FDML: A Collaborative Machine Learning Framework for Distributed Features

1. Problem formulation

m 个不同的 parties，每个party有相同training samples

如图所示是n个samples

$$\{(\xi_i^1, \xi_i^2, \ldots, \xi_i^m), y_i\}_{i=1}^n$$

下图，代表了第i个sample的第j个party的features

$$\xi_i^j \in \mathbb{R}^{d^j}$$

下图代表第i个sample的所有features，是每个party的concatenation

$$\xi_i \in \mathbb{R}^d$$

$p(x, \xi)$是一个local model，输出是一个prediction

Feature Distributed Machine Learning (FDML) model：

(这里的α^j是一个sub-model,可以看作是一个local features 到 local prediction的映射)

(σ可以聚合local intermediate predictions，a_j是权重)

$$p(x, \xi) = \sigma\left(\sum_{j=1}^{m} a_j \alpha^j (x^j, \xi^j)\right), \qquad (1)$$

这里的model也是一个复合模型，每个sub-model都是可以不同的

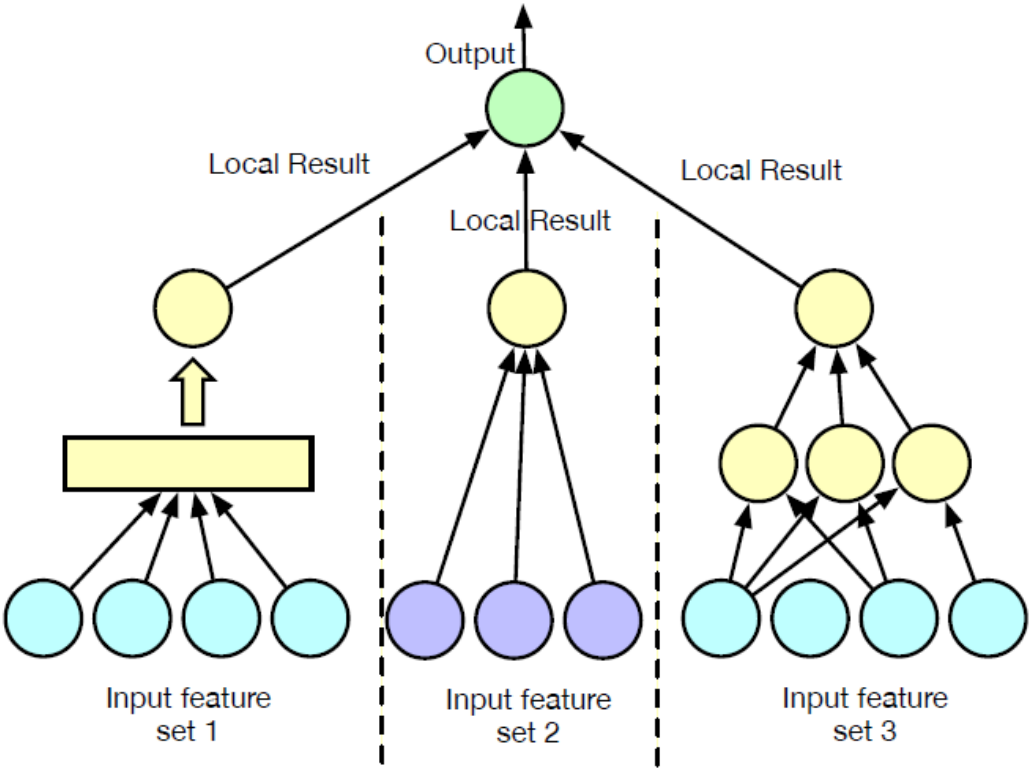


Figure 1: An illustration of the FDML model (2), where each party may adopt an arbitrary local model that is trainable via SGD. The local predictions, which only depend on the local model parameters, are aggregated into a final output using linear and nonlinear transformations (1).

