UNIT 4

FL Algorithms

- Federated Learning (FL) has various algorithms designed to address different challenges, such as **non-IID data, communication efficiency, and security**. Here are some of the most common FL algorithms:

Federated Averaging (FedAvg)

- **Description**: Clients perform multiple local training steps using SGD and send the updated model weights to the server, which averages them.
- **Advantages**: Reduces communication costs compared to simple model averaging.
- **Challenges**: Struggles with non-IID data and slow convergence in heterogeneous environments.

How Federated Averaging Works

1. **Initialization:**
    - A central server initializes a global model w^t at iteration t.
2. **Local Training:**
    - A subset of participating clients (e.g., edge devices) receive the global model.
    - Each client updates the model using its local data by performing multiple steps of **Stochastic Gradient Descent (SGD)**.
3. **Model Aggregation:**
    - The locally trained models w_i^t from the selected clients are sent back to the central server.
    - The server averages these local updates based on the number of data samples at each client:

$$w^{t+1} = \sum_{i=1}^{K} \frac{n_i}{N} w_i^t$$

where:

- K is the number of selected clients,
- n_i is the number of local data points at client i,
- $N = \sum_{i=1}^{K} n_i$ is the total data used in the round.

4. **Global Update:**
    - The server updates the global model using the averaged model weights.
    - Steps 2-4 are repeated for multiple rounds until convergence.

FL Algorithms

Federated Stochastic Gradient Descent (FedSGD)

- **Description**: Clients compute gradients using their local data and send them to the server after every training step.
- **Advantages**: Simpler than FedAvg and provides theoretical convergence guarantees.
- **Challenges**: High communication cost since gradients are sent frequently.

How FedSGD Works

FedSGD follows these key steps:

1. Initialization

- The **server** initializes a global model w^t at round t.

2. Local Gradient Computation

- A subset of **clients** is selected to participate in the current training round.
- Each client i downloads the global model w^t and computes the gradient $\nabla F_i(w^t)$ using its local dataset.

3. Gradient Aggregation

- Clients send their gradients $\nabla F_i(w^t)$ to the server.
- The server **aggregates** the gradients using a weighted average:

$$\nabla F(w^t) = \sum_{i=1}^{K} \frac{n_i}{N} \nabla F_i(w^t)$$

where:

- K is the number of selected clients,
- n_i is the number of samples on client i,
- $N = \sum_{i=1}^{K} n_i$ is the total number of samples used in that round.

4. Global Model Update

- The server updates the global model using the aggregated gradient:

$$w^{t+1} = w^t - \eta \nabla F(w^t)$$

where η is the learning rate.

- This process repeats until convergence.

- **Weights (w)** = "What the model knows" (the learned parameters).
- **Gradients (∇F)** = "How the model should improve" (guides the weight updates).

Comparison: FedSGD vs. FedAvg

Feature	FedSGD	FedAvg
Local Computation	Computes **only one** gradient step per round	Performs **multiple** local updates per round
Communication Cost	**High** (gradients sent every round)	**Lower** (only model updates sent)
Convergence Speed	**Slower** (requires more rounds)	**Faster** (due to multiple local updates)
Handles Non-IID Data?	❌ Poor	✅ Better

When to Use FedSGD?

- **If communication cost is not a concern** (e.g., high-speed networks).
- **If strict theoretical guarantees are needed** (since FedSGD is closely aligned with centralized SGD).
- **If client devices have very limited compute power**, making multiple local updates (as in FedAvg) impractical.

Feature	FedSGD	FedAvg
Local Computation	Each client performs **only one** gradient update per round.	Each client performs **multiple** local updates before sending updates.
Communication Cost	**High** – Clients send gradients at every step, increasing bandwidth usage.	**Lower** – Clients send model weights after multiple local updates.
Update Aggregation	Server **averages gradients** received from clients.	Server **averages model weights** received from clients.
Convergence Speed	**Slower** – Requires more communication rounds due to frequent updates.	**Faster** – Multiple local updates improve convergence.
Handles Non-IID Data?	**Poorly** – Frequent updates can lead to inconsistency across clients.	**Better** – Multiple local updates help smooth out discrepancies.
Computational Load on Clients	**Lower** – Clients perform only one gradient computation per round.	**Higher** – Clients perform multiple local updates before sending results.

