Federated Machine Learning : Concept and Applications

HUANG liu

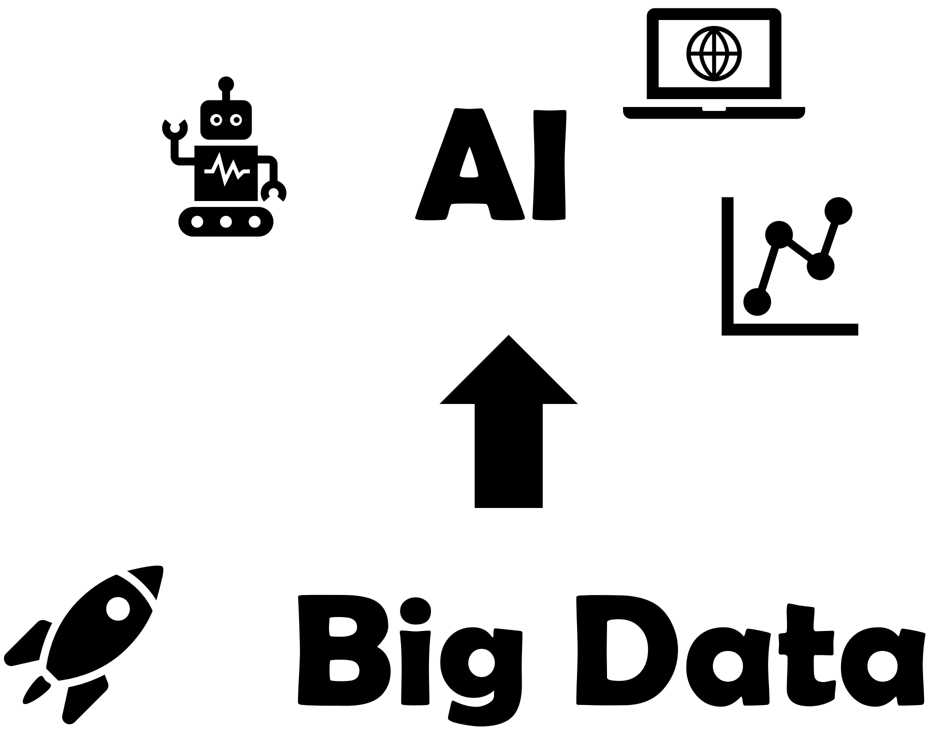
e0575772@u.nus.edu

- Background
- Federated Learning
- Related Work
- Application
- Conclusion

1 Background

Growing Interest in AI

- Alpha GO
- Training dataset: 300,000 games



Real World Situations

Limited amount of data
Low quality

Fuse data

High cost to exchange:
Strict laws
Complicated administrative procedures

Legally

Industry competition
Privacy security

Ensure security

Major challenges

Solution

Data Island

Data Security

→

Federated Learning

2 Federated Learning

Definition

Federated Learning is a machine learning setting where the goal is to train a high-quality centralized model while training data remains distributed over a large number of clients each with unreliable and relatively slow network connections. [1]

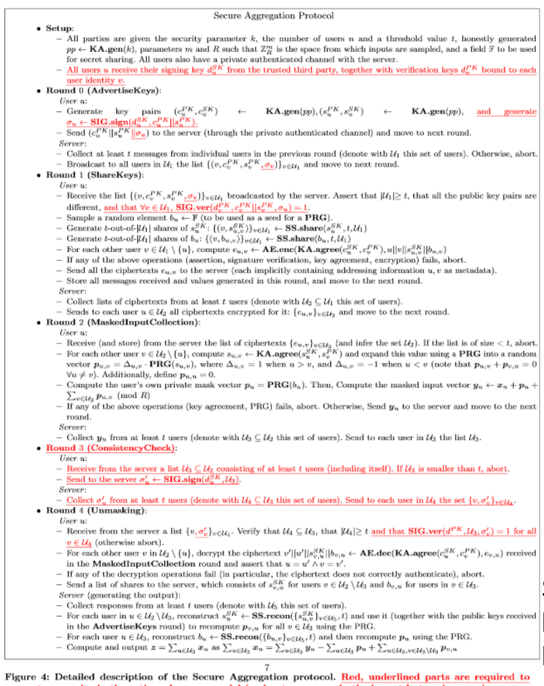
[1] Federated Learning: Strategies for Improving Communication Efficiency. CoRRabs/1610.05492(2016). arXiv:1610.05492 http://arxiv.org/abs/1610.05492

Privacy of Federated Learning

- Secure Multi-party Computation (SMC).
- Differential Privacy
- Homomorphic Encryption

Differentially Private Federated Learning: A Client Level Perspective

$$w_{t+1} = w_t + \underbrace{\frac{1}{m_t}\Big(\overbrace{\sum_{k=0}^{m_t} \triangle w^k / \max\big(1, \frac{\|\triangle w^k\|_2}{S}\big)}^{\text{Sum of updates clipped at } S} + \overbrace{\mathcal{N}(0, \sigma^2 S^2)}^{\text{Noise scaled to } S}\Big)}_{\text{Gaussian mechanism approximating sum of updates}}$$


Secure Aggregation Protocol

- **Setup**:
  - All parties are given the security parameter k, the number of users n and a threshold value t, honestly generated $pp \leftarrow \mathbf{KA.gen}(k)$, parameters m and R such that $\mathbb{Z}_R^m$ is the space from which inputs are sampled, and a field $\mathbb{F}$ to be used for secret sharing. All users also have a private authenticated channel with the server.
  - All users u receive their signing key d_u^{SK} from the trusted third party, together with verification keys d_v^{PK} bound to each user identity v.
- **Round 0 (AdvertiseKeys)**:

  User u:
  - Generate key pairs $(c_u^{PK}, c_u^{SK}) \leftarrow \mathbf{KA.gen}(pp), (s_u^{PK}, s_u^{SK}) \leftarrow \mathbf{KA.gen}(pp)$, and generate $\sigma_u \leftarrow \mathbf{SIG.sign}(d_u^{SK}, c_u^{PK} \| s_u^{PK})$.
  - Send $(c_u^{PK} \| s_u^{PK} \| \sigma_u)$ to the server (through the private authenticated channel) and move to next round.

