Two-Dimensional Learning Rate Decay: Towards Accurate Federated Learning with Non-IID Data

Kaiwei Mo
Department of Computer Science
City University of Hong Kong
Hong Kong, China
kaimo6-c@my.cityu.edu.hk

Chen Chen
Theory Lab
Huawei HK
Hong Kong, China
chen.chen8@huawei.com

Jiamin Li
Department of Computer Science
City University of Hong Kong
Hong Kong, China
jiamin.li@my.cityu.edu.hk

Hong Xu
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Hong Kong, China
henryxu@cse.cuhk.edu.hk

Chun Jason Xue
Department of Computer Science
City University of Hong Kong
Hong Kong, China
jasonxue@cityu.edu.hk

***Abstract*—In federated learning a global model is trained with training data geographically distributed over a number of clients. To reduce the communication cost over the expensive wide area network, clients complete multiple local iterations before synchronization. However, since the training data are non-iid, such infrequent synchronization would compromise the accuracy after model convergence.**

In order to tackle this problem, we propose Two-Dimensional Learning Rate Decay (2D-LRD) in this paper, which aims to improve the model performance by adaptively tuning the learning rate on two dimensions: round-dimension and iteration-dimension during the model training. That is, we gradually decrease the learning rate and decrease the learning rates of local iterations in a synchronization round with different speeds. Based on our experiments and analysis, we find that the sum of the inner product of round updates is a valuable signal for learning rate tuning. We perform evaluation and demonstrate that 2D-LRD can make great progress compared to the baseline scheme.

I. Introduction

Federated learning (FL) [1], [2] is a new learning paradigm where geographically distributed clients can join the training process without sharing their own private data. In the FL setting there are two main constraints: data privacy, limited resources. Because data is generated locally and can not be exchanged, the local data distribution on clients are different from others, that is, in FL clients' local datasets may not be identically and independently distributed (i.e., non-IID). Meanwhile, the network and compute resources on clients are limited.

To protect the data privacy and save the communication cost, in 2016, McMahan et al. proposed the FedAvg algorithm [2]. In the FedAvg method, clients do the local computation for multiple iterations in a synchronization round. Nowadays, the FedAvg is the most common method that is usually regarded as the baseline method in FL in the literature.

In an attempt to handle the accuracy loss incurred by non-IID data, some works focus on local or public datasets sharing [3]–[5], and some explore the hyperparameters like synchronization frequency [6], [7]. However, they violate the data privacy constraints and their practicality is weak. Meanwhile, in [8] the authors theoretically analyze that a diminishing learning rate can improve model performance but without providing a practical method to decrease the learning rate. Therefore, we aim to find a method that is practical and does not violate the constraints to tackle this problem.

In this work, we aim to decrease the learning rate during the training period so as to reduce the accuracy loss, which is a new approach in FL. The key insight we decrease learning rate is that: a parameter's local optima are varied across different devices with non-IID distribution. Moreover, clients refine model parameters for multiple iterations before synchronization. For the above two reasons, the average of updates might be inaccurate for global optimization, which compromises the model accuracy. To eliminate the error and improve the model performance, we can decrease the learning rates of iterations which make an adverse effect for optimization. Meanwhile, the accumulated error amplifies for the later iterations in a synchronization round (Sec. II-B), thus we should decrease learning rate with different speed in a round.

Based on theoretical and empirical analysis, we propose 2D-LRD, a federated optimization algorithm that addresses the accuracy loss incurred by non-IID data. Our proposed algorithm consists two part: first is to find a signal for tuning the learning rate, second is to tune the learning rate. For the first part, Plug's procedure [9]–[11] forms the basis of our diagnostic for convergence.For the second part, to reduce the cumulative error, the learning rate is decreased by multiplied with a decay coefficient considering two dimension: synchronization round dimension and iteration dimension.

We implement 2D-LRD with PyTorch and evaluate its performance on twelve clients, and the outcome shows that 2D-LRD can improve the model performance without violating FL's constraints mentioned above. In our experiment, the

accuracy improvement ranges from 3.3% to 15.2% in our evaluation compared with FedAvg method.

