Differentially Private Learning with Small Public Data*

Jun Wang, Zhi-Hua Zhou
National Key Laboratory for Novel Software Technology
Nanjing University, Nanjing 210023, China
{wangj, zhouzh}@lamda.nju.edu.cn

Abstract

Differentially private learning tackles tasks where the data are private and the learning process is subject to differential privacy requirements. In real applications, however, some public data are generally available in addition to private data, and it is interesting to consider how to exploit them. In this paper, we study a common situation where a small amount of public data can be used when solving the Empirical Risk Minimization problem over a private database. Specifically, we propose Private-Public Stochastic Gradient Descent, which utilizes such public information to adjust parameters in differentially private stochastic gradient descent and fine-tunes the final result with model reuse. Our method keeps differential privacy for the private database, and empirical study validates its superiority compared with existing approaches.

Introduction

With the gradual popularization of artificial intelligence and machine learning, privacy is becoming a control concern when mining sensitive data. To gain statistical knowledge while avoiding leakage of personal information, Dwork et al. (2006) presented differential privacy, which gives a mathematically rigorous definition of privacy protection and becomes a standard guarantee nowadays. Many classic learning algorithms have been modified to satisfy its requirements, such as least squares (Wang 2018), SVM (Rubinstein et al. 2009), LASSO (Talwar, Thakurta, and Zhang 2015) and neural networks (Abadi et al. 2016).

Among these algorithms, Empirical Risk Minimization (ERM) is a general framework that minimizes a given loss function over a training set. Combining with different types of regularizers, it covers a lot of specific learning approaches. Many algorithms for the ERM task with a differential privacy guarantee have been developed (Chaudhuri, Monteleoni, and Sarwate 2011; Kifer, Smith, and Thakurta 2012; Bassily, Smith, and Thakurta 2014; Wang, Ye, and Xu 2017; Zhang et al. 2017; Lee and Kifer 2018). However, unlike the usual result about $O\left(1/\sqrt{m}\right)$ excess risk bounds for non-private ERM where m is the size of the training set (Zinkevich 2003), differentially private ERM approaches usually have additional polynomial dependences on dimension p and privacy parameter ϵ (Bassily, Smith, and Thakurta 2014). These extra costs of privacy sometimes make the private ERM become a formidable job for high dimensional datasets or strict privacy requirements.

In many real-world applications, when we perform learning jobs on a private database, we can also get access to a public database about the task. For example, if a headmaster wants to evaluate teaching quality according to private questionnaires from students, some reports from his teaching supervision group which are publicity could be used as well. Besides, a doctor can study typical cases published by the centers for disease control when he analyses the epidemic trend from private medical records in his hospital, a lawyer can search non-private judgments from case information disclosure online when he looks for the patterns of judgments from his customers' information. Thus, a natural problem is, how can we utilize these public data to help differentially private learning.

Some straightforward solutions may immediately come to mind, such as:

- Train a model on the public database at first, and replace it with the model of private data if it is incompetent.
- Train two models on the public and private database respectively, and use their ensemble as the final result.
- Add those public data into the private database and train a model together.

However, in practice, utilizing public data is non-trivial. Because on one hand, the public data usually is not sufficient to train a well-performing model solely; actually, if some domain has a sufficient amount of public data, then private data in such domain would not be highly-valued. This makes the first two solutions impractical. On the other hand, merging public data into the private database may be illegal in cases, and will cause nonnegligible communication costs; needless to say, such a process cannot handle cases where the public data is in accumulating. Thus, the third straightforward way

*This research was supported by the National Key R&D Program of China (2018YFB1004300), NSFC (61921006), and the Collaborative Innovation Center of Novel Software Technology and Industrialization.
Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

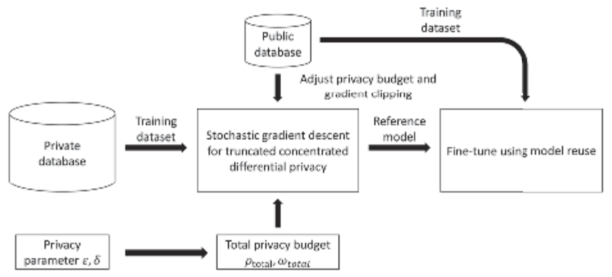
Public database
Training dataset
Adjust privacy budget and gradient clipping
Private database
Training dataset
Stochastic gradient descent for truncated concentrated differential privacy
Reference model
Fine-tune using model reuse
Privacy parameter ε, δ
Total privacy budget $\rho_{total}, \omega_{total}$

Figure 1: The flowchart of PPSGD. PPSGD is a two-stage algorithm to utilize a small amount of public data when solving differentially private ERM. In its first stage, the private database is input to a differentially private SGD, while the public database is used to adjust parameters iteratively to control the noise scale. In its second stage, the final result of the first stage is taken as a pre-trained model, and the public database is used again to fine-tune it.

is not ideal either.

In this paper, we propose Private-Public Stochastic Gradient Descent (PPSGD), which is a general approach to solve differentially private ERM with additional public data. As shown in figure 1, PPSGD consists of two stages: *differentially private stochastic gradient descent* and *model reuse*. To take full advantage of the public database while avoiding overfitting, we use it in the first stage to help allocate the *privacy budget* which controls the rigidness of privacy enforcement, and adjust the *gradient clipping* to enable adaptive gradient rescaling, whereas in the second stage to fine-tune the final result. Since public data does not participate in gradient estimations directly and model reuse ensures its output will not deviate far away from the pre-trained model, we control their influence while improving the overall performance, as well as preserve differential privacy over the private database.

Our contributions can be summarized as follows:

1. We formalize the problem of learning by Empirical Risk Minimization over a private database together with a small public database, which is a common situation and promising field of differentially private approaches.

