Fast and Differentially Private Algorithms for Decentralized Collaborative Machine Learning

Aurélien Bellet, Rachid Guerraoui, Mahsa Taziki, Marc Tommasi

Fast and Differentially Private Algorithms for Decentralized Collaborative Machine Learning

Aurélien Bellet*[1], Rachid Guerraoui†[2], Mahsa Taziki†[2], and Marc Tommasi*[3]

[1]INRIA
[2]EPFL
[3]Université de Lille

Abstract

Consider a set of agents in a peer-to-peer communication network, where each agent has a personal dataset and a personal learning objective. The main question addressed in this paper is: *how can agents collaborate to improve upon their locally learned model without leaking sensitive information about their data?* Our first contribution is to reformulate this problem so that it can be solved by a block coordinate descent algorithm. We obtain an efficient and fully decentralized protocol working in an asynchronous fashion. Our second contribution is to make our algorithm differentially private to protect against the disclosure of any information about personal datasets. We prove convergence rates and exhibit the trade-off between utility and privacy. Our experiments show that our approach dramatically outperforms previous work in the non-private case, and that under privacy constraints we significantly improve over purely local models.

1 Introduction

Connected personal devices are now widespread: they can collect and process increasingly large and sensitive user data. For instance, a smartphone is able to log webpages that its owner visited but also the physical locations that he/she went to, how much he/she walks in a day, etc; smart home devices can record voice commands, room temperature, energy consumption, and so on. While this information can be leveraged through machine learning to provide useful personalized services to the user, it also raises serious privacy concerns. Indeed, a common practice is to centralize data from all users on an external server for batch processing, sometimes without explicit consent from users and with little oversight. On the other hand, if the data is considered too sensitive to be shared (due to legislation or because the user opts out), then one has to learn on each device separately without taking advantage of the multiplicity of data sources (e.g., information from similar users). This approach respects privacy but leads to poor accuracy, in particular for new or moderately active users who have not yet collected much data.

Ideally, users (agents) should be able to collaborate to learn more accurate models while ensuring that their data stay on their local device and that the algorithm does not leak sensitive information to others. Specifically, we are interested in the *fully decentralized setting*: agents operate asynchronously and communicate over a network in a peer-to-peer fashion, without any central entity to aggregate results or even to coordinate the protocol. Such a peer-to-peer setting can be a constraint on the physical network (e.g. IoT). Besides, these decentralized architectures can scale to large sets of users, and intrinsically provide additional security guarantees as it is more difficult for a malicious party to systematically collect all the information transmitted

*first.last@inria.fr
†first.last@epfl.ch

over the network. Peer-to-peer algorithms scale well by design due to their locality even in situations where there is a central server to route the communications. It was recently shown that decentralized network can perform better than centralized one because it avoids communication bottleneck at the master node [16]. Decentralized collaborative learning has been recently considered in [25], but this work did not consider any privacy constraints. In fact, while there has been a large body of work on privacy-preserving machine learning from centralized data, notably based on differential privacy (see [9, 4, 2] and references therein), the case where sensitive datasets are distributed across multiple data owners has been much less studied, let alone the fully decentralized setting. Existing approaches for the distributed case [20, 12, 24, 21, 23] require a central (sometimes trusted) server, assume the local data distribution is the same for all users and/or are designed to learn a single global model rather than a personal model for each user.

In this paper, we ask a challenging question: given decentralization and privacy constraints, can agents improve upon their purely local models through collaboration? Our contributions towards a positive answer to this question are three-fold. First, we propose a decentralized and asynchronous block coordinate descent algorithm for collaborative learning. Taking advantage of the structure of the problem, this algorithm accommodates general loss functions, with simple updates and provable convergence rates. Second, we design a differentially-private scheme based on randomly perturbing each update of our algorithm. This scheme guarantees that the messages sent by the users over the network during the execution of the algorithm do not reveal significant information about any data point of any local dataset. We formally analyze the utility loss due to privacy, with interesting implications on the optimal way to scale the noise across iterations. Third, we conduct experiments to validate our approach. The empirical results show that the trade-off between utility and privacy is in line with our theoretical findings, and that under strong privacy constraints we can still outperform the purely local models in terms of accuracy.

The rest of the paper is organized as follows. Section 2 introduces the problem setting, the notion of differential privacy, and discusses relevant work. In Section 3, we present our decentralized block coordinate descent algorithm and its convergence guarantees. Section 4 introduces a differentially private version of our algorithm and studies the utility loss. Finally, Section 5 is dedicated to numerical experiments. Detailed proofs can be found in the supplementary material.

2 Preliminaries and Background

We start by describing the decentralized collaborative learning framework that we consider in this paper, and briefly present existing results. We then review the notion of differential privacy and go over some relevant work in this area.

2.1 Decentralized Collaborative Learning of Personal Models

Problem setting. Consider a set of n agents. Each agent i has a local data distribution μ_i over the space $\mathcal{X} \times \mathcal{Y}$ and has access to a set $\mathcal{S}_i = \{(x_i^j, y_i^j)\}_{j=1}^{m_i}$ of $m_i \geq 0$ training examples drawn i.i.d. from μ_i. The goal of agent i is to learn a model $\theta \in \mathbb{R}^p$ with small expected loss $\mathbb{E}_{(x_i,y_i)\sim\mu_i}[\ell(\theta; x_i, y_i)]$, where the loss function $\ell(\theta; x_i, y_i)$ is convex in θ and measures the performance of θ on data point (x_i, y_i). In a setting where agent i is not aware of the existence of other users and acts on its own, the standard approach is to learn a model by minimizing its (potentially regularized) empirical loss:

$$\theta_i^{loc} \in \operatorname*{arg\,min}_{\theta\in\mathbb{R}^p} \left[\mathcal{L}_i(\theta; \mathcal{S}_i) = (1/m_i) \sum_{j=1}^{m_i} \ell_i(\theta; x_i^j, y_i^j) + \lambda_i \|\theta\|^2\right], \text{with } \lambda_i \geq 0. \tag{1}$$

In this paper, agents do not learn in isolation but rather participate in a decentralized peer-to-peer network over which they can exchange information. Such collaboration gives them the opportunity to learn a better model than (1), for instance by allowing some agents to compensate for their lack of data. Formally, let

$G = (\llbracket n \rrbracket, E, W)$ be a weighted connected graph over the set of agents where $E \in \llbracket n \rrbracket \times \llbracket n \rrbracket$ is the set of edges and $W \in \mathbb{R}^{n \times n}$ is a nonnegative weight matrix. W_{ij} gives the weight of edge $(i, j) \in E$ with the convention that $W_{ij} = 0$ if $(i, j) \notin E$ or $i = j$. Following previous work (see e.g., [11, 25]), we assume that the edge weights reflect a notion of "task relatedness": the weight W_{ij} between agents i and j tends to be large if the models minimizing their respective expected loss are similar. In order to scale to large networks, our goal is to design *fully decentralized algorithms*: each agent i only communicates with its neighborhood $\mathcal{N}_i = \{j : W_{ij} > 0\}$ without global knowledge of the network, and can proceed without synchronizing with other agents across the network. Overall, the problem can thus be seen as a multi-task learning problem over a large number of tasks (agents) with imbalanced training sets, which must be solved in a fully decentralized way.

