<< عنوان پژوهش : برسی Federated Learning >>

محمد حسین هاشمی

1404/7/23

تیم AI

فهرست مطالب

●مقدمه :

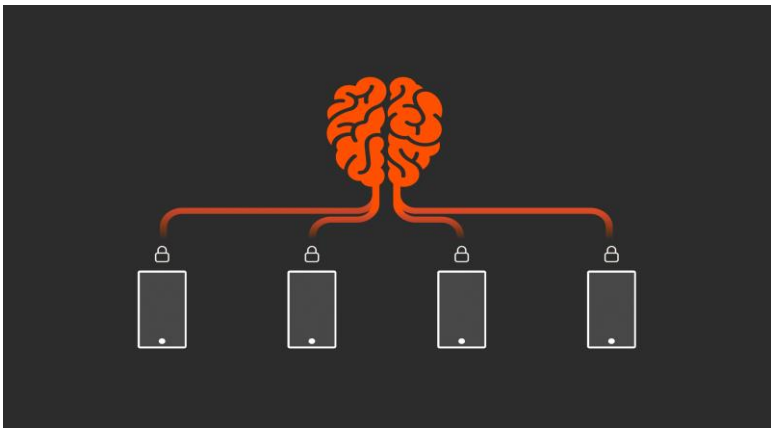
امروزه موبایل ها و تبلت ها برای خیلی از افراد دستگاه اصلی کامپیوتری محسوب می شوند ، این دستگاه ها پر از سنسور های قدرتمند مثل دوربین ، میکروفون و GPS هستند ، و چون همیشه همراه کاربر هستند ، حجم بسیار زیادی داده جمع می کنند ، داده هایی که اغلب خصوصی هستند ! حالا اگر بتوانیم روی این حجم زیاد و متنوع از داده ها مدل یادگیری ماشین بسازیم ، برنامه ها خیلی هوشمند تر و کاربردی تر می شوند . اما مشکل اینجاست که نمی توانیم همه این داده ها را در یک سرور مرکزی جمع کنیم ، هم بخاطر حریم خصوصی افراد و هم بخاطر به صرفه نبودن این انتقال !

اینجاست که Federated Learning وارد می شود .

❄ **Federated Learning چیست ؟** ایده این است که بجای اینکه داده ها از دستگاه کاربران (مثلا گوشی) به سرور فرستاده بشود ، فقط به روز رسانی های مدل ارسال شود ، یعنی گوشی روی داده های خودش مدل را تمرین می دهد و بعد بجای ارسال داده خام ، فقط تغییرات یا آپدیت های مدل به سرور فرستاده می شوند ، سپس سرور این تغییرات رو جمع می کند و مدل کلی را آپدیت می کند .

❄ مزایای Federated learning :

1. داده خام منقل نمی شود که این از نظر امنیت و حریم خصوصی خیلی خوبه !
2. مدل همچنان می تواند از داده های همه کاربران استفاده کند که این باعث افزایش دقت و کیفیت مدل می شود .



● چالش ها :

1. **نا همگنی داده ها (Non-IID Data) :** داده های کاربران معمولا از یک توزیع مشابه نیستند . به عنوان مثال رفتار خرید کاربران در مناطق مختلف یا علایق مختلف می تواند خیلی متفاوت باشد . همچنین بعضی کاربران داده های زیادی دارند و بعضی خیلی کم !
2. **چالش های سیستم و ارتباطات :** ارسال وزن ها به سرور و دریافت مدل به روز شده از طریق اینترنت ممکن است کند یا غیر قابل اعتماد باشد . همچنین دستگاه های کاربران از نظر پردازش ، حافظه و باتری متفاوت هستند که اجرای یک مدل بزرگ روی همه دستگاه ها را سخت می کند .
3. **چالش های امنیت و حریم خصوصی :** حتی اگر داده ها منتقل نشوند ، می توان با آنالیز آپدیت ها ، اطلاعات حساس کاربران را حدس زد .

