Improved Differential Privacy for SGD via Optimal Private Linear Operators on Adaptive Streams

Anonymous Author(s)
Affiliation
Address
email

Abstract

Motivated by recent applications requiring differential privacy over adaptive streams, we investigate the question of optimal instantiations of the matrix mechanism [1] in this setting. We prove fundamental theoretical results on the applicability of matrix factorizations to the adaptive streaming setting, and provide a parameter-free fixed-point algorithm for computing optimal factorizations. We instantiate this framework with respect to concrete matrices which arise naturally in machine learning, and train user-level differentially private models with the resulting optimal mechanisms, yielding significant improvements in a notable problem in federated learning with user-level differential privacy.

1 Introduction and background

An important setting for private data analysis is that of streaming inputs and outputs—often dubbed *continual release*. Hiding individual information is especially challenging in such settings since the arrival of one person's data may affect all future outputs of the system. A significant line of work formalizes *differential privacy* (DP, [2])) under continual release and builds algorithms that meet the resulting definition (e.g. [3–8]). One prominent application of private, continual-release algorithms is to adapt iterative optimization algorithms such as SGD so that their outputs satisfy DP.

The problem of privately computing *cumulative sums* plays a key role in both theory and applications. Given a set of vectors $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in $\mathbf{x}_i \in \mathbb{R}^d$, the task is to approximate the sequence of prefix sums $(\mathbf{x}_1, \mathbf{x}_1 + \mathbf{x}_2, \ldots, \mathbf{x}_1 + \cdots + \mathbf{x}_n)$ while satisfying DP. Solutions to this task form the core building block in DP algorithms for online PCA [9], online marginal estimation [4, 3, 10], online top-k selection [11], and training ML models [12, 6, 13], among others. For example, a common approach to private optimization is to add noise to the gradient estimates in SGD[14–16]. Kairouz et al. [6] make the observation that the key DP primitive in such contexts is not the independent estimation of individual gradients, but rather the accurate estimation of cumulative sums of gradients. Lowering the error of the DP algorithm's approximation to the cumulative sum translates directly to improved optimization.[1]

The structure of continual-release algorithms imposes two major constraints: first, the algorithm must be computable online—that is, we must produce the output at a given time using only prior inputs—and, second, privacy analysis must account for *adaptively defined inputs*—that is, the guarantee should hold even against an adversary that selects inputs based on all previous outputs of the system.

[1] SGD with constant learning rate η serves as an intuitive illustration: performing SGD on parameters θ starting from $\mathbf{0}$, the t-th iterate is simply $\theta_t = -\eta \sum_{i=1}^{t} g_i$, where g_i is the gradient computed on step i. That is, the learned model parameters θ_t are exactly a scaled version of the cumulative sum of gradients so far. It is the total error in these cumulative sums that matters most, not the error in the private estimates of each individual g_i. See Theorem 5.1 of [6] for a formal statement.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

In learning applications, the privacy analysis must be adaptive even when the actual input stream is not, because of the way a continual-release DP primitive is used inside the larger algorithm [13].

In this paper, we revisit the design of continual-release algorithms for cumulative sums and related problems: we give new tools for analyzing privacy in the adaptive setting, new methods to design optimal (within a class) algorithms for summation-style problems, and applications to central problems in private machine learning. Although we focus on learning as the primary application of our algorithmic and analytic tools, our techniques apply to a wide range of private computations over streaming data such as online monitoring [17, 18], tracking distributional changes over data streams [19], and detection of emerging trends [20].

Prior Approaches. Prior approaches to DP approximation of cumulative sums fall broadly into two categories. First, in streaming settings, existing work is generally based on the *binary-tree estimator* [4, 3]. This estimator embeds the values to be summed as leaf nodes in a complete binary tree $\mathcal{T}$, with internal nodes representing the sum of all leaves below them. The mechanism views the *entire tree* as the object to be privately released (ensuring privacy by adding independent noise to each node). An important refinement of Honaker [21] leverages the multiple noisy observations of the same value thus introduced to produce lower-variance estimates of the appropriate prefix sums. Kairouz et al. [6] apply Honaker's online variant, dubbed `Honaker Online`, to the *follow-the-regularized-leader* approach to optimization [22–24]. The tree structure of these mechanisms allows for privacy analysis in the adaptive setting, as observed by [13] and formalized by [8].

The second, more general approach to cumulative sums has previously only been applied in *offline* settings, in which the input is received and outputs are produced as a single batch. The idea is to view cumulative sums as a special case of linear query release (since each output is a pre-specified linear combination of the inputs) and the tree-based approaches as instantiations of the widely-studied *factorization* framework (as in,for example, [25, 1, 26–28]). Consider the general task of computing a private estimate of a linear mapping $\mathbf{x} \mapsto \mathbf{A}\mathbf{x}$. Given any factorization $\mathbf{A} = \mathbf{B}\mathbf{C}$, a DP estimate of $\mathbf{A}\mathbf{x}$ can be computed as

$$\widehat{\mathbf{A}\mathbf{x}} = \mathbf{B}\left(\mathbf{C}\mathbf{x} + \mathbf{z}\right) \tag{1}$$

where $\mathbf{z}$ represents a sample from a noise distribution $\mathcal{D}$. Choosing $\mathcal{D}$ in a way that appropriately depends on the sensitivity of the map $\mathbf{x} \mapsto \mathbf{C}\mathbf{x}$, one can prove privacy of the noised vector $\mathbf{C}\mathbf{x} + \mathbf{z}$, and hence of the mapping $\mathbf{x} \mapsto \widehat{\mathbf{A}\mathbf{x}}$. For example, in the case of cumulative sums, the matrix $\mathbf{A}$ is the lower-triangular matrix $\mathbf{S}$ with 1's on and below the diagonal; the binary tree mechanisms correspond to a matrix $\mathbf{C}_{\mathcal{T}}$ with one row per tree node. A typical choice for the noise $\mathbf{z}$ is to select it from a spherical Gaussian distribution.

A focus of existing work (e.g. [1, 26–28]) is to choose the factorization $\mathbf{BC}$ to optimize some measure of overall accuracy (such as total mean squared error) subject to the privacy constraint on $\mathbf{C}$. However, with the exception of the independent, parallel work of Fichtenberger et al. [29], streaming constraints and adaptive privacy were not explicitly considered. In the streaming setting, one naturally requires the ith element of $\mathbf{A}\mathbf{x}$ to be computable using only the first i elements of $\mathbf{x}$. This corresponds to requiring that the linear operator of interest $\mathbf{A}$ has a lower-triangular structure when represented as a matrix, a requirement that is met by all the matrices under consideration in this work. Distributing the $\mathbf{B}$ in Eq. (1), we see $\widehat{\mathbf{A}\mathbf{x}} = \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{Z}$. As long as $\mathbf{A}\mathbf{x}$ is lower triangular, then, *any* matrix mechanism can be implemented by an online algorithm, via the distribution induced by $\mathbf{B}\mathbf{z}$. However, this observation does not address a key problem: under what conditions is the resulting mechanism *adaptively* private?

