Stochastic Adaptive Line Search for Differentially Private Optimization

Chen Chen
Department of Computer Science
University of Georgia
Athens, GA
cchen@cs.uga.edu

Jaewoo Lee
Department of Computer Science
University of Georgia
Athens, GA
jwlee@cs.uga.edu

***Abstract*—The performance of private gradient-based optimization algorithms is highly dependent on the choice of step size (or learning rate) which often requires non-trivial amount of tuning. In this paper, we introduce a stochastic variant of classic backtracking line search algorithm that satisfies Rényi differential privacy. Specifically, the proposed algorithm adaptively chooses the step size satisfying the the Armijo condition (with high probability) using noisy gradients and function estimates. Furthermore, to improve the probability with which the chosen step size satisfies the condition, it adjusts per-iteration privacy budget during runtime according to the reliability of noisy gradient. A naive implementation of the backtracking search algorithm may end up using unacceptably large privacy budget as the ability of adaptive step size selection comes at the cost of extra function evaluations. The proposed algorithm avoids this problem by using the sparse vector technique combined with the recent privacy amplification lemma. We also introduce a privacy budget adaptation strategy in which the algorithm adaptively increases the budget when it detects that directions pointed by consecutive gradients are drastically different. Extensive experiments on both convex and non-convex problems show that the adaptively chosen step sizes allow the proposed algorithm to efficiently use the privacy budget and show competitive performance against existing private optimizers.**

***Index Terms*—differential privacy, stochastic gradient descent, line search, privacy budget adaptation**

I. Introduction

We consider solving the following finite-sum optimization problem under differential privacy [1]–[6]:

$$\arg\min_{\mathbf{w}\in\Theta} F(\mathbf{w}; D) := \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{w}; \mathbf{d}_i)\,, \tag{1}$$

where $D = \{\mathbf{d}_1, \ldots, \mathbf{d}_n\}$ is i.i.d. examples drawn from an unknown data distribution and f represents the loss on one training example. This formulation includes a wide range of machine learning problems, for example, training a neural network with weights $\mathbf{w}$ for classification. Stochastic gradient descent (SGD) has been widely used, especially for large-scale problems, to solve the problem of form (1) due to its simplicity and low iteration cost. For differential privacy, the SGD update typically has the form of:

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta_t\left(\mathbf{g}_t + Y(\epsilon_t)\right),$$

where $\eta_t > 0$ is a step size, Y is a noise (e.g., Gaussian) variable whose scale is determined by the per-iteration privacy budget ϵ_t, and $\mathbf{g}_t$ is the gradient evaluated on a subset $B_t \subseteq D$ of examples selected for the current iteration t:

$$\mathbf{g}_t = \frac{1}{|B_t|}\sum_{\mathbf{d}_i\in B_t} \nabla f(\mathbf{w}_t; \mathbf{d}_i).$$

Despite its prevalent use in differentially private optimization, the use of SGD in practice faces two major challenges. First, the direction pointed by the stochastic gradient $\mathbf{g}_t$ may not be a descent direction. Even worse, depending on the magnitude of noise $Y(\epsilon_t)$, the update direction may still not be a descent direction even when $\mathbf{g}_t$ is a descent direction. A natural question is how to decide whether the privacy budget ϵ_t is sufficiently large enough to get the learning signal, i.e., $\mathbf{g}_t$ is not dominated by $Y(\epsilon_t)$. Second, the efficiency of SGD largely relies on the choice of step size η_t. It can be chosen independent of data, e.g., a constant step size [4], [7]. However, these step sizes are often problem-specific and require a degree of fine-tuning. The methods with data-dependent step sizes [8] require allocating extra privacy budget for selection and efficiently controlling the growth rate of cumulative budget.

In this work, we propose a Rényi differentially private backtracking line search algorithm that adaptively sets the step size using the Armijo condition and empirically show that it can improve the performance of algorithm on both convex and non-convex problems. Armijo line search [9], [10] is a classical technique to find a step size η that gives sufficient reduction in the objective function f. Recently, [11] introduced a stochastic version in which both objectives and gradients are approximated using a random subset of data. To be specific, it uses backtracking algorithm to find a step size η that satisfies

$$f_B(\mathbf{w}_t - \eta\nabla f_B(\mathbf{w}_t)) \le f_B(\mathbf{w}_t) - \alpha\eta\|\nabla f_B(\mathbf{w}_t)\|_2^2\,, \tag{2}$$

where $\alpha \in (0, 1)$ is a hyperparameter and $f_B(\cdot)$ denotes that f is evaluated on the minibatch B. However, privatizing the Armijo line search is a non-trivial task. A naive privatization of this search algorithm may require unacceptably large privacy budget as it requires multiple function evaluations on the

978-1-7281-6251-5/20/$31.00 ©2020 IEEE

dataset. Motivated by the observation that the Armijo line search sequentially evaluates *threshold* queries

$$q(\eta) = f(\mathbf{w}_t) - f(\mathbf{w}_t - \eta \nabla f(\mathbf{w}_t)) - \alpha \eta \|\nabla f(\mathbf{w}_t)\|_2^2 \geq 0$$

for different values of η, the proposed algorithm adopts the Sparse Vector technique [2], [12], which allows the algorithm to pay the privacy budget only for η that satisfies the condition. Applying the sparse vector algorithm on a randomly subsampled data further allows the algorithm to relax the budget constraint using the recent privacy amplification results [13]. While in a deterministic (i.e., noise-free) setting, it is guaranteed that there exists η that satisfies the condition (2), in a stochastic private setting, the backtracking algorithm may fail to terminate or return an arbitrarily small step size due to the noise from two different sources: (i) gradient approximation and (ii) noise added for privacy. When the backtracking algorithm fails to return within the pre-specified number of iterations, to decide whether more accurate gradients are necessary, the proposed algorithm evaluates another gradient at $\mathbf{w}_t$ and measure the angle between two gradients. When two gradients evaluated at $\mathbf{w}_t$ are pointing to very different directions, the algorithm increases the privacy budget for gradient evaluation.

Our contributions are summarized as follows:

- We propose a Rényi differentially private SGD with Armijo line search. To the best of our knowledge, this is the first private SGD algorithm with line search ability.
- We introduce an adaptive privacy budget controlling strategy based on the moving average of angles between consecutive gradients, which detects if gradients are pointing to very different directions.
- To evaluate the effectiveness of the proposed algorithm, we conduct extensive experiments on real datasets and compare its performance to existing algorithms.

The rest of this paper are organized as: Section II reviews the related work; Section III summarizes important definitions and lemmas used in the paper; Section IV presents the main algorithm; and experimental results are presented in Section V; Section VI concludes the paper.

II. Related Work

Many techniques have been proposed for first-order optimization algorithms in non-private setting, focusing on step size selection or reducing the noise involved in stochastic gradients, such as Adam [14], SVRG [15], SplitSGD [16], etc. The technique related to this paper is Amijo line search [9], which is a classic and famous step size selection approach. Recent works [11], [17] have shown that combining SGD with line-search achieves fast convergence for both convex and non-convex problems, and is robust to the precise choices of hyper-parameters, for over-parameterized models, with the price of additional objective evaluation (feed-forward steps for neural networks). In this paper, we show that, with essential randomization techniques, it can fit well into the privacy framework, and the privacy budget can be carefully controlled.

There are many differentially private mechanisms we can use to release various statistics. One advanced tool highly related to our paper is the sparse vector technique (SVT) [2]. The sparse vector algorithm sequentially processes a sequence of threshold queries. For each query in the sequence, the algorithm evaluates it with noise, compares the result with the noisy threshold, and outputs the binary value. The carefully scaled noise ensures that the algorithm only pays the privacy budget when the query is privately evaluated as above the threshold. Although [12] shows that many extensions of SVT are not private, it also demonstrates the correctness of the original version (used in our approach), which is further confirmed in [18].

