Non-IID Distributed Learning with Optimal Mixture Weights

Jian Li[1][0000−0003−4977−1802], Bojian Wei*[1,2][0000−0001−5882−2523], Yong Liu(✉)[3][0000−0002−6739−621X], and Weiping Wang[1][0000−0002−8618−4992]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3] Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China
{lijian9026,wangweiping}@iie.ac.cn, szwboj@126.com, liuyonggsai@ruc.edu.cn

Abstract. Distributed learning can well solve the problem of training model with large-scale data, which has attracted much attention in recent years. However, most existing distributed learning algorithms set uniform mixture weights across clients when aggregating the global model, which impairs the accuracy under Non-IID (Not Independently or Identically Distributed) setting. In this paper, we present a general framework to optimize the mixture weights and show that our framework has lower expected loss than the uniform mixture weights framework theoretically. Moreover, we provide strong generalization guarantee for our framework, where the excess risk bound can converge at $\mathcal{O}(1/n)$, which is as fast as centralized training. Motivated by the theoretical findings, we propose a novel algorithm to improve the performance of distributed learning under Non-IID setting. Through extensive experiments, we show that our algorithm outperforms other mainstream methods, which coincides with our theory.

Keywords: Distributed learning · Excess risk bound · Optimal mixture weights.

1 Introduction

With the development of Internet of Things (IoT) technology and the popularity of intelligent terminal devices, it is difficult to continue the traditional centralized training of machine learning algorithms. Fortunately, distributed learning [23, 29, 2] provides an effective way for model training with large-scale data.

In standard distributed learning, many clients collaboratively train a global model under the coordination of a central server, where the training samples are splited on clients to alleviate the storage and computing limitations of the server. Recently, there are many studies analyze the properties of distributed learning from different perspectives [18, 3]. `EasyASR` [26] provides a distributed

* Jian Li and Bojian Wei contribute equally to this work.

platform for training and serving large-scale automatic speech recognition models, which supports both pre-defined networks and user-customized networks. In high-dimensional settings, Acharya et al. [1] analyzed the communication problem in distributed learning, and obtained algorithms that enjoy optimal error with logarithmic communication by relaxing the boundedness assumptions. Random topologies [30] is applied to tackle the unreliable networks problem in distributed learning, which can achieve comparable convergence rate to centralized learning. `MRE` [24] aims to reduce the error in IID distributed learning, where the error bound meets the existing lower bounds up to poly-logarithmic factors. Deep Q-learning based synchronization policies [31] is used for parameter server-based distributed training, which can generalize to different cluster environments and datasets.

With the increasing attention paid to privacy-preserving, data sharing in distributed learning has been strictly limited. Thus, federated learning [19, 28] was proposed to maintain or improve the performance of distirbuted learning while protecting users' privacy. However, local distributions on different clients may be different due to the personality of users, which brings us the Non-IID problem in distributed learning, where the global model is difficult to converge to the optimal solution. To solve the problem, `FedAvg` [19] runs multi-step SGD (stochastic gradient descent) on clients and aggregates local models by periodically communications. `FedProx` [16] introduces a proximal term to constrain the divergence between local models and the global model. There are many other studies try to tackle the Non-IID problem by different algorithms [27, 22, 11].

Although many related work has presented various methods to improve the performance of Non-IID distributed learning, the mixture weights for model aggregation are usually fixed as $\frac{n_k}{n}$, where n_k denotes the sample size on the k-th client and n denotes the total sample size among all clients. In fact, the uniform mixture weights is a good choice under IID settings, but it can not reflect the heterogeneous characteristics. When we use the uniform mixture weights to Non-IID distributed learning, the global model will shift to the local model with larger sample size, which impairs the performance of global model. Furthermore, most existing algorithms for distributed learning lack generalization guarantees, which restricts their portability to some extent.

In this paper, we present a general framework for Non-IID distributed learning, where the mixtures weights can be optimized to promote the global model to converge to the optimal solution. We also provid a strong generalization guarantee for our distributed learning framework based on local Rademacher complexity. The main contributions of this paper are summarized as follows.

- **A general framework.** We present a general framework for Non-IID distributed learning, where the mixtures weights are optimized together with model parameters by minimizing the objective. Theoretically, we demonstrate that our framework has lower expected loss than distributed learning with uniform mixture weights.
- **A strong generalization guarantee.** To our best knowledge, we derive a sharper excess risk bound for Non-IID distributed learning with convergence

rate of $\mathcal{O}(1/n)$ based on local Rademacher complexity for the first time, which meets the current bounds in centralized learning and is much faster than the existing bounds of distributed learning.
- **A novel algorithm.** Based on our general framework and theoretical findings, we propose a novel distributed learning algorithm `DL-opt`, which optimizes the mixture weights on server-side with validation samples and constrains local Rademacher complexity with an additional regularization term on the local objective. Through extensive experiments, we show that `DL-opt` significantly outperforms distributed learning with uniform mixture weights.
- **An effective extension to federated learning.** We extend `DL-opt` to federated learning, named `FedOMW`, which executes periodical communications to alternately optimize the mixture weights and model parameters. We illustrate that `FedOMW` performs better than `FedAvg` and `FedProx` with a clear margin through a series of experiments.

2 Preliminaries and Notations

In this section, we first introduce the Non-IID distributed learning scenario and then demonstrate the general notations used in this paper.

In a Non-IID distributed learning scenario, there are K clients and a central server, where the local training samples $\mathcal{D}_k = \{(\boldsymbol{x}_{ik}, y_{ik})\}_{i=1}^{n_k}$ on the k-th client are drawn from a local distribution ρ_k with size of n_k. The underlying local distribution is different on different clients: $\rho_i \neq \rho_j$. We denote $n = \sum_{k=1}^{K} n_k$ the total number of training samples across all clients.

