Federated Learning Optimization Algorithm Based on Dynamic Client Scale

Luya Wang, Wenliang Feng, and Ruoheng Luo(✉)

Shenzhen University, Shenzhen, China
{wangluya2021,fengwenliang}@email.szu.edu.cn, neal@szu.edu.cn

Abstract. Federated learning methods typically learn models from the local iterative updates of a large number of clients. The interest in the impact of client quantity on the training dynamics of federated learning algorithms has been growing in recent years. Increasing the client scale during the training process not only improves data parallelism efficiency but also accelerates the training of federated learning. When optimizing models using a large number of clients, the learning rate needs to adapt to the client scale and the aggregation of updates in order to maximize speed while maintaining model quality. However, the current approach mainly relies on empirically-derived linear learning rate scaling rules, which cannot adapt to the dynamic client scale in federated learning. In this regard, we propose ASNES, an algorithm that dynamically adapts to the client scale in federated learning. By continuously adapting to the client quantity and aggregation of updates, ASNES achieves acceleration for different client scales. In experimental evaluations, ASNES demonstrates favorable performance compared to other benchmark algorithms.

Keywords: Adaptive optimization · Federated learning · Large batch training

1 Introduction

With the proliferation and popularization of smartphones, drones, and Internet-of-Things devices, the amount of data generated at the network edge is undergoing explosive growth [12]. Traditional centralized learning faces insurmountable challenges in terms of communication, computation, and storage, making it unable to meet the requirements of edge computing. To facilitate large-scale machine learning in edge computing environment, a new paradigm named federated learning [7,11] has proposed to allow distributed devices collaboratively train model without exposing their raw data. In federated learning, the participating devices do not send their local data to a central server for centralized training. Instead, the device perform multiple local updates using stochastic gradient descent and then send their local model update results to the server.

This work was supported by the National Nature Science Foundation (NSF) of China: Grant 62102266.

V. C. M. Leung et al. (Eds.): Qshine 2023, LNICST 574, pp. 109–117, 2024.
https://doi.org/10.1007/978-3-031-65123-6_8

In federated learning, there are a large number of client nodes participating in model training, resulting in a significant amount of data being processed during each iteration. To enhance the efficiency of data parallel training, there has been increasing attention on combining large-batch optimization with federated learning [1,2,15]. These efforts involve adapting the batch size during the training process to reduce the training time in federated learning. Furthermore, [8,17,18] accelerates the model training speed in scenarios with large batch sizes by adjusting the learning rate. While these efforts have demonstrated the feasibility of adjusting batch sizes to reduce training time in federated learning, they have not explored the impact of large-batch training on model convergence when the client scale dynamically changes in federated learning.

Although some research exploring large scale federated learning, there are still some issues that need to be addressed in order to further improve its performance. In this paper, we propose a algorithm for federated learning, called Adaptive Scale Nesterov (ASNES) algorithm. ASNES make large-batch training significantly more userfriendly in federated learning. Without changing the learning rate, ASNES can adapt to vastly different scale with large speed-up and near-identical model quality. Experimental results are presented on the Fashione-MNIST and CIFAR-10 dataset to demonstrate the effectiveness of proposed ASNES algorithm.

2 Related Work

As the demand of locally data storing and training models at edge devices, Federated Learning rapidly attracts growing interest in recent years. One prominent characteristic of federated learning is that only a subset of clients participate in training during each iteration due to system limitations. In the FedAvg [11] algorithm, during each communication round, the server broadcasts the global model to the currently available clients. These clients then use their local datasets to update the model and send their model updates back to the server. With a larger number of clients, there is a greater likelihood of diverse data samples being included in the training process, which can help reduce data bias and improve convergence [5,13,16].