Differentially private optimization algorithms can be roughly grouped into three categories. Output perturbation algorithms train a model without noisy perturbation, then perturb the model before releasing, based on a calculation of sensitivity, such as [19]–[22]. Objective perturbation algorithms protect the privacy of the training data through optimizing a noise-perturbed objective, for example, [19], [23]. The aforementioned algorithms usually put strict assumptions on the objective functions, such as convexity and smoothness, which limits their applicable domain. The type of algorithms mostly related to this paper is the gradient perturbation algorithms, which perturb the data-dependent intermediate results (i.e. gradients) during the model training, and the total privacy is calculated by compositing the privacy costs of all iterations. Since privacy is achieved immediately after the data-dependent step, gradient perturbation algorithms do not put strict assumptions on the objective function, and can be applied in a broader range of problems, such as neural networks. The first gradient perturbation algorithm was proposed in [7], with the "strong composition" method to account for the privacy loss over multiple iterations. Later, "moment accountant" method [4] gave a tighter bound on privacy amplification and accountant. These algorithms directly satisfy (ϵ, δ)-differential privacy, and many algorithms were build based on them, such as [24], [25]. These algorithms take gradient calculation as the only data-dependent step, thus there is no extra source of information which can be used to adaptively tune hyperparameters such as step size, per-iteration privacy budget, and/or clipping threshold. [8] is an exception, which proposed an adaptive gradient perturbation algorithm based on full gradient descent, with extra budget paid for objective evaluation, and it satisfies zCDP, without privacy amplification. The convergence property of private optimization algorithms is showed in [26], [27].

Rényi differential privacy (RDP) is a recent privacy framework proposed in [3], which stands between pure and approximate DP, and its privacy amplification lemma is presented in [13]. The privacy guarantee of our algorithm fits into the RDP framework, and we show that it can help account for the two sources of privacy leaks. This differs from the "moment accountant" technique, which only accounts for Gaussian perturbations, and also different from the zCDP framework, which does not yet have privacy amplification of sub-sampling.

III. Preliminaries

We assume the training data D is composed of n individuals: $\{\mathbf{d}_1, \dots, \mathbf{d}_n\}$, stored at a trusted data curator. For labeled data, each individual datum contains a feature vector and its label. Two datasets D and D' are considered to be *neighboring* if they differ by one individual, i.e., $|(D \setminus D') \cup (D' \setminus D)| = 1$, denoted by $D \sim D'$. We use bold-face letters to represent vectors and a subscript to indicate iteration number (e.g. $\mathbf{w}_t$ denotes the value of $\mathbf{w}$ at iteration t).

Differential privacy is a *de facto* standard for protecting the privacy of individuals in sensitive datasets.

Definition 1 ((ϵ, δ)-Differential Privacy (DP))**.** *[1] [28] Given privacy parameters $\epsilon \geq 0, 0 \leq \delta \leq 1$, a randomized mechanism $\mathcal{M}$ satisfies (ϵ, δ)-DP if for every event $S \subseteq range(\mathcal{M})$, and for every pair of neighboring datasets $D \sim D'$,*

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\epsilon} \Pr[\mathcal{M}(D') \in S] + \delta \,. \tag{3}$$

When $\delta = 0$, it is called *pure* DP and when $\delta > 0$, it is referred to as *approximate* DP.

Rényi Differential Privacy (RDP) is a relaxation of pure-DP and tracks privacy leakage of data access using Rényi divergence.

Definition 2 (Rényi Divergence)**.** *For probability distributions $P(x)$ and $Q(x)$ over a set Ω, and let $\alpha \in (1, +\infty)$. Then Rényi α-divergence $D_\alpha[P(x)\|Q(x)] := \frac{1}{\alpha-1}\log\left[P(x)^\alpha Q(x)^{1-\alpha}\right]$, where $P(x)$ and $Q(x)$ are pdf (or pmf) of the distributions, and α is the* order *of the divergence.* [1]

Definition 3 ((α, ϵ)-Rényi Differential Privacy (RDP))**.** *[3] Given a real number $\alpha \in (1, +\infty)$ and privacy parameter $\epsilon \geq 0$, a randomized mechanism $\mathcal{M}$ satisfies (α, ϵ)-RDP if for every pair of neighboring datasets $D \sim D'$, the Rényi α-divergence between $\mathcal{M}(D)$ and $\mathcal{M}(D')$ satisfies*

$$D_\alpha[\mathcal{M}(D)\|\mathcal{M}(D')] \leq \epsilon$$

When $\alpha = +\infty$, (α, ϵ)-RDP coincides with $(\epsilon, 0)$-DP. The privacy guarantee of RDP can be converted to and interpreted in terms of (ϵ, δ)-DP using the following result.

Proposition 1 (RDP to (ϵ, δ)-DP)**.** *[3] If $\mathcal{M}$ satisfies (α, ϵ)-RDP, then it satisfies (ϵ', δ)-DP for $\epsilon' = \epsilon + \frac{\log(1/\delta)}{\alpha-1}$.*

One method to achieve RDP is through the Gaussian mechanism, which scales noise to the L_2 sensitivity of a query.

Definition 4 (L_1 (resp. L_2) Sensitivity)**.** *Let $q : \mathcal{D}^n \to \mathbb{R}^k$ be a vector-valued function over datasets. The L_1 (resp. L_2) sensitivity of q, denoted as $\Delta_1(q)$ (resp. $\Delta_2(q)$), is defined as $\Delta_r(q) = \sup_{D\sim D'} \|q(D) - q(D')\|_r$, for $r = 1$ (resp. 2).*

Lemma 1 (Gaussian Mechanism)**.** *[3] Let $q : \mathcal{D}^n \to \mathbb{R}^k$ be a vector-valued function over datasets. Let $\mathcal{M}$ be a mechanism releasing $q(D) + \gamma$ where $\gamma \sim \mathcal{N}(0, \sigma^2\mathbb{I}_k)$, then $\mathcal{M}$ is $(\alpha, \epsilon(\alpha))$-RDP for $\epsilon(\alpha) = \alpha\Delta_2^2(q)/(2\sigma^2)$.*

[1]To avoid confusion, we use the curly α to denote the order of Rényi divergence and RDP, and plain α to denote the hyperparameter in Armijo condition.

Equivalently, Gaussian mechanism ensures $\mathcal{M}$ to satisfy $(\alpha, \alpha\rho)$-RDP for $\alpha > 1$, where $\rho := \Delta_2^2(q)/(2\sigma^2)$ can be considered as a "privacy budget" independent of α.

Other important lemmas about RDP include:

Lemma 2 (ϵ-DP to RDP)**.** *[29] If $\mathcal{M}$ satisfies (ϵ, 0)-DP, then the Rényi divergence $D_\alpha[\mathcal{M}(D)\|\mathcal{M}(D')] \leq \frac{1}{2}\alpha\epsilon^2$. In other words, $\mathcal{M}$ also satisfies $(\alpha, \frac{1}{2}\alpha\epsilon^2)$-RDP.*

Lemma 3 (Private composition for RDP)**.** *[3] For mechanisms $\mathcal{M}_1$ and $\mathcal{M}_2$ applied on dataset D, if $\mathcal{M}_1$ satisfies (α, ϵ_1)-RDP and M_2 satisfies (α, ϵ_2)-RDP, then $\mathcal{M}_1 \circ \mathcal{M}_2$ satisfies $(\alpha, \epsilon_1 + \epsilon_2)$-RDP.*

Lemma 4 (Subsampled Mechanism and Privacy Amplification for RDP)**.** *[13] For a randomized mechanism $\mathcal{M}$ and a dataset D, if $\mathcal{M}$ satisfies $(\alpha, \epsilon(\alpha))$-RDP with respect to B, where B is a subsample of D sampled by $B = \{\mathbf{d}_i | \iota_i = 1, \iota_i \overset{i.i.d}{\sim} Bernoulli(q) \text{ for } i \in [n]\}$. Then then $\mathcal{M}$ satisfies $(\alpha, \epsilon'(\alpha))$-RDP with respect to D for any integer $\alpha \geq 2$, where $\epsilon'(\alpha) \leq \frac{1}{\alpha-1}\log\left\{(1-q)^{\alpha-1}(\alpha q - q + 1) + \binom{\alpha}{2}q^2(1-q)^{\alpha-2}e^{\epsilon(2)} + 3\sum_{l=3}^{\alpha}\binom{\alpha}{l}q^l(1-q)^{\alpha-l}e^{(l-1)\epsilon(l)}\right\}$.*

The Sparse Vector is a technique used to answer a sequence of threshold queries $\{q_i\}$, $i = 1, 2, \cdots$. Given a publicly known threshold T, it sequentially processes each q_i and produces an output $a_i \in \{\top, \perp\}$. Each a_i indicates whether $q_i(D)$ is above ($\top$) or below ($\perp$) the threshold. It terminates after outputting the predefined number c of "$\top$" values, and its privacy cost is proportional to c. In other words, given a fixed privacy budget, it can release binary answers to threshold queries until it outputs c "above" threshold answers regardless of how many "below" threshold answers are generated. AboveThreshold is a basic version with $c = 1$.

