Differentially Private Stochastic Coordinate Descent

Georgios Damaskinos,[1*] **Celestine Mendler-Dünner,**[2*]
Rachid Guerraoui,[1] **Nikolaos Papandreou,**[3] **Thomas Parnell**[3]

[1]École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
[2]University of California, Berkeley
[3]IBM Research, Zurich
georgios.damaskinos@epfl.ch, mendler@berkeley.edu, rachid.guerraoui@epfl.ch, {npo,tpa}@zurich.ibm.com

Abstract

In this paper we tackle the challenge of making the stochastic coordinate descent algorithm differentially private. Compared to the classical gradient descent algorithm where updates operate on a single model vector and controlled noise addition to this vector suffices to hide critical information about individuals, stochastic coordinate descent crucially relies on keeping auxiliary information in memory during training. This auxiliary information provides an additional privacy leak and poses the major challenge addressed in this work. Driven by the insight that under independent noise addition, the consistency of the auxiliary information holds in expectation, we present DP-SCD, the first differentially private stochastic coordinate descent algorithm. We analyze our new method theoretically and argue that decoupling and parallelizing coordinate updates is essential for its utility. On the empirical side we demonstrate competitive performance against the popular stochastic gradient descent alternative (DP-SGD) while requiring significantly less tuning.

1 Introduction

Stochastic coordinate descent (SCD) (Wright 2015) is an appealing optimization algorithm. Compared with the classical stochastic gradient descent (SGD), SCD does not require a learning rate to be tuned and can often have favorable convergence behavior (Fan et al. 2008; Dünner et al. 2018; Ma et al. 2015; Hsieh, Yu, and Dhillon 2015). In particular, for training generalized linear models, SCD is the algorithm of choice for many applications and has been implemented as a default solver in several popular packages such as Scikit-learn, TensorFlow and Liblinear (Fan et al. 2008).

However, SCD is not designed with privacy concerns in mind: SCD builds models that may leak sensitive information regarding the training data records. This is a major issue for privacy-critical domains such as health care where models are trained based on the medical records of patients.

Nevertheless, the low tuning cost of SCD is a particularly appealing property for privacy-preserving machine learning (ML). Hyperparameter tuning costs not only in terms of additional computation, but also in terms of spending the privacy budget.

Our goal is to extend SCD with mechanisms that preserve data privacy, and thus enable the reuse of large engineering efforts invested in SCD (Bradley et al. 2011a; Ioannou, Mendler-Dünner, and Parnell 2019), for privacy-critical use-cases. In this paper we therefore ask the question: *Can SCD maintain its benefits (convergence guarantees, low tuning cost) alongside strong privacy guarantees?*

We employ differential privacy (DP) (Dwork, Roth et al. 2014) as our mathematical definition of privacy. DP provides a formal guarantee for the amount of leaked information regarding participants in a database, given a strong adversary and the output of a mechanism that processes this database. DP provides a fine-grained way of measuring privacy across multiple subsequent executions of such mechanisms and is thus well aligned with the iterative nature of ML algorithms.

The main challenge of making SCD differentially private is that an efficient implementation stores and updates not only the model vector $\boldsymbol{\alpha}$ but also an auxiliary vector $\boldsymbol{v} := \boldsymbol{X}\boldsymbol{\alpha}$ to avoid recurring computations. These two vectors are coupled by the data matrix $\boldsymbol{X}$ and need to be *consistent* for standard convergence results to hold. However, to provide rigorous privacy guarantees, it is vital to add independent noise to both vectors which prohibits this consistency.

Contributions. We present DP-SCD, a differentially private version of the SCD algorithm (Shalev-Shwartz and Zhang 2013b) with formal privacy guarantees. In particular, we make the following contributions.

- We extend SCD to compute each model update as an aggregate of independent updates computed on a random data sample. This parallelism is crucial for the utility of the algorithm because it reduces the amount of the noise per sample that is necessary to guarantee DP.
- We provide the first analysis of SCD in the DP setting and derive a bound on the maximum level of noise that can be tolerated to guarantee a given level of utility.
- We empirically show that for problems where SCD has closed form updates, DP-SCD achieves a better privacy-utility trade-off compared to the popular DP-SGD algorithm (Abadi et al. 2016) while, at the same time, being free of a learning rate hyperparameter that needs tuning. Our implementation is available[1].

---

[1]https://github.com/gdamaskinos/dpscd

2 Preliminaries

Before we dive into the details of making SCD differentially private, we first formally define the problem class considered in this paper and provide the necessary background on SCD and differential privacy.

2.1 Setup

We target the training of Generalized Linear Models (GLMs), the class of models to which SCD is most commonly applied. This class includes convex problems of the following form:

$$\min_{\boldsymbol{\theta}} \mathcal{F}(\boldsymbol{\theta}; \boldsymbol{X}) := \min_{\boldsymbol{\theta}} \left\{ \frac{1}{N} \sum_{i=1}^{N} \ell_i(\boldsymbol{x}_i^\top \boldsymbol{\theta}) + \frac{\lambda}{2} \|\boldsymbol{\theta}\|^2 \right\} \quad (1)$$

The model vector $\boldsymbol{\theta} \in \mathbb{R}^M$ is learnt from the training dataset $\boldsymbol{X} \in \mathbb{R}^{M \times N}$ that contains the N training examples $\boldsymbol{x}_i \in \mathbb{R}^M$ as columns, $\lambda > 0$ denotes the regularization parameter, and ℓ_i the convex loss functions. The norm $\|\cdot\|$ refers to the L_2-norm. For the rest of the paper we use the common assumption that for all $i \in [N]$ the data examples $\boldsymbol{x}_i$ are normalized, i.e., $\|\boldsymbol{x}_i\| = 1$, and that the loss functions ℓ_i are $1/\mu$-smooth. A wide range of ML models fall into this setup including ridge regression and L_2-regularized logistic regression (Shalev-Shwartz and Zhang 2013b).

Threat model. We consider the classical threat model used in (Abadi et al. 2016). In particular, we assume that an adversary has white-box access to the training procedure (algorithm, hyperparameters, and intermediate output) and can have access even to $\boldsymbol{X} \backslash \boldsymbol{x}_k$, where $\boldsymbol{x}_k$ is the data instance the adversary is targeting. However, the adversary cannot have access to the intermediate results of any update computation. We make this assumption more explicit in §3.1.

2.2 Primal-Dual Stochastic Coordinate Descent

The primal SCD algorithm repeatedly selects a coordinate $j \in [M]$ at random, solves a one dimensional auxiliary problem and updates the parameter vector $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}^+ \leftarrow \boldsymbol{\theta} + \boldsymbol{e}_j \zeta^\star \quad \text{where } \zeta^\star = \arg\min_{\zeta} \mathcal{F}(\boldsymbol{\theta} + \boldsymbol{e}_j \zeta; \boldsymbol{X}) \quad (2)$$

where $\boldsymbol{e}_j$ denotes the unit vector with value 1 at position j. Problem (2) often has a closed form solution; otherwise $\mathcal{F}$ can be replaced by its second-order Taylor approximation.