When we increase the scale of clients after a certain point, the convergence accuracy of algorithm becomes significantly lower than the baseline. [6] concluded that there is a generalization problem for large-batch training. It means that even the large batch can get a low training loss, the test loss will be still significant higher than the training loss. For small batch, the training loss and test loss are close to each other. [8] empirically found that simply scaling the learning rate linearly with respect to batch size works better up to certain batch sizes. To avoid optimization instability due to linear scaling of learning rate, [3] proposed a highly hand-tuned learning rate which involves a warm-up strategy that gradually increases the learning rate to a larger value and then switching to the regular learning rate policy (e.g. exponential or polynomial decay). However, empirical study [9,14] shows that learning rate scaling heuristics with batch size do not hold across all problems or across all batch sizes.

More recently, to reduce hand-tuning of hyperparameters, adaptive learning rates for large batch training garnered significant interests. Several recent works successfully scaled the batch size to large values using adaptive learning rate without degrading the performance, thereby, finishing ResNet-50 training on ImageNet in few minutes. [18,19] adjust learning rate based on the ratio of the parameter value and the gradients computed at each iteration. [10,17] consider the gradient variance for learning rate scaling in large-batch training. The gradient variance is a crucial factor to consider for reliably scaling large-batch training. AdaScale [17], a state-of-the-art learning rate scaling method, estimates the state of the traninig by using gradient variance and adaptively adjust learning rate depending on the variance of gradients computed by clients at each iteration. For instance, it increases learning rate linearly in case of high gradient variance, but rarely increases learning rate in case of low gradient variance. In this way, AdaScale successfully achieved the higher accuracies that those of the linear scaling method in various machine leaning tasks.

3 Preliminaries

We consider a federated learning system that consists of N clients and a single server. client n has a local dataset $\mathcal{D}_n = \{x, y\}$ with D_n data samples, where x and y denotes the data feature and the corresponding label, respectively. Therefore, the total dataset in the system is $\mathcal{D}$, which has $D = \sum_{n=1}^{N} D_n$ data samples. The server and clients coordinately minimize the empirical risk as

$$\min_{w \in \mathbb{R}^d} f(w) := \frac{1}{N} \sum_{n=1}^{N} f_n(w) \tag{1}$$

where $f_n(w) = \mathbb{E}_{(x_i, y_i) \sim \mathcal{D}_n}[f_i(w; x_i, y_i)]$ is the loss function of the nth client. The functions $f_n(w)$ may be nonconvex.

A common approach to solving (1) in federated settings is FedSGD [7,11]. In the t-th round of FedSGD, the server randomly selects a subset $\mathcal{S}$ from all clients and broadcasts its global model w_t to each selected client. These clients independently update the model based on their local data. Typically, clients utilize their local loss function to perform multiple rounds of stochastic gradient descent (SGD). Let the updated local models be $w_{n,1}, w_{n,2}, \cdots, w_{n,k}$, so the update of client n can be written as $\nabla_{n,t} := w_{n,1} - w_{n,k}$, for $n \in \mathcal{S}_t$. In any case, each selected client then sends the update back to the server, where the global update is computed by aggregating all the updates of selected client:

$$w_{k+1} = w_k - \eta_t \nabla_t, \nabla_t := \frac{1}{|\mathcal{S}_t|} \sum_{n \in \mathcal{S}_t} \nabla_{n,t} \tag{2}$$

where η_t represents the learning rate of the server during the t-th iteration.

In order to improve the efficiency of data parallelism training when we have more computational clients, we need to increase the number of participating

clients in the training process. Increasing the number of computational clients while keeping the workload per client constant allows us to train models with a larger batch size, thereby improving training efficiency. [18] uses different learning rate for different layers based on the norm of the parameters $\|w\|$ and the norm of the gradients $\|\nabla\|$. [4] adjust the learning rate by tracking the variance $\mathbb{E}[\frac{1}{|S|}\sum_{n\in\mathcal{S}}\|\nabla_{n,t}\|^2]$ and $\mathbb{E}[\|\nabla_t\|^2]$ between the local updates of clients and the aggregated update on the server to adapt learning rate.