Let $\mathcal{H}$ be the hypothesis space consisting of labeling functions $h : \mathcal{X} \to \mathcal{Y}$, where $\mathcal{X}$ denotes the input space and $\mathcal{Y}$ denotes the output space. The labeling function is formed as $h(\boldsymbol{x}) = \boldsymbol{w}^T \phi(\boldsymbol{x})$, where $\boldsymbol{w}$ denotes the vector of learnable parameters and $\phi(\cdot)$ denotes a fixed feature mapping. Let $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ be the loss function, we denote the loss space associated to $\mathcal{H}$ by $\mathcal{G} = \{\ell(h(\boldsymbol{x}), y) | h \in \mathcal{H}\}$. For the k-th client, we define the expected loss as

$$\mathcal{L}_k(h; \boldsymbol{w}) = \mathbb{E}_{(\boldsymbol{x},y)\sim\rho_k} \left[\ell(h(\boldsymbol{x}), y)\right],$$

and the corresponding empirical loss as

$$\widehat{\mathcal{L}}_k(h; \boldsymbol{w}) = \frac{1}{n_k} \sum_{i=1}^{n_k} \ell(h(\boldsymbol{x}_{ik}), y_{ik}).$$

The target of distributed learning is to obtain a global model. For traditional distributed learning, the global model $\boldsymbol{w}$ is obtained by aggregating local models ($\boldsymbol{w}_k$ denotes the local model on the k-th client) which are trained locally to converge on clients. For federated learning, the global model is obtained by alternately performing client-side local training and server-side model aggregating. The objective of distributed learning can be formed as

$$\min_{\boldsymbol{w}\in\mathcal{H}} \sum_{k=1}^{K} p_k \widehat{\mathcal{L}}_k(h; \boldsymbol{w}),$$

where p_k is the mixture weight of the k-th client.

Many distributed learning algorithms use uniform mixture weights ($p_k = n_k/n$) to aggregate local models, which make use of local sample sizes, and they work well when local training samples are independently drawn from an identical distribution (IID situation). However, under Non-IID setting, the uniform mixture weights fails to capture the discrepancy among local distributions. To this end, we consider optimizing the mixture weights together with $\boldsymbol{w}$ to get an optimal solution, which can truely minimize the objective under Non-IID setting.

Therefore, we present a general framework for Non-IID distributed learning, and the general objective is defined as

$$\min_{\boldsymbol{w}\in\mathcal{H}} \min_{\boldsymbol{p}\in\mathcal{P}} \widehat{\mathcal{L}}(h;\boldsymbol{w},\boldsymbol{p}) = \sum_{k=1}^{K} p_k \widehat{\mathcal{L}}_k(h;\boldsymbol{w}), \tag{1}$$

where $\boldsymbol{p} = [p_1, \cdots, p_K]$ is the vector of mixture weights and $\mathcal{P}$ is the parameter space of $\boldsymbol{p}$. The above objective is the empirical general loss of Non-IID distributed learning, and the corresponding expected general loss is $\mathcal{L}(h;\boldsymbol{w},\boldsymbol{p}) = \sum_{k=1}^{K} p_k \mathcal{L}_k(h;\boldsymbol{w})$.

In our framework, we relax the constraint on $\boldsymbol{p}$: the sum of K elements is 1 ($\sum_{k=1}^{K} p_k = 1$) and the value of each element p_k is in $(0, 1)$ and expands the range of feasibility, where each element p_k can take any value under the assumption that $|p_k|$ is upper bounded by τ ($\tau < \infty$). Thus, many gradient-based algorithms can be applied to optimize the mixture weights to get the optimal solution.

3 Generalization Guarantee

We introduce two specific estimators in hypothesis space $\mathcal{H}$: The empirical estimator is defined as

$$\widehat{h} = \operatorname*{arg\,min}_{\boldsymbol{w}\in\mathcal{H},\boldsymbol{p}\in\mathcal{P}} \widehat{\mathcal{L}}(h;\boldsymbol{w},\boldsymbol{p}),$$

and the optimal estimator is defined as

$$h^* = \operatorname*{arg\,min}_{\boldsymbol{w}\in\mathcal{H},\boldsymbol{p}\in\mathcal{P}} \mathcal{L}(h;\boldsymbol{w},\boldsymbol{p}),$$

where $\widehat{h}$ minimizes the empirical general loss of Non-IID distributed learning and h^* minimizes the corresponding expected general loss.

Excess risk is often used to represent the generalization performance of an estimator [7], which measures the gap between the empirical estimator and the optimal estimator. We define the excess risk of Non-IID distributed learning as follows:

$$\mathcal{L}(\widehat{h};\boldsymbol{w},\boldsymbol{p}) - \mathcal{L}(h^*;\boldsymbol{w},\boldsymbol{p}). \tag{2}$$

In the previous work, generalization error of centralized learning and distributed learning with uniform mixture weights has been widely studied, which is actually the upper bound of excess risk [7]. Through Rademacher complexity [6, 12] and stability theory [8, 9], the current generalization error bounds for centralized learning and distributed learning with uniform mixture weights converge at $\mathcal{O}(1/\sqrt{n})$. The convergence rate of generalization error bounds for centralized learning can be improved to $\mathcal{O}(1/n)$ by local Rademacher complexity [5] and some advanced techniques in stability. However, there is no existing work on the generalization error bounds for distributed learning with convergence rate of $\mathcal{O}(1/n)$.

In the following part, we will derive a sharper excess risk bound for Non-IID distributed learning to give a stronger generalization guarantee on the general framework defined in this paper.

3.1 Excess Risk Bound with Local Rademacher Complexity

We first introduce two important assumptions.

Assumption 1. *Assume that the loss function is λ-Lipschitz continuous and upper bounded by M ($M > 0$), that is*

$$|\ell(h(\boldsymbol{x}), y) - \ell(h(\boldsymbol{x}'), y')| \leq \lambda \left|h(\boldsymbol{x}) - h(\boldsymbol{x}')\right|$$

and

$$|\ell(h(\boldsymbol{x}), y)| < M, \quad \forall (\boldsymbol{x}, y) \in \mathcal{X} \times \mathcal{Y}.$$

Assumption 2. *Assume that the loss function satisfies the Bernstein condition: For some $B > 0$, it holds that*

$$\mathbb{E}\left[\ell(h(\boldsymbol{x}), y) - \ell(h^*(\boldsymbol{x}), y)\right]^2 \leq B\left(\mathcal{L}(h; \boldsymbol{w}, \boldsymbol{p}) - \mathcal{L}(h^*; \boldsymbol{w}, \boldsymbol{p})\right).$$

Assumption 1 is a commonly used assumption in generalization analysis [4, 25], where many loss functions meet this condition, such as hinge loss, margin loss and their variants. Meanwhile, Assumption 2 is widely used in statistical learning theory, such as local Rademacher complexity [5, 20, 10] and stability [14, 13, 8].