The applications to optimization raise their own set of critical questions: How can we efficiently compute such factorizations? Which linear operators are the appropriate ones to factorize in the case of SGD? Finally, can these factorizations actually produce improved privacy/accuracy tradeoffs for real-world machine learning tasks?

Contributions. We provide a deeper understanding of the matrix mechanism in the adaptive streaming setting. We show that if $\mathbf{z}$ in Eq. (1) is drawn from a Gaussian distribution of appropriately computed variance, then the resulting mechanism is differentially private in the adaptive streaming setting, *independent of the structure of* $\mathbf{B}$ *and* $\mathbf{C}$. We make an explicit connection here with the easier-to-see privacy under adaptive streams of lower-triangular factorizations. Furthermore, we

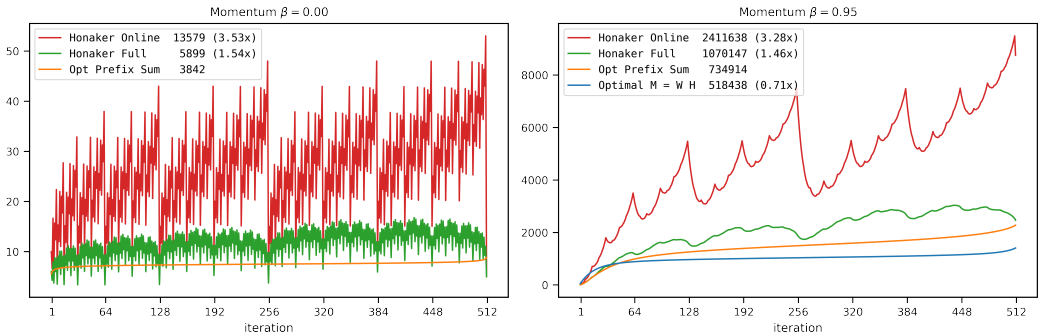

Figure 1: Left: Per-iteration squared error of three mechanisms for DP online prefix sums. `Honaker Online` and `Honaker Full` correspond to DP binary tree aggregation with the streaming and full Honaker estimators (respectively), while `Opt Prefix Sum` corresponds to the optimal matrix factorization. The tree-based mechanisms suffer from variability in the error due to the binary tree structure. Right: Squared error in the DP estimates of the iterates of momentum SGD for four mechanisms. Momentum is treated as post-processing of cumulative sums for the first 3 mechaisms, while `Optimal M = W H` uses the optimal factorization of the momentum matrix (see Section 4).

show that this property is specific to the Gaussian mechanism, and does not extend to arbitrary noise distributions.

For natural notions of error and adjacency specified in Section 3, we present a fast and parameter-free fixed-point algorithm for computing optimal factorizations and prove a local convergence guarantee for this algorithm, leveraging representations of optimal factorizations which are to our knowledge novel in the literature on the matrix mechanism. The optimal computed factorizations show a significant improvement over existing state-of-the-art private streaming prefix sum methods [21], additionally removing the artifacts of the binary tree data structure (see Fig. 1). Furthermore, our fixed-point algorithm is two orders of magnitude more computationally efficient than existing optimization methods for the matrix mechanism [26, 27].

Going beyond prefix sums (which correspond to constant learning rate SGD as noted above), we construct matrix mechanisms that directly encode more sophisticated optimization algorithms directly as linear operators on gradients: in particular, arbitrary combinations of (data independent) learning rate schedules and momentum.

We compute optimal factorizations of these general matrices via fixed-point iterations, and use them to train user-level differentially private language models on a canonical federated learning benchmark, showing that these factorizations significantly improve the privacy/utility curve (in fact, closing 2/3rds of the gap to non-private training left by the previous state-of-the-art for single pass algorithms).

Notation. Matrices will be denoted by bolded capital letters (e.g. $\mathbf{A}$, $\mathbf{B}$), with some symbols reserved for special matrices, notably $\mathbf{S}$ (prefix sums) and $\mathbf{M}$ (momentum, defined in Section 4 and illustrated in Appendix A). Vectors will be denoted by bolded lowercase letters (e.g. $\mathbf{x}$, $\mathbf{y}$). All matrices, unless otherwise stated, are assumed to be square matrices of some fixed dimensionality n (that is, elements of $\mathbb{R}^{n\times n}$). Vectors are similarly assumed to be elements of $\mathbb{R}^n$. For a real symmetric matrix $\mathbf{A}$, the smallest and the largest eigenvalues are denoted by $\lambda_{\min}(\mathbf{A})$ and $\lambda_{\max}(\mathbf{A})$.

2 Privacy for adaptive streams

In this section, we provide structural results that quantify the classes of mechanisms/algorithms for cumulative sums and other computations in the *continual release model* [4, 3] that remain private even when the inputs stream is defined *adaptively*. In the continual release model, a mechanism that receives a stream of inputs $\mathbf{x} = (x_1, \ldots, x_n)$ and produces a stream of outputs $a_1, ..., a_n$, where output a_i is intended to approximate some function of the prefix $x_1, ..., x_i$ and must be generated before x_{i+1} is received. We specify which parts of the input can depend on a single person's data via a *neighbor* relation $\mathcal{N}$ on data streams. Two streams are neighbors if they differ in one person's data. For example, if one person's data directly affects exactly one input in the stream ("*event-level*

privacy"), then we say two data streams are *neighboring* if they differ in exactly one element (that is, they are at Hamming distance 1).

The original works on continual release analyzed privacy in a *nonadaptive* model: a mechanism $\mathcal{M}$ is (ε, δ)-differentially private for nonadaptive continual release if, for all pairs of adjacent data streams $\mathbf{x}, \mathbf{y}$, the corresponding distributions on output streams $\mathcal{M}(\mathbf{x})$ and $\mathcal{M}(\mathbf{y})$ are (ε, δ)-indistinguishable, denoted $\mathcal{M}(\mathbf{x}) \approx_{\varepsilon,\delta} \mathcal{M}(\mathbf{y})$. That is, for all events E, we have $\Pr(\mathcal{M}(\mathbf{x}) \in E) \leq e^{\varepsilon} \Pr(\mathcal{M}(\mathbf{y}) \in E) + \delta$ and $\Pr(\mathcal{M}(\mathbf{y}) \in E) \leq e^{\varepsilon} Pr(\mathcal{M}(\mathbf{x}) \in E) + \delta$.