Lemma 5 (Above Threshold Mechanism)**.** *[2] Let $\{q_i\} = q_1, q_2, ...$ be a series of queries having the same L_1 sensitivity $\Delta_1(q)$, and T be a publicly known threshold. The* AboveThreshold *algorithm first perturbs T by adding Laplace noise, i.e, $\hat{T} = T + \lambda$ where $\lambda \sim Lap(0, \frac{2\Delta_1(q)}{\epsilon})$ and generates output $\{a_i\}$ as follows.*

$$a_i = \begin{cases} \top & \text{if } q_i(D) + \nu_i \geq \hat{T}, \\ \perp & \text{if } q_i(D) + \nu_i < \hat{T}, \end{cases}$$

where $\nu_i \sim Lap(0, \frac{4\Delta_1(q)}{\epsilon})$. The mechanism terminates if $a_i = \top$. The AboveThreshold *satisfies $(\epsilon, 0)$-DP.*

IV. Algorithms

This section describes each component of the proposed algorithm in detail.

A. Noisy Backtracking Line Search

We start with Noisy Backtracking Line Search (NoisyBTLS) algorithm which performs backtracking line search in a differentially private manner. The pseudocode of the algorithm is shown in Algorithm 1. NoisyBTLS is an application of AboveThreshold algorithm [2], introduced in Lemma 5, to a line search task.

Algorithm 1 Noisy Backtracking Line Search (NOISYBTLS), Laplace [resp. Gaussian] version

```
1: Input: Objective function $f$, dataset $D$, model parameter $\mathbf{w}$, gradient $\mathbf{g}$, initial learning rate $\eta_0$, privacy budget $\epsilon_{BT}$ [resp. $\rho_{BT}$], sensitivity $\Delta_q$.
2: Hyper-parameters: $\alpha, \beta$, maximum iterations max_it.
3: $\epsilon_1 \leftarrow \frac{\epsilon_{BT}}{2}$, $\epsilon_2 \leftarrow \frac{\epsilon_{BT}}{4}$ [resp. $\sigma_1^2 \leftarrow \frac{3}{2\rho_{BT}}, \sigma_2^2 \leftarrow \frac{3}{\rho_{BT}}$]
4: Sample noisy threshold $\hat{T} = \lambda$, where $\lambda \sim Lap(0, \frac{\Delta_q}{\epsilon_1})$ [resp. $\lambda \sim \mathcal{N}(0, \Delta_q^2\sigma_1^2)$]
5: $\eta \leftarrow \eta_0$
6: for $i = 1, 2, \ldots,$ max_it do
7:     $q_i \leftarrow f(\mathbf{w}; D) - \alpha\eta\|\mathbf{g}\|_2^2 - f(\mathbf{w} - \eta\mathbf{g}; D)$
8:     $\hat{q}_i \leftarrow q_i + \nu_i$ where $\nu_i \sim Lap(0, \frac{\Delta_q}{\epsilon_2})$ [resp. $\nu_i \sim \mathcal{N}(0, \Delta_q^2\sigma_2^2)$]
9:     if $\hat{q}_i \geq \hat{T}$ then
10:        Output: $\eta$                ▷ found a suitable step size
11:    $\eta \leftarrow \beta\eta$
12: Output: 0     ▷ failed to find $\eta$ within max_it iterations
```

The algorithm starts by adding noise to the threshold $T = 0$, producing a noisy threshold $\hat{T} = \lambda$, where λ is a random noise drawn from a Laplace distribution. Instead of Laplace noise, one can also choose to add Gaussian noise in Algorithm 1, and we show in Theorem 3 that the algorithm with Gaussian noise satisfies RDP. At each iteration, the algorithm evaluates a query $q_i(\eta_i, D) = f(\mathbf{w}) - f(\mathbf{w} - \eta_i\nabla f(\mathbf{w})) - \alpha\eta_i\|\nabla f(\mathbf{w})\|_2^2$ with noise ν_i and compares it (i.e., $q_i + \nu_i$) with the noisy threshold $\hat{T}$. If $q_i + \nu_i \geq \hat{T}$, the algorithm outputs η_i and halts. Otherwise, it decreases the step size η_i by multiplying with β and continues with the next iteration. Here, $\beta \in (0, 1)$ is a user-defined multiplicative factor that determines how fast the step size is decreased. One crucial difference with the original ABOVETHRESHOLD algorithm is that we set a limit on the number of iterations. If there is no limit, when the query value is dominated by noise, it would fail to terminate or returns a too small step size, which does not help make progress and could lead to increase in the objective value at the next iteration. Hence, when the algorithm fails to return within the specified maximum number of iterations, the algorithm computes a diagnostic statistic to test whether higher privacy budget is necessary and adjusts the budget according to the test result. We discuss details of this procedure in Section IV-B. The use of Sparse Vector technique in Algorithm 1 significantly reduces the privacy budget needed to find η, from a scale linear to the size of the search space to a constant, which greatly improves its utility. A naive implementation (e.g. applying a naive additive privacy composition lemma [28]) would result in $(\epsilon_1 + \texttt{max\_it} \cdot \epsilon_2, 0)$-DP.

Theorem 1. *Let Δ_f be an upper bound on the objective function f such that $|f(\mathbf{w}; \mathbf{d})| \leq \Delta_f$ for $\forall\mathbf{d} \in \mathcal{D}$ and $\mathbf{w} \in \Theta$. Given the candidate gradient $\mathbf{g}$ either privately released or publicly available, Algorithm 1 with Laplace noise, $\epsilon_1 = \frac{\epsilon}{2}, \epsilon_2 = \frac{\epsilon}{4}$, and $\Delta_q = \Delta_f$ satisfies $(\epsilon, 0)$-DP.*

The proof is presented in the full version of the paper [30]. Theorem 1 requires f is upper bounded by a constant Δ_f. If there is no a priori known upper bound on a loss function f, we enforce the bound by applying the objective clipping [8]: $f(\mathbf{w}; D) = \sum_{i=1}^{n} \min\{f(\mathbf{w}; \mathbf{d}_i), \Delta_f\}$. Since the Laplace version of Algorithm 1 is ϵ-DP, one can use Lemma 2 to convert its privacy guarantee to that of RDP. Instead, in the following theorem, we directly derive the Rényi divergence of output distributions between two neighboring datasets and show it results in a tighter bound on the privacy loss.

Theorem 2. *Under the same conditions of Theorem 1, the Laplace version of Algorithm 1 is $(\alpha, \epsilon(\alpha))$-RDP, where*

$$\epsilon(\alpha, \epsilon_1, \epsilon_2) = \frac{1}{\alpha - 1}\log\left\{\left[\frac{\alpha}{2\alpha - 1}e^{\epsilon_1(\alpha-1)} + \frac{\alpha - 1}{2\alpha - 1}e^{-\epsilon_1\alpha}\right] \cdot \left[\frac{\alpha}{2\alpha - 1}e^{2\epsilon_2(\alpha-1)} + \frac{\alpha - 1}{2\alpha - 1}e^{-2\epsilon_2\alpha}\right]\right\} \tag{4}$$

We next show that Algorithm 1 with Gaussian noise also satisfies RDP.

Theorem 3. *Under the same conditions of Theorem 1, the Gaussian version of Algorithm 1 is $(\alpha, \epsilon(\alpha))$-RDP, where*

$$\epsilon(\alpha, \sigma_1^2, \sigma_2^2) = \alpha(4\sigma_1^2 + \sigma_2^2)/2\sigma_1^2\sigma_2^2 \tag{5}$$

The proofs of both theorems are presented in the full version of the paper [30]. One can easily verify from (5) that, when only one privacy parameter ρ is given, running Gaussian version of NOISYBTLS with $\sigma_1^2 \leftarrow 3/(2\rho), \sigma_2^2 \leftarrow 3/\rho$ would satisfy $(\alpha, \alpha\rho)$-RDP.