2. We propose PPSGD, which starts with stochastic gradient descent for truncated concentrated differential privacy, and we dynamically adjust privacy budget and gradient clipping to control noise intensity and individual contribution actively.

3. Experiments over both synthesis datasets and real-world datasets validate the superiority of our method and also manifest a new application of model reuse.

The rest of this paper is organized as follows. We discuss related works and introduce backgrounds in sections 2 and 3 respectively. In section 4, we present our main method. Finally, Section 5 reports the results of empirical study and Section 6 concludes.

Related Work

Differentially private ERM has been studied a lot. Chaudhuri, Monteleoni, and Sarwate (2011) gave two approaches to solve this issue: output perturbation and objective perturbation. Kifer, Smith, and Thakurta (2012) generalized objective perturbation to approximately differential privacy. Zhang et al. (2017) improved output perturbation according to the new stability result of gradient descent. Song, Chaudhuri, and Sarwate (2013) introduced stochastic gradient descent (SGD) to differentially private ERM. Bassily, Smith, and Thakurta (2014) gave an SGD based private algorithm and proved its optimality. Then, many gradient based methods have been proposed, such as Wang, Ye, and Xu (2017) proposed differentially private stochastic variance reduced descent, Lee and Kifer (2018) used adaptive privacy budgets in gradient descent for concentrated differential privacy.

Model reuse focuses on the problem that how can we reuse pre-trained models to help the learning process which typically lacks enough data to train a model directly. It is a key component in *Learnware* (Zhou 2016). Many methods of model reuse have been proposed, such as Li, Tsang, and Zhou (2012) tried to optimize a performance measure by reusing models trained for other performance measures. Tommasi, Orabona, and Caputo (2013) designed a biased regularizer based on the pre-trained models. Segev et al. (2016), Yang et al. (2017), and Wu, Liu, and Zhou (2019) reused forest models, deep models, and multiclass models respectively. Moreover, some theoretical analyses about model reuse have been established in recent years (Kuzborskij and Orabona 2017; Du et al. 2017), which show that model reuse enjoys a fast rate of generalization bound when the pre-trained models are good enough.

To the best of our knowledge, there are a few works that take into account public data in differentially private learning. Papernot et al. (2016) used the private database to train *teacher* models at first, then learned a *student* model with generative adversarial networks by noisy voting among all of the teachers over a public but unlabeled database. Avent et al. (2017) proposed a hybrid model of *local differential privacy*[1] and classic differential privacy. Their approach operates with the hybrid differential privacy model for computing heavy hitters. Feldman et al. (2018) introduced *amplification by iteration*, where the privacy guarantees of elements used in early iterations of an iterative private process are stronger than elements used in subsequent iterations if intermediate results are hidden. Thus, the noise needed to add to an SGD algorithm could be shrunken to protect privacy if the public elements are arranged in the last few iterations. Moreover, their method requires the loss function to be smoothness and indeed ensures local differential privacy.

Preliminaries

Differential privacy, which formulates as definition 1, is based on *neighboring databases*. Let $D = \{d_1, d_2, ..., d_m\}$ be a private database of m elements drawn from universe $\mathcal{D}$.

[1]Local differential privacy (Kasiviswanathan et al. 2008) allows database participants to perturb their data by themselves. It is more strict than differential privacy.

A database D' is a neighbor of D (denote as $D' \sim D$) if $|D'| = m$ and D' has exactly one different element from D.

Definition 1. *(Dwork, Roth, and others 2014) For $\epsilon > 0$ and $\delta \geq 0$, a ramdomized algorithm $\mathcal{A}$ that maps $\mathcal{D}^m$ into some range $\mathcal{S}$ is said to be (ϵ, δ)-differential privacy, if for all neighboring databases $D, D' \in \mathcal{D}^m$ and for any subset $S \subset \mathcal{S}$, we have*

$$\Pr(\mathcal{A}(D) \in S) \leq e^{\epsilon} \Pr(\mathcal{A}(D') \in S) + \delta.$$

To give a meaningful privacy guarantee, one usually sets $\epsilon < 1$ and $\delta \ll 1/m$. When $\delta = 0$, $\mathcal{A}$ is said to be *pure ϵ-differential privacy*. In this paper, we only consider *approximately differential privacy* case, i.e., we suppose $\delta > 0$.

Truncated concentrated differential privacy (tCDP) (Bun et al. 2018) is a variation of differential privacy. We use it in our algorithm to perform composition easily. The definition of tCDP is given in the following, where privacy guarantee is stronger when ρ is smaller or ω is larger.

Definition 2. *(Bun et al. 2018) For $\rho > 0$ and $\omega > 1$, a ramdomized algorithm $\mathcal{A}$ that maps $\mathcal{D}^m$ into some range $\mathcal{S}$ satisfies (ρ, ω)-tCDP, if for all neighboring databases $D, D' \in \mathcal{D}^m$ we have*

$$\forall \alpha \in (1, \omega)\ D_{\alpha}(\mathcal{A}(D) \| \mathcal{A}(D')) \leq \rho\alpha,$$

where $D_{\alpha}(\cdot\|\cdot)$ is the Rényi divergence of order α (Rényi and others 1961).

In this paper, we mainly use the following properties.

