Decentralized Federated Learning: Enhancing IFCA for Efficient Clustered Model Training

Jonas Kirch

December 2, 2024

TU Dortmund University - Department of Computer Science

Content

- Problem Formulation
- Decentralized IFCA (DIFCA)
- First Results
- Discussion and Ideas

Problem Formulation

Federated Learning

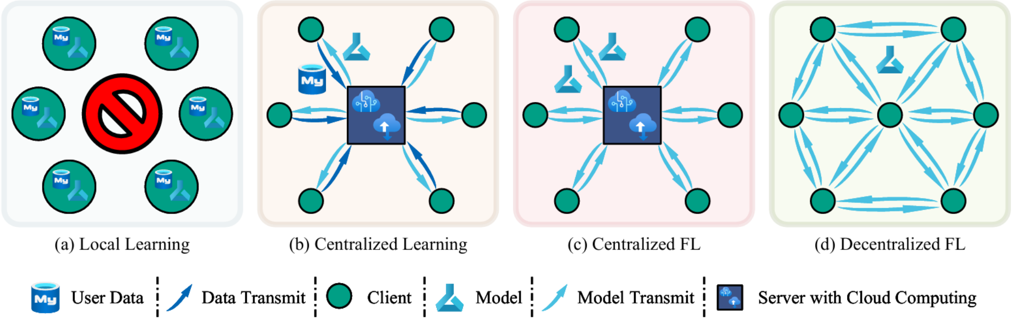

Fig. 2. Illustration of local learning, centralized learning, CFL, and DFL. (a) Clients are trained with local user data only. The clients neither share raw data nor communicate with each other. (b) After clients send the user data packets to the server, the server trains a general model using all the data. The generalized model is then shared with all clients. (c) Clients send the locally trained model parameters to the server. The server aggregates all the local models and then transmits the aggregated global model parameters to all the clients. (d) Clients share their locally trained model with other clients. Subsequent clients then continue to learn, personalize, and adapt the model locally, while also exchanging and propagating the model parameters that possess local knowledge.

Figure 1: Federated Learning [2]

Iterative Federated Clustering Algorithm (IFCA)

Learning Setting:

- center machine, m worker machines
- k different data distributions $\mathcal{D}_1, \ldots, \mathcal{D}_k$
- machines are partitioned into k disjoint clusters $\mathcal{S}_1, \ldots, \mathcal{S}_k$
- Each worker machine $i \in \mathcal{S}_j$ contains n i.i.d. data points $z^{i,1}, \ldots, z^{i,n} \sim \mathcal{D}_j$ with $z^{i,l} = (x^{i,l}, y^{i,l})$

Iterative Federated Clustering Algorithm (IFCA)

$f(\theta; z) : \Theta \to \mathbb{R}$ is the loss function, we minimize the population loss:

$$\boxed{F^j(\theta) := \mathbb{E}_{z \sim \mathcal{D}_j}[f(\theta; z)], \quad j \in [k]}$$

Since we only have finite data, wee specifically try to find solutions θ associated with $Z \subseteq \{z^{i,1}, \ldots, z^{i,n}\}$ that minimize the empirical loss:

$$\boxed{F^j(\theta, Z) = \frac{1}{|Z|} \sum_{z \in Z} f(\theta; z)}$$

Iterative Federated Clustering Algorithm (IFCA)

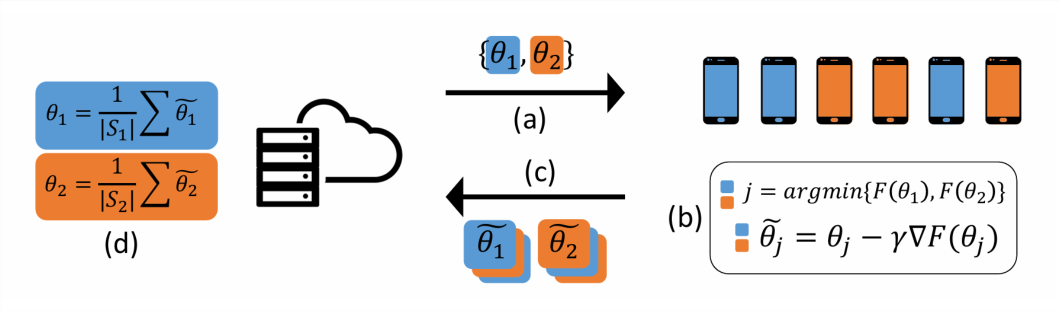

Figure 2: IFCA [3]

Decentralized IFCA (DIFCA)

