Federated Averaging

Outline

- Motivation
- Main Works

1. FederatedAveraging
2. FedAvg with shared data
3. LoAdaBoost FedAvg
4. Asynchronous Federated Optimization

- Conclusion

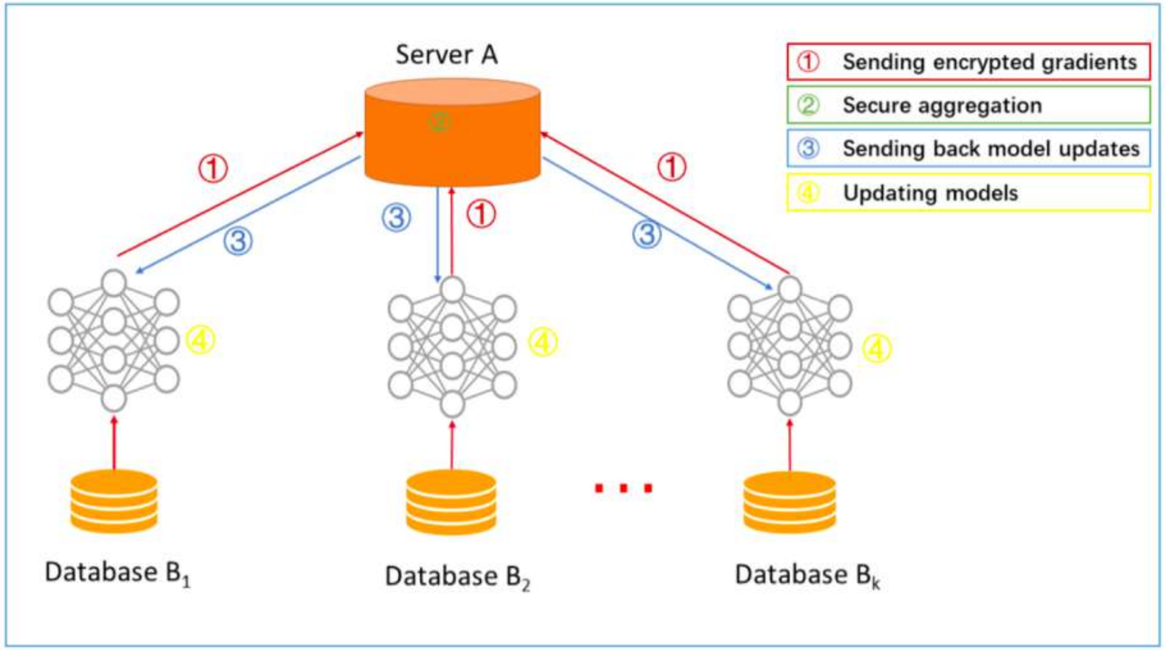
Motivation

key properties different from a typical distributed optimization problem:

- Non-IID
- Unbalanced
- Massively Distributed
- Limited Communication

Training process

Step 1: participants locally compute training gradients, mask a selection of gradients with encryption , differential privacy or secret sharing techniques, and send masked results to server
Step 2: Server performs secure aggregation without learning information about any participant
Step 3: Server send back the aggregated results to participants
Step 4: Participants update their respective model with the decrypted gradients



The FederatedAveraging Algorithm

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

```
Server executes:
  initialize w_0
  for each round t = 1, 2, ... do
    m ← max(C · K, 1)
    S_t ← (random set of m clients)
    for each client k ∈ S_t in parallel do
      w^k_{t+1} ← ClientUpdate(k, w_t)
    w_{t+1} ← Σ_{k=1}^{K} (n_k / n) w^k_{t+1}

ClientUpdate(k, w):   // Run on client k
  𝓑 ← (split 𝒫_k into batches of size B)
  for each local epoch i from 1 to E do
    for batch b ∈ 𝓑 do
      w ← w − η∇ℓ(w; b)
  return w to server
```

- Update (central server)

$$w_{t+1} \leftarrow w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k,$$

- Update (client)

$$\forall k,\ w_{t+1}^{k} \leftarrow w_t - \eta g_k$$

- Three key parameters
  - **C** (the fraction of clients that perform computation on each round)
  - **E** (the number of training passes each client makes over its local dataset on each round)
  - **B** (the local minibatch size used for the client updates)