Relevant work. Most of the work in decentralized learning and optimization has focused on the distributed consensus problem, where the goal is to find a single global model which minimizes the sum of the local loss functions (see e.g., [19, 22, 6, 26, 5]. For decentralized learning of personal models, [25] considered a general objective function which trades off between models with small empirical local loss and models that are smooth within neighborhoods (see Eq. 3 in Section 3). At the cost of introducing many auxiliary variables, they are able to cast the objective as a partial consensus problem over the network which can be solved using a decentralized gossip ADMM algorithm [27]. It involves minimizing a perturbed version of the local loss of two neighboring agents at each iteration t and has an $O(1/t)$ convergence rate. Privacy constraints were not considered in this work.

2.2 Differential Privacy

Differential Privacy (DP) [7] has emerged as a powerful measure of how much information about any individual entry of a dataset is contained in the output of an algorithm. Formally, let $\mathcal{M}$ be a randomized mechanism taking a dataset as input, and let $\epsilon > 0, \delta \geq 0$. We say that $\mathcal{M}$ is (ϵ, δ)-differentially private if for all datasets $\mathcal{S} = \{z_1, \ldots, z_i, \ldots, z_m\}, \mathcal{S}' = \{z_1, \ldots, z_i', \ldots, z_m\}$ differing in a single data point and for all sets of possible outputs $\mathcal{O} \subseteq \text{range}(\mathcal{M})$, we have:

$$Pr(\mathcal{M}(\mathcal{S}) \in \mathcal{O}) \leq e^{\epsilon} Pr(\mathcal{M}(\mathcal{S}') \in \mathcal{O}) + \delta, \tag{2}$$

where the probability is over the randomness of the mechanism. At a high level, one can see (2) as ensuring that $\mathcal{M}(\mathcal{S})$ does not leak much information about any individual data point contained in $\mathcal{S}$. DP has many attractive properties: in particular it provides strong robustness against background knowledge attacks and does not rely on computational assumptions. The composition of several DP mechanisms remains DP, albeit a graceful degradation in the parameters (see [10, 14] for strong composition results). We refer to [9] for more details on DP.

Relevant work. DP has been mostly considered in the context where a "trusted curator" has access to all data. Existing DP schemes for machine learning in this setting typically rely on the addition of appropriately scaled noise to the learned model (output perturbation) or to the objective function itself (objective perturbation), see for instance [4]. The private multi-party setting, in which sensitive datasets are distributed across multiple data owners, is known to be harder [17] and has been less studied in spite of its relevance for many applications. Local DP [6, 15], consisting in locally perturbing the data points themselves before releasing them, often achieves poor accuracy (especially when local datasets are small). An alternative strategy is to rely on DP aggregation of models locally trained by each party [20, 12]. DP schemes for (stochastic) gradient descent in the distributed setting have also been proposed, based on perturbing the gradients, the iterates and/or the objective [24, 21, 13, 23]. Apart from local DP, the above methods do not apply to our setting for a combination of reasons. In particular, the local data distribution is different for each party and we learn a personal model for each agent instead of a single global model. Last but not least, we seek an asynchronous and fully decentralized algorithm without any master node to perform aggregation or coordinate the protocol. We are not aware of any previous DP machine learning schemes designed for this setting.

3 Decentralized Collaborative Learning with Block Coordinate Descent

We start by introducing some convenient notations. For any $V \in \mathbb{R}^{np}$ and $i \in [\![n]\!]$, we will denote by $V_i \in \mathbb{R}^p$ its i-th block of size p. We also define the matrices $U_i \in \mathbb{R}^{np \times p}$, $i \in [\![n]\!]$, such that $(U_1, \dots, U_n) = I_{np}$. We thus have $V_i = U_i^T V$ for any $V \in \mathbb{R}^{np}$.

3.1 Objective Function

Our goal in collaborative learning is to jointly learn the models of the agents by leveraging both their local datasets and the similarity information embedded in the network graph. We rely on the principle of graph regularization used in [11, 25] to favor models that vary smoothly on the graph. Specifically, representing the set of all models as a stacked vector $\Theta = [\Theta_1; \dots; \Theta_n] \in \mathbb{R}^{np}$, the objective function we wish to minimize is defined as follows:

$$\mathcal{Q}_{\mathcal{L}}(\Theta) = \frac{1}{2} \sum_{i<j}^{n} W_{ij} \|\Theta_i - \Theta_j\|^2 + \mu \sum_{i=1}^{n} D_{ii} c_i \mathcal{L}_i(\Theta_i; \mathcal{S}_i), \tag{3}$$

where $\mu > 0$ is a trade-off parameter, $D_{ii} = \sum_{j=1}^{n} W_{ij}$ is a normalization factor and $c_i \in (0, 1] \propto m_i$ is the "confidence" of agent i.[1] Minimizing (3) implements a trade-off between having similar models for strongly connected agents and models that are accurate on their respective local datasets (the higher the confidence of an agent, the more importance given to the latter part). This allows agents to leverage relevant information from their neighbors — it is particularly salient for agents with less data which can gain useful knowledge from better-endowed neighbors without "polluting" them with their own inaccurate model.

We now discuss a few assumptions and properties of $\mathcal{Q}_{\mathcal{L}}$. We assume that for any $i \in [\![n]\!]$, the local loss function $\mathcal{L}_i$ of agent i is convex in its first argument with L_i^{loc}-Lipschitz continuous gradient. This implies that $\mathcal{Q}_{\mathcal{L}}$ is convex in Θ.[2] If we further assume that each local loss $\mathcal{L}_i$ is σ_i^{loc}-strongly convex in its first argument with $\sigma_i^{loc} > 0$ (this is the case for instance when the local loss is L2-regularized), then $\mathcal{Q}_{\mathcal{L}}$ is σ-strongly convex with $\sigma \geq \mu \min_{1 \leq i \leq n} [D_{ii} c_i \sigma_i^{loc}] > 0$. In other words, for any $\Theta, \Theta' \in \mathbb{R}^{np}$ we have $\mathcal{Q}_{\mathcal{L}}(\Theta') \geq \mathcal{Q}_{\mathcal{L}}(\Theta) + \nabla \mathcal{Q}_{\mathcal{L}}(\Theta)^T (\Theta' - \Theta) + \frac{\sigma}{2} \|\Theta' - \Theta\|_2^2$. The partial derivative of $\mathcal{Q}_{\mathcal{L}}(\Theta)$ corresponding to the variables in Θ_i is given by

$$[\nabla \mathcal{Q}_{\mathcal{L}}(\Theta)]_i = D_{ii}(\Theta_i + \mu c_i \nabla \mathcal{L}_i(\Theta_i; \mathcal{S}_i)) - \sum_{j \in \mathcal{N}_i} W_{ij} \Theta_j. \tag{4}$$

For $i \in [n]$, the i-th *block Lipschitz constant* L_i of $\nabla \mathcal{Q}_{\mathcal{L}}(\Theta)$ satisfies $\|[\nabla \mathcal{Q}_{\mathcal{L}}(\Theta + U_i d)]_i - [\nabla \mathcal{Q}_{\mathcal{L}}(\Theta)]_i\| \leq L_i \|d\|$ for any $\Theta \in \mathbb{R}^{np}$ and $d \in \mathbb{R}^p$. It is easy to see that $L_i = D_{ii}(1 + \mu c_i L_i^{loc})$.

3.2 Proposed Algorithm

Our goal is to minimize the objective function (3) in a fully decentralized manner. Specifically, we operate in the asynchronous execution model [3, 1, 18]: each agent has a *local* clock ticking at the times of a rate 1 Poisson process, and wakes up when it ticks without waiting for other agents. As local clocks are i.i.d., we can equivalently consider a single clock which ticks when one of the local clocks ticks. This provides a more convenient way to state and analyze the algorithms in terms of a global clock counter t. For communication, we rely on a broadcast-based model [1, 18] where agents communicate by sending messages to all their neighbors at once (without expecting a reply). This is in contrast to gossip-based algorithms which rely on bidirectional communication between pairs of agents. The broadcast-based model is very appealing in

[1] In practice we will set $c_i = m_i / \max_j m_j$ (plus some small constant when $m_i = 0$).

