Rényi Differentially Private ERM for Smooth Objectives

Chen Chen
University of Georgia

Jaewoo Lee
University of Georgia

Daniel Kifer
Penn State University

Abstract

In this paper, we present a Renyi Differentially Private stochastic gradient descent (SGD) algorithm for convex empirical risk minimization. The algorithm uses output perturbation and leverages randomness inside SGD, which creates a "randomized sensitivity", in order to reduce the amount of noise that is added.

One of the benefits of output perturbation is that we can incorporate a periodic averaging step that serves to further reduce sensitivity while improving accuracy (reducing the well-known oscillating behavior of SGD near the optimum).

Renyi Differential Privacy can be used to provide (ϵ, δ)-differential privacy guarantees and hence provide a comparison with prior work. An empirical evaluation demonstrates that the proposed method outperforms prior methods on differentially private ERM.

1 INTRODUCTION

Research on machine learning with differential privacy is nearing the point of making it practical to build accurate models that strongly protect the privacy of individuals in the training set [1, 9, 20, 14, 23, 3, 31, 32, 30, 15]. The introduction of Rényi Differential Privacy (RDP) [18] offers the promise of additional significant improvements in accuracy in exchange for a very modest relaxation in privacy guarantees.[1]

In this paper we propose a novel Rényi differentially private stochastic gradient descent algorithm for convex empirical risk minimization that outperforms prior work. This method uses output perturbation (with numerical computation of sensitivity) along with the randomness of batches inside stochastic gradient descent to protect privacy.

Let $D = \{d_1, \ldots, d_n\}$ be a dataset of n i.i.d. samples drawn from an underlying distribution $\mathcal{P}$. The training of many models, including logistic regression and support vector machines, can be reduced to solving an equation such as:

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

where f measures the "loss" of the model with parameter vector $\mathbf{w}$ on a data point. This procedure is known as Empirical Risk Minimization (ERM). In many cases, f is convex and twice differentiable with respect to $\mathbf{w}$ [9, 32].

With privacy, the goal is to return a noisy version of the optimal parameter vector $\mathbf{w}^*$, where the noise distribution must satisfy (Rényi) differential privacy constraints [18] that ensure the confidentiality of any record in the training data.

Our solution has the following key features. First, it uses a variation of mini-batch stochastic gradient descent (SGD). This choice provides a computational speedup over full-batch methods [32, 15] and also has implications for privacy – for a large enough dataset, SGD can train a model with many fewer passes over the data than full-batch gradient descent and so each data point has a smaller influence on the optimal parameter vector.

Second, we use output perturbation (noise is added to the optimal parameter vector [31, 32]) instead of input perturbation (e.g., where additional noise is added to each mini-batch gradient [1, 27, 15, 3]). This choice allows us to perform periodic averaging of the intermediate parameter vectors encountered in SGD. In convex ERM problems, averaging helps improve convergence (intuitively by reducing the oscillations around the optimum) [2]. Crucially, we show that averaging also helps us use less noise to protect privacy. Furthermore, when the input data are initially permuted, the

[1] RDP is weaker than ϵ-differential privacy but stronger than (ϵ, δ)-differential privacy.

Proceedings of the 22nd International Conference on Artificial Intelligence and Statistics (AISTATS) 2019, Naha, Okinawa, Japan. PMLR: Volume 89.

SGD updates are a random combination of contraction mappings and an expansion mapping. This random behavior allows us to further reduce the amount of noise that is necessary to protect privacy.

By working in the framework of Rényi differential privacy, we are providing a stronger privacy guarantee than (ϵ, δ)-differential privacy yet can still compare against algorithms for (ϵ, δ)-differential privacy using RDP to DP conversion theorems [18]. Our contributions are summarized as follows:

- We propose an algorithm, based on SGD and output perturbation, for solving convex empirical risk minimization problems under Rényi differential privacy.
- The inherent randomness in SGD means that the influence of one point on a parameter update is a random variable. We take advantage of this randomness to reduce the amount of noise that is necessary for protecting privacy with output perturbation.
- Typically, SGD makes slow progress as it approaches the optimum. To counter this effect, we add periodic averaging steps to reduce the oscillations of SGD and to further reduce the amount of noise that is necessary for output perturbation.
- We perform extensive experiments on real datasets against other recently proposed algorithms. We empirically show the effectiveness of the proposed algorithm for a wide range of privacy parameter values.

The rest of this paper is organized as follows. In Section 2, we review related work. In Section 3, we provide background on differential privacy and convex ERM. We present our algorithm in Section 4 and experiments in Section 5.

2 RELATED WORK

The problem of differentially private optimization has received a great deal of attention from the machine learning community, especially in the context of ERM [9, 27, 14, 23, 3, 29, 1, 31, 32, 30, 20, 15]. There has been significant advances in both theory and algorithm development.

Bassily et al. [3] derived the minimax lower bound for excess empirical risk and proved that their private variant of SGD matches the lower bound. They used a *gradient perturbation* approach in which the algorithm optimizes the objective function only using gradients perturbed by random noise. However, their private SGD algorithm requires prohibitively large, $\mathcal{O}(n^2)$, number of iterations. In [30], Wang et al. also take the gradient perturbation approach to privatize an accelerated version of SGD, called SVRG [13]. They showed that their algorithm also matches the lower bound with improved gradient complexity. Talwar et al. [28] further refined the bound for a specific task by adding more restrictive constraints. Abadi et al. [1] showed that private SGD can be successfully used for deep learning using techniques that would later be known as Rényi differential privacy [18]. Feldman et al. [12] analyzed the privacy amplification effect of noisy contractive mappings in Stochastic Gradient Langevin Dynamics (SGLD), which perform SGD updates using Gaussian noise perturbed gradients. A technique called PATE [20] dramatically improved the private training of deep networks but requires a large public dataset. Using concentrated differential privacy [6], Lee and Kifer showed that private SGD algorithms based on gradient perturbation can be improved in practice with adaptive step sizes and careful allocation of privacy budgets between iterations [15].