II. Background and Motivation

A. Background for FL

Federated learning (FL) is a specialized ML technique which collaboratively learns a prediction model across decentralized devices with their local data [2]. With the complexity of computing tasks and the requirement of users' data privacy protection, FL becomes more and more important. In FL, clients compute parameters and gradients locally on private data. Then, client gradients are periodically aggregated in the server side to update the global model. In our work, synchronization round is a round which contains multiple local iterations performed before one global synchronization. In FL the training usually adopts BSP (Bulk Synchronous Parallel) [2].

In each FL client, models are trained in the similar approach as conventional distributed ML. Model parameters are optimized with the local data by minizing the training error, which is represented by loss function $f(s, \omega)$ (1). In (1), $|D|$ denotes the number of samples in the dataset. The common optimization approach is mini-batch gradient descent where data are divided into batches and trained iteratively.

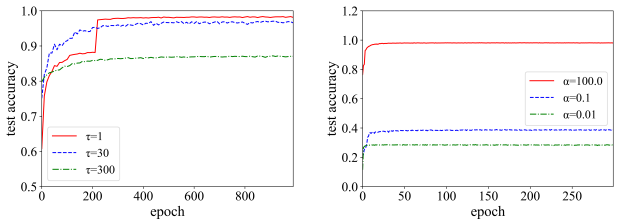
$$\omega^* = \arg\min L(\omega) = \arg\min \frac{1}{|D|} \sum_{s \in D} f(s, \omega). \quad (1)$$

Unlike typical distributed machine learning, FL has two distinct properties.

Decentralized clients. FL exploits the computation capacity of heterogeneous devices, e.g. mobile phones and IoT devices. The decentralized architecture makes communication cost extremely high [1], [2]. Therefore, parameter synchronization is far less frequent compared with conventional distributed machine learning. Without timely global parameter update, clients training with their local data are more likely to fall into premature stagnation, which downgrades the overall training efficiency. Figure 1(b) depicts the test accuracy when the synchronization frequency is adjusted from 1 to 30 and 300 iterations. When the frequency is 300, test accuracy is reduced by 15% compared with distributed training.

Non-IID training data. FL is designed to train on heterogeneous datasets. Specifically, training data on each FL client is not independent and identically distributed and cannot be exchanged with other clients. Training performance on each client could vary significantly if the local data is imbalanced in size and statistical distribution. Yet it is proved that data bias is deficient to training [3], [4]. Thus such non-iid property will serverly hinder global model from approaching optimal result.

Fig. 1 presents the impact to FL training performance of the two properties. In Fig. 1(a), when the synchronization frequency is reduced from 1 to 300. The test accuracy is downgraded by 15%. However, the communication overhead decreases on account of infrequent synchronization. Fig. 1(b) shows the test accuracy change when the data portion each clients owns varies. The local data of each clients are distributed in the Dirichlet process [12], [13]. A smaller α value means more imbalanced data distribution between the two clients. Compared with IID training, the test accuracy drops up to 0.26 when α is 0.01.



(a) test accuracy change when the synchronization frequency changes. (b) Test accuracy change when the non-iid level changes.

Fig. 1. The FL task trains the LeNet-5 model using MNIST dataset. There are two clients. The learning rate is 0.01 and batch size is 100.

TABLE I
THE DEFINITION OF VARIABLES IN ANALYSIS.

Variable	Definition
θ_j^{global}	Parameter values of the global model in the j_{th} iteration.
θ_j^i	Parameter values of the i_{th} client's model in the j_{th} iteration.
x_k	the input of the k_{th} sample in a batch.
y_k	the output of the k_{th} sample in a batch.
η	Learning rate.
B	the number of samples in a batch in FL system.
$l(\theta)$	the loss function of θ. In this analysis, we use the quadratic loss function ($l(\theta) = \frac{1}{2}(\theta^T x - y)^2$).
$l'(\theta)$	the gradient of $l(\theta)$ ($l'(\theta) = (\theta^T x - y)x$).

The non-negligible performance degradation proves that violating the aforementioned properties to achieve a better training accuracy is undesirable. We identify a new angle to address the loss of accuracy in FL scenario. Through both theoretical analysis and empirical experiments, we prove that the loss of accuracy can be alleviated without data sharing.