[2] This follows from the fact that the first term in (3) is a Laplacian quadratic form, hence convex in Θ.

wireless distributed systems, since sending a message to all neighbors has the same cost as sending to a single neighbor.

Given the above constraints, we propose a decentralized coordinate descent algorithm to minimize (3). Suppose that at time step t, agent i wakes up. Two consecutive actions are performed by i:

- *Update step*: agent i updates its local model based on the most recent information $\Theta_j(t)$ received from its neighbors $j \in \mathcal{N}_i$:

$$\begin{aligned}\Theta_i(t+1) &= \Theta_i(t) - \frac{1}{L_i}[\nabla \mathcal{Q}_{\mathcal{L}}(\Theta(t))]_i \\ &= (1-\alpha)\Theta_i(t) + \alpha\Big(\sum_{j\in\mathcal{N}_i} \frac{W_{ij}}{D_{ii}}\Theta_j(t) - \mu c_i \nabla \mathcal{L}_i(\Theta_i(t); \mathcal{S}_i)\Big),\end{aligned} \tag{5}$$

  where $\alpha = 1/(1+\mu c_i L_i^{loc}) \in (0,1]$.

- *Broadcast step*: agent i sends its updated model $\Theta_i(t+1)$ to its neighborhood $\mathcal{N}_i$.

All other variables in the network remain unchanged at that iteration.

The update step (5) consists in a block coordinate descent update with respect to Θ_i and only requires agent i to know the models $\Theta_j(t)$ previously broadcast by its neighbors $j \in \mathcal{N}_i$. Furthermore, the agent does not need to know the global iteration counter t, hence no global clock is needed. The algorithm is thus fully decentralized and asynchronous. Interestingly, notice that this block coordinate descent update is adaptive to the confidence level of each agent in two respects: (i) globally, the more confidence, the more importance given to the gradient of the local loss compared to the neighbors' models, and (ii) locally, when $\Theta_i(t)$ is close to a minimizer of the local loss $\mathcal{L}_i$ (which is the case for instance if we initialize $\Theta_i(0)$ to such a minimizer), agents with low confidence will trust their neighbors' models more aggressively than agents with high confidence (which will make more conservative updates).[3] This is in line with the intuition that agents with low confidence should diverge more quickly from their local minimizer than those with high confidence.

Under our assumption that the local clocks of the agents are i.i.d., the above algorithm can be seen as a randomized block coordinate descent algorithm [28]. It enjoys a fast linear convergence rate when $\mathcal{Q}_{\mathcal{L}}$ is strongly convex, as shown in the following result.

Proposition 1 (Convergence rate)**.** *For any $T > 0$, let $(\Theta(t))_{t=1}^T$ be the sequence of iterates generated by the proposed algorithm running for T iterations from an initial point $\Theta(0) \in \mathbb{R}^{np}$. Let $\mathcal{Q}_{\mathcal{L}}^\star \in \min_{\Theta\in\mathbb{R}^{np}} \mathcal{Q}_{\mathcal{L}}(\Theta)$. When $\mathcal{Q}_{\mathcal{L}}$ is σ-strongly convex, we have:*

$$\mathbb{E}\left[\mathcal{Q}_{\mathcal{L}}(\Theta(T)) - \mathcal{Q}_{\mathcal{L}}^\star\right] \leq \Big(1 - \frac{\sigma}{nL_{max}}\Big)^T \left(\mathcal{Q}_{\mathcal{L}}(\Theta(0)) - \mathcal{Q}_{\mathcal{L}}^*\right).$$

Proof. This follows from a slight adaptation of the proof of [28] (Theorem 1 therein) to the block coordinate descent case. Note that the result can also be obtained as a special case of our Theorem 2 (later introduced in Section 4.2) by setting the noise scale $s_i(t) = 0$ for all t, i. □

Remark 1. *For general convex $\mathcal{Q}_{\mathcal{L}}$, an $O(1/t)$ rate can be obtained, see [28] for details.*

The above result shows that each iteration shrinks the suboptimality gap by a constant factor. While this factor degrades linearly with the number of agents n, this is compensated by the fact that the number of iterations done in parallel also scales roughly linearly with n (because agents wake up asynchronously). We thus expect the algorithm to scale gracefully with the size of the network if the number of updates per agent remains constant. The value $\frac{\sigma}{L_{max}} \geq \frac{\mu \min_{1\leq i\leq n}[D_{ii}c_i\sigma_i^{loc}]}{\max_{1\leq i\leq n}[D_{ii}(1+\mu c_i L_i^{loc})]} > 0$ is the ratio between the lower and upper bound on the curvature of $\mathcal{Q}_{\mathcal{L}}$. Focusing on the relative differences between agents and assuming constant

[3] This second property is in contrast to a (centralized) gradient descent approach which would use the same constant, more conservative step size (equal to the standard Lipschitz constant of $\mathcal{Q}_{\mathcal{L}}$) for all agents.

σ_i^{loc}'s and L_i^{loc}'s, it indicates that the algorithm converges faster when the degree-weighted confidence of agents is approximately the same. On the other hand, two types of agents can represent a bottleneck for the convergence rate: (i) a high-confidence and high-degree agent (the overall progress is then very dependent on the updates of that agent), and (ii) a low-confidence agent which is also poorly connected (and thus converges slowly).

Remark 2 (Comparison to existing ADMM algorithm)**.** *Our algorithm has several advantages over the decentralized ADMM introduced by [25]. It is much simpler (no auxiliary variable needed), achieves linear convergence rate for strongly convex functions, and each iteration is computationally cheaper (only one local gradient step instead of full minimization). We will show in Section 5 that our algorithm indeed performs much better in practice.*

4 Differentially Private Collaborative Learning

As described above, the algorithm introduced in the previous section has many interesting properties. However, it is not differentially-private: while there is no direct exchange of data between agents, the sequence of iterates broadcast by an agent may reveal information about its private dataset through the gradient of the local loss. In this section, we start by defining the privacy model of interest and then introduce an appropriate scheme to make our algorithm private. We study its utility loss and the trade-off between utility and privacy.

4.1 Privacy Model

At a high level, our goal is to prevent eavesdropping attacks. We assume the existence of an adversary who observes all the information sent over the network during the execution of the algorithm, but cannot access the agents' internal memory. We want to ensure that such an adversary cannot learn much information about any individual data point of any agent's dataset. This is a very strong notion of privacy: each agent does not trust any other agent or any third-party to process its data, hence the privacy-preserving mechanism must be implemented at the agent level. Furthermore, note that our privacy model protects any agent against all other agents even if they collude (i.e., share the information they receive).[4]

To formally define this privacy model, we rely on the notion of Differential Privacy introduced in Section 2.2. Following the notations of (2), each agent i runs a mechanism $\mathcal{M}_i(\mathcal{S}_i)$ which takes its local dataset $\mathcal{S}_i$ and outputs all the information sent by i over the network during the execution of the algorithm (i.e., the sequence of iterates broadcast by the agent). Our goal is to make $\mathcal{M}_i(\mathcal{S}_i)$ (ϵ, δ)-DP for all agents i simultaneously. Note that learning purely local models (1) is a perfectly private baseline according to the above definition as agents do not exchange any information. Below, we present a way to collaboratively learn better models while ensuring privacy.

