Continual and Sliding Window Release for Private Empirical Risk Minimization

Abstract

It is difficult to continually update private machine learning models with new data while maintaining privacy. Data incur increasing privacy loss – as measured by differential privacy – when they are used in repeated computations. In this paper, we describe regularized empirical risk minimization algorithms that continually release models for a recent window of data. One version of the algorithm uses the entire data history to improve the model for the recent window. The second version uses a sliding window of constant size to improve the model, ensuring more relevant models in case of evolving data. The algorithms operate in the framework of stochastic gradient descent. We prove that even with releasing a model at each time-step over an infinite time horizon, the privacy cost of any data point is bounded by a constant ϵ differential privacy, and the accuracy of the output models are close to optimal. Experiments on MNIST and Arxiv publications data show results consistent with the theory.

Introduction

Differential privacy (often abbreviated as DP) is a rigorous mathematical definition that characterizes the privacy properties of algorithms. It measures the information that can be learned about an individual by observing the output of the algorithm. The definition is quite general, and since its introduction in Dwork et al. [2006], differential privacy has become the standard privacy notion in many areas. It is currently used by major online service providers and organizations including the United States Census Bureau [Abowd 2018].

Differentially private machine learning is commonly used to protect personal privacy in analytics of data such as in medical, social and online applications [Kim, Jang, and Yoo 2018, Task and Clifton 2012]. In these scenarios, data often evolve, and new machine learning models must be computed incorporating the new data. However, model recomputation poses a challenge – it degrades privacy as more information is leaked along with the new model. In this paper, we consider the problem of releasing updated ML models while limiting privacy loss.

Our aim is to release models based on the "new" data, while also leveraging historical information. There are multiple reasons for this approach. First, we do not wish training sets to be disjoint, but instead want *sliding window* models, e.g. those that are accurate on data for the *last n days* – reflecting recent trends in data. Second, the most recent block of data may be insufficient for training complex models, we wish to leverage historical data to enhance and stabilize the model. Unfortunately, reusing historical data in this context poses a challenge to maintaining differential privacy.

Previous works on differential privacy for evolving datasets have considered questions of releasing a model for the entire cumulative data. In Chan, Shi, and Song [2011], Dwork et al. [2010], specific simple statistics such as a count are released at constant intervals of time. The work in Cummings et al. [2018] can operate with more general queries such as linear histogram queries and empirical risk minimization (ERM), but publishes models at exponentially growing intervals. In contrast to these approaches, our priority is to maintain up-to-date private models with accuracy guarantees for "recent data".

In our approach, models are published at constant time intervals, but can be applied to general computations such as linear queries over histograms and ERM; we focus particularly on the important case of regularized ERM. At time t, the models make use of historical data – either the entire historical data, or most recent w items. The models are shown to maintain guaranteed accuracy on the most recent data, and good accuracy on the historic data where it is still relevant. We show that even with constant interval release over an unbounded time horizon, the privacy leak in our method is bounded by a constant ϵ-DP.

Our contributions. Our objective is to periodically release models that are accurate on a recent *target* data window of size b_0, while making use of a larger *source* data window w. This problem can be viewed as domain adaptation or transfer learning (see Redko et al. [2020] for a survey on transfer learning). In a scenario unconcerned about privacy, simple transfer learning can be applied repeatedly to achieve these models; our challenge is to ensure differential privacy through repeated data use.

In the section *Continual Cumulative Updates* we describe algorithms for the case where the entire historical dataset is used as the training window w. As more data accumulates, we maintain differentially private *base* models on the growing historical data. These base models are updated regularly to get source models, and at longer intervals they are recomputed from scratch. We prove bounds on the performance of the model on both the new and old data. We show that the privacy loss is bounded by ϵ-DP even when data is reused over an infinite time horizon.

The question of a fixed-size (or sliding-window) of historical data w is considered in the *Sliding window model release* section. This problem arises when the data distribution changes over time, so that very old data is no longer representative. Thus, our main challenge is to update models to "forget" old data. We maintain a hierarchic sequence of updates to a base model that are removed and added appropriately with the sliding window, and show privacy and utility bounds as above.

In the experimental results, we show the performance for logistic regression on two datasets – MNIST and Arxiv publication records. Empirical results show that the proposed algorithms demonstrate the expected trade-offs between privacy and utility for varying levels of regularization and datasize. Accuracy of the private models approaches non-private performance for strong privacy ($\epsilon = 1$) given appropriate levels of regularization and minimum update sizes.

For the sake of simplicity, we have focused the discussion on a single data stream which contains both the source w and target b_0. But in the usual transfer learning setup, these methods can be used for *private dynamic transfer learning* – where w is taken in the source domain which has more data, while b_0 is taken in a target domain which is sparser in data, and both datasets evolve with time.

Related Work

Differentially private machine learning has been a major topic in recent years; see Ji, Lipton, and Elkan [2014] for a survey. The most prominent setting is the static database, with the objective of empirical risk minimization with differential privacy. Differentially private logistic regression was described in Chaudhuri and Monteleoni [2009] and extended to general regularized empirical risk minimization by Chaudhuri, Monteleoni, and Sarwate [2011]. Further analyses of DP-ERM for strongly convex functions were described in Kifer, Smith, and Thakurta [2012], Wang, Ye, and Xu [2017] and Algorithm 3 of Bassily, Smith, and Thakurta [2014].

Evolving databases have been considered in different forms. Online learning is a setup where the evolving input and loss functions may be adversarial, and the objective is to minimize *regret*, which measures overall error. Differentially private versions of online learning have been studied in Jain and Thakurta [2013], Guha Thakurta and Smith [2013] and Agarwal and Singh [2017]. A recent work has considered batch online DP learning minimizing regret and excess population risk [Kairouz et al. 2021]. In contrast, our setup involves independently evolving data, a sliding wondow and the objective is to minimize the empirical error.