B. Private Backtracking Line Search Based Stochastic Gradient Descent

Now we present our main algorithm, called Differentially Private Backtracking Line Search-based Stochastic Gradient Descent (DP-BLSGD). Algorithm 2 shows the pseudocode.

Starting with initial parameter vector $\mathbf{w}_0$, at iteration t, the algorithm evaluates the gradient $\nabla f(\mathbf{w}_t)$ over a mini-batch B. To bound the sensitivity, it applies the gradient clipping [4]. Specifically, it computes the per-example gradient $\overline{\mathbf{g}}_i = \nabla f_i(\mathbf{w}_t, \mathbf{d}_i)$ for each $\mathbf{d}_i \in B$ and applies the clipping function to $\overline{\mathbf{g}}_i$, i.e., $\mathsf{clip}(\overline{\mathbf{g}}_i, C_{grad})$. The clipping function is defined as

$$\mathsf{clip}(\mathbf{g}, C) = \frac{\mathbf{g}}{\max(1, \|\mathbf{g}\|_2/C)}\,. \tag{6}$$

The application of clipping function in line 5 ensures that the L_2 norm of every per-example gradient in the summation is no greater than the threshold C_{grad}, and hence it bounds the L_2 sensitivity of summed gradient to C_{grad}. After summing the clipped per-example gradients, it adds Gaussian noise with variance $C_{grad}^2/2\rho_{grad}$ to each coordinate.

The step size η for iteration t is computed by calling the NOISYBTLS function with passing noisy gradient $\tilde{\mathbf{g}}_t$ as input. When the step size η returned by NOISYBTLS is greater than 0, the algorithm performs SGD update in line 13.

Algorithm 2 Rényi Differentially Private Backtracking Line Search Based Sub-sampled Gradient Descent (DP-BLSGD)

```
1:  **Input**: Dataset $D = \{\mathbf{d}_1, ..., \mathbf{d}_n\}$, loss function $F(\mathbf{w}, D)$, clipping thresholds $C_{obj}, C_{grad}$, sampling ratio $q$, privacy budget for line search $\epsilon_{BT}$, privacy budget for gradient $\rho_{grad}$, budget increase rate $\xi$, initial learning rate $\eta_0$, total privacy budget $\epsilon_{tot}(\alpha)$.
2:  **Initialize** $\mathbf{w}_0$ randomly. $\overline{\theta} \leftarrow 90°$.
3:  **for** $t = 0, 1, ...$ **do**
4:      Sample a mini-batch $B$ by sampling ratio $q$
5:      $\mathbf{g}_t \leftarrow \sum_{\mathbf{d}_i \in B} \mathsf{clip}(\nabla f_i(\mathbf{w}_t, \mathbf{d}_i), C_{grad})$    ▷ see (6)
6:      $\tilde{\mathbf{g}}_t \leftarrow \frac{1}{|B|}(\mathbf{g}_t + \gamma)$, where $\gamma \sim \mathcal{N}(0, (C^2_{grad}/2\rho_{grad})\mathbb{I})$
7:      $\epsilon_{tot}(\alpha) \leftarrow \epsilon_{tot}(\alpha) - \mathsf{amp}(\alpha\rho_{grad})$    ▷ see IV-C
8:      $\eta \leftarrow 0$, $\epsilon_{ls}(\alpha) \leftarrow \epsilon(\alpha, \epsilon_{BT}/2, \epsilon_{BT}/4))$    ▷ see (4)
9:      **while** $\eta = 0$ and $\epsilon_{tot}(\alpha) > \epsilon_{ls}(\alpha)$ **do**
10:         $\eta \leftarrow$ NoisyBTLS$(f, B, \mathbf{w}_t, \tilde{\mathbf{g}}_t, \eta_0, \epsilon_{BT})$
11:         $\epsilon_{tot}(\alpha) \leftarrow \epsilon_{tot}(\alpha) - \mathsf{amp}(\epsilon_{ls}(\alpha))$    ▷ see IV-C
12:         **if** $\eta > 0$ **then**
13:             $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t - \eta\tilde{\mathbf{g}}_t$
14:             Update $\theta_t$ and $\overline{\theta}$ according to (7)
15:         **else if** $\epsilon_{tot}(\alpha) > \alpha\rho_{grad}$ **then**
16:             $\epsilon_{tot}(\alpha) \leftarrow \epsilon_{tot}(\alpha) - \mathsf{amp}(\alpha\rho_{grad})$  ▷ see IV-C
17:             $\rho_{grad}, \epsilon_{BT}, \tilde{\mathbf{g}}_t \leftarrow$ CheB$(\rho_{grad}, \epsilon_{BT}, \xi, \tilde{\mathbf{g}}_t)$
18:         **else**
19:             **break**
20:         $\epsilon_{ls}(\alpha) \leftarrow \epsilon(\alpha, \epsilon_{BT}/2, \epsilon_{BT}/4))$    ▷ see (4)
21:     **if** $\epsilon_{tot}(\alpha) \leq \alpha\rho_{grad}$ **then**
22:         **break**
23: **Output**: $\mathbf{w}_{t+1}$
```

We now discuss how the proposed algorithm dynamically adjusts the privacy budget to account for the case in which the algorithm fails to find a reasonably large step size.

a) Privacy budget adaptation: When NoisyBTLS fails to find a step size within `max_it` iterations, there are two possiblities. First, the current privacy budget ρ_{grad} assigned for evaluating the gradient is too small that noise dominates the gradient. The remedy for this case is to increase the privacy budget. The second possibility is that ρ_{grad} is large enough not to remove the gradient signal but the large noise in NoisyBTLS prevents it from finding the step size satisfying the condition (2). In this case, we need a more accurate measurement of gradient and an increased privacy budget for the backtracking line search. To distinguish these two case, DP-BLSGD maintains the moving average of angles between two consecutive gradients, and it is updated at every iteration (line 14 in Algorithm 2) as follows:

$$\begin{aligned} \theta_t &\leftarrow \text{AngleBetween}(\tilde{\mathbf{g}}_t, \tilde{\mathbf{g}}_{t-1}) \\ \overline{\theta} &\leftarrow \psi\overline{\theta} + (1-\psi)\theta_t \,, \end{aligned} \tag{7}$$

where $\psi \in (0, 1)$ is a parameter controlling the decay rate of old information. Note that $\overline{\theta}$ is initialized to 90° for the first iteration and line 14 is not executed when $t = 0$. When $\eta = 0$ is returned by NoisyBTLS, the algorithm evaluates another gradient $\tilde{\mathbf{g}}_{t2}$ using the budget of ρ_{grad} and measures the angle θ between $\tilde{\mathbf{g}}_t$ and $\tilde{\mathbf{g}}_{t2}$ (line 5 in Algorithm 3). If θ is greater than the moving average-based threshold θ_{max}, the algorithm increases the privacy budget ρ_{grad} for gradient computation. When θ is smaller than the minimum threshold θ_{min}, it indicates that the search might fail because the privacy budget ϵ_{BT} assigned for noisy backtracking line search (i.e., Algorithm 1) is too small. Hence, we increase ϵ_{BT} in this case. The threshold values θ_{max} and θ_{min} are calculated as follows:

Algorithm 3 Check and Enlarge Budget for SGD (CheB)

```
1:  **Input**: current budget $\rho_{grad}$, $\epsilon_{BT}$, budget increase rate $\xi$, perturbed gradient $\tilde{\mathbf{g}}_t$.
2:  Sample a mini-batch $B$ by sampling ratio $q$
3:  $\mathbf{g}_{t2} \leftarrow \sum_{\mathbf{d}_i \in B} \left(\nabla f(\mathbf{w}_t; \mathbf{d}_i) / \max(1, \frac{\|\nabla f(\mathbf{w}_t; \mathbf{d}_i)\|_2)}{C_{grad}})\right)$
4:  $\tilde{\mathbf{g}}_{t2} \leftarrow \frac{1}{|B|}(\mathbf{g}_{t2} + \gamma_2)$, where $\gamma_2 \sim \mathcal{N}(0, (C^2_{grad}/2\rho_{grad})\mathbb{I})$
5:  $\theta \leftarrow$ AngleBetween$(\tilde{\mathbf{g}}_t, \tilde{\mathbf{g}}_{t2})$
6:  Calculate $\theta_{max}$ and $\theta_{min}$    ▷ see (8)
7:  **if** $\tilde{\mathbf{g}}_t \cdot \tilde{\mathbf{g}}_{t2} < 0$ or $\theta > \theta_{max}$ **then**
8:      $\rho_{grad} \leftarrow (1+\xi)\rho_{grad}$
9:  **else if** $\theta < \theta_{min}$ **then**
10:     $\epsilon_{BT} \leftarrow (1+\xi)\epsilon_{BT}$
11: $\tilde{\mathbf{g}}_t \leftarrow (\tilde{\mathbf{g}}_t + \tilde{\mathbf{g}}_{t2})/2$
12: **Output**: $\rho_{grad}, \epsilon_{BT}, \tilde{\mathbf{g}}_t$
```

$$\theta_{max} \leftarrow \phi_{max} \times \overline{\theta}\,, \quad \theta_{min} \leftarrow \phi_{min} \times \overline{\theta}\,, \tag{8}$$

where $\phi_{max} > 1$ and $0 < \phi_{min} < 1$ are hyper-parameters. Empirically, we observe this budget adaptation strategy is especially effective for convex optimization problems.