4.2 Privacy-Preserving Scheme

The privacy-preserving version of our algorithm consists in replacing the update step in (5) by the following one (assuming that at time t agent i wakes up):

$$\widetilde{\Theta}_i(t+1) = (1-\alpha)\Theta_i(t) + \alpha\Bigg(\sum_{j\in\mathcal{N}_i}\frac{W_{ij}}{D_{ii}}\Theta_j(t) - \mu c_i(\nabla\mathcal{L}_i(\Theta_i(t);\mathcal{S}_i) + \eta_i(t))\Bigg), \tag{6}$$

where $\eta_i(t) \sim Laplace(0, s_i(t))^p \in \mathbb{R}^p$ is a noise vector drawn from a Laplace distribution with finite scale $s_i(t) \geq 0$.[5] The difference with the non-private update is that agent i adds appropriately scaled Laplace noise

[4] We assume a honest-but-curious model for the agents: they want to learn as much as possible from the information that they receive but they truthfully follow the protocol.

[5] When $s_i(t) = 0$, we use the convention $Laplace(0, 0) = 0$ w.p. 1.

to the gradient of its local loss $\mathcal{L}_i$. It then sends the resulting noisy iterate $\widetilde{\Theta}_i(t+1)$, instead of $\Theta_i(t+1)$, to its neighbors.

Assume that update (6) is run T_i times by agent i within the total $T > 0$ iterations across the network. Let $\mathcal{T}_i = \{t_i^k\}_{k=1}^{T_i}$ be the set of iterations at which agent i woke up and consider the mechanism $\mathcal{M}_i(\mathcal{S}_i) = \{\widetilde{\Theta}_i(t_i+1) : t_i \in \mathcal{T}_i\}$. The following theorem gives the scale of the additional noise at each iteration, $s_i(t_i)$, to provide desired overall differential privacy guarantees.

Theorem 1 (Differential privacy of $\mathcal{M}_i$)**.** *Let $i \in [\![n]\!]$ and assume that $\mathcal{L}_i(\theta; \mathcal{S}_i) = \frac{1}{m_i}\sum_{k=1}^{m_i} \ell_i(\theta; x_i^k, y_i^k) + \lambda_i\|\theta\|^2$ where $\ell(\cdot; x, y)$ is L_0-Lipschitz with respect to the L_1-norm for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$. For any $t_i \in \mathcal{T}_i$, let $s_i(t_i) = \frac{2L_0}{\epsilon_i(t_i) m_i}$ for some $\epsilon_i(t_i) > 0$. For any $\bar{\delta}_i \in [0, 1]$ and initial point $\widetilde{\Theta}(0) \in \mathbb{R}^{np}$ independent of $\mathcal{S}_i$, the mechanism $\mathcal{M}_i(\mathcal{S}_i)$ is $(\bar{\epsilon}_i, \bar{\delta}_i)$-DP with*

$$\bar{\epsilon}_i = \min\Bigg\{ \sum_{t_i=1}^{T_i} \epsilon_i(t_i), \sum_{t_i=1}^{T_i} \frac{(e^{\epsilon_i(t_i)} - 1)\epsilon_i(t_i)}{e^{\epsilon_i(t_i)} + 1} + \sqrt{2\epsilon_i(t_i)^2 \log\Big(e + \frac{\sqrt{\sum_{t_i=1}^{T_i} \epsilon_i(t_i)^2}}{\bar{\delta}_i}\Big)}, \\ \sum_{t_i=1}^{T_i} \frac{(e^{\epsilon_i(t_i)} - 1)\epsilon_i(t_i)}{e^{\epsilon_i(t_i)} + 1} + \sqrt{2\epsilon_i(t_i)^2 \log(1/\bar{\delta}_i)} \Bigg\}.$$

Remark 3. *We can obtain a similar result if we assume L_0-Lipschitzness of ℓ w.r.t. L_2-norm (instead of L_1) and use Gaussian noise (instead of Laplace). Details are in the supplementary material.*

Theorem 1 shows that $\mathcal{M}_i(\mathcal{S}_i)$ is $(\bar{\epsilon}_i, 0)$-DP for $\bar{\epsilon}_i = \sum_{t_i=1}^{T_i} \epsilon_i(t_i)$. One can also achieve a better scaling for $\bar{\epsilon}_i$ at the cost of setting $\bar{\delta}_i > 0$ (see [14] for a discussion of the trade-offs in the composition of DP mechanisms). Further note that the noise scale needed to guarantee DP for an agent i is inversely proportional to the size m_i of its local dataset $\mathcal{S}_i$. This is a key property for collaborative learning: agents with more local data (and hence larger confidence and more informative gradients) will add less noise and thus pass useful information to their neighbors. In contrast, agents with small datasets will add more noise but will marginally influence other agents due to their low confidence.

The next result quantifies how the added noise affects the convergence.

Theorem 2 (Utility loss)**.** *For any $T > 0$, let $(\Theta(t))_{t=1}^T$ be the sequence of iterates generated by T iterations of update (6) from an initial point $\Theta(0) \in \mathbb{R}^{np}$. For σ-strongly convex $\mathcal{Q}_\mathcal{L}$, we have:*

$$\mathbb{E}\left[\mathcal{Q}_\mathcal{L}(\widetilde{\Theta}(T)) - \mathcal{Q}_\mathcal{L}^\star\right] \leq \Big(1 - \frac{\sigma}{nL_{max}}\Big)^T \left(\mathcal{Q}_\mathcal{L}(\Theta(0)) - \mathcal{Q}_\mathcal{L}^\star\right) \\ + \frac{1}{nL_{min}} \sum_{t=0}^{T-1} \sum_{i=1}^{n} \Big(1 - \frac{\sigma}{nL_{max}}\Big)^t \big(\mu D_{ii} c_i s_i(t)\big)^2,$$

where $L_{min} = \min_{1 \leq i \leq n} L_i$.

Theorem 2 shows that the optimization error of the private algorithm after T iterations decomposes into two terms. The first term is the same as in the non-private setting and decreases with T. The second term gives an additive error due to the addition of noise, which takes the form of a weighted sum of the variance of the noise added to the iterate at each iteration (note that we indeed recover the non-private convergence rate of Proposition 1 when the noise scale is 0). When the noise scale used by each agent is constant across iterations, i.e. $s_i(t) = s_i$ for any $i \in [\![n]\!]$ and $t \geq 0$, this additive error is a sum of a geometric series which converges to a finite number as $T \to \infty$.

In practical scenarios, each agent i has a overall privacy budget $(\bar{\epsilon}_i, \bar{\delta}_i)$. Assume that the agents agree on a value for T (e.g., using Proposition 1 to achieve the desired precision). Each agent i is thus expected to wake up $T_i = T/n$ times, and can use Theorem 1 to appropriately distribute its privacy budget across the T_i iterations and stop after T_i updates. While distributing the budget equally across the T_i iterations is a simple

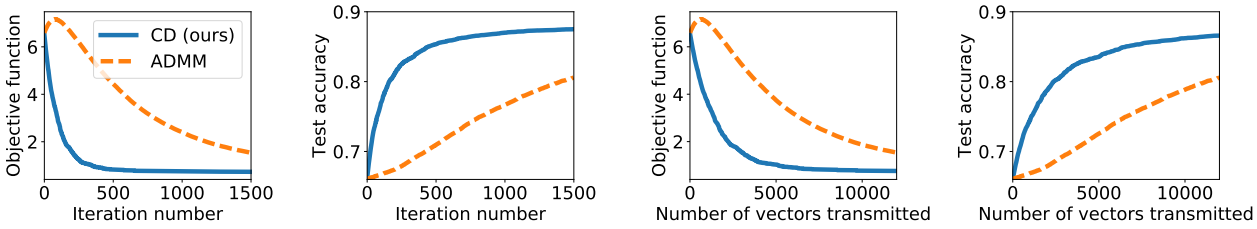


Figure 1: Our block coordinate descent algorithm compared to the existing ADMM algorithm.

and practical strategy, Theorem 2 suggests that better utility can be achieved if the noise scale increases with time. Assume that agents know in advance the clock schedule for a particular run of the algorithm, i.e. agent i knows the global iterations $\mathcal{T}_i$ at which it will wake up. The following result then gives the noise allocation policy minimizing the utility loss.