FedAvg with shared data

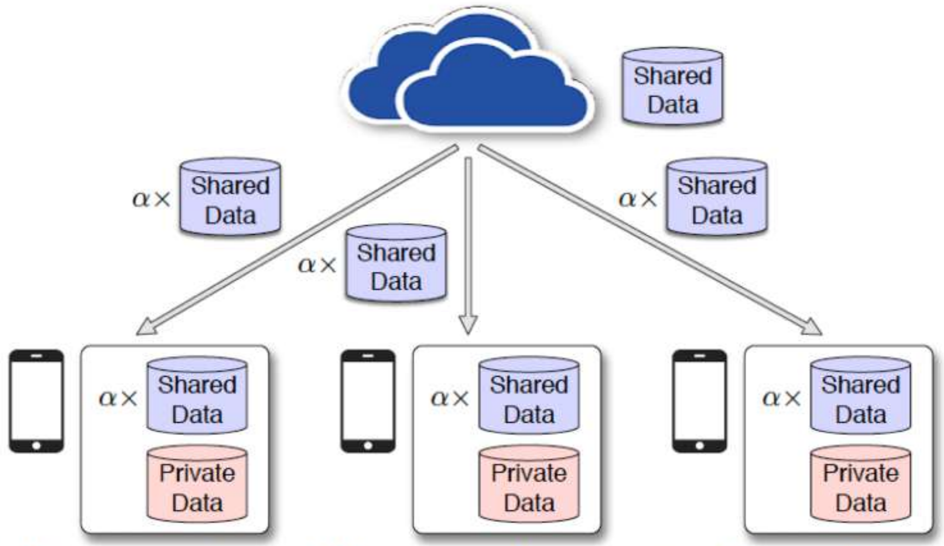


Figure 6: Illustration of the data-sharing strategy.

- A globally shared dataset G
- The warm-up model trained on G
- A random α portion of G are distributed to each client
- Each client is trained on the shared data together with the private data

FedAvg with shared data

$$\beta = \frac{||G||}{||D||} \times 100\%$$

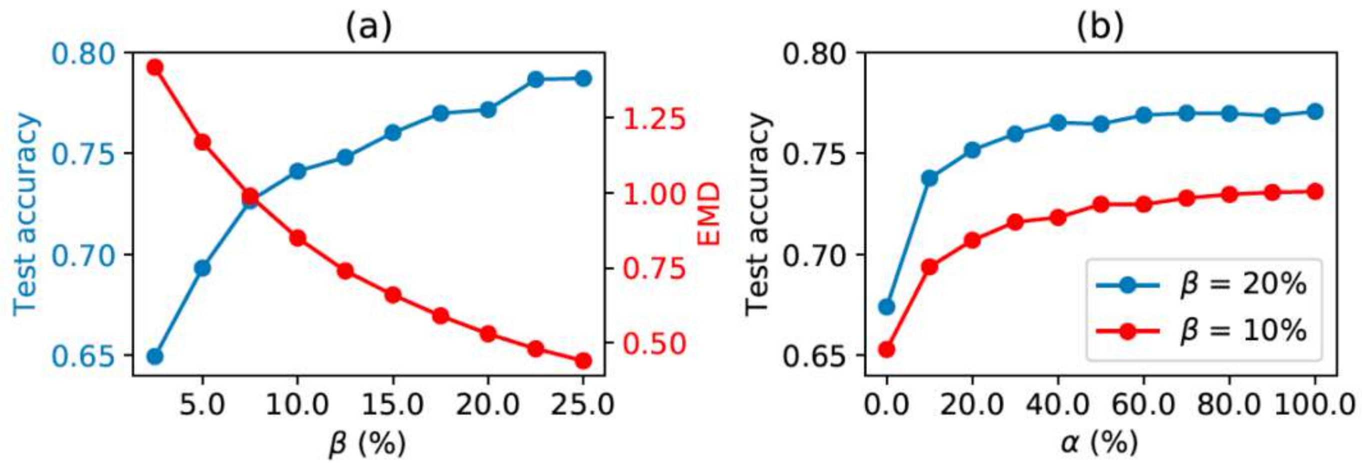


Figure 7: (a) Test accuracy and EMD vs. β (b) Test accuracy vs. the distributed fraction α

- The test accuracy increases as β increases. Even with a lower β = 10%, we can still achieve a test accuracy.

- Only a random portion of G needs to be distributed to each client.

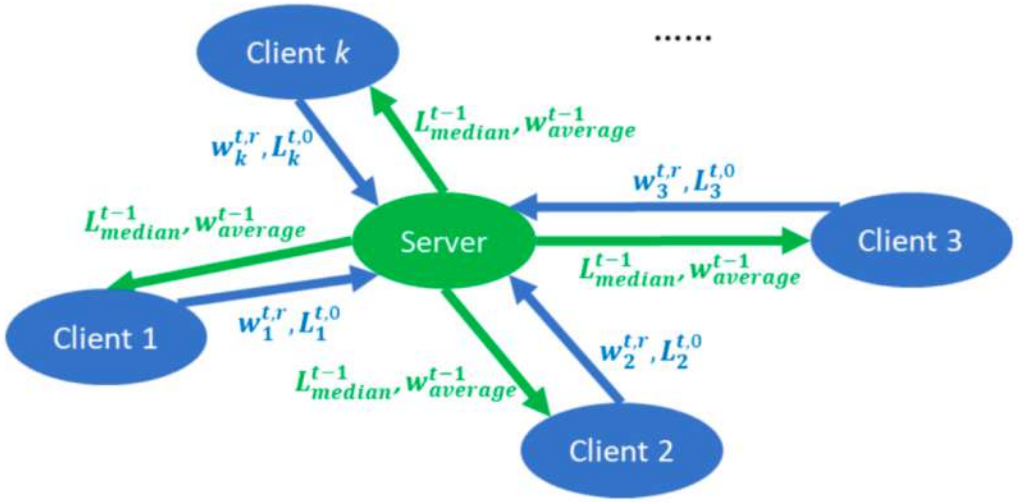
LoAdaBoost FedAvg

- Loss-based Adaptive Boosting FederatedAveraging

- Focus on medical data

- Three issues in federation learning:
    1. the local client-side computation complexity
    2. the communication cost
    3. the test accuracy