  Server:
  - Collect at least t messages from individual users in the previous round (denote with $\mathcal{U}_1$ this set of users). Otherwise, abort.
  - Broadcast to all users in $\mathcal{U}_1$ the list $\{(v, c_v^{PK}, s_v^{PK}, \sigma_v)\}_{v \in \mathcal{U}_1}$ and move to next round.
- **Round 1 (ShareKeys)**:

  User u:
  - Receive the list $\{(v, c_v^{PK}, s_v^{PK}, \sigma_v)\}_{v \in \mathcal{U}_1}$ broadcasted by the server. Assert that $|\mathcal{U}_1| \geq t$, that all the public key pairs are different, and that $\forall v \in \mathcal{U}_1$, $\mathbf{SIG.ver}(d_v^{PK}, c_v^{PK} \| s_v^{PK}, \sigma_u) = 1$.
  - Sample a random element $b_u \leftarrow \mathbb{F}$ (to be used as a seed for a **PRG**).
  - Generate t-out-of-$|\mathcal{U}_1|$ shares of s_u^{SK}: $\{(v, s_{u,v}^{SK})\}_{v \in \mathcal{U}_1} \leftarrow \mathbf{SS.share}(s_u^{SK}, t, \mathcal{U}_1)$
  - Generate t-out-of-$|\mathcal{U}_1|$ shares of b_u: $\{(v, b_{u,v})\}_{v \in \mathcal{U}_1} \leftarrow \mathbf{SS.share}(b_u, t, \mathcal{U}_1)$
  - For each other user $v \in \mathcal{U}_1 \setminus \{u\}$, compute $e_{u,v} \leftarrow \mathbf{AE.enc}(\mathbf{KA.agree}(c_u^{SK}, c_v^{PK}), u \| v \| s_{u,v}^{SK} \| b_{u,v})$
  - If any of the above operations (assertion, signature verification, key agreement, encryption) fails, abort.
  - Send all the ciphertexts $e_{u,v}$ to the server (each implicitly containing addressing information u, v as metadata).
  - Store all messages received and values generated in this round, and move to the next round.

  Server:
  - Collect lists of ciphertexts from at least t users (denote with $\mathcal{U}_2 \subseteq \mathcal{U}_1$ this set of users).
  - Sends to each user $u \in \mathcal{U}_2$ all ciphertexts encrypted for it: $\{e_{u,v}\}_{v \in \mathcal{U}_2}$ and move to the next round.
- **Round 2 (MaskedInputCollection)**:

  User u:
  - Receive (and store) from the server the list of ciphertexts $\{e_{u,v}\}_{v \in \mathcal{U}_2}$ (and infer the set $\mathcal{U}_2$). If the list is of size $< t$, abort.
  - For each other user $v \in \mathcal{U}_2 \setminus \{u\}$, compute $s_{u,v} \leftarrow \mathbf{KA.agree}(s_u^{SK}, s_v^{PK})$ and expand this value using a **PRG** into a random vector $\boldsymbol{p}_{u,v} = \Delta_{u,v} \cdot \mathbf{PRG}(s_{u,v})$, where $\Delta_{u,v} = 1$ when $u > v$, and $\Delta_{u,v} = -1$ when $u < v$ (note that $\boldsymbol{p}_{u,v} + \boldsymbol{p}_{v,u} = 0$ $\forall u \neq v$). Additionally, define $\boldsymbol{p}_{u,u} = 0$.
  - Compute the user's own private mask vector $\boldsymbol{p}_u = \mathbf{PRG}(b_u)$. Then, Compute the masked input vector $\boldsymbol{y}_u \leftarrow \boldsymbol{x}_u + \boldsymbol{p}_u + \sum_{v \in \mathcal{U}_2} \boldsymbol{p}_{u,v} \pmod{R}$
  - If any of the above operations (key agreement, PRG) fails, abort. Otherwise, Send $\boldsymbol{y}_u$ to the server and move to the next round.

  Server:
  - Collect $\boldsymbol{y}_u$ from at least t users (denote with $\mathcal{U}_3 \subseteq \mathcal{U}_2$ this set of users). Send to each user in $\mathcal{U}_3$ the list $\mathcal{U}_3$.
- **Round 3 (ConsistencyCheck)**:

  User u:
  - Receive from the server a list $\mathcal{U}_3 \subseteq \mathcal{U}_2$ consisting of at least t users (including itself). If $\mathcal{U}_3$ is smaller than t, abort.
  - Send to the server $\sigma'_u \leftarrow \mathbf{SIG.sign}(d_u^{SK}, \mathcal{U}_3)$.

  Server:
  - Collect σ'_u from at least t users (denote with $\mathcal{U}_4 \subseteq \mathcal{U}_3$ this set of users). Send to each user in $\mathcal{U}_4$ the set $\{v, \sigma'_v\}_{v \in \mathcal{U}_4}$.
- **Round 4 (Unmasking)**:

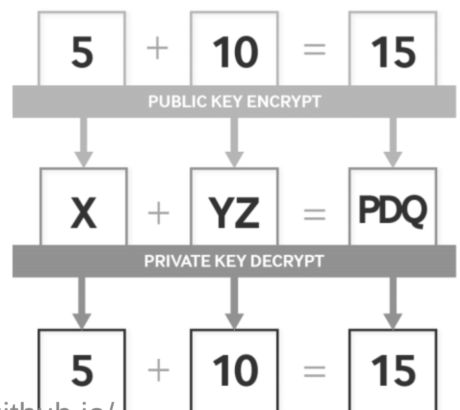
  User u:
  - Receive from the server a list $\{v, \sigma'_v\}_{v \in \mathcal{U}_4}$. Verify that $\mathcal{U}_4 \subseteq \mathcal{U}_3$, that $|\mathcal{U}_4| \geq t$ and that $\mathbf{SIG.ver}(d^{PK}, \mathcal{U}_3, \sigma'_v) = 1$ for all $v \in \mathcal{U}_4$ (otherwise abort).
  - For each other user v in $\mathcal{U}_2 \setminus \{u\}$, decrypt the ciphertext $v' \| u' \| s_{v,u}^{SK} \| b_{v,u} \leftarrow \mathbf{AE.dec}(\mathbf{KA.agree}(c_u^{SK}, c_v^{PK}), e_{v,u})$ received in the **MaskedInputCollection** round and assert that $u = u' \wedge v = v'$.
  - If any of the decryption operations fail (in particular, the ciphertext does not correctly authenticate), abort.
  - Send a list of shares to the server, which consists of $s_{v,u}^{SK}$ for users $v \in \mathcal{U}_2 \setminus \mathcal{U}_3$ and $b_{v,u}$ for users in $v \in \mathcal{U}_3$.

