A Survey on Communication-Efficient Federated Learning

Wu Ningyuan
No. 517021911042

***Abstract*—Recently, mobile devices are equipped with advanced computing devices, which opens up countless probabilities for researches and applications. Traditional distributed machine learning requires the mobile ends to upload the data to train the global model. However, with the increasing awareness of data privacy, a new framework named federated learning is proposed. In federated learning, participants just need to upload the local model instead of uploading the data. Yet, when the number of parameters is tremendous, the communication costs become a problem. This paper focuses on different approaches to reducing communication cost to help improve the framework performance.**

Key words: Federated Learning, communication costs

I. Introduction

In recent years, the computing capabilities of mobile devices increase dramatically to meet people's growing demands. At the same time, people generate countless data every day. However, a great amount of data cannot be utilized well because of data privacy. However, federated learning, a recently proposed privacy-preserving machine learning framework, enables companies to turn waste data into treasure, which opens up possibilities for researches and applications.

In the traditional distributed deep learning framework, data from mobile devices is processed by central server. Then the data is used to train models that are good for social progress. However, with people's increasing awareness of data privacy, data from mobile devices is harder and harder to obtain. Apart from that, endless data is uploaded, which places extremely high demands on the communication systems.

In response to this situation, Google proposes federated learning framework. In federated learning, central servers are also able to get models they want without directly getting data from mobile devices. The core idea of federated learning is to train local models on different mobile devices and then the local model parameters are uploaded and aggregated. With these parameters, desired models can be obtained. Generally, the federated learning process includes the following process [1]:

Step 1: The server decides the training task and corresponding data requirements. Then the server transmits the initialized global model to clients.

Step 2: Participants can use the received model and the information in the device to train the model to minimize the loss function. After training, the model parameters are uploaded to a central server for use.

Step 3: The central server aggregates and processes the received parameters to obtain a new global model.

Step 2 and Step 3 are repeated until the global loss function is minimized.

Compared with traditional distributed deep learning framework, federated learning has the following advantages: a)highly efficient use of bandwidth b)Privacy c)Low Latency.

However, this new framework is limited by the following challenges [2]:

1) unbalanced data: The amount of data can vary widely between devices.

2) Non-IID data: The data of each device may not be independently and uniformly distributed, which leads to the possibility that the whole trained model may not be applicable to some devices.

3) Unreliable participating clients: Some participants may maliciously tamper with the data in the device or upload parameters to cause the overall model to deviate.

Despite the aforementioned challenges, communication costs in federated learning needs to be optimized. With deep learning model involving, each iteration of update may include millions of parameters, with puts a lot of pressure on communication systems. The condition can be worsen because of unreliable network conditions of clients and the unbalanced speed of download and upload. Reassuringly, some papers are written to reducing the cost of communication in federated learning.

II. Related Works

The following methods are designed to reduce communication costs:

Terminal Computing

To reduce communication pressure, we can perform more computations on the mobile end in each iteration to make the global model achieve faster convergence, thereby reducing the number of training rounds. The authors in [3] propose two primary ways to reduce computation cost: 1) increased parallelism, in which more clients are selected into the team of uploading parameters 2) increased computation on each participant, where a new algorithm named FedAvg is proposed.

The traditional federated learning algorithm is Federated Stochastic Gradient Descent(FedSGD for short), in which a single batch gradient calculation is done per round. This approach is computationally efficient, but requires large numbers of iterations to reach convergence. The proposed Federated Averaging(FedAvg for short) algorithm is like the mini-batched

version of FedSGD. In FedAvg, clients can use a smaller batch size to increase computation to have a better local model before each round of uploading.

The authors then conduct a comparison between the FedSGD and FedAvg. After simulations, the results demonstrate that the increased parallelism does not dramatically decrease communication cost. However, the increased computation receive relatively good results. As is shown in fig.1, when the data is IID, the MNIST CNN simulation results show that the FedAvg algorithm can reduce communication costs more than 30 times. However, the increase drops to 2.8 times when the data is non-IID (using the same hyper-parameters). In most instances , data in federated learning is non-IID.

Model Compression

The authors in [4] propose the two approaches named structured and sketched updates to reducing the size of model updates.

A. Structured updates

This kind of design restricts the updates to have a pre-specified structure. Two kinds of structures are discussed in the paper: low rand and random mask.

In the low rank structure, the update is restructured to have a low rank matrix by using the product of two matrices. One is randomly generated and another is the optimized matrix. We just need to send the optimized one to the server.

In the random mask structure,update of clients are restricted to be a sparse matrix in each iteration. As a result,in each round, clients just need to sent the non-zero element to the server.