B. Accuracy Loss Analysis

We will start with theoretical analysis of the loss of accuracy. Notations are listed in Table I. To simplify the analysis, we consider a two-client FL scenario and compare it with standalone training. We adopts quadratic loss function in our analysis. The method of analysis is universal for all common loss functions.

Here, we assume that model initialization, hyperparameters are the same in both modes and SGD is applied. Moreover, in each iteration, the data batch in standalone ML training mode is the sum of the mini-batch data retrieved by the two clients in FL. We suppose that the number of iterations in a synchronization round is more than 2.

In the FL mode, the local update of two clients in the first iteration of a sychronization round can be denoted as (2).

$$\theta_2^1 = \theta_1^1 - \eta \cdot (\frac{\sum l'(\theta_1^1)}{B}), \theta_2^2 = \theta_1^2 - \eta \cdot (\frac{\sum l'(\theta_1^2)}{B}). \quad (2)$$

Similarly, for the second iteration, the local update is computed based on the first update as follows.

$$\theta_3^1 = \theta_2^1 - \eta \cdot (\frac{\sum l'(\theta_2^1)}{B}), \theta_3^2 = \theta_2^2 - \eta \cdot (\frac{\sum l'(\theta_2^2)}{B}). \quad (3)$$

In the standalone training mode, the data trained in the first iteration should be the aggregation of the local data of two FL clients. Therefore, the batch size is twice of the FL client's. The update of first and second iteration are as (4) and (5).

$$\theta_2^{global} = \theta_1^{global} - \eta \cdot (\frac{\sum l'(\theta_1^{global})}{2B}), \quad (4)$$

$$\theta_3^{global} = \theta_2^{global} - \eta \cdot (\frac{\sum l'(\theta_2^{global})}{2B}). \quad (5)$$

Since we assume model initialization are the same, the initial parameters are same in both modes, as shown in (6).

$$\theta_1^1 = \theta_1^2 = \theta_1^{global}. \quad (6)$$

Meanwhile, the data trained in the first iteration are the same in both modes. Given such condition, we can derive the relationship between local updates and global updates as follows.

$$\frac{\sum l'(\theta_1^{global})}{2B} \cdot 2 = \frac{\sum l'(\theta_1^1)}{B} + \frac{\sum l'(\theta_1^2)}{B}. \quad (7)$$

$$\frac{\theta_2^1 + \theta_2^2}{2} = \theta_2^{global}, \quad (8)$$

This relationship indicates that the updates in standalone mode is the average updates in FL mode. The model parameters aggregated after first iteration would not diverge from the true optima in standalone mode.

Though (8) is valid for the second iteration, the equality in (6) between client local update and standalone update not longer persists. Firstly, two FL clients owns different local data and solely perform local updates within a synchronization round. This leads to the first inequality. Meanwhile, standalone training always update with complete data batch while FL clients only have access to partial data. This, in turn, brings the second inequality.

$$\begin{aligned} l'(\theta_2^{global}) &= ((\theta_2^{global})^T x_i - y_i)x_i \\ &\neq ((\theta_2^k)^T x_i - y_i)x_i = l'(\theta_2^k), \end{aligned} \quad (9)$$

and k can be 1 or 2.We can derive that

$$\frac{\sum l'(\theta_2^{global})}{2B} \cdot 2 \neq \frac{\sum l'(\theta_2^1)}{B} + \frac{\sum l'(\theta_2^2)}{B}, \quad (10)$$

so

$$\theta_3^1 + \theta_3^2 \neq 2 \cdot \theta_3^{global}. \quad (11)$$

In the following iterations, FL clients' updates are further localized and the equality in (8) is not valid. Since FL clients do not have complete data batch, their updates are always suboptimal within the synchronization round compared with standalone training. Therefore, when clients train iteratively over suboptimal intermediate results, the ultimate parameters will diverge and increasingly accumulate training error.