Continual release of query answers on the entire dataset was described in Dwork et al. [2010] and Chan, Shi, and Song [2011] for a restricted input type. They consider a single bit of input at every round (such as a single element possibly being added to a the set) and publish a differentially private count in every round. A question of adaptive analysis on evolving data was considered in Cummings et al. [2018]. Building on previous works [Blum, Ligett, and Roth 2013, Hardt and Rothblum 2010] on adaptive queries, Cummings et al. [2018] describe releasing responses to a broader class of queries. Instead of every round, these results are published at exponentially growing intervals. Our algorithm instead provides updated algorithms are constant-sized (b_0) intervals, which is a significant step towards full continual release. Private matrix analysis in the sliding window model has been studied by Upadhyay and Upadhyay [2020].

Transfer learning or domain adaptation has been studied from various perspectives ([David et al. 2010, Mansour, Mohri, and Rostamizadeh 2009]). A recent survey can be found in [Redko et al. 2020]. A variant called *hypothesis transfer learning* [Kuzborskij and Orabona 2013, Mansour, Mohri, and Rostamizadeh 2008] is particularly relevant for us. In this variant, once a suitable hypothesis (model) on the source domain is computed, the source data is no longer accessed, and only the hypothesis (model) is used for learning in the target domain.

Preliminaries

Empirical risk minimization Let $D = \{(\mathbf{x}_i, y_i) : i \in [1, n]\}$ represent a dataset of training examples drawn from some underlying distribution $\mathcal{D} \sim \mathcal{X} \times \mathcal{Y}$. Consider a set of candidate models $\mathcal{F}$ where $f : \mathcal{X} \to \mathcal{Y}$ for $f \in \mathcal{F}$ and f is parameterized by weights $\mathbf{w} \in \mathcal{W}$ e.g. $\mathcal{W} = \mathbb{R}^d$. The loss of model $f \in \mathcal{F}$ for each data point is given by a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. Empirical risk minimization describes the learning paradigm of selecting the predictor $\hat{f} \in \mathcal{F}$, parameterized by weights $\mathbf{w}_{\hat{f}}$ such that,

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \hat{L}_D(\mathbf{w_f}) = \arg\min_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \ell(f(\mathbf{x_i}), y_i) \quad (1)$$

$\hat{L}_D(\mathbf{w})$ represents the empirical risk, the true risk is denoted by $L(\mathbf{w}) = \mathbb{E}_{D \sim \mathcal{D}}[\ell(f(\mathbf{x_i}), y_i)]$. Regularized empirical risk minimization with regularization parameter $\lambda > 0$ includes a penalization term in the loss function of the form $\ell(f(\mathbf{x_i}), y_i) \leftarrow \ell(f(\mathbf{x_i}), y_i) + \lambda \|\mathbf{w}_f\|_2^2$, ensuring the λ-strong convexity of the loss function.The following assumptions are commonly made about the loss function ℓ:

Definition 1 (Convexity). *A function $f : \mathcal{X} \to \mathcal{S}$ satisfies λ-strong convexity if for all $x, y \in \mathcal{X}$:*

$$f(x) \geq f(y) + \nabla f(x)^T (x - y) + \frac{\lambda}{2} \|x - y\|_2^2$$

Definition 2 (Lipschitzness). *A function $f : \mathcal{X} \to \mathcal{S}$ is L-Lipschitz if for any $x, y \in \mathcal{X}$: $\|f(x) - f(y)\|_2 \leq L\|x - y\|_2$*

Definition 3 (smoothness). *A differentiable function $f : \mathcal{X} \to \mathcal{S}$ is β-smooth if for ant $x, y \in \mathcal{X}$: $\|\nabla f(x) - \nabla f(y)\|_2 \leq \beta \|x - y\|_2$*

Differential privacy is a rigorous mathematical definition of privacy, describing the privacy properties of a randomized algorithm A. It is defined in terms of a pair of neighboring databases (D, D'): Two databases D, D' are neighboring if $H(D, D') \leq 1$, where $H(\cdot, \cdot)$ represents the Hamming distance. This corresponds to event-level privacy, where the presence or absence of single datapoints is obscured.

Definition 4 (Differential privacy). *A randomized algorithm A satisfies ϵ-differential privacy if for all neighboring databases D, D' and for all possible outputs $O \subseteq Range(A)$,*

$$\Pr[A(D) \in O] \leq e^{\epsilon} \cdot \Pr[A(D') \in O].$$

The L2 sensitivity of a function is the maximum change in the function value between neighboring databases: $\Delta f = \max_{D,D' \in \mathcal{D}} |f(D) - f(D')|_2$. The sensitivity is used to determine the noise added by A to achieve differential privacy, e.g. the Laplace mechanism works as follows:

Definition 5 (Laplace mechanism). *Given any function $f : \mathcal{D} \to O$ and privacy parameter ϵ, for any $D \in \mathcal{D}$, the Laplace Mechanism returns: $f(D) + \nu$, where $\nu \sim \mathrm{Lap}\left(\frac{\Delta f}{\epsilon}\right)$.*

$\mathrm{Lap}(b)$ is the Laplace distribution with mean 0 and scale b. It is known that this mechanism preserves differential privacy [Dwork et al. 2006] (abbreviated as ϵ-DP), and that if a sequence of randomized algorithms A_i are applied on a dataset, each with ϵ_i-DP, then the sequence satisfies $(\sum_i \epsilon_i)$-DP. More advanced composition theorems also exist [Kairouz, Oh, and Viswanath 2015].

Problem description

Suppose data points $(\mathbf{x}, y) \in \mathcal{X} \times \mathcal{Y}$ arrive in a continual manner and dataset $D = \{(\mathbf{x_i}, y_i) : i \in [0, t]\}$ represents all points that have arrived by time t. The dataset $D_{[t_1:t_2]} = \{(\mathbf{x_i}, y_i) : i \in [t_1, t_2], 0 \leq t_1 \leq t_2 \leq t\}$ represents the set of data points that arrived in the time interval $[t_1, t_2]$, with $D_{[t_1:t_2]} \subseteq D_{[0:t]}$.

Our goal is to compute an ERM model for the most recent batch b_0 of data i.e. $D_{[t-b_0:t]}$. We assume that it is sufficient to release a model at constant intervals representing a datasize of at least b_0, that is at each t that is a multiple of b_0. Further, we assume that a minimum block size $B \geq b_0$ is necessary for accurate models. We wish to use information from a historical data window $D_{[t-w:t]}$ of size $w > b_0$ to augment the training on $D_{[t-b_0:t]}$.

