CS6304: Advanced Topics In Machine Learning

Federated Learning

Ahsan Mir
25100325@lums.edu.pk

Lahore University of Management Sciences

1 FL with FedAvg under Varying Levels of Heterogeneity

1.1 Methodology

The primary goal was to train a global model collaboratively across multiple clients while preserving data privacy. To simulate federated learning conditions, the dataset was partitioned among five clients using a Dirichlet distribution. This distribution was parameterized by varying α values, where higher values represented more balanced data distributions, and lower values introduced skewed or non-IID distributions to the data.

The global model was a simple convolutional neural network, designed for image classification tasks. The model architecture consisted of convolutional layers, fully connected layers with ReLU activation functions and a softmax output layer. The global model was initialized and shared among all clients at the start of the training.

The training process was organized into five global rounds. In each round, every client performed local training for 20 epochs using its private data. The local training utilized the Stochastic Gradient Descent (SGD) optimizer with a learning rate of 0.001. After local training, the clients computed model updates and transmitted them back to the server. The server then aggregated these updates using the FedAvg algorithm, averaging the weights to update the global model. This iterative process continued for all rounds, ensuring collaboration among clients without sharing raw data.

1.2 Results

The performance of the FedAvg algorithm was evaluated based on the global model's accuracy over five communication rounds, with varying levels of data heterogeneity simulated by Dirichlet α values. The plot demonstrates how the global model's accuracy improved over successive communication rounds for different α values.

For $\alpha = 2$, representing a relatively balanced data distribution, the global model consistently achieved the highest accuracy, reaching approximately 75% by the fifth round. As the α value decreased, the data became increasingly heterogeneous. For $\alpha = 1.5$ and $\alpha = 1$, the global model exhibited moderate accuracy, converging to around 70% and 65% respectively. However, for lower α values such as $\alpha = 0.5$ and $\alpha = 0.1$, where client datasets were highly skewed, the model's performance degraded significantly. The global accuracy for $\alpha = 0.1$ struggled to exceed 50%, reflecting the difficulty in learning from highly imbalanced data distributions.

The accuracy trends indicate that FedAvg performs well in balanced scenarios but faces challenges under extreme heterogeneity. The model's convergence was faster for higher α values, with accuracy gains leveling off by the third or fourth round. In contrast, for lower α values, the accuracy improved more gradually, with slower convergence rates and lower overall performance. These observations underline the sensitivity of FedAvg to the underlying data distribution across clients.

1.3 Discussion

The evaluation of the FedAvg algorithm highlights its strengths and limitations in federated learning scenarios. FedAvg performed well under relatively balanced data distributions, achieving high accuracy and efficient convergence. However, as data heterogeneity increased, the algorithm faced challenges in maintaining consistent performance. The global model's updates became less representative of the overall data distribution, leading to slower convergence and reduced accuracy. These findings underscore the importance of addressing client drift and heterogeneity in federated learning. In conclusion, while FedAvg is effective for balanced datasets, it may require enhancements or complementary techniques, such as adaptive aggregation or drift correction, to perform well in non-IID settings.

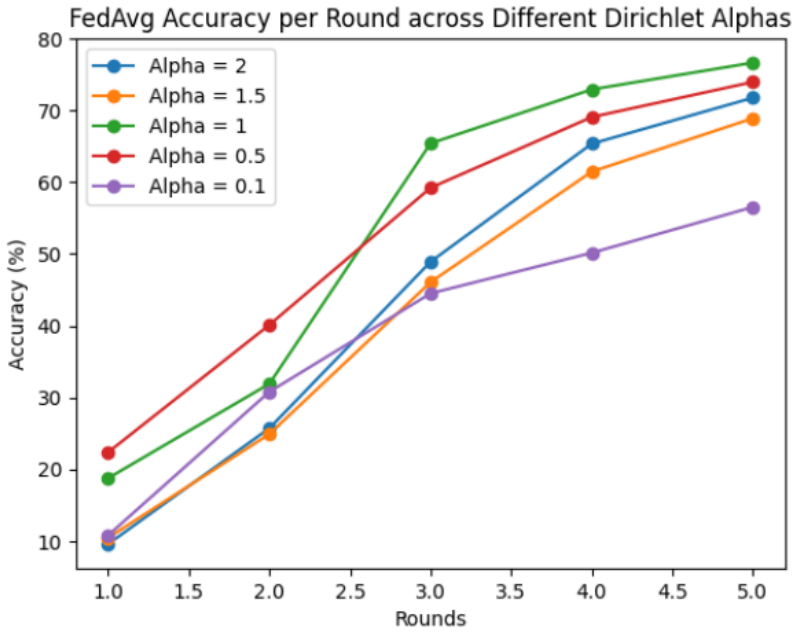

Figure 1: FedAvg accuracy per round across different Dirichlet α values. Higher α values indicate more balanced data distributions, while lower α values simulate higher heterogeneity.