B. Sketched updates

Sketched updates mean that before uploading, updating models are compressed into an encoding form. Authors propose two ways of performing the sketching: subsampling and probabilistic quantization. In the subsampling method, each client only needs to update a random subset of the update matrix. The server then averages these updates to derive an unbiased estimate of the true average. Another way named probabilistic quantization, in which every updating matrix is vectorized and quantized. To reduce the quantization error, authors use a type of structured rotation matrix which is the product of a Walsh-Hadamard matrix and a binary diagonal matrix.

The simulation results show that random mask method achieve better accuracy than the low rank approach. However, the combination of sketching tools can achieve higher compression rate and faster convergence.

Selective Model Updating

Authors in [2] develops the communication-efficient federated deep learning framework with asynchronous model update and temporally weighted aggregation. This paper is inspired from the fine-tuning of the neural network.

A. Asynchronous Model Update

The training model can be divided into two types: one is shallow layers and another is deep layers. Shallow layers are applicable to different tasks and datasets, which means that trained parameters in shallow layers represent general features of different datasets. Correspondingly, deep layers focus on the specific features. Luckily, the global model demands that the parameters can represent general features, which provides an opportunity for researchers to reduce communication cost on this point.

In the proposed asynchronous model update, shallow layers and deep layers are updated at different frequencies. And the frequency of updating parameters of shallow layers is much larger than that of deep layers. As illustrated in the paper, this approach can save about 40 percent of communication costs.

B. Temporally Weighted Aggregation

In federated learning, participants and their data are changing in each iteration. However, as depicted in the paper, the most recently updated model ought to have a higher weight on the global model. Therefore, the paper proposed a new aggregation method which takes the timeliness into account.

$$\omega_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} * (\frac{e}{2})^{-(t-timestamp^k)} * \omega^k \qquad (1)$$

In this equation, e denotes the natural algorithm and $timestamp^k$ represents the time in iteration k. And ω^k is used to represent parameters in iteration k. The larger the difference value between t and $timestamp^k$ is, the lower the impact on the global model.

III. Discussion

In [3], the authors achieve good results by applying the idea of mini batch training to federated learning. In other words, this method reduces communication costs at the cost of increased computation, as demonstrated by results on a variety of model architectures.

However, the performance of FedAvg algorithm is sensitive to the data condition. When the data is non-IID, the performance degrade greatly. Non-IID data is widely distributed in the field of federated learning. When processing the non-IID data, we can mix private data with public shared data to reach a approximate independent distribution. Apart from that, each participant can upload the data distribution to the server, then we can perform data augmentation in datasets which obeys the distribution that is in the smaller part of all distribution types.

In [4], the key idea is the structured and sketched updates. We can see from the results that random mask approach have better accuracy than the low rank approach. The combination of sketching tools can achieve higher compression rate and faster convergence. Another model compression method is drop-out. For example, in deep learning, we can drop out some neurons to prevent over-fitting. The same idea can be applied to federated learning to help compress the model. Apart from the model compression, we can also compress the loss funtion to help save communication costs.

Apart from these two algorithms, the highlight of the paper [2] is that problems of data-unbalance,non-IID distribution in

Algorithm 2 Server Component of TWAFL

function SERVEREXECUTION ▷ Run on the server
  initialize ω_0
  for each client $k \in \{1, 2, ..., K\}$ **do**
    $timestamp_g^k \leftarrow 0$
    $timestamp_s^k \leftarrow 0$
  end for
  for each round $t = 1, 2, ...$ **do**
    if t mod $rounds\_in\_loop \in set_{ES}$ **then**§
      $flag \leftarrow$ True
    else
      $flag \leftarrow$ False
    end if
    $m \leftarrow \max(C * K, 1)$
    $S_t \leftarrow$ (random set of m clients)
    for each client $k \in S_t$ **in parallel do**
      if $flag$ **then**
        $\omega^k \leftarrow$ ClientUpdate$(k, \omega_t, flag)$
        $timestamp_g^k \leftarrow t$
        $timestamp_s^k \leftarrow t$
      else
        $\omega_g^k \leftarrow$ ClientUpdate$(k, \omega_{g,t}, flag)$
        $timestamp_g^k \leftarrow t$
      end if
    end for
    $\omega_{g,t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} * f_g(t,k) * \omega_g^k$ †
    if $flag$ **then**
      $\omega_{s,t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} * f_s(t,k) * \omega_s^k$ ‡
    end if
  end for
end function

§ $rounds\_in\_loop = 15$ and $set_{ES} = \{11, 12, 13, 14, 0\}$
† $f_g(t,k) = a^{-(t - timestamp_g^k)}$
‡ $f_s(t,k) = a^{-(t - timestamp_s^k)}$