A crucial approach for improving the computational complexity of each SCD update is to keep an auxiliary vector $\boldsymbol{v} := \boldsymbol{X}^\top \boldsymbol{\theta}$ in memory. This avoids recurring computations, as the loss function accesses the data $\boldsymbol{X}$ through the linear map $\boldsymbol{X}^\top \boldsymbol{\theta}$ (see Problem (1)). The auxiliary vector is then updated in each iteration as $\boldsymbol{v}^+ \leftarrow \boldsymbol{v} + \zeta^\star \boldsymbol{x}_j$.

Dual SCD. SCD can be equivalently applied to the dual formulation of Problem (1), commonly referred to as SDCA (Shalev-Shwartz and Zhang 2013b). The dual optimization problem has the following form:

$$\min_{\boldsymbol{\alpha}} \mathcal{F}^*(\boldsymbol{\alpha}; \boldsymbol{X}) := \min_{\boldsymbol{\alpha}} \left\{ \frac{1}{N} \sum_{i=1}^{N} \ell_i^*(-\alpha_i) + \frac{\|X\boldsymbol{\alpha}\|^2}{2\lambda N^2} \right\} \quad (3)$$

where $\boldsymbol{\alpha} \in \mathbb{R}^N$ denotes the dual model vector and ℓ_i^* the convex conjugate of the loss function ℓ_i. Since the dual objective (Problem (3)) depends on the data matrix through the linear map $\boldsymbol{X}\boldsymbol{\alpha}$, the auxiliary vector is naturally defined as $\boldsymbol{v} := \boldsymbol{X}\boldsymbol{\alpha}$ in SDCA. We use the first order optimality conditions to relate the primal and the dual model vectors which results in $\boldsymbol{\theta}(\boldsymbol{\alpha}) = \frac{1}{\lambda N} \boldsymbol{X}\boldsymbol{\alpha}$ and leads to the important definition of the duality gap (Dünner et al. 2016):

$$\mathrm{Gap}(\boldsymbol{\alpha}) := \mathcal{F}^*(\boldsymbol{\alpha}; \boldsymbol{X}) + \mathcal{F}(\boldsymbol{\theta}(\boldsymbol{\alpha}); \boldsymbol{X})$$

$$= \langle \boldsymbol{X}\boldsymbol{\alpha}, \boldsymbol{\theta}(\boldsymbol{\alpha}) \rangle + \frac{\lambda}{2} \|\boldsymbol{\theta}(\boldsymbol{\alpha})\|^2 + \frac{\lambda}{2} \|\boldsymbol{\theta}\|^2 \quad (4)$$

By the construction of the two problems, the optimal values for the objectives match in the convex setting and the duality gap attains zero (Shalev-Shwartz and Zhang 2013b). Therefore, the model $\boldsymbol{\theta}$ can be learnt from solving either Problem (1) or Problem (3), where we use the map $\boldsymbol{\theta}(\boldsymbol{\alpha})$ to obtain the final solution when solving Problem (3).

While the two problems have similar structure, they are quite different from a privacy perspective due to their data access patterns. When applied to the dual, SCD computes each update by processing a single example at a time, whereas the primal SCD processes one coordinate across all the examples. Several implications arise as differential privacy is defined on a per-example basis.

2.3 Differentially Private Machine Learning

Differential privacy is a guarantee for a function f applied to a database of sensitive data (Dwork, Roth et al. 2014). In the context of supervised ML, this function is the update function of the algorithm and the data is typically a set of input-label pairs $(\boldsymbol{x}_i, y_i)$ that are used during model training. Two input datasets are *adjacent* if they differ only in a single input-label pair. Querying the model translates into making predictions for the label of some new input.

Definition 1 (Differential privacy)**.** *A randomized mechanism $\mathcal{M} : \mathcal{D} \to \mathbb{R}$ satisfies (ϵ, δ)-DP if for any two adjacent inputs $d, d' \in \mathcal{D}$ and for any subset of outputs $S \subseteq \mathbb{R}$ it holds that:* $\Pr[\mathcal{M}(d) \in S] \leq e^\epsilon \Pr[\mathcal{M}(d') \in S] + \delta$.

The Gaussian mechanism. The *Gaussian mechanism* is a popular method for making a deterministic function $f : \mathcal{D} \to \mathbb{R}$ differentially private. By adding Gaussian noise to the output of the function we can hide particularities of individual input values. The resulting mechanism is defined as: $\mathcal{M}(d) := f(d) + \mathcal{N}(0, S_f^2 \sigma^2)$ where the variance of the noise needs to be chosen proportional to the sensitivity S_f of the function f. This definition can be readily extended to the multi-dimensional case in order to fit the general ML setting: An iterative ML algorithm can be viewed as a function $\boldsymbol{f} : \mathbb{R}^{M \times N} \to \mathbb{R}^M$ that repeatedly computes model updates from the data and thus requires the addition of multi-dimensional noise at each iteration t:

$$\boldsymbol{\mathcal{M}}_t(\boldsymbol{d}) = \boldsymbol{f}(\boldsymbol{d}) + \mathcal{N}(0, S_f^2 \sigma^2 \boldsymbol{I}), \ \boldsymbol{I} \in \mathbb{R}^{M \times M} \quad (5)$$

where $\boldsymbol{I}$ denotes the identity matrix and the sensitivity is:

$$S_f := \max_{\text{adjacent } \boldsymbol{d}, \boldsymbol{d}'} \|\boldsymbol{f}(\boldsymbol{d}) - \boldsymbol{f}(\boldsymbol{d}')\| \quad (6)$$

Algorithm 1: DP-SCD (for Problem (3))

Input: N examples $\boldsymbol{x}_i \in \mathbb{R}^M$ and the corresponding
labels y_i, T: # iterations, L: mini-batch size,
(ϵ, δ): DP parameters, C: scaling factor