  Server (generating the output):
  - Collect responses from at least t users (denote with $\mathcal{U}_5$ this set of users).
  - For each user in $u \in \mathcal{U}_2 \setminus \mathcal{U}_3$, reconstruct $s_u^{SK} \leftarrow \mathbf{SS.recon}(\{s_{u,v}^{SK}\}_{v \in \mathcal{U}_5}, t)$ and use it (together with the public keys received in the **AdvertiseKeys** round) to recompute $\boldsymbol{p}_{v,u}$ for all $v \in \mathcal{U}_3$ using the PRG.
  - For each user $u \in \mathcal{U}_3$, reconstruct $b_u \leftarrow \mathbf{SS.recon}(\{b_{u,v}\}_{v \in \mathcal{U}_5}, t)$ and then recompute $\boldsymbol{p}_u$ using the PRG.
  - Compute and output $\boldsymbol{z} = \sum_{u \in \mathcal{U}_3} \boldsymbol{x}_u$ as $\sum_{u \in \mathcal{U}_3} \boldsymbol{x}_u = \sum_{u \in \mathcal{U}_3} \boldsymbol{y}_u - \sum_{u \in \mathcal{U}_3} \boldsymbol{p}_u + \sum_{u \in \mathcal{U}_3, v \in \mathcal{U}_2 \setminus \mathcal{U}_3} \boldsymbol{p}_{v,u}$

7

Figure 4: Detailed description of the Secure Aggregation protocol. **Red, underlined parts are required to guarantee security in the active-adversary model (and not necessary in the honest-but-curious one).**

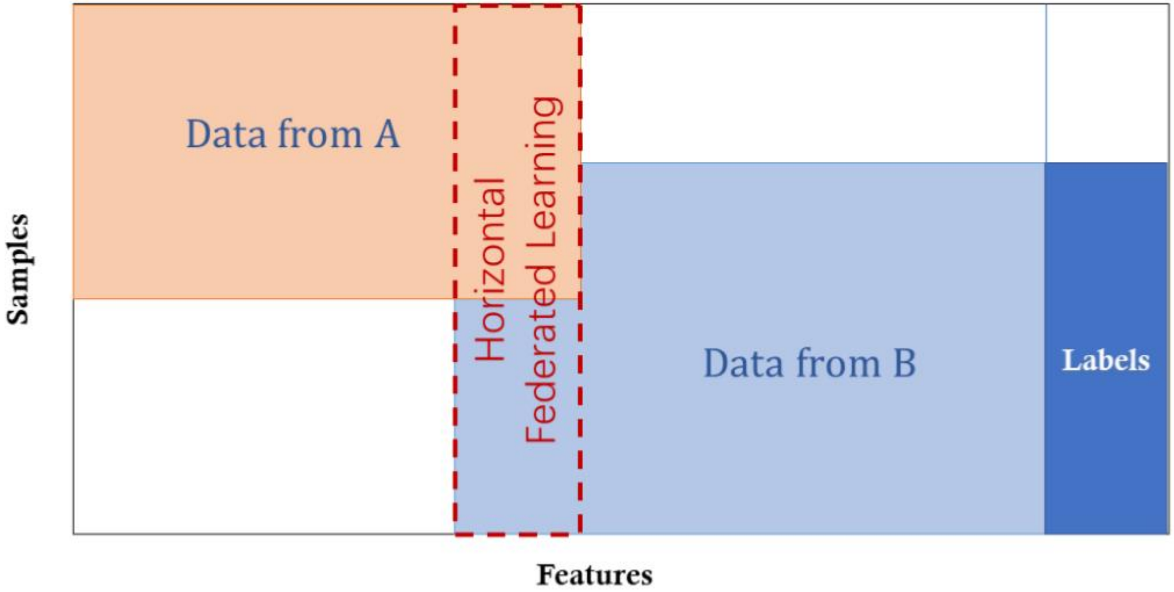
SMPAI: Secure Multi-Party Computation for Federated Learning