4 Algorithm

In this section, we will present out proposed ASNES algorithm. To accelerate the training speed of the global model, common methods include more accurate estimation of model update and more precise update using momentum. Our proposed algorithm automatically adjusts the learning rate based on the number of clients, enabling it to be applied to training with larger batch sizes.

By increasing the number of client nodes in federated learning, the variance of the stochastic gradient decreases. As a result, we can perform larger update operations, such as increasing the learning rate. This approach allows for more aggressive updates, potentially leading to faster convergence of the global model. In the ASNES algorithm, during the t-th iteration, the learning rate is adaptively adjusted based on the number of client $|\mathcal{S}_t|$ participating in training and the variance of gradients $\|\nabla_t\|$. By observing the changes in the number of client nodes and gradient variance, ASNES dynamically adjusts the learning rate to effectively update the global model. Specifically, when more client nodes are involved in training, ASNES increases the learning rate to facilitate faster convergence of information between nodes. Conversely, when the gradient variance is high, ASNES reduces the learning rate to avoid issues caused by overly large updates that may lead to divergence. Through adaptive learning rate adjustment, ASNES achieves a better balance between convergence speed and model stability in distributed training.

At the beginning of each iteration t, a subset of clients $\mathcal{S}_t$ are required to participate in the current training process. Each client calculates its local updates $\nabla_{n,t}$ and their variances $\|\nabla_{n,t}\|^2$, and then sends them to the server. This process allows the server to collect information from clients and make decisions about adjusting the global model. By sharing the local updates and variances, the server can analyze the distribution of updates across the client nodes and take appropriate actions, such as estimate the variance of local updates $\sigma_t^2 = \frac{1}{|\mathcal{S}_t|-1}\sum_{n\in\mathcal{S}_t}\|\nabla_{n,t}\|^2 - \frac{\mathcal{S}_t}{\mathcal{S}_t-1}\|\nabla_t\|^2$ and the variance of global update $\nu_t^2 = \|\nabla_t\|^2 - \frac{1}{\mathcal{S}_t}\sigma_t^2$. The adjustment of the learning rate can be summarized as follows:

$$\eta_t = \frac{\sigma_t^2 + \nu_t^2}{\frac{1}{\mathcal{S}_t}\sigma_t^2 + \nu_t^2}. \tag{3}$$

In addition to adjusting the learning rate, the server leverages momentum to accelerate the global model updates. Nesterov momentum is an optimization

technique commonly used in deep learning algorithms. It is an extension of the standard momentum method that helps accelerate the convergence of gradient-based optimization methods. In Nesterov momentum, the update of global model parameters is adjusted based on the current momentum direction. Unlike simple momentum, Nesterov momentum calculates a lookahead gradient that takes a step forward in the direction of the accumulated momentum and then evaluates the gradient at that position. This allows the algorithm to anticipate the future direction of the gradient and make more intelligent updates to the model. Below are the steps to describe how the server utilizes Nesterov momentum for updates in federated learning:

$$u_{k+1} = \beta u_k + \nabla_t \tag{4a}$$

$$v_k = \beta u_{k+1} + \nabla_t \tag{4b}$$

$$w_{k+1} = w_k - \eta_t v_k \tag{4c}$$

Combining the Nesterov momentum with adaptive scale learning rate, we proposed ASNES algorithm is shown in Algorithm 1. By utilizing adaptive learning rate adjustment methods and Nesterov momentum in federated learning, the server can effectively update the global model parameters while considering both the client scale and the current momentum direction. This enables more intelligent and efficient model updates to be achieved.