● معماری و پروتکل Federated Learning :

✦ **Device** : در این قسمت به این موضوع می پردازیم که در دستگاه کاربر (مثلا گوشی اندروید) در Federated Learning چه اتفاقاتی می افتد.

1. **ذخیره داده محلی :** اولین وظیفه دستگاه (گوشی) این است که داده های کاربر رو به صورت محلی ذخیره کند . به عنوان مثال ، تصور کن یک اپ پیشنهاد فیلم داری ، این اپ داده هایی مثل ((چه فیلمی پیشنهاد داده شده ؟؟ آیا کاربر فیلم را پسندیده یا نه ؟؟ و ....)) ذخیره می کند و این داده ها می روند توی Example Store(مثلا یک دیتا بیس SQLite)
2. **FL Runtime** : این بخش مثل یک مامور از طرف سرور در دستگاه یا همون گوشی است . وقتی سرور یک وظیفه (Task) به گوشی میده ، FL Runtime سراغ Example Store می رود ، داده ها را می خواند و :
   - یا مدل را با داده های محلی آموزش می دهد .

- یا کیفیت مدل را با همون داده ها می سنجد .

3. **جریان کنترل (Control Flow)** : فرایند آموزش مدل نباید تجربه کاربری را خراب کند ، پس فقط وقتی گوشی بیکار باشد و در حال شارژ باشد و به یک اینترنت غیر حجمی (مثل WiFi) وصل باشد ، اون موقع FL Runtime فعال می شود !

✦ **Server :** بخش سرور بر روی این موضوع تمرکز دارد که چطور زیر ساختی طراحی کنند که مقیاس پذیر (یعنی درست کار کردن با تعداد کم دستگاه و همینطور تعداد میلیاردی دستگاه ها) ، انعطاف پذیر (هم آپدیت های کوچک و هم خیلی بزرگ از طرف دستگاه ها را مدیریت کند) و همینطور مقاوم به تغییر شرایط (مثلا غیر یکنواخت بودن ترافیک شبکه) باشد .

❄ **Actor Model :** در معماری سرور از مفهومی به نام Actor Model ها استفاده می شود. این ها مثل کارگرانی ساده هستند که کار های مشخصی را در سرور انجام می دهند . مزایا :

1. اگر تعداد در خواست ها و آپدیت ها از سوی گوشی موبایل ها زیاد باشد ، می تونیم Actor های بیشتری بسازیم .
2. این Actor ها انعطاف پذیری مکانی دارند ، یعنی می توانند بین دیتاسنتر های مختلف در سراسر دنیا پخش بشوند ، این مزیت باعث می شود که اگر دستگاه ها از فواصل دور مثلا از قاره ای دیگر به سرور متصل می شوند ، پردازش نزدیک به خودشون انجام بشود .
3. همچنین این Actor ها موقتی هستند و تنها زمانی که لازم باشند ساخته می شوند و خب این باعث صرفه جویی در منابع می شود .

Actor Model های متفاوتی با وظایف متفاوت وجود دارند مثل Coordinator ها که هماهنگ کننده اصلی در سرور هستند یا Selector ها که انتخابگر دستگاه ها هستند و ....