We find that the aforementioned sub-optimal updates could be allevaited by adjusting the learning rate. When the data are not-IID, a smaller learning rate can reduce the impact of localized training. If learning rate is introduced to (11), it can be rewritten as follows.

$$\begin{aligned} &\theta_2^1 - \eta \cdot (\frac{\sum l'(\theta_2^1)}{B}) + \theta_2^2 - \eta \cdot (\frac{\sum l'(\theta_2^2)}{B}) \\ &\neq 2 \cdot (\theta_2^{global} - \eta \cdot (\frac{\sum l'(\theta_2^{global})}{2B})). \end{aligned} \quad (12)$$

In (12), when η approaches 0, the average value of FL updates is closer to the optimal result. The adjustment of learning rate could control the deviation caused by local data training. This provides a new angle in improving FL training to achieve the true optima.

Due to inconsistent landscape concavity of the loss functions on different clients, the model parameters aggregated after multiple local iterations would diverge from the true optimum that should be reached after the same number of iterations with timely (i.e., once-per-iteration) synchronization. To reduce the extent of such divergence and attain a better model performance, an intuitive idea is: speculatively reducing the step size in the local iterations expect the first one in a synchronization round, especially for the later ones where the accumulated error amplifies.

Then, we conduct a preliminary experiment to validate our idea mentioned in the last paragraph. Training model LeNet-5 (non-convex) and SVM (convex) [14] are evaluated. Lenet-5 is trained using CIFAR-10 dataset[1] with 10 classes in total. In each experiment, the learning rate of each client are initialized to the same value (both are 0.01). To clearly show the effectiveness of adjusting learning rate, training and testing are conducted simultaneously to evaluate the model performance. We compare our idea with the results in constant learning rate. Ours set the learning rate of local iteration except the first iteration to 0 starting from the 100th/1000th epoch. Figure 2 depicts comparison of the test accuracy curves and the value a randomly-selected parameter. In the SVM experiment, the accuracy is improved from 59% to 65% after tuning at epoch 100. In the LeNet-5 experiment, the accuracy is improved from 54% to 61% after tuning at epoch 1000. Both parameter values in two experiments change greatly, especially for the experiment applied with SVM, from 0.05 to about 0.8.

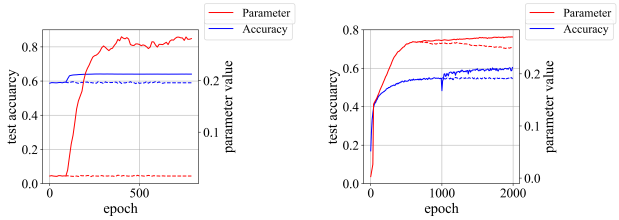
III. Adaptive Learning Rate Tuning

In this section, we discuss our proposed 2D-LRD algorithm by answering two primary questions, when to the tune learning rate and how to tune the learning rate.

A. Signal to Tune Learning Rate

Learning rate controls the speed at which a model learns the problem. The theoretical analysis in section II-B already shows that a smaller learning rate could effectively eliminate

[1]http://www.cs.toronto.edu/ kriz/cifar.html



(a) SVM, tuning at epoch 100. (b) LeNet-5, tuning at epoch 1000.

Fig. 2. We set up experiments by training models with 2 clients. It seems that the parameter values change greatly , and the models perform better. The dashed line show the measured variables without decreasing learning rate.

Algorithm 1: Diagnosis for Convergence (Server)

input : initial parameter value θ_0, average round updates g_1,g_2,..., observation window size o >0, local iteration ID k >0

output: decrease times d, global parameters θ_r

▷ initialization

$S, g_0, r, r_0, d \leftarrow 0$; ▷ r is synchronization round ID, and r_0 is the ID of the synchronization round when the hyperparameter d was last modified.

Procedure Aggregate(θ_k^1,θ_k^2...,θ_k^n)

  $r \leftarrow r + 1$;

  $\theta_r \leftarrow \frac{1}{n}\sum_{i=1}^{n} \theta_k^i$;

  $g_r \leftarrow \theta_r - \theta_{r-1}$;

  $S \leftarrow S + g_r g_{r-1}$;

  if $r > o + r_0$ *and* $S < 0$ **then**

    $d \leftarrow d + 1$; ▷ every time the model is detected in the stationary phase, decrease times plus one.

    $S \leftarrow 0$;

    $r_0 \leftarrow r$;

  end

  return θ_r, d

some of training loss caused by FL clients. However, a smaller learning rate will largely extend the training time as it takes more iterations to converge. Therefore, we adopt a diagnostic algorithm to determine the proper timing to adjust learning rate.