这里需要共享的只有$\alpha^j(x^j, \xi^j)$，以此来得到final prediction

（raw data 和 sub-model parameters是不会泄露的）

目标函数:

L 是loss function, $z(x^j)$是sub-model x^j的正则项

$$\text{minimize}_x \frac{1}{n} \sum_{i=1}^{n} L(x; \xi_i, y_i) + \lambda \sum_{j=1}^{m} z^j(x^j), \tag{3}$$

2. Asynchronous SGD for FDML

$i(t)$是sample $\xi_{i(t)}$在第 t iteration的 index

如下是在sample $i(t)$，即 t iteration的objective function

$$F_t(x) := L(x; \xi_{i(t)}, y_{i(t)}) + \lambda \sum_{j=1}^{m} z^j(x^j), \tag{4}$$

整个training set的objective function：

（T是总的iteration number）

$$\text{minimize}_x F(x) := \frac{1}{T}\sum_t F_t(x), \tag{5}$$

$\nabla F(x) \in R^D$是 F 的gradient，$\nabla^j F(x) \in R^{D^j}$是 F对于sub-model 参数$x^j$的partial gradient，即$\nabla^j F(x) := \frac{\partial F(x)}{\partial x^j}$,那么 ∇F就是所有的partial gradient 的concatenation

$$\nabla^1 F(x), \nabla^2 F(x), \ldots, \nabla^m F(x).$$

2.1 The Synchronous Algorithm

- 简单直接并行化SGD

第 t 个iteration，对于party j 的objective function的梯度：

这里用H来代表$\sum_k^m \alpha^k(x^k, \xi_{i(t)}^k)$

$$\nabla^j F_t(x) = \lambda \frac{\partial z^j(x^j)}{\partial x^j} +$$

$$L'\left(\sigma\left(\sum_{k=1}^m \alpha^k(x^k, \xi_{i(t)}^k)\right)\right)\sigma'\left(\sum_{k=1}^m \alpha^k(x^k, \xi_{i(t)}^k)\right)\frac{\partial \alpha^j(x^j, \xi_{i(t)}^j)}{\partial x^j} \tag{6}$$

$$:= H\left(\sum_{k=1}^m \alpha^k(x^k, \xi_{i(t)}^k)\right)\frac{\partial \alpha^j(x^j, \xi_{i(t)}^j)}{\partial x^j} + \lambda \frac{\partial z^j(x^j)}{\partial x^j}, \tag{7}$$

为了计算$\nabla^j F$，每个party只需要得到$\sum_k^m \alpha^k(x^k, \xi_{i(t)}^k)$，这就是在第 t 个iteration所有local prediction results 的聚合，剩余的项都可以用第 j 个party的local data计算得到。

2.2 The Asynchronous Algorithm

在异步情况下，每个party j 都会异步地更新自己的参数x_t^j，任意的两个parties都可能是处于不同的iteration。

但是，这里假设不同的parties运行samples是**go through in the same order**

通常是随机生成sample index sequence，解决方案是：**pseudo random number generator**

local parameters update:

$$x_{t+1}^j = x_t^j - \eta_t\left(H\left(\sum_{k=1}^m \alpha^k(x^k_{t-\tau_t^j(k)}, \xi_{i(t)}^k)\right)\frac{\partial \alpha^j(x_t^j, \xi_{i(t)}^j)}{\partial x^j} + \lambda \frac{\partial z^j(x_t^j)}{\partial x^j}\right), \tag{8}$$

此时，很有可能 party 去quest local predictions 的 aggregation 时候，可能是 **stale versions**，$x_{t-\iota_t^j(k)}$

这里的$\iota_t^j(k)$,代表的是i从party j 到 party k 在 party j 的第 k 个 teration的 “ lag”

3. Distribution Implementation

3.1 Implementation

PS架构： workers 计算 gradients ， server更新model

但是，在FDML中，server**只要更新一个local prediction matrix** $A_{i,j}$ (n行，m列)

用来hold 对于sample i 的最新的 m 个prediction

而且，worker在 FDML中是participating party，不仅需要计算 gradients，而且需要更新自己的 local model parameters

整个过程：

- a sample coordinator随机shuffle sample indices, 然后生成一个schedule $i(t)$，每次都要找到all features以及拥有sample对应的label。在算法开始之前每个party 就有：$\xi_i^j, y_i{}_{i=1}^n$
- 每个party update 自己对sample $i(t)$ 的 local prediction $A_{i(t),j}$

$$A_{i(t),j} := \alpha^j(x_t^j, \xi_{i(t)}^j)$$

(Push request: 即 worker j 在iteration t，upload value c)