Fig. 1: Server Component of TWAFL

Algorithm 3 Client Component of TWAFL

function CLIENTUPDATE$(k, w, flag)$ ▷ Run on client k
  $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
  if $flag$ **then**
    $\omega \leftarrow \omega$
  else
    $\omega_s \leftarrow \omega$
  end if
  for each local epoch i from 1 to E **do**
    for batch $b \in \mathcal{B}$ **do**
      $\omega \leftarrow \omega - \eta * \bigtriangledown \ell(w; b)$
    end for
  end for
  if $flag$ **then**
    return ω to server
  else
    return ω_s to server
  end if
end function

Fig. 2: Client Component of TWAFL

federated learning are also reflected by data preprocessing in the result analyses part.

As illustrated in the table 2 the proposed temporally weighted aggregation asynchronous federated learning (TWAFL for short) performs better than FedAVG. However, TWFL that employs temporally weighted aggregation without asynchronous model performs the best while AFL that adopts the asynchronous model update only without using temporally weighted aggregation performs the worst. This result indicates that temporally weighted aggregation is useful to the optimization of the process of federated learning. Though the communication cost is efficiently decreased, the algorithm still needs to be optimized.

The reason that AFL performs not that good may be that shallow layers drop too many parameters, which possibly lead to under-fitting of the local model. As a result, the number of iterations increase correspondingly.

Therefore, based on this paper, we can do an experiment on the number of layers in a shallow network for the adjustment of number of layers may be helpful to achieve the balance of the accuracy and communication cost, and further increase the performance of the whole federated learning framework.

Though the temporally aggregation updating algorithm have a good performance, it sill have space to improve. In the paper, researchers only experiment on the circumstances when a takes the value e and e/2. However, e/2 may be not the best solution. It may take some time, but it deserves. The number of clients is tremendous, even 1 percent of improvement of the whole model can save great amount of computing resources.

When we take a further look at the parameter a which controls the influence of time effect on the model aggregation, we can find that when the performance of the whole model inversely have an improvement. In another way, taking less account of timeliness leads to an improved global model, which is in conflict with the hypothesis that the most recently updated model should have a higher weight in the aggregation.

After consulting relevant materials, I found that the concept of compound interest in economics can be applied to the optimization of time-weighted model. In the enlightenment of the concept of internal rate of return, upload and download of parameters is like the inflow and outflow of cash in a bank account. The earlier the money is saved , the compound interest will be higher. Correspondingly, the earlier the parameters is uploaded, the more the impact is on the global model. This hypothesis at first sight is conflicted with the hypothesis in the paper. However, it is more of a supplement than a rebuttal. Combined with the hypothesis in the paper, we can reasonably explain the reason why the parameter a is not the bigger the better.

IV. CONCLUSION

This paper aims to have an overview of different approaches to reducing communication costs. There are some challenges and research directions in deploying communication-efficient federated learning at scale to be discussed as follows:

1) More refined model design

In [4], the authors are trying to find some new ways to simplify the updating model. However, this has to go a long way. In federated learning, clients have to upload millions of parameters. Parameters which is truly useful to the global model may be just in small proportion. We have to find effective judgement to trim unimportant filters and neurons to reduce model redundancy.

2) Effective data preprocessing

Like deep learning, the core is never inventing the perfect model, for the data and its distribution is ever-changing. The most important thing is using the effective data preprocessing method to extract useful features. With the computing capabilities of mobile ends increasing dramatically, they have the ability to process large amount of data. As a result, after effective data preprocessing, the parameters of the global model will drop greatly when the accuracy increase by a wide margin.

3) Combination of different algorithms

The paper mentioned above just talk about the new-proposed single algorithm and test its performance. Therefore, different algorithms can be combined to reduce the size of model updates. However, the trade-off between accuracy and communication overhead for the combination technique needs to be further evaluated.

V. Acknowledgement

I would like to thank Prof.Cui for the imparting of knowledge and professional help she provided in the paper writing.

References

[1] W. Y. B. Lim, N. C. Luong, D. T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, and C. Miao, "Federated learning in mobile edge networks: A comprehensive survey," *arXiv preprint arXiv:1909.11875*, 2019.

[2] Y. Chen, X. Sun, and Y. Jin, "Communication-efficient federated deep learning with asynchronous model update and temporally weighted aggregation," *arXiv preprint arXiv:1903.07424*, 2019.

[3] H. B. McMahan, E. Moore, D. Ramage, S. Hampson *et al.*, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:1602.05629*, 2016.

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