Clase 226 — Federated learning: intro

Parte: 7 — Ética, Fairness y Privacidad · Fuente: McMahan et al. Communication-Efficient Learning of Deep Networks from Decentralized Data (AISTATS 2017) + Kairouz et al. Advances and Open Problems in Federated Learning (FnTML 2021). Duración estimada: 75 min.

Clase 226 — Federated learning: intro

> *Parte: 7 — Ética, Fairness y Privacidad · Fuente: McMahan et al. Communication-Efficient Learning of Deep Networks from Decentralized Data (AISTATS 2017) + Kairouz et al. Advances and Open Problems in Federated Learning (FnTML 2021). Duración estimada: 75 min.*

Objetivo

Entrenar un modelo central sin centralizar los datos. Cada cliente (móvil, hospital, banco) entrena local sobre su data, sube solo pesos o gradientes al servidor, que agrega vía FedAvg. Implementar FedAvg manual sobre regresión logística, ver cómo degrada con datos non-IID, y demostrar el ataque básico de gradient leakage (los gradientes filtran data).

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Explicar el setup FL: server + K clientes, rondas, partial participation, comunicación de pesos en vez de data.
- Implementar el algoritmo FedAvg (McMahan 2017): w_{t+1} = Σ_k (n_k / n) * w_k^t.
- Distinguir cross-device (millones de móviles, intermitentes) vs cross-silo (decenas de hospitales, estables).
- Reconocer el efecto de datos non-IID: FedAvg degrada cuando cada cliente ve una distribución distinta.
- Identificar los riesgos: model poisoning, gradient leakage (Zhu 2019), y las defensas (secure aggregation, DP-FedAvg, Krum).

Temas

#	Tema	Por qué importa
1	Setup FL: server + clientes + rondas	La data nunca sale del cliente; solo viaja
2	FedAvg: muestreo, local epochs, agregación	Es el baseline; todo lo demás se compara c
3	Cross-device vs cross-silo	Define el régimen: K=10^6 intermitentes vs
4	Heterogeneidad: non-IID + system + partial	El paper asume IID; la realidad no lo es.
5	Ataques: model poisoning, gradient leakage	Compartir gradientes ≠ proteger data (Zhu
6	Defensas: secure aggregation, DP, Krum/Mec	Lo que se usa en producción real (Google G

Definiciones y características

- Federated learning (FL): paradigma de entrenamiento donde K clientes con data local entrenan un modelo conjunto coordinados por un servidor, sin compartir data cruda.
- FedAvg: el algoritmo base (McMahan 2017). Cada ronda: server envía pesos w_t; un subset de clientes corre E épocas de SGD local; server promedia: w_{t+1} = Σ (n_k / n) * w_k.
- Cross-device FL: K = millones de teléfonos, clientes intermitentes (sin conexión, batería baja), participación parcial obligatoria. Ejemplo: Gboard de Google.
- Cross-silo FL: K = decenas de organizaciones (hospitales, bancos), clientes estables y siempre online.

Ejemplo: consorcios médicos contra cáncer.

- Non-IID data: la distribución P_k(x, y) cambia por cliente. Caso típico: cada hospital ve una mezcla distinta de patologías. FedAvg degrada y oscila.
- Gradient leakage: dado un gradiente compartido (especialmente con batch chico), un atacante puede reconstruir el input original vía optimización inversa (Zhu, Liu, Han, Deep Leakage from Gradients, NeurIPS 2019).
- Secure aggregation (Bonawitz 2017): protocolo criptográfico donde el server solo ve la suma de pesos de N clientes, nunca los individuales — anula el gradient leakage contra el server.
- DP-FedAvg: agregar ruido gaussiano calibrado a los pesos antes de subirlos → garantías formales de differential privacy (Clase 225) sobre la presencia de un cliente.

Dataset / recursos

- Dataset: sintético tabular (clasificación binaria, 2000 muestras × 10 features), split en K=10 clientes. Self-contained, sin descargas.
- Librerías: numpy, scikit-learn. Implementamos FedAvg manualmente — sin flower, PySyft ni TensorFlow Federated, para que el algoritmo quede claro.

Ejercicios

1. Particionado IID: generar 2000 muestras, split aleatorio uniforme en K=10 clientes. Verificar que cada cliente tiene ~200 muestras con clases balanceadas.
2. Local training: implementar local_train(X, y, w, epochs=5, lr=0.05) — regresión logística con SGD manual. Devuelve nuevos pesos w_k.
3. FedAvg loop: 20 rondas. En cada ronda, samplear 5 de 10 clientes; entrenar local; agregar vía fedavg(weights, sizes). Trackear loss global.
4. Centralizado vs federado: entrenar la misma logística sobre todo el data junto. Verificar que FL converge a una accuracy comparable (gap < 0.03 en setting IID).
5. Non-IID: re-partir asignando a cada cliente solo 1-2 clases (cliente 0 ve mayoritariamente clase 0, etc.). Correr FedAvg. Mostrar que la accuracy global degrada y oscila más.