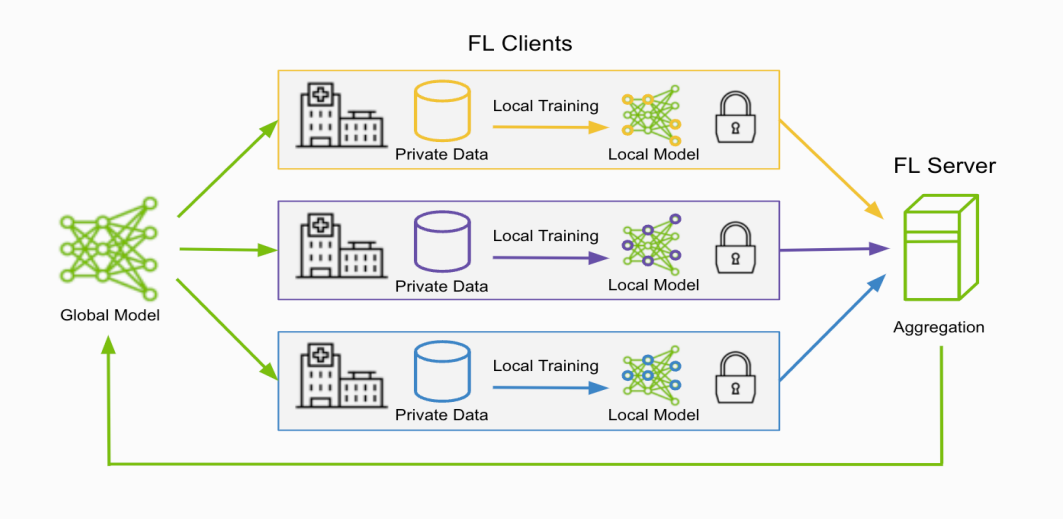
✦ **پروتکل (Protocol)** : در این بخش قواعد ارتباطی بین سرور و دستگاه ها را برسی می کنیم.

در ابتدا دستگاه ها به سرور می گویند که آماده اند ، سپس سرور از بین هزاران دستگاه ، تعدادی را برای شرکت در یک round انتخاب می کند .حالا مدل کلی (Global Model) که روی سرور مرکزی است برای دستگاه های انتخاب شده فرستاده میشود ، هر دستگاه با داده های محلی خودش مدل را کمی بهبود می دهد ، منظور از بهبود هم یعنی قدم به قدم وزن های مدل را تغییر می دهد تا خطا کمتر شود . سپس دستگاه ها به روزرسانی ها را می فرستند به سرور و آن این ها را با هم ادغام می کند . این می شود Round اول .

❄ **سه فاز اصلی هر Round :**

1. **Selection** : در این بخش دستگاه هایی که دارای شرایط مورد نظر سرور هستند ، انتخاب می شوند .
2. **Configuration** : سرور برای هر دستگاه انتخاب شده ، یک FL Plan و یک FL Checkpoint(مدل کلی فعلی) ارسال می کند.
3. **Reporting** : دستگاه ها به روز رسانی ها را به سرور می فرستند ، سرور با الگوریتمی مثل FedAvg که جلوتر معرفی خواهد شد ، به روزرسانی ها را ادغام می کنند.

❄ **Failure Modes** : چندین سناریو متفاوت از شکست و خرابی سرور وجود دارد مثل خرابی Selector ها یا Coordinator ها و ... که در هر حالت ، معماری Federated Learning به صورتی است که متوقف نمی شود و تنها ممکن است که یک Round موقتا متوقف شود یا در بدترین حالت یک Round دوباره تکرار شود .

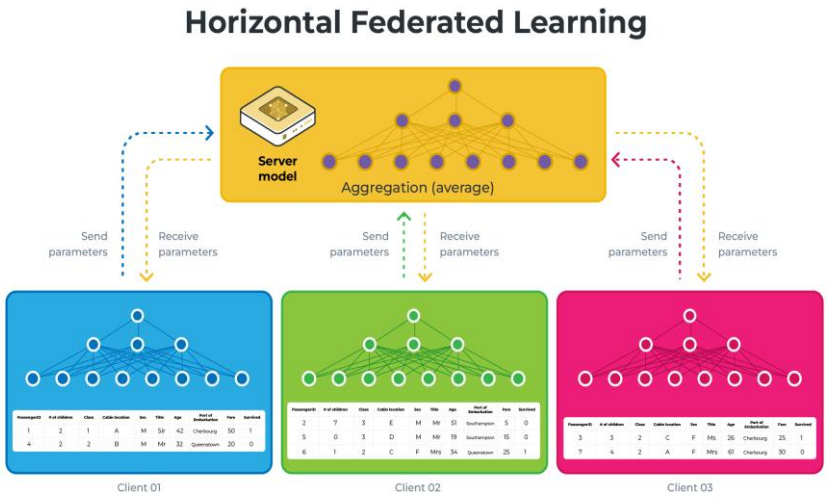


● انواع Federated Learning :

