FDML: A Collaborative Machine Learning Framework for Distributed Features

Yaochen Hu[1], Di Niu[1], Jianming Yang[2], Shengping Zhou[2]
[1] ECE, University of Alberta
[2] PCG, Tencent

Data Matters but...

- Performance of a machine learning model often depends on the **availability** of data

- A large quantity of useful data may be generated on and held by multiple **distributed** parties

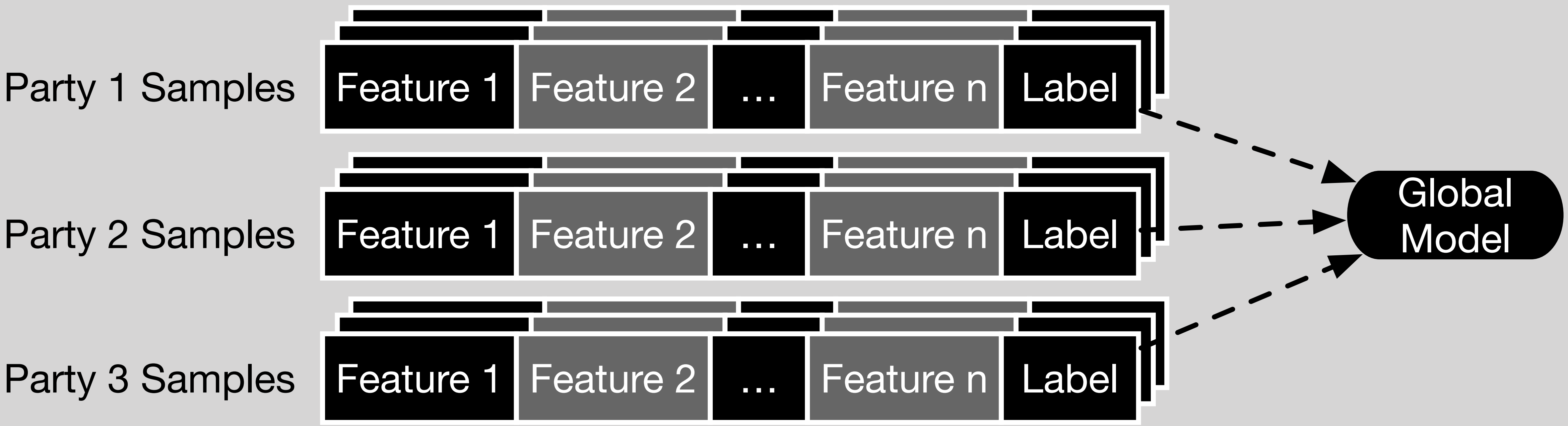
- Extra management and business compliance overhead, privacy concerns, or even regulation and judicial issues