https://images.app.goo.gl/pCjC7bf5FzRKaDsz8

Categorization

- Horizontal Federated Learning

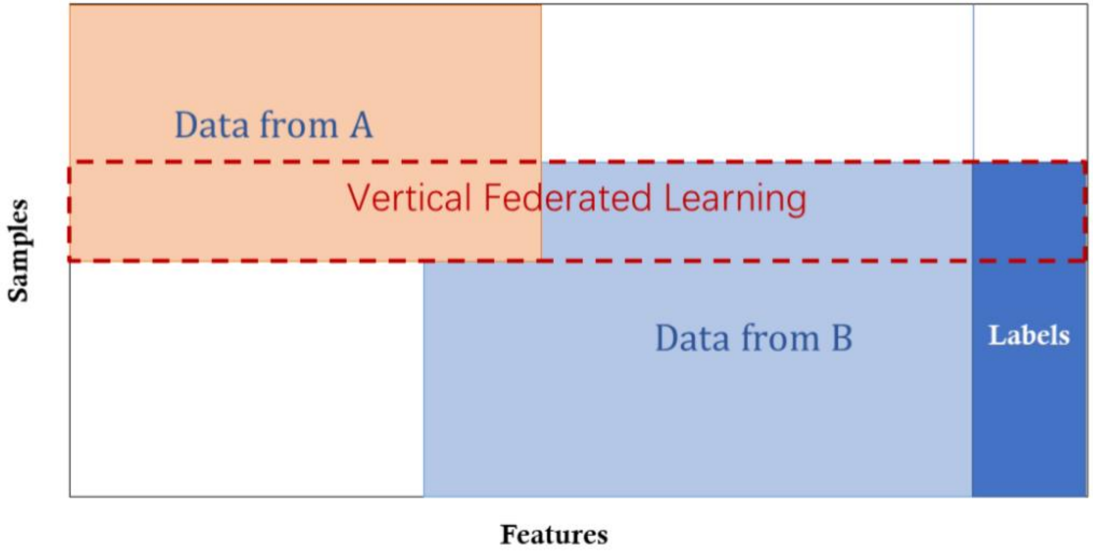


Similar Features
Various Samples

Eg: two banks in different cities

Categorization

- Horizontal Federated Learning
- Vertical Federated Learning

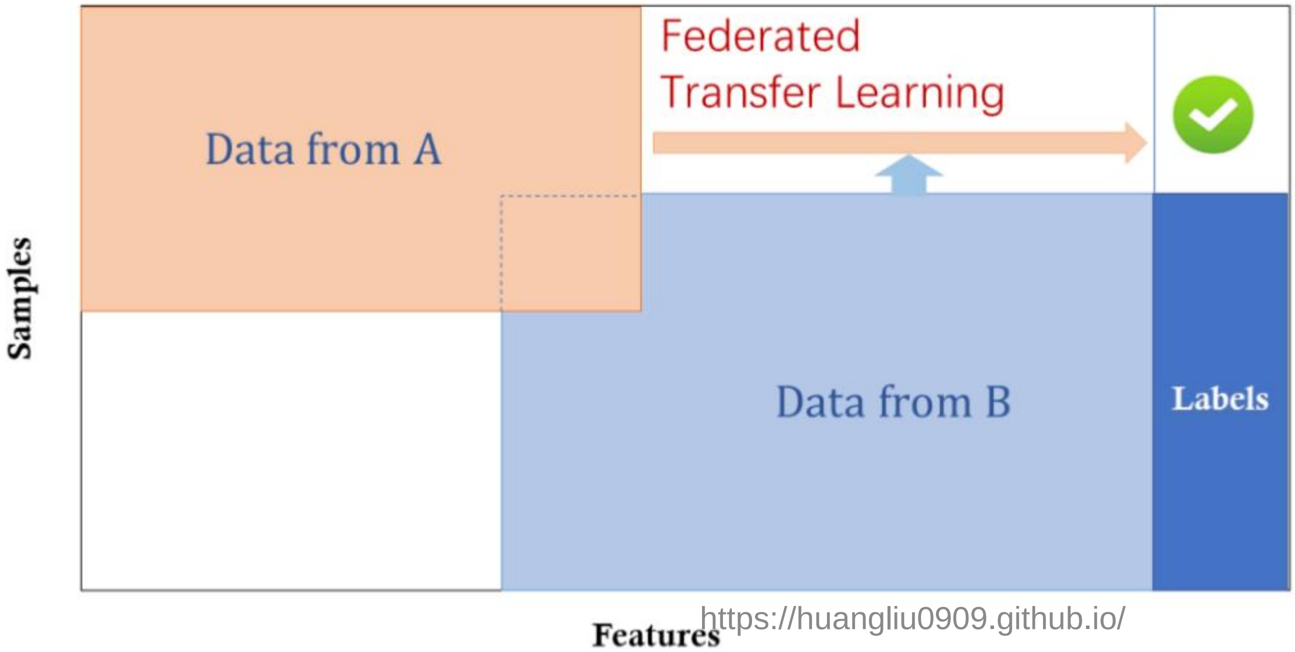


Similar Samples
Various Features

Eg: bank and shopping mall in the same city

Categorization

- Horizontal Federated Learning
- Vertical Federated Learning
- Federated Transfer Learning

