论文分享

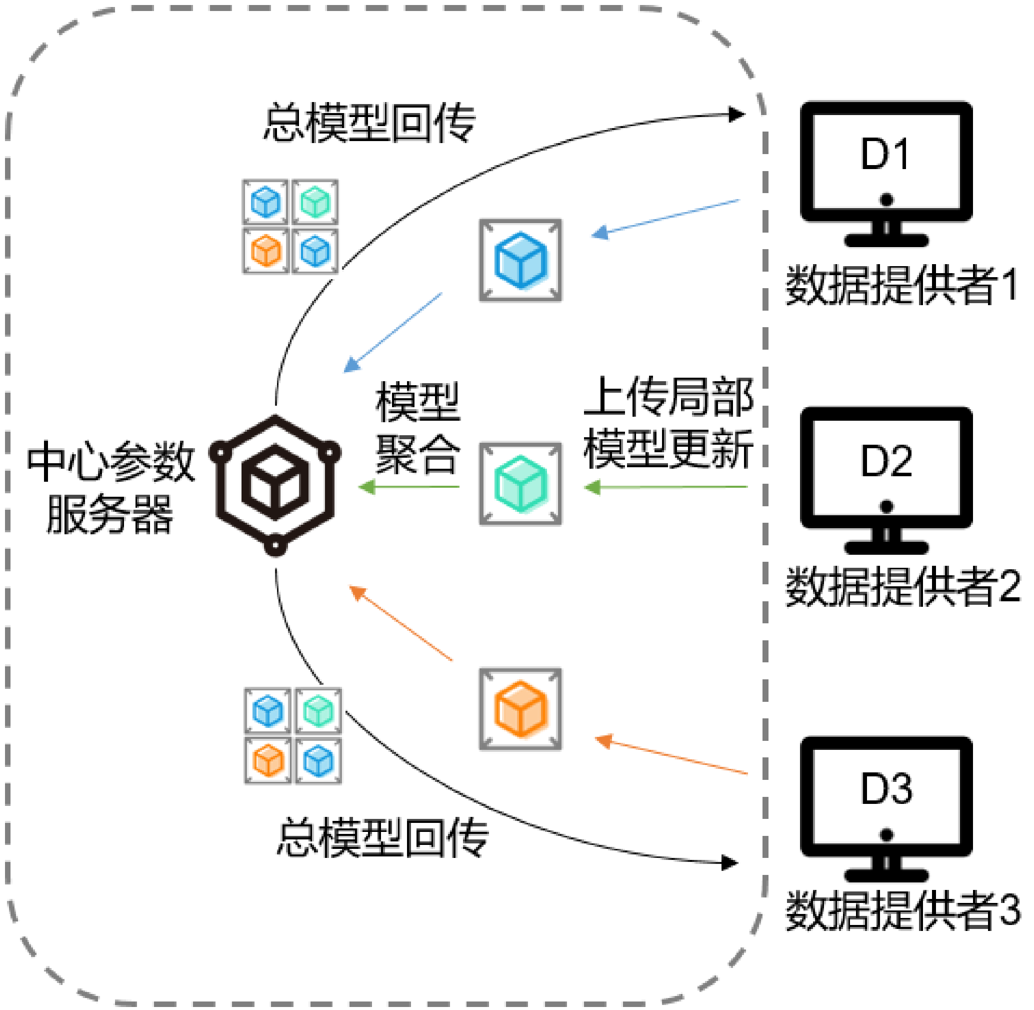
浙江大学
计算机科学与技术学院
分享人：张凤达
2019年11月2日

Federated Learning
(联邦学习)