- **How can we use the distributed features without exposing them?**

KDD

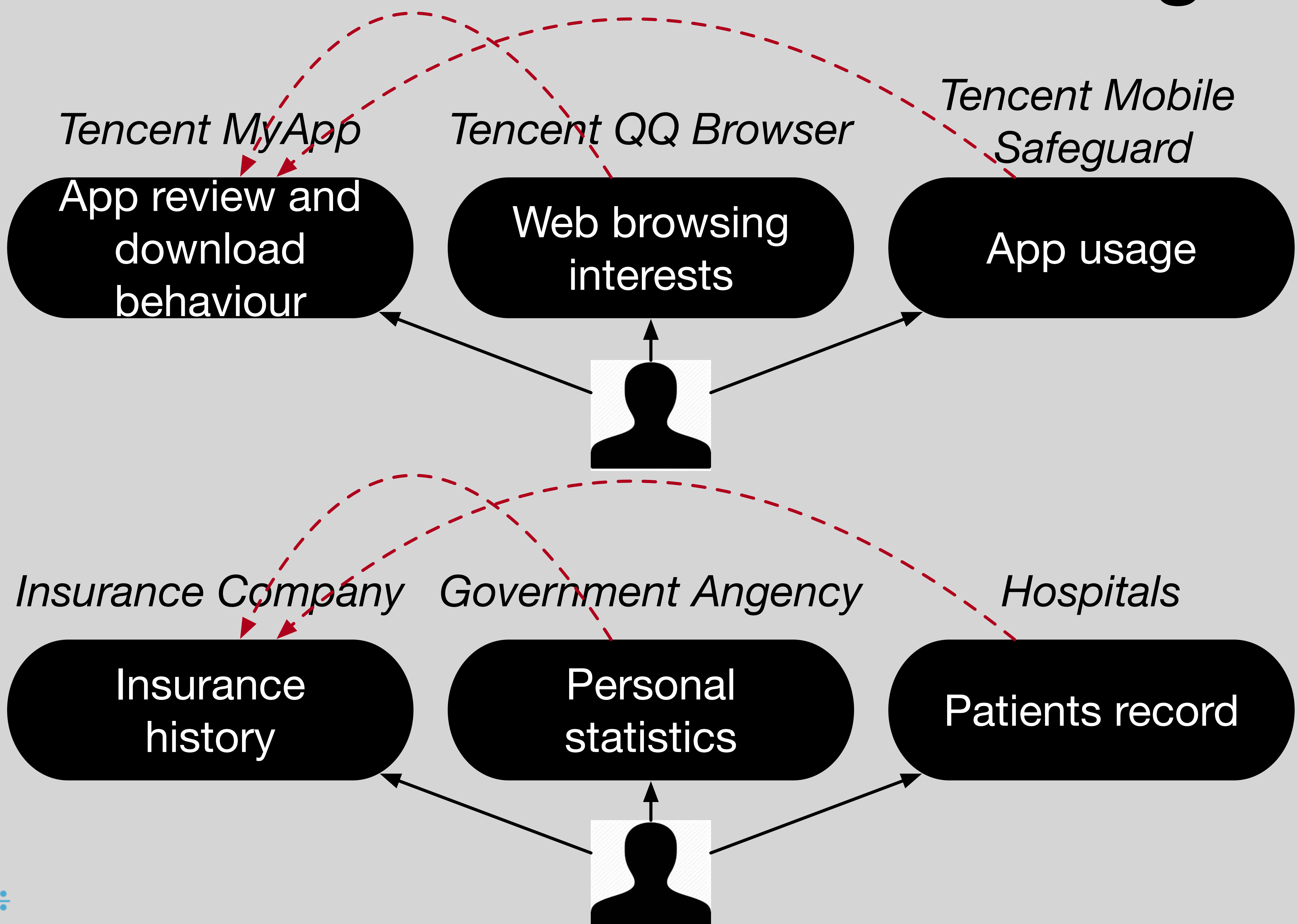
Traditional Distributed Machine Learning: Speedup and Scalability

To speed up training over vast data or handling big model that cannot fit in a single machine's memory [Mu Li et al, OSDI 2014]



Data samples with full features are distributed in order to **speed up training**

Distributed Feature Sharing



Goals

Train a Joint Machine Learning Model with Distributed Features that

- Better prediction performance than local data only and approaching to the performance of centralized training
- Minimize information leakage
- Efficient in both large numbers of features and samples