We consider two versions: cumulative continual updates (with window $D_{[t-w:t]} = D_{[0,t]}$) and sliding window ($D_{[t-w:t]} = D_{[t-j,t]}$) for a constant $j > b_0$. In both these cases, our objective is to obtain an accurate model for $D_{t-b_0,t}$, with an additional objective that the model should also have high accuracy for $D_{t-w,t}$. Note that this may not be possible if the distributions of the source window W and target b_0 are significantly different. We also require a differential privacy guarantee protecting the presence or absence of a single data sample in the dataset (event-level privacy as per Chan, Shi, and Song [2011]). A constant differential privacy guarantee must be ensured in all cases.

We assume that data arrives at a constant rate. In scenarios where this is not the case, the results can be applied by defining dynamic time steps as the interval that accumulates a certain constant number of data points.

Basic approaches and multi-resolution release

Before describing the main results, we briefly discuss some basic and existing approaches that are relevant. Consider a simple scenario where in each time-step a single element may arrive. The sensitivity of the *count* function is 1 – as a single element changes it by 1. In each time step t, the count can be released with a $\mathrm{Lap}\,(1/\epsilon)$ noise, this guarantees ϵ-DP for the release at time t, however, data that arrives in early rounds suffer information leak in each round. By the composition properties of differential privacy, data that contributes to T releases of the count, has $T\epsilon$-DP. To ensure ϵ-DP for each element, the noise scale must grow with time, requiring $\mathrm{Lap}\,(T/\epsilon)$ noise at round T – which is excessive for most purposes.

Certain important classes of functions have smaller sensitivity that yields more efficient algorithms. Linear queries over histograms was considered in [Cummings et al. 2018, Hardt and Rothblum 2010, Blum, Ligett, and Roth 2013] etc. Suppose U is a finite data universe of size N, and $D \in U^n$ is a database. We write it as a histogram x, where x^i as the fraction of x of type $i \in [N]$, that is: $x^i = n_i/n$. A linear query is described by a vector $f \in [0,1]^N$, where the objective is to return $\langle f, x \rangle$. In this format, the presence or absence of a single element changes the histogram entry for the type by at most $1/n$, and thus has sensitivity of $1/n$.

In a machine learning context, similar low sensitivity arises for λ-strongly convex loss functions and regularized ERM (Eq.1). In this case, it can be shown that the sensitivity is bounded by $O\left(\frac{1}{\lambda n}\right)$ [Chaudhuri, Monteleoni, and Sarwate 2011]. This result is particularly significant, since many important machine learning methods, including regularized convex Stochastic Gradient Descent fit this mold [Wu et al. 2017]. An algorithm for private SGD is shown as Algorithm 1. For such a mechanism, where the sensitivity is $O(1/n)$, if one ERM model is released in every round with a $\mathrm{Lap}\,(1/\epsilon)$ noise, then the information leak is bounded by $\Theta((\log n)\epsilon)$-DP.

Algorithm 1: Private SGD via Output Perturbation (PSGD) [Wu et al. 2017]

```
1: Input: $D = \{(\mathbf{x}_t, y_t)\}$, inverse learning rate $\gamma$, sensitivity $\Delta_\epsilon$, number of iterations $m$.
2: $\mathbf{w} \leftarrow SGD(D)$ with $k$ passes and learning rate $\frac{1}{\gamma i}$ for iteration $i$.
3: return $\mathbf{w} + \nu$ where $\nu \overset{d}{\sim} Lap\,(\Delta_\epsilon)$.
```

Hierarchical and multiresolution release approach. As a warm-up, we describe a mechanism to release models with a constant ϵ-differential privacy guarantee. This approach does not achieve the continual release property of producing an output at each round trained using all previous data, but releases these global outputs at exponentially growing intervals (such as in Cummings et al. [2018]).

In this approach, a query result (such as a model trained via as shown in Algorithm 1) is released at times $t = 2^k B$ for $k = 0, 1, 2, 3, \ldots$. This approach is capable of more than releasing the simple cumulative model. By using a binary hierarchy, we can release models at $\log t$ different scales of sizes $qB, q \in \mathbb{Z}^+$. The approach is simply the following: At any time $t = (2^k)qB$ (for $q, k \in \mathbb{Z}^+$), we release a model computed on data in the interval $[t - 2^k B : t]$. Algorithm 2 shows the idea.

Theorem 1. *Algorithm 2 satisfies ϵ-differential privacy.*

Algorithm 2: Private Multi-Resolution Release

```
1: Input: $D$, $B$, $\Delta_\epsilon = \frac{4L}{\lambda B \epsilon}$ where $L$ is the Lipschitz constant of $\ell$.
2: for $t = 2^k qB$ for $q, k \in \mathbb{Z}^+$ do
3:     $\mathbf{w^t} \leftarrow PSGD(D = \{(\mathbf{x}_j, y_j) | j \in [t - 2^k B + 1, t]\}, \Delta_\epsilon)$ (Algorithm 1)
4:     Release $\mathbf{w^t}$
5: end for
```

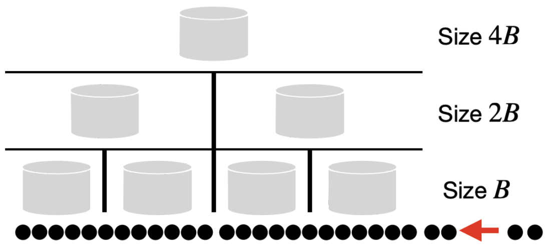


Figure 1: Incoming datapoints are partitioned by Algorithm 2 to form a hierarchy of models with datasizes $2^k B$. Tailored noise addition results in a constant privacy guarantee for any datapoint.