Homework verificable

Notebook con:

1. Implementar FedAvg + variante FedProx (agrega regularización μ/2 · ||w_k - w_t||² al loss local — Li et al. 2020) sobre el mismo dataset non-IID.
2. Comparar curvas de loss FedAvg vs FedProx vs central (3 curvas).
3. Implementar DP-FedAvg: ruido gaussiano σ=0.01 sobre los pesos agregados. Medir pérdida de accuracy.
4. Reproducir gradient leakage simple sobre 1 muestra: dado el gradiente de regresión lineal con batch=1, recuperar x vía minimización de ||_w L(w; x_hat, y) - g||².
5. Discutir (≤200 palabras): cuándo FL paga el costo de comunicación vs entrenamiento central + acuerdo de data sharing.

Criterio de aceptación: FedProx supera a FedAvg en setting non-IID (loss final menor); DP-FedAvg con σ=0.01 pierde <0.05 de accuracy; reconstrucción de x con MSE < 0.01 respecto al original.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
FedAvg diverge en non-IID	Demasiadas épocas locales → client drift.
Loss global oscila ronda a ronda	Sampleo de pocos clientes por ronda. Fix:
Cliente con n_k=0 rompe la agregación	División por cero en pesos. Fix: filtrar c
Promedio simple en vez de weighted	Clientes con poca data pesan igual que los
"FL = privado por default"	Falso. Los pesos filtran data (Zhu 2019).
Pesos cargados como listas python en el se	Costo de comunicación se dispara. Fix: env

Preguntas frecuentes

¿FL es differential privacy?

No, son ortogonales. FL no centraliza data, pero los pesos compartidos pueden filtrar muestras individuales. DP da garantías formales (Clase 225). Producción real usa FL + secure aggregation + DP combinados.

¿Cuándo elegir FL en vez de entrenar centralizado?

Cuando la data no puede moverse: regulación (HIPAA, GDPR), data en edge (teclados, wearables), competidores que quieren modelo conjunto sin compartir clientes. Si podés centralizar, centralizá — es 10-100× más barato en comunicación y converge mejor.

¿Cuánto cuesta la comunicación?

Es el cuello de botella real. Cada ronda envía O(|w|) bytes por cliente — para una red de 10M params en float32 son 40 MB por cliente, por ronda. Por eso existen técnicas de compresión, quantization y top-k sparsification.

¿FedAvg está obsoleto?

Sigue siendo el baseline. Variantes modernas: FedProx (regularización local, 2020), SCAFFOLD (control variates para non-IID, 2020), FedOpt (Adam/Yogi del lado server, 2021). Todas se comparan contra FedAvg.

¿Qué framework usar en producción?

Flower es agnóstico al ML framework y el más adoptado en investigación reciente. TensorFlow Federated si ya estás en TF. PySyft si querés combinar con SMPC/HE. Para móvil: TFF Lite y la stack interna de Google (no abierta).

Referencias

- McMahan, B., Moore, E., Ramage, D., Hampson, S., Arcas, B. Communication-Efficient Learning of Deep Networks from Decentralized Data (AISTATS 2017) — el paper original de FedAvg: <https://arxiv.org/abs/1602.05629>.
- Kairouz, P. et al. Advances and Open Problems in Federated Learning (FnTML 2021) — survey canónico, 50+ autores: <https://arxiv.org/abs/1912.04977>.
- Zhu, L., Liu, Z., Han, S. Deep Leakage from Gradients (NeurIPS 2019) — el ataque de reconstrucción desde gradientes: <https://arxiv.org/abs/1906.08935>.
- Bonawitz, K. et al. Practical Secure Aggregation for Privacy-Preserving Machine Learning (CCS 2017) — el protocolo de Google.

- Flower framework — librería FL agnóstica al framework ML.
- Li, T., Sahu, A., Talwalkar, A., Smith, V. Federated Learning: Challenges, Methods, and Future Directions (IEEE Signal Processing, 2020).

Siguiente clase

Clase 227 — GDPR y AI Act (EU)

Apéndice: notebook (primer bloque)

Self-contained: solo numpy + sklearn. Simulamos FL a mano — sin flower, PySyft ni TFF — para que el algoritmo quede transparente. Seed 42.

```
import numpy as np
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

rng = np.random.default_rng(42)

X, y = make_classification(
    n_samples=2000, n_features=10, n_informative=6,
    n_redundant=2, n_classes=2, random_state=42,
)
X = (X - X.mean(0)) / X.std(0)
# Columna de bias (sesgo) para regresión logística manual
X_b = np.hstack([X, np.ones((X.shape[0], 1))])
n, d = X_b.shape
print(f'dataset: X={X_b.shape}, y={y.shape}, balance={y.mean():.3f}')
```

Archivos complementarios

- notebook.ipynb