Theorem 1. *(Bun et al. 2018) Given an arbitrary algorithm $\mathcal{A}$, the Gaussian mechanism which adds i.i.d. $\mathcal{N}(0, \sigma^2)$ noise to each coordinates of $\mathcal{A}$'s output ensures (ρ, ∞)-tCDP, where $\sigma \geq \Delta_2(\mathcal{A})/\sqrt{2\rho}$, $\Delta_2(\mathcal{A}) = \sup_{D\sim D'} \|\mathcal{A}(D) - \mathcal{A}(D')\|_2$. Among them, $\Delta_2(\mathcal{A})$ is known as L2 sensitivity, it measures the maximal difference of $\mathcal{A}$'s outputs while changing a single element.* [2]

Theorem 2. *(Bun et al. 2018) If an algorithm $\mathcal{A}$ satisties (ρ, ω)-tCDP, for $\delta \geq 1/\exp((\omega - 1)^2\rho)$, $\mathcal{A}$ satisfies (ϵ, δ)-differential privacy with $\epsilon = \rho + 2\sqrt{\rho \log(1/\delta)}$.*

Theorem 3. *(Bun et al. 2018) Let $\mathcal{A}_1 : \mathcal{D}^m \to \mathcal{S}_1$ satisfy (ρ_1, ω_1)-tCDP and $\mathcal{A}_2 : \mathcal{D}^m \times \mathcal{S}_1 \to \mathcal{S}_2$ be such that $\mathcal{A}_2(\cdot, s_1)$ satisfies (ρ_2, ω_2)-tCDP for all $s_1 \in \mathcal{S}_1$. Define $\mathcal{A}_3 : \mathcal{D}^m \to \mathcal{S}_2$ by $\mathcal{A}_3(D) = \mathcal{A}_2(D, \mathcal{A}_1(D))$. Then $\mathcal{A}_3$ satisfies $(\rho_1 + \rho_2, \min(\omega_1, \omega_2))$-tCDP.*

Theorem 4. *(Bun et al. 2018) Let $\mathcal{A}_1 : \mathcal{D}^m \to \mathcal{S}$ satisfy (ρ, ω)-tCDP and $\mathcal{A}_2 : \mathcal{D}^M \to \mathcal{S}$ be such that $\mathcal{A}_2(D) = \mathcal{A}_1(D_S)$ where $D_S \in \mathcal{D}^m$ is sampled uniformly randomly from D. Let $s = m/M$, if $\rho, s \in (0, 0.1]$, $\log(1/s) \geq 3\rho(2 + \log_2(1/\rho))$ and $\omega \geq \log(1/s)/(2\rho)$, then $\mathcal{A}_2$ satisfies $(13s^2\rho, \log(1/s)/(4\rho))$-tCDP.*

PPSGD

In this section, we first provide our settings formally, then propose Private-Public Stochastic Gradient Descent (PPSGD).

[2] Bun et al. proposed a more ingenious Sinh-normal mechanism for tCDP, but here we use the Gaussian mechanism to maintain a fair competition with comparative methods in experiments though our work can be easily generalized to the Sinh-normal mechanism.

Settings and Notations

Let the data universe $\mathcal{D} = \mathcal{X} \times \mathcal{Y}$ where $\mathcal{X} \subset \mathbb{R}^p$ is the instance space and $\mathcal{Y} \subset \mathbb{R}$ is the label set. Given a loss function l, to minimize the expected loss of an unknown distribution $\mathcal{D}_{\mathcal{X}} \times \mathcal{D}_{\mathcal{Y}}$ over $\mathcal{X} \times \mathcal{Y}$, Empirical Risk Minimization solves $\mathbf{w}$ such that

$$\min_{\mathbf{w}\in\mathcal{W}} f(\mathbf{w}, D) = \frac{1}{m} \sum_{i=1}^{m} l(\mathbf{w}, \mathbf{x}_i, y_i), \tag{1}$$

where $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_m, y_m)\}$ is a database drawn from $\mathcal{D}_{\mathcal{X}} \times \mathcal{D}_{\mathcal{Y}}$. We assume $\mathcal{W}$ is a closed convex set, $l(\mathbf{w}, \mathbf{x}_i, y_i)$ is convex and L-Lipschitz for all $\mathbf{x} \in \mathcal{X}, y \in \mathcal{Y}$.

As a private algorithm, we require $\mathbf{w}$ differentially private for D. Besides, we assume a public database $\hat{D} = \{(\hat{\mathbf{x}}_1, \hat{y}_1), (\hat{\mathbf{x}}_2, \hat{y}_2), ..., (\hat{\mathbf{x}}_n, \hat{y}_n)\}$ is available, which is drawn from the same distribution of D. The public database can be used without the concern of privacy, but may be insufficient to solve the objective $\mathbf{w}$ directly.

General Framework

We start by discussing how to utilize public data in differentially private ERM. Specifically, we focus on gradient based iterative algorithms (Bassily, Smith, and Thakurta 2014) which usually performs

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \eta(\nabla f(\mathbf{w}_t, \cdot) + \mathbf{n}_t) \tag{2}$$

for $t = 1, 2, ...T$, where $\nabla f(\mathbf{w}_t, \cdot)$ stands for the gradient and $\mathbf{n}_t$ is a noise vector added to protect privacy. Generally, three aspects of (2) can be improved with public data: getting a more accurate estimator of $\nabla f(\mathbf{w}_t, \cdot)$, minishing the scale of $\mathbf{n}_t$, and increasing T.

Note that public elements are drawn from the same distribution as private elements, adding them into mini-batches could improve the accuracy of gradient estimators definitely. However, since the public database is relatively small, we have to either utilize public information in only a few iterations or use each public element repeatedly. In the former case, public data has little influence on the overall performance. In the latter case, there will be serious overfitting.

Differential privacy is immune to post-processing. That is, we cannot make a differential privacy algorithm's output less differentially private without additional knowledge about the private database. Thus, the public database seems unable to minish the total scale of $\mathbf{n}_t$ dramatically—although we can reduce the sensitivity by adding public elements into mini-batches or shrink the noise by reweighting methods, similar to what happened before, serious overfitting still destroys the overall performance when emphasizing too much on the public database.