LoAdaBoost FedAvg



- Different from FedAvg, the learning process average was performed for E/2 instead of E epochs.
- $\triangle L_k^{t,0} = L_k^{t,0} - L_{median}^{t-1} \leq 0$ finish
- Otherwise, retrain for E/2−1 more epochs, (repeated for E/2−r+1 epochs), stop until
- $\triangle L_k^{r} = L_k^{t,r} - L_{median}^{t-1} \leq 0$ or total epochs reached 3E/2

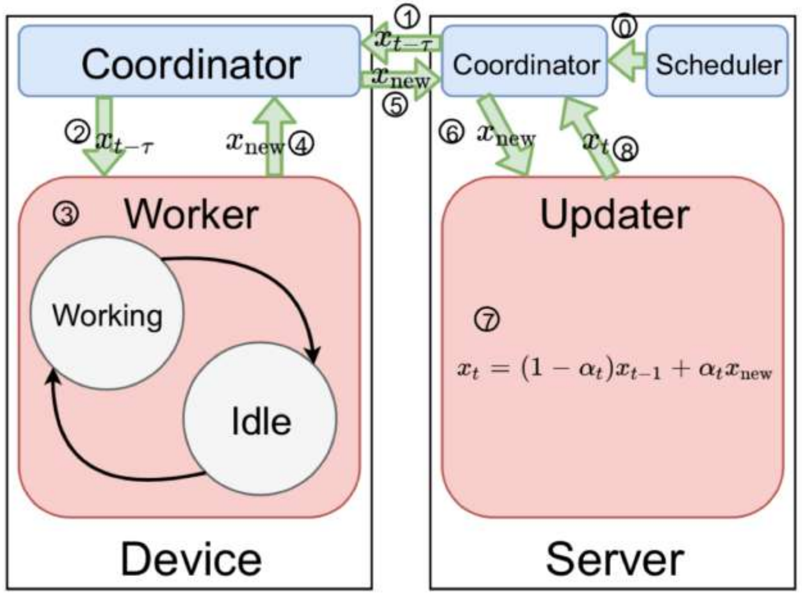
Asynchronous Federated Optimization

- Previous: Synchronous
  1. Too many devices checking in at the same time
  2. Difficult to synchronize the selected devices at the end of each epoch

- Asynchronous federated optimization
  The key idea : use a weighted average to update the global model.

$$x_t = (1 - \alpha)x_{t-1} + \alpha x_{new},$$

Asynchronous Federated Optimization



Algorithm 1 Asynchronous Federated Optimization (FedAsync)

Server Process

Input: $\alpha \in (0, 1)$

Initialize x_0, $\alpha_t \leftarrow \alpha, \forall t \in [T]$

Scheduler Thread

Scheduler periodically triggers some training tasks on some workers, and sends them the latest global model with time stamp

Updater Thread

for all epoch $t \in [T]$ **do**

  Receive the pair (x_{new}, τ) from any worker

  Optional: $\alpha_t \leftarrow \alpha \times s(t - \tau)$, $s(\cdot)$ is a function of the staleness

  $\boxed{x_t = (1 - \alpha_t)x_{t-1} + \alpha_t x_{new}}$

end for

Worker Processes

for all $i \in [n]$ in parallel **do**

  If triggered by the scheduler:

  Receive the pair of the global model and its time stamp (x_t, t) from the server

  $\tau \leftarrow t$, $x^i_{\tau,0} \leftarrow x_t$

  For μ-weakly convex F:

    Define $g_{x_t}(x; z) = f(x; z) + \frac{\rho}{2}\|x - x_t\|^2$, where $\rho > \mu$

  for all local iteration $h \in [H^i_\tau]$ **do**

    Randomly sample $z^i_{\tau,h} \sim \mathcal{D}^i$

$$x^i_{\tau,h} \leftarrow \begin{cases} \text{Option I, for strongly convex } F: \\ x^i_{\tau,h-1} - \gamma \nabla f(x^i_{\tau,h-1}; z^i_{\tau,h}) \\ \text{Option II, for weakly convex } F: \\ x^i_{\tau,h-1} - \gamma \nabla g_{x_t}(x^i_{\tau,h-1}; z^i_{\tau,h}) \end{cases}$$

  end for

  Push $(x^i_{\tau,H^i_\tau}, \tau)$ to the server

end for