The empirical Rademacher complexity of $\mathcal{G}$ is formed as

$$\widehat{\mathcal{R}}(\mathcal{G}) = \mathbb{E}_{\boldsymbol{\epsilon}}\left[\sup_{h \in \mathcal{H}} \sum_{k=1}^{K} \frac{p_k}{n_k} \sum_{i=1}^{n_k} \epsilon_{ik} \ell(h(\boldsymbol{x}_{ik}), y_{ik})\right],$$

and the empirical Rademacher complexity of $\mathcal{H}$ is formed as

$$\widehat{\mathcal{R}}(\mathcal{H}) = \mathbb{E}_{\boldsymbol{\epsilon}}\left[\sup_{h \in \mathcal{H}} \sum_{k=1}^{K} \frac{p_k}{n_k} \sum_{i=1}^{n_k} \epsilon_{ik} h(\boldsymbol{x}_{ik})\right],$$

where $\{\epsilon_{ik}\}_{i\in[n_k]}^{k\in[K]}$ are independent Rademacher variables sampling uniformly from $\{-1,+1\}$.

We define the empirical local Rademacher complexity of $\mathcal{G}$ and $\mathcal{H}$ on training samples as follows:

$$\widehat{\mathcal{R}}(\mathcal{G},r)=\widehat{\mathcal{R}}\left(\{\ell_h|\ell_h\in\mathcal{G},\mathbb{E}[\ell_h-\ell_{h^*}]^2\leq r\}\right),$$
$$\widehat{\mathcal{R}}(\mathcal{H},r)=\widehat{\mathcal{R}}\left(\{h|h\in\mathcal{H},\mathbb{E}[\ell_h-\ell_{h^*}]^2\leq r\}\right),$$

where $\ell_h=\ell(h(\boldsymbol{x}),y)$, for simplicity.

Without loss of generality, the feature mapping $\phi(\cdot)$ mentioned Section 2 is assumed to be upper bounded by κ: $\kappa=\sup_{\boldsymbol{x}\in\mathcal{X}}\|\phi(\boldsymbol{x})\|\leq\infty$, and it is often used in kernel methods. Moreover, the depth and structure of neural networks are becoming deeper and more diverse in order to model more complex tasks, so the value of hidden vector after feature mapping should be constrained by normalization or other techniques to avoid training problems such as no convergence. Thus, this condition also applies to current deep learning methods.

We present the excess risk bound for Non-IID distributed learning with local Rademacher complexity in the following theorem.

Theorem 1 (Excess Risk Bound). *Let d be the VC dimension of hypothesis space $\mathcal{H}$, $\psi(r)$ be a sub-root function and r^* be the fixed point of $\psi(r)$. Assume that $\|\boldsymbol{w}\|^2\leq\frac{r}{\lambda^2\kappa^2}$, under Assumption 1 and 2, $\forall\delta\in(0,1]$ and $\forall r\geq r^*$, it holds that*

$$\psi(r)\geq\frac{\lambda\tau\sqrt{2dK\log[\frac{en}{d}]}}{n}\geq\lambda\mathbb{E}\left[\widehat{\mathcal{R}}(\mathcal{H})\right]. \tag{3}$$

With probability at least $1-\delta$, the following bound holds:

$$\mathcal{L}(\widehat{h};\boldsymbol{w},\boldsymbol{p})-\mathcal{L}(h^*;\boldsymbol{w},\boldsymbol{p})\leq\frac{705}{B}r^*+\frac{(11M+27B)\log(1/\delta)}{n}. \tag{4}$$

The proof is in Appendix A.1 of the supplementary file.

In Theorem 1, we derive a sharper excess risk bound for Non-IID distributed learning related to our general framework, which provides strong generalization guarantee for algorithms under our framework.

According to Theorem 1, the fixed point r^* dominates the excess risk of Non-IID distributed learning, which is affected by local Rademacher complexity with the sub-root function $\psi(r)$. In (3), we have proved that local Rademacher complexity can converge at $\mathcal{O}(1/n)$. Meanwhile, the rest part in (4) also has the convergence rate of $\mathcal{O}(1/n)$ due to the self-bounding property [5]:

$$\mathbb{E}\left[\widehat{\mathcal{R}}(\mathcal{H})\right]\leq\widehat{\mathcal{R}}(\mathcal{H})+\sqrt{\frac{2\mathbb{E}\left[\widehat{\mathcal{R}}(\mathcal{H})\right]\log(1/\delta)}{n}}.$$

Therefore, if we ignore the constants and other unrelevant factors, the excess risk bound for Non-IID distributed learning can be rewritten as

$$\mathcal{L}(\widehat{h};\boldsymbol{w},\boldsymbol{p})-\mathcal{L}(h^*;\boldsymbol{w},\boldsymbol{p})\leq\mathcal{O}\left(\frac{\sqrt{K}}{n}\right).$$

Note that K is the total number clients, so the above result shows that the generalization performance can be worse if there are too many clients participated in a Non-IID distributed learning, which is consistent with the actual application. Moreover, when there is only one client, the above result degrades into $\mathcal{O}(1/n)$, which meets the best excess risk bound for centralized learning. Thus, our general framework for Non-IID distributed learning has strong generalization guarantee and we provide a sharper excess risk bound for Non-IID distributed learning with convergence rate of $\mathcal{O}(1/n)$ for the first time.

3.2 Comparison with Current Framework

In this part, we will demonstrate that our general framework has lower expected loss than current distributed learning framework.

Traditional distributed learning framework uses uniform mixture weights ($p_k = \frac{n_k}{n}$) to aggregate the global model, we denote the empirical estimator of this uniform framework by $\widehat{h}_{\text{uf}}$. The expected loss of $\widehat{h}_{\text{uf}}$ is formed as $\mathcal{L}(\widehat{h}_{\text{uf}}; \boldsymbol{w}, \boldsymbol{p})$, we show that the expected loss of our framework is upper bounded by the expected loss of uniform framework in the following theorem.