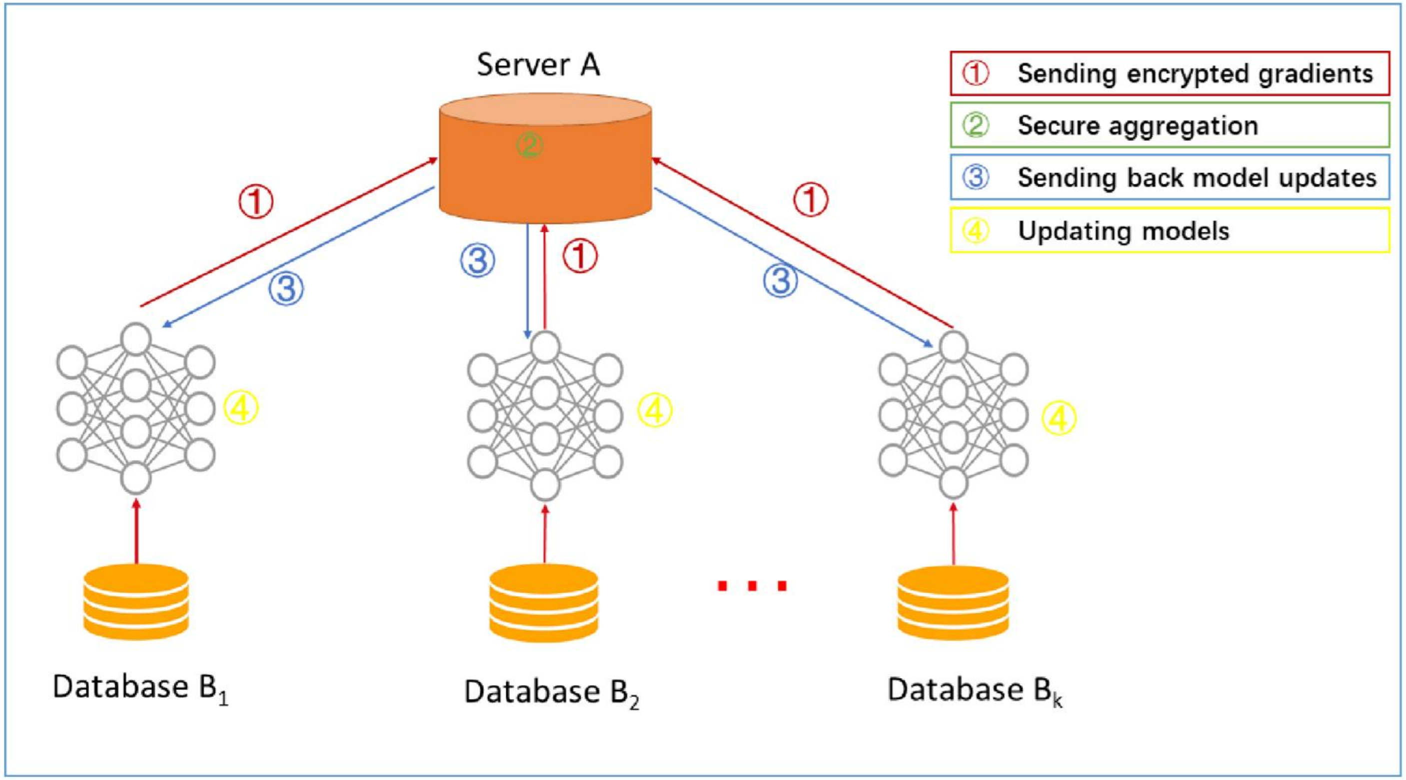


Various Samples
Various Features

Eg: bank in China and shopping mall in Singapore

“Similarity”

Architecture of HFL

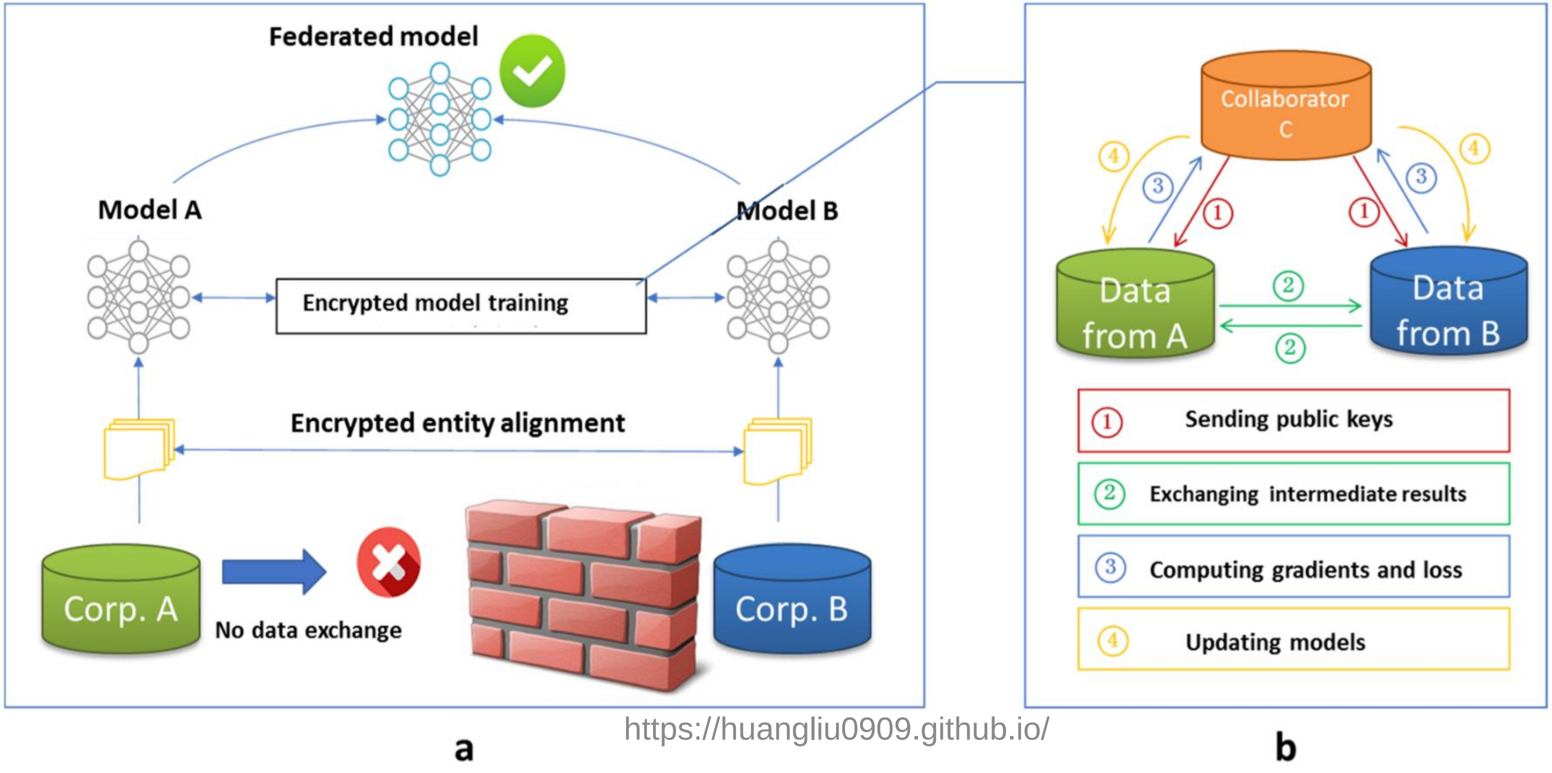


No information leakage between any parties.
Independent from specific ML algorithms
All participants will share the final model parameters.

Architecture of VFL

Suppose A has training data Xa while B has training data Xb and labels y. We want to model how Xa and Xb jointly influence the value of label.

Since A and B cannot exchange data directly, we need a third party, C to help with the model training.