In the early training stage, training loss might oscillate and parameter values are likely to change a lot as the processed training data is small and highly variant, which is called transient phase. When the model starts to converge and stabilize, it transits to stationary phase. We observe that phase transition also presents in FL scenario. Yet, due to the non-IID data and low synchronization frequency, entering stationary phase also indicates client model stagnating in sub-optimal parameter values [15]. Decreasing learning rate during phase transition could avoid training error accumulating in the subsequent iterations beforehand.

We find Pflug's procedure could promptly identify which

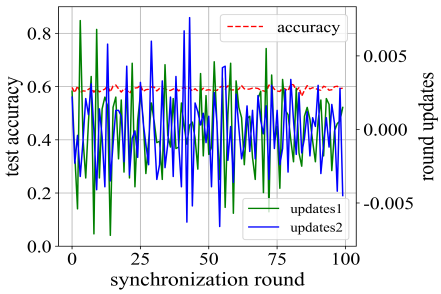


Fig. 3. Two parameter synchronization round gradients in SVM. When the model is in a stationary phase, the updates of adjacent rounds are mostly in the opposite direction.

phase clients enters [9]–[11]. The key idea of Pflug's procedure is to keep a running sum of the inner product of two consecutive synchronization round gradients $g_{n-1}g_n$. In the transient phase the gradients are roughly to the same direction, and thus their inner product is positive. In the stationary phase, the training process oscillates around a single point and the gradients point to different directions [15], which represents in Fig. 3, making the inner product negative. Therefore, when the summation is positive, the model is in the transient phase; when it becomes negative values, then it enters stationary phase. Pflug's procedure proves to be an effective indicator for phase transition in [11].

During each gradient aggregation, servers will actively compute the sum S of two consecutive gradient inner product. Simply considering a single S value could lead to frequent learning rate decreasing. An observation window is added to control the frequency of adjusting learning rate.

B. How to Tune Learning Rate

Upon stationary phase transition, we adjust learning rate using Two-Dimensional Learning Rate Decay (2D-LRD) algorithm.

Two-Dimensional Learning Rate Decay

We propose a Two-Dimensional Learning Rate Decay (2D-LRD) algorithm which considers both synchronization round dimension and iteration dimension. We introduce a decay coefficient as defined in (13), which incorporates both round-dimension decay α and iteration-dimension decay β. Learning rate will be reduced by the coefficient whenever a transition to stationary phase is detected.

$$decay\_coefficient = \alpha^{\beta}. \tag{13}$$

Decay in synchronization round-dimension decay α can reduce learning rate incrementally so that decay efficient can approach zero and completely eliminate training error. Iteration-dimension decay β is designed to adjust the learning rate of each iteration within the same sychronization round. As proven before, when the model trains for more iteration within a synchronization round in FL, the training error accumulates and the parameters gradually deviate from the optimal value, and the later iterations in a synchronization

Algorithm 2: Tune Learning Rate (Client)

input : synchronization frequency $\tau > 0$, initial parameter value θ_0, initial learning rate η_0, attenuation constant $C > 0$

$d, k \leftarrow 0$; ▷ Initialization. k is local iteration ID

▷ run Procedure LocalCompute once in each local iteration.

Procedure LocalCompute(k)

  $\alpha \leftarrow$ getRoundDecayRatio(d);

  $\beta \leftarrow k\%\tau$;

  $\eta \leftarrow \eta_0 \times \alpha^{\beta}$;

  Compute local update u_k^i;

  $\theta_k^i \leftarrow \theta_{k-1}^i + \eta \times u_k^i$; ▷ u_k^i:local update in k_{th} iteration on the i_{th} client.

  $k \leftarrow k + 1$;

  if $k\%\tau == 0$ **then**

    $\theta_r, d \leftarrow$ server_node.Aggregate(θ_k^i); ▷ synchronize global parameters θ_r and updates the d.

  end

Procedure getRoundDecayRatio(d)

  if $C * d < 1$ **then**

    $\alpha \leftarrow (1 - C \times d)$;

  else

    $\alpha, \tau \leftarrow 1$; ▷ when the hyperparameter d is large enough, turning to timely synchronization.

  end

  return α

round would generate a greater error. Therefore, β serves for exponentially reducing learning rate so that it could offset the accumulated training error. 2D-LRD enables learning rate adjustment controlled by both the training performance within a synchronization round and across different rounds.