FL Algorithms

FedProx (Federated Proximal)

- **Description**: Adds a **proximal term** to the local training objective to prevent drastic weight changes, improving model stability for heterogeneous data.
- **Advantages**: Handles device heterogeneity and non-IID data better than FedAvg.
- **Challenges**: Requires careful tuning of the proximal term hyperparameter.

FL Algorithms

In the **FedProx** algorithm, the local optimization problem is modified by adding a proximal term:

$$\min_w F_i(w) + \frac{\mu}{2}||w - w^t||^2$$

where:

- $F_i(w)$ is the local objective function on client i,
- w^t is the **global model** at round t,
- w is the **local model** being trained on the client,
- μ is a **proximal coefficient** (a hyperparameter that controls the strength of the regularization),
- $||w - w^t||^2$ is the **proximal term**, which penalizes large deviations from the global model.

How It Works in Gradient Descent

During local training, gradient descent in FedProx modifies the weight update rule as:

$$w^{t+1} = w^t - \eta \nabla F_i(w) - \eta\mu(w - w^t)$$

where:

- η is the learning rate,
- $\nabla F_i(w)$ is the gradient of the local loss function,
- $\mu(w - w^t)$ acts as a regularization force pulling w back toward w^t.

FL Algorithms

FedNova (Federated Normalized Averaging)

- **Description**: Normalizes local updates to reduce bias caused by varying local training steps.
- **Advantages**: Improves convergence in heterogeneous settings.
- **Challenges**: Slightly increased computation on the server for normalization.

FedNova

🔷 How FedNova Works:

FedNova modifies the standard **FedAvg** aggregation method by:

- **Normalizing local updates**: Instead of averaging raw gradients, FedNova normalizes updates by the effective number of local steps taken by each client.
- **Weighted aggregation**: The global model update is computed using a weighted sum of local updates, ensuring fairness among clients with different amounts of computation.

Advantages of FedNova:

✅ **Mitigates Client Drift**: Ensures fair contribution from all clients, reducing divergence.

✅ **Handles Heterogeneous Data**: Works better in real-world FL settings with non-IID data.

✅ **Improves Convergence**: Leads to faster and more stable training.

FedNova

Mathematically, if g_i is the local gradient update of client i, and τ_i is the number of local steps taken by client i, the **FedAvg** aggregation is:

$$g = \sum_{i=1}^{N} \frac{n_i}{n} g_i$$

where n_i is the number of data samples on client i.

Instead, **FedNova** normalizes local updates before aggregation:

$$g = \sum_{i=1}^{N} \frac{n_i}{n} \cdot \frac{g_i}{\tau_i}$$

This removes the bias introduced by varying local steps τ_i, leading to a more stable and efficient training process.

FedNova

🔷 Comparison with Other FL Algorithms:

Algorithm	Handles Non-IID Data?	Addresses Client Variability?	Improves Convergence?
FedAvg	❌ No	❌ No	⚠️ Moderate
FedProx	✅ Yes (adds a proximal term)	❌ No	✅ Yes
FedNova	✅ Yes	✅ Yes	✅ Yes

FL Algorithms

Scaffold (Stochastic Controlled Averaging)

- **Description**: Uses control variates to correct local updates and prevent model drift due to non-IID data.
- **Advantages**: More stable training in non-IID settings compared to FedAvg.
- **Challenges**: Requires maintaining control variates, increasing memory overhead.

How SCAFFOLD Works

- SCAFFOLD improves FL training using **control variates** (correction terms) that reduce drift between local and global updates.
- **Key Components:**

1. Client Control Variate (c_i)

Each client maintains a control variate to **adjust local gradients** based on its data distribution.