Federated Learning بسته به نحوه توزیع داده ها بین کاربران یا سازمان ها به سه نوع اصلی تقسیم می شود :

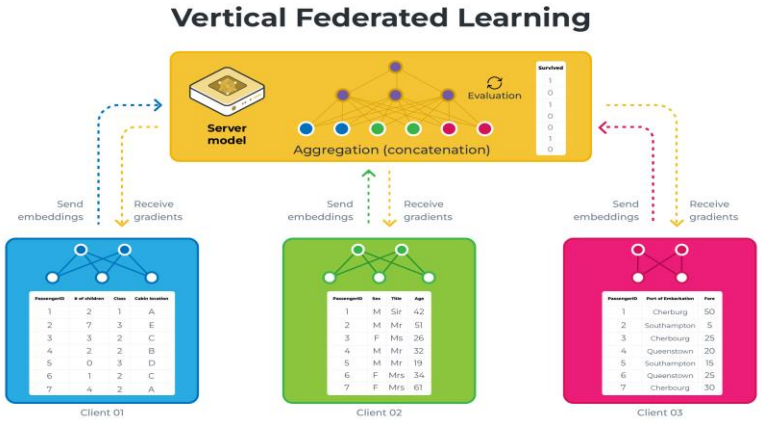
1. **یادگیری فدرال افقی(Horizontal FL)** : همه کاربران دارای ویژگی های مشابه هستند اما داده هاشون روی نمونه ها متفاوت است .

   مثال : چند بیمارستان که ساختار پرونده بیماراشون مشابه است اما بیماران بیماری های متفاوت دارند .



2. **یادگیری فدرال عمودی (Vertical FL)** : همه کاربران نمونه های مشابه دارند ولی ویژگی هاشون متفاوته .

   مثال : یک بانک و یک فروشگاه اینترنتی که همان مشتریان را دارند اما داده های متفاوتی جمع آوری میکنند .



3. **یادگیری انتقالی فدرال (Federated Transfer Learning)** : کاربران هم از نظر نمونه ها و هم ویژگی ها متفاوت هستند اما هنوز می توانند با هم دانش مشترک یاد بگیرند ، این روش برای همکاری بین حوضه های مختلف که داده های مشترک کمی دارند ، خیلی مفیده .

● الگوریتم های Federated Learning :

علی رغم نوآوری های FL در یادگیری ماشین ، مشکلاتی هم نیز دارند مثل :

- **ناهمگونی داده ها**
- **مشکلات ارتباطی**
- **لزوم حفظ حریم خصوصی**

به همین خاطر الگوریتم های مختلفی طراحی شدند تا این محدودیت ها را مدیریت کنند ، این الگوریتم ها را می شود توی 4 دسته اصلی قرار داد :

1. **روش های کلاسیک** : مثل FedAvg و FedSGD که پایه ای ترین روش ها برای آموزش FL هستند :

★ **الگوریتم FedSGD** : این الگوریتم بدین صورت است که سرور مرکزی مدل اولیه را برای **همه** دستگاه ها می فرستد و هر دستگاه روی داده های خودش فقط یک گرادیان محاسبه می کند (یعنی وزن ها را آپدیت نمی کند !) حالا دستگاه ها این گرادیان ها را برای سرور می فرستند ، سرور میانگین همه این گرادیان ها را محاسبه می کند و سپس با این میانگین ، وزن های مدل اصلی را آپدیت می کند .