Then our problem now becomes how to determine the value of the decay coefficient. When detecting phase transition, the number of phase transition d is updated and is used to compute the round-dimensional decay. The larger d is, the more decay learning rate should have. Meanwhile, an attenuation constant C is added to find control the decay rate. When $C \times d$ is smaller than 1, learning rate decay is set to $1 - C \times d$. Otherwise, a large product indicates there are frequent phase transition and sole local computation could not serve the need to find the true optimal results. Then, the synchronization frequency is τ is set to 1 so that model will be trained with timely synchronization. As for iteration-dimensional decay, β increases linearly with the number of iterations completes within a synchronization round, which conforms with our analysis in Sec. II. Algorithm 2 presents the details of 2D-LRD.

IV. Performance Evaluation

In this section, we deploy the proposed 2D-LRD algorithm in a 12-client FL cluster and evaluate its performance.

TABLE II
Combinations of Hyperparameters, Model, and Dataset.

Hyperparameter				
η_0	τ	o	Model	Dataset
0.01	10	500	LeNet-5	CIFAR-10
0.05	10	300	LSTM	KWS[2]
0.05	10	10	LSTM	ag_news[3]
0.01	10	50	CNN	EMnist[4]

A. Experiment Setup

Dataset, model and hyperparameters. We conduct experiments on different training models and datasets to prove the generality of 2D-LRD. Similar to [3], here training data are evenly partitioned so that the data are severely non-IID. Meanwhile, the hyperparameters (initial learning rate η_0, synchronization frequency τ, observation window size o) are initialized to different values to prove the robustness of the algorithm. Table II shows details of each experiment setup.

In addition, the default batch size and attenuation constant C in each case are set to 100 and 0.2 respectively. The number of samples selected in each epoch is the same as the total number of samples in the training set.

Scheme compared. We compare 2D-LRD with the baseline approach which applies FedAvg algorithm and then compare with two other schemes AD and AT to highlight 2D-LRD's benefit.

1) *FedAvg* FedAvg is a common basedline algorithm in FL [2]. Clients conduct local computation for several iterations and then synchronize with the servers. No learning rate adjustment is conducted during training process.
2) *All-Decreasing (AD)* Every time the model is entering stationary phase, *all the learning rates* of local iterations will be *multiplied* by a decay coefficient. The decay coefficient is a constant. Here we set the decay coefficient as 0.8 in the evaluation.
3) *Additive-Tuning (AT)* Every time the model is entering stationary phase, the learning rate starting from the second iteration will be decreased by a decay value same as the round-dimensional decay in 2D-LRD.

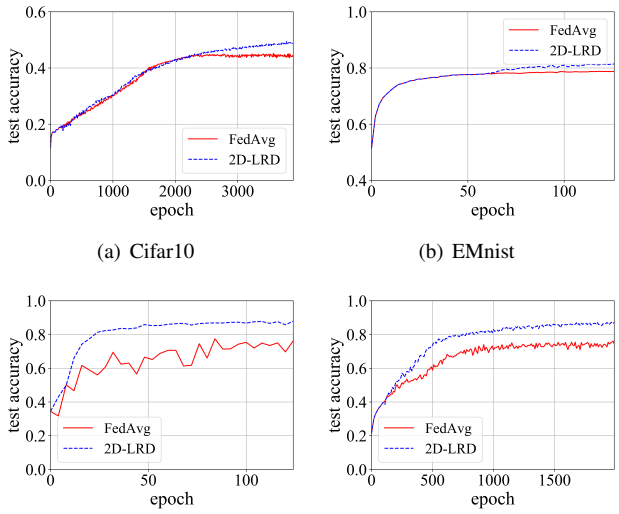
B. Visual Behavior

We first evaluate 2D-LRD with FedAvg which does not adjust learning rate. Figure 4 show the test accuracy performance on the aforementioned four experiment setup. In the figure, we can see that our 2D-LRD outperforms the FedAvg algorithm obviously in all experiments. This is because we eliminate the error by decreasing the learning rate. In particular, we further make a quantitative performance comparison between