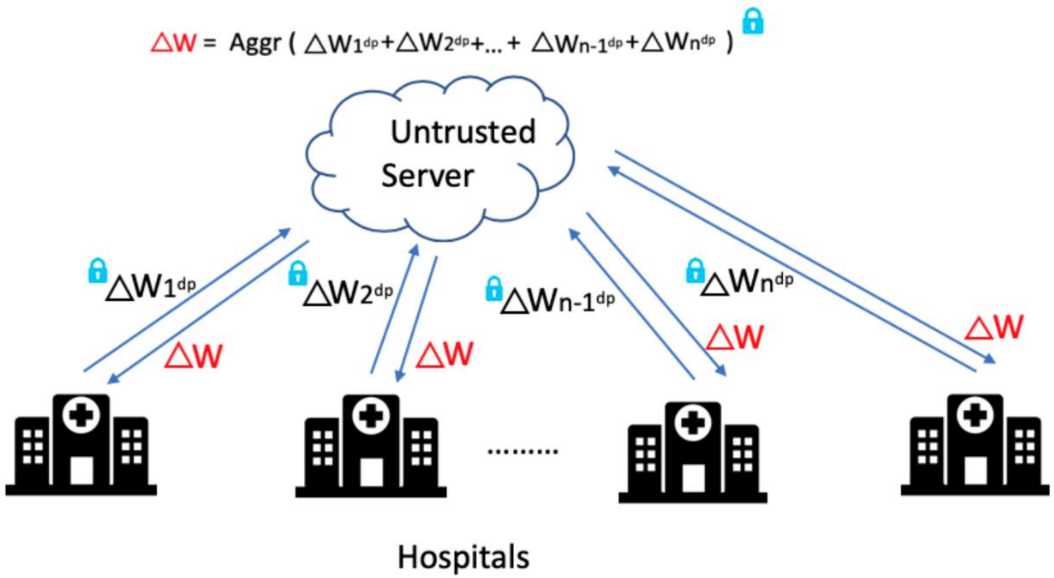
Architecture of FTL

- The same architecture as VFL
- Differ in detail when trying to find the common representation among the parties
- Incentive mechanism: after the model is built, the local model's performance depends on how much this party contribute to the whole federated system.

3 Related Work

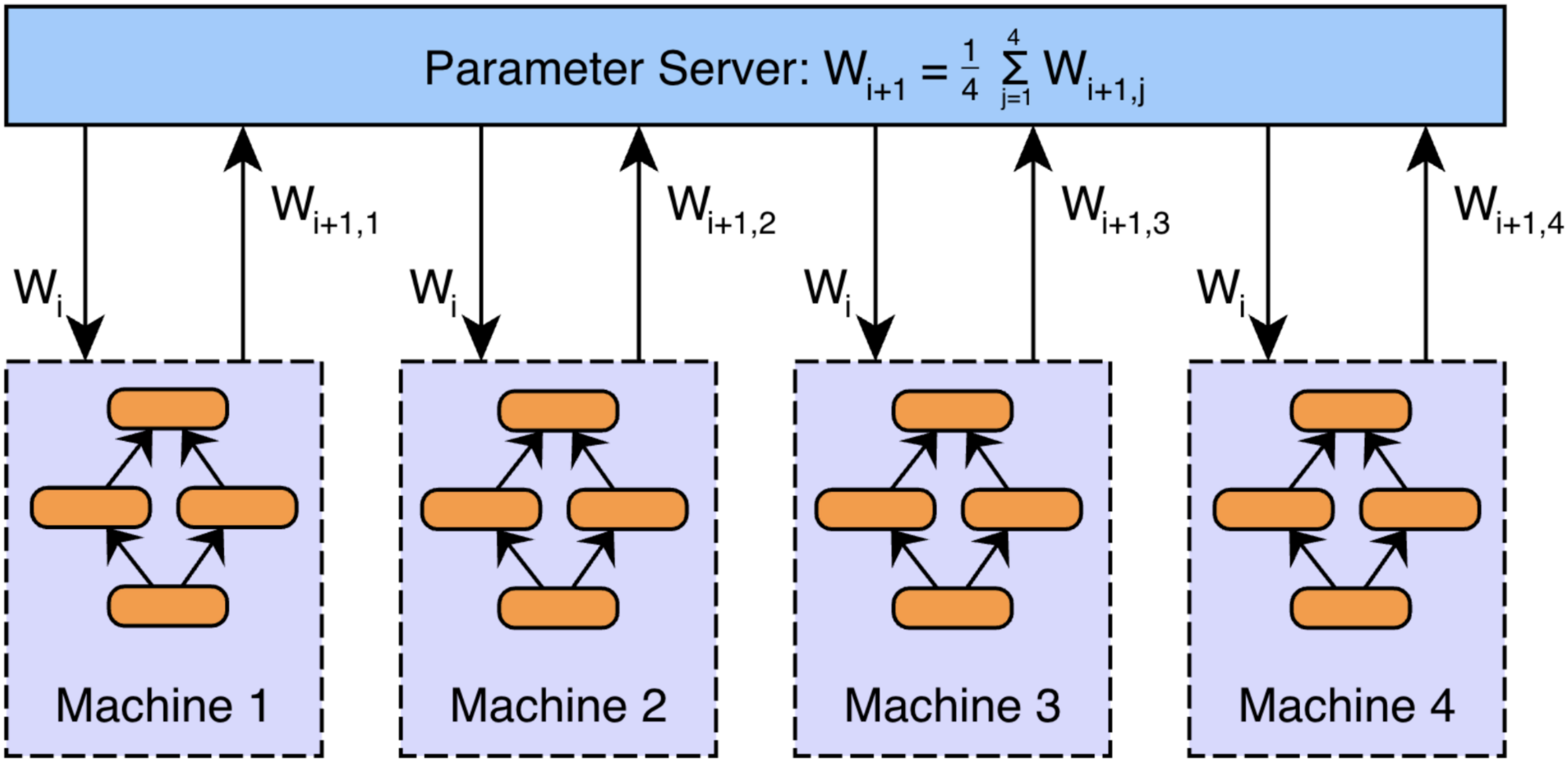
Privacy-preserving ML

Federated learning can be considered as privacy-preserving decentralized collaborative machine learning.
Most of the privacy protection techniques using in privacy-preserving machine learning can be applied in Federated learning



https://www.csail.mit.edu/news/preserving-privacy-machine-learning-models

Distributed Machine Learning



https://code-it.ro/an-introductory-guide-on-distributed-training-of-neural-networks/