Let us now consider the utility achieved by Algorithm 2. Denote the output of a non-private stochastic gradient descent algorithm by $\mathbf{w}$, this corresponds to line 2 in Algorithm 1. Let $\mathbf{w}^*$ represent the non-private minimizer $\mathbf{w}^* = \arg\min_{\mathbf{w}} \hat{L}_D(\mathbf{w})$. Denote the corresponding private released parameters by $\mathbf{w}_{priv}$. In other words, $\mathbf{w}_{priv}$ is the privatized set of parameters returned in line 3 of Algorithm 1. Theorem 2 describes the utility of Algorithm 2.

Theorem 2 ([Wu et al. 2017]). *. Consider 1-pass private stochastic gradient descent via Algorithm 1 for β-smooth and L-Lipschitz loss function ℓ. Suppose $sup_{\mathbf{w}\in\mathcal{W}} \|\ell'(\mathbf{w})\| \le G$, $\|\mathbf{x}\|_1 \le 1$, $\mathcal{W}$ has diameter R and $\mathbf{w} \in \mathbb{R}^d$. Then, for a dataset of size $2^k B$ with $k \in [0, \lfloor log_2(\frac{t}{B}) \rfloor]$*

$$\mathbb{E}[\hat{L}_D(\mathbf{w}_{priv}) - \hat{L}_D(\mathbf{w}^*)] \le \frac{((L + \beta R^2) + G^2)\log(2^k B)}{\lambda 2^k B} + \frac{4dG^2}{\epsilon \lambda B}$$

While this produces a model based on the most recent data at different scales, this approach does not quite give us the continual or sliding window release we require, and instead releases results in windows of varying sizes. It also assumes a large minimum block size of B that is sufficient for accurate training. It does not produce updated models for smaller blocks of size b_0 and in many cases does not make use of historical data.

Continual cumulative updates

To release a model at constant intervals using the cumulative data, we adapt the previous hierarchical mechanism. Our approach is: at any time, we have a *base* model $\mathbf{w}_g$, which has been computed on an early sequence of data up to time t_g, in the regular offline manner, with added privacy (e.g. using Algorithm 1). As new data arrive, we update $\mathbf{w}_g$. The update itself proceeds in a hierarchical manner. After a suitable interval, e.g. when $t = 2t_g$, the process is reset, and the base model is recomputed on the entire dataset. Let us now consider the details of the scheme.

Upon receiving b_0 items after t_g, i.e., at time $t = t_g + b_0$, the model f^{t_g} is updated to f^t using the data $D_{[t_g+1:t]}$. This update itself is a regularized optimization:

$$f^t = \arg\min_{\mathbf{w}\in\mathcal{W}} \frac{1}{b_0} \sum_{i\in[t_g+1,t]} \ell(f(\mathbf{x}_i), y_i) + \lambda \|\mathbf{w} - \mathbf{w}_g\|_2^2 \tag{2}$$

This is solved via a private algorithm shown as Algorithm 3, which can be realized as an SGD algorithm such as Algorithm 1.

This optimization treats the base model $\mathbf{w}_g$ as a regularization point, or *origin*, and thus the output model f^t stays close to this global model. Further updates at time $t^+ = t + b_0$ can be done with respect to f^t as: $f^{t^+} = \arg\min_{\mathbf{w}\in\mathcal{W}} \frac{1}{b_0} \sum_{i\in[t+1,t+b_0]} \ell(f(\mathbf{x}_i), y_i) + \lambda \|\mathbf{w} - \mathbf{w}_t\|_2^2$. A new base model is computed at $t = 2^k B$ $D_{[t-w:t]}$ (Algorithm 4).

Algorithm 3: Private Biased Regularized ERM (PBERM)

```
1: Input: Base model $f_g$ with weights $\mathbf{w}_g$, $D_{[t_g+1:t]}$, $\lambda$, $\epsilon$, and $L$.
2: $\mathbf{w}^t = \arg\min_{w\in\mathcal{W}} \frac{1}{b_0} \sum_{i\in[t_g+1,t]} \ell(f(\mathbf{x}_i), y_i) + \lambda \|\mathbf{w} - \mathbf{w}_g\|^2$
3: return $\mathbf{w}^t + \nu$ where $\nu \overset{d}{\sim} Lap\left(\frac{4L}{\lambda b_0 \epsilon}\right)$
```

Algorithm 4: Private Continual Release

```
1: Input: $D$, $\lambda$, $\epsilon$, $L$, $b_0$, $B$.
2: for $t \in \mathbb{Z}^+$ and $t \ge B$ do
3:     if $t = 2^k B$ for $k \in \mathbb{Z}^+$ then
4:         Learn base model $f_g$ on $D_{[0:t]}$ using Algorithm 2.
5:         Release $f_g$ and save $f_c = f_g$ and $t_g = t$
6:     else if $t - t_g = ib_0$ for $i \in \mathbb{Z}^+$ then
7:         if $t - t_g = 2^j b_0$ for $j \in \mathbb{Z}^+$ then
8:             Learn $f^t$ on $D_{[t_g:t]}$ and regularize with model $f_g$ using Algorithm 3
9:             Release $f^t$ and save $f_c = f^t$
10:        else
11:            Learn $f^t$ on $D_{[t-b_0:t]}$, regularize with $f_c$ using Algorithm 3
12:            Release $f^t$
13:        end if
14:    end if
15: end for
```

Privacy and Utility Guarantees

Theorem 3 ([Wu et al. 2017])**.** *Algorithm 3 satisfies ϵ-differential privacy.*

Theorem 4 demonstrates that Algorithm 4 allows the continual release of models over the incoming dataset with only a constant increase in privacy loss. When used together, the total privacy loss of Algorithms 2 and 4 is then at most 2ϵ.