In many use cases—including those arising in iterative gradient-based optimization algorithms—the nonadaptive model is inadequate since the inputs x_i may be generated in real time as a function of previous outputs $a_1, ..., a_{i-1}$. To summarize the more general, adaptive definition [13, 8], consider an adversary that *adaptively* defines two input sequences $\mathbf{x} = (x_1, \ldots, x_n)$ and $\mathbf{y} = (y_1, \ldots, y_n)$. The adversary must satisfy the promise that these sequences correspond to neighboring data sets. The privacy game proceeds in rounds. At round t, the adversary generates x_t and y_t. The game accepts these if the input streams defined so far are valid, meaning that there exist completions $(x'_{t+1}, ..., x'_T)$ and $(y'_{t+1}, ..., y'_T)$ so that $((x_1, ..., x_t, x'_{t+1}, ..., x'_T), (y_1, ..., y_t, y'_{t+1}, ..., y'_T)) \in \mathcal{N}$. For example, in the case of event-level privacy, the game simply checks that the two streams differ in at most one position so far.

The game is parameterized by a bit $\mathsf{side} \in \{0, 1\}$ which is unknown to the adversary but constant throughout the game. The game hands either x_t or y_t to the mechanism $\mathcal{M}$, depending on side. The mechanism's output a_t is then sent to the adversary. The privacy requirement is that the adversary's views with $\mathsf{side} = 0$ and $\mathsf{side} = 1$ be (ε, δ) indistinguishable. One can substitute other relevant notions of indistinguishability like those from CDP [30, 31], Renyi DP [32], or Gaussian DP [33]. The mechanisms we consider generally satisfy Gaussian DP.

The nonadaptive version of the definition is much easier to work with. It is therefore natural to look for classes of mechanisms for which the two definitions are equivalent (and thus for which a nonadaptive privacy proof implies the more general guarantee). We first observe that such a transfer statement holds for "pure" ε-DP (in which $\delta = 0$). We defer all the proofs to the supplementary material.

Proposition 2.1. *Every mechanism that is $(\varepsilon, 0)$ nonadaptively DP in the continual release model satisfies the adaptive version of the definition, with the same parameters.*

Unfortunately, not all privacy proofs for the nonadaptive model transfer to the adaptive setting. Indeed, we show that there are additive-noise mechanisms (which simply add noise from a predefined distribution to some function of the data) that are nonadaptively (ε, δ)-DP, but *not* private in the adaptive setting (Supplementary Material D.1). That said, for an important class of additive-noise mechanisms, based on Gaussian noise, a transfer result does indeed hold.

Adaptive privacy for Gaussian noise addition mechanism In Theorem 2.1 we show that every Gaussian noise addition mechanism that is (ε, δ)-DP in the nonadaptive model is also private in the adaptive setting; the same holds true for Gaussan DP and zCDP. To prove this we crucially use the rotational invariance of spherical Gaussian distribution.

Theorem 2.1. *Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be a stream of data samples, with each row $\mathbf{x}_i \in \mathbb{R}^d$ representing the data sample observed at time i. Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be a lower-triangular full-rank query matrix, and let $\mathbf{A} = \mathbf{BC}$ be any factorization. Let $\mathbf{Z} \in \mathcal{N}(0, \sigma^2)^{n \times d}$ s.t. $\mathbf{B}(\mathbf{CX} + \mathbf{Z})$ satisfies (ε, δ)-DP in the continual release model, when the rows of $\mathbf{X}$ are chosen non-adaptively. Then, the same estimate satisfies the same DP guarantee even when the rows of $\mathbf{X}$ are chosen adaptively.*

The proof of Theorem 2.1 can be understood as leveraging distributional equivalence of an orbit of mechanisms: those factorizations expressible as $\mathbf{BUU}^*\mathbf{C}$ for $\mathbf{U}$ unitary. This observation can similarly be leveraged to show a subtly distinct fact:

Proposition 2.2. *For any factorization $\mathbf{A} = \mathbf{BC}$ where $\mathbf{A}$ is lower-triangular, there exists a factorization $\mathbf{A} = \widehat{\mathbf{B}}\widehat{\mathbf{C}}$ which induces a distributionally equivalent matrix mechanism under Gaussian noise with $\widehat{\mathbf{B}}$ and $\widehat{\mathbf{C}}$ lower triangular. This factorization can be explicitly computed from $\mathbf{A} = \mathbf{BC}$ via an appropriate QR decomposition. Normalizing to all-nonnegative entries on the diagonal, this factorization is unique.*

3 Computing optimal factorizations

Once one writes down the matrix mechanism in Eq. (1), it becomes immediately natural to pursue optimizing the error of matrix factorizations in some metric of choice. We consider the expected squared reconstruction error of the estimated tensor $\widehat{\mathbf{Ax}}$, which has previously been noted as an appropriate formulation of error for the setting of training private ML models [6, Theorem 5]. Further, for simplicity we restrict our attention to the single-pass setting. That is, for the remainder of the paper we will assume:

Definition 3.1. *Two data matrices $\mathbf{X}$ and $\mathbf{X}'$ in $\mathbb{R}^{n\times d}$ will be considered to be neighboring if they differ by a single row, with the ℓ_2 norm of the difference in this row at most ζ.*

Under this notion of sensitivity, one can make the estimation of any query $\mathbf{AX}$ (ε, δ)-DP via the following theorem, which Theorem 2.1 immediately extends to adaptive streams:

Theorem 3.1 (Adapted from [1])**.** *Consider a query matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ along with a fixed factorization $\mathbf{A} = \mathbf{B}_{n\times n}\mathbf{C}_{n\times n}$, s.t. $\gamma = \max_{i\in[n]} \left\|\mathbf{C}_{[:,i]}\right\|_2$, the maximum column norm of $\mathbf{C}$. Let $\mathbf{X} \in \mathbb{R}^{n\times d}$ be a data matrix with each row of $\mathbf{X}$ having ℓ_2-norm at most ζ. The algorithm that outputs $\mathbf{B}\left(\mathbf{CX} + \mathbf{Z}\right)$ with $\mathbf{Z} \sim \mathcal{N}\left(0, \frac{\gamma^2\zeta^2(2\log(1.25/\delta))}{\varepsilon^2}\right)^{n\times d}$ satisfies (ε, δ)-DP.*

In this setting, following the distribution of $\mathbf{Z}$ from Theorem 3.1, the expected reconstruction error of a factorization $\mathbf{A} = \mathbf{BC}$ can be computed directly [1, Proposition 9], [26, Equation 3]:

$$\mathcal{L}(\mathbf{B}, \mathbf{C}) = \left(\max_{i\in[1,\ldots,n]} \left\|\mathbf{C}_{[:,i]}\right\|_2^2\right) \|\mathbf{B}\|_F^2 \tag{2}$$

As has been noted [1, 26], Eq. (2) can be manipulated to yield a convex program, for which hand-tuned algorithms exist [26, Section 4]. Theorem 2.1 shows, for the first time, that arbitrary factorizations found by minimizing this optimization problem can be applied in the adaptive streaming setting.

We present an alternative characterization of these optima, which reformulates the optimization problem as a fixed-point problem. We show that simply iterating an explicit mapping converges to this fixed point with appropriate (though implicitly defined) initialization, and observe numerically that the associated algorithm achieves fast, global convergence.