Proposition 2. *Let $C = 1 - \sigma/nL_{max}$ and for any agent $i \in [\![n]\!]$ define $\lambda_{\mathcal{T}_i}(i) = \sum_{t\in\mathcal{T}_i} \frac{\sqrt[3]{C}-1}{\sqrt[3]{C^T}-1}\sqrt[3]{C^t}$. Assuming $s_i(t_i) = \frac{2L_0}{\epsilon_i(t_i)m_i}$ for $t_i \in \mathcal{T}_i$ as in Theorem 1, the following privacy parameters guarantee optimize the utility loss while ensuring that the budget $\bar{\epsilon}_i$ is matched exactly:*

$$\epsilon_i(t) = \frac{\sqrt[3]{C}-1}{\sqrt[3]{C^T}-1}\sqrt[3]{C^t}\frac{\bar{\epsilon}_i}{\lambda_{\mathcal{T}_i}(i)} \text{ for } t \in \mathcal{T}_i, \text{ and } \epsilon_i(t) = 0 \text{ otherwise.}$$

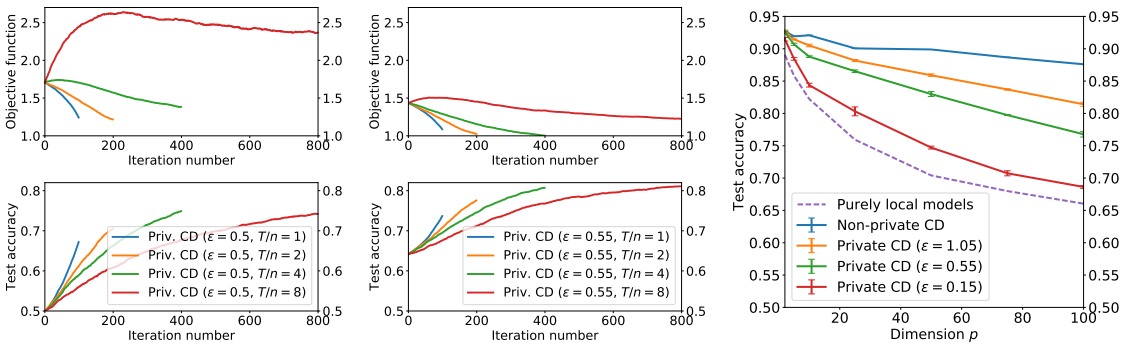
The above noise allocation policy requires the agents to know the schedule in advance as well as the global iteration counter. This is an unrealistic assumption in the fully decentralized setting where no global clock is available. Still, Proposition 2 may be useful to design heuristic strategies that are practical, for instance, based on using the *expected* global time for the agent to wake up at each of its iterations. We leave this for future work.

Remark 4. *Theorem 2 implies that it is beneficial to have a good warm start point $\Theta(0)$: however, $\Theta(0)$ must also be DP. In the supplementary material, we describe a strategy to generate such a private warm start based on the propagation of locally perturbed models throughout the network.*

5 Numerical Experiments

Task description. To be able to compare our algorithm to the one in [25], we conducted experiments on the collaborative linear classification task introduced by the same authors. We briefly recall the setup. Consider a set of $n = 100$ agents. Each of these agents has an underlying target linear separator in $\mathbb{R}^p$ (unknown to the agent) whose first two entries are drawn from a centered normal distribution and the remaining entries are set to 0. The weight between two agents i and j is given by $W_{ij} = \exp((\cos(\phi_{i,j}) - 1)/\gamma)$, where $\phi_{i,j}$ is the angle between the target models and $\gamma = 0.1$ (negligible weights are ignored). Each agent i receives a random number m_i of training points (m_i is drawn uniformly between 10 and 100), and each training point is drawn uniformly around the origin and labeled according to the target model. We then add some label noise, independently flipping each label with probability 0.05. The loss function used by all agents is the logistic loss $\ell(\theta; x, y) = \log(1 + \exp(-y\theta^T x))$ (which is 1-Lipschitz), and the L2 regularization parameter of an agent i is set to $\lambda_i = 1/m_i > 0$ to ensure the overall strong convexity. The hyperparameter μ is tuned to maximize accuracy of the non-private algorithm on a validation set of random problems instances. For each agent, the test accuracy of a model is estimated on a separate sample of 100 test points.

Non-private setting: CD versus ADMM. We start by comparing our coordinate descent algorithm (5) to the ADMM algorithm proposed by [25] in the non-private setting. Both algorithms are fully decentralized and asynchronous, but recall that our algorithm relies on a broadcast communication model (a node sends



(a) Init. with constant vector (b) Private init. ($\epsilon = 0.05$) (c) Overall results

Figure 2: Results in the private setting (averaged over 5 runs). (a)-(b): Evolution of the objective and test accuracy along the iterations for two types of initialization ($p = 100$). (c) Final test accuracy for different dimensions and several privacy regimes. Best seen in color.

information to all its neighbors) while the ADMM algorithm is gossip-based (a node exchanges information with a random neighbor). Which communication model is the most efficient strongly depends on the network infrastructure, but we can meaningfully compare the algorithms by tracking the objective value and the test accuracy with respect to the number of iterations and the number of p-dimensional vectors transmitted along the edges of the network. Both algorithms are initialized using the purely local models, i.e. $\Theta_i(0) = \Theta_i^{loc}$ for all $i \in [\![n]\!]$. Figure 1 shows the results (averaged over 5 runs) for dimension $p = 100$: our coordinate descent algorithm significantly outperforms ADMM despite the fact that ADMM makes several local gradient steps at each iteration (10 in this experiment). We believe that this is mostly due to the fact that the 4 auxiliary variables *per edge* needed by ADMM to encode smoothness constraints are updated only when a particular edge is activated. In contrast, our CD algorithm does not require auxiliary variables.

Private setting. We now turn to the privacy-preserving setting. In this experiment, each agent has the same overall privacy budget $\bar{e}_i = \bar{e}$. It splits its privacy budget equally across $T_i = T/n$ iterations using Theorem 1 with $\bar{\delta}_i = \exp(-5)$, and stops updating when it is done. We first illustrate empirically the trade-offs implied by Theorem 2: namely that running more iterations per agent reduces the first term of the bound but increases the second term because more noise is added at each iteration. This behavior is easily seen in Figure 2(a), where $\Theta(0)$ is initialized to a constant vector. In Figure 2(b), we have initialized the algorithm with a private warm start solution with $\epsilon = 0.05$ (see supplementary material). The results confirm that for a modest additional privacy budget, a good warm start point can lead to lower values of objective function with less iterations (as suggested again by Theorem 2). The gain in test accuracy here is significant.

