EE5102/CS6302 - Advanced Topics in Machine Learning

Assignment 4 - Federated Learning

08 November 2024

Submission Guidelines

This is the fourth graded assignment of the course, counting towards 10% of your final grade. This is a group assignment, with groups previously assigned. Each group should make one submission, but evaluations will be done individually. Clearly specify each member's contribution in the report. Ambiguous contributions may lead to mark deductions or, if necessary, individual viva exams.

Please follow the guidelines below carefully while preparing your submission. You need to submit:

- A comprehensive report in pdf format (generated from the provided LaTeX template). This pdf should consolidate all work done throughout the assignment, with links to any associated code files. Organize the report to clearly present your analysis of each task. Divide the report into sections, including:
  - **Introduction:** A brief overview of the assignment's objective, datasets, and selected models.
  - **Methodology:** Describe your approach to each task, including dataset preparation, model configurations, assumptions, and task setup.
  - **Results:** Present the results for each task, including relevant visualizations (e.g., plots, images) that support your findings.
  - **Discussion:** Summarize key insights and observations from your results, concluding with a summary of main takeaways.
  - **Contribution:** Detailed breakdown of each group member's contribution.
- Submit the assignment on LMS by Friday, 22 November 2024. Start early; no extensions will be granted.

Introduction

Current development of deep learning has caused significant changes in numerous research fields, and had profound impacts on almost all societal and industrial sectors, including computer vision, natural language processing, multimodal learning, medical analysis, graph learning, and more. However, the success of deep learning heavily relies on large-scale data and there has been increasing awareness in the public and scientific communities for data privacy. Specifically, in the real world, data is commonly distributed among different entities (e.g., edge devices and companies). Due to the increasing emphasis on data sensitivity, strict legislations have been proposed to govern data collection and utilization. Thus, the traditional centralized training paradigm, which requires to aggregate data, fails to deploy in realistic scenarios. Driven by such realistic challenges, federated learning (FL) [2] has emerged as a popular research field because it can train a global model for different participants without centralizing data owned by the distributed parties.

FL, in its essence, is a machine learning framework designed for scenarios where data is distributed across multiple clients (e.g., mobile devices, sensors, or organizations) that cannot share their data due to privacy, security, or data ownership constraints. The objective is to collaboratively train a model that performs well on the combined data across all clients, without moving the data from its source. Federated learning has several practical applications, such as predictive text on mobile devices, personalized healthcare models, and smart city systems. Roughly speaking, the classical federated paradigm can be abstractly divided into the following two steps: server-side collaboration and client-side optimization. The former could be regarded as that a central server aggregates parameters from participants and then distributes the global model (averaged parameters) back. The latter represents that the client optimizes the distributed model on the local private data. Therefore, FL achieves the privacy-preserving collaboration to learn a shared model without data consolidation.

Despite great advancements in federated learning, current federation has three major challenges as:

1. Generalization. The distributed data is normally collected from different sources with diverse preferences and naturally brings the non-independent and identically distributed (Non-IID) characteristics.

2. Robustness. As a privacy-aware collaborative paradigm, ensuring federated robustness plays a crucial role in guaranteeing federated effectiveness. In practice, due to its distributed nature, federated learning fails to ensure the client trustworthiness and is highly vulnerable to different malicious behaviors. A small set of malicious clients can easily manipulate the training process by uploading poisoned local models to the server.
3. Fairness. Federated learning functions as a collaborative paradigm. The crucial cooperation pre-requirement is to satisfy the multi-party interest allocation for its sustained development viability.

In this course, we have only focused on generalization aspect.

Federated Average (FedAvg): The foundational approach in FL is FedAvg where each client trains a model with the same architecture on its local data and then sends its model updates to a central server. The server aggregates these updates by averaging them and sends the result back to each client. This approach performs reasonably well when the data distributions across clients are similar (homogeneous). However, FedAvg struggles in heterogeneous data scenarios where each client's local data distribution differs significantly.