• نکته این الگوریتم این است که دستگاه ها وزن ها را به روزرسانی نمی کنند ، تنها گرادیان ها را می فرستند .

★ **الگوریتم FedAvg** : FedAvg میاد میگه بجای اینکه مدل کلی را به همه دستگاه ها بفرستیم که فقط هم یکبار هر دستگاه فرایند آموزش را داشته باشد ، تنها به تعدادی از

دستگاه ها مدل را بفرستیم و بزاریم هر دستگاه چندین بار روی داده های خودش تمرین کند ! یعنی :

- سرور مدل کلی رو برای تعدادی **دستگاه** می فرستد .
- هر دستگاه روی داده خودش چند epoch(E) آموزش با سایز Batch(B) مشخص انجام می دهد .
- در پایان هر دستگاه آپدیت های مدل به روز شده اش را برای سرور می فرستد و سرور همه این آپدیت ها را میانگین گیری وزنی می کند .

◉ **نکته** : اگر $E = 1$ و $B = \infty$ باشد ← FedSGD

اگر $E > 1$ و B کوچک باشد ← FedAvg

2. **Optimization-Awar Techniques** : برای مقابله با داده های متنوع و شرایط پیچیده طراحی شدند مثل FedProx و Scaffold :

- **FedProx** = نسخه ای بهبود یافته از FedAvg که یک Proximal Term به تابع هدف اضافه می کند . در واقع این Term باعث می شود تا آپدیت های دستگاهی که داده هاش خیلی متفاوت (Non IID) از بقیه دستگاه است ، خیلی از مدل کلی فاصله نگیرد و دور نشود !

محدودیت ها : می تواند باعث کند تر شدن همگرایی شود .

- **Scaffold** = برای جلوگیری از انحراف دستگاه ها از مدل کلی در داده های Non IID ، این روش از Control Varienes استفاده می کند تا جهت آپدیت ها را ردیابی و اصلاح کند تا از مدل کلی فاصله نگیرند .

محدودیت ها : پیاده سازی می تواند پیچیده باشد .

- **FedNova** = یک دستگاه ممکن است 100 تا epoch و یک دستگاه دیگر 10 تا epoch روی داده هاش انجام دهد ، خب برای اینکه سهم آن ها در به روز

رسانی مدل کلی متناسب با تعداد epoch ها یا اندازه batch نباشد ، FedNova با نرمال سازی آپدیت های هر دستگاه ، این اثر را خنثی می کند .

3. **استراتژی های کارآمد در ارتباطات :** الگوریتم هایی که سعی می کنند مصرف پهنای باند و هزینه تبادل داده را کم کنند :

- **Compression & Quantization :** دستگاه ها بجای ارسال وزن های دقیق (مثلا Float 32) از نسخه فشرده (مثلا 8 بیت یا حتی 1 بیت) استفاده می کنند . یا حتی می توانیم فقط بخش مهم گرادیان ها یا وزن ها را بفرستیم و گرادیان های کوچک را حذف کنیم .
- **انتخاب زیر مجموعه ای از دستگاه ها :** یعنی بجای اینکه در هر دور همه دستگاه ها شرکت کنند ، فقط یک زیر مجموعه تصادفی یا هوشمند انتخاب شود (همان منطق الگوریتم FedAvg) ، این طوری بار شبکه کاهش میابد .

4. **پروتکل های تقویت کننده حریم خصوصی :** حتی وقتی داده ها روی دستگاه های محلی می مانند ، باز هم Federated Learning ذاتا ایمن نیست ! چون یک مهاجم می تواند از روی گرادیان ها یا آپدیت های مدل ، اطلاعات حساس مثل داده های آموزشی یا هویت افراد را حدس بزند . برای رفع این خطر چند روش امنیتی به FL اضافه می کنند :

- **Differential Privacy :** روش کار بدین صورت است که به به روز رسانی های محلی (گرادیان ها یا وزن ها) قبل از فرستادن به سرور نویز تصادفی اضافه می کنند ، اینکار باعث می شود تا داده مربوط به یک نفر را نشود به طور دقیق بازسازی کرد .
- **Homomorphic Encryption :** دستگاه ها آپدیت ها را رمزنگاری هومورفیک می کنند و سرور می تواند روی داده رمزنگاری شده عملیات ریاضی (جمع ، ضرب و ...) انجام دهد بدون اینکه نیاز به باز کردن رمز باشد .