Non-IID local data

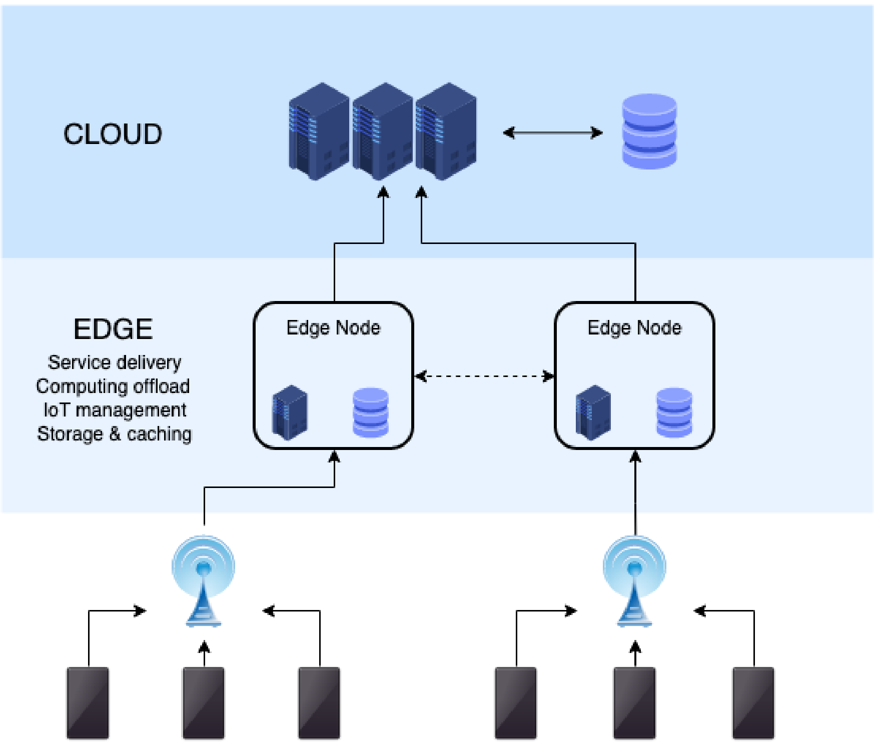
Distributed Machine Learning

- DML: the parameter server allocates data on distributed working nodes and computes model parameters in a scheduled way.

- FL: each working node i.e. data holder, can independently decide when and how to join federated learning.

  Focus on privacy protection

Edge Computing

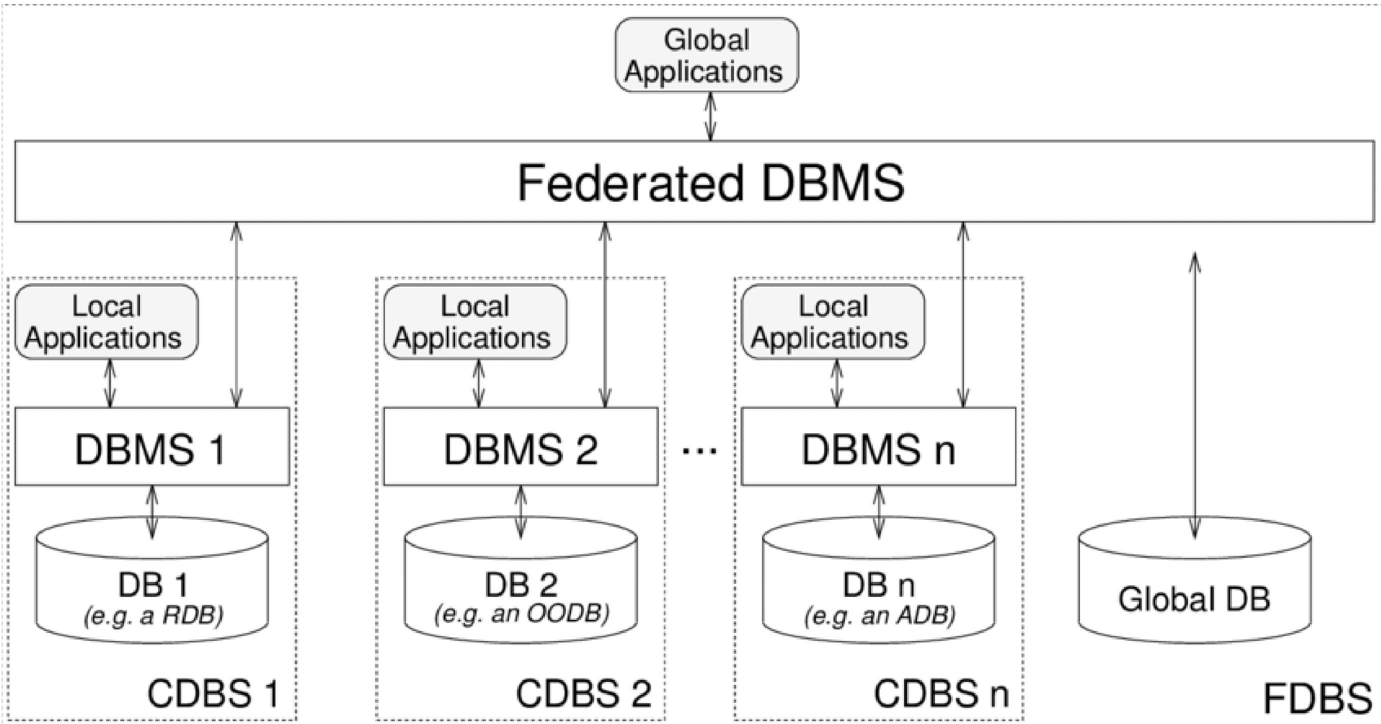


https://images.app.goo.gl/1fNMc1QT1Lw8jpcQ7

Federated learning can provide protocols of implementation details for edge computing, can work as an operation system for edge computing.

Federated Database Systems

- Systems that integrate multiple database units and manage the integrated system as a whole



https://images.app.goo.gl/iiqN8SEtWUxxmmWC9

Federated Database Systems

- VS Distributed database system

The data in each database unit is heterogeneous.

- VS Federated learning

Similar in terms of type and storage in data.

Focus on basic operations of data rather than training a machine learning model.

No privacy protection.

4 Application

Smart Retail

Goal: provide customers with personized services, such as product recommendation and sales service

purchasing power

Bank Saving

user's preference

Social Media

information of product

e-shops

Smart Retail

Problem: data are scattered and heterogeneous

Solution: federated learning & transfer learning

> "Heterogeneous data are any data with high variability of data types and formats. They are possibly ambiguous and low quality due to missing values, high data redundancy, and untruthfulness. It is difficult to integrate heterogeneous data to meet the business information demands."
>
> Wang, L. (2017). Heterogeneous Data and Big Data Analytics. *Automatic Control and Information Sciences*, *3*(1), 8-15.