The work most similar to ours is that of Wu et al. [31]. They proposed an *output perturbation* [11, 9, 32] based SGD algorithm in which noise is calibrated according to the stability analysis of gradient descent. The use of randomly permuted mini-batches in their algorithm is for utility only, whereas we analyze how permutations affect privacy cost (i.e., how they compose with output perturbation and require less additive noise). Zhang et al. [32] also used the output perturbation to privatize full batch gradient descent. Output perturbation approaches must derive bounds on the sensitivity of the underlying non-private algorithm and the added noise scales with this bound. The calculation of sensitivity in these prior works often uses loose inequalities, resulting in higher noise levels than are necessary. We use numerical computation of sensitivity to substantially decrease the amount of noise added to protect privacy.

Chaudhuri et al. [9] proposed a general framework called *objective perturbation* as an alternative to gradient and output perturbation – they produce a random objective function by adding a linear noise term to the original objective. To ensure privacy, the resulting problem must be fully solved to optimality using an off-the-shelf optimizer. Kifer et al. [14] further improved the utility of the objective perturbation method by using (ϵ, δ)-differential privacy. Again, the optimization must be solved to optimality (whereas in practice, the optimizers stop with an approximation whose error is larger than machine precision).

3 PRELIMINARIES

A record $d_i \in \mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ is a tuple $(\mathbf{x}_i, y_i)$, where $x_i \in \mathcal{X}$ is a feature vector and $y_i \in \mathcal{Y}$ is a target. A dataset $D \in \mathcal{Z}^n$ is a set of n records $\{d_1, \ldots, d_n\}$.

Two datasets D and D' are *neighbors*, denoted $D \sim D'$, if D' can be obtained from D by replacing one record with another one from $\mathcal{Z}$.

3.1 Differential Privacy

Differential privacy is the current gold-standard in protecting the privacy of an input dataset. It is formally defined as follows.

Definition 1 ((ϵ, δ)-DP [11, 10]). *Given privacy parameters $\epsilon \geq 0$, $\delta \geq 0$, a randomized mechanism (algorithm) $\mathcal{M}$ satisfies (ϵ, δ)-differential privacy if for every event $S \subseteq \text{range}(\mathcal{M})$ and for all pairs of neighboring databases $D \sim D'$,*

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

The case where $\delta = 0$ is referred to as *pure* differential privacy and the case where $\delta > 0$ is referred to as *approximate* differential privacy.

Pure differential privacy limits the ability of an attacker to make inferences about the specific record of any individual. Approximate differential privacy provides the same guarantees with probability $1 - \delta$, and allows failure of privacy (e.g. release of the entire raw data) to occur with probability δ. This event is known informally as the "all-bets-are-off" scenario.

A recent relaxation of differential privacy, known as Rényi differential privacy [18] provides weaker protections than pure differential privacy and stronger protections than approximate differential privacy (notably avoiding the "all-bets-are-off" scenario).

3.2 Rényi Differential Privacy

While pure differential privacy requires that constraint $e^{-\epsilon} \leq \frac{\Pr[\mathcal{M}(D) \in S]}{\Pr[\mathcal{M}(D') \in S]} \leq e^{\epsilon}$ always holds, Rényi differential privacy (RDP) allows this ratio to be a random variable and constrains it using the Rényi divergence:

Definition 2 (Rényi Divergence). *Let P_1 and P_2 be probability distributions over a set Ω and let $\alpha \in (1, \infty)$. Rényi α-divergence D_α is defined as:* $\mathrm{D}_\alpha(P_1 \parallel P_2) = \frac{1}{\alpha - 1} \log(\mathbb{E}_{x \sim P_2}\left[P_1(x)^\alpha P_2(x)^{-\alpha}\right])\,.$

Rényi differential privacy requires two parameters: a moment α and a parameter ϵ that bounds the moment.

Definition 3 ((α, ϵ)-RDP [18]). *Given a privacy parameter $\epsilon \geq 0$ and an $\alpha \in (1, \infty)$, a randomized mechanism $\mathcal{M}$ satisfies (α, ϵ)-Rényi differential privacy (RDP) if for all D_1 and D_2 that differ on the value of one record,* $\mathrm{D}_\alpha(\mathcal{M}(D_1) \parallel \mathcal{M}(D_2)) \leq \epsilon\,.$

Note that when $\alpha = \infty$, Rényi divergence becomes max divergence and (α, ϵ)-RDP becomes pure $(\epsilon, 0)$-differential privacy. While the semantics of RDP are still an area of research, it is currently being used in practice as follows [18].

First, one chooses an ϵ' and a very small $\delta' > 0$ with the intention of providing strictly stronger privacy protections than (ϵ', δ')-differential privacy. Then one uses the following conversion result to find corresponding α and ϵ parameters for (α, ϵ)-RDP.

Proposition 1 (Conversion to (ϵ, δ)-DP [18]). *If $\mathcal{M}$ satisfies (α, ϵ)-RDP, it satisfies (ϵ', δ')-differential privacy when $\epsilon' \geq \epsilon + \frac{\log 1/\delta}{\alpha - 1}$ and $\delta' \geq \delta$.*

The workhorse method for achieving RDP is the Gaussian mechanism, which relies on the L_2 sensitivity.

Definition 4 (L_2 sensitivity). *Let $q : \mathcal{Z}^n \to \mathbb{R}^k$ be a vector-valued function over datasets. The L_2 sensitivity of q, denoted by $\Delta_2(q)$ is defined as*

$$\Delta_2(q) = \max_{D \sim D'} \|q(D) - q(D')\|_2\,,$$

where the max is over all pairs neighboring datasets.