1 **Init:** $\boldsymbol{\alpha} = \boldsymbol{0}$; $\boldsymbol{v} = \boldsymbol{0}$; shuffle examples $\boldsymbol{x}_i$
2 $\sigma \leftarrow$ *smallest noise magnitude, s.t.,* MA $\left(\delta, \sigma, \frac{L}{N}, T\right) = \epsilon$
3 **for** $t = 1, 2, ...T$ **do**
4     $\Delta\boldsymbol{\alpha} = 0$; $\Delta\boldsymbol{v} = 0$
5   Sample a mini-batch $\mathcal{B} \subset [N]$ of L examples
6   **for** $j \in \mathcal{B}$ **do**
7        $\zeta_j = \arg\min_\zeta \mathcal{G}_j^*(\alpha_j, \zeta, \boldsymbol{v}, \boldsymbol{x}_j)$
8        $\zeta_j \mathrel{/{=}} \max\left(1, \frac{|\zeta_j|}{C}\right)$ // scale
9        $\Delta\boldsymbol{\alpha} \mathrel{+{=}} \boldsymbol{e}_j\zeta_j$
10       $\Delta\boldsymbol{v} \mathrel{+{=}} \zeta_j\boldsymbol{x}_j$
11   **end**
       // update the model
12     $\boldsymbol{\alpha} \mathrel{+{=}} \Delta\boldsymbol{\alpha} + \boldsymbol{e}_\mathcal{B}(\mathcal{N}(0, \sigma^2 2C^2 \boldsymbol{I}_L))$, $\boldsymbol{I}_L \in \mathbb{R}^{L \times L}$
13     $\boldsymbol{v} \mathrel{+{=}} \Delta\boldsymbol{v} + \mathcal{N}(0, \sigma^2 2C^2 \boldsymbol{I}_M)$, $\boldsymbol{I}_M \in \mathbb{R}^{M \times M}$
14 **end**
15 **return** $\boldsymbol{\theta} = \frac{1}{\lambda N}\boldsymbol{v}$ // primal DP model

---

DP-SGD. As an application of the Gaussian mechanism, we consider stochastic gradient descent (SGD) used for solving Problem (1). SGD iteratively updates the model as $\boldsymbol{\theta}^+ \leftarrow \boldsymbol{\theta} - \eta\boldsymbol{g}_\xi$ with $\boldsymbol{g}_\xi := \frac{1}{|\xi|}\sum_{i \in \xi}\nabla\ell_i(\boldsymbol{x}_i^\top\boldsymbol{\theta})\boldsymbol{x}_i + \lambda\boldsymbol{\theta}$ being the gradient computed on a dataset sample ξ (also know as *mini-batch*), and η denoting the learning rate. According to the Gaussian mechanism, DP-SGD adds noise to each update $\boldsymbol{g}_\xi$ with the variance given by the sensitivity of the update function, i.e., $S_f = \|\boldsymbol{g}_\xi\|$ for our example. In practice, an additional gradient clipping step enforces a desired bound on the sensitivity S_f (Abadi et al. 2016).

Privacy accounting. Measuring the privacy leakage of a randomized mechanism $\mathcal{M}$ boils down to computing (ϵ, δ). More precisely, computing a bound for the privacy loss ϵ that holds with probability $1 - \delta$. In the context of ML, $\mathcal{M}$ often consists of a sequence of mechanisms $\mathcal{M}_t$ that, for example, denote the model update at each iteration t. All these mechanisms have the same pattern in terms of sensitive (training) data access for most iterative ML algorithms, including SCD. Computing (ϵ, δ) given the individual pairs (ϵ_t, δ_t) is a problem known as composability. However, standard composition theorems often provide loose bounds on (ϵ, δ) and methods such as the *moments accountant* (Abadi et al. 2016) are useful tools to compute tighter bounds for the overall privacy loss. The moments accountant method is tailored to the Gaussian mechanism and employs the log moment of each $\mathcal{M}_t$ to derive the bound of the total privacy loss. It can be viewed as a function that returns the privacy loss bound:

$$\epsilon = \text{MA}(\delta, \sigma, q, T) \tag{7}$$

where σ is the noise magnitude, q is the sampling ratio (i.e., the ratio of the data that each $\mathcal{M}_t$ uses over the total data), and T is the number of individual mechanisms $\mathcal{M}_t$.

3 Differentially Private Stochastic Coordinate Descent

We first focus on Problem (3) and discuss the primal problem in §3.2. Algorithm 1 summarizes our differentially private SCD algorithm (DP-SCD). The crucial extension in comparison with standard SDCA is that we consider updates that independently process a mini-batch $\mathcal{B}$ of L coordinates sampled uniformly at random in each inner iteration (Steps 6-11). This is not only beneficial from a performance perspective, as the updates can be executed in parallel, but it also serves as a hyperparameter that can improve the privacy-utility trade-off of our algorithm (similar to the lot size in (Abadi et al. 2016)). The subproblem formulation $\mathcal{G}_j^*$ in Step 7 is inspired by the distributed coordinate descent method of Ma et al. (2015). In particular, we reuse the local subproblem formulation of their method for the case where each parallel step updates only a single $j \in \mathcal{B}$:

$$\mathcal{G}_j^*(\alpha_j, \zeta, \boldsymbol{v}, \boldsymbol{x}_j) := \frac{1}{N}\ell_j^*(-\alpha_j - \zeta)$$
$$+ \frac{1}{2\lambda N^2}\left(\|\boldsymbol{v}\|^2 + 2\boldsymbol{x}_j^\top\boldsymbol{v}\zeta + L\|\boldsymbol{x}_j\|^2\zeta^2\right) \tag{8}$$

In each iteration $t > 0$ we then minimize Equation (8) over ζ for all $j \in \mathcal{B}$ to obtain the respective coordinate updates ζ_j. This minimization can often be computed in closed form, e.g., for ridge regression, or SVMs. However, approximate solutions are sufficient for convergence, e.g., as specified in (Smith et al. 2017, Assumption 1). For logistic regression we use a single Newton step to approximate the coordinate update ζ_j.

Finally, to bound the sensitivity of each update step we rescale ζ_j to have magnitude no larger than C (Step 8). We can then use the Gaussian mechanism to make $\boldsymbol{\alpha}$ and $\boldsymbol{v}$ differentially private (Steps 12-13). We address two main questions regarding DP-SCD:

1. How much noise do we need to guarantee (ϵ, δ)-DP?
2. Can we still give convergence guarantees for this new algorithm under the addition of noise?

We answer the first question in §3.1 by analyzing the sensitivity of our update function with respect to $(\boldsymbol{\alpha}, \boldsymbol{v})$. For the second question, the main challenge is that after the addition of noise the consistency between $\boldsymbol{\alpha}$ and $\boldsymbol{v}$, i.e., $\boldsymbol{v} = \boldsymbol{X}\boldsymbol{\alpha}$, no longer holds. In §4 we show how to address this challenge and prove convergence for our method. We provide the cost analysis and implementation details related to DP-SCD in Appendix A.1 and Appendix A.2 respectively[2].