- party Pull 最新的的 $\sum_{k=1}^m A_{i(t),k}$
- party update x_t^j 到 x_{t+1}^j

a fully synchronous algorithm可以在最短的iterations 达到converge

但是，导致了更大的等待同步的时间

a asynchronous algorithm可以减少每轮的时间，但是需要更多的iterations达到converge

为了减少总的时间，需要最快iteration的party比最慢的iteration超出的不能超过bound ι (加入bound来保证 convergence)

SGD algorithm可以被更好的 mini-batch SGD替换

Algorithm 1 A Distributed Implementation of FDML

Require: each worker j holds the local feature set $\{\xi_i^j, y_i\}_{i=1}^n$, $j = 1, \ldots, m$; a sample presentation schedule $i(t)$, $t = 1, \ldots, T$, is pre-generated randomly and shared among workers.

Output: model parameters $x_T = (x_T^1, \ldots, x_T^m)$.

Server:

Initialize the local prediction matrix $[A_{i,j}]_{n\times m}$.

while True **do**

if *Pull request* (worker: j, iteration: t) received **then**

if t is not τ iterations ahead of the slowest worker **then**

Send $\sum_{k=1}^m A_{i(t),k}$ to Worker j

else

Reject the *Pull request*

end if

end if

if *Push request* (worker: j, iteration: t, value: c) received **then**

$A_{i(t),j} := c$.

end if

end while

Worker j ($j = 1, \ldots, m$) asynchronously performs:

for $t = 1, \ldots, T$ **do**

Push $c := \alpha^j(x_t^j, \xi_{i(t)}^j)$ to Server

while Pull not successful **do**

Pull $\sum_{k=1}^m A_{i(t),k}$ from Server

end while

$$\nabla^j F_t := \left(H(\textstyle\sum_{k=1}^m A_{i(t),k}) \cdot \dfrac{\partial \alpha^j(x_t^j, \xi_{i(t)}^j)}{\partial x^j} + \lambda \dfrac{\partial z^j(x_t^j)}{\partial x^j} \right)$$

Update the local weights as

$$x_{t+1}^j := x_t^j - \eta_t \nabla^j F_t . \tag{9}$$

end for

3.2 Privacy

不会泄露 weights 和 features 信息

共享的信息只是：local prediction （是对于 local weights 和 local features 的复合函数）

此外，也可加入扰动项，noise 对于local prediction

$$\alpha^j(x_t^j, \xi_{i(t)}^j)$$

4. Experiments

an app recommendation task at Tencent MyApp

利用另外两个app中的信息达到 cross-domain的效果，提高准确性

Dataset:

- **Tencent MyApp data** : 5, 000, 000 labeled samples indicating whether a user will download an app or not
- **a9a**,: classical census dataset, where the prediction task is to determine whether a person makes over $50K a year.

Model:

- logistic regression (**LR**)
- a two layered fully connected neural network (**NN**)

Training schemes:

- **Local:** 7, 000 local feature or 67 features of a9a
- **Centralized:** collect all the 8, 700 features or using all the 124 features in a9a
- **FDML**: 8, 700 features distributed or a9a classification model on all 124 features from two different parties

---

如下Tables， FDML在保证local features的同时，**优于 Local scheme，接近于Centralized的情况**

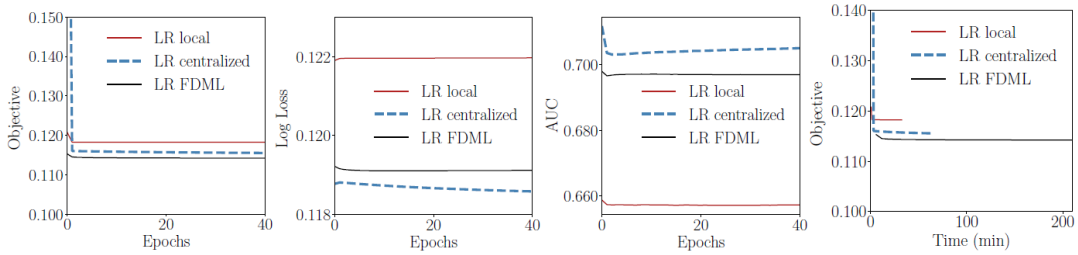
Table 1: The performance of different algorithms for Tencent MyApp data.

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

Table 2: The performance of different algorithms for a9a data.