Following the standard federated learning setup, suppose there are M clients, (indexed by i) with respective private data D_i. $N_i = |D_i|$ means the private data scale for the ith client private dataset. We further assume each data sample (x, y), where x is the input attribute and y is the label. The local data follows the distribution $P_i(x, y)$. We denote the global model parameter as w_G. Formally, federated solutions generally seek to learn an ideal global model, to minimize the weighted empirical loss among clients as:

$$w_G^* = \min_w \sum_{i=1}^{M} \alpha_i L_i(w, D_i), \tag{1}$$

where α_i denotes the pre-allocated aggregation weight and $L_i(w, D_i)$ represents the empirical loss representing client-specific loss function. The process can further be disassembled into the following three steps:

$$\begin{aligned} w_i &\leftarrow w && \text{Distribute} \\ \min_{w_i} \mathbb{E}_{x \in D_i}&[L_i(w_i, x)] && \text{Optimize} \\ w_G &= \sum_{i=1}^{M} \alpha_i w_i && \text{Aggregate} \end{aligned} \tag{2}$$

Client Drift: The challenge arises because each client's model optimizes for a local objective that may conflict with the objectives of other clients. Consequently, the model learned by each client could represent a different function, leading to *client drift* — where each client drifts toward a solution suitable for its local objective, rather than the global one. In such cases, the averaged global model may become ineffective, as it combines incompatible local functions.

To better understand this, consider a scenario where we have five noisy approximations of a single underlying function. Averaging these approximations typically produces a good estimate of the true function, as averaging emphasizes shared patterns and reduces noise. This is essentially what happens in the homogeneous case, where each client's local objective aligns with the global objective. However, if we instead have five vastly different functions — each suited to distinct local objectives — their average will likely lack the functionality of any individual function and may even be unrecognizable. This issue is exacerbated in neural networks, which are non-convex and thus do not preserve functionality under simple averaging. Consequently, averaging models trained on different distributions results in a global model that may not perform well for any client.

Addressing Heterogeneity in FL: If averaging locally trained models is ineffective in heterogeneous cases, how should we address heterogeneity? A key insight is that differentiation is a linear operation, meaning that if we limit each client to one gradient step per round, the aggregated updates across clients approximate the gradient of the global objective. This idea is the basis of **FedSGD**.

To illustrate, consider a centralized classification objective $L_{\text{centralized}}$. Given data points $\{x_1, x_2, \ldots, x_n\}$ and labels Y, we define the centralized objective as the sum of individual loss terms:

$$L_{\text{centralized}} = \sum_{i=1}^{n} \|Y - f(w, x_i)\|^2,$$

where w represents the model parameters. Since differentiation is linear, the gradient of this global objective is:

$$\nabla L_{\text{centralized}} = \sum_{i=1}^{n} \nabla \|Y - f(w, x_i)\|^2.$$

In federated learning, the total dataset remains the same across the system, but each client only has access to a subset of it. The gradient computed by client i on its local dataset $D_i = \{x_1, x_2, \ldots, x_m\}$ is:

$$\nabla L_{\text{client } i} = \sum_{j=1}^{m} \nabla \|Y - f(w, x_j)\|^2.$$

When the server aggregates the gradients from all clients, the result is equivalent to the total gradient over all data points. This equivalence holds because the total dataset in both the centralized and federated scenarios remains the same, and the order of addition does not affect the sum. Here, we assume that the global objective is defined as the sum of the local losses rather than their average, though this distinction is not significant.:

$$\nabla L_{\text{global}} = \sum_{i=1}^{n} \nabla \|Y - f(w, x_i)\|^2,$$

which matches the centralized gradient.

This equivalence between federated and centralized learning holds as long as each client performs only one gradient descent step per round before aggregation. This approach, known as **FedSGD**, guarantees convergence to the centralized solution. However, it is impractical in real-world scenarios where frequent communication between clients and the server is not feasible. Therefore, there is a need for algorithms that can effectively handle data heterogeneity, even when clients perform multiple local updates between communication rounds.

This assignment focuses on understanding the issue of **local drift** caused by data heterogeneity and evaluating different methods to address it. By exploring these approaches, you will gain insight into how researchers tackle this open challenge in federated learning.