b) Intelligent backtracking: To reduce the number of times the algorithm redundantly backtracks due to unnecessarily large initial step size η_0, DP-BLSGD maintains a list Ω of previously selected step sizes. After every τ iterations, η_0 is updated as $\eta_0 \leftarrow \min\{\varsigma \cdot \max\{\Omega\}, \eta_0\}$, and Ω is reset to an empty set. The parameter $\varsigma > 1$ guarantees the line search starts with sufficiently large initial step size but not too large to avoid redundant backtrackings. In our experiments in Section V, we set $\varsigma = 1.2$. Note that, in a non-private setting, the line search algorithm proposed in [11] resets the initial step size in a similar way, but it simply resets η_0 to a multiple of η selected in the previous iteration. However, in a private setting, this strategy of resetting η_0 at every iteration can make the search unstable as step sizes selected by a noisy backtracking algorithm can fluctuate due to noise.

Theorem 4. *Algorithm 2 satisfies RDP.*

The proof is presented in the full version of the paper [30].

C. Privacy Budget Tracking

Each sub-routine in the proposed algorithm incurs different amount of privacy loss. To ensure that the total privacy budget spent by the algorithm is smaller than the given total privacy budget $\epsilon_{tot}(\alpha)$ (so that the entire algorithm satisfies

$(\alpha, \epsilon_{tot}(\alpha))$-RDP), the algorithm computes the total privacy loss incurred by each function call under RDP framework. Specifically, it computes the amount of required privacy budget using Theorem 2 and Lemma 1, followed by privacy amplification through Lemma 4 (denoted as amp in Algorithm 2). Following the RDP composition (Lemma 3), the algorithm subtracts it from the running privacy budget before calling each sub-routine. Recall that in RDP the privacy loss is a function of α (the order of Rényi divergence). In a practical implementation of the algorithm to satisfy (ϵ', δ)-DP, one can first calculate $(\alpha, \epsilon(\alpha))$ for a series of α values (e.g., integers between 2 and 500). Its privacy parameters can be converted into those of (ϵ', δ)-DP using the conversion result in Proposition 1. During the execution, the algorithm maintains and keeps track of total privacy budget spent for each value of α. The algorithm halts and returns the result when the remaining budget $\epsilon(\alpha)$ for all α values are lower than the minimum budget required to call a sub-routine.

D. Clipping Threshold Adaptation

The gradient and objective clipping techniques allow to effectively bound the sensitivity but, if they are used with incorrectly chosen threshold values, they can degrade the utility. For example, if C_{grad} is too high but the norm of gradient is small, it is likely that the noise dominates the gradient due to high sensitivity. On the other hand, if C_{grad} is too small, then it clips out useful information. During the model training, the norm of gradients decreases as the parameter vector $\mathbf{w}_t$ gets closer to the optimal values, and hence a large clipping threshold might not be necessary in the later stage of training. Motivated by this, we propose to adaptively decrease the clipping threshold C_{grad} and C_{obj} if the algorithm decides to increase ρ_{grad} (line 8 in Algorithm 3) during a single SGD update. The algorithm decreases the threshold only once per each SGD update regardless of how many times ρ_{grad} is increased.

$$C_{grad} \leftarrow (1-\zeta) C_{grad}\,, \quad C_{obj} \leftarrow (1-\zeta) C_{obj}\,, \tag{9}$$

where ζ is a hyperparameter that determines the rate of decrease. The proposed algorithm with the clipping threshold adaptation is called DP-BLSGD-AC. Note that this strategy does not require any extra privacy budget since the condition is based on privately released information.

V. Experimental Results

A. Models

We evaluate the performance of proposed algorithm on both convex and nonconvex problems. For convex problems, we consider training two models: logistic regression and linear SVM. Let $\mathbf{d}_i = (\mathbf{x}_i, y_i)$, for $i = 1, \ldots, n$, where $\mathbf{x}_i$ is a feature vector and $y_i \in \{-1, +1\}$ is its label. The objective function of logistic regression is

$$F(\mathbf{w}; D) := \frac{1}{n} \sum_{i=1}^{n} \log(1 + \exp(-y_i \mathbf{w}^T \mathbf{x}_i))\,,$$

which is convex and smooth. For SVM, we use the hinge loss, which is convex but non-smooth, and

$$F(\mathbf{w}; D) := \frac{1}{n} \sum_{i=1}^{n} \max(0, 1 - y_i \mathbf{w}^T \mathbf{x}_i)\,.$$

We also apply our algorithms on non-convex problems, training neural networks for image classification tasks. We trained the neural networks with two different architectures: multi-layer perception (MLP) and convolution neural network (CNN). The details of network architectures are discussed in Section V-G. To avoid overfitting, we applied a L_2 regularization on the model parameters with a coefficient $\mu = 0.001$ for all models.

TABLE I: Summary of datasets

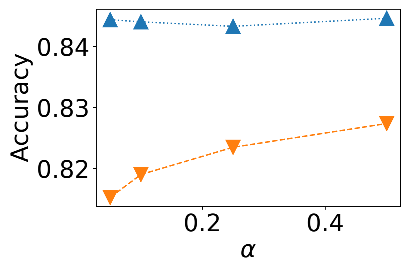
Dataset	Size	Dimension	Baseline
Adult	48,842	124	0.761
Bank	45,211	33	0.883
IPUMS-BR	38,000	53	0.507
IPUMS-US	40,000	58	0.513
MNIST	60,000/10,000	$1 \times 28 \times 28$	$\sim$0.1
FMNIST	60,000/10,000	$1 \times 28 \times 28$	$\sim$0.1
Cifar-10	50,000/10,000	$3 \times 32 \times 32$	$\sim$0.1

B. Datasets and Pre-proessing

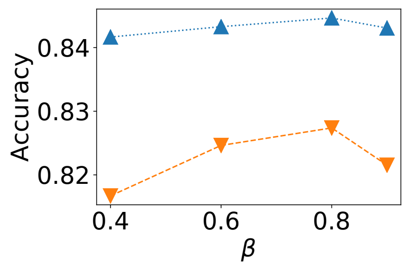
A summary of all datasets used in our experiments are shown in Table I. Four census datasets were used for convex optimization: Adult [31], Bank [32], IPUMS-BR, and IPUMS-US [33]. All categorical attributes are pre-processed by one-hot encoding, and numeric ones are scaled to [0, 1]. Three datasets were used for training neural networks: MNIST, FMNIST, and Cifar-10. Note that these three datasets have separate dataset for testing. For other datasets, we report the averaged performance of 10-fold cross validation. MNIST and Fashion MNIST (FMNIST) datasets contain gray-scale images, while Cifar-10 dataset consists of RGB images. Each pixel in each channel is re-scaled into [-1, 1]. We run each experiment 5 times and report the averaged performance.