Algorithm	Train loss	Test loss	Test AUC	Time(s)
LR local	0.3625	0.3509	0.8850	41
LR centralized	0.3359	0.3247	0.9025	45
LR FDML	**0.3352**	**0.3246**	**0.9026**	99
NN local	**0.3652**	0.3484	0.8864	53
NN centralized	0.4008	**0.3235**	**0.9042**	57
NN FDML	0.4170	0.3272	0.9035	110

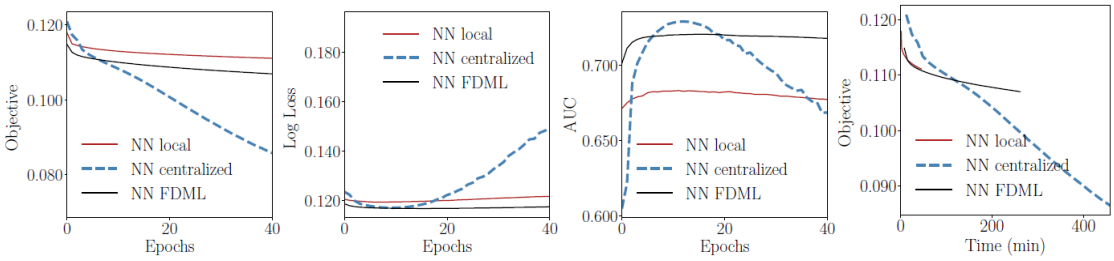
LR model：objective value，loss , AUC值 FDML与Centralized情况下都比较好



(a) Training objective vs. epoch (b) Tesiting log loss vs. epoch (c) Tesiting AUC vs. epoch (d) Training objective vs. time

Figure 2: A comparison between the three model training schemes for the LR model. All curves are plotted for epochs 1–40, including the time curve in (d).

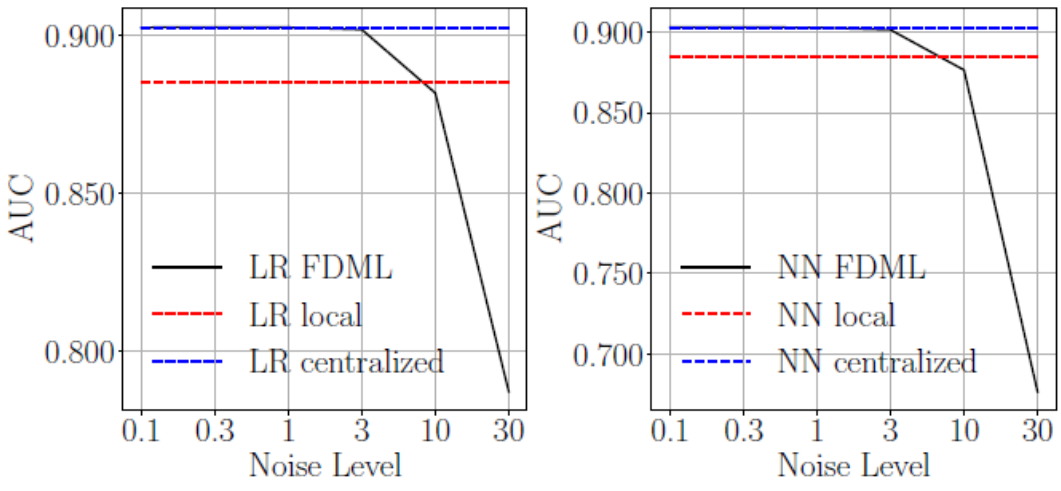
NN model：两种机制下，local scheme都是最快的，没有communication 和 synchronous 的开销



(a) Training objective vs. epoch (b) Tesiting log loss vs. epoch (c) Tesiting AUC vs. epoch (d) Training objective vs. time

Figure 3: A comparison between the three model training schemes for the NN model. All curves are plotted for epochs 1–40, including the time curve in (d).

DP 机制： 不同的noise level



(a) Test AUC vs. added noise for LR (b) Test AUC vs. added noise for NN

Figure 4: Testing AUC under different levels of added noise during training for a9a data set

5. Conclusion

motivation： 相同的 training sample 在不同的app 中有不同的features，但是一个app中的data必须对其他的app是confidential

convergence：达到与目前最快的data-parallel SGD in a stale synchronous parallel， $O(1/\sqrt{T})$, T是iteration number

results : FDML AUC、loss 效果接近于centralized training （后者model更复杂）

future work: add momentum and privacy