Task 1: FedSGD vs Centralized

In this task, you will study the theoretical equivalence between FedSGD and centralized training in a controlled scenario — with no averaging, no mini-batches, and full gradient descent (GD) steps. The code for both methods is provided in the accompanied Notebook.

Your task is to identify and correct any mistakes that prevent the gradients from being similar in both scenarios. If you believe that FedSGD and Centralized training should not be equivalent, provide a reasoned argument to support your view.

Task 2: FedAvg under Varying Levels of Heterogeneity [3]

In this task, you will implement the FedAvg algorithm and evaluate its performance at different levels of label heterogeneity, using Dirichlet distributions to vary the label skew. Plot the accuracy results and observe the impact of increasing heterogeneity on model performance. You should notice that as label skew increases, performance typically decreases. Refer to the Notebook Task 2.

Once you have examined how varying heterogeneity affects FedAvg, explore these questions.

1. Consider an extreme scenario with 10 clients and 10 classes, where each client only has data from one class. Each client trains its model using cross-entropy loss on its single-class dataset (e.g., a client with only "cat" images trains on just those images). Reflect on what you think the local model will learn in this case. Will it learn features specific to each class, such as "cat" features? If not, explain your reasoning. You may want to experiment with this setup before you answer.

2. Imagine a hypothetical scenario where Yann LeCun and Mustafa Siddiqui (the "King of Federated Learning") walk into a bar. They begin debating the impact of permutation invariance on FedAvg.

   Yann argues that even if two neural networks approximate the same function, their weights can vary significantly due to permutation invariance. For instance, in a layer with two convolutional filters, client 1's model might have the first filter learning edges and the second learning textures, while client 2's model has these reversed. Although these models functionally represent the same features, averaging their weights could lead to a meaningless result, as corresponding filters wouldn't align correctly. Yann concludes that, because of this, even in homogeneous scenarios, averaging weights might not make sense.

Mustafa acknowledges Yann's point but insists that FedAvg works empirically. He then proceeds to debunk Yann's argument by pointing out a key aspect of FedAvg that Yann has overlooked.

What do you think is the flaw in Yann's argument?

Task 3: Implementing SCAFFOLD for Heterogeneity [1]

In this task, you will implement the SCAFFOLD algorithm, a technique designed to mitigate the effects of data heterogeneity by controlling client drift. SCAFFOLD uses control variates at both the client and server levels to adjust local updates, helping to keep the local objectives aligned with the global objective. Please refer to the mentioned paper for further details.

Mathematical Steps for SCAFFOLD

1. **Control Variates Initialization**: Each client i and the server maintain control variates, c_i and c respectively. These control variates are used to adjust the updates and counteract local drift.
2. **Local Model Update**: For each client i in a given round:
    - The client initializes its local model with the current global model, $w_i \leftarrow w_G$.
    - It then performs K local update steps, with each step adjusting the model parameters using:

      $$w_i \leftarrow w_i - \eta\,(g_i(w_i) + c - c_i),$$

      where $g_i(w_i)$ is the mini-batch gradient at w_i, c is the server control variate, and c_i is the client control variate. This adjustment reduces client drift by aligning local updates with the global objective.
3. **Control Variate Update at the Client**: After completing all K local updates, the client updates its control variate to:

   $$c_i^+ \leftarrow c_i - c + \frac{1}{K\eta}(w_G - w_i),$$

   where w_G is the initial global model and w_i is the locally updated model after K steps. This update helps synchronize the client's control variate with the server's.
4. **Global Model Update at the Server**: The server aggregates the client updates to adjust the global model and its global covariate variable. They can be aggregated in different ways. Refer to original paper and implement that way of aggregation in your implementation.

Once your implementation is complete and you have compared it with FedAvg. for varying degrees of heterogeneity levels, explore the answer to the following.

Suppose someone modified the SCAFFOLD algorithm so that, at the start of each communication round, each client sets its local control variate c_i equal to the global control variate c received from the server (i.e., $c_i = c$), instead of maintaining a client-specific c_i across rounds. How would this modified version of SCAFFOLD compare to FedAvg?