The Gaussian mechanism for RDP takes a vector-valued function q (i.e. a query) over datasets and adds appropriately scaled Gaussian noise.

Lemma 1 (Gaussian Mechanism [18]). *Let $q : \mathcal{Z}^n \to \mathbb{R}^k$ be a vector-valued function over datasets. Let $\mathcal{M}$ be a mechanism that releases $\mathcal{N}(q(D), \sigma^2 \mathbf{I}_k)$. Then for any pair of neighboring datasets D and D' and any $\alpha \in (1, \infty)$:* $\mathrm{D}_\alpha(\mathcal{M}(D) \parallel \mathcal{M}(D')) \leq \alpha \Delta_2^2(q)/(2\sigma^2)$. *In particular, if $\sigma^2 = \alpha \Delta_2^2(q)/(2\epsilon)$, then $\mathcal{M}$ satisfies (α, ϵ)-RDP.*

3.3 Convex Empirical Risk Minimization

Recall that a record d_i consists of a feature vector $\mathbf{x}_i$ and a target y_i. In the empirical risk minimization framework, fitting many convex models can be rephrased as optimizing the following equation.

$$\mathbf{w}^* = \arg\min_{\mathbf{w} \in \Theta} F(\mathbf{w}, D) := \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{w}, \mathbf{x}_i, y_i)\,, \quad (1)$$

where Θ is a convex parameter space. F is called the *objective function* and f is known as the *loss function*. For example a regularized of logistic regression model can be fit to data by solving $\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} (\log(1 + e^{-y_i(\mathbf{w} \cdot \mathbf{x}_i)}) + \lambda ||\mathbf{w}||_2^2)$, in which case the function f equals $\log(1 + e^{-y_i(\mathbf{w} \cdot \mathbf{x}_i)}) + \lambda ||\mathbf{w}||_2^2$. Following prior work on private convex ERM (e.g., [9, 32, 15]), we impose the following conditions on f.

1. **Differentiability.** Loss function f is continuously differentiable with respect to $\mathbf{w}$ and has a continuous Hessian except on a set of Lebesgue measure 0.

2. **L-smooth.** There exists an $L > 0$ such that for any $\mathbf{w}, \mathbf{w}' \in \Theta$ and for all $d_i = (\mathbf{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y}$, $\|\nabla f(\mathbf{w}, d_i) - \nabla f(\mathbf{w}', d_i)\|_2 \leq L\|\mathbf{w} - \mathbf{w}'\|_2$.

3. **Strong convexity.** There exists a $\mu > 0$ such that $f(\mathbf{w}, d_i)$ is μ-strongly convex in the first argument, i.e., for any $\mathbf{w}, \mathbf{w}' \in \Theta$ and any d_i,

$$f(\mathbf{w}', d_i) \geq f(\mathbf{w}, d_i) + \nabla f(\mathbf{w}, d_i)^\intercal(\mathbf{w}' - \mathbf{w}) + \frac{\mu}{2}\|\mathbf{w} - \mathbf{w}'\|_2^2 .$$

4. **Bounded Gradient.** There exists an $R > 0$ that satisfies $\|\nabla f(\mathbf{w}, d)\|_2 \leq R$ for $\forall \mathbf{w} \in \Theta$, $d \in \mathcal{Z}$.

Conditions 1, 2, and 3 imply bounds on the eigenvalues of the Hessian [5], i.e., for $\forall \mathbf{w} \in \Theta$ and $d_i \in \mathcal{Z}$,

$$\mu \mathbf{I} \preceq \nabla^2 f(\mathbf{w}, d_i) \preceq L\mathbf{I} . \tag{2}$$

except on a set of Lebesgue measure 0. Typically the bounded gradient condition is satisfied by ensuring that the feature vectors $\mathbf{x}_i$ lie inside a ball of some radius R'.

3.4 Gradient Descent Operator

Empirical risk minimization, especially in large-scale problems, is often solved with stochastic gradient descent (SGD). At each iteration, SGD chooses a random subset of training data B, called a mini-batch. It uses the gradient on the mini-batch to approximate the full-data gradient of F (Equation (1)) then updates the parameter vector $\mathbf{w}$ as follows:

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \frac{\eta_k}{|B|}\sum_{i \in B} \nabla f(\mathbf{w}_k, d_i) , \tag{3}$$

where $\eta_k > 0$ is the step size (which slowly decreases with iteration k). We use the notation ∇f_B to denote the mini-batch gradient for B, i.e., $\nabla f_B(\cdot) = \frac{1}{|B|}\sum_{i \in B} \nabla f(\cdot, d_i)$. The SGD update in Equation (3) can be viewed as an operator on the parameter space Θ defined as

$$\mathcal{T}_B(\cdot) = \mathrm{Id}(\cdot) - \eta \nabla f_B(\cdot) , \tag{4}$$

where Id is the identity operator. When f_B is smooth and strongly convex, $\mathcal{T}_B$ forms a contraction mapping [26], which means that applying $\mathcal{T}_B$ to any two parameter vectors $\mathbf{w}, \mathbf{w}' \in \Theta$ (using the same mini-batch B) shrinks the distance between $\mathbf{w}$ and $\mathbf{w}'$ by a constant factor. Mathematically, for any $\mathbf{w}, \mathbf{w}' \in \Theta$, we have $\|\mathcal{T}_B(\mathbf{w}) - \mathcal{T}_B(\mathbf{w}')\| \leq \rho\|\mathbf{w} - \mathbf{w}'\|$, where $\rho < 1$ is a contraction coefficient. Note that in (Rényi) differential privacy, a pair of neighboring databases D and D' might produce slightly different minibatches B and B', which will require us to compare $\mathcal{T}_B(\mathbf{w})$ with $\mathcal{T}_{B'}(\mathbf{w}')$.

