{\boldsymbol{\alpha}}}(f) - \hat{\mathcal{L}}_{\mathcal{D}_{\boldsymbol{\alpha}}}(f))| \\
&= \frac{\alpha_i}{m_i}|\sup_{f\in\mathcal{H}}\left(\ell_f(z'_{i,j}) - \ell_f(z_{i,j})\right)| \\
&\le \frac{\alpha_i}{m_i}\mathcal{M}
\end{aligned}$$

Note: the inequality we used above holds for bounded functions inside the supremum. □

Let S denote a random sample of size m drawn from a distribution as the one generating out data (i.e. m_i samples from $\mathcal{D}_i$ for each i). Now, using Lemma 1, McDiarmid's inequality gives:

$$\begin{aligned}
\mathbb{P}(\phi(S) - \mathbb{E}(\phi(S)) \ge t) &\le \exp\left(-\frac{2t^2}{\sum_{i=1}^{N}\sum_{j=1}^{m_i}\frac{\alpha_i^2}{m_i^2}\mathcal{M}^2}\right) \\
&= \exp\left(-\frac{2t^2}{\mathcal{M}^2\sum_{i=1}^{N}\frac{\alpha_i^2}{m_i}}\right)
\end{aligned}$$

For any $\delta > 0$, setting the right-hand side above to be $\delta/4$ and using (11), we obtain that with probability at least $1-\delta/4$:

$$\mathcal{L}_{\mathcal{D}_{\boldsymbol{\alpha}}}(h) \le \hat{\mathcal{L}}_{\mathcal{D}_{\boldsymbol{\alpha}}}(h) + \mathbb{E}_S\left(\sup_{f\in\mathcal{H}}\left(\mathcal{L}_{\mathcal{D}_{\boldsymbol{\alpha}}}(f) - \hat{\mathcal{L}}_{\mathcal{D}_{\boldsymbol{\alpha}}}(f)\right)\right) + \sqrt{\frac{\log\left(\frac{4}{\delta}\right)\mathcal{M}^2}{2}}\sqrt{\sum_{i=1}^{N}\frac{\alpha_i^2}{m_i}} \tag{13}$$

To deal with the expected loss inside the second term, introduce a ghost sample (denoted by S'), drawn from the same distributions as our original sample (denoted by S). Denoting the weighted empirical loss with respect to the ghost sample by $\mathcal{L}'_{\mathcal{D}_{\boldsymbol{\alpha}}}$, $\beta_i = m_i/m$ for all i, and using the convexity of the supremum, we obtain:

$$\begin{aligned}
&\mathbb{E}_S\left(\sup_{f\in\mathcal{H}}\left(\mathcal{L}_{\mathcal{D}_{\boldsymbol{\alpha}}}(f) - \hat{\mathcal{L}}_{\mathcal{D}_{\boldsymbol{\alpha}}}(f)\right)\right) \\
&= \mathbb{E}_S\left(\sup_{f\in\mathcal{H}}\left(\mathbb{E}_{S'}\left(\hat{\mathcal{L}}'_{\mathcal{D}_{\boldsymbol{\alpha}}}(f)\right) - \hat{\mathcal{L}}_{\mathcal{D}_{\boldsymbol{\alpha}}}(f)\right)\right) \\
&\le \mathbb{E}_{S,S'}\left(\sup_{f\in\mathcal{H}}\left(\hat{\mathcal{L}}'_{\mathcal{D}_{\boldsymbol{\alpha}}}(f) - \hat{\mathcal{L}}_{\mathcal{D}_{\boldsymbol{\alpha}}}(f)\right)\right) \\
&= \mathbb{E}_{S,S'}\left(\sup_{f\in\mathcal{H}}\left(\frac{1}{m}\sum_{i=1}^{N}\sum_{j=1}^{m_i}\frac{\alpha_i}{\beta_i}\left(\ell_f(z'_{i,j}) \quad -\ell_f(z_{i,j})\right)\right)\right)
\end{aligned}$$