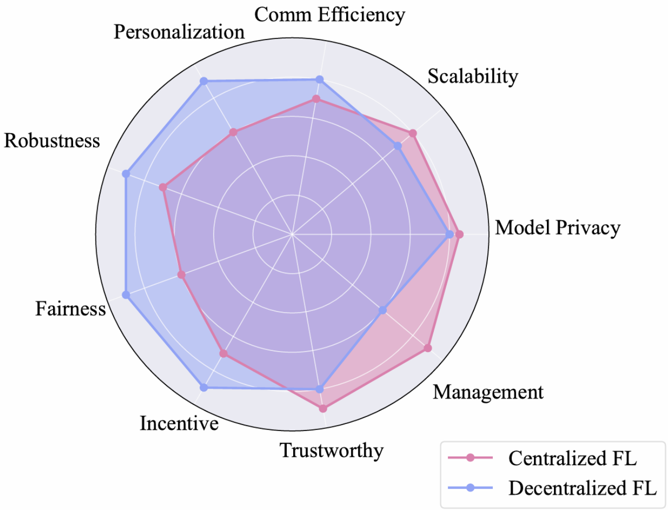

Figure 3: CFL vs DFL

Decentralized IFCA (DIFCA)

- Same learning setting and goal as IFCA
- Decentralized model aggregation
- Variable samples per client
- Confidence based inference

Decentralized IFCA (DIFCA)

Algorithm 1: Decentralized Iterative Federated Clustering Algorithm (DIFCA)

Input: number of clusters k, $j \in [k]$, step size γ, models $\theta_j^{(0)}$, number of local epochs τ, number of training exchange partners R, number of iterations T

server: publish all models $\theta_j^{(0)}$ with list of contact information to all clients
init list L from server with all clients
$M_t \longleftarrow$ subset of worker machines (participating devices)
for $t = 1$ ***to*** $T - 1$ **do**
    for *worker machine* $i \in M_t$ *in parallel* **do**
        run local inference on all models $\theta_j^{(t)}$
        $l_i^{(t)} = \mathrm{argmin}(F_i(\theta_j^{(t)})$
        cluster identity estimate: $\hat{j} = \mathrm{argmin}_{j \in [k]}(F_i(\theta_j^{(t)})$
        broadcast $\hat{j}$
        update list L_i with all cluster identities
        $\tilde{\theta}_i = \mathrm{LocalUpdate}(\theta_{\hat{j}}^{(t)}, \gamma, \tau)$
    for *cluster* $c \in k$ **do**
        server: send one token to random worker machine in c
    for *worker machine* $i \in M_t$ *in parallel* **do**
        $cnt = 0$
        while $cnt < R$ **do**
            peer_exchange(cluster$_i$) ; `// send` $\theta_i^{(t)}, n_i^{(t)}$ `and receive` $\theta_j^{(t)}, n_j^{(t)}$
            $\frac{\theta_i^{(t)} + \theta_j^{(t)}}{2}$
            $cnt{+}{+}$
    for *tokenized worker machine* $i \in M_t$ **do**
        broadcast Model $\theta_{\hat{j}}^{(t)}$
server listens and updates representative models $\theta_j^{(t)}$ after each epoch through broadcast
return $\theta_j^{(T)}$

Figure 4: DIFCA

Final Experiment Setup

Per Cluster:

- 60000 train data, 10000 test data
- $m = 1200$ clients
- $p = 4$ Clusters or Distributions (Rotated Mnist)
- Number of Samples per client random with $min = 1$ and $max = \lfloor \frac{num\_data}{2 \cdot m\_per\_cluster} \rfloor$
- Local Training Epochs per Round $\tau = 10$ and Step Size $\gamma = 0.1$

First Results

First Results

DIFCA vs IFCA Accuracies				
Epochs	DIFCA train	IFCA train	DIFCA test	IFCA test
10	0.696	0.7612	0.6617	0.7131
20	0.771	0.8058	0.7399	0.764
30	0.8097	0.8272	0.7769	0.7905
40	0.8079	0.8410	0.7756	0.8064
50	0.8166	0.8529	0.784	0.8194
60	0.8162	0.8622	0.7864	0.8298
70	0.8336	0.8708	0.7993	0.8387
80	0.8629	0.8787	0.8274	0.8473
90	0.8562	0.8866	0.8209	0.8544
100	**0.8695**	**0.8933**	**0.8356**	**0.861**

Table 1: First Experiments with DIFCA (and IFCA) on 48 worker nodes, k = 4, 15000 train data, 2500 test data & different numbers of data for each client

Discussion and Ideas

TODOs

- How to estimate k?
  - Something like OPTICS reachability plot after -1^{st} round?
- Determine number of exchanges per epoch (hyperparameter?)
  - How does it scale with high numbers of exchanges e.g $num\_data - 1$ exchanges
- How to determine representatives for clusters?
- Test for different data sizes for each client