Since the Moore-Penrose pseudoinverse yields the minimal ℓ_2-norm solution to a set of underdetermined linear equations [34, Theorem 2.1.1], we note that for a fixed $\mathbf{C}$ term, the optimal $\mathbf{B}$ may be expressed as $\mathbf{B}^\star_{\mathbf{C}} = \mathbf{AC}^\dagger$.

We can remove the dependence of Eq. (2) on its $\mathbf{C}$-term by considering a constrained optimization problem. Since $\mathbf{A} = \mathbf{BC}$ implies $\mathbf{A} = \left(\alpha\mathbf{B}\right)\left(\frac{1}{\alpha}\mathbf{C}\right)$, for any linear space of matrices $\mathbf{V}$,

$$\begin{aligned}\min_{\mathbf{C}\in\mathbf{V}} \mathcal{L}\left(\mathbf{AC}^\dagger, \mathbf{C}\right) &= \min_{\mathbf{C}\in\mathbf{V}} \left(\max_{i\in[1,\ldots,n]} \left\|\mathbf{C}_{[:,i]}\right\|_2^2\right) \left\|\mathbf{AC}^\dagger\right\|_F^2 \\ &= \min_{\mathbf{C}\in\mathbf{V}, \max_{i\in[1,\ldots,n]}\left\|\mathbf{C}_{[:,i]}\right\|_2^2=1} \left\|\mathbf{AC}^\dagger\right\|_F^2 .\end{aligned} \tag{3}$$

The properties of the optimization problem Eq. (3) have been studied previously. In particular, [26, Section 3] studied the reparameterization:

$$\mathbf{X}^\star = \underset{\mathbf{X}\text{ is PD}, \mathbf{X}_{[i,i]}\leq 1, 1\leq i\leq n}{\arg\min} \operatorname{tr}(\mathbf{A}^*\mathbf{A}\mathbf{X}^{-1}) \tag{4}$$

which essentially reparameterizes Eq. (3) (with $\mathbf{V} = \mathbb{R}^n$) in terms of $\mathbf{C}^*\mathbf{C}$. To recover a matrix-mechanism factorization of $\mathbf{A}$, then, one may utilize any $\mathbf{C}$ such that $\mathbf{X}^\star = \mathbf{C}^*\mathbf{C}$, e.g. $\mathbf{C} = \sqrt{\mathbf{X}^\star}$. Proposition 2.2 can then be used to construct a lower-triangular factorization if desired.

Yuan et al. [26] show: 1) Any solution $\mathbf{X}^\star$ of Eq. (4) must have diagonal entries exactly 1. 2) Any solution $\mathbf{X}^\star$ may be taken to be strictly within the positive-definite cone, with minimal eigenvalue

bounded from below in terms of the eigenvalues of $\mathbf{A}$. 3) For any full-rank $\mathbf{A}$, $\mathbf{X} \mapsto \mathrm{tr}(\mathbf{A}^*\mathbf{A}\mathbf{X}^{-1})$ is strictly convex over symmetric, positive-definite matrices. Therefore the solution to Eq. (4) is unique.

By analyzing Eq. (4) directly, we derive a characterization of solutions in terms of an explicit fixed-point problem:

Theorem 3.2. *Minimizers $\mathbf{X}^\star$ of Eq.* (4) *are in one-to-one correspondence with fixed points of the function $\phi : \mathbb{R}_+^n \to \mathbb{R}_+^n$ defined by*

$$\phi(\mathbf{v}) = \mathrm{diagpart}\left(\sqrt{\mathrm{diag}(\mathbf{v})^{1/2}\,\mathbf{A}^*\mathbf{A}\;\mathrm{diag}(\mathbf{v})^{1/2}}\right). \tag{5}$$

Moreover, for such a fixed point $\mathbf{v}^\star$, the associated minimizer $\mathbf{X}^\star$ may be written as

$$\mathbf{X}^\star = \mathrm{diag}(\mathbf{v}^\star)^{-1/2}\left(\mathrm{diag}(\mathbf{v}^\star)^{1/2}\,\mathbf{A}^*\mathbf{A}\;\mathrm{diag}(\mathbf{v}^\star)^{1/2}\right)^{1/2}\mathrm{diag}(\mathbf{v}^\star)^{-1/2}, \tag{6}$$

and therefore this pair $(\mathbf{X}^\star, \mathbf{v}^\star)$ satisfies

$$\mathbf{A}^*\mathbf{A} = \mathbf{X}^\star\,\mathrm{diag}\left(\mathbf{v}^\star\right)\mathbf{X}^\star. \tag{7}$$

The sum of the elements of ϕ represents a quantity of independent interest in quantum information, the so-called Jozsa fidelity [35, 36]. Several nontrivial mathematical statements follow immediately by combining this Theorem with preceding observations on the problem Eq. (4).

Corollary 3.1. *There is a unique setting of $\mathbf{v}^\star$ which satisfies Eq.* (7) *for $\mathbf{X}$ with all 1s on the diagonal, symmetric and positive definite. This $\mathbf{v}^\star$ uniquely determines $\mathbf{X}$.*

Corollary 3.2. *ϕ as defined in Eq.* (5) *has a unique fixed point.*

We can show local, though not yet global, convergence of these iterates.

Theorem 3.3. *ϕ defined in Eq.* (5) *is a local contraction around its fixed point in a suitable metric, and hence there is a neighborhood of this fixed point in which iterates of ϕ converge to this fixed point. The precise norm of this contraction, and the size of the neighborhood in which convergence is guaranteed, can both be estimated in terms of minimum and maximum eigenvalues of $\mathbf{A}$.*

This result can be shown by linearizing the mapping ϕ around its fixed point and performing an involved estimate of its Jacobian at the fixed point. As such, it is implied by Theorem E.1, which states that the linearization of ϕ around its fixed point is a contraction in a suitable metric, and therefore the Banach fixed-point theorem applies.

Remark. Notably missing from quantification of the contraction is the dimension n. Indeed, the argument is dimension-independent in a strong sense: it applies to the suitably generalized definition of ϕ where $\mathbf{A}$ is any bounded linear operator on a Hilbert space with bounded inverse.

Empirical performance of the fixed-point method Experimentally, *simply iterating* the mapping ϕ is sufficient to converge to the global optimum, extremely quickly, from any initial point (modulo potential numerical issues in computing the matrix square root, discussed in Appendix E.3). Yuan et al. [26, Algorithm 1] yields a global linear and local quadratic rate, with similar asymptotics to iterating ϕ (each dominated by an n^3 term), though at the cost of introducing a parameter T. Iterating ϕ, on the other hand, is parameter-free. We implemented [26, Algorithm 1] as well as a direct gradient-descent-based method to numerically compare convergence speed. As a canonical benchmark, we computed optimal factorizations of the 512×512 prefix-sum matrix $\mathbf{S}$ (Appendix F provides a visualization of this optimal factorizaiton). Our fixed-point algorithm was significantly faster than either of the alternatives to compute optima, finding a lower-loss matrix in 2.5 seconds than either alternative found in over 5 minutes. See Appendix E.3 for details. The speed and simplicity of our fixed-point algorithm was a significant enabler of the explorations of mechanisms presented in the next section.