However, since noise vectors $\mathbf{n}_t$ are spread in every iteration, as shown in (Lee and Kifer 2018), the performance of iterative private algorithms can be improved significantly by minishing the noise scale in key iterations. Inspired by their approach, we use the public database to control the noise dynamicly by adjusting the following parameters:

- Privacy budget: The gradients usually shrink with the process of learning. Thus, it is reasonable that minishing the

Algorithm 1 PPSGD

```
Input: Private database $\{\mathbf{x}_i, y_i\}_{i=1}^m$, public database $\{\hat{\mathbf{x}}_i, \hat{y}_i\}_{i=1}^n$, loss function $l$, privacy parameters $(\epsilon, \delta)$, learning rate $\eta$, batch size $s$, parameters $\phi, \alpha, \varphi, \beta, \lambda$
Output: Model $\mathbf{w}$
 1: Calculate $(\rho_{total}, \omega_{total})$ according to (3)
 2: Initialize $\mathbf{w}_0, \rho_0$ and $C_0$
 3: $t = 0$
 4: while $(\rho_{total} \geq \rho_t)$ do
 5:     Uniformly randomize $S_t \subset [m]$ with $|S_t| = s$
 6:     $\mathbf{g}_t = \frac{1}{s} \sum_{i \in S_t} \Pi_{|\cdot| \leq C_t} \nabla l(\mathbf{w}_t, \mathbf{x}_i, y_i)$
 7:     Calculate $\rho_t^s, \omega_t^s$ according to (4)
 8:     $\mathbf{g}_t = \mathbf{g}_t + \mathcal{N}\left(0, \frac{2C_t^2}{s^2 \rho_t^s}\right) \mathbf{e}$
 9:     $\mathbf{w}_{t+1} = \mathbf{w}_t - \eta \mathbf{g}_t$
10:     $\rho_{total} = \rho_{total} - \rho_t$
11:     $\mathbf{g}_t^{public} = \frac{1}{n} \sum_{i \in [n]} \nabla l(\mathbf{w}_{t+1}, \hat{\mathbf{x}}_i, \hat{y}_i)$
12:     $mse_t = \frac{2pC_t^2}{s^2 \rho_t^s}$
13:     if $(\phi \cdot \text{norm}(\mathbf{g}_t^{public}) < \sqrt{mse_t})$ then
14:         $\rho_{t+1} = (1 + \alpha)\rho_t$
15:     else
16:         $\rho_{t+1} = \rho_t$
17:     end if
18:     if $(\varphi \cdot \text{norm}(\mathbf{g}_t^{public}) < C_t)$ then
19:         $C_{t+1} = (1 - \beta)C_t$
20:     else
21:         $C_{t+1} = C_t$
22:     end if
23:     $t = t + 1$
24: end while
25: $\mathbf{w}^* = \arg\min_{\mathbf{w}} \frac{1}{n} \sum_{i \in [n]} l(\mathbf{w}, \hat{\mathbf{x}}_i, \hat{y}_i) + \lambda \|\mathbf{w} - \mathbf{w}_t\|^2$
26: return $\mathbf{w}^*$
```

noise scale, i.e., increasing the privacy budget, along with the decreasing of the gradient scale can help the algorithm to converge to a better result.

- Gradient clipping: In existing works, the sensitivity of each iteration of an iterative private algorithm always is regarded as a priori (such as the Lipschitz constant L) or remains unchanged in the learning process. However, since the privacy guarantee must apply to the worst case, outliers that usually do not contribute to the performance could increase the sensitivity enormously. Besides, the gradients of most data elements are decreasing for many loss functions. So, adjusting the gradient clipping threshold, i.e., controlling the sensitivity actively may minish the noise intensity and outliers' influence.

Finally, to increase the number of iterations T, we can utilize public data to fine-tune the result in the end. Since we need not add noise when using public data only, a larger learning rate or number of iterations may be preferred. We use the model reuse formulation where we only need to introduce a single parameter λ.

Algorithm 1 shows detailed steps of PPSGD. In the following, we will focus respectively on three aspects: private SGD, parameter adaptation and model reuse.

Gradient Estimation and Privacy Guarantee

The *while* loop in *line 4* is the first stage of figure 1, i.e., the differentially private SGD. As mentioned before, we use tCDP in our algorithm since its composition property is more straightforward for our dynamic privacy budget allocation. Theorem 2 points out that tCDP is a stronger model than approximately differential privacy. Thus, for given parameters (ϵ, δ), we can solve $(\rho_{total}, \omega_{total})$ with

$$\begin{aligned} \rho_{total} &= \epsilon + 2\log(1/\delta) - 2\sqrt{\log(1/\delta)(\epsilon + \log(1/\delta))} \\ \omega_{total} &= \sqrt{\log(1/\delta)/\rho_{total}} + 1 \end{aligned} \tag{3}$$

in *line 1* and ensure that each $(\rho_{total}, \omega_{total})$-tCDP algorithm satisfies (ϵ, δ)-differential privacy.

In each iteration, according to (2), our primary tasks are estimating $\nabla f(\mathbf{w}_t, \cdot)$ and sampling $\mathbf{n}_t$. For the former task, we randomly select a subset S_t of size s and calculate its average gradient in *line 6*, where

$$\Pi_{|\cdot| \leq C_t} \mathbf{x} = \min(1, C_t / \|\mathbf{x}\|) \cdot \mathbf{x}$$

is a project operator, it ensures the norm of each gradient not greater than the gradient clipping threshold C_t.