C. Baselines

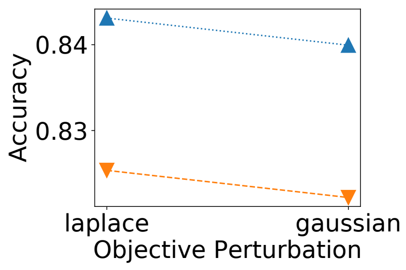
For convex problems, we compare the performance of our proposed algorithms, DP-BLSGD, DP-BLGD, and DP-BLSGD-AC, with 8 baseline algorithms: DP-AGD [8], DP-SGD [4], OutPert-RSGD [22], OutPert-GD [20], ObjPert [19], [23], PrivGene [34], majority, and non-private. DP-AGD is the adaptive full-batch gradient descent algorithm, which selects step size by NoisyMin. DP-BLGD is the batch gradient descent version of our DP-BLSGD, which uses the budget increasing technique of DP-AGD. DP-SGD is the gradient perturbation algorithm presented in [4]. OutPert-RSGD and OutPert-GD are both output perturbation algorithms, the former calculates sensitivity based on permuted SGD with averaging, and the latter calculates sensitivity depending on batch gradient descent. ObjPert is the objective perturbation algorithm which inject noise into loss



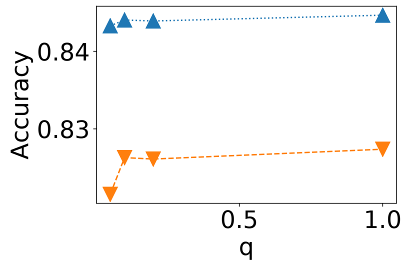
(a) α for BTLS



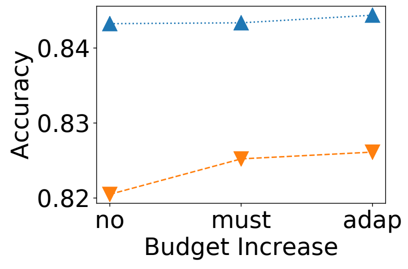
(b) β for BTLS



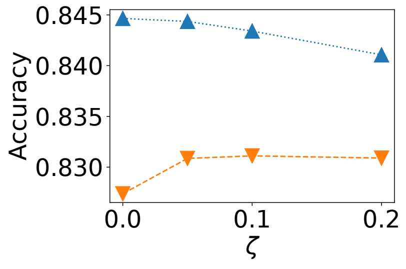
(c) Noise distribution



(d) Sampling ratio q



(e) Budget allocation strategy



(f) Clipping threshold decay rate ζ

Fig. 1: Impact of hyperparameters on the performance (▲: $\epsilon = 0.2$, ▼: $\epsilon = 0.05$)

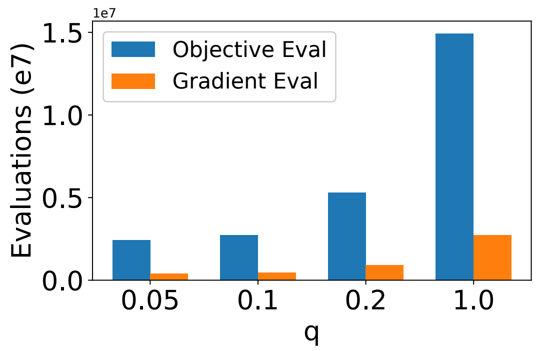


Fig. 2: Number of gradient and objective evaluations for different sampling ratio at $\epsilon = 0.4$

function. PRIVGENE is private model fitting based on genetic algorithm. NON-PRIVATE is the non-private baseline, which uses L-BFGS to search for an optimal solution. MAJORITY classifies every sample as the major class. For the baseline algorithms which require smoothness of the loss function, the SVM experiments are performed on Huberized SVM, where

$$F(\mathbf{w}, D) := \frac{1}{n}\sum_{i=1}^{n}\begin{cases}1 - y_i\mathbf{w}^T\mathbf{x}_i & \text{if } y_i\mathbf{w}^T\mathbf{x}_i < 1-\hbar\\ \frac{1}{4\hbar}(1+\hbar-y_i\mathbf{w}^T\mathbf{x}_i)^2 & \text{if } |1-y_i\mathbf{w}^T\mathbf{x}_i| \le \hbar\\ 0 & \text{otherwise}\end{cases}$$

and $\hbar$ is a hyperparameter set to 0.5.

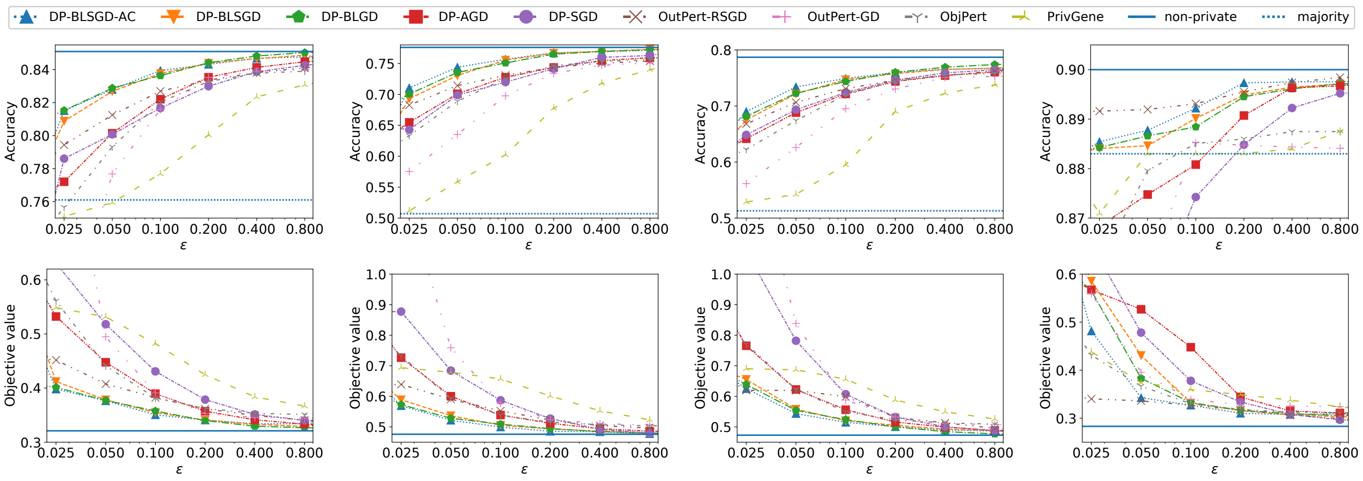
For neural network models, we compare our algorithms with the gradient perturbation algorithms proposed in [4], which injects Gaussian noise into the sub-sampled clipped gradients, for both SGD and Adam versions. The NON-PRIVATE baseline shows the performance of ADAM optimizer.

D. Hyperparameter setting

We fix $\delta = 10^{-8}$ for all experiments. In order to make fair comparisons between algorithms, we convert RDP into (ϵ, δ)-DP using the conversion tool given in Proposition 1. For convex models, we set the hyperparameters as follows. To initialize the initial privacy budget ϵ_{BT} and ρ_{grad}, we heuristically determine a per-iteration budget as $\epsilon_{iter} = \epsilon/(2 \times 50)$. The intuition behind this setting is that we expect the algorithm would approximately require 50 iterations. Given the per-iteration ϵ_{iter}, we set $\epsilon_{BT} = \epsilon_{iter}$ and $\rho_{grad} = \frac{1}{2}\epsilon_{iter}^2$.

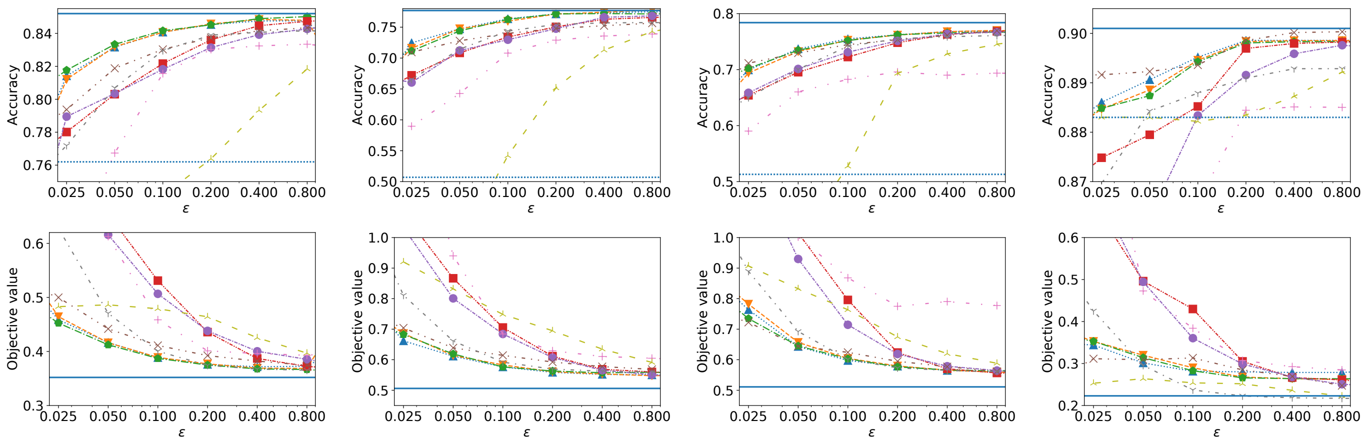
The sampling rate is set to $q = 0.1$, and the clipping thresholds are set as $C_{grad} = 3$ and $C_{obj} = 1$ for all gradient perturbation algorithms. The privacy budget increase parameter is set as $\xi = 0.3$. The hyperparameters of NOISYBTLS algorithm are set as follows: $\alpha = 0.5, \beta = 0.8, \tau = 10, \varsigma = 1.2, \phi_{max} = 1.1, \phi_{min} = 0.5$, and $\psi = 0.8$. For DP-BLSGD-AC, we set the clipping threshold decrease rate parameter $\zeta = 0.05$. Hyperparameters of the baseline algorithms are set as suggested in their papers.