Algorithm 1: Vanilla Algorithm (NSGD)

Input: data D, number of epochs T, initial step size η_0, noise scale parameter σ^2

```
Split D into mini-batches B_0, ..., B_{m-1}
Initialize w_0, t = 0
for s = 1 to T do
    η = η_0 / s
    for j = 0 to m − 1 do
        t = t + 1
        g_t = (1/|B_j|) Σ_{i∈B_j} ∇f(w_{t−1}, d_i)
        w_t = w_{t−1} − η g_t
sample Y ~ N(0, σ^2)
return w_priv = w_t + Y
```

4 ALGORITHM

4.1 Vanilla Mini-batch SGD

We begin the description of our private SGD algorithm starting with a plain version shown in Algorithm 1 which, modulo a few small details, is equivalent to the bolt-on method of [31]. We will then add in optimizations that improve accuracy, resulting in our proposed method, shown in Algorithm 3.

Algorithm 1 is a non-randomized SGD method. We will later add randomness into the mini-batch construction by permutating record order. Please note that the randomness we use is different from competing approaches for RDP. For instance Abadi et al. [1] base their privacy analysis on the assumption that each mini-batch is an independent random sample of the data. Such random sampling is a slow operation. Hence practical implementations do not use this approach. Instead they permute the data and form disjoint mini-batches by taking consecutive records from the permuted data (the resulting mini-batches are therefore not independent since they cannot overlap). Thus we base our privacy analysis on the assumption that the data have been permuted in order to create disjoint mini-batches.

Contraction Mapping. Let us first analyze Algorithm 1. It is similar to standard (mini-batched) SGD in that it updates parameters by iteratively applying the SGD operator $\mathcal{T}_{B_i}$ to the current iterate, i.e., $\mathbf{w}_{t+1} = \mathcal{T}_{B_i}(\mathbf{w}_t)$. One important difference is that, while standard SGD constructs the i^{th} batch B_i by randomly selecting the records in the batch, NSGD builds the B_i by splitting D into m mini-batches, denoted by $B_0, \ldots, B_{m-1}$, and then accesses them sequentially in a cyclic order. This type of algorithm is known as the incremental gradient method [4] and has

been widely used to solve large-scale problems.

This sequential access has important implications for our sensitivity analysis. Let D and D' be neighboring datasets. If D is the input to the algorithm, the algorithm will split it into batches $B_0, \ldots, B_{m-1}$ and will iteratively produce a sequence of parameter estimates $\mathbf{w}_0$, $\mathbf{w}_1, \mathbf{w}_2, \ldots$, where $\mathbf{w}_0$ is some initial starting value and the updates are performed using the rule $\mathbf{w}_i = \mathcal{T}_{B_{i-1 \mod m}}(\mathbf{w}_{i-1})$. On the other hand, if D' is the input, the algorithm will split it into batches $B'_0, \ldots, B'_{m-1}$ and produce a sequence of parameter estimates $\mathbf{w}'_0$, $\mathbf{w}'_1, \mathbf{w}'_2, \ldots$, where $\mathbf{w}'_0 = \mathbf{w}_0$ is the default starting value and the updates use the rule $\mathbf{w}'_i = \mathcal{T}_{B'_{i-1 \mod m}}(\mathbf{w}'_{i-1})$.

Following [31], note that since D and D' differ on the value of one record, there will be a j such that mini-batches B_j and B'_j differ on the value of one record. For all $i \neq j$, we have $B_i = B'_i$. This means that as long as $(i \mod m) \neq j$, $\mathbf{w}_{i+1}$ and $\mathbf{w}'_{i+1}$ are obtained using the same operator $\mathcal{T}_{B_i}$ (i.e., $\mathbf{w}_{i+1} = \mathcal{T}_{B_i}(\mathbf{w}_i)$ and $\mathbf{w}'_{i+1} = \mathcal{T}_{B_i}(\mathbf{w}'_i)$). Now, when the loss function f satisfies condition 1, 2, and 3 in Section 3.3, operator $\mathcal{T}_{B_i}$ is a contraction, so that the parameter estimates move closer to each other. That is, let $\Delta_{i+1} = ||\mathbf{w}_{i+1} - \mathbf{w}'_{i+1}||$ and $\Delta_i = ||\mathbf{w}_i - \mathbf{w}'_i||$. The contraction property means that $\Delta_{i+1} < \Delta_i$.

However, when $(i \mod m) = j$, the parameter updates use different operators: $\mathcal{T}_{B_j}$ and $\mathcal{T}_{B'_j}$ so $\mathbf{w}_{i+1} = \mathcal{T}_{B_j}(\mathbf{w}_i)$ and $\mathbf{w}'_{i+1} = \mathcal{T}_{B'_j}(\mathbf{w}'_i)$. In this situation, the operators force the parameter estimates further apart. This discussion is summarized in the following lemma, whose proof appears in the supplementary materials.

Lemma 2. *Let B and B' be mini-batches that differ on the value of one record. Define the operator $\mathcal{T}_B(\cdot) = \mathrm{Id}(\cdot) - \eta \nabla f_B(\cdot)$ (and similarly for B'). Let $\mathbf{w}$ and $\mathbf{w}'$ be any two vectors in Θ. Let $\rho = \max\{|1-\eta\mu|, |1-\eta L|\}$ (where μ is the strong convexity parameter and L is the smoothness parameter). Then:*

$$||\mathcal{T}_B(\mathbf{w}) - \mathcal{T}_B(\mathbf{w}')|| \leq \rho ||\mathbf{w} - \mathbf{w}'|| \quad \textit{(same batch } B\textit{)}$$

$$||\mathcal{T}_B(\mathbf{w}) - \mathcal{T}_{B'}(\mathbf{w}')|| \leq \rho ||\mathbf{w} - \mathbf{w}'|| + \frac{2\eta R}{|B|}$$

where the first equation shows the effect of using the same operator $\mathcal{T}_B$ and the second equation shows the effect of using $\mathcal{T}_B$ to update $\mathbf{w}$ and a different operator $\mathcal{T}_{B'}$ to update $\mathbf{w}'$.