一、背景
- 1） AI 深度学习
- 2） 《GDPR》《CCPA》
- 3） 数据孤岛

二、概念
- 1） 本质
- 2） 目标
- 3） 挑战

三、进展
- 1） 首次提出
- 2） 学术研究
- 3） 业界应用



Paper-1
Active Federated Learning
（主动联邦学习）

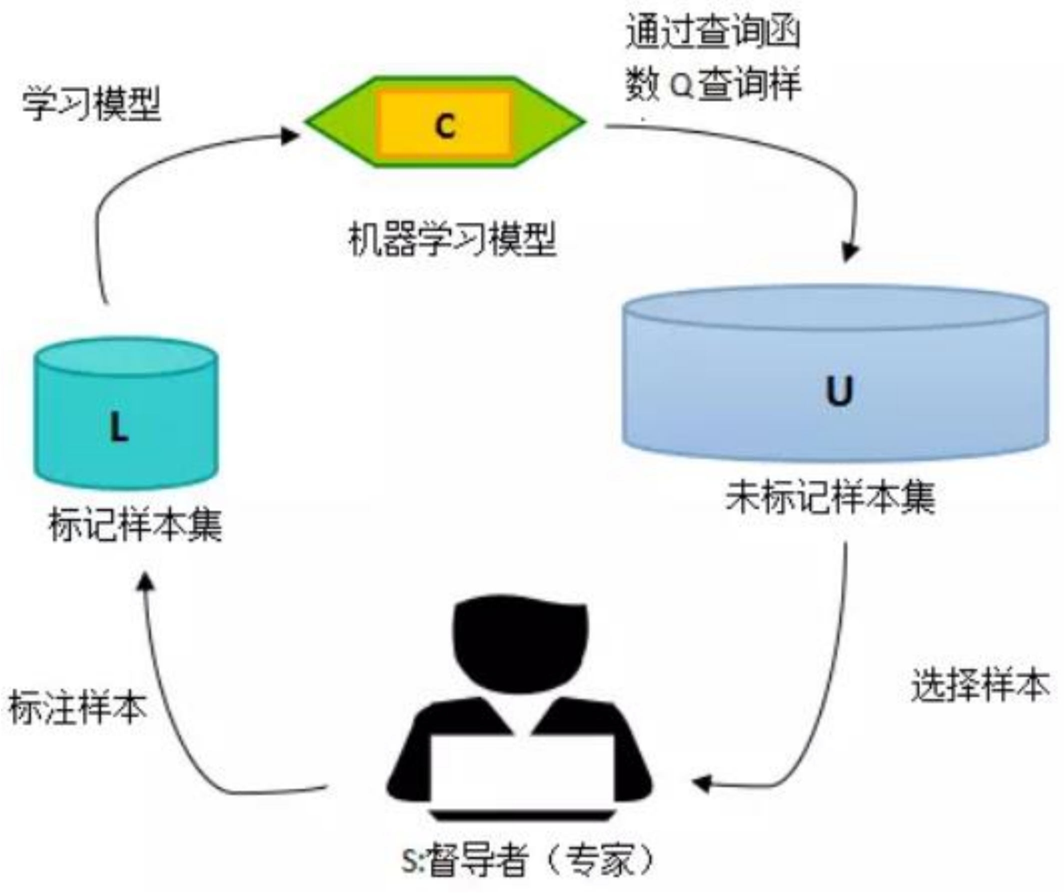
Active Learning
（主动学习）

一、背景

样本标注成本过高，希望使用较少的训练样本来获得性能较好的模型。

二、原理

区别于从样本总体中随机的抽取样本进行学习，主动学习（Active Learning）会在对样本进行评估后，选取“较难”分类的样本供模型学习，进而提高模型的学习效率。



Motivation
(动机)

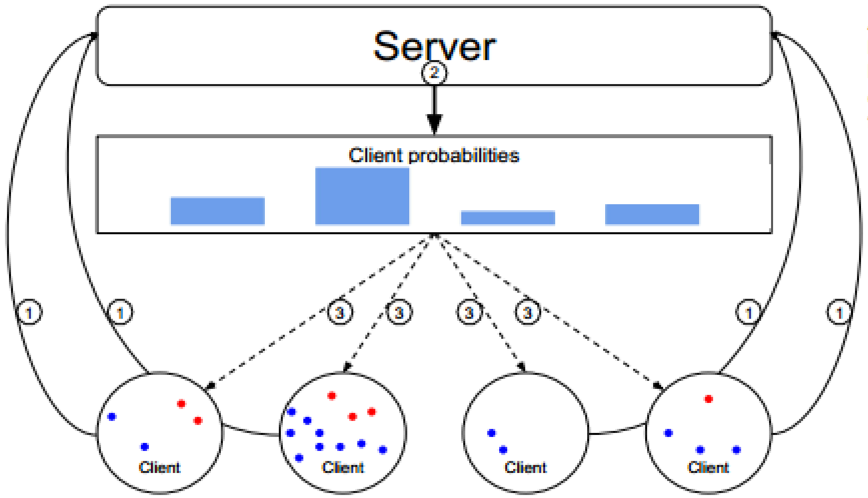
Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:

initialize w_0
for each round $t = 1, 2, \dots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w)**:** *// Run on client k*
  $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
  for each local epoch i from 1 to E **do**
    for batch $b \in \mathcal{B}$ **do**
      $w \leftarrow w - \eta \nabla \ell(w; b)$
  return w to server

$$v_k^{(t+1)} = \begin{cases} \mathcal{V}(\mathbf{x}_k, \mathbf{y}_k; \mathbf{w}^{(t)}) & \text{if } U_k \in S_t \\ v_k^{(t)} & \text{otherwise.} \end{cases}$$