3.1 Privacy Analysis

We view the training procedure of Algorithm 1 as a sequence of mechanisms $\mathcal{M}_t$ where each mechanism corresponds to one outer iteration (Steps 3-14) and computes an update on a mini-batch of L examples. We assume these mechanisms to be atomic from an adversary point of view, i.e., we assume no access to the individual coordinate updates within

---
[2]The full version of this paper (Damaskinos et al. 2020) includes our technical Appendix.

the mini-batch $\mathcal{B}$. The update computation (Steps 6-11) before the noise addition, corresponds to $\boldsymbol{f}$ according to Equation (5). For determining the sensitivity of $\boldsymbol{f}$ it is important to note that all updates within a mini-batch $\mathcal{B}$ touch different data points and are computed independently. The output of each mechanism $\mathcal{M}_t$ is the concatenation $[\boldsymbol{\alpha}; \boldsymbol{v}]$ of the updated model vector $\boldsymbol{\alpha}$ and the updated auxiliary vector $\boldsymbol{v}$. The sensitivity of this mechanism is given as follows.

Lemma 1 (Sensitivity of DP-SCD). *Assume the columns of the data matrix $\boldsymbol{X}$ are normalized. Then, the sensitivity of each mini-batch update computation (Steps 6-11 in Algorithm 1) is bounded by: $S_f \leq \sqrt{2}C$.*

Proof Sketch. The sensitivity depends on $\boldsymbol{f}(\boldsymbol{X})$, $\boldsymbol{f}(\boldsymbol{X}')$ that differ solely due to the missing example for $\boldsymbol{X}'$, as the updates for a given mini-batch are independent. The bound follows from Step 8, the triangle inequality and the assumption of normalized data. The full proof can be found in Appendix A.3. $\square$

Theorem 1 (Privacy bound for DP-SCD). *Algorithm 1 is (ϵ, δ)-differentially private for any $\epsilon = \mathcal{O}\left(q^2 T\right)$ and $\delta > 0$ if we choose $\sigma = \Omega\left(\frac{1}{\epsilon} q \sqrt{T \log(1/\delta)}\right)$.*

Proof. Each mechanism $\mathcal{M}_t$ is made differentially private by using the Gaussian mechanism from Equation (5). The output dimension is $M+L$ in our case. The moments of each mechanism $\mathcal{M}_t$ are bounded given Lemma 1 and (Abadi et al. 2016, Lemma 3). Hence, based on (Abadi et al. 2016, Theorem 1), we derive the proposed lower bound on σ that guarantees (ϵ, δ)-DP for the output model. $\square$

3.2 Primal Version

The primal formulation of DP-SCD (Algorithm 2 in Appendix B) computes the updates in a coordinate-wise manner, thus making differentially private learning more challenging than in the dual formulation. For each update computation, the primal version accesses a single coordinate j across all examples. This means the sampling ratio q for each mechanism $\mathcal{M}_t$ is equal to 1 and the number of parallel updates (L) is no longer an effective parameter to regulate the privacy-utility trade-off. Additionally, using the Gaussian mechanism, the sensitivity bound, and thus the noise addition necessary for DP, is significantly larger (as $S_f \leq 2C\sqrt{L(L+1)}$) compared to the dual version (see Lemma 4 in Appendix B). In conclusion, the dual version (Algorithm 1) is preferable over the primal.

4 Convergence Analysis

We recall that the main challenge for generalizing the convergence guarantees of SCD to DP-SCD, is how to handle potential inconsistencies between the auxiliary vector $\boldsymbol{v}$ and the model vector $\boldsymbol{\alpha}$. Note that a variant of Algorithm 1 that only updates $\boldsymbol{\alpha}$ and recomputes $\boldsymbol{v}$ in every iteration would overcome this issue. However, such a variant involves several disadvantages that make it highly impractical. First it involves a significant computational overhead. Second it makes the mini-batch size ineffective as a tuning parameter,

as the sampling ratio (q in Equation (7)) would be 1 for each update. Third, on the final step it would need to employ the entire dataset to map the dual model to the primal model ($\boldsymbol{\theta} := \frac{1}{\lambda N} \boldsymbol{X} \boldsymbol{\alpha}$), which creates a massive privacy leakage. All these can be avoided by maintaining the auxiliary vector and computing the final primal model as $\boldsymbol{\theta} := \frac{1}{\lambda N} \boldsymbol{v}$.

To analyze the convergence of Algorithm 1 we split each mini-batch iteration in two steps: an update step and a perturbation step. We denote the privacy preserving model sequence by $\{\boldsymbol{\alpha}_t\}_{t>0}$ and the intermediate, non-public models before the perturbation step by $\{\hat{\boldsymbol{\alpha}}_t\}_{t>0}$. We use the same notation for the corresponding $\boldsymbol{v}$ sequences. The *update step* includes the computation of L coordinate updates (i.e., $\hat{\boldsymbol{\alpha}}_{t-1} = \boldsymbol{\alpha}_{t-1} + \Delta \boldsymbol{\alpha}_t$ and $\hat{\boldsymbol{v}}_{t-1} = \boldsymbol{v}_{t-1} + \Delta \boldsymbol{v}_t$) and the *perturbation step* adds Gaussian noise to the two vectors $\boldsymbol{\alpha}$ and $\boldsymbol{v}$ independently (i.e., $\boldsymbol{\alpha}_t = \hat{\boldsymbol{\alpha}}_{t-1} + \boldsymbol{\eta}_\alpha$ and $\boldsymbol{v}_t = \hat{\boldsymbol{v}}_{t-1} + \boldsymbol{\eta}_v$, where $\boldsymbol{\eta}$ denotes the Gaussian noise). Hence, we get the following sequence:

$$\ldots \rightarrow \{\boldsymbol{\alpha}_{t-1}, \boldsymbol{v}_{t-1}\} \overset{\text{(update)}}{\rightarrow} \{\hat{\boldsymbol{\alpha}}_{t-1}, \hat{\boldsymbol{v}}_{t-1}\} \overset{\text{(perturb)}}{\rightarrow} \{\boldsymbol{\alpha}_t, \boldsymbol{v}_t\} \rightarrow \ldots$$

Our approach is to show that the update step decreases the objective even if the update is computed based on a noisy version of $\boldsymbol{\alpha}, \boldsymbol{v}$ and the amount of decrease is larger than the damage caused by adding noise in the perturbation step. The key observation that allows us to derive convergence guarantees in this setting is the following.