If we use a fixed step size η, instead of diminishing step size, in Algorithm 1 (i.e., removing Line 4) and set $m = 1$, then the naive algorithm coincides with private full-batch gradient descent (GD) proposed in [32]. The following Proposition shows how the sensitivity of the resulting algorithm can be computed using the recurrence relations from Lemma 2.

Proposition 2. *If we run Algorithm 1 for arbitrary number of epochs with a fixed step size η, its sensitivity Δ satisfies*

$$\Delta \leq \frac{2\eta R}{|B|(1-\rho^m)},$$

where $\rho = \max\{|1-\eta\mu|, |1-\eta L|\}$. In particular, when $m = 1$ and $\eta = \frac{2}{L+\mu}$, $\Delta \leq \frac{2R}{n\mu}$.

Note that the above sensitivity is tighter than the one computed in [32], which is $\frac{5R(L+\mu)}{n\mu L}$.

With a fixed number of epochs and a step size η_t that depends on the epoch number t, we can use Lemma 2 as follows. If D and D' only differ in the last batch, then their initial parameters are the same. After the first epoch, they differ by at most $\frac{2\eta_1 R}{|B|}$, after the second epoch, they differ by at most $\rho^m \frac{2\eta_1 R}{|B|} + \frac{2\eta_2 R}{|B|}$ and so on. It is easy to see that this is the worst case (i.e. the upper bound is maximized when D and D' differ on the last of the m mini-batches) and hence the resulting upper bound (which can be easily computed numerically) is the sensitivity. After the last epoch, noise with variance $\alpha\Delta^2/(2\epsilon)$ can be added to the final parameter vector to achieve (α, ϵ)-RDP.

4.2 Randomized Permuted Batches.

If we permute the data *once* before starting the SGD updates, then the mini-batches become randomized and are subsequently processed in cyclic order. This allows us to add less noise because, with randomized batches, it is no longer possible to create "bad" neighbors that always differ in the last batch – now the batch they differ in will be $0, 1, \ldots, m-1$ with equal probability.

Let D and D' be *any* two databases that differ on one record. Consider what happens when the algorithm is run with input D (world 1) and with input D' (world 2). Given the same input randomness, when the data are split into batches, for any j, with probability 1/m world 1 and world 2 will differ in the j^{th} batch only. So, starting with the same weight initial vector $\mathbf{w}_0$, the first $j-1$ weight updates in an epoch will be contractions, the j^{th} update will be an expansion, and the remaining $m-j$ will be contractions. Hence, after the first epoch, the difference in weights between the two worlds is at most $\Delta^{(1)} = \rho^{m-j} 2\eta_1 R/|B|$ and after the second epoch, the difference is bounded by $\Delta^{(2)} = \rho^{m-j}(\rho^j \Delta^{(1)} + 2\eta_2 R/|B|)$, etc. At the end the algorithm adds Gaussian noise to the vector.

This scenario can be abstracted as follows. There exists a set of mechanisms[2] $\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_m$ with as-

[2]For example, M_i can apply a function f_i on the input data and then add $\mathcal{N}(0, \sigma^2 I)$ noise to the answer.

Algorithm 2: Sensitivity calculation

Input: total number of epochs T, initial step size η_0, averaging interval τ, number of mini-batches m, mini-batch size ν

Function ComputeSensitivity(T, m, η_0, τ)
  for $j = 1$ **to** m **do**
    $\mathbf{\Delta}[j] = 0$ // initialization
  for $s = 1$ **to** T **do**
    $\eta = \eta_0/s$
    $\rho = \max\{|1-\eta\mu|, |1-\eta L|\}$
    for $j = 1$ **to** m **do**
      $\mathbf{\Delta}[:] = \rho\mathbf{\Delta}[:]$ // contraction
      $\mathbf{\Delta}[j] = \mathbf{\Delta}[j] + \frac{2\eta R}{\nu}$ // expansion
  return Δ

Function ComputeH(α, $\mathbf{\Delta}$, $\mathbf{q}$)
  $H_\alpha = 0$
  for $j = 1$ **to** m **do**
    $H_\alpha = H_\alpha + \mathbf{q}[j] \cdot \exp(\alpha(\alpha-1)\mathbf{\Delta}[j]^2/(2\sigma^2))$
  return H_α

sociated L_2 sensitivities $\Delta_1, \ldots, \Delta_m$ and an associated probability vector $q = [q_1, \ldots, q_m]$ (that adds up to 1). An algorithm $\mathcal{M}$, on input D, randomly samples an i from q and returns $\mathcal{M}_i(D)$. An upper bound on the privacy cost of $\mathcal{M}$ is given by Lemma 3.

Lemma 3. *Define* $H_\alpha(P_1; P_2) = e^{(\alpha-1)\,\mathrm{D}_\alpha(P_1 \parallel P_2)}$. *Let* $\mathcal{M}_1, \ldots, \mathcal{M}_m$ *be mechanisms and* $q = [q_1, \ldots, q_m]$ *be a probability vector over* $1, \ldots, m$. *Let* $\mathcal{M}$, *on input* D, *sample* $i \sim q$ *and return* $\mathcal{M}_i(D)$. *Then*

$$H_\alpha(\mathcal{M}(D_1); \mathcal{M}(D_2)) \le \sum_{j=1}^{m} q_j H_\alpha(\mathcal{M}_j(D_1); \mathcal{M}_j(D_2)).$$

The final privacy cost of $\mathcal{M}$ can be obtained by noting $\mathrm{D}_\alpha(P_1 \parallel P_2) = \log(H_\alpha(P_1; P_2))/(\alpha - 1)$. Algorithm 2 combines Lemmas 2 and 3 to compute the cost of the version of Algorithm 1 in which data records are first randomly permuted.