Theorem 4. *Algorithm 4 satisfies ϵ-differential privacy.*

Theorem 5 bounds the excess empirical risk of the continual cumulative models released via Algorithm 4. We

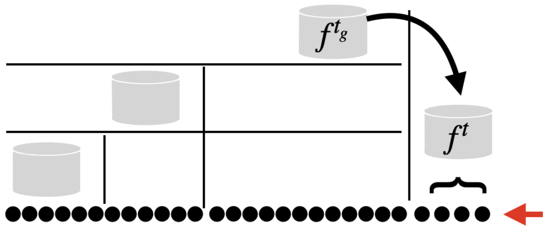


Figure 2: Algorithm 4 updates recent global models using newly arrived data.

denote the model released by $\mathbf{w}_{new}$ and the new data by $D^{new} \sim \mathcal{D}^{new}$. The risk for the true minimizer $\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathbb{E}_{D^{new} \sim \mathcal{D}^{new}}[\ell(f(\mathbf{x}_i), y_i)]$ is given by $L(\mathbf{w}^*)$. Suppose in one particular iteration, we start with a current model $\mathbf{w}_g$ and after update obtain a model $\mathbf{w}_{new}$. Also let us write the expected error of $\mathbf{w}_g$ on the new block of data of size $2^j b_0$ as R_g and assume that the new update block of data represents an I.I.D. sample from some underlying distribution. Note we do not assume that the entire dataset is an I.I.D. sample from the a distribution, only that the most recent block of data is. Theorem 5 demonstrates that, as can be expected, the excess error is low when b_0 is large, and when R_g is small – that is, when the new data set is large and when the old model is a good fit. The error increases additively as $1/\epsilon$.

Theorem 5. *Suppose the loss function used in Algorithm 3 is L-Lipschitz and that* $|\ell|_\infty \leq M$. $\lambda \leq \frac{1}{\|\mathbf{w}^* - \mathbf{w}_g\|_2^2 b_0}$ *with* $\mathbf{w} \in \mathbb{R}^d$ *and update batch data size* $2^j b_0$*, then with probability at least* $1 - e^{-\eta}$*:*

$$\hat{L}_{D_{new}}(\mathbf{w}_{new}) - L(\mathbf{w}^*) \leq$$
$$\sqrt{\frac{2\eta R_g}{2^j b_0}} + \frac{1.5M\eta + 1}{2^j b_0} + \ln\left(\frac{d}{e^{-\eta}}\right)\frac{4dL^2}{\lambda b_0 \epsilon}$$

The proof of this theorem is based their non-private analogs in the hypothesis transfer learning literature [Kuzborskij and Orabona 2013]. Theorem 6 describes the utility of the updated model on the older data. Suppose $\mathbf{w}_g$ was obtained via empirical risk minimization on dataset $D_g \sim \mathcal{D}_g$, with empirical risk $\hat{L}_{D_g}(\mathbf{w}_g)$. The true minimizer for this problem is denoted by $\mathbf{w}_g^* = \arg\min_{\mathbf{w} \in \mathcal{W}} \mathbb{E}_{D \sim \mathcal{D}_g}[\ell(\mathbf{w}, (x_i, y_i))]$ with $L(\mathbf{w}_g^*) = \mathbb{E}_{D \sim \mathcal{D}_g}[\ell(\mathbf{w}_g^*, (x_i, y_i))]$.

Theorem 6. . *Suppose the loss function used in Algorithm 4 is L-lipschitz and λ-strongly convex,* $\lambda \leq \frac{1}{\|\mathbf{w}_{new}^* - \mathbf{w}_g\|^2 b_0}$ *with* $\mathbf{w} \in \mathbb{R}^d$*. Then with probability at least* $1 - e^{-\eta}$

$$|\hat{L}_{D_g}(\mathbf{w}_{new}) - L(\mathbf{w}_g^*)| - |\hat{L}_{D_g}(\mathbf{w}_g) - L(\mathbf{w}_g^*)|$$
$$\leq L\,\|\mathbf{w}^* - \mathbf{w}_g\| \left(\sqrt{2\left(\ln\left(\frac{d}{e^{-\eta}}\right)\frac{2dL^2}{\lambda\epsilon} + 1\right)} + 1\right).$$

Theorem 6 implies that when $\mathbf{w}_g$ is close to optimal model for the new data $\mathbf{w}_{new}^*$, then the excess empirical error on the old data using the updated model is small. Thus, when the data distribution does not change significantly, the model remains valid on the old data. In such cases, increasing the regularization weight λ helps both models. In scenarios where the data distribution changes rapidly, a smaller λ is appropriate, which naturally increases the error of w_g. However, in cases where the distribution evolves, a sliding window of source data will be more appropriate.

Sliding window model release

Evolving data may drift away from its original structure, as the underlying reality and generating distribution slowly changes. In such cases, instead of using the entire cumulative data, which may no longer be relevant for up to date models, we would like to use a sliding window of sufficient size w to generate a source model that can be applied to the most recent w_0 block. Here we use $w_0 \sim b_0$ as the unit of model learning. The challenge in achieving this effect is to update model corresponding to w while maintaining bounded differential privacy. In this section, we develop an algorithm to maintain a model over a sliding window w. This model can then be used as the source to create a model for the most recent block as in the previous section.

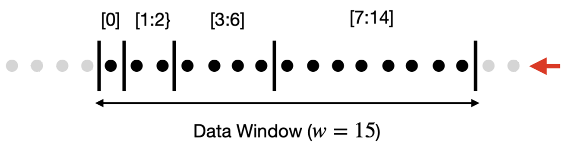
Sliding Window Algorithm

For simplicity of explanation, we consider a window W of size $w = (\sum_{j \in [0,J]} 2^j w_0) - 1$, denoted by $D_{(t-w+1,t)} = \{(\mathbf{x}_i, y_i) : i \in [t - w + 1, t]\}$. The window is initially split into blocks of size 2^j for $j \in [0, J]$ as shown in Figure 3(a). We refer to as $f^{(k-1)}$ a model trained over a large (a constant fraction size, such as half-sized) block in W. The rest of W is split into smaller blocks, each of size power of 2, and models. Figure 3(b) shows the cascade of fine tuning of $f^{(k-1)}$ using the rest of the data of W. As the window progresses with time, these blocks and models are updated or reused appropriately as shown in Figure 3(c) and (d). When the largest block of data is no longer contained in the window, the process is refreshed to the form of Figure 3(b). Observe that we treat the training of new models as fine tuning of the original model, because as seen for continual release, models are close to both source and target domains when these are drawn from similar data. Algorithm 5 and Example 1 describes the method in more detail.