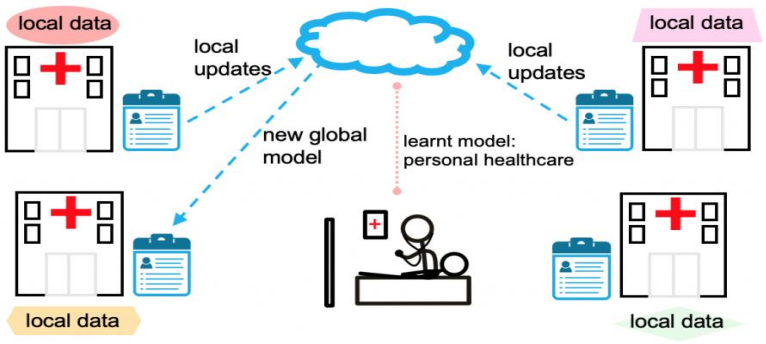
محدودیت ها : می تواند خیلی سنگین از نظر محاسبات باشد .

- **Secure Multi-Party Computation(SMPC)** : ایده این روش این است که آپدیت ها بین چند سرور تقسیم می شود و هیچکدام به تنهایی نمی توانند به داده اصلی دسترسی داشته باشند .

محدودیت ها : می تواند باعث تاخیر یا زمان بر شدن فرایند آموزش شود !

● کاربرد های عمومی Federated Learning :

1. **سلامت و پزشکی** : ساخت مدل های پیشبینی بیماری بدون اشتراک مستقیم داده های بیماران
2. **موبایل و اپلیکیشن** : شخصی سازی کیبورد ها برای پیشبینی کلمات ، سیستم های پیشنهادگر موزیک ، ویدیو ، فروشگاه ها و .... .
3. **خودرو های هوشمند** : آموزش مدل های رانندگی خودکار با داده های تولید شده خودرو های مختلف بدون انتقال حجم عظیم داده به مرکز .
4. **اینترنت اشیا** : تحلیل داده های سنسور ها و دستگاه ها (مثل خانه هوشمند) بدون ارسال داده خام به سرویس های ابری .
5. **مالی و بانکی** : کشف تقلب یا تحلیل ریسک با داده های توزیع شده بین بانک ها بدون نیاز به افشای اطلاعات مشتریان .



● کاربرد های تخصصی Federated Learning در امنیت :

1. **کشف بدافزار و حملات سایبری** : آموزش و ساخت مدل های تشخیص بدافزار بدون نیاز به انتقال لاگ ها یا داده های حساس شبکه به سرور مرکزی .
2. **تشخیص نفوذ (Intrusion Detection System(IDS))** : سازمان های مختلف می توانند با هم یک مدل IDS بسازند که حملات سایبری جدید را شناسایی کنند ، بدون اینکه لاگ های خام خود را افشا کنند .
3. **حفاظت از حریم خصوصی در تحلیل لاگ ها** : تحلیل لاگ ها برای کشف تهدید معمولا حاوی اطلاعات حساس است ، با FL می شود مدل های امنیتی را روی لاگ های محلی آموزش داد .