Theorem 1, i.e., the Gaussian mechanism of tCDP, states that noise vector $\mathbf{n}_t$ can be sampled from a Gaussian distribution based on its L2 sensitivity and the privacy budget. After the gradient clipping, it's clear that the average gradient has an L2 sensitivity of $2C_t/s$. For another, we allocate ρ_t privacy budget for the t-th iteration, i.e., the t-th update of (2) should satisfies (ρ_t, ω_{total})-tCDP. Note the subtle difference that the privacy guarantee of PPSGD acts on the whole private database while we estimate $\nabla f(\mathbf{w}_t, \cdot)$ on a small subset S_t only. Fortunately, Theorem 4 proposes *amplification of sampling*. That is, in order to ensure (ρ_t, ω_{total})-tCDP before random sampling, we only need to satisfy (ρ_t^s, ω_t^s)-tCDP for S_t where[3]

$$\rho_t^s = \frac{\rho_t}{13(s/m)^2}, \quad \omega_t^s = \frac{\log(m/s)}{2\rho_t^s}. \tag{4}$$

Thus, we can sample $\mathbf{n}_t$ in *line 8* according to Theorem 1.

For the composition between iterations, the following theorem states that PPSGD satisfies the privacy requirement.

Theorem 5. *Given any public database and parameters required in algorithm 1, PPSGD is (ϵ, δ)-differential privacy for the private database.*

Proof. As mentioned before, after adding noise in *line 8*, the t-th execution of *lines 5 to 8* satisfies (ρ_t, ω_{total})-tCDP for the private database. Since *line 10* and the condition of the *while* loop guarantee that $\sum_t \rho_t \leq \rho_{total}$, according to theorem 3, the composition of all executions of *lines 5 to 8* satisfies $(\rho_{total}, \omega_{total})$-tCDP.

At the meanwhile, only *lines 5 to 8* access the private database in algorithm 1. So, according to the post-processing property of differential privacy (theorem 3), we have PPSGD also satisfies $(\rho_{total}, \omega_{total})$-tCDP.

[3]Several conditions are required for amplification of sampling, details are specified in Theorem 4.

Finally, review (3) and theorem 2, PPSGD is (ϵ, δ)-differential privacy for the private database since it satisfies the stronger $(\rho_{total}, \omega_{total})$-tCDP. $\square$

Privacy Budget and Gradient Clipping

As mentioned before, we adjust the privacy budget ρ_t and gradient clipping threshold C_t to control the noise scale. While performing this job, the central role of the public database is to estimate the norm of $\nabla f(\mathbf{w}_t, \cdot)$, which is in *line 11*. Then, we calculate the mean square error caused by the noise, i.e., the dimension p times the variance of the noise in each coordinate. We adjust the privacy budget based on the ratio of gradient's norm and the square root of the mean square error in *line 13*. Our intuition is: when the scale of the noise is much larger than the scale of the gradient, the gradient descent is more likely to be a random walk, and we need to increase the privacy budget to make the optimization process continue meaningful.

We adjust the gradient clipping analogously in *line 18*, where the ratio of gradient's norm and gradient clipping threshold is calculated. Note that the gradient clipping is not performed when we calculate the average gradient in the public database, to avoid the chain reaction that decreases the gradient clipping threshold too fast. Moreover, if we trust that elements in the public database are well-chosen, i.e., there is no outlier in the public database, using the maximum norm of gradients instead of the norm of average gradient in *line 18* is more reasonable since the norm of a certain element's gradient not necessarily decreases with the learning process. Here we choose the norm of average gradient since we randomly select elements in the dataset to form the public database in our experiments.

Model Reuse

To fine-tune the overall result, we perform a simple form of model reuse in *line 25*, corresponding to the second stage in figure 1. We mainly use the idea of biased regularization in our algorithm, whose typical formulation is given by

$$\arg\min_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} l(\mathbf{w}, \mathbf{x}_i, y_i) + \lambda \|\mathbf{w} - \mathbf{w}_0\|^2,$$

where $\mathbf{w}_0$ is a pre-trained model and reused as a biased regularizer to ensure $\mathbf{w}$ will not deviate far away from it.

During model reuse, only the final result of the differentially private SGD and the public database are used. Since we need no additional knowledge about the private database, we can use an arbitrary optimization method without any noise at this stage. What's more, the empirical error in the public database is also a good estimator of the generalization error, due to several fast rate convergence bounds (Kuzborskij and Orabona 2017).

Experiments

In this section, we empirically evaluate the performance of PPSGD and compare it to the following baselines:

1. Differentially private algorithms which merge the public database into private database and train a model together. Specifically, we consider following algorithms:
   (a) *SgdAdv* (Bassily, Smith, and Thakurta 2014) adds equivalent noise in each iteration of an SGD algorithm and applies advanced composition theorem.
   (b) *Agd* (Lee and Kifer 2018) is a gradient descent algorithm for concentrated differential privacy with dynamic adaption of the privacy budget.
   (c) *OutPert* (Zhang et al. 2017) adds Gaussian noise after a non-private gradient descent.
   (d) *ObjPert* (Kifer, Smith, and Thakurta 2012) adds a noise term to its optimization objective, then solves the new objective non-privately.
2. *PNSGD* (Feldman et al. 2018, Corollary 30) arranges public data at the end of the training set and uses amplification by iteration to shrunk noise.
3. *OnlyPub* runs non-private SGD on the public database.
4. *AgdAvg* is the ensemble method takes average over *Agd* and *OnlyPub*.

Also, we denote *NonPriv* to represent the non-private SGD, which does not consider privacy preserving.

We use two loss functions in our experiments: hinge loss and square loss. Hinge loss is formulated as

$$l_{hinge}(\mathbf{w}, \mathbf{x}, y) = \max(0, 1 - y\mathbf{w}^{\mathrm{T}}\mathbf{x}),$$

square loss is formulated as

$$l_{square}(\mathbf{w}, \mathbf{x}, y) = (y - \mathbf{w}^{\mathrm{T}}\mathbf{x})^2.$$

As in most cases, we add an L2 regularizer $\|\mathbf{w}\|_2^2$ to both of the loss functions. Since the regularizer is independent of private data, adding its gradient after added noise is no harm for the privacy guarantee.