Figure 2(c) represents the results for various dimensions p between 2 and 100, averaged over 5 runs. We have used the same private warm start strategy as in Figure 2(b), and the number of iterations per node was tuned based on a validation set of random problems instances. We see that even under a small privacy budget (0.15), the resulting models significantly outperform the purely local learned models (a perfectly private baseline). In the supplementary material, we display additional results showing that all agents (irrespective of their dataset size) get an improvement in test accuracy. This improvement is especially large for users with smaller local datasets, effectively correcting the imbalance in dataset size. We also show that perturbing the data itself (local DP [6, 15]) leads to very inaccurate models. These results demonstrate the relevance of our privacy-preserving collaborative learning approach.

6 Conclusion

We introduced and analyzed an efficient algorithm for decentralized collaborative learning under privacy constraints. We believe that this problem is becoming more and more relevant as connected objects become ubiquitous. Further research is needed to address more dynamic scenarios: agents may join or leave during the execution, data may be collected on-line, etc.

Acknowledgments This work was partially supported by grant ANR-16-CE23-0016-01, by a grant from CPER Nord-Pas de Calais/FEDER DATA Advanced data science and technologies 2015-2020 and by European ERC Grant 339539 - AOC (Adversary-Oriented Computing).

References

[1] Tuncer Can Aysal, Mehmet Ercan Yildiz, Anand D. Sarwate, and Anna Scaglione. Broadcast gossip algorithms for consensus. *IEEE Transactions on Signal Processing*, 57(7):2748–2761, 2009.

[2] Raef Bassily, Adam D. Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *FOCS*, 2014.

[3] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE/ACM Transactions on Networking*, 14(SI):2508–2530, 2006.

[4] Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12:1069–1109, 2011.

[5] Igor Colin, Aurélien Bellet, Joseph Salmon, and Stéphan Clémençon. Gossip Dual Averaging for Decentralized Optimization of Pairwise Functions. In *ICML*, 2016.

[6] John C. Duchi, Michael I. Jordan, and Martin J. Wainwright. Privacy Aware Learning. In *NIPS*, 2012.

[7] Cynthia Dwork. Differential Privacy. In *ICALP*, volume 2, 2006.

[8] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, 2006.

[9] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407, 2014.

[10] Cynthia Dwork, Guy N. Rothblum, and Salil Vadhan. Boosting and differential privacy. In *FOCS*, 2010.

[11] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi-task learning. In *KDD*, 2004.

[12] Jihun Hamm, Yingjun Cao, and Mikhail Belkin. Learning privately from multiparty data. In *ICML*, 2016.

[13] Zhenqi Huang, Sayan Mitra, and Nitin Vaidya. Differentially private distributed optimization. *ICDCN*, 2015.

[14] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. In *ICML*, 2015.

[15] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. Extremal mechanisms for local differential privacy. *Journal of Machine Learning Research*, 17:1–51, 2016.

[16] Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *NIPS*, 2017.

[17] Andrew McGregor, Ilya Mironov, Toniann Pitassi, Omer Reingold, Kunal Talwar, and Salil Vadhan. The limits of two-party differential privacy. In *FOCS*, 2010.

[18] Angelia Nedic. Asynchronous broadcast-based convex optimization over a network. *IEEE Transactions on Automatic Control*, 56(6):1337–1351, 2011.

[19] Angelia Nedic and Asuman E. Ozdaglar. Distributed Subgradient Methods for Multi-Agent Optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

[20] Manas A. Pathak, Shantanu Rane, and Bhiksha Raj. Multiparty differential privacy via aggregation of locally trained classifiers. In *NIPS*, 2010.

[21] Arun Rajkumar and Shivani Agarwal. A differentially private stochastic gradient descent algorithm for multiparty classification. In *AISTATS*, 2012.

[22] S. Sundhar Ram, Angelia Nedic, and Venugopal V. Veeravalli. Distributed Stochastic Subgradient Projection Algorithms for Convex Optimization. *Journal of Optimization Theory and Applications*, 147(3):516–545, 2010.

[23] Reza Shokri and Vitaly Shmatikov. Privacy-preserving deep learning. In *CCS*, 2015.

[24] Shuang Song, Kamalika Chaudhuri, and Anand D. Sarwate. Stochastic gradient descent with differentially private updates. In *GlobalSIP*, 2013.

[25] Paul Vanhaesebrouck, Aurélien Bellet, and Marc Tommasi. Decentralized Collaborative Learning of Personalized Models over Networks. In *AISTATS*, 2017.

[26] Ermin Wei and Asuman E. Ozdaglar. Distributed Alternating Direction Method of Multipliers. In *CDC*, 2012.

[27] Ermin Wei and Asuman E. Ozdaglar. On the O(1/k) Convergence of Asynchronous Distributed Alternating Direction Method of Multipliers. In *GlobalSIP*, 2013.

[28] Stephen J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

Supplementary Material

This supplementary material is organized as follows. Section A contains the proofs of the results in the main text. Section B deals with the interesting special case of model propagation and its use as a private warn start strategy. Finally, Section C presents additional experimental results.

A Proofs

A.1 Proof of Theorem 1

We first show that for agent i and an iteration $t_i \in \mathcal{T}_i$, the additional noise $\eta_i(t_i)$ provides $(\epsilon_i(t_i), 0)$-differential privacy for the published $\Theta_i(t_i + 1)$. In the following, two datasets $\mathcal{S}_i^1$ and $\mathcal{S}_i^2$ are called neighbors if they differ in a single data point. We denote this neighboring relation by $\mathcal{S}_i^1 \approx \mathcal{S}_i^2$.

We will need the following lemma.

Lemma 1. *For two neighboring datasets $\mathcal{S}_i^1$ and $\mathcal{S}_i^2$ of the size m_i:*

$$\|\nabla \mathcal{L}_i(\Theta_i; \mathcal{S}_i^1) - \nabla \mathcal{L}_i(\Theta_i; \mathcal{S}_i^2)\|_1 \leq \frac{2 \cdot L_0}{m_i}.$$

Proof. Assume that instead of data point (x_1, y_1) in $\mathcal{S}_i^1$, there is (x_2, y_2) in $\mathcal{S}_i^2$. As $\mathcal{S}_i^1$ and $\mathcal{S}_i^2$ are neighboring datasets, the other data points in $\mathcal{S}_i^1$ and $\mathcal{S}_i^2$ are the same. Hence:

$$\|\nabla \mathcal{L}_i(\Theta_i; \mathcal{S}_i^1) - \nabla \mathcal{L}_i(\Theta_i; \mathcal{S}_i^2)\|_1 = \frac{1}{m_i} \|\nabla \ell(\Theta_i; x_1, y_1) - \nabla \ell(\Theta_i; x_2, y_2)\|_1 \leq \frac{2 \cdot L_0}{m_i},$$

since the L_0-Lipschitzness of $\ell(\cdot; x, y)$ (with respect to the L_1-norm) for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$ implies that for any $\Theta_i \in \mathbb{R}^p$ and $(x, y) \in \mathcal{X} \times \mathcal{Y}$, we have $\|\nabla \ell(\Theta_i; x, y)\|_1 \leq L_0$. □

We continue the proof by bounding the *sensitivity* of $\Theta_i(t_i+1)$ to find the noise scale needed to satisfy $(\epsilon, 0)$-differential privacy. Using Eq. 4, Eq. 5 and Lemma 1, we have:

$$\begin{aligned}
sensitivity(\Theta_i(t_i+1)) &= \max_{\mathcal{S}_i^1 \approx \mathcal{S}_i^2} \|\Theta_i(t_i+1)\|_1 \\
&= \max_{\mathcal{S}_i^1 \approx \mathcal{S}_i^2} \|\frac{1}{L_i}[\nabla \mathcal{Q}_{\mathcal{L}}(\Theta(t_i))]_i\|_1 \qquad (7)\\
&= \frac{\mu c_i D_{ii}}{L_i} \max_{\mathcal{S}_i^1 \approx \mathcal{S}_i^2} \|\nabla \mathcal{L}_i(\Theta_i; \mathcal{S}_i^1) - \nabla \mathcal{L}_i(\Theta_i; \mathcal{S}_i^2)\|_1 \qquad (8)\\
&\leq \frac{2\mu c_i D_{ii} L_0}{m_i L_i},
\end{aligned}$$

where (7)-(8) follow from the fact that $[\nabla \mathcal{Q}_{\mathcal{L}}(\Theta(t_i))]_i$ is the only quantity in the update (5) which depends on the local dataset of agent i.