Remark 1 (Consistency in expectation). *Given the construction of the model updates and the independent noise with zero mean that is added to both sequences, Algorithm 1 preserves the consistency between $\boldsymbol{\alpha}$ and $\boldsymbol{v}$ in expectation, i.e., $\mathbb{E}[\boldsymbol{v}] = \boldsymbol{X}\mathbb{E}[\boldsymbol{\alpha}]$.*

4.1 Update Step

Each iteration of Algorithm 1 computes a mini-batch update $\Delta \boldsymbol{\alpha}$ that is applied to the model $\boldsymbol{\alpha}$ and indirectly to the auxiliary vector $\boldsymbol{v}$ in Steps 12 and 13, respectively. We denote by $\Delta \boldsymbol{\alpha}^{\text{tmp}}$ the unscaled version of this update, i.e., the update computed excluding Step 8. We add this step back later in our analysis. Lemma 2 gives a lower bound for the decrease in the objective achieved by performing this update even if $\Delta \boldsymbol{\alpha}^{\text{tmp}}$ is computed based on noisy versions of $\boldsymbol{\alpha}, \boldsymbol{v}$ where $\mathbb{E}[\boldsymbol{v}] = \mathbb{E}[\boldsymbol{X}\boldsymbol{\alpha}]$ but $\boldsymbol{v} \neq \boldsymbol{X}\boldsymbol{\alpha}$.

Lemma 2 (Update step - objective decrease lower bound). *Assuming ℓ_i are $1/\mu$-smooth, then the update step of Algorithm 1 decreases the objective, even if computed based on a noisy version of $\boldsymbol{\alpha}, \boldsymbol{v}$. The decrease is lower-bounded as follows:*

$$\mathbb{E}[\mathcal{S}(\boldsymbol{\alpha}) - \mathcal{S}(\boldsymbol{\alpha} + \Delta \boldsymbol{\alpha}^{\text{tmp}})] \geq \frac{\mu \lambda L}{\mu \lambda N + L} \mathbb{E}[\mathcal{S}(\boldsymbol{\alpha})] \quad (9)$$

where $\mathcal{S}$ denotes the dual suboptimality defined as: $\mathcal{S}(\boldsymbol{\alpha}) := \mathcal{F}^*(\boldsymbol{\alpha}; \boldsymbol{X}) - \min_{\boldsymbol{\alpha}} \mathcal{F}^*(\boldsymbol{\alpha}; \boldsymbol{X})$.

Proof Sketch. We build on (Ma et al. 2015, Lemma 3) to relate the decrease of each parallel update computed on the subproblems $\mathcal{G}_j^*$ to the global function decrease. Then, we take expectation w.r.t the randomization of the noise and proceed along the lines of (Ma et al. 2015, Lemma 5) to

involve the duality gap in our analysis. Finally, based on an inequality for the duality gap and Remark 1 we arrive at the bound stated in Lemma 2. Note that with $L = 1$ we recover the classical result of SDCA (Shalev-Shwartz and Zhang 2013b) for the sequential case. The full proof can be found in Appendix D.1. □

Incorporating update scaling. When computing the update $\Delta\alpha$ in Algorithm 1, each coordinate of $\Delta\alpha^{\text{tmp}}$ is *scaled* to a maximum magnitude of C (Step 8) in order to bound the sensitivity of each update step. In strong contrast to SGD, where this scaling step destroys the unbiasedness of the gradients and thus classical convergence guarantees no longer hold, for DP-SCD the scaling only translates into a smaller function decrease. This is a remarkable property of SCD when analyzed in the DP setting.

To incorporate scaling into our analysis we use the following inequality which is guaranteed to hold for some $\kappa \in [0, 1)$ due to the convexity of the objective.

$$\mathcal{S}(\alpha + \Delta\alpha) \leq (1 - \kappa)\mathcal{S}(\alpha + \Delta\alpha^{\text{tmp}}) + \kappa\mathcal{S}(\alpha)$$
$$\Leftrightarrow \mathbb{E}[\mathcal{S}(\alpha) - \mathcal{S}(\alpha + \Delta\alpha)] \geq$$
$$(1 - \kappa)\mathbb{E}[\mathcal{S}(\alpha) - \mathcal{S}(\alpha + \Delta\alpha^{\text{tmp}})] \qquad (10)$$

The scaling step preserves the linear convergence of Lemma 2 and decreases the lower-bound on the RHS of Inequality (9) by a factor of $(1 - \kappa)$. Note that for $\kappa = 0$ (i.e., no scaling) the solution is exact, and the smaller the scaling factor C, the larger the κ.

4.2 Perturbation Step

To derive a utility guarantee for DP-SCD, it remains to show that adding noise at the end of each mini-batch update does not increase the objective more than the decrease achieved by the rescaled update $\Delta\alpha$.

Lemma 3 (Perturbation step - objective increase upper bound)**.** *The perturbation step of Algorithm 1 increases the objective by at most:*

$$\mathbb{E}[|\mathcal{S}(\alpha + \Delta\alpha + \eta) - \mathcal{S}(\alpha + \Delta\alpha)|] \leq \frac{L\sigma^2}{2\lambda N^2} \qquad (11)$$

Proof. Given the L_2 regularization, $\mathcal{F}$ is λ-strongly convex for any convex ℓ_i. By reusing the Fenchel-Rockafellar duality pair presented by Dünner et al. (2016) (see Equation (A)-(B) in their work) and incorporating the $1/N$ rescaling of Problem (3), we get that $\mathcal{F}^*$ is $\frac{1}{\lambda N^2}$-smooth. We thus have $\mathcal{F}^*(\alpha' + \eta) \leq \mathcal{F}^*(\alpha') + \eta^\top\nabla\mathcal{F}^*(\alpha') + \frac{1}{2\lambda N^2}\|\eta\|^2$. Setting $\alpha' = \alpha + \Delta\alpha$, subtracting $\min_a \mathcal{F}^*(\alpha)$ on both sides and taking expectations w.r.t the randomness in the perturbation noise, the claim follows from $\mathbb{E}[\eta] = 0$ and $\mathbb{E}[\|\eta\|^2] = L\sigma^2$. □

Finally, we combine the different steps from our analysis (Lemma 2, Lemma 3, and Inequality (10)) with the privacy bound from Theorem 1 and derive our main result stated in Theorem 2. The proof can be found in Appendix D.2.

Theorem 2 (Utility guarantee for Algorithm 1)**.** *Suppose that ℓ_i is convex and $1/\mu$-smooth $\forall i$. If we choose L, C such that $L(2(1 - \kappa)\mu\lambda - 1) > \mu\lambda N$ for $\kappa \in (0, 1)$, and T such*

that $T = \mathcal{O}\left(\log\left(\frac{\lambda N^4\epsilon^2}{L^3 \log(1/\delta)}\right)\right)$, then the suboptimality of Algorithm 1 is bounded as:

$$\mathbb{E}[\mathcal{S}(\alpha^{(T)})] \leq \mathcal{O}\left(\frac{L^3}{\lambda N^4\epsilon^2}\log\left(\frac{\lambda N\epsilon}{L}\right)\log\left(\frac{1}{\delta}\right)\right) \quad (12)$$

where L/N is the sampling ratio q.