[2]KeyWord Spotting dataset (KWS), a subset of the Speech Commands dataset [16]

[3]Training set contains 4 types of news and each kind of news contains 30,000 pieces, and test set contains 12,000 pieces of news [17]

[4]an extension of Mnist dataset, which contains 814,255 characters. 62 unbalanced classes [18]



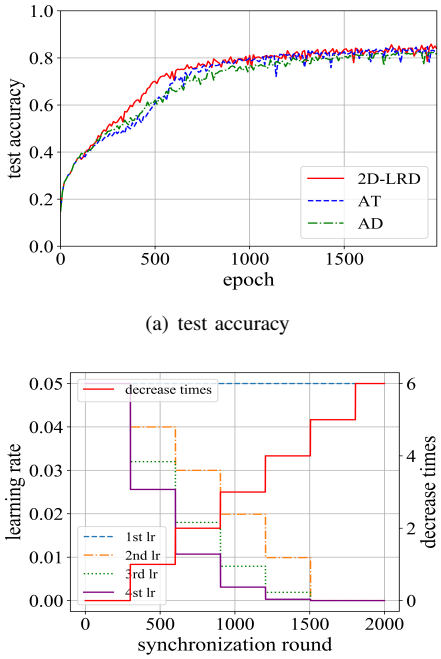
(a) Cifar10 (b) EMnist

(c) ag_news (d) KWS

Fig. 4. Test accuracy curves under 2D-LRD and FedAvg.

TABLE III
HIGHEST ACCURACY (%) AFTER CONVERGENCE.

Scheme	Dataset			
	EMnist	Cifar10	KWS	ag_news
FedAvg	78.8	45.3	76.1	76.3
2D-LRD	81.4	49.4	87.4	87.9



(a) test accuracy

(b) learning rate and decrease times (d)

Fig. 5. Test accuracy under 2D-LRD, AT and AD with the KWS dataset, and the first four learning rates in a synchronization round under 2D-LRD.

2D-LRD and FedAvg in Table III, which lists the highest accuracy attained when the training model has converged using the FedAvg algorithm. Because we set different datasets in different experiments, we use the datasets' names to represent combinations in Table III. In the evaluation, when the accuracy is not improved within 50 epochs, we judge the convergence of the model. According to Table III, accuracy improvement ranges from 3.3% to 15.2% in the evaluation, which show the importance of avoiding premature stagnation.

Then, we compare 2D-LRD's performance with AD and AT to further highlight the improvement of 2D decay design. Figure 5(a) depicts the test accuracy curves of different approach. AD converges more slowly than 2D-LRD and AT as it decreases learning rate of the first iteration within a round. Yet we prove that the first iteration will not produce any training error. Though AT skips the first iteration, it adopts additive decay. 2D-LRD leverages both the iteration-dimension and round-dimension decay to accelerate the decay rate.

Figure 5(b) plots the sampled four learning rate changes across the synchronization rounds. Each line i in the Figure 5(b) represents the learning rate of the i^{th} iteration in each synchronization round. The learning rate of the first iteration (blue) remains the same throughout the training process, which conforms with our theoretical analysis. Others decrease to nearly half of its inital learning rate before the first 1000 rounds. The learning rate becomes 0 roughly from the 1500 round and the training synchronization frequency is adjusted to 1 simultaneously.

V. RELATED WORK

Federated learning allows multiple clients to collaboratively train a model without data sharing. Existing works on federated learning mainly focus on two improving its communication efficiency and tackle statistical heterogeneity. We classify these studies into two parts:

Communication efficiency. In the federated learning setting, devices have limited communication capabilities, thus improving communication efficiency is important. In [2], the Federated Averaging (FedAvg) is presented that allows clients to perform local training of multiple epochs, which further reduces the number of communication rounds by averaging model weights from the worker nodes. In [1], [19], the authors proposed structured and sketched updates to decrease the completion time of each communication round. Nishio et al. [20] proposed a resource-aware selection algorithm that maximizes the number of participating devices in each round.