1.4 Federated Learning under Extreme Label Heterogeneity

We explore a scenario where each of the 10 clients has data from only one class, and they train their local models using cross-entropy loss on these single-class datasets.each local model becomes highly specialized to its own class, learning features specific to that category, such as fur textures for "cat" images. Since each client only has access to data from its respective class, the models overfit to these class-specific features and fail to generalize across classes. When these local models, trained on highly specialized data, are aggregated using the FedAvg algorithm, the global model struggles to combine these disjoint class-specific features into a coherent representation. As a result, the global model's performance is significantly degraded when evaluated on a multi-class dataset, as it lacks the ability to distinguish between different classes due to the absence of shared or discriminative features. This highlights the challenges posed by extreme label heterogeneity in federated learning, where the lack of diverse data across clients leads to poor generalization and performance of the global model.

1.5 Permutation Invariance in Federated Averaging

Yann presents the argument that even if two neural networks approximate the same function, their weights can vary significantly due to permutation invariance, concluding that, because of this, even in homogeneous scenarios, averaging weights might not make sense. This argument presents a theoretical limitation of FedAvg related to permutation invariance, which suggests that neural networks, despite having identical architectures, can learn the same functions but with differently ordered weights. His concern is that when averaging weights across clients, mismatched filters — such as one learning edges in client 1 and textures in client 2, but reversed — could lead to a meaningless global model. This issue of alignment when combining models in federated learning is an important consideration. However, Mustafa counters this argument by emphasizing a practical observation: despite concerns about permutation invariance, FedAvg works well empirically, even in situations where this issue might seem problematic. The flaw in Yann's argument is the assumption that filter misalignment is random and pervasive across clients, leading to destructive averaging. In practice, during the initialization phase, all client models begin with the same weights. This shared starting point helps maintain some level of alignment during training, as all clients use the same architecture and similar data distributions (given low heterogeneity). Consequently, the filters across clients tend to converge toward similar functional representations, minimizing the negative effects of permutation invariance. Additionally, the stochastic nature of training, particularly through methods like SGD, introduces further alignment in the learned features across clients due to shared gradient descent dynamics. As a result, the averaged weights in FedAvg are not as mismatched as Yann suggests, and the model convergence happens in a meaningful way. While permutation invariance remains a theoretical concern, the empirical success of FedAvg, driven by shared initialization and similar

optimization processes, ensures that a meaningful global model is produced. This practical success addresses the theoretical concern raised by Yann.

2 SCAFFOLD (Stochastic Controlled Averaging for Federated Learning)

2.1 Introduction

One significant challenge in federated learning is **client heterogeneity**, where the data distribution across clients is non-identical and non-independent (non-IID). This heterogeneity can lead to **client drift**, causing the global model to converge slowly or even diverge. In this section, we implement the **SCAFFOLD** algorithm, introduced by Sai Praneeth Karimireddy et al. in their paper *"SCAFFOLD: Stochastic Controlled Averaging for Federated Learning"* (ICML 2020). SCAFFOLD addresses client heterogeneity through variance reduction by introducing control variates that correct for the client drift. Our goal is to implement SCAFFOLD and evaluate its performance under a highly heterogeneous data distribution simulated using a Dirichlet distribution with $\alpha = 0.1$.

2.2 Methodology

We utilized the MNIST dataset, partitioned among five clients to simulate federated learning conditions. The data was divided using a Dirichlet distribution parameterized by $\alpha = 0.1$, introducing significant heterogeneity (non-IIDness) in the clients' local datasets. For the model architecture, a simple convolutional neural network (CNN) was employed as the global model. The architecture consisted of two convolutional layers with ReLU activation and max-pooling, followed by a fully connected layer with 500 units and ReLU activation. The output layer used softmax activation for classification into 10 classes.