4 The matrix mechanism for SGD

To define our gradient descent algorithm, let $\mathbf{G} \in \mathbb{R}^{n\times d}$ be the matrix of gradients (playing the role of $\mathbf{X}$ above), with row vector $\mathbf{G}_{[i,:]} \in \mathbb{R}^{1\times d}$ the gradient observed on iteration i; we abuse

Algorithm 1 DP Matrix Factorization SGD

Inputs:
  factorization $\mathbf{M} = \mathbf{BC}$
  overall learning rate η
  noise level σ, clipping norm ζ
  examples χ_i, $i \in \{1, \ldots, n\}$
$\boldsymbol{\theta}_{[0,:]} := 0 \in \mathbb{R}^d$
Sample $\mathbf{Z} \in \mathbb{R}^{n \times d}$, $\mathbf{Z}_{[i,j]} \sim \mathcal{N}(0, \sigma^2)$ iid
for i in $1, \ldots, n$ **do**
  $\mathbf{v} := \nabla_{\boldsymbol{\theta}_{[i-1,:]}} \ell(\boldsymbol{\theta}; \chi_i)$
  $\mathbf{G}_{[i,:]} := \mathbf{v} \cdot \min\left\{\frac{\zeta}{\|\mathbf{v}\|_2}, 1\right\}$
  $\boldsymbol{\theta}_{[i,:]} := -\eta\big(\mathbf{M}_{[i,:]}\mathbf{G}_{[1:i,j]} + \mathbf{B}_{[i,:]}\mathbf{Z}\big)$

Algorithm 2 Heavy-ball momentum

$\theta_0 := 0$, $m_0 := 0$
for i in $1, \ldots, n$ **do**
  $m_i := \beta \cdot m_{i-1} + g_i$
  $\theta_i := \theta_{i-1} - \eta_i m_i$

notation slightly by writing $\mathbf{G}_{[1:i,:]} \in \mathbb{R}^{n \times d}$ formed by taking the first i rows of $\mathbf{G}$, with zeros for the as-of-yet unobserved gradient rows for iterations $i+1, \ldots, n$ (the lower triangular structure of $\mathbf{M}$ implies using $\mathbf{G}_{[1:i,:]}$ vs $\mathbf{G}$ does not in fact change the value computed). With this notation, we define Algorithm 1, a general template for private SGD algorithms.

The power of this general formulation comes largely from the following privacy guarantee:

Theorem 4.1. *Under the "replace with zero" notion of differential privacy [6, Defn. 2.1] over examples (records) χ_i, taking $\mathbf{v} = 0$ when $\chi_i = \perp$, the iterates $\boldsymbol{\theta}_{[i,:]}$ released by Algorithm 1 satisfy equivalent (ε, δ)-DP to the Gaussian mechanism with noise variance σ^2 applied to records with ℓ_2 sensitivity at most $\zeta\gamma$, where $\gamma = \max_i \left\|\mathbf{C}_{[:,i]}\right\|_2$ is the maximum column norm of $\mathbf{C}$.*

We now consider different instantiations of Algorithm 1. Let $\mathbf{S}$ be the prefix-sum matrix as defined in Appendix A, and let $\mathbf{C}_{\mathcal{T}}$ be the matrix representation of the binary tree (Appendix C), so for appropriate choices of the reconstruction matrices $\mathbf{B}_{\text{hs}}$ and $\mathbf{B}_{\text{hf}}$, $\mathbf{S} = \mathbf{B}_{\text{hs}}\mathbf{C}_{\mathcal{T}}$ gives the `Honaker Online` mehcanism, and $\mathbf{S} = \mathbf{B}_{\text{hf}}\mathbf{C}_{\mathcal{T}}$ gives the `Honaker Full` mechanism. In particular, using the `Honaker Online` factorization in Algorithm 1 recovers the non-momentum DP-FTRL algorithm of Kairouz et al. [6].

However, Kairouz et al. [6] observed that for non-convex objectives, DP-FTRL with momentum provided superior privacy/accuracy tradeoffs. Given prefix sums, momentum can be implemented as post processing by estimating individual gradients/updates as the difference of successive cumulative sums (multiplication by $\mathbf{S}^{-1}$, and then passing these into a standard momentum SGD optimizer. We show that performance can be improved by directly incorporating momentum and a-priori learning rate schedules directly into the DP mechanism.

A basic but important observation is that momentum SGD can be expressed as a linear map of gradients $\mathbf{g} \to \mathbf{Mg}$. We consider the classic momentum algorithm of Polyak [37], with per-iteration learning rates[2] $\eta_1, \ldots, \eta_n$ and momentum $\beta \in [0, 1)$, as in Algorithm 2. Alternatively, we can express momentum SGD as a linear operator on the gradients:

Proposition 4.1. *For any $\beta \in [0, 1]$, $n \geq 1$, per iteration learning rates $\eta_1, \ldots, \eta_n$, define the lower-triangular matrix $\mathbf{M} \in \mathbb{R}^{n \times n}$ as the product of lower-triangular matrices $\mathbf{M}^{(\eta)}$ and $\mathbf{M}^{(\beta)}$:*

$$\mathbf{M}^{(\eta)}_{[i,j]} = \begin{cases} \eta_j & i \geq j \\ 0 & \textit{otherwise} \end{cases}, \quad \mathbf{M}^{(\beta)}_{[i,j]} = \begin{cases} \beta^{i-j} & i \geq j \\ 0 & \textit{otherwise} \end{cases}, \qquad \textit{and} \qquad \mathbf{M} = \mathbf{M}^{(\eta)}\mathbf{M}^{(\beta)}. \quad (8)$$

Then, for any vector of scalar per-iteration gradients $\mathbf{g} = [g_1, \ldots, g_n] \in \mathbb{R}^n$, the sequence of iterates $\boldsymbol{\theta} = [\theta_1, \ldots \theta_n]$ produced by Algorithm 2 can equivalently be written $\boldsymbol{\theta} = -\mathbf{Mg}$.

We apply the fixed-point algorithm of Section 3 to $\mathbf{M}$ to obtain optimal matrix mechanisms $\mathbf{M} = \mathbf{BC}$, which indeed leads to improved performance.[3] We can also convert mechanisms that produce DP prefix sums to produce momentum iterates via post processing: given any matrix mechanism for

[2]The schedule may be arbitrarily, but must be chosen a priori in a data-independent way.

[3]The definition $\mathbf{M} = \mathbf{M}^{(\eta)}\mathbf{M}^{(\beta)}$ is for the convenience, and is unrelated to the optimal factorization.