Active Federated Learning algorithm for a binary classification problem. The red and blue dots on each client show the private data on the client. At each training step the following happens:

1. Clients send their valuations to the server.
2. Server converts individual client valuations into probability of each client being selected in the next batch.
3. Server selects next training batch randomly using these client probabilities.

Figure 1: **Active Federated Learning framework** for a binary classification problem.

Algorithm
(算法)

Algorithm 1: Sampling algorithm

Input: Client Valuations $\{v_1, ..., v_K\}$, tuning parameters $\alpha_1, ..., \alpha_3$, number of clients per round m
Output: Client indices $\{k_1, ..., k_m\}$
Sort users by v_k
For the $\alpha_1 K$ users with smallest v_k, $v_k = -\infty$
for k *from* 1 *to* K **do**
| $p_k \propto e^{\alpha_2 v_k}$
end
Sample $(1 - \alpha_3)m$ users according to their p_k, producing set $\mathcal{S}'$
Sample $\alpha_3 m$ from the remaining users uniformly at random, producing set $\mathcal{S}''$
return $\mathcal{S} = \mathcal{S}' \cup \mathcal{S}''$

$$v_k = \frac{1}{\sqrt{n_k}} l(\mathbf{x}_k, \mathbf{y}_k; \mathbf{w})$$

The $\alpha 1$ proportion of users with the smallest valuations will have their valuations set to -∞. They can still be selected by random sampling.
$\alpha 2$ is our softmax temperature.
$\alpha 3$ is the proportion of users which are selected uniformly at random.
($\alpha 1 = 0.75$; $\alpha 2 = 0.01$; $\alpha 3 = 0.1$)

Experiments
(实验)

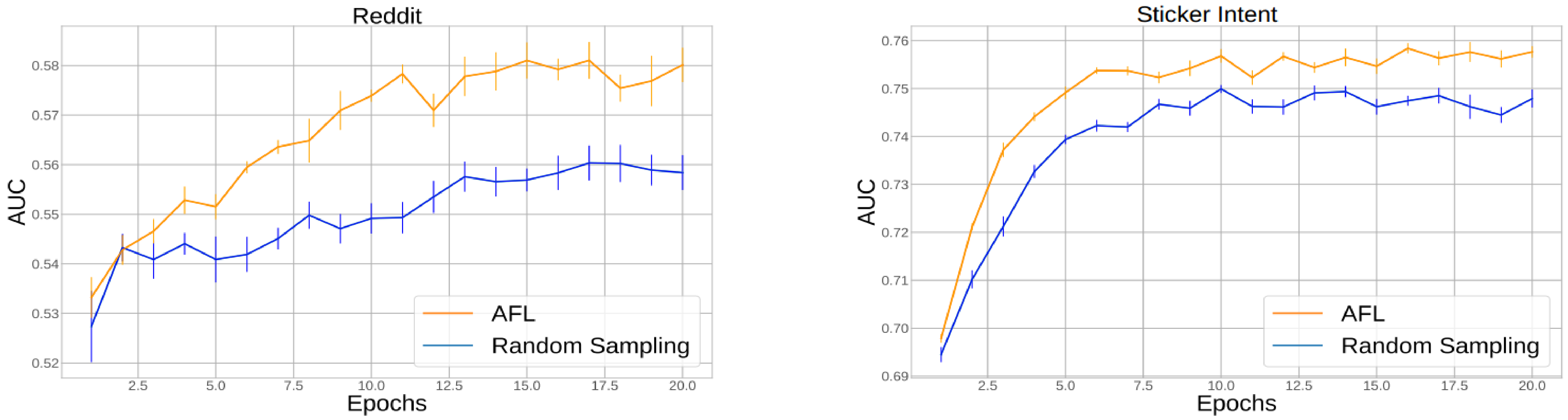


Figure 2: Comparison of AUC increase on Reddit and Sticker Intent datasets

注：

AUC曲线： ROC曲线下与坐标轴围成的面积

ROC曲线： 的横坐标是**伪阳性率**（假正类率，False Positive Rate），纵坐标是**真阳性率**（真正类率，True Positive Rate）

Paper-2
FedMD: Heterogenous Federated Learningvia Model Distillation
（联邦蒸馏学习）

Knowledge Distillation
(知识蒸馏)

Paper：Distilling the Knowledge in a Neural Network (2014 NIPS)

Motivation: Model compression.