Algorithm 1. ASNES Algorithm

Initialize: the global model w_0, the learning rate of server η_0, the learning rate of client η
for $t = 1, \cdots, T$ **do**
  Sample subset $\mathcal{S}_t$ of clients
  for each client $n \in \mathcal{S}_t$ in parallel **do**
    $w_{n,0} = w_t$
    for $k = 1, \cdots K$ **do**
      Sample a batch data $(x_{n,k}, y_{n,k})$
      Compute unbiased stochastic gradient: $g_{n,k} := \nabla_{w_{n,k}} f_n(w_{n,k}; x_{n,k}, y_{n,k})$
      $w_{n,k+1} = w_{n,k} - \eta g_{n,k}$
    end for
    $\nabla_{n,t} = w_{n,0} - w_{n,K}$
  end for
  $\nabla t := \frac{1}{|\mathcal{S}|} \sum_{n \in \mathcal{S}} \nabla_{n,t}$
  Server update the learning rate η_t via (3)
  Server update the global model w_t via (4)
end for

5 Experiments

5.1 Setup

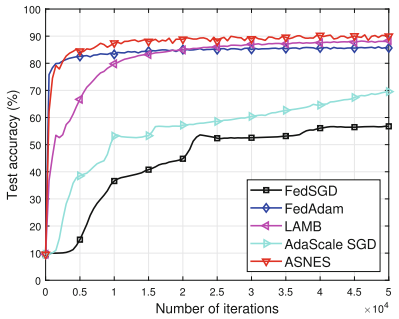
Dataset and Model. We test the experiments on Fashion-MNIST dataset and CIFAR-10 dataset. To simulate the data distribution of clients in federated learning, we tested two scenarios: independent and identically distributed (IID) and non-independent and identically distributed (non-IID). In the IID scenario, the dataset is shuffled and randomly divided among all clients. In the non-IID scenario, the dataset is partitioned and assigned to different clients based on specific categories, with each client holding only a subset of categories. For the Fashion-MNIST dataset, we trained a LeNet model and evaluated its convergence and performance through testing. For the CIFAR-10 dataset, we trained a ResNet-18 model and evaluated its convergence and performance through testing. In the federated learning training, there are a total of 100 clients participating in the global model training. To simulate the dynamic changes in the number of clients in federated learning, the server randomly selects between 10 to 90 clients for aggregation during each iteration. For non-IID data distribution, we set each client node to contain only three categories from the dataset.

Benchmarks. We compare several classical and efficient methods with the proposed FLO in our experiments, including FedSGD [11], FedAdam [13], LAMB [19], and AdaScale SGD [4]. The FedSGD algorithm performs multiple local updates on client and then aggregates them on the server to update the global model. This algorithm serves as a fundamental baseline for other federated learning algorithms. The FedAdam algorithm performs local updates and global update by using the Adam optimizer with adaptive learning rates simultaneously on both the client and server side. The LAMB algorithm introduces a trust ratio for the update of each layer, allowing the model to be trained with larger batch sizes. The Adascale SGD improves the performance of global updates by adjusting the learning rate based on the variance of gradients and the number of clients. Below, we will provide the details of the hyperparameter settings in the experiment. The learning rate for the Fashion-MNIST experiment is set to 0.00001, while for the CIFAR-10 experiment, the learning rate is set to 0.0001. For FedAdam and LAMB, we fix a momentum parameter $\beta_1 = 0.9$ and a second momentum parameter $\beta_2 = 0.999$. For ASNES algorithm, we set $\beta = 0.9$ in the Fashion-MNIST experiment and $beta = 0.85$ in the CIFAR-10 experiment.

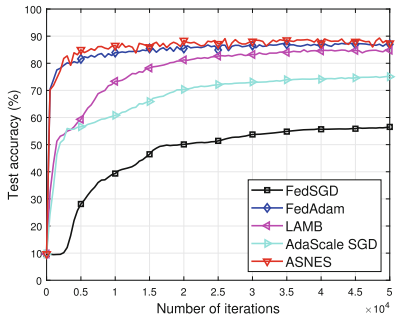
5.2 Evaluation Results

Figure 1 illustrates the performance comparison between ASNES and other algorithms on Fashion-MNIST and CIFAR-10 datasets with different distributions. We have observed that ASNES performs excellently on both homogeneous and heterogeneous data distributions. In the Fashion-MNIST dataset, ASNES, FedAdam, and LAMB achieve similar performance in both homogeneous and heterogeneous data distributions, with ASNES showing a slight improvement in

