CS839: Differential Privacy and Learning March 12, 2026

Lecture 16: DP-FTRL and Correlated Noise

Instructor: Gavin Brown *Scribe:* Jingyi Gao

Disclaimer: This document is intended as an informal supplement to in-class note-taking. It has not been given the level of scrutiny expected in polished lecture notes, let alone that reserved for peer-reviewed publications.

In this lecture, we explore correlated noise mechanisms, the most well-known example of which is DP-Follow-The-Regularized-Leader (DP-FTRL). To lay the groundwork for understanding correlated noise in Differentially Private Gradient Descent (DP-GD), we first review the factorization mechanism, the binary tree mechanism, and prefix sums.

1 Factorization for Linear Queries

Definition 1.1 (Linear Queries)**.** We want to answer k linear queries $f_1, f_2, \ldots, f_k$ over a dataset $x_1, \ldots, x_n \in \mathcal{U}$, where $|\mathcal{U}| = m$. For each $j \in [k]$, the query is defined as $f_j(x) = \frac{1}{n}\sum_{i=1}^{n} \varphi_j(x_i)$, with predicates $\varphi_j : \mathcal{U} \to \{0, 1\}$. The overall workload is a function $F : \mathcal{U}^n \to [0, 1]^k$.

To rewrite this in matrix notation:

- We represent the data as a histogram $h_x \in [0, 1]^m$, where $(h_x)_u = \frac{1}{n}|\{i \in [n] : x_i = u\}|$.
- We represent each predicate as a vector $v_{\varphi_j} \in \{0, 1\}^m$, where $v_{\varphi_j} = (\varphi_j(u_1), \varphi_j(u_2), \ldots, \varphi_j(u_m))$.
- Consequently, each query can be written as an inner product: $f_j(x) = \langle v_{\varphi_j}, h_x \rangle$.
- Our workload matrix F is constructed by stacking these predicate vectors:

$$F = \begin{bmatrix} v_{\varphi_1} \\ v_{\varphi_2} \\ \vdots \\ v_{\varphi_k} \end{bmatrix}$$

Our goal is to accurately approximate Fh_x.

The standard Gaussian mechanism would simply return $Fh_x + Z$, where $Z \sim \mathcal{N}(0, \sigma^2 I)$. However, this may be less accurate than necessary. For example, if all queries are identical, the basic Gaussian mechanism would redundantly add independent noise to each repeated query. The **factorization mechanism**, on the other hand, can evaluate the query once and reuse the result for the remaining queries.

The factorization mechanism operates as follows:

1. Find matrices R (Reconstruction) and M (Measurement) such that $F = RM$.
2. Return $R(Mh_x + Z)$.

If we expand the factorization mechanism's output:

$$R(Mh_x + Z) = RMh_x + RZ = Fh_x + \mathcal{N}(0, \sigma^2 RR^T)$$

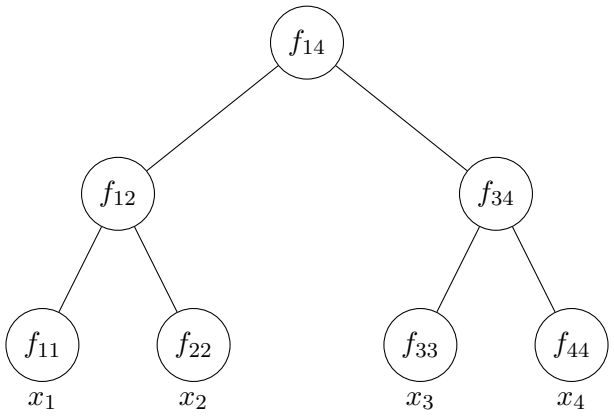
From the covariance matrix of the noise ($\sigma^2 RR^T$), we can clearly see that the answers for all queries will be correlated with one another.

In Lecture 6 we covered the accuracy and privacy considerations of this approach, but we won't need this information today.

2 Binary Tree Mechanism & Prefix Sums

We now discuss an important instance of this approach.

Suppose we have T individuals, each holding a single bit x_t. Our task is to estimate the prefix sum $S_t = \sum_{\tau=1}^{t} x_\tau$ for all $t = 1, \ldots, T$. This fits naturally into the linear query framework.