● ریسک های امنیتی در FL :

علاوه بر تهدید های حریم خصوصی ، سیستم های FL در معرض حملات مستقیم امنیتی هم هستند :

1. Poisoning Attacks :
   - Data Poisoning : کلاینت یا دستگاه خرابکار داده های جعلی یا برچسب های اشتباه توی دیتا خودش می زارد ، مثلا تو دیتای تشخیص اسپم ، پیام های اسپم را با برچسب ایمن وارد می کند !
   - Model Poisoning : بجای تغییر داده ، مهاجم مستقیما به روزرسانی مدل را دستکاری می کند .
2. Back door Atacks : مهاجم یک رفتار مخفی به مدل اضافه می کند که فقط در شرایط خاص فعال می شود ، مثلا مدل تشخیص چهره به طور عادی درست کار می کند اما اگر کسی عینک قرمز بزند ، همیشه آن فرد را به عنوان کاربر مجاز شناسایی کند .

● نتایج تجربی :

2NN	IID		Non-IID	
C	$B = \infty$	$B = 10$	$B = \infty$	$B = 10$
0.0	1455	316	4278	3275
0.1	1474 (1.0×)	87 (3.6×)	1796 (2.4×)	664 (4.9×)
0.2	1658 (0.9×)	77 (4.1×)	1528 (2.8×)	619 (5.3×)
0.5	— (—)	75 (4.2×)	— (—)	443 (7.4×)
1.0	— (—)	70 (4.5×)	— (—)	380 (8.6×)
CNN, $E = 5$				
0.0	387	50	1181	956
0.1	339 (1.1×)	18 (2.8×)	1100 (1.1×)	206 (4.6×)
0.2	337 (1.1×)	18 (2.8×)	978 (1.2×)	200 (4.8×)
0.5	164 (2.4×)	18 (2.8×)	1067 (1.1×)	261 (3.7×)
1.0	246 (1.6×)	16 (3.1×)	— (—)	97 (9.9×)

این جدول در رابطه با مقایسه دو مدل (یک شبکه ساده 2 لایه برای MNIST به نام 2NN و یک CNN) در حالت های IID و Non-IID داده است .

پارامتر های مهم :

- C ← کسری یا درصدی از دستگاه ها که در هر Round شرکت می کنند (مثلا 10٪)
- E ← تعداد Epoch های آموزش محلی روی هر کلاینت
- B ← سایز Batch محلی

مثلا در مدل 2NN ، IID ، وقتی c=0.1 و B = 10 باشد ، فقط 87 تا Round نیاز داره مدل .

✪ نکات کلیدی جدول :

- استفاده از Batch کوچک و چندEpoch محلی باعث می شود الگوریتم خیلی سریع تر همگرا شود (یعنی همان منطق الگوریتم FedAvg)
- وقتی داده ها Non-IID هستند ، الگوریتم FedAvg همچنان خوب عمل می کند .

MNIST CNN, 99% ACCURACY					
CNN	E	B	u	IID	NON-IID
FEDSGD	1	∞	1	626	483
FEDAVG	5	∞	5	179 (3.5×)	1000 (0.5×)
FEDAVG	1	50	12	65 (9.6×)	600 (0.8×)
FEDAVG	20	∞	20	234 (2.7×)	672 (0.7×)
FEDAVG	1	10	60	34 (18.4×)	350 (1.4×)
FEDAVG	5	50	60	29 (21.6×)	334 (1.4×)
FEDAVG	20	50	240	32 (19.6×)	426 (1.1×)
FEDAVG	5	10	300	20 (31.3×)	229 (2.1×)
FEDAVG	20	10	1200	18 (34.8×)	173 (2.8×)
SHAKESPEARE LSTM, 54% ACCURACY					
LSTM	E	B	u	IID	NON-IID
FEDSGD	1	∞	1.0	2488	3906
FEDAVG	1	50	1.5	1635 (1.5×)	549 (7.1×)
FEDAVG	5	∞	5.0	613 (4.1×)	597 (6.5×)
FEDAVG	1	10	7.4	460 (5.4×)	164 (23.8×)
FEDAVG	5	50	7.4	401 (6.2×)	152 (25.7×)
FEDAVG	5	10	37.1	192 (13.0×)	41 (95.3×)

این جدول شرحی از مقایسه دو مدل (MNIST CNN و Shakespear LSTM) بین FedSGD و FedAvg در حالت های مختلف است .