Theorem 2. *Suppose that the distributed learning framework is applied to solve a binary-classification task, where $\mathcal{Y} = \{0, 1\}$. Let the loss function ℓ be the cross-entropy loss. For some $\mathcal{P}$ and ρ_k ($k \in [K]$), we have*

$$\mathcal{L}(\widehat{h}; \boldsymbol{w}, \boldsymbol{p}) \leq \mathcal{L}(\widehat{h}_{\text{uf}}; \boldsymbol{w}, \boldsymbol{p}). \tag{5}$$

Proof (Proof of Theorem 2). For simplicity, we consider that there are only two clients and the sample sizes n_k are equal. Given a single point $\boldsymbol{x}$, we assume that the local distribution on the first client satisfies $\rho_1(\boldsymbol{x}, 0) = 0$, $\rho_1(\boldsymbol{x}, 1) = 1$, and the local distribution on the second client satisfies $\rho_2(\boldsymbol{x}, 0) = \frac{1}{2}$, $\rho_2(\boldsymbol{x}, 1) = \frac{1}{2}$. We denote $\Pr_0$ the probability that h assigns to class 0 and $\Pr_1 = 1 - \Pr_0$ that h assigns to class 1.

Note that the objective is the weighted sum of local loss functions and the mixture weights are $[p_1, p_2] = [\frac{1}{2}, \frac{1}{2}]$ in the uniform framework. Then, the expected loss of $\widehat{h}_{\text{uf}}$ is

$$\begin{aligned}
\mathcal{L}(\widehat{h}_{\text{uf}}; \boldsymbol{w}, \boldsymbol{p}) =& \mathbb{E}_{(\boldsymbol{x},y)} \left[-\log \Pr_y \right] = \frac{1}{4} \log \frac{1}{\Pr_0} + \frac{3}{4} \log \frac{1}{\Pr_1} \\
=& \frac{1}{4} \log 4 + \frac{3}{4} \log \frac{4}{3} + \mathrm{KL}\left([\frac{1}{4}, \frac{3}{4}] \,||\, [\Pr_0, \Pr_1] \right) \\
\geq& \frac{1}{4} \log 4 + \frac{3}{4} \log \frac{4}{3},
\end{aligned}$$

where $\mathrm{KL}(\cdot)$ denotes the Kullback-Leibler divergence. Furthermore, we set $\Pr_0 = \frac{1}{4}$ and $\Pr_1 = \frac{3}{4}$. Thus, the expected loss of uniform framework becomes

$$\mathcal{L}(\widehat{h}_{\text{uf}}; \boldsymbol{w}, \boldsymbol{p}) = \frac{1}{4} \log 4 + \frac{3}{4} \log \frac{4}{3} = \log \frac{4}{\sqrt[4]{27}}.$$

Under the same settings, the expected loss of our framework is

$$\begin{aligned}\mathcal{L}(\widehat{h}; \boldsymbol{w}, \boldsymbol{p}) &= \min_{\boldsymbol{w}\in\mathcal{H}} \min_{\boldsymbol{p}\in\mathcal{P}} \left\{ \log \frac{1}{\Pr_0}, \frac{1}{2}\log\frac{1}{\Pr_0} + \frac{1}{2}\log\frac{1}{\Pr_1} \right\} \\ &= \min\left\{ \log\frac{4}{3}, \frac{1}{2}\log 4 + \frac{1}{2}\log\frac{4}{3} \right\} \\ &= \log\frac{4}{3} \leq \log\frac{4}{\sqrt[4]{27}}.\end{aligned}$$

This completes the proof.

According to Theorem 2, our general framework for Non-IID distributed learning has lower expected loss than current uniform framework, which demonstrates that our framework has surpassed uniform framework theoretically.

4 Algorithm

Our general framework aims to optimize mixture weights $\boldsymbol{p}$ together with the global model $\boldsymbol{w}$, and we relax the constraints on $\boldsymbol{p}$, so we consider applying SGD to get the optimal mixture weights.

4.1 `DL-opt`: Distributed Learning with Optimal Mixture Weights

In Non-IID distributed learning, training samples are stored on K clients, where local distribution vary across clients because of personal properties. Here, we use classic distributed learning method to train $\boldsymbol{w}_k$ on the k-th client locally until it converges. Meanwhile, we define an additional constraint $\|\boldsymbol{w}\|^2 \leq \frac{r}{\lambda^2\kappa^2}$ on $\|\boldsymbol{w}\|$ in Section 3 to provide strong generalization guarantee for our general framework, which indicates that the norm of $\boldsymbol{w}$ can not be very large. To this end, we add $\|\boldsymbol{w}\|$ to the local objective as a regularization term. Thus, the local objective on the k-th client is formed as

$$\min_{\boldsymbol{w}_k\in\mathcal{H}} L_k(\mathcal{D}_k) = \frac{1}{n_k}\sum_{i=1}^{n_k} \ell(h_k(\boldsymbol{x}_{ik}), y_{ik}) + \gamma\|\boldsymbol{w}_k\|, \tag{6}$$

where γ is a tunable parameter and $h_k = \boldsymbol{w}_k^T\phi(\boldsymbol{x}_{ik})$ related to Section 2.

On the other hand, it is unwise to optimize $\boldsymbol{p}$ on client-side, because the properties of other clients can not be integrated on the k-th client, which may cause the global model to deviate from the global optima. In order to capture global information and improve the performance of the aggregated global model $\boldsymbol{w} = \sum_{k=1}^{K} p_k\boldsymbol{w}_k$, we optimize the mixture weights on the central server with a group of validation samples $\mathcal{D}_{\text{val}}$, where the validation samples $\mathcal{D}_{\text{val}}$ are randomly sampled from each client in a small proportion.

