Hybrid FL:
Algorithms and Implementation

Xinwei Zhang*  Wotao Yin†  Mingyi Hong*  Tianyi Chen#

*University of Minnesota
† UCLA
# RPI

NeurIPS-SpicyFL 2020
February 13, 2021
Introduction

- Horizontal FL & Vertical FL
- Limitations of HFL & VFL
- Hybrid FL - Setting & Challenge
Horizontal & Vertical FL

Horizontal FL

- Each client: full feature, full model, partial data id
- Application
  - Google Keyboard [1]
  - Image classification [2]
- Algorithm
  - FedAvg [3], Scaffold [4], …

Horizontal & Vertical FL

Vertical FL

- Each client: full data id, partial features; partial model
- Training done at the server (has access to data)
- Application
- Algorithm
  - FedBCD [5], VAFL [6]

Hybrid FL

- Partial feature & partial data id
- Supermarkets:
  - Client: Walmart, Amazon, Ebay, Targets
  - Sample index: costumers
  - Feature: shopping records on different categories of products
  - Goal: jointly predict customers’ behavior
- Not covered by HFL & VFL
Hybrid FL: Setting

- Local Model
- Server
- Global Model

- Client 1
  - Data 1
  - Data 2
  - Partial Feature

- Client 2
  - Data 2
  - Data 3
  - Partial and overlapping Data

- Client 3
  - Data 1
  - Data 3

Data 1, Data 2, and Data 3 represent different types of data that are distributed among clients and processed using local models. The server aggregates the results from these local models to create a global model.
Hybrid FL: Challenges

- Limited Data Sharing
- Partial and overlapping Data

Local Model

Server

Global Model

Client 1

Data 1

Data 2

Client 2

Data 2

Data 3

Client 3

Data 1

Data 3

Limited Data Sharing
Our Contribution

- Propose a new and flexible Hybrid FL problem formulation.
- Develop an efficient algorithm for Hybrid FL
  1. enables knowledge sharing
  2. maintains data locality
  3. without requiring sample synchronization
- Numerical results on the real data set show competitive performance to centralized training.
Hybrid FL Model Design

- System Design
- Problem Formulation
- Regularizer Design
System Structure

• Recall:
  – Challenge 1: no data sharing
  – Challenge 2: build both local and global models
  – Challenge 3: sample synchronization

• Our solution:
  – Local training + Model merging
System Model

Server: model merging
Client: local training
Different model size, cannot do averaging!

Global Model

Client 1: Local Model 1
$D_1 = [1, 2, ...]$

Client 2: Local Model 2
$D_2 = [1, 3, ...]$

Client M: Local Model M
$D_M = [2, 5, ...]$
How to Merge Models?

- Global models and local models have different sizes
- However, they have overlapping functionality blocks
- Our Strategy: Identify and Match the blocks with similar functionalities between client and server
How to Identify Similar Functionalities?

- **Assumption**: Same feature should be processed by the same feature extractor
A Refined System Model

Server

θ₀

θ₀,1  θ₀,2  ...  θ₀,D

Client 1

w₁

θ₁,1  θ₁,2  ...  

x₁  x[1]  x[2]  ...  

𝒟₁ = [1, 2, ... ]

Client 2

w₂

θ₂,1  θ₂,3  ...  

x₂  x[1]  x[3]  ...  

𝒟₂ = [1, 3, ... ]

Client M

wₘ

θₘ,2  θₘ,5  ...  

xₘ  x[2]  x[5]  ...  

𝒟ₘ = [2, 5, ... ]
How to Match?

- Extractors of the same feature should be similar: \( \theta_{m,d} \approx \theta_{0,d}, d = 1, \ldots, D \)
- The global classifier is an (linear [7]) ensemble of the local ones; needs to find the best matching matrix \( \Pi_m \)

Problem Formulation

\[
\begin{align*}
\min_{\{\theta_{m,d}\}, w_m} & \quad \frac{1}{MN} \sum_{m=1}^{M} \sum_{n \in \mathcal{N}_m} \ell_m (h_m(\{\theta_{m,d}\}; x_m^n; w_m; y^n) + r(\{\theta_{m,d}\}, w_m; \theta_0, \{\theta_{0,d}\}) \\
\end{align*}
\]