The suboptimality is proportional to the magnitude of the noise and hence finding the exact minimizer requires $\sigma \to 0$ (i.e., $\epsilon \to \infty$). The smaller the σ the larger the ϵ and thus the less private the learning is. We empirically confirm that DP-SCD converges smoother with a smaller σ in §5.

Noteworthy, there is no lower bound on the suboptimality. DP-SCD converges to a ball around the optimum and fluctuates randomly within this ball. The radius of this ball depends on σ and constitutes our utility loss upper bound (Theorem 2). The utility is equal to this radius when the function decrease (Lemma 2 after incorporating update scaling) is equal to the increase by the perturbation bound (Lemma 3).

Theorem 2 constitutes the first analysis of coordinate descent in the differentially private setting and we hope it serves as a stepping stone for future theoretical results.

5 Experiments

Our empirical results compare our new DP-SCD algorithm against SCD, SGD and DP-SGD. We also include SEQDP-SCD as a baseline, to depict the importance of independent updates within a given mini-batch. SEQDP-SCD adopts the natural SCD method of performing sequential (and thus correlated) updates. We prove in Appendix C that the update correlation drives the sensitivity to $S_f \leq 2C\sqrt{L(L+1)}$ and thus makes the noise addition significantly larger than the one for DP-SCD.

We test the performance on three popular GLM applications, namely ridge regression, logistic regression and L_2-regularized SVMs[3]. We provide detailed information regarding our setup in Appendix E. In particular, we describe the datasets (YearPredictionMSD, Phishing, Adult), the metrics, the values for each hyperparameter, and the deployment setup of our implementation.

5.1 Results

Tuning cost. The hyperparameters of SEQDP-SCD, DP-SCD, SGD and DP-SGD are $\{L, C\}$, $\{L, C\}$, $\{\eta, |\xi|\}$ and $\{\eta, L, C\}$ respectively; SCD requires no tuning. We tested a total of 156 configurations for DP-SCD as opposed to a total of 2028 for DP-SGD. For large-scale datasets that require significant amount of training time and resources, the difference in the number of hyperparameters constitutes an appealing property of DP-SCD. Noteworthy, the SGD tuning is not useful for the DP-SGD tuning, as the best choice for η depends on the choice of C (Thakkar, Andrew, and McMahan 2019).

---

[3]The hinge loss is not smooth and thus SVMs do not fall into the setup of our convergence analysis. Therefore, our experiments regarding SVMs illustrate that our convergence regime may not be tight and further function classes may be worth exploring.

(a) Ridge regression (YearPredictionMSD)   (b) Logistic regression (Phishing)   (c) SVMs (Adult)
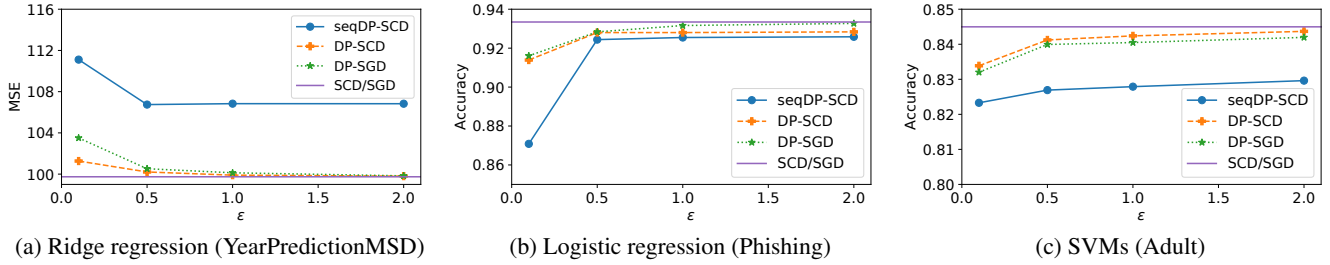
Figure 1: Privacy-utility trade-off. Better utility means lower MSE or larger accuracy. DP-SCD outperforms DP-SGD for the applications that enable exact update steps (namely ridge regression and SVMs).

Privacy-utility trade-off. Figure 1 quantifies the trade-off between privacy and utility for different privacy levels, i.e., different ϵ values under a fixed $\delta = 0.001$ (Wang, Ye, and Xu 2017; Zhang et al. 2017). We observe that SEQDP-SCD has the worst performance due to the significantly larger noise. DP-SCD performs better than DP-SGD for ridge regression and SVMs, and worse for logistic regression, as the steps of DP-SCD in the case of ridge regression and SVMs are more precise despite suffering more noise than DP-SGD. Moreover, DP-SCD requires $\sqrt{2}$ more noise than DP-SGD (for the same privacy guarantee) due to the need of a shared vector (§3). However, each update of DP-SCD finds an exact solution to the minimization problem for ridge regression and SVMs (and an approximate one for logistic regression), whereas DP-SGD takes a direction opposite to the gradient estimate. In SGD, we need to often be conservative (e.g., choose a small learning rate) in order to account for bad estimates (due to sampling) of the actual gradient.

Convergence. Figure 2 shows the impact of noise on the convergence behavior on the YearPredictionMSD dataset. In particular, for a given ϵ, we select the best (in terms of validation MSE) configuration (also used in Figure 1(a)), and measure the decrease in the objective with respect to epochs (not time as that would be implementation-dependent). We empirically verify the results of Theorem 2 by observing that the distance between the convergence point and the optimum depends on the level of privacy. Moreover, DP-SCD and DP-SGD converge with similar speed for $\epsilon = 0.1$, but to a different optimum (as also shown in Figure 1(a)). Decreasing the amount of noise ($\epsilon = 1$), makes DP-SCD converge almost as fast as SGD and with more stability compared to $\epsilon = 0.1$. This is aligned with the results of §4, i.e., the fact that the larger amount of noise (decrease in ϵ) makes the decrease in the suboptimality more noisy.