For data preprocessing, we normalize each feature into the interval $[-1, 1]$, including the label of regression tasks. To control the sensitivity, we also normalize each sample to a unit norm. Since our experiments are not for parameter selection, we use a non-private SGD to select the coefficient of the regularizer for each dataset. *SgdAdv* and *OutPert* are quite sensitive to their parameters, we run these algorithms in different parameter settings and report the best result. When using the square loss, we estimate the norm of $\mathbf{w}$ non-privately and thus give an upper bound of the gradient for *ObjPert* and *OutPert*. For *Agd* and *SgdAdv*, we set their gradient clipping threshold as a priori. *ObjPert*, *OutPert*, and *PNSGD* have additional requirements on the loss function such as continuous Hessian or smoothness, which are violated by the hinge loss. For simplicity, we ignore these conditions directly. It might overestimate the performance of these algorithms, but without influencing on other approaches including PPSGD.

For both synthesis datasets and real-world datasets, our experiments are repeated for 20 times, and average accuracy or mean square error is presented.[4]

[4]Detailed experimental setups, results, and Matlab codes for PPSGD can be found at http://www.lamda.nju.edu.cn/code_PPSGD.ashx.

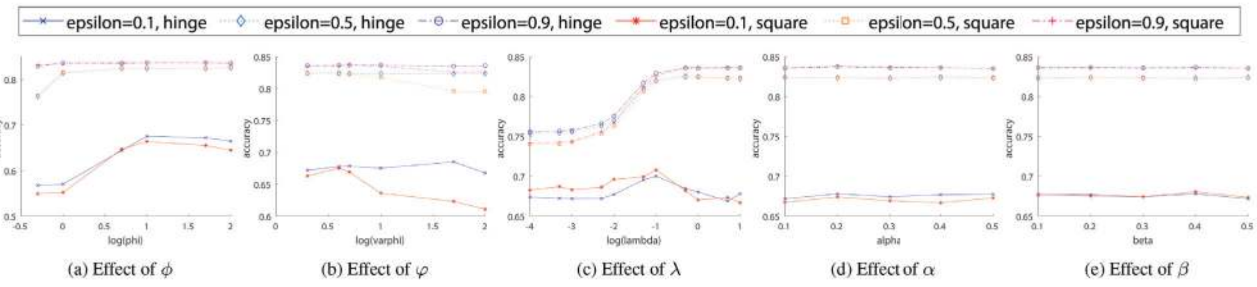
Figure 2: Accuracy comparisons of various parameter settings. Synthesis datasets with 9950 private samples, 50 public samples and 50 dimensions is used.

Synthesis Datasets and Effects of Parameters

There are 5 prime internal parameters in PPSGD. Among them, ϕ and α control the allocation of privacy budget, φ and β control the decreasing of gradient clipping threshold, λ is the weight of the biased regularizer in model reuse. In this subsection, we evaluate the effects of parameters on a toy problem where samples are generated with 100 dimensional Gaussian distribution, each training set has 9950 private samples and only 50 public samples.

Part (a) of figure 2 shows how accuracy changes with various values of ϕ. It's can be found that an extremely small ϕ will increase the privacy budget too fast and degrade the performance. For a relatively large ϵ, the performance is stable with the increasing of ϕ. But a extremely large ϕ hurts performance since the noise conceals the gradient.

The impact of parameter φ is shown in part (b) of figure 2. For hinge loss, we find that φ hardly affect the performance since its gradient is

$$\nabla_{\mathbf{w}} l_{hinge}(\mathbf{w}, \mathbf{x}, y) = \begin{cases} -y\mathbf{x} & 1 > y\mathbf{w}^{\mathrm{T}}\mathbf{x} \\ 0 & \text{otherwise} \end{cases},$$

which means that the gradients of misclassificated samples remain unchanged during the learning process. For square loss, a moderate $\varphi \approx 5$ could enhance the performance because its gradient is

$$\nabla_{\mathbf{w}} l_{square}(\mathbf{w}, \mathbf{x}, y) = 2(\mathbf{w}^{\mathrm{T}}\mathbf{x} - y)\mathbf{x},$$

which usually decreases for majority samples.

As shown in part (c) of figure 2, λ has more delicate effects on the performance. An excessively small λ could lose the performance due to overfitting, while an overlarge λ invalidate model reuse. Different λ is preferred according to the privacy parameter ϵ.

Parts (d) and (e) of figure 2 show that parameters α and β have almost no impact on the performance. Unlike *Agd* (Lee and Kifer 2018), PPSGD adjusts the privacy budget and gradient clipping without privacy cost. Thus, it can adjust parameters successively even meets a small α or β.

When using PPSGD in more situations (such as the real-world datasets in the next subsection), we can find the effects of parameters are relatively robustness for PPSGD. That is, their best settings are similar in various dimensions, training set sizes, and privacy requirements. To sum up, we set $\phi = 10$, $\alpha = \beta = 0.3$, $\varphi = 100$ for hinge loss, $\varphi = 5$ for square loss, and $\lambda \in \{0.01, 0.1, 1\}$. Figure 3 shows the comparisons between PPSGD and baselines. As we can see, the performance of PPSGD is superior to other compared methods with various ϵ.

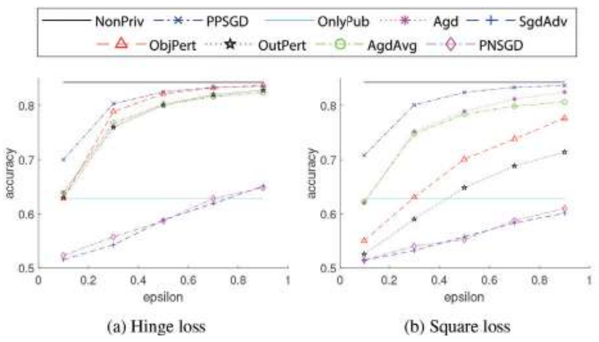
Figure 3: Accuracy comparisons on synthesis dataset.