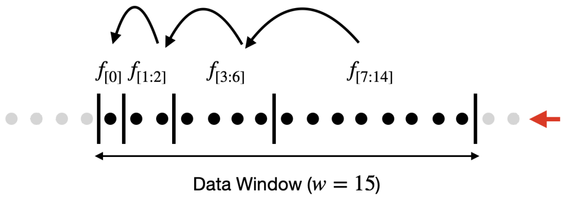
Algorithm 5: Private Sliding Window ERM ($PSWERM$)

```
1: Input: Dataset, $\lambda > 0$, $\epsilon$, $L$, window size $w$, minimum update batch size $w_0$.
2: /* For simplicity assume $w = 2^k - 1$ with $k \in \mathbb{Z}^+$ and $w_0 = 1$*/
3: Partition $D_{[0:w]}$ into buckets, $w^{(i)}$ of sizes $2^i$, $0 \leq i < k$
4: Use Algorithm 1 with $\Delta_\epsilon = \frac{6L}{\lambda\epsilon 2^{k-1}}$ on bucket $w^{(k-1)}$ to get model $f^{(k-1)}$
5: For $0 \leq i < k - 1$, get model $f^i$ using bucket $w^i$ and regularized by $f^{i+1}$
6: for the window sliding along the stream do
7:     Incorporate newly arrived data by modifying the model dependency chain as
       required (see Example 1)
8:     Release the model at the end of the model dependency chain
9: end for
```

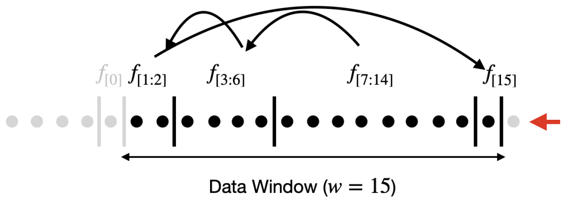
Example 1. Here we describe how the model dependency chains are updated for the case $w_0 = 1$, and $w = 7$ in Algorithm 5 (also see Figure 3 for an example with larger window



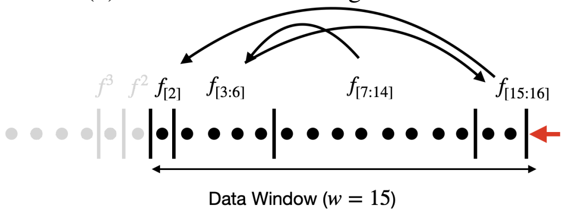
(a) Sliding window size w, $w_0 = 1$.



(b) Cascade of fine tuning of models.



(c) Sliding window update.



(d) Sliding window update.

Figure 3: Sliding window source based model release.

$w = 15$). We skip describing the method for general w_0 and w as its involved, but it follows the same idea discussed here. Let's denote the model trained on $D_{[i:j]}$ as $f_{[i:j]}$. The base dependency chain created for the first window $D_{[0:6]}$ (Line 4) is $f_{[0]} \leftarrow f_{[1:2]} \leftarrow f_{[3:6]}$. Here $f_{[i]} \leftarrow f_{[j]}$ means that $f_{[j]}$ is used as the regularizer to train $f_{[i]}$. When we receive $D_{[7]}$ (and $D_{[0]}$ goes out of window), $f_{[7]}$ is trained ($f_{[0]}$ is discarded) using the dependency chain $f_{[7]} \leftarrow f_{[1:2]} \leftarrow f_{[3:6]}$. Next when $D_{[8]}$ is received, $f_{[7:8]}$ and $f_{[2]}$ are trained and the chain becomes $f_{[2]} \leftarrow f_{[7:8]} \leftarrow f_{[3:6]}$. Next upon receiving $D_{[9]}$ the dependency chain becomes $f_{[9]} \leftarrow f_{[7:8]} \leftarrow f_{[3:6]}$. At the next step $D_{[3:6]}$ goes out of the window and all previous models are discarded to create new buckets and the chain becomes $f_{[4]} \leftarrow f_{[5:6]} \leftarrow f_{[7:10]}$.

In [Datar et al. 2002], a hierarchy is used over a sliding window for simple statistics, but allows the window to be slightly larger or smaller. In contrast, we keep the window size strictly fixed.

Privacy and Utility Guarantees Algorithm 5 allows the release of continually updated ERM models over a sliding window of data, with only constant privacy loss.

Theorem 7. *Algorithm 5 satisfies ϵ-differential privacy.*

Denote the private released model for the window by f^w, parameterized by weights $\mathbf{w}$. As before, the true minimizer is denoted by $\mathbf{w}^* = \arg\min_{\mathbf{w}} \mathbb{E}_{D^{new} \sim \mathcal{D}^{new}}[\ell(f(\mathbf{x}_i), y_i)]$ where $D^{new} \sim \mathcal{D}^{new}$ is the last batch of data used to update the model. See the appendix for an analogue of Theorem 6 in this scenario.

Theorem 8. *Suppose the loss function used in Algorithm 5 is L-Lipschitz, $|\ell|_\infty \leq M$, and $\lambda \leq \frac{1}{\|\mathbf{w}^* - \mathbf{w}_g\|_2^2 w_0}$ with $\mathbf{w} \in \mathbb{R}^d$, then with probability at least $1 - e^{-\eta}$:*

$$\hat{L}_{D^{new}}(\mathbf{w}) - L(\mathbf{w}^*) \leq \sqrt{\frac{2\eta L(\mathbf{w}^*)}{w_0}} + \frac{1.5M\eta + 1}{w_0} + \ln(\frac{d}{e^{-\eta}})\frac{12dL^2}{\lambda w_0 \epsilon}$$

Sampling based enhancements. Sampling can be used to amplify differential privacy [Balle, Barthe, and Gaboardi 2018]. In our algorithms, sampling will reduce noise in updates with larger blocks. In the current version, all update steps use a noise scale of $O(\frac{1}{b_0})$. Sampling can be used to enhance privacy so that smaller noise scales can be used for updates that are computed over blocks larger than b_0. Sampling based algorithms are described in the appendix.