4.3 SGD with Averaging

One of the drawbacks of SGD is its progress slows as it approaches the optimum (due to the requirement for diminishing step size [22], it has sublinear convergence rate even when the objective function is strongly convex). To alleviate this problem, every τ epochs, RSGD-AR performs averaging of parameters [25, 21] (line 13) over the most recent τ epochs. It then resets the step size (line 14), thus emulating restarts. Instead of starting from scratch, the algorithm uses the averaged value of last $m\tau$ iterates as the starting value of the restart [17]. It is known that both averaging and warm restarts help improve convergence properties of stochastic approximation algorithms. The full algorithm, with these enhancements (RSGD-AR), is described in Algorithm 3.

Algorithm 3: RSGD-AR Algorithm

Input: total number of epochs T, initial step size η_0, averaging interval τ, number of mini-batches m, noise scale parameter σ^2

1 Permute the dataset D
2 Construct mini-batches $B_0, \ldots, B_{m-1}$
3 Initialize $\mathbf{w}_0$
4 $t = 0$, $h = 0$
5 **for** $s = 1$ **to** T **do**
6 $h = h + 1$ // epoch count of each run
7 $\eta = \eta_0/h$
8 **for** $j = 0$ **to** $m - 1$ **do**
9 $t = t + 1$
10 $\mathbf{g}_t = \frac{1}{|B_j|} \sum_{i \in B_j} \nabla f_i(\mathbf{w}_{t-1}, d_i)$
11 $\mathbf{w}_t = \mathbf{w}_{t-1} - \eta \mathbf{g}_t$
12 **if** $s \mod \tau = 0$ **then**
13 $\mathbf{w}_t = \frac{1}{m\tau} \sum_{k=0}^{m\tau-1} \mathbf{w}_{t-k}$ // averaging
14 $h = 0$ // step size reset
15 $t = t + 1$
16 sample $Y \sim \mathcal{N}(0, \sigma^2)$
17 **return** $\mathbf{w}_{\mathsf{priv}} = \mathbf{w}_t + Y$

Let $T_j = j\tau$ denote an epoch at which the j^{th} averaging is performed (so it is update number $jm\tau + j$).

$$\|\mathbf{w}_{jm\tau+j} - \mathbf{w}'_{jm\tau+j}\|_2 = \left\| \frac{1}{m\tau} \sum_{k=(j-1)mt+j}^{jm\tau+j-1} \mathbf{w}_k - \mathbf{w}'_k \right\|_2 \le \frac{1}{m\tau} \sum_{k=(j-1)m\tau+j}^{jm\tau+j-1} \|\mathbf{w}_k - \mathbf{w}'_k\|_2 \quad (5)$$

Thus, for a given permutation of the data, we can average the upper bound on parameter differences before applying Lemma 3. That is, every τ epochs, we average the most recent $m\tau$ values of Δ in Algorithm 2. Putting it all together:

Proposition 3. *Algorithm 3 with averaging satisfies* (α, ϵ)*-RDP, where*

$$\epsilon = \frac{1}{\alpha-1} \log \left(\frac{1}{m} \sum_{j=1}^{m} e^{\frac{\alpha(\alpha-1)(\Delta[j])^2}{2\sigma^2}} \right).$$

5 EXPERIMENTAL RESULTS

In this section, we evaluate the performance of the proposed algorithm using 5 real datasets: (1) `Adult` [7, 16] data extracted from the 1994 US Census Data. (2) `BANK` [16] data about marketing campaigns of a

Dataset	Size (n)	Dimension.
Adult	48,842	124
BANK	45,211	33
IPUMS-US	40,000	58
IPUMS-BR	38,000	53
KDDCup99	4,898,431	120

Table 1: Characteristics of datasets

finance institution. (3) `IPUMS-BR` and (4) `IPUMS-US` data extracted from IPUMS data [24] (5) `KDDCup99` data collected from a simulated network. Table 1 shows the number of records and number of attributes (after pre-processing).

Baselines We compare the proposed algorithm, RSGD-AR, against seven baseline algorithms,[3] namely, ObjPert [14, 9], OutPert-GD [32], DP-AGD [15], SGD-MA [1], NSGD (described below), NonPrivate, and Majority. ObjPert is an objective perturbation method that optimizes the objective function perturbed with random noise. OutPert-GD is an output perturbation method that injects Gaussian noise to the solution obtained by running GD with a fixed step size. DP-AGD is a gradient perturbation method that uses carefully chosen step sizes with adaptive privacy budget. SGD-MA is also a gradient perturbation based SGD algorithm that uses an improved composition method, called moments accountant. NSGD takes the Vanilla algorithm (essentially equivalent to [31]) but uses numerical sensitivity calculations to reduce added noise. NonPrivate optimizes the objective with L-BFGS [19] and does not add privacy noise. Majority predicts the most frequent label. We repeat 5-fold cross validation 20 times and report average classification accuracy and final objective value.

Parameter settings Throughout all the experiments the value of privacy parameter δ is fixed to 10^{-8} for the `Adult`, `BANK`, `IPUMS-US`, and `IPUMS-BR` datasets and to 10^{-12} for the `KDDCup99` dataset. The mini-batch size is fixed to 4,000 for RSGD-AR and $\sqrt{n}$ for SGD-MA, where n is the size of dataset.

Preprocessing We performed the following standard preprocessing operations. Every categorical attribute is converted into a set of binary variables using one-hot encoding. For each unique category, a new binary attribute is created. All numerical attributes are rescaled into the range $[0, 1]$ to ensure that they have the same scale. Additionally, for those methods that require feature vectors to lie inside a bounded space, we normalize each observation to a unit norm (i.e., $\|\mathbf{x}_i\|_2 = 1$ for $i = 1, 2, \ldots, n$).