For neural network models, we set subsampling rate $q = 1/200$, $C_{grad} = 3$, and $C_{obj} = 3$. Since all the algorithms being compared can achieve RDP and they are all gradient perturbation-based ones, we plot the performance over iterations, and use RDP for composition of mechanisms. We set the hyperparameters of NOISYBTLS as $\alpha = 0.001$ and $\beta = 0.8$. The reason is that for over-parameterized models we empirically observed that setting α to small helps fasten training. For DP-ADAM, we use the default parameter settings. For DP-SGD, after tuning, we set $\eta = 0.2$ for MNIST and FMNIST models, and $\eta = 0.1$ for Cifar-10 models.



(a) Adult (b) IPUMS-BR (c) IPUMS-US (d) Bank

Fig. 3: Logistic regression result by ϵ (Top: Classification accuracy; Bottom: Objective value)



(a) Adult (b) IPUMS-BR (c) IPUMS-US (d) Bank

Fig. 4: SVM result by ϵ (Top: Classification accuracy; Bottom: Objective value)

E. Effect of Hyperparameters

We evaluate the effect of 6 important hyperparameters in our algorithms: α, β used for backtracking line search, noise distribution (Laplace or Gaussin), sampling ratio q, budget allocation strategy, and clipping threshold decay rate ζ. Figure 1 shows the effects of varying hyperparameters on the performance of logistic regression model. For this experiment, Adult dataset was used. As shown in Figure 1a and 1b, the algorithm shows relatively robust performance against the choice of α and β when $\epsilon = 0.2$. When $\epsilon = 0.05$, the algorithm achieved its best performance at $\alpha = 0.5$ and $\beta = 0.8$. This is because choosing smaller step sizes allows the algorithm to control the variance due to noise. Larger values of α will encourage the algorithm to choose small step size by setting the expected reduction in the objective high. Small β would give the algorithm a large jump of candidate step size each time, therefore generally $\beta = 0.8$ is a suitable choice. The proposed algorithm achieved slightly higher accuracy when the noise for backtracking line search was drawn from the Laplace distribution. When $\epsilon = 0.2$, the algorithm is robust to the choice of ampling ratio q, but q cannot be too small at high privacy level, since gradients calculated on smaller batches have higher variance. Although the performance is similar for stochastic and full gradient descent, as Figure 2 shows, subsampling can greatly reduce the number of objective and gradient evaluations. For budget allocation mechanisms for SGD, "no" means never increase budget, "must" means always increase budget regardless of angle measurement, and "adap" means adaptively increase budget based on angle measurement. We can see the angle measurement is indeed beneficial. It is hard to determine the effect of adaptively decreasing clipping threshold, since the results show that it helps improve the performance when $\epsilon = 0.05$, but it does not when $\epsilon = 0.2$. Therefore in the next section we plot the performance of BLSGD with and without adaptive clipping.

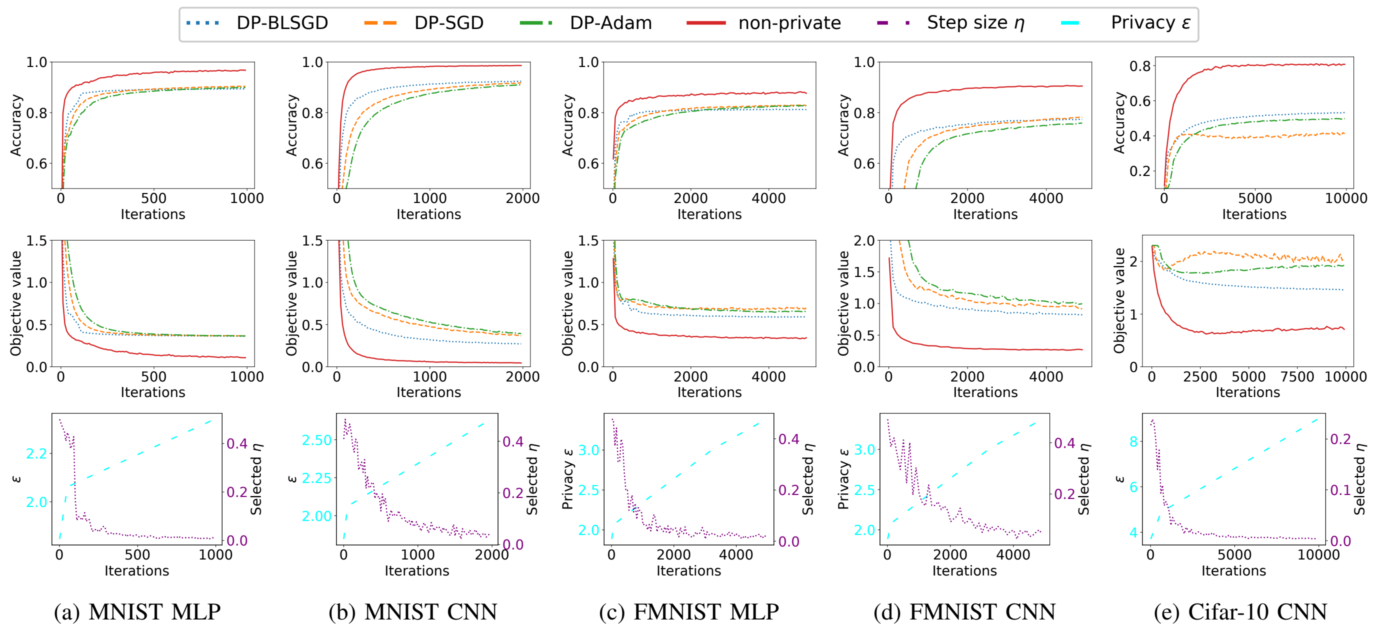


Fig. 5: Neural network model results (Top: Classification Accuracy; Middle: Objective Value; Bottom: Privacy ϵ and selected step size)

F. Performance of Convex Optimization

Figure 3 and Figure 4 plots the testing data accuracy (top) and objective values (bottom) of the algorithms against the privacy parameter ϵ, for logistic regression and SVM, respectively. For the algorithms we proposed, we show results using Laplace version of NOISYBTLS, since it slightly outperforms the one using Gaussian version. DP-BLSGD, DP-BLSGD-AC, and its full-batch version (DP-BLGD) outperform the baseline algorithms in most cases. They outperform the DP-AGD algorithm, which shows that the NOISYBTLS based technique performs better than the NOISYMIN based step size selection. Since the DP-BLGD applies the same budget increasing mechanism as DP-AGD, and both algorithms use full gradient descent, it shows that the improvement of DP-BLGD over DP-AGD is a result of Armijo line search technique. Our algorithms also outperform the state-of-the-art output perturbation algorithm, OUTPERT-RSGD, on 3 out of 4 datasets. OBJPERT and DP-SGD show low performance when ϵ is small. This indicates that step size selection and adaptive budget control are useful tools to achieve a high privacy level. The Bank dataset is an exception, which OUTPERT-RSGD outperforms our algorithms. But this dataset has very small training range since the MAJORITY and NON-PRIVATE baselines are very close, which might affect the performance of gradient perturbation based algorithms. When clipping threshold adaptation is applied (DP-BLSGD-AC), the performance can be slightly increased in some datasets.