After local training, local models $\boldsymbol{w}_k$ are uploaded to the central server to aggregate the global model $\boldsymbol{w}$. Then, we use SGD to optimize $\boldsymbol{p}$ on the validation

samples $\mathcal{D}_{\text{val}}$. We first get K predictions $[h_1(\boldsymbol{x}^{\text{val}}), \cdots, h_K(\boldsymbol{x}^{\text{val}})]$ by K local models. Next, we combine the mixture weights with the prediction vector as $\sum_{k=1}^{K} p_k h_k(\boldsymbol{x}^{\text{val}})$. Note that $\mathcal{D}_{\text{val}}$ is relevant to the task, so we can use the same loss function as local objectives to construct the central objective, that is

$$\min_{\boldsymbol{p}\in\mathcal{P}} L_{\boldsymbol{p}}(\mathcal{D}_{\text{val}}) = \frac{1}{n_{\text{val}}} \sum_{j=1}^{n_{\text{val}}} \ell\left(\sum_{k=1}^{K} p_k h_k(\boldsymbol{x}_j^{\text{val}}), y_j^{\text{val}}\right), \tag{7}$$

where n_{val} is the sample size of $\mathcal{D}_{\text{val}}$.

The pseudo code of `DL-opt` is listed in Algorithm 1.

Algorithm 1 `DL-opt` (Distributed Learning with Optimal Mixture Weights)

Input: $\bigcup_{k=1}^{K} \mathcal{D}_k$ (local samples), $\mathcal{D}_{\text{val}}$ (validation samples), $\boldsymbol{w}^0 \in \mathcal{H}$ (model parameters), $\boldsymbol{p}^0 \in \mathcal{P}$ (mixture weights), $\mathcal{T}_l, \mathcal{T}_c$ (total iterations of local training and central training), $\eta_{\boldsymbol{w}}, \eta_{\boldsymbol{p}}$ (learning rates).
Output: $\boldsymbol{w}^{\text{global}}$.

Client-side local training

K clients download the initial model: $\boldsymbol{w}_k^0 \leftarrow \boldsymbol{w}^0$ $(k = 1, 2, \cdots, K)$
for $k = 1, 2, \cdots, K$ **do**
  for $t = 1, 2, \cdots, \mathcal{T}_l$ **do**
    $\boldsymbol{w}_k^t = \boldsymbol{w}_k^{t-1} - \eta_{\boldsymbol{w}} \nabla_{\boldsymbol{w}_k} L_k(\mathcal{D}_k)$
  end for
end for

Server-side central training and aggregating

K clients upload local models $\boldsymbol{w}_k^{\mathcal{T}_l}$ $(k = 1, 2, \cdots, K)$
for $t = 1, 2, \cdots, \mathcal{T}_c$ **do**
  $\boldsymbol{p}^t = \boldsymbol{p}^{t-1} - \eta_{\boldsymbol{p}} \nabla_{\boldsymbol{p}} L_{\boldsymbol{p}}(\mathcal{D}_{\text{val}})$
end for
$\boldsymbol{w}^{\text{global}} = \sum_{k=1}^{K} p_k^{\mathcal{T}_c} \boldsymbol{w}_k^{\mathcal{T}_l}$

4.2 `FedOMW`: Federated Learning Version of `DL-opt`

Federated learning is a new distributed learning paradigm preserving users' privacy, which is a rising star in recent years. In addition to the encryption and compression techniques, federated learning uses an alternating communication mechanism to train the global model (shown in `FedAvg` [19]). In this part, we extend `DL-opt` to federated learning and propose a novel Non-IID federated learning algorithm `FedOMW` (Federated Learning with Optimal Mixture Weights).

The local objective and central objective in `FedOMW` are the same as `DL-opt`, because both of them are induced from our general framework for Non-IID distributed learning. The main difference between `FedOMW` and `DL-opt` is that `FedOMW` executes client-side local training and server-side central training periodically instead of training local models to converge. Moreover, the validation

samples can not be sampled from clients for privacy issues. To this end, the feature vectors after feature mapping and encrypting can be uploaded to the central server in a small proportion. Alternatively, we can also train a generator locally on each client and use it to generator several samples on the central server to get $\mathcal{D}_{\text{val}}$. For some very common learning tasks such as sentiment analysis of comments and next-word prediction, the validation samples are easy to get without sharing or uploading from clients. For example, if we apply distributed learning to solve a flower recognition task, we can easily get many flower pictures from Internet, which are used to construct $\mathcal{D}_{\text{val}}$ after preprocessing, and the whole process of constructing $\mathcal{D}_{\text{val}}$ will not bring privacy issues. Thus, the strategy of introducing $\mathcal{D}_{\text{val}}$ is not difficult to implement in federated learning.

We list the pseudo code of `FedOMW` in Algorithm 2.

Algorithm 2 `FedOMW` (Federated Learning with Optimal Mixture Weights)

Input: $\bigcup_{k=1}^{K} \mathcal{D}_k$ (local samples), $\mathcal{D}_{\text{val}}$ (validation samples), $\boldsymbol{w}^0 \in \mathcal{H}$ (model parameters), $\boldsymbol{p}^0 \in \mathcal{P}$ (mixture weights), $\mathcal{T}$ (total communication rounds), $\mathcal{T}_l, \mathcal{T}_c$ (total iterations of local training and central training), $\eta_{\boldsymbol{w}}, \eta_{\boldsymbol{p}}$ (learning rates).
Output: $\boldsymbol{w}^{\text{global}}$.

Server-side central training and aggregating

K clients download the initial model: $\boldsymbol{w}_k^0 \leftarrow \boldsymbol{w}^0$ $(k = 1, 2, \cdots, K)$
for $\nu = 1, 2, \cdots, \mathcal{T}$ **do**
  $\boldsymbol{w}_k^{\mathcal{T}_l} \leftarrow$ Client-side local training $(k = 1, 2, \cdots, K)$
  for $t = 1, 2, \cdots, \mathcal{T}_c$ **do**
    $\boldsymbol{p}^t = \boldsymbol{p}^{t-1} - \eta_{\boldsymbol{p}} \nabla_{\boldsymbol{p}} L_{\boldsymbol{p}}(\mathcal{D}_{\text{val}})$
  end for
  $\boldsymbol{w}^{\nu} = \sum_{k=1}^{K} p_k^{\mathcal{T}_c} \boldsymbol{w}_k^{\mathcal{T}_l}$
end for
$\boldsymbol{w}^{\text{global}} = \boldsymbol{w}^{\mathcal{T}}$

Client-side local training

for $k = 1, 2, \cdots, K$ **do**
  for $t = 1, 2, \cdots, \mathcal{T}_l$ **do**
    $\boldsymbol{w}_k^t = \boldsymbol{w}_k^{t-1} - \eta_{\boldsymbol{w}} \nabla_{\boldsymbol{w}_k} L_k(\mathcal{D}_k)$
  end for
end for
K clients upload local models $\boldsymbol{w}_k^{\mathcal{T}}$ $(k = 1, 2, \cdots, K)$

5 Experiments

In this section, we evaluate all algorithms on various real-world datasets with Non-IID partitioning.