Real World Datasets

In this subsection, we evaluate the performance of PPSGD on 8 real-world datasets. The specific information of each dataset are represented in table 1. We maintain the parameter settings of last subsection for PPSGD, except set $\varphi = 10$ for square loss when $\epsilon \geq 0.3$. In each experiment, 80 percent of samples are randomly selected to the training set and other samples compose the testing set. Within each (private) training set, we randomly select 0.1 percent (0.01 percent for

Table 1: Characteristics of real-world datasets.

Classification dataset	# Sample	# Feature	% Positive
adult-a	32561	123	24.1
ipums-br	38000	52	50.6
ipums-us	39928	57	51.3
magic04	19020	10	64.8
mini-boo-ne	130064	50	28.1
skin	245057	3	20.8
Regression dataset	**# Sample**	**# Feature**	**Variance**
cadata	20640	8	0.23
stability	10000	12	0.15

Table 2: Performance comparisons on real-world datasets, where the upper part shows results of classification tasks (evaluated by accuracy) and the lower part shows results of regression tasks (evaluated by mean square error). The best private result on each dataset, loss function and ϵ is bolded.

Dataset	Loss funcion	ϵ	PPSGD	SgdAdv	Agd	OutPert	ObjPert	PNSGD	AgdAvg	OnlyPub	NonPriv
adult-a	hinge	0.1	**0.7882**	0.7597	0.7863	0.7645	0.7643	0.7597	0.7872	0.7441	0.8401
		0.5	0.8241	0.7864	0.8225	0.8130	**0.8288**	0.7597	0.8029		
	square	0.1	**0.7941**	0.6842	0.7689	0.6600	0.5332	0.7557	0.7729	0.7706	0.8412
		0.5	**0.8231**	0.7816	0.8229	0.7683	0.6106	0.7598	0.7899		
ipums-br	hinge	0.1	**0.7170**	0.6083	0.7078	0.7056	0.7000	0.5121	0.6862	0.6411	0.7638
		0.5	**0.7588**	0.7137	0.7472	0.7392	0.7559	0.6714	0.6966		
	square	0.1	**0.7112**	0.5784	0.6914	0.5459	0.5293	0.5205	0.6683	0.6363	0.7623
		0.5	**0.7463**	0.7032	0.7443	0.6781	0.6047	0.6367	0.6787		
ipums-us	hinge	0.1	**0.7137**	0.5956	0.7022	0.7119	0.6994	0.5546	0.7036	0.5958	0.7775
		0.5	**0.7701**	0.7167	0.7556	0.7491	0.7668	0.6002	0.7180		
	square	0.1	**0.7192**	0.5828	0.6831	0.5430	0.5363	0.5332	0.6750	0.6254	0.7766
		0.5	**0.7639**	0.7016	0.7517	0.6864	0.6314	0.6330	0.6868		
magic04	hinge	0.1	0.7549	0.6999	**0.7622**	0.7480	0.7577	0.6831	0.7497	0.6981	0.7837
		0.5	**0.7842**	0.7655	0.7771	0.7665	0.7810	0.7331	0.7456		
	square	0.1	**0.7576**	0.6366	0.7563	0.6841	0.6162	0.6425	0.7408	0.7150	0.7762
		0.5	0.7712	0.7555	**0.7724**	0.7506	0.7388	0.7224	0.7499		
mini-boo-ne	hinge	0.1	**0.7685**	0.7371	0.7151	0.7184	0.7551	0.7151	0.7370	0.6950	0.8252
		0.5	0.8045	0.7662	0.7151	0.7584	**0.8049**	0.7151	0.7366		
	square	0.1	**0.7829**	0.7253	0.7255	0.5904	0.5854	0.7153	0.7443	0.7198	0.8248
		0.5	**0.8134**	0.7746	0.7424	0.7199	0.6534	0.7153	0.7589		
skin	hinge	0.1	0.9069	**0.9079**	0.9065	0.9078	0.9061	0.8933	0.9068	0.8697	0.9077
		0.5	0.9072	**0.9078**	0.9069	0.9077	0.9064	0.9071	0.9065		
	square	0.1	**0.9079**	**0.9079**	0.9072	0.9066	0.9071	0.8606	0.9074	0.8529	0.9079
		0.5	0.9077	**0.9079**	0.9074	0.9078	0.9069	0.9048	0.9074		
$\uparrow$ **Accuracy** (the larger the better) | **Mean Square Error** (the smaller the better) $\downarrow$											
cadata	square	0.3	**0.1027**	0.2280	0.1461	0.3058	1.1162	0.2913	0.1253	0.3327	0.0948
		0.7	**0.0996**	0.2079	0.1294	0.2326	0.4943	0.2446	0.1221		
stability	square	0.3	**0.0604**	0.2716	0.0973	0.1007	0.1255	0.4618	0.0912	0.1147	0.0529
		0.7	**0.0558**	0.1008	0.0682	0.0751	0.0656	0.2435	0.0804		

datasets skin and mini-boo-ne) of samples to form the public dataset. That is, the size of each public dataset always is less than 50.

Tables 2 summarizes the results on real-world datasets. Generally, the performance of PPSGD is superior or highly competitive to other compared private approaches. The public data is insufficient, which makes *AgdAvg* and *OnlyPub* behave badly. Compare to simply merge public data into the private database, our framework achieves significant performance gains in most cases.