Mechanism	$\mathbf{B}$	$\mathbf{C}$	s.t. $\mathbf{M} = \mathbf{BC}$
`Honaker Online`	$\mathbf{MS}^{-1}\mathbf{B}_{\text{hs}}$	$\mathbf{C}_{\mathcal{T}}$	Equivalent to DP-FTRL of Kairouz et al. [6]
`Honaker Full`	$\mathbf{MS}^{-1}\mathbf{B}_{\text{hf}}$	$\mathbf{C}_{\mathcal{T}}$	
`Opt Prefix Sum`	$\mathbf{MS}^{-1}\mathbf{B}_S^*$	$\mathbf{C}_S^\star$	for optimal $\mathbf{S} = \mathbf{B}_S^\star\mathbf{C}_S^\star$
`Optimal M = W H`	$\mathbf{B}_M^\star$	$\mathbf{C}_M^\star$	for optimal $\mathbf{M} = \mathbf{B}_M^\star\mathbf{C}_M^\star$

Table 1: Instantiations of Algorithm 1 for various factorizations of the SGD matrix $\mathbf{M} = \mathbf{BC}$.

the prefix sum problem given as a factorization $\mathbf{S} = \mathbf{BC}$, we can convert this to a mechanism for momentum as $\mathbf{M} = \hat{\mathbf{B}}\mathbf{C}$ where $\hat{\mathbf{B}} = \mathbf{MS}^{-1}\mathbf{B}$. Straightforward calculations show this representation is equivalent to the "Momentum Variant" of DP-FTRL [6]. This allows us to consistently evaluate the mechanisms in terms of the total variance (squared error) induced in the outputs by unit-variance noise $\mathbf{z}$ in the mechanism via Eq. (2). Table 1 summarizes the four instantiations of Algorithm 1 for various choices of the momentum matrix $\mathbf{M}$; note in particular that for $\beta = 0$ and a fixed learning rate schedule $\eta_i = 1$, $\mathbf{M} = \mathbf{S}$. Fig. 1 compares the squared errors in the estimation of the momentum iterates (without learning rate scaling) for three methods.

Computational efficiency Some care is necessary for the efficient implementation of Algorithm 1, particularly the computation of $\mathbf{M}_{[i,:]}\mathbf{G}_{[1:i,j]} + \mathbf{B}_{[i,:]}\mathbf{Z}$ on line 11. First, we observe that via Proposition 4.1 we can efficiently compute the $\mathbf{MG}$ term via Algorithm 2, rather than as a matrix operation. This leaves the computation of the noise $\mathbf{B}_{[i,:]}\mathbf{Z}$. For applications to ML, d could be $10^6 - 10^9$, and with even $n = 10^4$ rounds this might make the total calculation quite expensive if not prohibitive. With an efficient TensorFlow implementation, in our experiments with $d \approx 4 \times 10^6$ and $n = 2048$, we found we could compute the noise directly. However, for larger applications we show (in Appendix G) that one can compute structured matrices that well-approximate the optimal $\mathbf{B}^\star$ for the prefix sum matrix while allowing for $\mathcal{O}(d)$ calculation of the per-round noise vectors (on par with that of the binary tree mechanism, which can also provide computational efficiency with a careful implementation, see Table 2). The key is to observe the diagonal dominance of the optimal $\mathbf{B}^\star$ (see Appendix F), leading to an approximation $\hat{\mathbf{B}}$ that is the sum of a lower-triangular d-banded matrix with the remaining entries in the lower triangle extracted from a low-rank approximation computed via alternating-least-squares.

5 Experimental results

The results of Sections 2 and 4 significantly expand the space of mechanisms which can be used in training ML models with differential privacy in the single-pass setting. In this section we demonstrate that these techniques can in fact significantly advance the state-of-the-art in private ML.

User-level privacy for language models Private training is particularly important for generative language models: training language models on data from the right distribution is critical for utility (e.g., user input in a mobile keyboard [38, 39]), but this data is often privacy-sensitive. Futher, language models have been shown to be capable of memorizing training data [40–42]. Further, in this setting it is important to consider user-level DP, where the DP guarantee extends to all of the training examples (tokens) from any one user, as opposed to a single training example [43]. In our setting, this corresponds to ensuring that each user's examples contribute to a bounded ℓ_2-norm update to a single row of $\mathbf{X}$ (Definition 3.1). This is accomplished by extending Algorithm 1 in the natural way to Federated Averaging [44]: instead of a single gradient, we take $\mathbf{G}_{[i,:]}$ to be the sum of the individually-clipped-to-ζ updates of all users (100 or 167 in our experiments) participating in the current round, with each user contributing to a single round over the course of training.

For these reasons, we focused on the StackOverflow next-word prediction problem in the federated setting, introduced in [45] and publicly hosted in TensorFlow-Federated (TFF) [46]. This task was explored extensively in [6], and serves as a major benchmark in federated learning, used in [45, 6, 47–49], and others. The StackOverflow dataset contains sufficiently many clients to support single-pass

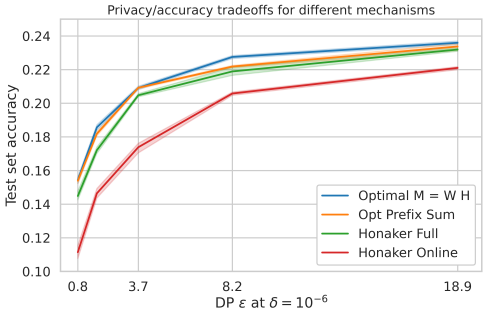

Figure 2: Test accuracy for the StackOverflow next-word-prediction task. A grid search over client and server learning rates and momentum β, with the best hyperparameters selected based on validation set accuracy. We then re-ran 11 repetitions with the best hyperparameters and report the mean test set accuracy with confidence intervals. All models trained with 100 clients per round with a constant learning rate schedule.

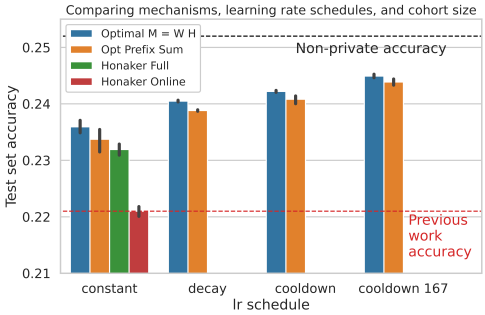

Figure 3: Test accuracy for mechanisms incorporating learning rate decay in the manner of Section 4 at $\varepsilon = 18.9$. The lowest horizontal line represents test accuracy of previous state of the art at $\varepsilon = 18.9$; highest horizontal line, test accuracy of non-private model. The final bar group shows learning rate cooldown with 167 clients/round, the maximum possible for a single training pass of 2048 training rounds.

algorithms with 100 clients per round, similar to the baseline setup of [6]. For this reason, we are able to provide true (ε, δ) privacy quantifications for the models we train.[4]

Results. We compare the four mechanisms of Table 1 on this problem; full experimental methodology, as well as additional plots (e.g., for validation error vs. training rounds) are provided in Appendix H. Fig. 2 shows that even with constant learning rates, our matrix factorization approaches significantly outperform the previous state-of-the-art across a range of privacy ε's. Further, thanks to the results of Section 2, we are able to apply the `Honaker Full` mechanism for comparison. Fig. 3 shows that applying learning rate decay (dropping the learning rate by $0.15\times$ for the last 512 rounds) and learning rate cooldown (linearly dropping the learning rate from $1.0\times$ to $0.05\times$ over the last 512 rounds) show added improvements. These experiments all used 100 clients/round as in [6], but the rightmost bars shows increasing this to 167 (the maximum possible for a single pass of 2048 rounds) provides additional accuracy. In combination, these techniques close more than 2/3rds of the gap between private and non-private training.