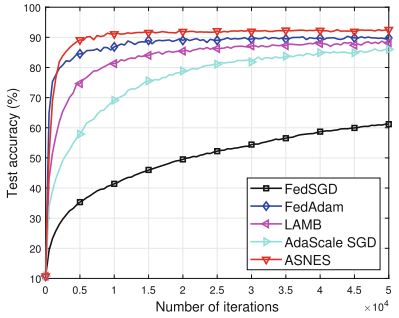
convergence speed. On the more complex CIFAR-10 dataset, ASNES demonstrates even more pronounced advantages in convergence compared to other benchmark algorithms. Furthermore, in the experiments with heterogeneous data distributions in CIFAR-10, ASNES algorithm with Nesterov momentum outperforms FedAdam by 5% in terms of performance. These findings highlight the strong performance of ASNES across different data distributions and complexities.



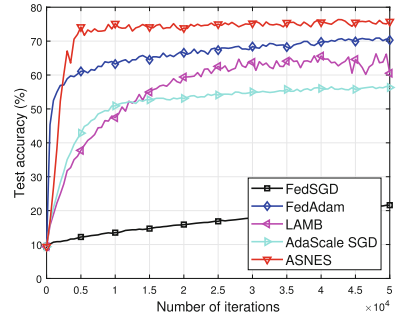
(a) Fashion-MNIST dataset under IID distribution.



(b) Fashion-MNIST dataset under non-IID distribution.



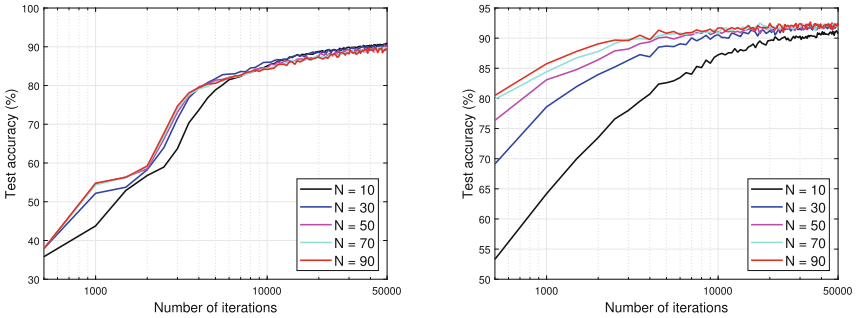
(c) CIFAR-10 dataset under IID distribution.



(d) CIFAR-10 dataset under non-IID distribution.

Fig. 1. Test Accuracy of the Fashion-MNIST and CIFAR-10 dataset

Figure 2 illustrates the training curves of ASNES algorithm with varying numbers of clients. We observe that the convergence of the ASNES algorithm improves as the number of client nodes increases. More clients participating in the training process imply a larger batch size, and ASNES can adaptively adjust the learning rate to accelerate training based on the scale of the client nodes. Particularly in the CIFAR-10 dataset, we can see a positive correlation between the convergence of the algorithm and the number of clients.



(a) Fashion-MNIST dataset. (b) CIFAR-10 dataset.

Fig. 2. The test accuracy of the Fashion-MNIST and CIFAR-10 datasets across different numbers of clients.

6 Conclusion

In this paper, we highlight that adapting the learning rate based on the scale of federated learning can reduce its training time. By observing the variance of aggregated gradients and the current client scale, we can adjust the learning rate in each iteration to accelerate the convergence of the model. Moreover, Nesterov momentum makes wiser updates by estimating the future position of parameters. By combining the learning rate adaptation method for adaptive federated learning scale and Nesterov momentum, we propose the ASNES algorithm to adapt to the dynamic changes in the number of working nodes under federated learning and accelerate the convergence speed of the model.