Task 4: Gradient Conflict Analysis and Harmonization [5]

In this task, you will explore how data heterogeneity affects the consistency of client updates in federated learning. Specifically, you will examine how increasing heterogeneity leads to more conflicts among client updates. After that, you will implement the Gradient Harmonization (FedGH) algorithm, which aims to address these conflicts and improve performance in heterogeneous data scenarios. A **conflict** is defined as a situation where two updates have an angle greater than 90 degrees, or equivalently, a negative cosine similarity. Gradient Harmonization resolves these conflicts to ensure that client updates are better aligned, which helps mitigate the effects of data heterogeneity. In this task, you are supposed to implement FedGH [5] that combines FedAvg. with gradient harmonization.

Once your implementation is complete and you have compared it with FedAvg. for varying degrees of heterogeneity levels, explore the answer to the following.

In Gradient Harmonization, adjusting conflicting gradients changes each client's update direction from what it originally would have been without alignment.

1. Is it guaranteed that this new, harmonized gradient direction will lead to a minimization step towards the client's local objective? Or could it possibly increase the local objective's loss for some clients?

2. Consider an extreme scenario where each client's data distribution is vastly different, causing each client to learn a highly unique model during local training. If all clients' gradients are heavily conflicted, how might this affect the magnitude of the final harmonized gradient? Do you think it would be large, small, or unpredictable, and what implications might this have on the effectiveness of the global model update?

Task 5: Sharpness-Aware Minimization in Federated Learning (FedSAM) [4]

In this task, you will implement FedSAM, a federated learning variant that aims to find flatter minima at the local level, rather than sharp ones. The key idea in FedSAM is to use a minimax optimization approach, where each local client perturbs its model weights in the direction of gradient ascent to maximize the loss, and then performs gradient descent from this perturbed point. This helps the model generalize better by avoiding sharp minima, which are sensitive to small changes in input or model weights. Mathematically, the FedSAM objective is formulated as:

$$\min_{w} \max_{\|\delta_i\|_2 \leq \rho} \left\{ L(\tilde{w}) := \frac{1}{N} \sum_{i \in [N]} L_i(\tilde{w}) \right\},$$

where $\tilde{w} = w + \rho \frac{\nabla L_i(w)}{\|\nabla L_i(w)\|}$ is the perturbed model with the highest loss within a neighborhood defined by ρ, and L_i is the local objective for client i.

Once your basic implementation of this Task is complete and you have compared it with FedAvg. for varying degrees of heterogeneity levels, explore the answer to the following.

1. Why might a flatter minimum lead to a more generalizable solution than a sharp minimum?
2. In what ways does this property help address the challenges of data heterogeneity in federated learning?
3. Explore some other alternate measure of sharpness to measure the flatness of the region. You may search the literature or take help from some LLM regarding this task. Implement this in the basic FedSAM and see how does it compare with first-order approximation of flatness measure i.e. $\rho \frac{\nabla L_i(w)}{\|\nabla L_i(w)\|}$

Task 6: Consolidated Write-up Guidelines

While preparing your PDF file for submission, please focus on fair comparison of all the algorithm in discussion section. In particular, you should discuss and highlight merit and demerits of all the algorithms considered: Centralized, FedSGD, FedAVg, SCAFFOLD, FedGH, FedSAM, on the lines:

1. How do they compare in terms of handling increased levels of data heterogeneity?
2. How do they compare on terms of communication efficiency?
3. How do they compare in terms of fast or slow convergence rates? You can *at least* empirically evaluate this aspect as well.

References

[1] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian U Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 5132–5143. PMLR, 2020.

[2] Jakub Konečný, H Brendan McMahan, Daniel Ramage, and Peter Richtárik. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, 2016.

[3] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1273–1282. PMLR, 2017.

[4] Z Qu, X Li, R Duan, Y Liu, B Tang, and Z Lu. Generalized federated learning via sharpness aware minimization. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*. PMLR, 2022.

[5] X Zhang, W Sun, and Y Chen. Tackling the non-iid issue in heterogeneous federated learning by gradient harmonization. *IEEE Signal Processing Letters*, 2024.