2. Global Control Variate (c)

The server maintains a global control variate to track **the overall drift** in the network.

3. **Gradient Correction**

- During local training, each client updates its model using:

$$w_i^{t+1} = w_i^t - \eta(\nabla F(w_i^t) - c_i + c)$$

- w_i^t → Local model parameters of client i at round t.
- η → Learning rate.
- $\nabla F(w_i^t)$ → Local gradient of the loss function for client i.
- c_i → Client control variate (correction term to adjust client drift).
- c → Global control variate (average drift correction across all clients).

This correction helps align local updates with the global objective, leading to **faster convergence and more stable training**.

4. **Server Aggregation with Control Updates**

   - The global model is updated using **both client updates and control variates**:

$$w^{t+1} = w^t + \frac{1}{K} \sum_{i=1}^{K} (w_i^{t+1} - w^t)$$

   - Control variates are also averaged and updated to **minimize client drift**.

- w^t → Global model parameters at round t.
- K → Number of participating clients in the current round.
- w_i^{t+1} → Locally updated model parameters of client i.
- w^t → Previous global model parameters.

*By incorporating drift correction, this step helps prevent **overfitting** to dominant client distributions and improves **global model stability**.*

- The term $w_i^{t+1} - w^t$ represents the difference between the local and global model after training.
- By averaging these differences across all clients and applying it to the global model, SCAFFOLD ensures a **balanced** update.

FL Algorithms

FedOpt (Federated Optimization)

- **Description**: Applies server-side optimization techniques (e.g., Adam, momentum-based updates) instead of simple model averaging.
- **Advantages**: Faster convergence and better performance in heterogeneous environments.
- **Challenges**: Requires more computational resources on the server.

FedOpt

FedOpt improves upon FedAvg by introducing **adaptive optimization at the server level** rather than just averaging client updates.

✅ FedOpt General Update Rule

$$w^{t+1} = w^t - \eta_t \cdot v_t$$

Where:

- w^t → Global model parameters at round t.
- η_t → Adaptive learning rate (changes dynamically).
- v_t → Optimized global update computed using **adaptive optimizers** (e.g., Adam, Adagrad, or Yogi).

✅ Key Variants of FedOpt

FedOpt Variant	Optimizer Used on Server	Key Advantage
FedAdam	Adam (Momentum + Adaptive LR)	Fast convergence, reduces variance
FedAdagrad	Adagrad (Per-parameter LR scaling)	Stable updates, suitable for sparse data
FedYogi	Yogi (Improved Adam variant)	Prevents excessive update scaling

---

🔷 Comparison: FedAvg vs. FedOpt

Feature	FedAvg	FedOpt
Server Optimizer	**SGD (Simple Averaging)**	**Adaptive Optimizers (Adam, Adagrad, etc.)**
Convergence Speed	Slow for non-IID data	Faster, better stability
Handles Client Variability	❌ No	✅ Yes
Robust to Non-IID Data	❌ No	✅ Yes

- **When to Use FedOpt?**
- When **FedAvg struggles with slow convergence** on non-IID data.
- When clients have **high update variability** (different local steps, varying compute power).
- When FL needs **faster and more stable training** with adaptive learning rates.

Difference Between Gradients and Weights

Feature	Gradients (∇F)	Weights (w)
Definition	The **rate of change** of the loss function with respect to model parameters.	The actual **parameters** (coefficients) of the model that are learned during training.
Purpose	Indicates **how** to update the model weights to minimize the loss.	Represents the **current state** of the model at a given time step.
Computation	Computed using **backpropagation** (via differentiation).	Updated using gradient descent or other optimization algorithms.
Update Rule	Used to adjust weights as per: $w^{t+1} = w^t - \eta \nabla F(w^t)$	Directly modified during training based on gradients.
Representation	Matrix or vector representing **derivatives** of loss w.r.t. weights.	Matrix or vector storing **learned parameters** of the model.
Usage in FL	In **FedSGD**, clients send **gradients** to the	In **FedAvg**, clients send **weights** to the