References

1. Charles, Z., Garrett, Z., Huo, Z., Shmulyian, S., Smith, V.: On large-cohort training for federated learning. In: Proceedings of the 35th Advances in Neural Information Processing Systems, vol. 34, pp. 20461–20475. Curran Associates, Inc., Virtual Only (2021)
2. Dian, S., et al.: To talk or to work: dynamic batch sizes assisted time efficient federated learning over future mobile edge devices. IEEE Trans. Wirel. Commun. **21**(12), 11038–11050 (2022)
3. Goyal, P., et al.: Accurate, large minibatch SGD: training imagenet in 1 hour (2018)
4. Johnson, T., Agrawal, P., Gu, H., Guestrin, C.: AdaScale SGD: a user-friendly algorithm for distributed training. In: Proceedings of the 37th International Conference on Machine Learning, vol. 119, pp. 4911–4920. PMLR, Virtual Only (2020)
5. Karimireddy, S.P., Kale, S., Mohri, M., Reddi, S., Stich, S., Suresh, A.T.: SCAFFOLD: stochastic controlled averaging for federated learning. In: Proceedings of the 37th International Conference on Machine Learning, vol. 119, pp. 5132–5143. PMLR, Virtual Only (2020)

6. Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.T.P.: On large-batch training for deep learning: Generalization gap and sharp minima. In: Proceedings of the 5th International Conference on Learning Representations. OpenReview.net, Toulon (2017)
7. Konečný, J., McMahan, H.B., Yu, F.X., Richtárik, P., Suresh, A.T., Bacon, D.: Federated learning: strategies for improving communication efficiency (2017)
8. Krizhevsky, A.: One weird trick for parallelizing convolutional neural networks (2014)
9. Lee, N., Ajanthan, T., Torr, P., Jaggi, M.: Understanding the effects of data parallelism and sparsity on neural network training. In: Proceedings of the 9th International Conference on Learning Representations. OpenReview.net, Virtual Only (2021)
10. McCandlish, S., Kaplan, J., Amodei, D., Team, O.D.: An empirical model of large-batch training (2018)
11. McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B.A.Y.: Communication-efficient learning of deep networks from decentralized data. In: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, vol. 54, pp. 1273–1282. PMLR, Ft. Lauderdale (2017)
12. Mung, C., Tao, Z.: Fog and IoT: an overview of research opportunities. IEEE Internet Things J. **3**(6), 854–864 (2016)
13. Reddi, S.J., et al.: Adaptive federated optimization. In: Proceedings of the 9th International Conference on Learning Representations. OpenReview.net, Virtual Only (2021)
14. Shallue, C.J., Lee, J., Antognini, J., Sohl-Dickstein, J., Frostig, R., Dahl, G.E.: Measuring the effects of data parallelism on neural network training (2019)
15. Shengli, L., Guanding, Y., Rui, Y., Jiantao, Y., Fengzhong, Q.: Adaptive batchsize selection and gradient compression for wireless federated learning. In: GLOBECOM 2020 - 2020 IEEE Global Communications Conference, pp. 1–6. IEEE, Taipei (2020)
16. Yang, H., Fang, M., Liu, J.: Achieving linear speedup with partial worker participation in non-IID federated learning. In: Proceedings of the 9th International Conference on Learning Representations. OpenReview.net, Virtual Only (2021)
17. Yin, D., Pananjady, A., Lam, M., Papailiopoulos, D., Ramchandran, K., Bartlett, P.: Gradient diversity: a key ingredient for scalable distributed learning. In: Proceedings of the 21th International Conference on Artificial Intelligence and Statistics, vol. 84, pp. 1998–2007. PMLR, Playa Blanca (2018)
18. You, Y., Gitman, I., Ginsburg, B.: Large batch training of convolutional networks (2017)
19. You, Y., et al.: Large batch optimization for deep learning: training bert in 76 minutes. In: Proceedings of the 8th International Conference on Learning Representations. OpenReview.net, Addis Ababa (2020)