Experiments

In this section we experimentally evaluate the proposed algorithms in two datasets: (i) MNIST dataset of 60k images of handwritten digits of size 28×28. Here we classify the images to recognize the digits (10 class problem) using their 784 pixel values. The images are randomly ordered to form a stream. The experiments only show results till $t = 20$k. We use the test dataset provided with MNIST. (ii) Arxiv dataset contains metadata of 10k articles uploaded to Arxiv from 2007[1] and we classify the category of an article from 17 classes (e.g., 'cond-mat', 'math', etc. excluding the articles that contains multiple categories). The articles are ordered by their first submission date. We consider the vector of word counts from the title as the feature. Here we present the results with dimensions reduced to 512 using PCA to make the base classification efficient. We held out 25% data points as test and use the rest as the training sequence.

All experiments use a logistic regression classifier with SGD. Both MNIST and Arxiv dataset use minibatch of size 256 and run for 500 iterations. In both cases we use Algorithm 1 with $\gamma = 10$. All experiments are repeated 4 times and the plots show the median in solid line with shaded area between 25 and 75 percentiles. All experiments were done on an intel $i7$ cpu of a laptop running Linux. The code is implemented in python using standard libraries including pytorch. The hyperparameters are not tuned for better accuracy as the primary goal of the empirical evaluation is to support the theoretical results by showing the relative performance

[1]https://www.kaggle.com/Cornell-University/arxiv.

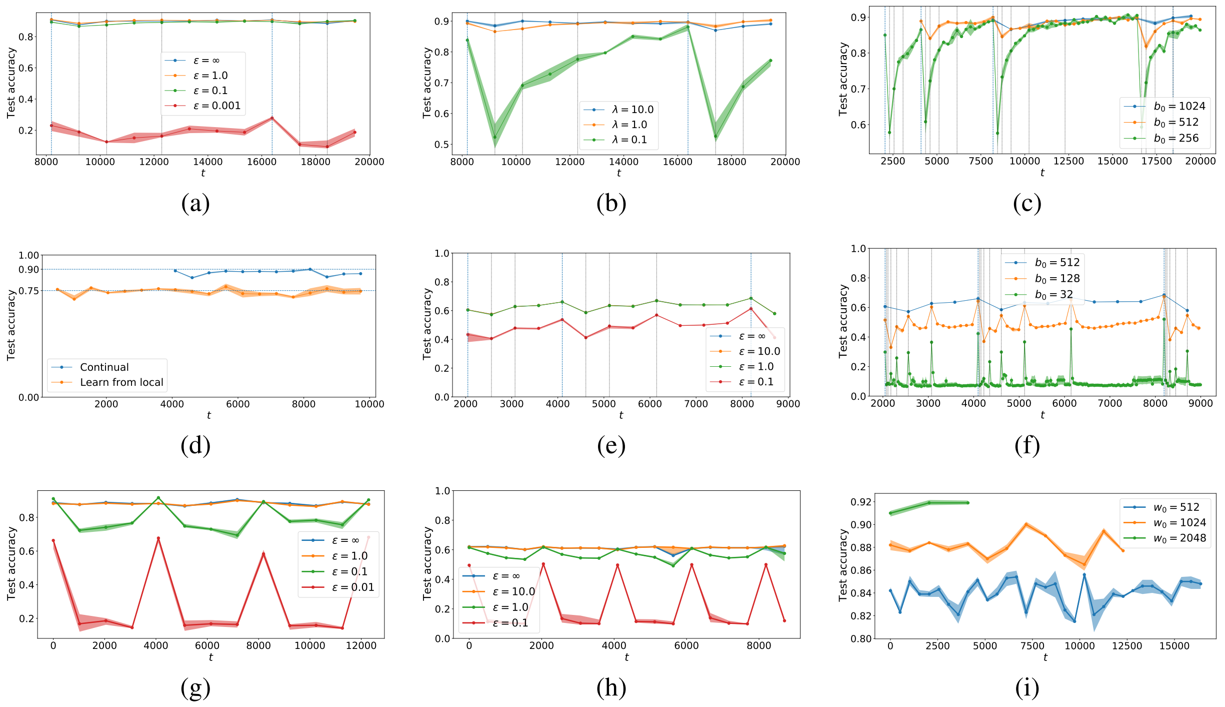


Figure 4: Results for multi-resolution continual release algorithm on MNIST (a, b, c, d) and Arxiv dataset(e, f). **(a, b, c)** $b_0 = 1024$, and $B = b_0 \times 8$, $\epsilon = 0.1$, and $\lambda = 1$. **(d)** Compare with baseline with $b_0 = 512$, $\epsilon = 0.1$ and $\lambda = 1$. **(e, f)** For the Arxiv dataset, $B = 2048$, $b_0 = 512$, $\lambda = 10$, and $\epsilon = 1$. Sliding window algorithm is evaluated in MNIST(**g, i**) and Arxiv(**i**) dataset. For MNIST $\lambda = 1, \epsilon = 1$, and $w_0 = 1024$. Arxiv uses $\lambda = 10, \epsilon = 1$, and $w_0 = 256$. In all cases $w = 7 \times w_0$. The accuracy for a sliding window is reported at the beginning of the window. For MNIST we consider sequence length of 20k, and we don't report accuracy when the window is partially full; Thus some of the results in **(i)** are cut.

for different privacy parameters.

The Lipschitz constant for cross entropy loss is calculated as $\frac{k-1}{2mk}\|X\|$ where k and m are the number of classes and the number of training samples; X is the feature matrix, and the $\|.\|$ represents the Frobenius norm as described in [Yedida, Saha, and Prashanth 2021].

Evaluating the continual release algorithm We evaluate the continual release algorithm in Figure 4 for the MNIST and Arxiv datasets. Both show the natural pattern – with increasing ϵ, the accuracy increases. In the MNIST dataset, the accuracy of the $\epsilon = 0.1, 1$ classifiers match the non-private version. With decreasing b_0 and λ, the accuracy drops.

In Figure 4(d) we also find that our continual release algorithm achieves better test accuracy than a baseline approach where each batch of size b_0 is trained and sanitized independently using Algorithm 4.