5.1 Logistic Regression & SVM

We compare performance on two different tasks: regularized logistic regression (LR) and support vector machine (SVM). For logistic regression, we define

$$f(\mathbf{w}) = \log(1 + \exp(-y_i \mathbf{w}^\intercal \mathbf{x}_i)) + \frac{\lambda}{2}\|\mathbf{w}\|_2^2,$$

where $\mathbf{x}_i \in \mathbb{R}^{p+1}$, $y_i \in \{-1, +1\}$, and $\lambda > 0$ is a regularization coefficient. The value of λ is fixed to 0.001 in all experiments. Figure 1 shows the classification accuracy (top) and final objective values (bottom) of algorithms on the four datasets. For all the values of ϵ tested, the proposed RSGD-AR algorithm consistently outperforms or performs competitively with other baselines. In all figures, the accuracy of SGD-MA suddenly surges to almost that of non-private algorithm at a certain value of ϵ (because, for each privacy budget it can handle, we ran it with multiple choices of pre-specified number of iterations, to tune its performance).

For SVM, in order to satisfy the differentiability conditions in Section 3.3, we use the huberized hinge loss function [8, 9] defined as

$$\ell_{\text{huber}}(z) = \begin{cases} 0 & \text{if } z > 1 + h \\ \frac{1}{4h}(1 + h - z)^2 & \text{if } |1 - z| \le h \\ 1 - z & \text{if } z < 1 - h, \end{cases}$$

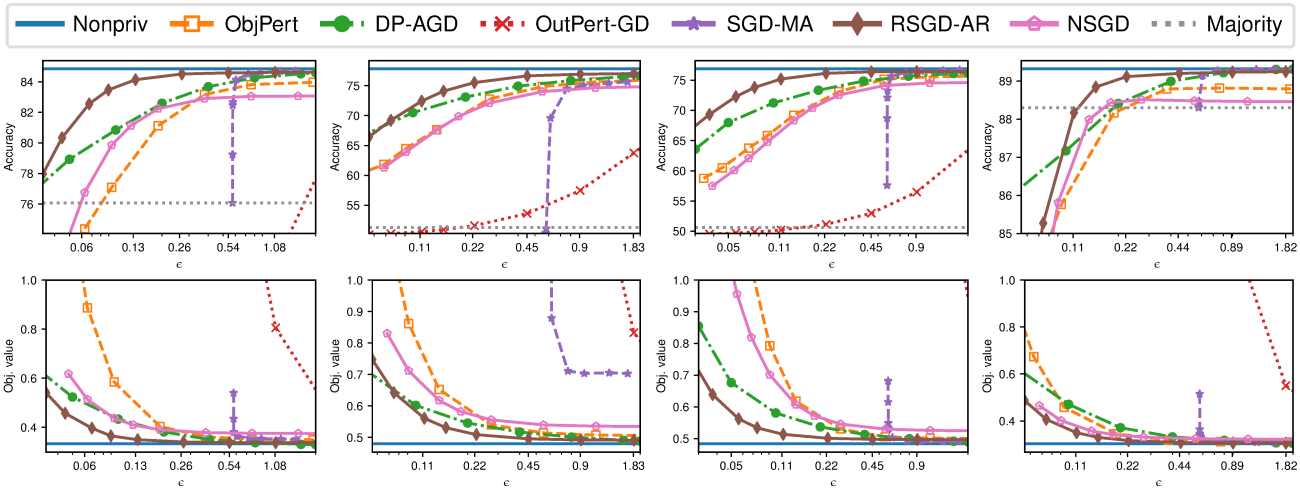
where $z = y\mathbf{w}^\intercal \mathbf{x}$ and h is a hyperparameter. In our experiments, we fixed $h = 0.5$. The performance of the proposed algorithm on SVM task is shown in Figure 2. As it is shown, RSGD-AR outperforms or achieves similar performance with other baseline algorithms.

5.2 Processing Time

To compare the speed of algorithms, we generated 5 subsamples of size 50k, 100k, 150k, 200k, and 250k from `KDDCup99` dataset [4] tasks are and measured each algorithm's processing time on them. As shown in Figure 3, ObjPert which uses L-BFGS to solve the perturbed problem is the fastest. It is observed that L-BFGS finds an approximate solution with reasonable accuracy on `KDDCup99` dataset in less than 20 iterations. While our method requires more iterations, it is as fast as the ObjPert algorithm.

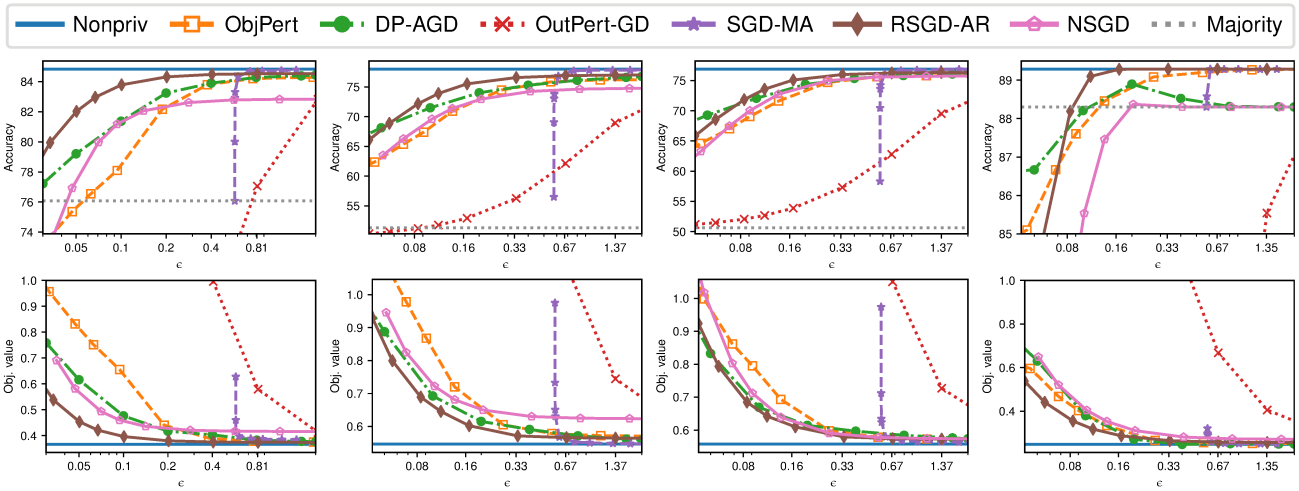
[3] We omit a comparison to the noisy SGLD in [12] because, due to its large per-iteration noise requirement, for small values of δ and datasets used in our experiments we observed it diverges unless ϵ is very large.