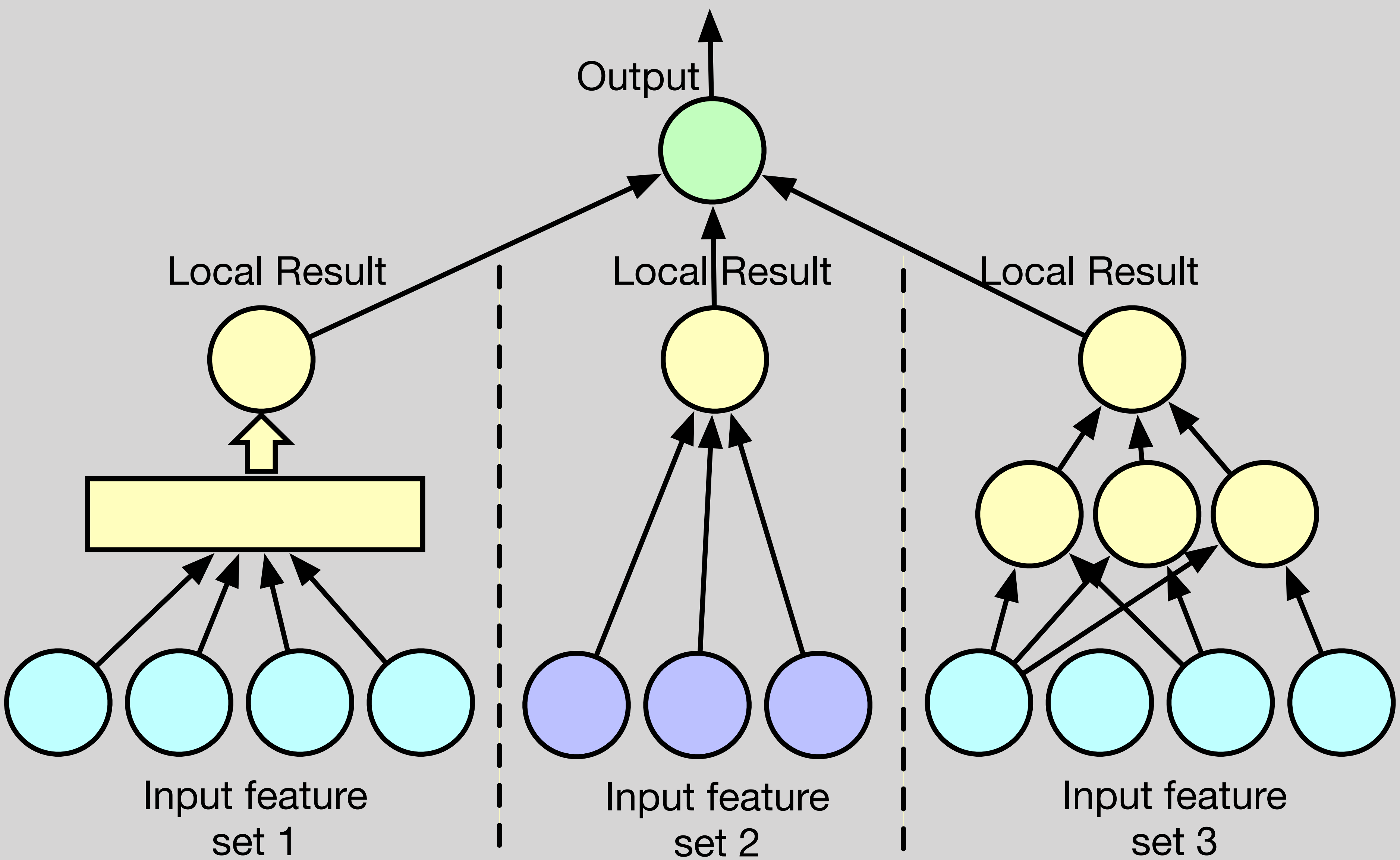
Our Contributions

Design, implement and evaluate the feature distributed machine learning (FDML) system

- A composition model
- Stochastic gradient decent based training algorithm
- Convergence guarantee
- Evaluation over real data trace