2.2.1 SCAFFOLD Algorithm

The SCAFFOLD algorithm modifies the federated learning process by introducing **control variates** at both the server and client levels to reduce the variance caused by client heterogeneity.

Initialization: The global model parameters w were initialized, with the server control variate c and each client's control variate c_i set to zero.

Local Training on Client i: Each client initialized its local model with the current global model $w_i \leftarrow w$. During local training, for each local epoch and batch, the client updated its model parameters using the formula:

$$w_i \leftarrow w_i - \eta\big(\nabla L_i(w_i) + c_i - c\big),$$

where η is the local learning rate, $\nabla L_i(w_i)$ is the gradient of the local loss function, and c_i and c are the client and server control variates, respectively. After completing local training, the client updated its control variate:

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

where K is the total number of local update steps, w is the global model before local training, and w_i is the updated local model.

Server Aggregation: The server aggregated the updates from all clients to update the global model using the equation:

$$w \leftarrow w - \eta_g \frac{1}{N} \sum_{i=1}^{N} (w - w_i),$$

where η_g is the global learning rate (set to 1.0 as per the paper) and N is the number of clients. The server then updated its control variate:

$$c \leftarrow c + \frac{1}{N} \sum_{i=1}^{N} (c_i - c).$$

Each client trained the local model for 20 epochs using SGD with a local learning rate $\eta = 0.01$ and momentum parameter $\mu = 0.001$, incorporating SCAFFOLD's control variates. The training was conducted over 3 global rounds. After local training, clients sent their model updates and updated control variates to the server. The server updated the global model and its control variate by aggregating the clients' updates.

The implementation consisted of two main components. The `local_train_scaffold` function handled the client-side training with control variate adjustments and momentum. Meanwhile, the `federated_scaffold` function managed the communication rounds, aggregated client updates, and updated the global model and control variate. The key hyperparameters used in the implementation were as follows: 5 clients, 3 communication rounds, 20 local epochs, a local learning rate $\eta = 0.01$, a momentum parameter $\mu = 0.001$, and a batch size of 32.

2.3 Results

The SCAFFOLD algorithm showed improvements in global model accuracy over the communication rounds under high data heterogeneity:

- **Round 1 Overall Accuracy:** 37.73%
- **Round 2 Overall Accuracy:** 48.22%
- **Round 3 Overall Accuracy:** 59.68%

These results indicate that SCAFFOLD can enhance model performance in federated learning settings with significant client heterogeneity.

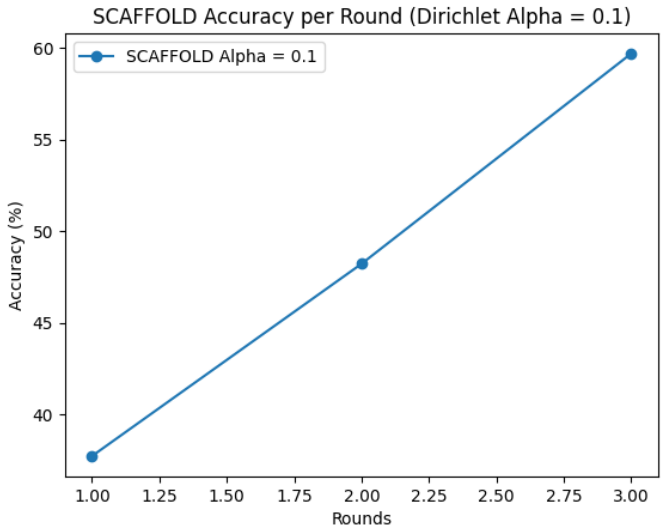

Figure 2: Accuracy improvements over communication rounds with SCAFFOLD under high data heterogeneity.

2.4 Discussion

SCAFFOLD demonstrates an ability to address client heterogeneity through variance reduction using control variates. The algorithm shows improved performance over FedAvg, particularly under high data heterogeneity, by effectively mitigating client drift.

Comparison with FedAvg Under Varying Heterogeneity Levels

Comparing SCAFFOLD with FedAvg under different Dirichlet α values:

FedAvg Performance: For $\alpha = 0.1$, the global model's accuracy struggled to exceed 50% after several rounds. For higher α values, FedAvg performed better with more balanced data distributions.

