Federated Learning

Shubham Agrawal
MT22124
CSE,IIITD

Federated Learning [1] is a distributed machine learning technique that enables multiple devices to jointly learn a common model without sharing their data with a central server. This is different from traditional machine learning, where data is usually centralized and models are trained on this central data. In Federated Learning, the data is distributed across multiple devices, and each device trains the model locally using its own data and sends model updates to the central server. The Federated Learning process typically has three main components: the central server, the local devices, and the Federated Learning algorithm. The central server orchestrates the overall training process and maintains the global model. The local devices (e.g., smartphones, IoT devices) train the model using their local data and communicate updates to the central server. Finally, the Federated Learning algorithm aggregates the updates from the local devices and updates the global model. Federated Learning offers several advantages, such as preserving data privacy, reducing network communication costs, enhancing scalability, and allowing devices to benefit from the data of other devices while keeping their own data private. Federated Learning preserves users' data privacy by keeping data on local devices, as the data does not leave the user's device. Moreover, the network communication costs are lower since only model updates are exchanged between the local devices and the central server instead of the actual data. Lastly, Federated Learning can scale to large numbers of devices and is especially helpful in scenarios where data is spread across different locations (hospitals in different continents use federated learning to improve their models by reducing biases through increased data diversity). Federated optimization has several key properties that make it different from a typical distributed optimization problem. Firstly, The training data on a given client node is usually non-Identical and non-Independent based on how the mobile device is used by a specific user, his/her location, age group etc. Therefore, any specific user's local dataset will not reflect the whole population's distribution. Similarly, different usage amounts lead to different amounts of local training data. Also, the number of clients participating in a federated setting can be very high, and their network connectivity can be unstable. In the following paragraphs, we will discuss the status quo of federated learning.

There were many algorithms proposed to aggregate client local updates in a setting where the data was heterogeneous and non-IID. FedAvg(FederatedAveraging algorithm)[1] is the most fundamental algorithm for federated learning of deep networks. It is based on iterative model averaging. It combines local stochastic gradient descent (SGD) on each client with a server that performs model averaging proportional to the amount of training data on the device. The authors performed their experiments in a controlled setting with unbalanced and non-IID data as follows:- The authors adopted a setting with K clients, each possessing a distinct local dataset. At the onset of every round, the server randomly sampled a fraction C of clients and disseminated the current global state of the algorithm to them (e.g., the current model parameters). Each sampled client executed some local computation leveraging the global state and its own dataset and communicated an update to the server. The server then aggregated these updates to its global state, and the procedure repeated. In future work, they suggest providing stronger guarantees via differential privacy [2], secure multi-party computation [3], or their combination.

SCAFFOLD [4] is another federated learning algorithm and is principally concerned with data heterogeneity. The authors argued that FEDAVG suffers from 'client-drift' when the data is heterogeneous (non-iid), resulting in unstable and slow convergence. They incorporated control variates (variance reduction) to correct for the 'client-drift' in its local updates. They proved that SCAFFOLD requires significantly fewer rounds of communication and is not much affected by data heterogeneity or client sampling. Intuitively, SCAFFOLD estimates the update direction for the server model (c) and the update direction for each client ci. It then calculates the difference $(c - ci)$ as an estimate of the client-drift which is used to correct the local update. This strategy successfully overcomes heterogeneity and converges in significantly fewer rounds of communication. They further show that SCAFFOLD could additionally take advantage of the similarity between the clients to further reduce the communication rounds required, proving the advantage of taking local steps over large-batch SGD. Further, they showed that while SCAFFOLD is always at least as fast as SGD, depending on the Hessian dissimilarity in the data, can be much faster. Finally, They derive stricter convergence rates for FEDAVG than previously known for convex and non-convex functions with client sampling and heterogeneous data.

