Clase 225 — Privacidad diferencial: intro

Parte: 7 — Ética, Fairness y Privacidad · Fuente: Dwork & Roth, The Algorithmic Foundations of Differential Privacy (2014) caps. 2-3 + Dwork, McSherry, Nissim, Smith (TCC, 2006) Calibrating Noise to Sensitivity. Duración estimada: 75 min.

Clase 225 — Privacidad diferencial: intro

> *Parte: 7 — Ética, Fairness y Privacidad · Fuente: Dwork & Roth, The Algorithmic Foundations of Differential Privacy (2014) caps. 2-3 + Dwork, McSherry, Nissim, Smith (TCC, 2006) Calibrating Noise to Sensitivity. Duración estimada: 75 min.*

Objetivo

Entender privacidad diferencial (DP) como la única definición formal de privacidad con garantías matemáticas — no "anonimización" heurística que se rompe con un join. Implementar el mecanismo de Laplace desde cero, observar el trade-off privacy-utility vía el presupuesto ε (epsilon), y mirar conceptualmente DP-SGD (Abadi et al. 2016): cómo se entrena un modelo sin que un atacante pueda inferir si tu registro estuvo en el training set.

Resultados de aprendizaje

Al finalizar, el estudiante podrá:

- Enunciar la definición de (ε, δ)-DP: P[M(D) S] ≤ e^ε · P[M(D') S] + δ para datasets vecinos D, D'.
- Calcular la sensibilidad Δf de funciones típicas (conteos, sumas acotadas, medias) y elegir ruido Laplace o Gaussiano calibrado.
- Implementar laplace_mechanism(value, sensitivity, epsilon) y verificar que ε chico → más ruido → menos utilidad.
- Aplicar composición básica: k consultas con ε cada una gastan k·ε del presupuesto total.
- Reconocer la idea de DP-SGD: per-sample gradient clipping + ruido gaussiano → entrenamiento DP (Opacus, TF-Privacy).

Temas

#	Tema	Por qué importa
1	Anonimización falla (Netflix Prize, AOL se	k-anonymity / pseudonimización son rotas p
2	Definición (ε, δ)-DP y datasets vecinos	El ε es la garantía; sin él, "privacidad"
3	Sensibilidad Δf	Calibra cuánto ruido hace falta. Conteo: Δ
4	Mecanismos Laplace y Gaussiano	Laplace para ε-DP puro; Gaussiano para (ε,
5	Composición y post-processing	Cada query gasta presupuesto; cualquier f(
6	DP-SGD (Abadi 2016)	Clip per-sample + ruido gaussiano. Es el e

Definiciones y características

- (ε, δ)-Differential Privacy (Dwork 2006): mecanismo aleatorizado M es (ε, δ)-DP si para todo par de datasets vecinos D, D' (que difieren en 1 registro) y todo conjunto de salidas S: P[M(D) S] ≤ e^ε · P[M(D') S] + δ. Si δ = 0 se llama ε-DP puro.
- Privacy budget ε: chico = más privacidad, menos utilidad. Valores típicos en la práctica: ε=0.1 (fuerte), ε=1 (estándar de la US Census 2020), ε=10 (débil — más marketing que garantía).
- δ: probabilidad de fallar la garantía. Regla: δ << 1/n (con n = tamaño del dataset).

- Sensibilidad Δf (L1): Δf = max_{D,D' vecinos} |f(D) − f(D')|. Conteo de registros: Δf=1. Suma de valores en [0, B]: Δf=B. Media sobre n fijo y valores en [0, B]: Δf = B/n.
- Mecanismo de Laplace: M(D) = f(D) + Lap(0, Δf/ε). Cumple ε-DP puro.
- Mecanismo Gaussiano: M(D) = f(D) + N(0, σ²) con σ ≥ sqrt(2 ln(1.25/δ)) · Δf / ε. Cumple (ε, δ)-DP.
- Composición básica: k mecanismos ε_i-DP componen a (Σ ε_i)-DP. Composición avanzada da sqrt(2k ln(1/δ)) · ε — mejor escala.
- Post-processing: si M es (ε, δ)-DP, entonces g(M(D)) también — no podés "des-privatizar" mirando la salida.
- DP-SGD (Abadi et al. 2016): en cada paso (1) calcular gradiente per-sample, (2) clipearlo a norma C, (3) promediar el batch, (4) sumar ruido gaussiano N(0, σ²C²). Tracking del ε vía moments accountant / RDP.

Dataset / recursos

- Dataset: sintético — un dataframe de salarios n=10_000 con valores en [0, 200_000]. Suficiente para Laplace, mean privado, histograma y DP-SGD demo. Sin descarga externa.
- Librerías: numpy, pandas, scikit-learn. En producción real: opacus (PyTorch), tensorflow-privacy, diffprivlib (IBM).