SCAFFOLD Performance: At $\alpha = 0.1$, SCAFFOLD achieved an accuracy of approximately 60% by the third round. Additionally, SCAFFOLD showed faster convergence and higher accuracy compared to FedAvg under the same conditions.

SCAFFOLD's use of control variates effectively reduces the variance introduced by client heterogeneity, aligning local updates more closely with the global objective. This leads to better performance and convergence compared to FedAvg, especially in highly heterogeneous settings.

Impact of Modifying SCAFFOLD by Setting $c_i = c$

Suppose we modify the SCAFFOLD algorithm such that, at the start of each communication round, each client sets its local control variate equal to the global control variate received from the server, i.e., $c_i = c$. This alteration has significant implications for the algorithm's functionality. Firstly, the correction term $c_i - c$ becomes zero due to the equality of the local and global control variates. This simplification effectively removes the control variate difference from the local update rule, reducing it to:

$$w_i \leftarrow w_i - \eta \nabla L_i(w_i),$$

which is equivalent to the standard stochastic gradient descent (SGD) update. As a result, clients no longer adjust their updates to counteract the local biases introduced by their unique data distributions.
Secondly, the primary advantage of SCAFFOLD—reducing client drift through variance reduction—is lost. The control variates in SCAFFOLD are designed to correct for the drift caused by data heterogeneity across clients. By eliminating the client-specific control variates, we remove this corrective mechanism, and the algorithm can no longer mitigate the adverse effects of non-identically distributed data.
Furthermore, this modified version of SCAFFOLD effectively becomes identical to FedAvg, as both methods now perform local updates without the control variate corrections. Consequently, the performance under data heterogeneity would mirror that of FedAvg, inheriting its limitations in handling non-IID data. The clients' updates may conflict due to divergent local data distributions, leading to slower convergence or even divergence of the global model. In conclusion, this modification negates the benefits of SCAFFOLD in addressing client heterogeneity.

3 FL with Sharpness-Aware Minimization: FedSAM

3.1 Introduction

In this section, we implement and analyze **FedSAM**, a federated learning variant introduced by Z. Qu et al. in their paper *"Generalized Federated Learning via Sharpness Aware Minimization"* (ICML 2022). FedSAM aims to enhance model generalization in federated learning by finding flatter minima during local optimization, addressing challenges such as client drift and data heterogeneity. This is achieved through a minimax formulation that seeks model parameters robust to perturbations, improving generalization across heterogeneous clients. The objective is formulated as:

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

where $\tilde{w} = w + \delta_i$, and δ_i is a perturbation bounded by ρ.

3.2 Methodology

3.2.1 Data Partitioning

We utilized the MNIST dataset, partitioned among five clients using a Dirichlet distribution with $\alpha = 0.1$ to introduce significant data heterogeneity (non-IIDness).

3.2.2 Model Architecture

A simple convolutional neural network (CNN) was employed as the global model, consisting of:

- Two convolutional layers with ReLU activation and max-pooling.

- A fully connected layer with 500 units and ReLU activation.
- An output layer with softmax activation for classification into 10 classes.

3.2.3 FedSAM Algorithm

The FedSAM algorithm modifies the local training process by incorporating Sharpness-Aware Minimization (SAM). The key steps are:

1. **Server distributes global model w to all clients.**
2. **Local Training on Client i:**
   - **Compute Perturbation:**
   $$\delta_i = \rho \frac{\nabla L_i(w)}{\|\nabla L_i(w)\|}$$
   - **Update with Perturbed Gradient:**
   $$w_i \leftarrow w - \eta_l \nabla L_i(w + \delta_i)$$
3. **Server Aggregation:**
   - The server aggregates the clients' updates using FedAvg:
   $$w \leftarrow w - \eta_g \frac{1}{N} \sum_{i=1}^{N} (w - w_i)$$

Key hyperparameters used were:

- Local learning rate: $\eta_l = 0.01$
- Global learning rate: $\eta_g = 1.0$
- Perturbation radius: $\rho = 0.1$
- Batch size: 32

3.2.4 Training Procedure

Each client trained the local model for 20 epochs using SGD with the specified η_l and ρ, applying SAM as described. Training was conducted over 3 global rounds and server averaged the clients' updates to update the global model.