5.1 Experimental Setup

In the following experiments, we mainly focus on multi-classification task, so the input space and output space can be expressed as $\mathcal{X} \in \mathbb{R}^{d_x}$ and $\mathcal{Y} \in \mathbb{R}^C$, where d_x denotes the input dimension and C denotes the output dimension related to C classes. We use cross-entry as the loss function. As shown in Section 2, the model h is formed as $h(\boldsymbol{x}) = \boldsymbol{w}^T\phi(\boldsymbol{x})$. Here, we use random Fourier feature [21] as the feature mapping, that is $\phi(\boldsymbol{x}) = \frac{1}{\sqrt{D}}\cos(\Omega^T\boldsymbol{x} + \boldsymbol{b})$, where $\phi : \mathbb{R}^{d_x} \to \mathbb{R}^D, \Omega \in \mathbb{R}^{d_x \times D}$, $\boldsymbol{b} \in \mathbb{R}^D$. According to [21], the entries in matrix Ω obey Gaussian distribution with $\Omega \sim \mathcal{N}(0, 1/\sigma^2)$ and the elements in vector $\boldsymbol{b}$ are uniformly sampled from $[0, 2\pi]$. We set $D = 2000$ for the following datasets.

Real-world datasets. The real-world datasets come from LIBSVM Data[4], which provides both training and testing data publicly. To construct a Non-IID partitioning setup [15, 17], we first divide the original training datasets into training samples $\bigcup_{k=1}^K \mathcal{D}_k$ and validation samples $\mathcal{D}_{\text{val}}$ according to the ratio of 8 : 2. Then, we split the training samples across 50 clients using a Dirichlet distribution [27] $\text{Dir}_K(0.01)$ to get the local training samples $\mathcal{D}_k$ for each client, the original testing datasets are used to evaluate the performance of the global model. The statistical information of all the datasets are listed in Table 1.

All the experiments are conducted on a Linux server equipped with two NVIDIA GeForce 2080ti, and all the algorithms are implemented by Pytorch[5]. We tune all the hyperparameters by grid search and list the best results in Table (Appendix B.1 of the supplementary file).

Table 1. Statistical information of datasets.

Datasets	Training Size	Testing Size	Dimensions	Classes
usps	7291	2007	256	10
pendigits	7494	3498	16	10
satimage	4435	2000	36	6
letter	15000	5000	16	26
dna	2000	1186	180	3
mnist	60000	10000	28×28	10

5.2 Experiments of Distributed Learning

In this part, we compare `DL-opt` to distributed learning with uniform mixture weights (abbreviated as `DL-u`). To ensure the fairness of comparison, we tune local learning rate $\eta_{\boldsymbol{w}}$ for `DL-u` and apply the same value to `DL-opt`. We set the epoch of local training as 200 and the epoch of central training as 100. The initial mixture weights $\boldsymbol{p}^0$ in `DL-opt` is the same as `DL-u` ($p_k = n_k/n$).

[4] Available at https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/

[5] Codes are available at https://github.com/Bojian-Wei/Non-IID-Distributed-Learning-with-Optimal-Mixture-Weights

Table 2. Test accuracy (%) of distributed learning algorithms on Non-IID datasets.

Algorithms	Datasets					
	usps	*pendigits*	*satimage*	*letter*	*dna*	*mnist*
`DL-u`	91.53±0.25	96.13±0.18	80.05±0.25	57.28±0.50	91.03±0.31	95.56±0.24
`DL-opt`	**92.49±0.09**	**96.53±0.35**	**83.25±0.27**	**62.95±1.31**	**92.80±0.31**	**96.29±0.13**

We run each experiment with 3 random seeds and record the average and standard deviation in Table 2. In Table 2, we bold the result results and underline the results which are not significantly worse than the best one. As shown in Table 2, we observe that `DL-opt` is significantly better than `DL-u` with confidence level 95% and `DL-opt` generally outperforms `DL-u` with a clear margin (more than 5% on *letter*). This illustrates that our general framework is effective in dealing with Non-IID distributed learning, which is consistent with our theoretical findings.

5.3 Experiments of Federated Learning

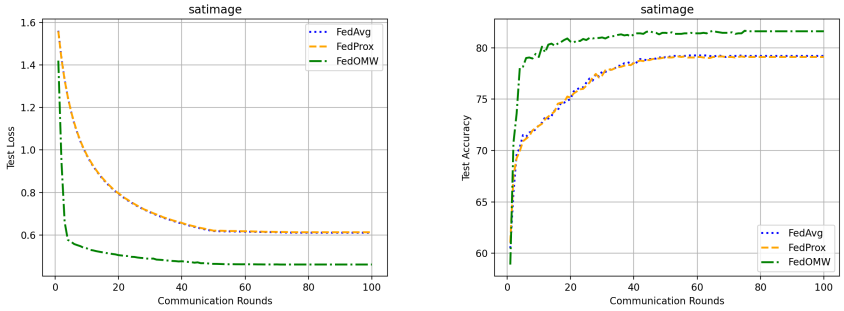
We also propose a federated learning algorithm `FedOMW` based on our framework. It is well know that `FedAvg` [19] and `FedProx` [16] are two mainstream algorithms in federated learning with uniform mixture weights. Thus, we conduct a comparative experiment to compare `FedOMW` with the two methods.

Table 3. Test accuracy (%) of federated learning algorithms on Non-IID datasets.