G. Performance of Neural Network models

For MLP, we have one hidden layer with 1000 units for MNIST, and 2 hidden layers with 256 units each for FMNIST; for CNN on MNIST and FMNIST, we stack a convolutional layer with 6 output channels, a max pooling layer, another convolutional layer with 16 output channels, another max pooling layer, and 2 fully connected layer with width 256 and 128, respectively. For CNN on Cifar-10, we stack a convolutional layer with 32 out channels, another with 64 out channels, a max-pooling layer, 2 convolutional layers with 128 out channels, a max-pooling layer, 2 convolutional layers with 256 out channels, a max-pooling layer, and 3 fully connected layers with size 4096, 1024, 512, respectively. In order for our DP-BLSGD to accumulate privacy parameter ϵ at the same speed with DP-SGD and DP-ADAM, we set the the NOISYBTLS to return $\beta^{\texttt{max\_it}}\eta_0$ instead of 0 if it evaluates over all the `max_it` candidates. (Thus, we let our algorithm DP-BLSGD just perform step-size selection, without budget increasing.) We also set the same per-iteration budget ρ_{iter} for three algorithms: for DP-SGD and DP-ADAM, we use ρ_{iter} to determine noisy scale σ^2; for DP-BLSGD, we use 10% of ρ_{iter} for NOISYBTLS (Gaussian version), and 90% for gradient perturbation. Therefore, each iteration is $(\alpha, \alpha\rho_{iter})$-RDP for all three algorithms, and it would be a fair comparison of performances against iterations.

The classification performance for neural network models are shown Figure 5. The bottom row shows the step size selected by DP-BLSGD. As an expected behavior, it is decreasing during training. DP-BLSGD outperforms DP-SGD and DP-ADAM in these aspects: For MLP networks, it can reach to a high testing accuracy in less iterations, and converge to an accuracy similar as other methods. For CNN networks on MNIST and FMNIST, it converges much faster than DP-SGD and DP-ADAM, and result in lower objective values. On Cifar-10 dataset, although the gap between non-private and private algorithms is larger, DP-BLSGD still achieves better performance in less iterations compare to the two private baselines, especially in objective value. Note that, since our DP-BLSGD did not perform budget increasing for neural network models, it uses the same per-iteration budget across the training as DP-SGD and DP-ADAM, so our method may still face too noisy gradients in later stage, preventing it from converging to a higher accuracy. Since neural network models needs much more iterations to train, the privacy leak would accumulates too fast if we keep increasing per-iteration budget. However, our results show that step-size selection through line search alone can help increase the performance to a certain level, and accelerate the convergence as well.

VI. CONCLUSIONS

We present a Rényi differentially private SGD algorithm, in which step sizes are adaptively chosen using the Armijo condition. To improve the reliability of chosen step sizes, we also introduce strategies for adaptive privacy budget allocation. Our empirical evaluations on both convex and nonconvex problems demonstrate that classical line search can help automatically set the step size and improve the utility. We also introduce practical techniques for improving the runtime adaptivity of private optimization algorithms, which allows the algorithm to accelerate by making quick progresses.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. 1943046.

REFERENCES

[1] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of cryptography conference*. Springer, 2006, pp. 265–284.

[2] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[3] I. Mironov, "Rényi differential privacy," in *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. IEEE, 2017, pp. 263–275.

[4] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2016, pp. 308–318.

[5] L. Yu, L. Liu, C. Pu, M. E. Gursoy, and S. Truex, "Differentially private model publishing for deep learning," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 332–349.

[6] J. Lou and Y.-m. Cheung, "An uplink communication-efficient approach to featurewise distributed sparse optimization with differential privacy," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[7] R. Bassily, A. Smith, and A. Thakurta, "Private empirical risk minimization: Efficient algorithms and tight error bounds," in *Foundations of Computer Science (FOCS), 2014 IEEE 55th Annual Symposium on*. IEEE, 2014, pp. 464–473.

[8] J. Lee and D. Kifer, "Concentrated differentially private gradient descent with adaptive per-iteration privacy budget," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2018, pp. 1656–1665.

[9] L. Armijo, "Minimization of functions having lipschitz continuous first partial derivatives," *Pacific Journal of mathematics*, vol. 16, no. 1, pp. 1–3, 1966.

[10] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[11] S. Vaswani, A. Mishkin, I. Laradji, M. Schmidt, G. Gidel, and S. Lacoste-Julien, "Painless stochastic gradient: Interpolation, line-search, and convergence rates," in *Advances in Neural Information Processing Systems*, 2019, pp. 3727–3740.

[12] M. Lyu, D. Su, and N. Li, "Understanding the sparse vector technique for differential privacy," *arXiv preprint arXiv:1603.01699*, 2016.

[13] Y. Zhu and Y.-X. Wang, "Poission subsampled rényi differential privacy," in *International Conference on Machine Learning*, 2019, pp. 7634–7642.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[15] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Advances in neural information processing systems*, 2013, pp. 315–323.

[16] M. Sordello and W. Su, "Robust learning rate selection for stochastic optimization via splitting diagnostic," *arXiv preprint arXiv:1910.08597*, 2019.

[17] L. Berrada, A. Zisserman, and M. P. Kumar, "Training neural networks for and by interpolation," in *International Conference on Machine Learning*, 2020, pp. 4758–4768.

[18] Z. Ding, Y. Wang, G. Wang, D. Zhang, and D. Kifer, "Detecting violations of differential privacy," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 475–489.

[19] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate, "Differentially private empirical risk minimization," *Journal of Machine Learning Research*, vol. 12, no. Mar, pp. 1069–1109, 2011.

[20] J. Zhang, K. Zheng, W. Mou, and L. Wang, "Efficient private erm for smooth objectives," *arXiv preprint arXiv:1703.09947*, 2017.

[21] X. Wu, F. Li, A. Kumar, K. Chaudhuri, S. Jha, and J. Naughton, "Bolt-on differential privacy for scalable stochastic gradient descent-based analytics," in *Proceedings of the 2017 ACM International Conference on Management of Data*. ACM, 2017, pp. 1307–1322.

[22] C. Chen, J. Lee, and D. Kifer, "Renyi differentially private erm for smooth objectives," in *The 22nd International Conference on Artificial Intelligence and Statistics*, 2019, pp. 2037–2046.

[23] D. Kifer, A. Smith, and A. Thakurta, "Private convex empirical risk minimization and high-dimensional regression," in *Conference on Learning Theory*, 2012, pp. 25–1.

[24] A. Koskela and A. Honkela, "Learning rate adaptation for differentially private stochastic gradient descent," *arXiv preprint arXiv:1809.03832*, 2018.

[25] D. Wang, M. Ye, and J. Xu, "Differentially private empirical risk minimization revisited: Faster and more general," in *Advances in Neural Information Processing Systems*, 2017, pp. 2722–2731.

[26] R. Bassily, V. Feldman, K. Talwar, and A. G. Thakurta, "Private stochastic convex optimization with optimal rates," in *Advances in Neural Information Processing Systems*, 2019, pp. 11 282–11 291.

[27] V. Feldman, T. Koren, and K. Talwar, "Private stochastic convex optimization: optimal rates in linear time," in *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, 2020, pp. 439–449.

[28] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor, "Our data, ourselves: Privacy via distributed noise generation," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2006, pp. 486–503.

[29] M. Bun and T. Steinke, "Concentrated differential privacy: Simplifications, extensions, and lower bounds," in *Theory of Cryptography Conference*. Springer, 2016, pp. 635–658.

[30] C. Chen and J. Lee, "Stochastic adaptive line search for differentially private optimization," *arXiv preprint arXiv:2008.07978*, 2020.

[31] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM transactions on intelligent systems and technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[32] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[33] S. Ruggles, K. Genadek, R. Goeken, J. Grover, and M. Sobek, "Integrated public use microdata series: Version 6.0 [dataset]," *Minneapolis: University of Minnesota*, vol. 23, p. 56, 2015.

[34] J. Zhang, X. Xiao, Y. Yang, Z. Zhang, and M. Winslett, "Privgene: differentially private model fitting using genetic algorithms," in *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, 2013, pp. 665–676.