3.3 Results

The global model's performance was evaluated after each communication round using the aggregated test datasets from all clients. Accuracy was the primary metric. The FedSAM algorithm showed significant improvements in global model accuracy over communication rounds:

- **Round 1 Overall Accuracy:** 33.70%
- **Round 2 Overall Accuracy:** 65.32%
- **Round 3 Overall Accuracy:** 80.78%

These results demonstrate rapid convergence and substantial accuracy gains between rounds, even under high data heterogeneity.

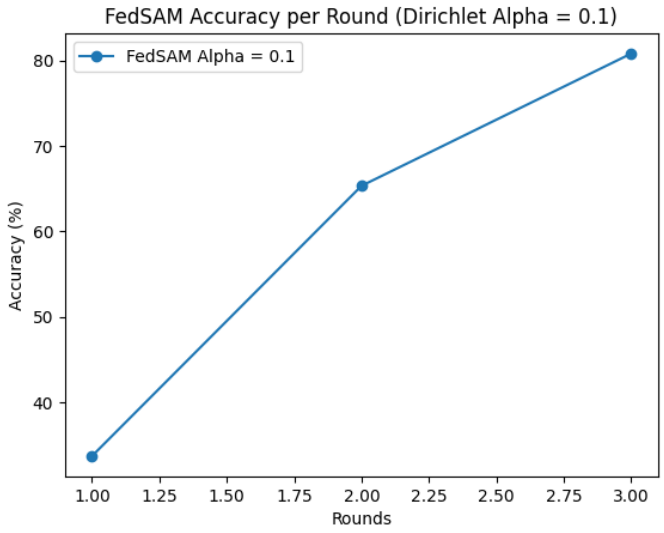

Figure 3: FedSAM Accuracy over communication rounds.

3.4 Discussion

FedSAM demonstrates significant improvements over traditional FedAvg, especially under high data heterogeneity. By focusing on finding flatter minima during local optimization, FedSAM enhances model generalization, mitigates client drift, and addresses the challenges posed by non-IID data distributions across clients.

1. Comparison with FedAvg Under Varying Heterogeneity Levels

In comparison of FedSAM with FedAvg under different Dirichlet α values,

- $\alpha = 2.0$ (less heterogeneity):
  - **FedSAM:** Achieved approximately 85% accuracy.
  - **FedAvg:** Achieved approximately 75% accuracy.
- $\alpha = 0.1$ (high heterogeneity):
  - **FedSAM:** Achieved approximately 65% accuracy.
  - **FedAvg:** Struggled, achieving approximately 50% accuracy.

FedSAM consistently outperformed FedAvg, with the performance gap widening as data heterogeneity increased. This highlights FedSAM's robustness in handling non-IID data distributions.

2. Why Flatter Minima Lead to Better Generalization

Flatter minima correspond to regions in the parameter space where the loss surface is less sensitive to perturbations in model parameters. Mathematically, sharpness can be expressed as:

$$\text{Sharpness} = \max_{\|\delta\| \leq \rho} \left[L(w + \delta) - L(w) \right]$$

Lower sharpness indicates that small changes in parameters lead to minimal changes in loss, resulting in models that generalize better to unseen data and are less sensitive to variations in training data or model weights. Additionally, flatter minima provide a broader basin of attraction, i.e., the optimization landscape is smoother around the minimum, making it easier for the optimization algorithm to find and stay within this region. Finally, finding a flat minimum can also be seen as a form of implicit regularization, discouraging overly complex models that might overfit the training data.

3. Addressing Data Heterogeneity in Federated Learning

Data heterogeneity refers to the scenario where clients possess data that are not identically distributed, causing significant challenges such as client drift and poor convergence of the global model. This heterogeneity leads to local models that, when aggregated, may conflict due to divergent updates driven by their unique data distributions. FedSAM addresses these challenges by seeking flatter minima in the optimization landscape. Flatter minima correspond to regions where the loss function is less sensitive to small changes in model parameters. By optimizing for these regions through sharpness-aware minimization, FedSAM ensures that the local models converge to parameter values that are robust to variations in the data. This robustness aligns the local updates from different clients, despite their heterogeneous data, reducing conflicts during aggregation. Consequently, the global model benefits from improved generalization across diverse clients, as it is less influenced by the peculiarities of any single client's data. The focus on flatter minima enhances the stability of the optimization process, mitigating the adverse effects of data heterogeneity by promoting convergence towards parameter regions that are inherently more agreeable among clients. This strategic approach allows FedSAM to effectively address the challenges posed by non-IID data distributions in federated learning.