- Classification part optimizes local models by the local agents;
- Regularizers part matches extractor and classifiers
  - Enable knowledge sharing
- Optimization variables:
  - extractors \(\{\theta_{m,d}\}_{d \in D_m}\)
  - matching patterns \(\Pi_m\)
  - classifiers \(\theta_0, w_m\)
Hybrid FL Algorithm Design

- Problem Separation
- HyFEM Algorithm
- Algorithm Analysis
Problem at Client

- Local problem at each client $m$:

$$
\min_{\theta_m, w_m} f_m(\theta_m, w_m) = \sum_{n \in \mathcal{N}_m} \ell_m(h(\theta_m; x^n_m); w_m; y^n_m) \\
+ \sum_{d \in \mathcal{D}_m} \text{dist}(\theta_{m,d}, \theta_{0,d}) + \mu \text{dist}(\prod_m \theta_0, w_m);
$$

1. Use local data, global model to update local model
Problem at Server

- Global problem at server:

\[
\begin{align*}
\min_{\theta_{0,d}} \sum_{m=1}^{M} \sum_{d \in D_m} \text{dist}(\theta_{m,d}, \theta_{0,d}); \\
\min_{\theta_0, \{\prod_m\}} \sum_{m=1}^{M} \text{dist}(\prod_m \theta_0, w_m), \\
\text{s.t. } \prod_m 1 = 1, m = 1, \ldots, M.
\end{align*}
\]

- No local data is used!
Hybrid Federated Matching Algorithm (HyFEM)

For $t = 0, \ldots, T - 1$ do
  For each client in parallel does
    $\{\theta_{m,d}^r, w_m^r\} = \arg\min f_m(\{\theta_{m,d}\}, w_m)$
  For the server does
    $\theta_{0,d}^{t+1} = \frac{1}{|\{m: d \in D_m\}|} \sum_{m:d \in D_m} \theta_{m,d}^{t,Q}$

Iteratively do
1. Pick client index $m'$
2. $\Pi_{m'} = \arg\min_{\Pi_{m'}} \text{dist}(w_m, \Pi_{m'}\theta_0)$
3. $\theta_0 = \arg\min_{\theta_0} \sum_{m=1}^{M} \text{dist}(w_m, \Pi_m\theta_0)$
Algorithm Analysis

1. Server does NOT have direct access local data
2. Avoids sample synchronization issue
3. Local models deal with local data with partial features
4. Global model deals with the full featured data

Under mild condition on $\ell_m$’s, the algorithm converges by extending the proof of the BCD algorithm [Razaviyayn et al 13].

Numerical Results

- Experiment Design
- Parameter Settings
- Results
Experiment Design

- **ModelNet40**
  - 40,000 train + 4000 test data
  - 40 classes
  - 12 views/sample (feature blocks)

- **Model**
  - Feature extractor: CNN layers of ResNet-34
  - Classifier: FC net with 1 hidden layer

- **Training set**
  - Partial views of partial classes / client

- **Testing set**
  - Local: partial views of all classes
  - Global: full views of all classes
## Experiment Settings

| Setting | Total view $|D|$ | # Client $M$ | # Views $|D_m|$ | # Classes | Used Data |
|---------|-----------|-------------|-------------|-----------|-----------|
| 1       | 4         | 4           | 2           | 20        | 75%       |
| 2       | 12        | 8           | 3           | 30        | 100%      |
| 3       | 12        | 8           | 6           | 15        | 88%       |

![Image of a grid chart with data points indicated by numbers and symbols.](image-url)
Numerical Result 1 (4 clients, 2 views/client)

- Good global model performance
- Local feature NOT enough for classification

![Graph showing test accuracy over communication](image)
Numerical Result 2 (4 clients, 3 views/client)

Hybrid FL outperforms centralized training
Numerical Result 3 (8 clients, 6 views/client)

Good local model performance even when client only has half features and 15 classes

Balance between local and global performance by changing $\mu$
Conclusion

1. Local models avoid data sharing and sample synchronization.
2. Model splitting makes use of the shared features.
3. Regularizing the classifiers enables knowledge sharing.
4. Hybrid FL can learn a good global model with incomplete data where HFL and VFL do not apply.
5. Offer an alternative for VFL