Recalling the relation between sensitivity and the scale of the addition noise in the context of differential privacy [8], we should have:

$$\epsilon_i(t_i) \cdot s_i^* \geq sensitivity(\Theta_i(t_i+1)) = \frac{2\mu c_i D_{ii} L_0}{m_i L_i},$$

where s_i^* is the scale of the noise added to $\Theta_i(t_i+1)$. In the following we show that $s_i^* \geq \frac{2\mu c_i D_{ii} L_0}{m_i L_i \epsilon_i(t_i)}$. To compute s_i^*, we see how the noise $\eta_i(t_i)$ affects $\widetilde{\Theta}_i(t_i+1)$. Using Eq. 6, definitions of α (Update step) and L_i (the block Lipschitz constant) we have:

$$\begin{aligned}
\widetilde{\Theta}_i(t_i+1) &= (1-\alpha)\Theta_i(t) + \alpha\Big(\sum_{j\in\mathcal{N}_i} \frac{W_{ij}}{D_{ii}}\Theta_j(t) - \mu c_i(\nabla \mathcal{L}_i(\Theta_i(t); \mathcal{S}_i) + \eta_i(t))\Big) \\
&= (1-\alpha)\Theta_i(t) + \alpha\Big(\sum_{j\in\mathcal{N}_i} \frac{W_{ij}}{D_{ii}}\Theta_j(t) - \mu c_i \nabla \mathcal{L}_i(\Theta_i(t); \mathcal{S}_i)\Big) - \alpha\mu c_i \eta_i(t) \\
&= \Theta_i(t_i+1) - \frac{\mu c_i \eta_i(t)}{1+\mu c_i L_i^{loc}} \\
&= \Theta_i(t_i+1) - \frac{\mu c_i D_{ii}}{L_i} \cdot \eta_i(t_i).
\end{aligned}$$

So the scale of the noise added to $\Theta_i(t_i+1)$ is:

$$s_i^* = \frac{\mu c_i D_{ii}}{L_i} \cdot s_i(t_i) = \frac{\mu c_i D_{ii}}{L_i} \cdot \frac{2L_0}{\epsilon_i(t_i) m_i} = \frac{2\mu c_i D_{ii} L_0}{\epsilon_i(t_i) m_i L_i}.$$

Therefore, $s_i^* \geq \frac{sensitivity(\Theta_i(t_i+1))}{\epsilon_i(t_i)}$ is satisfied, hence publishing $\widetilde{\Theta}_i(t_i+1)$ is $(\epsilon_i(t_i), 0)$-differentially private.

We have shown that at any iteration $t_i \in \mathcal{T}_i$, publishing $\Theta_i(t_i+1)$ by agent i is $(\epsilon_i(t_i), 0)$ differentially private. The mechanism $\mathcal{M}_i$ published all $\Theta_i(t_i+1)$ for $t_i \in \mathcal{T}_i$. Using the composition result for differential privacy established in [14], we have that the mechanism $\mathcal{M}_i(\mathcal{S}_i)$ is $(\bar{\epsilon}_i, \bar{\delta}_i)$-DP with $\bar{\epsilon}_i, \bar{\delta}_i$ as in Theorem 1.

Theorem 1 considers the case where $\ell(\cdot; x, y)$ is L_0-Lipschitz for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$ with respect to the L_1-norm. We could instead assume Lipschitzness with respect to the L_2-norm, in which case the noise to add should be Gaussian instead of Laplace. The following remark computes the additional normal noise to preserve differential privacy in this setting.

Remark 5. *Let $i \in [\![n]\!]$. In the case where $\ell(\cdot; x, y)$ is L_0^*-Lipschitz with respect to the L_2-norm for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$, for any $t_i \in \mathcal{T}_i$, let $s_i(t_i) \geq 2L_0^* \sqrt{2\ln(2/\delta_i(t_i))}/\epsilon_i(t_i)$ for some $\epsilon_i(t_i) > 0$ and $\delta_i(t_i) \in [0, 1]$. For the noise vector $\eta_i(t)$ drawn from a Gaussian distribution $\mathcal{N}(0, s_i(t))^p \in \mathbb{R}^p$ with scale $s_i(t_i)$, and for any $\bar{\delta}_i \in [0, 1]$*

and initial point $\widetilde{\Theta}(0) \in \mathbb{R}^{np}$ *independent of* $\mathcal{S}_i$*, the mechanism* $\mathcal{M}_i(\mathcal{S}_i)$ *is* $(\bar{\epsilon}_i, 1-(1-\bar{\delta}_i)\prod_{t_i=1}^{T_i}(1-\delta_i(t_i)))$*-DP with*

$$\bar{\epsilon}_i = \min\Bigg\{ \sum_{t_i=1}^{T_i} \epsilon_i(t_i), \sum_{t_i=1}^{T_i} \frac{(e^{\epsilon_i(t_i)}-1)\epsilon_i(t_i)}{e^{\epsilon_i(t_i)}+1} + \sqrt{2\epsilon_i(t_i)^2 \log\Big(e + \frac{\sqrt{\sum_{t_i=1}^{T_i}\epsilon_i(t_i)^2}}{\bar{\delta}_i}\Big)}, \\ \sum_{t_i=1}^{T_i} \frac{(e^{\epsilon_i(t_i)}-1)\epsilon_i(t_i)}{e^{\epsilon_i(t_i)}+1} + \sqrt{2\epsilon_i(t_i)^2 \log(1/\bar{\delta}_i)} \Bigg\}.$$

A.2 Proof of Theorem 2

We start by introducing a convenient lemma.

Lemma 2. *For any* $i \in [n]$*,* $\Theta \in \mathbb{R}^{np}$ *and* $d \in \mathbb{R}^p$ *we have:*

$$\mathcal{Q}_{\mathcal{L}}(\Theta + U_i d) \leq \mathcal{Q}_{\mathcal{L}}(\Theta) + d^T[\nabla \mathcal{Q}_{\mathcal{L}}(\Theta)]_i + \frac{L_i}{2}\|d\|^2.$$

Proof. We get this by applying Taylor's inequality to the function

$$\begin{aligned} q_{\Theta}^{x} \quad : \quad & \mathbb{R}^p \to \mathbb{R} \\ & d \mapsto \mathcal{Q}_{\mathcal{L}}(\Theta + U_i d). \end{aligned}$$

□

Recall that the random variable $\eta_i(t) \in \mathbb{R}^p$ represents the noise added by agent $i \in [\![n]\!]$ due to privacy requirements if it wakes up at iteration $t \geq 0$. To simplify notations we denote the scaled version of the noise by $\widetilde{\eta}_i(t) = \mu D_{ii} c_i \eta_i(t)$.