Note that the performance of *OutPert* and *ObjPert* decreases dramatically when using the square loss, this may because the gradient of square loss is much more difficult to be bounded compared to hinge loss. For *Agd* and *SgdAdv*, we use a gradient clipping threshold much smaller than the Lipschitz constant as a priori, which retains their performance benefits. This illustrates the power of our dynamic gradient clipping, which could minish the noise intensity adaptively. *PNSGD* doesn't perform well in our experiments, although it can utilize public samples directly. This might be due to it only uses each sample in one round, which is necessary to derive its theoretical bounds. Also, we emphasize that PNSGD ensures a kind of local differential privacy, which makes its performance difficult to compete directly with other methods.

Conclusion and Future Works

In this paper, we consider differential privacy learning by Empirical Risk Minimization where a public database could be helpful. We propose a simple yet effective approach named Private-Public Stochastic Gradient Descent (PPSGD). PPSGD includes two main stages, where the public database is used to adjust optimization parameters dynamically in the differentially private SGD and treated as the training set in model reuse. Empirical studies validate that our method can utilize small public data and perform better than existing approaches.

There are several future directions for our setting. Firstly, it is interesting to explore other approaches to utilize public data to improve the performance of differentially private learning. Secondly, the public database may play a greater role in local differential privacy, such as recognizing and resisting outliers caused by the huge noise. Thirdly, some more ingenious model reuse approaches have been proposed recently, which may be helpful to differentially private learning with a wider range of public data.

Acknowledgments

We would like to thank reviewers for helpful comments, and thank Le-Wen Cai, Yu-Jie Zhang for discussions.

References

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

Avent, B.; Korolova, A.; Zeber, D.; Hovden, T.; and Livshits, B. 2017. Blender: Enabling local search with a hybrid differential privacy model. In *Proceedings of the 26th USENIX Security Symposium*, 747–764.

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

Bun, M.; Dwork, C.; Rothblum, G. N.; and Steinke, T. 2018. Composable and versatile privacy via truncated cdp. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, 74–86.

Chaudhuri, K.; Monteleoni, C.; and Sarwate, A. D. 2011. Differentially private empirical risk minimization. *Journal of Machine Learning Research* 12:1069–1109.

Du, S. S.; Koushik, J.; Singh, A.; and Póczos, B. 2017. Hypothesis transfer learning via transformation functions. In *Advances in Neural Information Processing Systems 30*, 574–584.

Dwork, C.; McSherry, F.; Nissim, K.; and Smith, A. 2006. Calibrating noise to sensitivity in private data analysis. In *Proceedings of the 3rd Theory of Cryptography Conference*, 265–284.

Dwork, C.; Roth, A.; et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9(3–4):211–407.

Feldman, V.; Mironov, I.; Talwar, K.; and Thakurta, A. 2018. Privacy amplification by iteration. In *Proceedings of the 59th IEEE Annual Symposium on Foundations of Computer Science*, 521–532.

Kasiviswanathan, S. P.; Lee, H. K.; Nissim, K.; Raskhodnikova, S.; and Smith, A. 2008. What can we learn privately? In *Proceedings of the 49th IEEE Annual Symposium on Foundations of Computer Science*, 531–540.

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

Kuzborskij, I., and Orabona, F. 2017. Fast rates by transferring from auxiliary hypotheses. *Machine Learning* 106(2):171–195.

Lee, J., and Kifer, D. 2018. 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*, 1656–1665.

Li, N.; Tsang, I. W.; and Zhou, Z.-H. 2012. Efficient optimization of performance measures by classifier adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(6):1370–1382.

Papernot, N.; Abadi, M.; Erlingsson, U.; Goodfellow, I.; and Talwar, K. 2016. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755*.

Rényi, A., et al. 1961. On measures of entropy and information. In *Proceedings of the 4th Berkeley Symposium on Mathematical Statistics and Probability*.

Rubinstein, B. I.; Bartlett, P. L.; Huang, L.; and Taft, N. 2009. Learning in a large function space: Privacy-preserving mechanisms for svm learning. *arXiv preprint arXiv:0911.5708*.

Segev, N.; Harel, M.; Mannor, S.; Crammer, K.; and El-Yaniv, R. 2016. Learn on source, refine on target: a model transfer learning framework with random forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39(9):1811–1824.

Song, S.; Chaudhuri, K.; and Sarwate, A. D. 2013. Stochastic gradient descent with differentially private updates. In *Proceedings of the 2013 IEEE Global Conference on Signal and Information Processing*, 245–248.

Talwar, K.; Thakurta, A. G.; and Zhang, L. 2015. Nearly optimal private lasso. In *Advances in Neural Information Processing Systems 28*, 3025–3033.

Tommasi, T.; Orabona, F.; and Caputo, B. 2013. Learning categories from few examples with multi model knowledge transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36(5):928–941.

Wang, D.; Ye, M.; and Xu, J. 2017. Differentially private empirical risk minimization revisited: Faster and more general. In *Advances in Neural Information Processing Systems 30*, 2722–2731.

Wang, Y.-X. 2018. Revisiting differentially private linear regression: optimal and adaptive prediction & estimation in unbounded domain. *arXiv preprint arXiv:1803.02596*.

Wu, X.-Z.; Liu, S.; and Zhou, Z.-H. 2019. Heterogeneous model reuse via optimizing multiparty multiclass margin. In *Proceedings of the 36th International Conference on Machine Learning*, 6840–6849.

Yang, Y.; Zhan, D.-C.; Fan, Y.; Jiang, Y.; and Zhou, Z.-H. 2017. Deep learning for fixed model reuse. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 2831–2837.

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

Zhou, Z.-H. 2016. Learnware: on the future of machine learning. *Frontiers of Computer Science* 10(4).

Zinkevich, M. 2003. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning*, 928–936.