What is Knowledge : **Soft Target**

Loss: $$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)} \qquad D_{\mathrm{KL}}(P\|Q) = -\sum_i P(i) \ln \frac{Q(i)}{P(i)}.$$

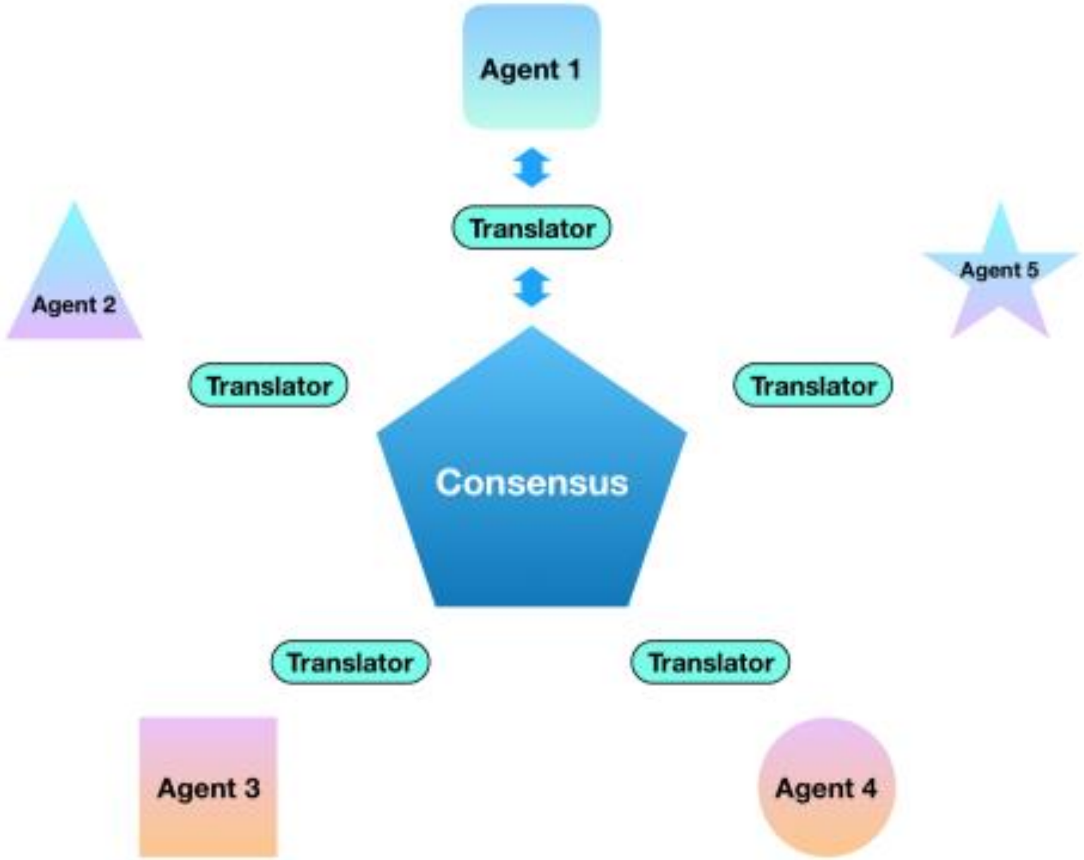
$$\boldsymbol{L = \lambda * L_{soft} + (1-\lambda) * L_{hard}}$$

Why it works: **extra information / dark knowledge**

Motivation
（动机）

蒸馏的优势：

1）模型异构
2）通信量
3）模型表示
4）数据非同分布



Algorithm
（算法）

Algorithm 1: The FedMD framework enabling federated learning for heterogeneous models.