FedProx [5] is another framework that deals with heterogeneity in federated networks. In the context of systems heterogeneity, FedAvg does not let participating devices do variable amounts of local work based on their underlying system constraints, instead, it simply discards devices that cannot compute their epochs within a given time window. From a statistical perspective, FedAvg has been empirically shown to diverge in settings where the data is non-identically distributed across devices. It even lacks convergence guarantees. FedProx, a federated optimization algorithm, tackles the challenges of heterogeneity both theoretically and empirically. FedProx can be seen as a generalization and re-parametrization of FedAvg. The main idea behind FedProx in this scenario is that each of the local objectives can be solved inexactly. This allows the balance between local computation and communication to be adjusted based on the number of local iterations that are done. They use γ-inexactness in their analysis to measure the amount of local computation from the local trainer at each round. As different devices are likely to make different progress towards solving the local subproblems due to variable systems conditions like processing power, battery levels etc, it is therefore important to allow γ to vary both by device and by iteration. In FedProx, they generalized FedAvg by allowing for variable amounts of work to be done locally across devices based on their available systems resources and then aggregated the partial solutions sent from the stragglers instead of discarding these devices. They also suggested adding a proximal term to the local subproblem to effectively limit the effect of variable local updates. The proximal term is helpful in two ways: (1) It tackles the issue of statistical heterogeneity by keeping the local updates closer to the initial (global) model without any need to manually set the number of local epochs. (2) It allows for safely incorporating variable amounts of local work resulting from systems heterogeneity. Theoretically, they provide convergence guarantees for the framework when learning over data from non-identical distributions (statistical heterogeneity), and while adhering to device-level systems constraints by allowing each participating device to do a variable amount of work (systems heterogeneity). They show that FedProx allows for more robust convergence than FedAvg across a suite of realistic federated datasets. In particular, in highly heterogeneous settings, FedProx shows significantly more stable and accurate convergence behaviour relative to FedAvg, improving absolute test accuracy by 22% on average.

A lot of efforts have also been done to bring works done in centralized settings to a federated setting. This paper [6] introduces a framework, 'FEDNAS', which uses a combination of federated learning and neural architecture search (NAS) to find the best neural architecture to handle the diverse data found on different devices. In a federated setting, the data distribution is concealed from the researchers, so they have to devise or select various architectures and adjust hyperparameters remotely to accommodate the dispersed data in order to attain a superior model architecture with enhanced accuracy. This procedure is exceedingly costly because it entails numerous iterations of training on edge devices, which incur a significantly higher communication expense and on-device computational load than the data centre's environment. So the authors resorted to neural architecture search in a federated setting. NAS includes three components: the search space, the searching algorithm, and the performance estimation strategy [7]. Their search space incorporates the mixed-operation search space as proposed in DARTS [8] and MiLeNAS [9], where they build it up as an entire model architecture while searching in two shared convolutional cells. Inside the cell, to bring the candidate operations between two nodes (e.g, convolution, max pooling, skip connection, zero) to a continuous search space, they proposed mixed operation using softmax over all possible operations. In FEDNAS, each device trains a candidate neural architecture on its own data and then combines the architectures to create the final global model. They show that FEDNAS performs better than current federated learning methods through experiments on well-known datasets. In order to find a neural architecture that is customized for each device, the paper uses an improved version of the gradient-based neural architecture search method called MiLeNAS[9]. This method is well-suited for edge devices that have limited computational resources. After conducting a local search, each device sends the model architecture and weights to the server. The server then combines the weights and architecture using a weighted aggregation method to create the updated server-side model. The updated model is then sent back to each device for further exploration in the next round. This approach leads to better model performance and greater resistance to data heterogeneity. Their experiments were directed towards finding an optimal architecture α and corresponding model parameters w that can fit the non-IID dataset distributed among many workers more efficiently and thus achieve better model performance. In this, they focused on searching for CNN architecture to enhance the performance of the image classification task. The paper introduces an AutoFL system that utilizes FedNAS and is built on top of the FedML research library for federated learning. The AutoFL system consists of two main components: the Server Manager and the Client Manager. These components are encapsulated as ComManager, Trainer, and Aggregator, which provide high-level APIs for the different layers. This system can be used to solve problems in the FedNAS setting. In future works, they suggest merging search and evaluation into one stage.