Algorithms	Datasets					
	usps	*pendigits*	*satimage*	*letter*	*dna*	*mnist*
`FedAvg`	90.82±0.26	95.45±0.14	79.52±0.19	51.17±0.64	90.30±0.01	95.15±0.22
`FedProx`	90.73±0.19	95.23±0.23	79.35±0.16	51.20±0.63	89.32±0.22	95.13±0.21
`FedOMW`	**92.81±0.02**	**96.92±0.04**	**81.97±0.25**	**63.57±0.46**	**90.98±0.59**	**96.29±0.05**

To ensure the fairness of comparison, we tune local learning rate $\eta_{\boldsymbol{w}}$ for `FedAvg` and apply the same value to `FedProx` and `FedOMW`. We set the epoch of local training as 2, the epoch of central training as 100 and the total communication round as 100. The initial mixture weights $\boldsymbol{p}^0$ in `FedOMW` remains $p_k = n_k/n$.

We also run each experiment with 3 random seeds and record the average and standard deviation in Table 3, and we also bold the result results and underline the results which are not significantly worse than the best one. According to Table 3, it is obvious that `FedOMW` performs significantly (confidence level 95%) better than `FedAvg` and `FedProx`, and `FedOMW` yields a marginal improvement up to 12% (on *letter*) compared to the other algorithms. Moreover, in Fig 1, we find that `FedOMW` not only performs better than the other algorithms, but also converges much faster. More experimental results can be found in Appendix B.2



(a) Test loss (b) Test accuracy

Fig. 1. Results of federated learning algorithms on Non-IID *satimage*.

of the supplementary file. Therefore, `FedOMW` is an effective algorithm to tackle the Non-IID problem in federated learning, and our general framework is proved to be well applied in classic distributed learning and federated learning scenarios.

5.4 Ablation Study

There are two important components in our framework: the optimization of mixture weights $\boldsymbol{p}$ and the regularization term of $\|\boldsymbol{w}\|$, where the optimization of $\boldsymbol{p}$ is the key strategy to improve the performance of distributed learning under Non-IID settings. In order to analyze the contribution of these two components to the proposed algorithms, we conduct an ablation experiment on both `DL-opt` and `FedOMW`. We report the results in Table 4 and Table 5, where `-p` denotes the algorithm only with the optimization of $\boldsymbol{p}$, `-w` denotes the algorithm only with the regularization term of $\|\boldsymbol{w}\|$ and `-non` denotes the algorithm without the two components.

Table 4. Ablation Results of `DL-opt` on Non-IID datasets.

Algorithms	**Datasets**					
	usps	*pendigits*	*satimage*	*letter*	*dna*	*mnist*
`DL-opt-non`	91.61	94.51	80.25	57.02	90.13	95.68
`DL-opt-w`	91.69	94.80	80.45	57.04	90.14	95.70
`DL-opt-p`	92.42	95.94	82.80	63.28	93.41	96.35
`DL-opt`	**92.58**	**96.28**	**83.20**	**63.42**	**93.59**	**96.40**

As shown in Table 4 and Table 5, we find that the performance of `DL-u` (equal to `DL-opt-non`) and `FedAvg` (equal to `FedOMW-non`) can be improved markedly by only optimizing mixture weights $\boldsymbol{p}$, which indicates the effectiveness of correcting

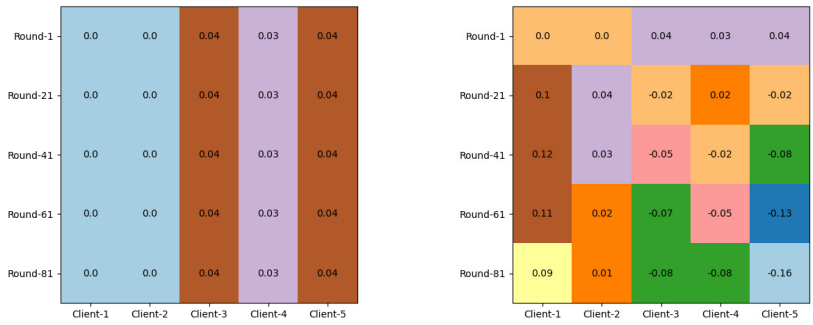
Table 5. Ablation Results of `FedOMW` on Non-IID datasets.

Algorithms	Datasets					
	usps	*pendigits*	*satimage*	*letter*	*dna*	*mnist*
`FedOMW-non`	90.63	95.51	79.75	49.38	89.54	95.37
`FedOMW-w`	90.68	95.71	79.65	49.90	89.66	95.39
`FedOMW-p`	92.36	96.77	82.25	61.94	90.39	96.34
`FedOMW`	**92.48**	**96.83**	**82.35**	**64.66**	**90.56**	**96.39**

local models' contributions before getting the global model in our framework. Moreover, the performance can be further improved by constraining $\|\boldsymbol{w}\|$, which coincides with our generalization theory.

5.5 Experiments of Mixture Weights

We visualize the mixture weights of 5 clients via central training in Fig 2. `DL-u` uses fixed uniform mixture weights, so $\boldsymbol{p}$ won't change during training. `DL-opt` optimizes the mixture weights on $\mathcal{D}_{\text{val}}$ through central training and the target is to minimize the classification loss. Thus, `DL-opt` adaptively assigns bigger mixture weights to the local model with smaller classification loss on $\mathcal{D}_{\text{val}}$. Combined with the above experiments, we can conclude that our min-min framework improves the performance of Non-IID distributed learning by selecting the optimal mixture weights.



(a) `DL-u` (b) `DL-opt`

Fig. 2. Mixture weights via central training on Non-IID *letter*.

6 Conclusion

In this paper, we present a general framework for Non-IID distributed learning, which optimizes the mixture weights together with model parameters to obtain

the optimal combination of local models. Compared to the classic distributed learning with uniform mixture weights, we demonstrate that our framework has lower expected loss theoretically. Furthermore, we provide a strong generalization guarantee for our framework based on local Rademacher complexity, where the excess risk bound can converge at $\mathcal{O}(1/n)$. Driven by our framework and theory, we propose an improved algorithm for Non-IID distributed learning and extend it to federated learning, where both of them perform significantly better than the current methods. The proof techniques in this paper may pave a way for studying generalization properties in other learning scenarios. Furthermore, we will study optimization errors and convergence guarantees in the future work.