Recall that for interval queries, we want to answer $f_{s,t}(x) = \sum_{\tau=s}^{t} x_\tau$. From Lecture 5, we learned about the Binary Tree Mechanism, and the specific procedure is as follows:

1. Add Lap $\left(\frac{O(\log T)}{\varepsilon}\right)$ noise to each node in the tree. This preserves DP, since changing a single bit can change $\log T$ counts (i.e., this is a bound on global sensitivity).
2. **Measure:** For each node in the tree, obtain noisy estimates $\widetilde{f}_{1,1}, \widetilde{f}_{1,2}, \widetilde{f}_{2,2}, \ldots$
3. **Reconstruct:** Calculate the desired prefix sums using the tree's nodes. For example:
   (a) $\widetilde{f}_{1,1}$
   (b) $\widetilde{f}_{1,2}$
   (c) $\widetilde{f}_{1,3} = \widetilde{f}_{1,2} + \widetilde{f}_{3,3}$
   (d) $\widetilde{f}_{1,4} = \widetilde{f}_{1,2} + \widetilde{f}_{3,4}$

   Notice that $\widetilde{f}_{1,2}$ is used in both the second and third reconstruction steps. Because of this reuse, the noise across these reconstructed answers is inherently correlated.

Remark 2.1. 1. Each output relies on at most $\log T$ noise variables, meaning the total accumulated error is polylog(T).

2. The noise is correlated.
3. There is nothing restrictive about the answers being single bits; this mechanism easily extends to multivariate settings where $x_i \in \mathbb{R}^d$.
4. Answers can be provided sequentially. Because we do not need to observe future data points to compute the current prefix sum, this serves naturally as a streaming or online algorithm.

3 Correlated Noise for DP-(S)GD

Recall that for DP-GD, we start with an initialized parameter θ_0. Our subsequent iterates are computed as:

$$\theta_1 = \theta_0 - \eta g_0$$
$$\theta_2 = \theta_1 - \eta g_1 = \theta_0 - \eta(g_0 + g_1)$$

In general, the update rule at step t is:

$$\theta_t = \theta_0 - \eta \sum_{\tau=0}^{t-1} g_\tau$$

This reveals that the gradient update is effectively a **prefix sum**! Therefore, our focus shifts: instead of trying to estimate each individual gradient g_t perfectly (the old goal), we now want to estimate the prefix sum $S_t := \sum_{i=1}^{t} g_i$ as accurately as possible (the new goal).

If we define the gradient matrix G and the prefix-sum matrix S as:

$$G = \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{t-1} \end{bmatrix}, \quad S = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_t \end{bmatrix}$$

We can express the relationship as $S = AG$, where A is the well-studied prefix-sum workload matrix:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ 1 & 1 & 0 & 0 & \dots & 0 \\ 1 & 1 & 1 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & 1 & \dots & 1 \end{bmatrix}$$

Applying the factorization mechanism here means we factor $A = BC$ and release:

$$B(CG + Z) = BCC^{-1}(CG + Z) = A(G + C^{-1}Z)$$

where Z is Gaussian noise. If we take C to be the identity matrix, we recover the standard mechanism we analyzed previously.

This is a clean formula, but it hides a lot of complications, some of which are active research questions.

1. **Adaptivity:** It is not immediately obvious whether this streaming process maintains privacy; unlike our previous privacy analysis for the factorization mechanism, the gradients (which in this setting play the role of *data points*) are chosen adaptively. (Though the privacy analysis has been solved, the analysis is non-trivial.)
2. **Factorization Quality:** It is unclear exactly what constitutes a "good" factorization $A = BC$ that minimizes error. A natural approach is to try to make each prefix-sum as accurate as possible, but this may not be optimal.
3. **Computation:** Two main questions here.
    - Can we efficiently compute a good factorization? Finding optimal matrices often requires solving a computationally intensive semidefinite program (SDP), whose size scales with the number of steps needed in the optimization process.
    - Can we efficiently generate the correlated noise during the training process? This could easily produce a large computational overhead at each training step.

This is a highly active research area. The work of Kairouz et al. [2021] introduced the example of **DP-Follow-The-Regularized-Leader (DP-FTRL)**, which is the most well-known version of DP training with correlated noise. For further literature review, technical details, and open problems, see the recent monograph of Pillutla et al. [2025].

References

Peter Kairouz, Brendan McMahan, Shuang Song, Om Thakkar, Abhradeep Thakurta, and Zheng Xu. Practical and private (deep) learning without sampling or shuffling. In *International Conference on Machine Learning*, pages 5213–5225. PMLR, 2021.

Krishna Pillutla, Jalaj Upadhyay, Christopher A Choquette-Choo, Krishnamurthy Dvijotham, Arun Ganesh, Monika Henzinger, Jonathan Katz, Ryan McKenna, H Brendan McMahan, Keith Rush, et al. Correlated noise mechanisms for differentially private learning. *arXiv preprint arXiv:2506.08201*, 2025.