پارامتر جدید :

- U : تعداد دفعات محاسبه گرادیان محلی

✪ نکات کلیدی جدول :

- هر چه تعداد کلاینت یا همون دستگاه های بیشتری همزمان کار کنند ، به طور موازی آپدیت های بیشتری تولید می شود و تعداد Round های لازم کمتر می شود ، اما این کاهش تعداد Round ها زمانی چشمگیر می شود که مقدار B ، ∞ نباشد و کوچک باشد (منطق FedAvg)
- اثر جانبی مفید الگوریتم FedAvg : میانگیری مدل ها بین دستگاه ها مثل نوعی Regularization عمل می کند ، یعنی مشابه اثر Dropout در شبکه های عصبی ، به این معنی که مدل کمتر overfit می شود !
- افزایش U باعث کاهش شدید تعداد Round های لازم و هزینه ارتباطی می شود .

● Trade_offs در FL :

طراحی یک سیستم FL که همزمان امن ، سریع ، دقیق و مقیاس پذیر باشد ، کار آسانی نیست ! همیشه بین چند چیز باید توازن برقرار کنیم :

Trade-off	معنی ساده	مثال/توضیح
Accuracy vs Privacy	دقت مدل در مقابل حریم خصوصی	مثلا اگر نویز زیادی به مدل اضافه کنیم ، حریم خصوصی قوی می شود اما دقت مدل پایین میاد
Computation vs Communication	محاسبات در مقابل ارتباطات	استفاده از رمزنگاری یا جمع آوری امن باعث افزایش زمان پردازش و پهنای باند می شود
Security vs Scalability	امنیت در مقابل مقیاس پذیری	با روش هایی می شه امنیت مدل را بالا برد ولی وقتی تعداد دستگاه ها خیلی زیاد بشود ، مدیریتشون سخت می شود

هیچ سیستمی نمی تواند همزمان همه چیز را به حداکثر برساند ، بسته به کاربرد ، مهندسان هوش مصنوعی باید تعیین کنند تا چگونه مدل پیاده سازی شود ، مثلا در حوضه سلامت حریم خصوصی و دقت ممکنه مهمتر از سرعت باشد .

● نتیجه گیری :

Federated Learning نشان داد که نه‌تنها یک چارچوب تکنیکی برای آموزش مدل‌های یادگیری ماشین بدون انتقال داده خام است، بلکه یک تحول بنیادین در نگاه به امنیت و حریم خصوصی داده‌ها محسوب می‌شود. در حالی که روش‌های سنتی بر جمع‌آوری و متمرکزسازی داده‌ها تکیه داشتند و همین امر باعث افزایش ریسک نفوذ و نقض حریم خصوصی می‌شد ، FL با تمرکز بر محلی‌سازی داده‌ها و تبادل تنها به‌روزرسانی‌های مدل توانسته است توازنی میان کارایی، دقت و حفاظت از داده‌ها برقرار کند . در نهایت ، اهمیت واقعی Federated Learning در این است که

نشان می‌دهد هوش مصنوعی می‌تواند بدون نقض حریم خصوصی کاربران، یاد بگیرد و پیشرفت کند و این همان مسیر مطمئنی است که می‌تواند اعتماد عمومی به فناوری‌های هوشمند را بازسازی و تقویت نماید !

● منابع :

https://proceedings.mlr.press/v54/mcmahan17a/mcmahan17a.pdf

https://proceedings.mlsys.org/paper_files/paper/2019/file/7b770da633baf74895be22a8807f1a8f-Paper.pdf?utm_source=chatgpt.com

https://zenodo.org/records/15830919