Mini-batch size and scaling factor interplay. DP-SCD involves two hyperparameters, namely the mini-batch size (L) and the scaling factor (C). Both hyperparameters affect the privacy-utility trade-off while the mini-batch size also controls the level of parallelism (§3). Figure 3 shows the impact of the choice of L, C to the utility for the YearPredictionMSD dataset with $\epsilon = 1$. For the given setup, the best (i.e., leading to the lowest MSE) value for C is 0.5. We observe that for this value the curve also flattens out, i.e., there is a wide range of values for L that achieve MSE close to the
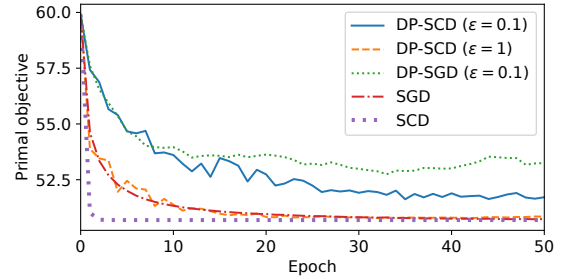

Figure 2: Impact of noise on convergence. Differential privacy does not prevent convergence but makes the objective reduction more noisy while also increases the distance to the optimum (aligned with the result of Theorem 2).
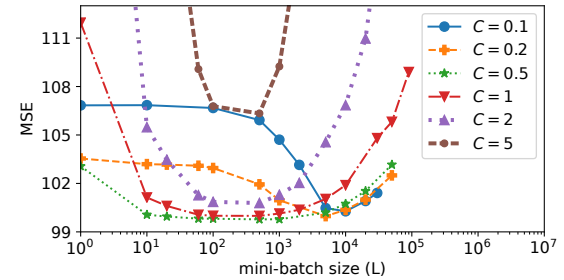

Figure 3: Impact of the mini-batch size (L) and the scaling factor (C) on the utility for DP-SCD. Deviating from the best choice for C ($C = 0.5$ for this setup), reduces the width of the flat area and moves the minimum to the right (for smaller C values) or upwards (for larger C values).

lowest. On the one hand, decreasing C reduces the width of the flat area around the minimum and shifts the minimum to the right (i.e., to larger mini-batch sizes). This can be attributed to the fact that larger C corresponds to smaller updates in terms of magnitude (Step 8 in Algorithm 1) and thus aggregating more of them (i.e., updating with larger mini-batch size) is necessary to minimize the MSE. On the other hand, increasing C also reduces the width of the flat area while also increases the MSE. The updates in this case have larger magnitudes (and thus require larger noise) thereby preventing DP-SCD from reaching a good minimum.

Method	Perturbation	Utility Bound
(Zhang et al. 2017)	Output	$\mathcal{O}\left(\frac{M}{N^2\epsilon^2}\right)$
(Chaudhuri and Monteleoni 2009; Chaudhuri, Monteleoni, and Sarwate 2011)	Inner (objective)	$\mathcal{O}\left(\frac{M}{N^2\epsilon^2}\right)$
(Wang, Ye, and Xu 2017)	Inner (update)	$\mathcal{O}\left(\frac{M\cdot\log(N)}{N^2\epsilon^2}\right)$
DP-SCD	Inner (update)	$\mathcal{O}\left(\frac{L^3\cdot\log(\frac{N}{L})}{N^4\epsilon^2}\right)$

Table 1: Comparison of utility bounds of (ϵ, δ)-DP algorithms for empirical risk minimization.

6 Related Work

Perturbation methods. Existing works achieve differentially private ML by perturbing the query output (i.e., model prediction). These works target both convex and non-convex optimization and focus on a specific application (Chaudhuri and Monteleoni 2009; Nguyên and Hui 2017), a subclass of optimization functions (properties of the loss function) (Chaudhuri, Monteleoni, and Sarwate 2011) or a particular optimization algorithm (Abadi et al. 2016; Talwar, Thakurta, and Zhang 2015). These approaches can be divided into three main classes. The first class involves *input perturbation* approaches that add noise to the input data (Duchi, Jordan, and Wainwright 2013). These approaches are easy to implement but often prohibit the ML model from providing accurate predictions. The second class involves *output perturbation* approaches that add noise to the model after the training procedure finishes, i.e., without modifying the vanilla training algorithm. This noise addition can be model-specific (Wu et al. 2017) or model-agnostic (Bassily, Thakurta, and Thakkar 2018; Papernot et al. 2018). The third class involves *inner perturbation* approaches that modify the learning algorithm such that the noise is injected during learning. One method for inner perturbation is to modify the objective of the training procedure (Chaudhuri, Monteleoni, and Sarwate 2011). Another approach involves adding noise to the output of each update step of the training without modifying the objective (Abadi et al. 2016). Our new DP-SCD algorithm belongs to the third class.

DP for Empirical Risk Minimization (ERM). Various works address the problem of ERM (similar to our setup §2.1), through the lens of differential privacy. Table 1 compares the utility bounds between DP-SCD and representative works for each perturbation method for DP-ERM. We simplify the bounds following (Wang, Ye, and Xu 2017) for easier comparison. The assumptions of these methods, described in (Wang, Ye, and Xu 2017, Table 1) and §4, are similar[4]. We highlight that the bound for DP-SCD is independent of the dimensionality of the problem (M) due to the dual updates, while also includes the mini-batch size (L) for quantifying the impact of the varying degree of parallelism. If $L = \mathcal{O}(N)$, then the ratio N/M determines whether the bound of DP-SCD is better (i.e., smaller). We plan to investigate the effect of this ratio in our future work.

Existing DP-ERM methods based on SGD typically require the tuning of an additional hyperparameter (learning rate or step size) as in DP-SGD (Abadi et al. 2016). The value of this hyperparameter for certain loss functions can be set based on properties of these functions (Wu et al. 2017).

Furthermore regarding (Wu et al. 2017), the authors build upon permutation-based SGD and employ output perturbation, but tolerate only a constant number of iterations.

Coordinate descent. SCD algorithms do not require parameter tuning if they update one coordinate (or block) at a time, by exact minimization (or Taylor approximation). One such algorithm is SDCA (Shalev-Shwartz and Zhang 2013b) that is similar to DP-SCD when setting $\epsilon \to \infty$, $L = 1$ and $C \to \infty$. Alternative SCD algorithms take a gradient step in the coordinate direction that requires a step size (Nesterov 2012; Beck and Tetruashvili 2013).

Parallelizable variants of SCD such as (Bradley et al. 2011b; Richtárik and Takáč 2016; Ioannou, Mendler-Dünner, and Parnell 2019) have shown remarkable speedup when deployed on multiple CPUs/GPUs (Parnell et al. 2017; Hsieh, Yu, and Dhillon 2015; Shalev-Shwartz and Zhang 2013a; Chiang, Lee, and Lin 2016; Zhuang et al. 2018), or multiple machines (Ma et al. 2015; Dünner et al. 2018). These works employ sampling to select the data to be updated in parallel. DP-SCD also employs sampling via the mini-batch size (L), similar to the lot size of DP-SGD (Abadi et al. 2016). This enables parallel updates and improves the privacy-utility trade-off. DP-SCD also builds on ideas from distributed learning, such as the Co-CoA method (Ma et al. 2015) where in our case each node computes its update based on a single datapoint from its data partition. Different distributed methods might also be worth exploring.