Introducing m independent Rademacher random variables and noting that $(\ell_f(z') - \ell_f(z))$ and $\sigma(\ell_f(z') - \ell_f(z))$ have the same distribution, as long as $\boldsymbol{z}$ and $\boldsymbol{z}'$ have the same distribution:

$$\begin{aligned}
&\mathbb{E}_S\left(\sup_{f\in\mathcal{H}}\left(\mathcal{L}_{\mathcal{D}_{\boldsymbol{\alpha}}}(f) - \hat{\mathcal{L}}_{\mathcal{D}_{\boldsymbol{\alpha}}}(f)\right)\right) \\
&\quad\le \mathbb{E}_{S,S',\sigma}\left(\sup_{f\in\mathcal{H}}\left(\frac{1}{m}\sum_{i=1}^{N}\sum_{j=1}^{m_i}\frac{\alpha_i}{\beta_i}\sigma_{i,j}\left(\ell_f(z_{i,j})'\right)\right.\right. \\
&\quad\quad \left.\left.-\ell_f(z_{i,j})\right)\right) \\
&\quad\le \mathbb{E}_{S',\sigma}\left(\sup_{f\in\mathcal{H}}\left(\frac{1}{m}\sum_{i=1}^{N}\sum_{j=1}^{m_i}\frac{\alpha_i}{\beta_i}\sigma_{i,j}\ell_f(z_{i,j})\right)\right) \\
&\quad+ \mathbb{E}_{S,\sigma}\left(\sup_{f\in\mathcal{H}}\left(\frac{1}{m}\sum_{i=1}^{N}\sum_{j=1}^{m_i}\frac{\alpha_i}{\beta_i}(-\sigma_{i,j})\,\ell_f(z_{i,j})\right)\right) \\
&\quad= 2\mathbb{E}_{S,\sigma}\left(\sup_{f\in\mathcal{H}}\left(\frac{1}{m}\sum_{i=1}^{N}\sum_{j=1}^{m_i}\frac{\alpha_i}{\beta_i}\sigma_{i,j}\ell_f(z_{i,j})\right)\right)
\end{aligned}$$

We can now link the last term to the empirical analog of the Rademacher complexity, by using the McDiarmid Inequality (with an observation similar to Lemma 1). Putting this together, we obtain that for any $\delta > 0$ with probability at least $1 - \delta/2$:

$$\begin{aligned}
\mathcal{L}_{\mathcal{D}_{\boldsymbol{\alpha}}}(h) \le \hat{\mathcal{L}}_{\mathcal{D}_{\boldsymbol{\alpha}}}(h) \\
+ 2\mathbb{E}_{\sigma}\left(\sup_{f\in\mathcal{H}}\left(\frac{1}{m}\sum_{i=1}^{N}\sum_{j=1}^{m_i}\frac{\alpha_i}{\beta_i}\sigma_{i,j}\ell_f(z_{i,j})\right)\right) \\
+ 3\sqrt{\frac{\log\left(\frac{4}{\delta}\right)M^2}{2}}\sqrt{\sum_{i=1}^{N}\frac{\alpha_i^2}{m_i}}
\end{aligned} \tag{14}$$

Finally, note that:

$$\begin{aligned}
&\mathbb{E}_{\sigma}\left(\sup_{f\in\mathcal{H}}\left(\frac{1}{m}\sum_{i=1}^{N}\sum_{j=1}^{m_i}\frac{\alpha_i}{\beta_i}\sigma_{i,j}\ell_f(z_{i,j})\right)\right) \\
&\quad\le \mathbb{E}_{\sigma}\left(\sum_{i=1}^{N}\alpha_i\sup_{f\in\mathcal{H}}\left(\frac{1}{m_i}\sum_{j=1}^{m_i}\sigma_{i,j}\ell_f(z_{i,j})\right)\right) \\
&\quad= \sum_{i=1}^{N}\alpha_i\mathbb{E}_{\sigma}\left(\sup_{f\in\mathcal{H}}\left(\frac{1}{m_i}\sum_{j=1}^{m_i}\sigma_{i,j}\ell_f(z_{i,j})\right)\right) \\
&\quad= \sum_{i=1}^{N}\alpha_i\mathcal{R}_i(\mathcal{H})
\end{aligned}$$

Bounding $\hat{\mathcal{L}}_{\mathcal{D}_{\boldsymbol{\alpha}}}(h) - \mathcal{L}_{\mathcal{D}_{\boldsymbol{\alpha}}}(h)$ with the same quantity and with probability at least $1 - \delta/2$ follows by a similar argument. The result then follows by applying the union bound. □