Acknowledgements This work was supported in part by the Excellent Talents Program of Institute of Information Engineering, CAS, the Special Research Assistant Project of CAS, the Beijing Outstanding Young Scientist Program (No. BJJWZYJH012019100020098), Beijing Natural Science Foundation (No. 4222029), and National Natural Science Foundation of China (No. 62076234, No. 62106257).

References

1. Acharya, J., Sa, C.D., Foster, D.J., Sridharan, K.: Distributed learning with sub-linear communication. In: ICML 2019. vol. 97, pp. 40–50 (2019)
2. Arjevani, Y., Shamir, O.: Communication complexity of distributed convex learning and optimization. In: NIPS 2015. pp. 1756–1764 (2015)
3. Aviv, R.Z., Hakimi, I., Schuster, A., Levy, K.Y.: Asynchronous distributed learning : Adapting to gradient delays without prior knowledge. In: ICML 2021. vol. 139, pp. 436–445 (2021)
4. Bartlett, P.L., Boucheron, S., Lugosi, G.: Model selection and error estimation. Machine Learning **48**, 85–113 (2002)
5. Bartlett, P.L., Bousquet, O., Mendelson, S.: Local Rademacher complexities. The Annals of Statistics **33**(4), 1497–1537 (2005)
6. Bartlett, P.L., Mendelson, S.: Rademacher and gaussian complexities: Risk bounds and structural results. Journal of Machine Learning Research **3**(Nov), 463–482 (2002)
7. Bottou, L., Bousquet, O.: The tradeoffs of large scale learning. In: Advances in Neural Information Processing Systems 21 (NIPS). pp. 161–168 (2008)
8. Bousquet, O., Elisseeff, A.: Stability and generalization. J. Mach. Learn. Res. **2**, 499–526 (2002)
9. Bousquet, O., Klochkov, Y., Zhivotovskiy, N.: Sharper bounds for uniformly stable algorithms. In: COLT. pp. 610–626 (2020)
10. Cortes, C., Kloft, M., Mohri, M.: Learning kernels using local rademacher complexity. In: Advances in Neural Information Processing Systems 26 (NIPS). pp. 2760–2768 (2013)
11. Karimireddy, S.P., Kale, S., Mohri, M., Reddi, S.J., Stich, S.U., Suresh, A.T.: SCAFFOLD: stochastic controlled averaging for federated learning. In: ICML 2020. vol. 119, pp. 5132–5143 (2020)
12. Koltchinskii, V., Panchenko, D.: Empirical margin distributions and bounding the generalization error of combined classifiers. Annals of Statistics pp. 1–50 (2002)

13. Kutin, S., Niyogi, P.: Almost-everywhere algorithmic stability and generalization error. In: Proceedings of the 18th Conference in Uncertainty in Artificial Intelligence (UAI). pp. 275–282 (2002)
14. Lange, T., Braun, M.L., Roth, V., Buhmann, J.M.: Stability-based model selection. In: Advances in Neural Information Processing Systems 15 (NIPS). pp. 617–624 (2002)
15. Li, T., Sahu, A.K., Talwalkar, A., Smith, V.: Federated learning: Challenges, methods, and future directions. IEEE Signal Processing Magazine **37**(3), 50–60 (2020)
16. Li, T., Sahu, A.K., Zaheer, M., Sanjabi, M., Talwalkar, A., Smith, V.: Federated optimization in heterogeneous networks. In: MLSys (2020)
17. Lin, S.B., Wang, D., Zhou, D.X.: Distributed kernel ridge regression with communications. Journal of Machine Learning Research **21**(93), 1–38 (2020)
18. Liu, Y., Liu, J., Wang, S.: Effective distributed learning with random features: Improved bounds and algorithms. In: ICLR 2021 (2021)
19. McMahan, B., Moore, E., Ramage, D., Hampson, S., y Arcas, B.A.: Communication-efficient learning of deep networks from decentralized data. In: AISTATS 2017. vol. 54, pp. 1273–1282 (2017)
20. Oneto, L., Ghio, A., Ridella, S., Anguita, D.: Local rademacher complexity: Sharper risk bounds with and without unlabeled samples. Neural Networks **65**, 115–125 (2015)
21. Rahimi, A., Recht, B.: Random features for large-scale kernel machines. In: NIPS 2007. pp. 1177–1184 (2007)
22. Reddi, S.J., Charles, Z., Zaheer, M., Garrett, Z., Rush, K., Konečný, J., Kumar, S., McMahan, H.B.: Adaptive federated optimization. In: ICLR 2021 (2021)
23. Richards, D., Rebeschini, P., Rosasco, L.: Decentralised learning with random features and distributed gradient descent. In: ICML 2020. vol. 119, pp. 8105–8115 (2020)
24. Sharif-Nassab, A., Salehkaleybar, S., Golestani, S.J.: Order optimal one-shot distributed learning. In: NeurIPS 2019. pp. 2165–2174 (2019)
25. Vapnik, V.: The nature of statistical learning theory. Springer Verlag (2000)
26. Wang, C., Cheng, M., Hu, X., Huang, J.: Easyasr: A distributed machine learning platform for end-to-end automatic speech recognition. In: AAAI 2021. pp. 16111–16113 (2021)
27. Wang, J., Liu, Q., Liang, H., Joshi, G., Poor, H.V.: Tackling the objective inconsistency problem in heterogeneous federated optimization. In: NeurIPS 2020 (2020)
28. Wei, B., Li, J., Liu, Y., Wang, W.: Federated learning for non-iid data: From theory to algorithm. In: PRICAI 2021. vol. 13031, pp. 33–48 (2021)
29. Woodworth, B.E., Patel, K.K., Srebro, N.: Minibatch vs local SGD for heterogeneous distributed learning. In: NeurIPS 2020 (2020)
30. Yu, C., Tang, H., Renggli, C., Kassing, S., Singla, A., Alistarh, D., Zhang, C., Liu, J.: Distributed learning over unreliable networks. In: ICML 2019. vol. 97, pp. 7202–7212 (2019)
31. Zhu, R., Yang, S., Pfadler, A., Qian, Z., Zhou, J.: Learning efficient parameter server synchronization policies for distributed SGD. In: ICLR 2020 (2020)