Evaluating the sliding window algorithm The sliding window algorithm is evaluated in Figure 4(g-i) for both the datasets. In both the datasets the accuracy increases with increasing ϵ. Both the datasets achieve the non-private accuracy at $\epsilon = 1$. Further with increasing the window size, w_0, the accuracy increases (Figure 4(i)).

Conclusion

There remains a gap between the existing theory of differential privacy and the requirements of practical deployments, such as locally private and evolving dataset scenarios. We addressed the problem of releasing up-to-date private machine learning models for evolving datasets over infinite timescales with only constant privacy loss, focusing on the utility of our models for recently arrived data. We find that the utility of these methods depends on the available update datasize and level of regularization. The theoretical results are validated by empirical experiments.

Limitations. The limitations of our approach include its assumptions of strong convexity and smoothness. The use of output perturbation also requires finding the exact minimizer. It will be ideal to remove these restrictions in future works. Our approach also does not achieve empirical risk minimization for the whole dataset at every step. While at exponentially growing intervals we can release a model that achieves the global ERM, the updates between these releases are only targeted at local data. The algorithm, as described, is also specific to regularized ERM via stochastic gradient descent. While in principle the concepts can work for other techniques, this question remains to be explored by future work.

In future, in addition to investigating the limitations and extensions outlined above, we aim to develop similar algorithms for other machine learning and optimization tasks. This includes the setting of continual and sliding window release for private submodular maximization.

References

Abowd, J. M. 2018. The US Census Bureau adopts differential privacy. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2867–2867.

Agarwal, N.; and Singh, K. 2017. The price of differential privacy for online learning. In *International Conference on Machine Learning*, 32–40. PMLR.

Balle, B.; Barthe, G.; and Gaboardi, M. 2018. Privacy amplification by subsampling: tight analyses via couplings and divergences. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, 6280–6290.

Bassily, R.; Smith, A.; and Thakurta, A. 2014. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, 464–473. IEEE.

Blum, A.; Ligett, K.; and Roth, A. 2013. A learning theory approach to noninteractive database privacy. *Journal of the ACM (JACM)*, 60(2): 1–25.

Chan, T.-H. H.; Shi, E.; and Song, D. 2011. Private and continual release of statistics. *ACM Transactions on Information and System Security (TISSEC)*, 14(3): 1–24.

Chaudhuri, K.; and Monteleoni, C. 2009. Privacy-preserving logistic regression. In *Advances in neural information processing systems*, 289–296.

Chaudhuri, K.; Monteleoni, C.; and Sarwate, A. D. 2011. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(3).

Cummings, R.; Krehbiel, S.; Lai, K. A.; and Tantipongpipat, U. 2018. Differential privacy for growing databases. *arXiv preprint arXiv:1803.06416*.

Datar, M.; Gionis, A.; Indyk, P.; and Motwani, R. 2002. Maintaining stream statistics over sliding windows. *SIAM journal on computing*, 31(6): 1794–1813.

David, S. B.; Lu, T.; Luu, T.; and Pál, D. 2010. Impossibility theorems for domain adaptation. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 129–136. JMLR Workshop and Conference Proceedings.

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

Dwork, C.; Naor, M.; Pitassi, T.; and Rothblum, G. N. 2010. Differential privacy under continual observation. In *Proceedings of the forty-second ACM symposium on Theory of computing*, 715–724.

Guha Thakurta, A.; and Smith, A. 2013. (Nearly) optimal algorithms for private online learning in full-information and bandit settings. *Advances in Neural Information Processing Systems*, 26: 2733–2741.

Hardt, M.; and Rothblum, G. N. 2010. A multiplicative weights mechanism for privacy-preserving data analysis. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, 61–70. IEEE.

Jain, P.; and Thakurta, A. 2013. Differentially private learning with kernels. In *International conference on machine learning*, 118–126. PMLR.

Ji, Z.; Lipton, Z. C.; and Elkan, C. 2014. Differential privacy and machine learning: a survey and review. *arXiv preprint arXiv:1412.7584*.

Kairouz, P.; Mcmahan, B.; Song, S.; Thakkar, O.; Thakurta, A.; and Xu, Z. 2021. Practical and Private (Deep) Learning Without Sampling or Shuffling. In Meila, M.; and Zhang, T., eds., *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, 5213–5225. PMLR.

Kairouz, P.; Oh, S.; and Viswanath, P. 2015. The composition theorem for differential privacy. In *International conference on machine learning*, 1376–1385. PMLR.

Kifer, D.; Smith, A.; and Thakurta, A. 2012. Private convex empirical risk minimization and high-dimensional regression. In *Conference on Learning Theory*, 25–1. JMLR Workshop and Conference Proceedings.

Kim, J. W.; Jang, B.; and Yoo, H. 2018. Privacy-preserving aggregation of personal health data streams. *PloS one*, 13(11): e0207639.

Kuzborskij, I.; and Orabona, F. 2013. Stability and hypothesis transfer learning. In *International Conference on Machine Learning*, 942–950. PMLR.

Mansour, Y.; Mohri, M.; and Rostamizadeh, A. 2008. Domain adaptation with multiple sources. *Advances in neural information processing systems*, 21: 1041–1048.

Mansour, Y.; Mohri, M.; and Rostamizadeh, A. 2009. Domain adaptation: Learning bounds and algorithms.

Redko, I.; Morvant, E.; Habrard, A.; Sebban, M.; and Bennani, Y. 2020. A survey on domain adaptation theory. *arXiv preprint arXiv:2004.11829*.

Task, C.; and Clifton, C. 2012. A guide to differential privacy theory in social network analysis. In *2012 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, 411–417. IEEE.

Upadhyay, J.; and Upadhyay, S. 2020. A Framework for Private Matrix Analysis. *arXiv preprint arXiv:2009.02668*.

Wang, D.; Ye, M.; and Xu, J. 2017. Differentially Private Empirical Risk Minimization Revisited: Faster and More General. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 30*, 2722–2731. Curran Associates, Inc.

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

Yedida, R.; Saha, S.; and Prashanth, T. 2021. LipschitzLR: Using theoretically computed adaptive learning rates for fast convergence. *Applied Intelligence*, 51(3): 1460–1478.