Appendix B
Proof of Theorem 2

Theorem 2. *For any $\boldsymbol{w}$, when $\lambda > 0$ and $\{\hat{\mathcal{L}}_i(\boldsymbol{w})\}_{i=1}^N$ are sorted in increasing order: $\hat{\mathcal{L}}_1(\boldsymbol{w}) \le \hat{\mathcal{L}}_2(\boldsymbol{w}) \le ... \le \hat{\mathcal{L}}_N(\boldsymbol{w})$, by setting:*

$$p = \underset{k}{\operatorname{argmax}}\{1 + \frac{M_k(\overline{\mathcal{L}}_k(\boldsymbol{w}) - \hat{\mathcal{L}}_k(\boldsymbol{w}))}{\lambda} > 0\}, \tag{6}$$

where $M_k = \sum_{i=1}^k m_i$,

$$\overline{\mathcal{L}}_k(h) = \frac{\sum_{i=1}^k m_i\hat{\mathcal{L}}_i(\boldsymbol{w})}{M_k} \tag{7}$$

is the average loss over the first k clients that have the smallest empirical risks. Then the optimal $\boldsymbol{\alpha}$ to the problem (5) is given by:

$$\alpha_i(\boldsymbol{w}) = \frac{m_i}{M_p}[1 + \frac{M_p(\overline{\mathcal{L}}_p(\boldsymbol{w}) - \hat{\mathcal{L}}_i(\boldsymbol{w}))}{\lambda}]_+, \tag{8}$$

where $[\cdot]_+ = max(0, \cdot)$.

Proof. The Lagrangian function of Eq. (5) is

$$\mathbb{L} = \boldsymbol{\alpha}^\top\hat{\mathcal{L}}(\boldsymbol{w}) + \frac{\lambda}{2}||\boldsymbol{\alpha}^\top\boldsymbol{m}^{\circ-\frac{1}{2}}||_2^2 - \boldsymbol{\alpha}^\top\boldsymbol{\beta} - \eta(\boldsymbol{\alpha}^\top\mathbf{1} - 1), \tag{15}$$

where $\hat{\mathcal{L}}(\boldsymbol{w}) = [\hat{\mathcal{L}}_1(\boldsymbol{w}), \hat{\mathcal{L}}_2(\boldsymbol{w}), ..., \hat{\mathcal{L}}_N(\boldsymbol{w})]^\intercal$, $\circ$ is the Hadamard root operation, $\boldsymbol{\beta}$ and η are the Lagrangian multipliers. Then the following Karush-Kuhn-Tucker (KKT) conditions hold:

$$\partial_{\boldsymbol{\alpha}}\mathbb{L}(\boldsymbol{\alpha}, \boldsymbol{\beta}, \eta) = 0, \tag{16}$$

$$\boldsymbol{\alpha}^\intercal\mathbf{1} - 1 = 0, \tag{17}$$

$$\boldsymbol{\alpha} \ge 0, \tag{18}$$

$$\boldsymbol{\beta} \ge 0, \tag{19}$$

$$\alpha_i\beta_i = 0, \forall i = 1, 2, ...N. \tag{20}$$

According to Eq. (16), we have:

$$\alpha_i = \frac{m_i(\beta_i + \eta - \hat{\mathcal{L}}_i(\boldsymbol{w}))}{\lambda}. \tag{21}$$

Since $\beta_i \ge 0$, we discuss the following cases:

1) When $\beta_i = 0$, we have $\alpha_i = \frac{m_i(\eta - \hat{\mathcal{L}}_i(\boldsymbol{w}))}{\lambda} \ge 0$. Note that we further have $\eta - \hat{\mathcal{L}}_i(\boldsymbol{w}) \ge 0$.
2) When $\beta_i > 0$, from the condition $\alpha_i\beta_i = 0$, we have $\alpha_i = 0$.

Therefore, the optimal solution to Eq. (5) is given by:

$$\alpha_i(\boldsymbol{w}) = [\frac{m_i(\eta - \hat{\mathcal{L}}_i(\boldsymbol{w}))}{\lambda}]_+, \tag{22}$$

where $[\cdot]_+ = max(0, \cdot)$.

We notice that $\sum_{i=1}^p \alpha_i = 1$, thus we can get:

$$\eta = \frac{\sum_{i=1}^p m_i\hat{\mathcal{L}}_i(\boldsymbol{w}) + \lambda}{\sum_{i=1}^p m_i}. \tag{23}$$

According to $\eta - \hat{\mathcal{L}}_i(\boldsymbol{w}) \ge 0$, we have Eq. (6) and Eq. (7). Finally, plugging Eq. (23) into Eq. (22) yields Eq. (8). □