Input: Public dataset $\mathcal{D}_0$, private datasets $\mathcal{D}_k$, independently designed model f_k, $k = 1 \ldots m$,
Output: Trained model f_k

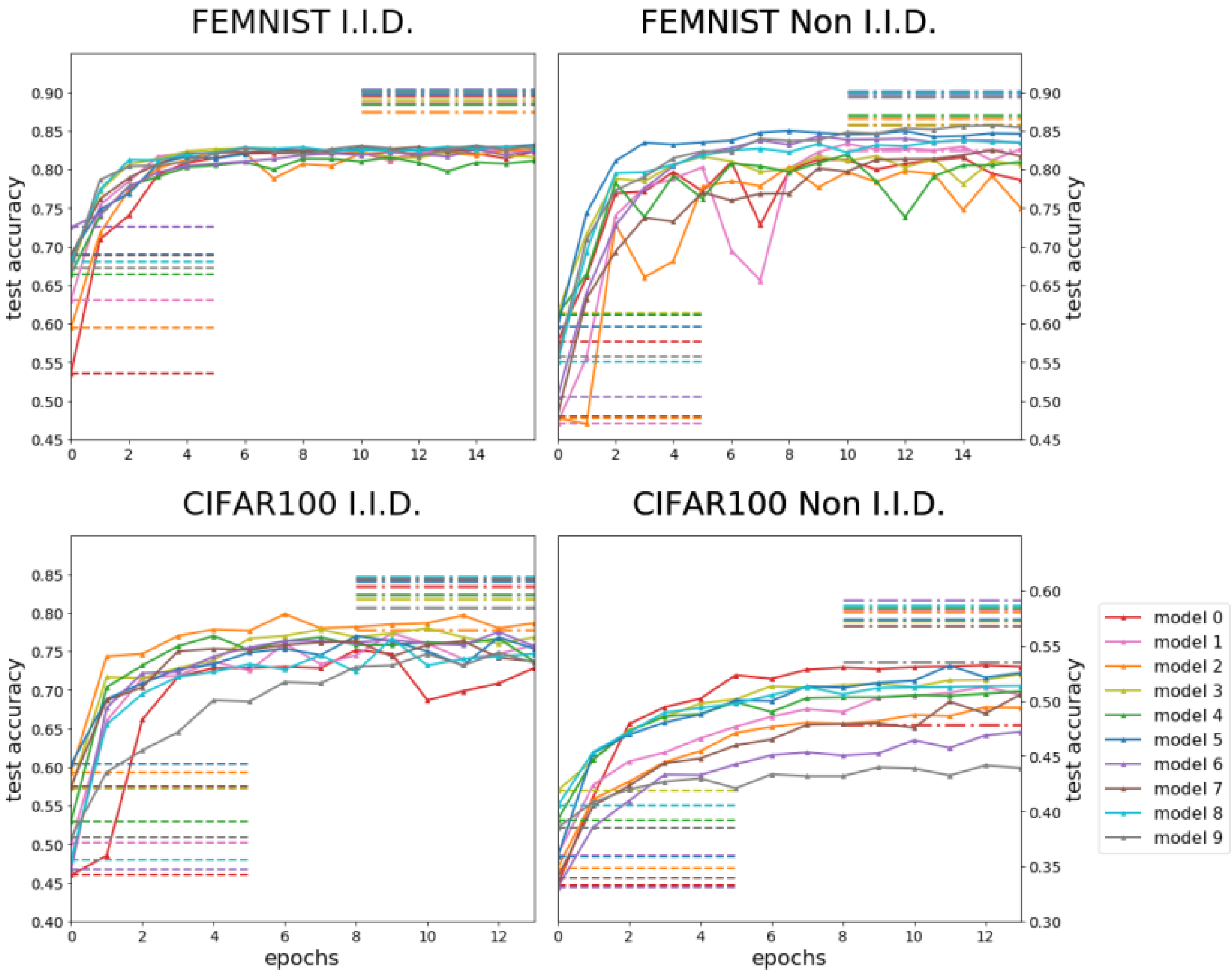
Transfer learning: Each party trains f_k to convergence on the public $\mathcal{D}_0$ and then on its private $\mathcal{D}_k$.

for j=1,2...P **do**

- **Communicate:** Each party computes the class scores $f_k(x_i^0)$ on the public dataset, and transmits the result to a central server.
- **Aggregate:** The server computes an updated consensus, which is an average $\tilde{f}(x_i^0) = \frac{1}{m}\sum_k f_k(x_i^0)$.
- **Distribute:** Each party downloads the updated consensus $\tilde{f}(x_i^0)$.
- **Digest:** Each party trains its model f_k to approach the consensus $\tilde{f}$ on the public dataset $\mathcal{D}_0$.
- **Revisit:** Each party trains its model f_k on its own private data for a few epochs.

end

Experiments
(实验)



谢谢