In the work "Towards Understanding Biased Client Selection in Federated Learning" [10], the authors looked at how choosing some clients over others in federated learning affects how fast the model learns. They also showed mathematically that picking clients with bigger local errors makes the model learn faster. They suggested a way to pick clients based on how well the model is doing globally and used the Power-of-Choice approach, which is a fast and flexible way to pick clients that balance speed and quality. They tested their approach and found that Power-of-Choice can learn up to three times faster and get 10% better accuracy than picking clients randomly. They also made some variations of Power-of-Choice that use less communication and computation resources. To save local computation costs, instead of evaluating the loss by going through the entire local dataset, they used mini-batches to approximate it. The authors demonstrated the trade-off between convergence speed and solution bias by adjusting the size d of the candidate client set A. The selection skew increases as d increases, which leads to faster error convergence but also a higher error floor. If d = m (m = max(CK, 1), K is the total clients and C is the proportion of clients to be sampled), we have random sampling without replacement. The selection skew increases as d increases, which leads to faster error convergence but also a higher error floor. If d = m, we have random sampling without replacement in the proportion of pk (proportion of training data in client k). They proved that even with three times fewer clients in each round than random picking, Power-of-Choice can learn two times faster and get 5% better accuracy. In future works, they suggest improving the fairness and robustness of POWER-OF-CHOICE by using a different metric in step 3 of the algorithm such as the clipped loss or the q-fair loss proposed instead of the one proposed.

Federated learning, which was first introduced by Google in 2017, is an area of ongoing research that is continuously evolving. The papers mentioned above showcase cutting-edge research on federated learning and its applications in various domains. Despite this progress, many questions and areas for future exploration remain, such as ensuring fairness and accountability in federated learning, dealing with dynamic and asynchronous client participation, incorporating domain knowledge and transfer learning in federated settings, and evaluating and benchmarking federated learning systems. Despite these open questions, federated learning is a rapidly advancing field with great potential to transform machine learning and enable new AI applications.

References

[1] McMahan, B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data. In Proceedings of AISTATS, pp. 1273–1282, 2017.

[2] Cynthia Dwork and Aaron Roth. The Algorithmic Foundations of Differential Privacy. Foundations and Trends in Theoretical Computer Science. Now Publishers, 2014.

[3] Slawomir Goryczka, Li Xiong, and Vaidy Sunderam. Secure multiparty aggregation with differential privacy: A comparative study. In Proceedings of the Joint EDBT/ICDT 2013 Workshops, 2013.

[4] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh. 2020. SCAFFOLD: Stochastic controlled averaging for federated learning. In Proceedings of the International Conference on Machine Learning, Vol. 119. 5132–5143.

[5] Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A. & Smith, V. Federated optimization in heterogeneous networks. arXiv preprint arXiv:1812.06127 (2018).

[6] He, C., Annavaram, M. & Avestimehr, S. Fednas: Federated deep learning via neural architecture search. https://sites.google.com/view/cvpr20-nas/ (2020).

[7] Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren, editors. Automatic Machine Learning: Methods, Systems, Challenges. Springer, 2019.

[8] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. arXiv preprint arXiv:1806.09055, 2018.

[9] Chaoyang He, Haishan Ye, Li Shen, and Tong Zhang. Milenas: Efficient neural architecture search via mixed-level reformulation. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

[10] Yae Jee Cho, Jianyu Wang, and Gauri Joshi. 2022. Towards Understanding Biased Client Selection in Federated Learning. In &lt;u&gt; Proceedings of The 25th International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 151)&lt;/u&gt; Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera (Eds.). PMLR, 10351–10375. https://proceedings.mlr.press/v151/jee-cho22a.html