KDD

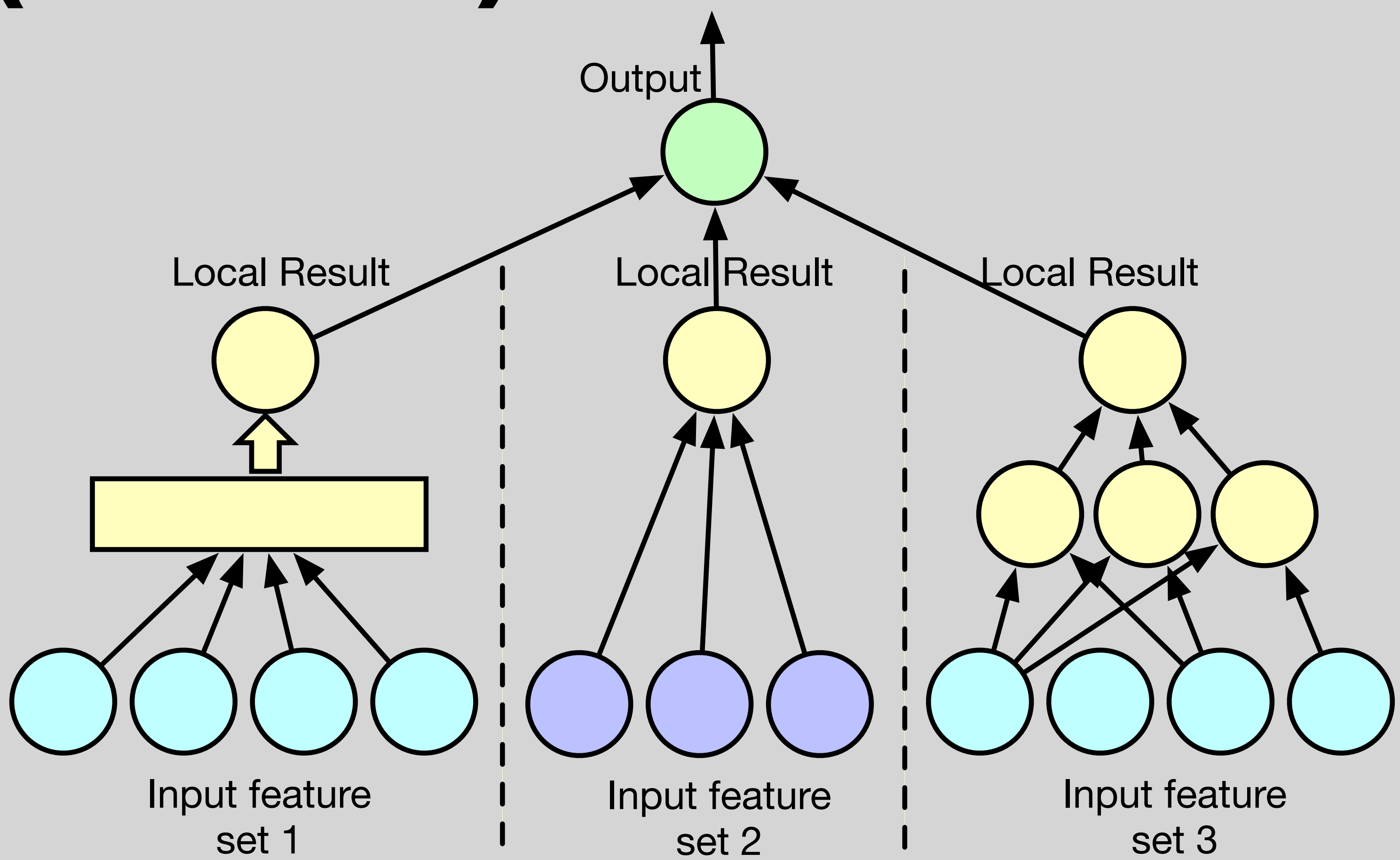
Feature Distributed Machine Learning (FDML) Model



- Composition Model
- Arbitrary "smooth" local model

KDD

Feature Distributed Machine Learning (FDML) Model



Data:

$$\mathcal{D} = \begin{bmatrix} \mathcal{D}_1^1 & \mathcal{D}_2^1 & \cdots & \mathcal{D}_M^1 \\ \mathcal{D}_1^2 & \mathcal{D}_2^2 & \cdots & \mathcal{D}_M^2 \\ \vdots & \vdots & \ddots & \vdots \\ \mathcal{D}_1^N & \mathcal{D}_2^N & \cdots & \mathcal{D}_M^N \end{bmatrix}$$

Variables:

$$x = (x_1^\top, \cdots, x_m^\top, \cdots, x_M^\top)^\top$$

Prediction Model:

$$f(x, D^i) = \sigma\left(\sum_{m=1}^{M} \alpha^m(x_m, D_m^i)\right)$$

KDD

Feature Distributed Machine learning System(FDML) Model

Prediction Model:

$$f(x, \{D^i\}) = \sigma\left(\sum_{m=1}^{M} \alpha^m(x_m, D_m^i)\right)$$

Training Problem:

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & \frac{1}{N}\sum_{i=1}^{N} l\left(\sum_{m=1}^{M} \alpha^m(x_m, D_m^i)\right), \\ \text{subject to} \quad & x_m \in X_m, m = 1, \ldots, M. \end{aligned}$$

Stale Synchronized SGD

Loss: $$F_t(x) = l\left(\sum_{m=1}^{M} \alpha^m(x_m, D_m^t)\right)$$

Gradient to local variables:

$$\frac{\partial F_t(x)}{\partial x_m} = l'\left(\sigma\left(\sum_{m=1}^{M} \alpha^m(x_m, D_m^t)\right)\right)\sigma'\left(\sum_{m=1}^{M} \alpha^m(x_m, D_m^t)\right)\frac{\partial \alpha^m(x_m, D_m^t)}{\partial x_m}$$

$$:= H\left(\sum_{m=1}^{M} \alpha^m(x_m, D_m^t)\right)\frac{\partial \alpha^m(x_m, D_m^t)}{\partial x_m},$$

Update: $$x_m^{t+1} := x_m^t - \eta_t \frac{\partial F_t(x)}{\partial x_m}$$

Stale Synchronized SGD

Local party: Evaluate $\alpha^m(x_m, D_m^t)$ and push it to the server

Pull $\sum_{m=1}^{M} \alpha^m(x_m, D_m^t)$

Update the weights $x_m^{t+1} := x_m^t - \eta_t \frac{\partial F_t(x)}{\partial x_m}$

Server:

For pulling request:

If time index is within threshold:

return $\sum_{m=1}^{M} \alpha^m(x_m, D_m^t)$

For push request:

Collect and update local cache

converges in $O(1/\sqrt{T})$

T being the total number of iterations

KDD

System Highlights

- Sample and feature preparation
  - Distribute samples with user identity (User ID, phone number, etc)
  - Link user features to samples in different local parties
- Sampling and alignment
  - Random shuffling and align the samples in the same order in different parties
- Differential Privacy Technique
  - Perturb the local prediction with controlled random noise

Experiment Dataset

- 5,000,000 samples indicating whether a user will download an app or not
- 8,700 (sparse) features in total, among which 7,000 features come from Tencent MyApp, 1700 features are from other two apps

KDD

Experiment Scenarios

- Logistic regression (LR) and a two layered fully connected neural network (NN)
  - **Local**: only use 7,000 local features from MyApp
  - **Centralized:** collect all the 8,700 features from all three apps to a central server
  - **FDML**: use FDML system to train a joint model on all 8,700 features distributed in three apps

KDD

Experiment Results

Table 1: The performance on Tencent MyApp data.

Algorithm	Train loss	Test loss	Test AUC	Time(s)
LR local	0.1183	0.1220	0.6573	546
LR centralized	0.1159	**0.1187**	**0.7037**	1063
LR FDML	**0.1143**	0.1191	0.6971	3530
NN local	0.1130	0.1193	0.6830	784
NN centralized	**0.1083**	0.1170	**0.7284**	8051
NN FDML	0.1101	**0.1167**	0.7203	4369

The smaller, the better for loss and auc

Conclusion

Learning privately over distributed features is an important problem

Feature Distributed Machine Learning (FDML) system can significantly outperforms the model with local features while keeping the distributed features private

KDD

Future Works

- More robust and efficient algorithm for dense and high dimensional local features
  - Y. Hu, et al. "Learning Privately over Distributed Features: An ADMM Sharing Approach." *arXiv preprint arXiv: 1907.07735* (2019).
- Support more feature interactions between parties, such as factorization machine model
  - Richer "symmetric feature interactions"
- Communication reduction

Thank you!

Yaochen Hu
PhD Candidate
University of Alberta

Email: yaochen@ualberta.ca