7 Conclusion

This paper presents the first differentially private version of the popular stochastic coordinate descent algorithm. We demonstrate that extending SCD with a mini-batch approach is crucial for the algorithm to be competitive against SGD-based alternatives in terms of privacy-utility trade-off. To achieve reliable convergence for our mini-batch parallel SCD, we build on a separable surrogate model to parallelize updates across coordinates, inspired by block separable distributed methods such as in (Smith et al. 2017). This parallel approach inherits the strong convergence guarantees of the respective method. In addition we provide a utility guarantee for our DP-SCD algorithm despite the noise addition and the update scaling. We also argue that the dual formulation of DP-SCD is preferable over the primal due to the example-wise access pattern of the training data, that is more aligned with the focus of differential privacy (i.e., to protect individual entries in the training data). Finally, we provide promising empirical results for DP-SCD (compared to the SGD-based alternative) for three popular applications.

---

[4]DP-SCD does not require the loss function to be Lipschitz.

Acknowledgements

Ethical Impact

Many existing industrial applications are based on the SCD algorithm while they employ sensitive data (e.g., health records) to train machine learning models. Our algorithm (DP-SCD) provides a direct mechanism to improve the privacy-utility trade-off while also preserves the core structure of SCD. Therefore, big industrial players can extend existing implementations (that are the outcome of a large amount of research and engineering efforts) to safeguard the privacy of the users up to a level that preserves the utility requirements of the application.

References

Abadi, M.; Chu, A.; Goodfellow, I.; McMahan, H. B.; Mironov, I.; Talwar, K.; and Zhang, L. 2016. Deep learning with differential privacy. In *CCS*, 308–318. ACM.

Bassily, R.; Thakurta, A. G.; and Thakkar, O. D. 2018. Model-Agnostic Private Learning. In *NIPS*, 7102–7112.

Beck, A.; and Tetruashvili, L. 2013. On the convergence of block coordinate descent type methods. *SIAM journal on Optimization* 23(4): 2037–2060.

Bradley, J. K.; Kyrola, A.; Bickson, D.; and Guestrin, C. 2011a. Parallel Coordinate Descent for L1-Regularized Loss Minimization. In *ICML*, ICML'11, 321–328.

Bradley, J. K.; Kyrola, A.; Bickson, D.; and Guestrin, C. 2011b. Parallel coordinate descent for l1-regularized loss minimization. *arXiv preprint arXiv:1105.5379* .

Chaudhuri, K.; and Monteleoni, C. 2009. Privacy-preserving logistic regression. In *NIPS*, 289–296.

Chaudhuri, K.; Monteleoni, C.; and Sarwate, A. D. 2011. Differentially private empirical risk minimization. *JMLR* 12(Mar): 1069–1109.

Chiang, W.-L.; Lee, M.-C.; and Lin, C.-J. 2016. Parallel dual coordinate descent method for large-scale linear classification in multi-core environments. In *KDD*, 1485–1494.

Damaskinos, G.; Mendler-Dünner, C.; Guerraoui, R.; Papandreou, N.; and Parnell, T. 2020. Differentially Private Stochastic Coordinate Descent. *arXiv preprint arXiv:2006.07272* .

Duchi, J. C.; Jordan, M. I.; and Wainwright, M. J. 2013. Local privacy and statistical minimax rates. In *FOCS*, 429–438. IEEE.

Dünner, C.; Forte, S.; Takáč, M.; and Jaggi, M. 2016. Primal-Dual Rates and Certificates. In *ICML*, 783–792. JMLR.org.

Dünner, C.; Parnell, T.; Sarigiannis, D.; Ioannou, N.; Anghel, A.; Ravi, G.; Kandasamy, M.; and Pozidis, H. 2018. Snap ML: A hierarchical framework for machine learning. In *NIPS*, 250–260.

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.

Fan, R.-E.; Chang, K.-W.; Hsieh, C.-J.; Wang, X.-R.; and Lin, C.-J. 2008. LIBLINEAR: A library for large linear classification. *JMLR* 9(Aug): 1871–1874.

Hsieh, C.-J.; Yu, H.-F.; and Dhillon, I. S. 2015. PASSCoDe: Parallel ASynchronous Stochastic dual Co-ordinate Descent. In *ICML*, volume 15, 2370–2379.

Ioannou, N.; Mendler-Dünner, C.; and Parnell, T. 2019. SySCD: A System-Aware Parallel Coordinate Descent Algorithm. In *NIPS*, 592–602. Curran Associates, Inc.

Ma, C.; Smith, V.; Jaggi, M.; Jordan, M. I.; Richtárik, P.; and Takáč, M. 2015. Adding vs. averaging in distributed primal-dual optimization. In *ICML*, 1973–1982.

Nesterov, Y. 2012. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization* 22(2): 341–362.

Nguyên, T. T.; and Hui, S. C. 2017. Differentially private regression for discrete-time survival analysis. In *CIKM*, 1199–1208.

Papernot, N.; Song, S.; Mironov, I.; Raghunathan, A.; Talwar, K.; and Erlingsson, U. 2018. Scalable Private Learning with PATE. In *ICLR*.

Parnell, T.; Dünner, C.; Atasu, K.; Sifalakis, M.; and Pozidis, H. 2017. Large-scale stochastic learning using GPUs. In *IPDPSW*, 419–428. IEEE.

Richtárik, P.; and Takáč, M. 2016. Parallel coordinate descent methods for big data optimization. *Mathematical Programming* 156(1-2): 433–484.

Shalev-Shwartz, S.; and Zhang, T. 2013a. Accelerated minibatch stochastic dual coordinate ascent. In *NIPS*, 378–385.

Shalev-Shwartz, S.; and Zhang, T. 2013b. Stochastic dual coordinate ascent methods for regularized loss minimization. *JMLR* 14(Feb): 567–599.

Smith, V.; Forte, S.; Ma, C.; Takáč, M.; Jordan, M. I.; and Jaggi, M. 2017. CoCoA: A general framework for communication-efficient distributed optimization. *JMLR* 18(1): 8590–8638.

Talwar, K.; Thakurta, A. G.; and Zhang, L. 2015. Nearly optimal private lasso. In *NIPS*, 3025–3033.

Thakkar, O.; Andrew, G.; and McMahan, H. B. 2019. Differentially Private Learning with Adaptive Clipping. *arXiv preprint arXiv:1905.03871* .

Wang, D.; Ye, M.; and Xu, J. 2017. Differentially private empirical risk minimization revisited: Faster and more general. In *NIPS*, 2722–2731.

Wright, S. J. 2015. Coordinate descent algorithms. *Mathematical Programming* 151(1): 3–34.

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 *SIGMOD*, 1307–1322.

Zhang, J.; Zheng, K.; Mou, W.; and Wang, L. 2017. Efficient private ERM for smooth objectives. *IJCAI* .

Zhuang, Y.; Juan, Y.; Yuan, G.-X.; and Lin, C.-J. 2018. Na-ive parallelization of coordinate descent methods and an application on multi-core l1-regularized classification. In *CIKM*, 1103–1112.