Statistical heterogeneity. Non-IID data is annoying since it compromises model performance. In [3], some common samples are copied to all worker nodes, and there are also some approaches sharing some other data like local loss function values to decide whether to continue local training or server-side proxy data [4], [5]. Furthermore, in [4], the authors use GAN models to locally generate some auxiliary data. Unlike data sharing, some works propose to add a specific momentum to the optimizer [21] or an extra regularizing term to the loss function [22] so as to reduce the negative impact of non-IID training data. In addition, some works concentrate on synchronization frequency tuning [6], [7].

VI. Conclusion

In this paper, we propose 2D-LRD, a practical and privacy-preserving FL scheme to improve the model accuracy in non-iid scenario. Through theoretical analysis, we identify that model will diverge from the true optimum in the iterative training when only computation on local data is allowed. Different from the conventional solutions training data violating the privacy constraints, 2D-LRD introduces an effective learning rate tuning algorithm to dynamically adjust learning rate whenever training enters premature stagnation phase. It uses Pflug's procedure to determine the training phase. Then, learning rate decays incrementally in synchronization round dimension and exponentially in iteration dimension to accelerate the amendment of diverging parameters. Experimental results demonstrate that the proposed algorithm achieves an up-to-15.2% test accuracy improvement compared with the FedAvg.

References

[1] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[2] McMahan Brendan, Moore Eider, Ramage Daniel, Hampson Seth, and y Arcas Blaise Aguera. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282. PMLR, 2017.

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

[4] Eunjeong Jeong, Seungeun Oh, Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479*, 2018.

[5] Li Huang, Yifeng Yin, Zeng Fu, Shifa Zhang, Hao Deng, and Dianbo Liu. Loadaboost: Loss-based adaboost federated machine learning with reduced computational complexity on iid and non-iid intensive care data. *Plos one*, 15(4):e0230706, 2020.

[6] Jianyu Wang and Gauri Joshi. Adaptive communication strategies to achieve the best error-runtime trade-off in local-update sgd. *arXiv preprint arXiv:1810.08313*, 2018.

[7] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K Leung, Christian Makaya, Ting He, and Kevin Chan. Adaptive federated learning in resource constrained edge computing systems. *IEEE Journal on Selected Areas in Communications*, 37(6):1205–1221, 2019.

[8] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189*, 2019.

[9] Georg Ch Pflug. Non-asymptotic confidence bounds for stochastic approximation algorithms with constant step size. *Monatshefte für Mathematik*, 110(3-4):297–314, 1990.

[10] George Yin. Stopping times for stochastic approximation. In *Modern Optimal Control: A Conference in Honor of Solomon Lefschetz and Joseph P. LaSalle*, pages 409–420, 1989.

[11] Jerry Chee and Panos Toulis. Convergence diagnostics for stochastic gradient descent with constant learning rate. In *International Conference on Artificial Intelligence and Statistics*, pages 1476–1485, 2018.

[12] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. Measuring the effects of non-identical data distribution for federated visual classification. *arXiv preprint arXiv:1909.06335*, 2019.

[13] Mikhail Yurochkin, Mayank Agarwal, Soumya Ghosh, Kristjan Greenewald, Trong Nghia Hoang, and Yasaman Khazaeni. Bayesian nonparametric federated learning of neural networks. *arXiv preprint arXiv:1905.12022*, 2019.

[14] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[15] Noboru Murata. A statistical study of on-line learning. *Online Learning and Neural Networks. Cambridge University Press, Cambridge, UK*, pages 63–92, 1998.

[16] Pete Warden. Speech commands: A dataset for limited-vocabulary speech recognition. *arXiv preprint arXiv:1804.03209*, 2018.

[17] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657, 2015.

[18] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

[19] Jakub Konečnỳ, Brendan McMahan, and Daniel Ramage. Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*, 2015.

[20] Takayuki Nishio and Ryo Yonetani. Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, pages 1–7. IEEE, 2019.

[21] Chengjie Li, Ruixuan Li, Haozhao Wang, Yuhua Li, Pan Zhou, Song Guo, and Keqin Li. Gradient scheduling with global momentum for non-iid data distributed asynchronous training. *arXiv preprint arXiv:1902.07848*, 2019.

[22] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.06127*, 2018.