[4] Additional experimental results on this dataset are provided in the supplementary material.



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

Figure 1: Logistic regression by varying ϵ (Top: classification accuracies, Bottom: objective values)



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

Figure 2: SVM by varying ϵ (Top: classification accuracies, Bottom: objective values)

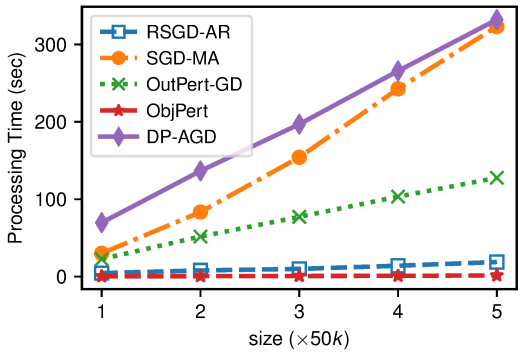


Figure 3: Processing times for 5 different subsamples of KDDCup99 dataset

6 CONCLUSIONS

We presented an SGD algorithm for Rényi differentially private convex empirical risk minimization. It is based on output perturbation, which allows it to take advantage of an averaging technique for accelerating SGD, and also accounts for the added privacy caused by batch randomization. This algorithm is suitable for large scale problems and experimentally outperformed prior (ϵ, δ)-differentially private algorithms.

Acknowledgements

This work was supported by NSF Awards CNS-1702760.

References

[1] 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*, pages 308–318. ACM, 2016.

[2] F. Bach. Adaptivity of averaged stochastic gradient descent to local strong convexity for logistic regression. *Journal of Machine Learning Research*, 15:595–627, 2014.

[3] R. Bassily, A. Smith, and A. Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *Proceedings of the 2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, FOCS '14, pages 464–473, Washington, DC, USA, 2014. IEEE Computer Society.

[4] D. P. Bertsekas. *Nonlinear programming*. Athena Scientific, 2nd edition, 2008.

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

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

[7] C.-C. Chang and C.-J. Lin. Libsvm: a library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

[8] O. Chapelle. Training a support vector machine in the primal. *Neural computation*, 19(5):1155–1178, 2007.

[9] K. Chaudhuri, C. Monteleoni, and A. D. Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(Mar):1069–1109, 2011.

[10] 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*, pages 486–503. Springer, 2006.

[11] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, pages 265–284. Springer, 2006.

[12] V. Feldman, I. Mironov, K. Talwar, and A. Thakurta. Privacy amplification by iteration. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 521–532. IEEE, 2018.

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

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

[15] 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.

[16] M. Lichman. UCI machine learning repository, 2013.

[17] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR) 2017 Conference Track*, Apr. 2017.

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

[19] J. Nocedal. Updating quasi-newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782, 1980.

[20] N. Papernot, S. Song, I. Mironov, A. Raghunathan, K. Talwar, and Úlfar Erlingsson. Scalable private learning with pate. In *International Conference on Learning Representations (ICLR)*, 2018.

[21] B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

[22] H. Robbins and S. Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[23] B. I. Rubinstein, P. L. Bartlett, L. Huang, and N. Taft. Learning in a large function space: Privacy-preserving mechanisms for svm learning. *Journal of Privacy and Confidentiality*, 4(1):4, 2012.

[24] S. Ruggles, K. Genadek, R. Goeken, J. Grover, and M. Sobek. Integrated public use microdata series, minnesota population center. http://international.ipums.org.

[25] D. Ruppert. Efficient estimations from a slowly convergent robbins-monro process. Technical report, Cornell University Operations Research and Industrial Engineering, 1988.

[26] E. K. Ryu and S. Boyd. Primer on monotone operator methods. *Appl. Comput. Math*, 15(1):3–43, 2016.

[27] S. Song, K. Chaudhuri, and A. D. Sarwate. Stochastic gradient descent with differentially private updates. In *GlobalSIP*, 2013.

[28] K. Talwar, A. Thakurta, and L. Zhang. Private empirical risk minimization beyond the worst case: The effect of the constraint set geometry. *arXiv preprint arXiv:1411.5417*, 2014.

[29] K. Talwar, A. G. Thakurta, and L. Zhang. Nearly optimal private lasso. In *Advances in Neural Information Processing Systems*, pages 3025–3033, 2015.

[30] D. Wang, M. Ye, and J. Xu. Differentially private empirical risk minimization revisited: Faster and more general. In *Advances in Neural Information Processing Systems 30*, pages 2719–2728. Curran Associates, Inc., 2017.

[31] 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*, SIGMOD '17, pages 1307–1322. ACM, 2017.

[32] J. Zhang, K. Zheng, W. Mou, and L. Wang. Efficient private erm for smooth objectives. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3922–3928. AAAI Press, 2017.