✓ cross-enterprise

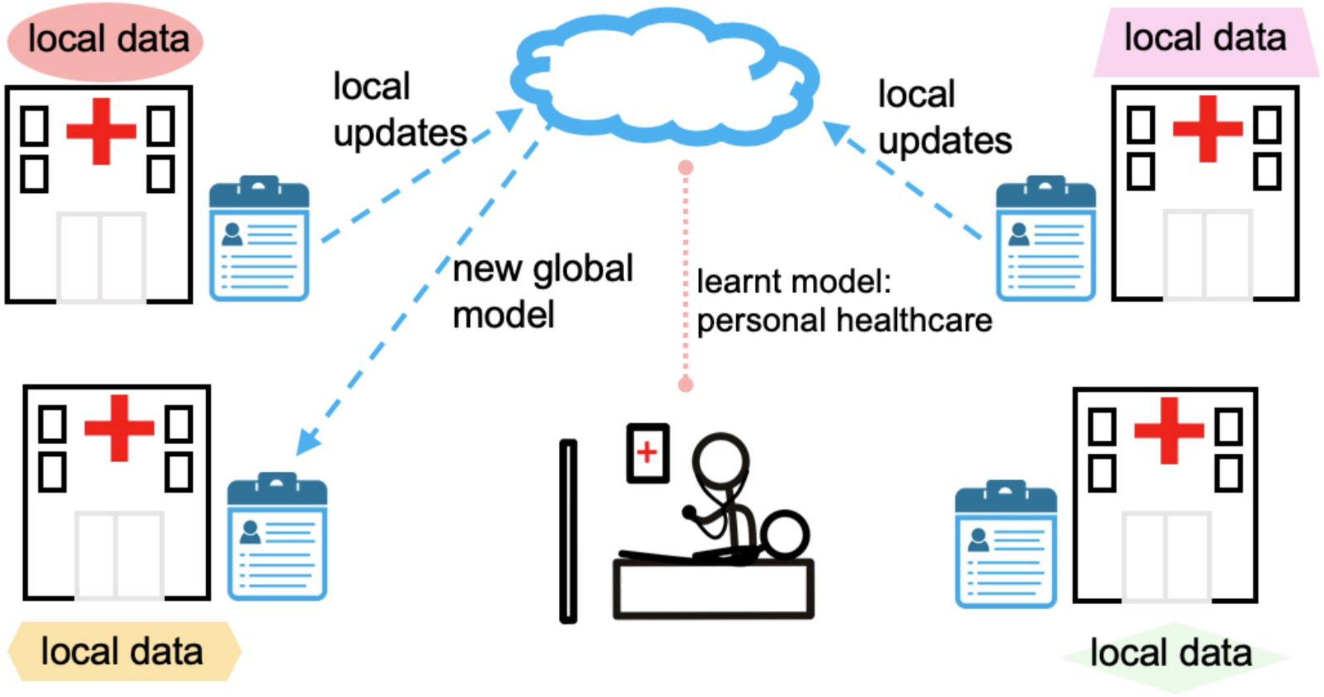
✓ cross-data

✓ cross-domain

Smart Healthcare

Problem: data island & data security

Solution: horizontal federated learning



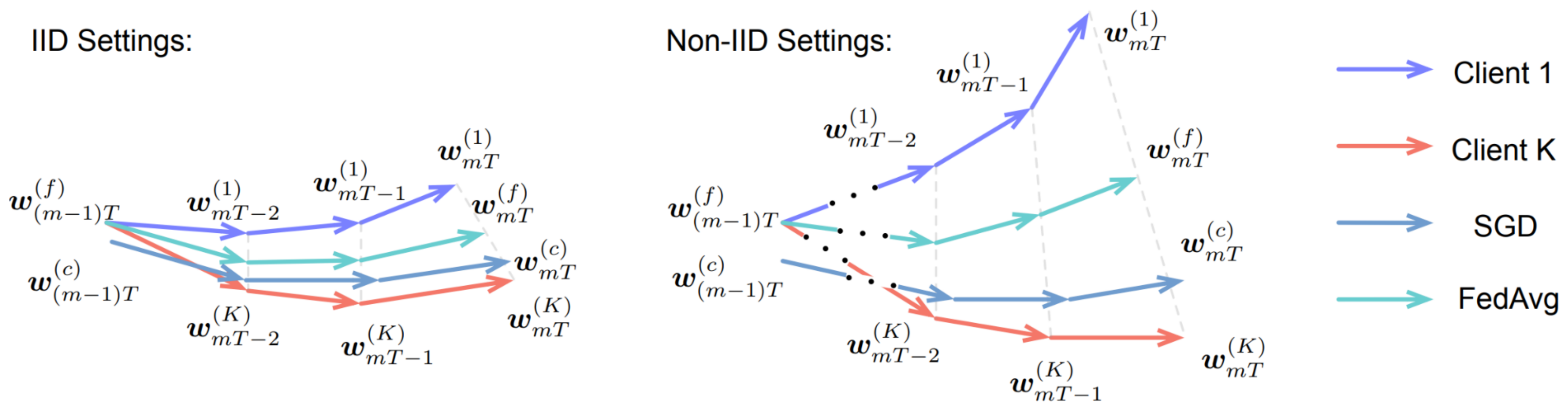
5 Conclusion

“It is expected that in the near future, federated learning would break the barriers between industries and establish a community where data and knowledge could be shared together with safety, and the benefits would be fairly distributed according to the contribution of each participant.”

My insights

- Non-IID data: adaptive optimizer

In the computing of each step, the learning rate is modified according to the historical gradient. The main advantage is to individually learn from the local distribution.



YueZhao,MengLi,LiangzhenLai,NaveenSuda,DamonCivin,andVikasChandra.2018. Federated Learning with Non-IID Data

My insights

- Asynchronous Federated Learning

All the framework above assume little delay for each nodes' model transferring to the server, i.e. the server has to collect all gradients before updating global model.

What if server wait for too much time for collecting all gradients in one round?

Every time the server receives a model from A, the server updates the global model and immediately sends the new model back to A. So there're less time waste in one round and more efficiency.

THANKS