Ejercicios

1. Laplace básico: implementar laplace_mechanism(value, sensitivity, epsilon) y verificar empíricamente sobre 10_000 corridas que la varianza es 2·(Δf/ε)².
2. Conteo privado: contar empleados con salario > 100k con ε {0.1, 1.0, 10.0}. Reportar error medio absoluto y discutir el trade-off.
3. Mean privado con clipping: clip salarios a [0, B], sumar con Laplace (Δf=B/n, ε=1), dividir por n. Mostrar bias vs varianza al variar B.
4. Histograma privado: 10 bins de salario, ruido Laplace independiente por bin (sensibilidad = 1 por bin). Comparar con histograma no privado.
5. Composición: hacer 10 conteos con ε=0.1 cada uno → presupuesto total ε=1.0. Mostrar acumulación empírica del ruido.

Homework verificable

Notebook con:

1. Cargar Adult / Census Income (UCI, ~32K filas).
2. Publicar un dashboard DP con 5 estadísticas (count, mean age, mean hours-per-week, count por género, count por education) bajo presupuesto total ε=1.0. Repartir el presupuesto entre queries y justificar.
3. Entrenar un LogisticRegression clásico para predecir income > 50k, reportar accuracy.
4. Re-entrenar con DP-SGD manual: clip per-sample gradient norm a C=1.0, sumar N(0, σ²) con σ=1.0. Reportar accuracy y comparar.
5. Discutir: ¿cuánta utilidad perdés? ¿el modelo DP es publicable sin riesgo de membership inference?

Criterio de aceptación: el dashboard cumple ε=1.0 total (verificable sumando los ε_i), el modelo DP-SGD entrena sin error y la pérdida de accuracy vs no-DP es < 10 pp.

Errores comunes

Síntoma / mensaje	Causa y cómo arreglar
"Privatizo la salida pero el atacante reco	Olvidaste clippear la entrada — un outlier
Calculo Δf de una media como B (no B/n)	Confundís sensibilidad de suma vs media. F
Hago 100 queries con ε=1 y digo "es ε=1-DP	Sin tracking, gastaste ε=100 por composici
ε=10 o ε=20 "porque así da mejor utility"	ε≥10 da garantía prácticamente nula (e^10
DP-SGD sin clippear per-sample	Sin clipping, la sensibilidad del gradient
Reutilizar el dataset privado para "valida	El proceso de validación también gasta bud

Preguntas frecuentes

¿Qué ε es "seguro"?

No hay un número universal. La US Census 2020 usó ε ≈ 19.6 (TopDown algorithm, criticado por flojo). Apple iOS reporta ε por feature (típicamente 2-8 por día). Recomendación práctica: empezar en ε=1, justificar cualquier valor mayor. ε≥10 es difícil de defender ante un comité de ética.

¿DP me protege de TODO ataque?

Te protege contra membership inference y reconstruction bajo el modelo de atacante con conocimiento auxiliar arbitrario. NO te protege contra: ataques al modelo no-DP entrenado paralelamente, side-channels (timing), o si el atacante tiene el dato original (no es encriptación).

¿Local DP vs Central DP?

Central DP: confiás en el curador (el servidor agrega ruido). Más utilidad. Local DP: cada usuario agrega ruido antes de mandar (Apple, RAPPOR de Google). Menos utilidad, pero no confiás en nadie. La elección depende del modelo de amenaza.

¿Vale la pena en deep learning?

Sí, con caveats. DP-SGD penaliza accuracy (5-15 pp típicos), y necesita batches grandes para que el ruido se promedie. Opacus / TF-Privacy automatizan todo. Es obligatorio si vas a publicar el modelo o usar datos médicos/financieros bajo regulación.

¿Y federated learning?

Federated learning (Clase 226) por sí solo NO es DP — el server ve gradientes que filtran información. Se combina con secure aggregation + DP para garantías reales (lo que hace Google Gboard).

Referencias

- Dwork, McSherry, Nissim, Smith. Calibrating Noise to Sensitivity in Private Data Analysis (TCC 2006) — el paper original que define DP.
- Dwork, C., Roth, A. The Algorithmic Foundations of Differential Privacy (Foundations and Trends, 2014) — el libro de referencia.
- Abadi et al. Deep Learning with Differential Privacy (CCS 2016) — DP-SGD + moments accountant.
- Opacus — DP-SGD en PyTorch (Meta).
- TensorFlow Privacy — DP-SGD en TF/Keras (Google).
- IBM diffprivlib — mecanismos básicos sklearn-compatible.

Siguiente clase

Clase 226 — Federated learning: intro

Apéndice: notebook (primer bloque)

Sintético: dataset de salarios n=10_000. Requiere: pip install numpy pandas scikit-learn.

```
import numpy as np, pandas as pd

rng = np.random.default_rng(42)
n = 10_000
B = 200_000   # cota superior salario (clipping para sensibilidad acotada)
salaries = np.clip(rng.lognormal(mean=10.8, sigma=0.6, size=n), 0, B)
df = pd.DataFrame({'salary': salaries})
print(f'n={n:,} | mean real={salaries.mean():,.0f} | >100k real={(salaries > 100_000).sum():,}')
```

Archivos complementarios

- notebook.ipynb