6 Conclusions

We have shown the general applicability of the Gaussian matrix mechanism to the streaming setting, introduced a highly efficient mechanism of determining optimal (in the sense of total ℓ_2^2 error) matrix mechanisms, used this approach to directly incorporate momentum and learning rate schedules into the DP mechanism, and empirically demostrated the resulting private SGD (or FedAvg in the federated setting) substantially improves on the state of the art for private ML.

While our focus has been on the application of these techniques to gradient-based optimization algorithms, we emphasize that the problem of producing private estimates for linear queries in the adaptive streaming setting is a fundamental DP primitive of much broader applicability, as noted in the introduction, and so our work immediately leads to improvements in those applications as well.

Finally, our work raises numerous natural follow-up questions which we hope will inspire subsequent work; we sketch these in Appendix B.

[4] With 342,477 users this dataset is still small compared to many real-world applications (e.g., Gboard has 5 billion downloads). The extrapolations verified by [43, 6] suggest the accuracy results of Fig. 3 would hold with $(\varepsilon = 1.36, \delta = 10^{-7})$-DP if the population and cohort size (clients per round) were both scaled by $10\times$.

References

[1] Chao Li, Gerome Miklau, Michael Hay, Andrew Mcgregor, and Vibhor Rastogi. The matrix mechanism: optimizing linear counting queries under differential privacy. *The VLDB Journal*, 24:757–781, 2015.

[2] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, pages 265–284, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-32732-5.

[3] T.-H. Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. *ACM Trans. on Information Systems Security*, 14(3):26:1–26:24, November 2011.

[4] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N. Rothblum. Differential privacy under continual observation. In *Proc. of the Forty-Second ACM Symp. on Theory of Computing (STOC'10)*, pages 715–724, 2010.

[5] Prateek Jain, Pravesh Kothari, and Abhradeep Thakurta. Differentially private online learning. In *Proc. of the 25th Annual Conf. on Learning Theory (COLT)*, volume 23, pages 24.1–24.34, June 2012.

[6] Peter Kairouz, Brendan McMahan, Shuang Song, Om Thakkar, Abhradeep Thakurta, and Zheng Xu. Practical and private (deep) learning without sampling or shuffling. In *ICML*, 2021.

[7] Adam Smith and Abhradeep Thakurta. (nearly) optimal algorithms for private online learning in full-information and bandit settings. In *Advances in Neural Information Processing Systems*, pages 2733–2741, 2013.

[8] Palak Jain, Sofya Raskhodnikova, Satchit Sivakumar, and Adam D. Smith. The price of differential privacy under continual observation. *ArXiv CoRR*, abs/2112.00828, 2021. URL `https://arxiv.org/abs/2112.00828`.

[9] Cynthia Dwork, Kunal Talwar, Abhradeep Thakurta, and Li Zhang. Analyze gauss: optimal bounds for privacy-preserving principal component analysis. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 11–20, 2014.

[10] Graham Cormode, Tejas Kulkarni, and Divesh Srivastava. Answering range queries under local differential privacy. *Proceedings of the VLDB Endowment*, 12(10):1126–1138, 2019.

[11] Adrian Rivera Cardoso and Ryan Rogers. Differentially private histograms under continual observation: Streaming selection into the unknown. *CoRR*, abs/2103.16787, 2021. URL `https://arxiv.org/abs/2103.16787`.

[12] Adam Smith, Abhradeep Thakurta, and Jalaj Upadhyay. Is interaction necessary for distributed private learning? In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 58–77. IEEE, 2017.

[13] Abhradeep Guha Thakurta and Adam Smith. (nearly) optimal algorithms for private online learning in full-information and bandit settings. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013. URL `https://proceedings.neurips.cc/paper/2013/file/c850371fda6892fbfd1c5a5b457e5777-Paper.pdf`.

[14] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 245–248. IEEE, 2013.

[15] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *Proc. of the 2014 IEEE 55th Annual Symp. on Foundations of Computer Science (FOCS)*, pages 464–473, 2014.

[16] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, Oct 2016. doi: 10.1145/2976749.2978318. URL `http://dx.doi.org/10.1145/2976749.2978318`.

[17] Nirvan Tyagi, Yossi Gilad, Derek Leung, Matei Zaharia, and Nickolai Zeldovich. Stadium: A distributed metadata-private messaging system. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 423–440, 2017.

[18] Úlfar Erlingsson, Ilya Mironov, Ananth Raghunathan, and Shuang Song. That which we call private, 2019.

[19] Matthew Joseph, Aaron Roth, Jonathan Ullman, and Bo Waggoner. Local differential privacy for evolving data. *Advances in Neural Information Processing Systems*, 31, 2018.

[20] Differential Privacy Team Apple. Learning with privacy at scale, 2017.

[21] James Honaker. Efficient use of differentially private binary trees. *Theory and Practice of Differential Privacy (TPDP 2015), London, UK*, 2015.

[22] H Brendan McMahan and Matthew Streeter. Adaptive bound optimization for online convex optimization. *arXiv preprint arXiv:1002.4908*, 2010.

[23] Brendan McMahan. Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 525–533, 2011.

[24] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[25] Moritz Hardt and Kunal Talwar. On the geometry of differential privacy. In *STOC*, 2010.

[26] Ganzhao Yuan, Yin Yang, Zhenjie Zhang, and Zhifeng Hao. Optimal linear aggregate query processing under approximate differential privacy. *CoRR*, abs/1602.04302, 2016. URL `http://arxiv.org/abs/1602.04302`.

[27] Ryan McKenna, Gerome Miklau, Michael Hay, and Ashwin Machanavajjhala. Optimizing error of high-dimensional statistical queries under differential privacy. *Proc. VLDB Endow.*, 11(10):1206–1219, jun 2018. ISSN 2150-8097. doi: 10.14778/3231751.3231769. URL `https://doi.org/10.14778/3231751.3231769`.

[28] Alexander Edmonds, Aleksandar Nikolov, and Jonathan Ullman. *The Power of Factorization Mechanisms in Local and Central Differential Privacy*, page 425–438. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 9781450369794. URL `https://doi.org/10.1145/3357713.3384297`.

[29] Hendrik Fichtenberger, Monika Henzinger, and Jalaj Upadhyay. Constant matters: Fine-grained complexity of differentially private continual observation, 2022. URL `https://arxiv.org/abs/2202.11205`.

[30] Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In *Theory of Cryptography Conference*, pages 635–658. Springer, 2016.

[31] Cynthia Dwork and Guy N. Rothblum. Concentrated differential privacy. *CoRR*, abs/1603.01887, 2016.

[32] Ilya Mironov. Rényi differential privacy. In *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pages 263–275. IEEE, 2017.

[33] Jinshuo Dong, Aaron Roth, and Weijie J. Su. Gaussian differential privacy. *CoRR*, abs/1905.02383, 2019. URL `http://arxiv.org/abs/1905.02383`.

[34] S. L. Campbell and C. D. Meyer. *Generalized inverses of linear transformations / S. L. Campbell, C. D. Meyer*. Pitman London ; San Francisco, 1979. ISBN 0273084224.

[35] Richard Jozsa. Fidelity for mixed quantum states. *Journal of Modern Optics*, 41(12):2315–2323, 1994. doi: 10.1080/09500349414552171. URL `https://doi.org/10.1080/09500349414552171`.

[36] Yeong-Cherng Liang, Yu-Hao Yeh, Paulo E M F Mendonça, Run Yan Teh, Margaret D Reid, and Peter D Drummond. Quantum fidelity measures for mixed states. *Reports on Progress in Physics*, 82(7):076001, jun 2019. doi: 10.1088/1361-6633/ab1ca4. URL https://doi.org/10.1088/1361-6633/ab1ca4.

[37] B.T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964. ISSN 0041-5553.

[38] Andrew Hard, Kanishka Rao, Rajiv Mathews, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *CoRR*, abs/1811.03604, 2018. URL http://arxiv.org/abs/1811.03604.

[39] Swaroop Ramaswamy, Om Thakkar, Rajiv Mathews, Galen Andrew, H. Brendan McMahan, and Françoise Beaufays. Training production language models without memorizing user data, 2020.

[40] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. The secret sharer: Evaluating and testing unintended memorization in neural networks. In *Proceedings of the 28th USENIX Conference on Security Symposium*, SEC'19, page 267–284, USA, 2019. USENIX Association. ISBN 9781939133069.

[41] Congzheng Song and Vitaly Shmatikov. Auditing data provenance in text-generation models. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 196–206, 2019.

[42] Nicholas Carlini, Florian Tramèr, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Úlfar Erlingsson, Alina Oprea, and Colin Raffel. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650. USENIX Association, August 2021. ISBN 978-1-939133-24-3. URL https://www.usenix.org/conference/usenixsecurity21/presentation/carlini-extracting.

[43] H Brendan McMahan, Daniel Ramage, Kunal Talwar, and Li Zhang. Learning differentially private recurrent language models. *arXiv preprint arXiv:1710.06963*, 2017.

[44] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA*, pages 1273–1282, 2017. URL http://proceedings.mlr.press/v54/mcmahan17a.html.

[45] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, Zachary Garrett, Keith Rush, Jakub Konečný, Sanjiv Kumar, and H. Brendan McMahan. Adaptive federated optimization. *CoRR*, abs/2003.00295, 2020. URL https://arxiv.org/abs/2003.00295.

[46] Alex Ingerman and Krzysztof Ostrowski. Introducing tensorflow federated, Mar 2019. URL https://blog.tensorflow.org/2019/03/introducing-tensorflow-federated.html.

[47] Zachary Charles, Zachary Garrett, Zhouyuan Huo, Sergei Shmulyian, and Virginia Smith. On large-cohort training for federated learning. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=Kb26p7chwhf.

[48] Chen Zhu, Zheng Xu, Mingqing Chen, Jakub Konečný, Andrew Hard, and Tom Goldstein. Diurnal or nocturnal? federated learning of multi-branch networks from periodically shifting distributions. In *International Conference on Learning Representations*, 2022.

[49] Karan Singhal, Hakim Sidahmed, Zachary Garrett, Shanshan Wu, Keith Rush, and Sushant Prakash. Federated reconstruction: Partially local federated learning. *CoRR*, abs/2102.03448, 2021. URL https://arxiv.org/abs/2102.03448.

[50] H. Brendan McMahan and Matthew Streeter. Adaptive bound optimization for online convex optimization. In *Proceedings of the 23rd Annual Conference on Learning Theory (COLT)*, 2010.

[51] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990. ISBN 0521386322. URL http://www.amazon.com/Matrix-Analysis-Roger-Horn/dp/0521386322%3FSubscriptionId%3D192BW6DQ43CK9FN0ZGG2%26tag%3Dws%26linkCode%3Dxm2%26camp%3D2025%26creative%3D165953%26creativeASIN%3D0521386322.

[52] Eugene P. Wigner. On weakly positive matrices. *Canadian Journal of Mathematics*, 15:313–317, 1963. doi: 10.4153/CJM-1963-036-4.

[53] H. W. Kuhn and A. W. Tucker. Nonlinear programming. In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability, 1950*, pages 481–492, Berkeley and Los Angeles, 1951. University of California Press.

[54] Nathan Srebro, Jason Rennie, and Tommi Jaakkola. Maximum-margin matrix factorization. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2005. URL https://proceedings.neurips.cc/paper/2004/file/e0688d13958a19e087e123148555e4b4-Paper.pdf.

[55] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), 2009. ISSN 0018-9162. doi: 10.1109/MC.2009.263. URL https://doi.org/10.1109/MC.2009.263.

[56] Prateek Jain, Praneeth Netrapalli, and Sujay Sanghavi. Low-rank matrix completion using alternating minimization. In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing (STOC)*. Association for Computing Machinery, 2013. ISBN 9781450320290.

[57] Google. Tensorflow-privacy. https://github.com/tensorflow/privacy, year=2019.

[58] Om Thakkar, Galen Andrew, and H Brendan McMahan. Differentially private learning with adaptive clipping. *arXiv preprint arXiv:1905.03871*, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See e.g. Definition 3.1, where we specialize to the single-pass setting.
   (c) Did you discuss any potential negative societal impacts of your work? [No] Our work is focused on private training of machine learning models.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] Our work explores technical methods to enable increased data minimization and provide greater (provable) privacy. As such, we rather discuss some of the social reasons one might want to guarantee privacy to users.
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] Results are stated in a self-contained manner.
   (b) Did you include complete proofs of all theoretical results? [Yes] All results are proved in the supplementary material.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] We are in the process of open-sourcing our code, but this process is not yet complete.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] Included in supplementary material, appendix H.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Details for generating these error bars can be found in appendix H, while the bars themselves are on the main plots.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] These details can be found in the 'Integration with federated learning' section in appendix H.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We use public data available from TensorFlow-Federated (TFF), and cited explicitly.
   (b) Did you mention the license of the assets? [No] Our data is hosted publicly by TFF, which handles the licensing.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No] No new assets.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [No] Only publicly available data was used in this paper.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] All data is publicly available.