Let i_t be the agent waking up at iteration t. Using Lemma 2, we have:

$$\begin{aligned} \mathcal{Q}_{\mathcal{L}}(\Theta(t+1)) &= \mathcal{Q}_{\mathcal{L}}\left(\Theta(t) - \frac{U_{i_t}}{L_{i_t}}\left([\nabla \mathcal{Q}_{\mathcal{L}}(\Theta(t))]_{i_t} + \widetilde{\eta}_{i_t}(t)\right)\right) \\ &\leq \mathcal{Q}_{\mathcal{L}}(\Theta(t)) - \frac{1}{L_{i_t}}[\nabla \mathcal{Q}_{\mathcal{L}}(\Theta(t))]_{i_t}^T\left([\nabla \mathcal{Q}_{\mathcal{L}}(\Theta(t))]_{i_t} + \widetilde{\eta}_{i_t}(t)\right) \\ &+ \frac{1}{2L_{i_t}}\|[\nabla \mathcal{Q}_{\mathcal{L}}(\Theta(t))]_{i_t} + \widetilde{\eta}_{i_t}(t)\|^2 \\ &= \mathcal{Q}_{\mathcal{L}}(\Theta(t)) - \frac{1}{L_{i_t}}\|[\nabla \mathcal{Q}_{\mathcal{L}}(\Theta(t))]_{i_t}\|^2 + \frac{1}{2L_{i_t}}\|[\nabla \mathcal{Q}_{\mathcal{L}}(\Theta(t))]_{i_t}\|^2 + \frac{1}{2L_{i_t}}\|\widetilde{\eta}_{i_t}(t)\|^2 \\ &\leq \mathcal{Q}_{\mathcal{L}}(\Theta(t)) - \frac{1}{2L_{max}}\|[\nabla \mathcal{Q}_{\mathcal{L}}(\Theta(t))]_{i_t}\|^2 + \frac{1}{2L_{min}}\|\widetilde{\eta}_{i_t}(t)\|^2, \end{aligned}$$

where $L_{min} = \min_{1 \leq i \leq n} L_i$.

Recall that under our Poisson clock assumption, each agent is equally likely to wake up at any step t. Subtracting $\mathcal{Q}_{\mathcal{L}}^*$ and taking the expectation with respect to i_t on both sides, we thus get:

$$\begin{aligned} \mathop{\mathbb{E}}_{i_t}[\mathcal{Q}_{\mathcal{L}}(\Theta(t+1))] - \mathcal{Q}_{\mathcal{L}}^* &\leq \mathcal{Q}_{\mathcal{L}}(\Theta(t)) - \mathcal{Q}_{\mathcal{L}}^* - \frac{1}{2L_{max}}\frac{1}{n}\sum_{j=1}^{n}\|[\nabla \mathcal{Q}_{\mathcal{L}}(\Theta(t))]_j\|^2 \\ &+ \frac{1}{2L_{min}}\frac{1}{n}\sum_{j=1}^{n}\|\widetilde{\eta}_j(t)\|^2 \\ &= \mathcal{Q}_{\mathcal{L}}(\Theta(t)) - \mathcal{Q}_{\mathcal{L}}^* - \frac{1}{2nL_{max}}\|\nabla \mathcal{Q}_{\mathcal{L}}(\Theta(t))\|^2 + \frac{1}{2nL_{min}}\|\widetilde{\eta}(t)\|^2, \end{aligned} \tag{9}$$

where $\widetilde{\eta}(t) = [\widetilde{\eta}_1(t); \ldots, ; \widetilde{\eta}_n(t)] \in \mathbb{R}^{np}$.

For convenience, let us define $P_t = \mathbb{E}[\mathcal{Q}_{\mathcal{L}}(\Theta(t))] - \mathcal{Q}^*_{\mathcal{L}}$ where $\mathbb{E}[\cdot]$ denotes the expectation with respect to all variables $\{i_t\}_{t\geq 0}$ and $\{\eta(t)\}_{t\geq 0}$. Using (9) we thus have:

$$P_{t+1} \leq P_t - \frac{1}{2nL_{max}} \mathbb{E}[\|\nabla \mathcal{Q}_{\mathcal{L}}(\Theta(t))\|^2] + \frac{1}{2nL_{min}} \mathbb{E}[\|\widetilde{\eta}(t)\|^2] \tag{10}$$

Recall that $\mathcal{Q}_{\mathcal{L}}$ is σ-strongly convex, i.e. for any $\Theta, \Theta' \in \mathbb{R}^{np}$ we have:

$$\mathcal{Q}_{\mathcal{L}}(\Theta') \geq \mathcal{Q}_{\mathcal{L}}(\Theta) + \nabla \mathcal{Q}_{\mathcal{L}}(\Theta)^T(\Theta' - \Theta) + \frac{\sigma}{2}\|\Theta' - \Theta\|_2^2.$$

We minimize the above inequality on both sides with respect to Θ'. We obtain that $\Theta' = \Theta^*$ minimizes the left-hand side, while $\Theta' = \Theta - \nabla \mathcal{Q}_{\mathcal{L}}(\Theta)/\sigma$. We thus have for any Θ:

$$\mathcal{Q}^*_{\mathcal{L}} \geq \mathcal{Q}_{\mathcal{L}}(\Theta) - \frac{1}{\sigma}\nabla \mathcal{Q}_{\mathcal{L}}(\Theta)^T \nabla \mathcal{Q}_{\mathcal{L}}(\Theta) + \frac{1}{2\sigma}\|\nabla \mathcal{Q}_{\mathcal{L}}(\Theta)\|_2^2 = \mathcal{Q}_{\mathcal{L}}(\Theta) - \frac{1}{2\sigma}\|\nabla \mathcal{Q}_{\mathcal{L}}(\Theta)\|^2.$$

Using the above inequality to bound $\|\nabla \mathcal{Q}_{\mathcal{L}}(\Theta(t))\|^2$ in (10), we obtain:

$$P_{t+1} \leq P_t - \frac{\sigma}{nL_{max}} P_t + \frac{1}{2nL_{min}} \mathbb{E}[\|\widetilde{\eta}(t)\|^2] = \left(1 - \frac{\sigma}{nL_{max}}\right) P_t + \frac{1}{2nL_{min}} \mathbb{E}[\|\widetilde{\eta}(t)\|^2].$$

A simple recursion on P_t gives:

$$P_t \leq \left(1 - \frac{\sigma}{nL_{max}}\right)^t P_0 + \frac{1}{2nL_{min}} \sum_{t'=0}^{t-1} \left(1 - \frac{\sigma}{nL_{max}}\right)^{t'} \mathbb{E}[\|\widetilde{\eta}(t')\|^2]. \tag{11}$$

For any $t \geq 0$ and $i \in [\![n]\!]$, the entries of $\eta_i(t)$ are drawn from independent Laplace distributions with mean 0 and scale $s_i(t)$, hence we have:

$$\mathbb{E}[\|\widetilde{\eta}(t)\|^2] = \mathbb{E}\left[\sum_{i=1}^n \|\mu D_{ii} c_i \eta_i(t)\|^2\right] = \sum_{i=1}^n 2(\mu D_{ii} c_i s_i(t))^2.$$

This concludes the proof.

A.3 Proof of Proposition 2

We start the proof with the following lemma.

Lemma 3. *Let $C = 1 - \sigma/nL_{max}$. Assume that for any $i \in [\![n]\!]$, we have $s_i(t_i) = \frac{2L_0}{\epsilon_i(t_i)m_i}$ for $t_i \in \mathcal{T}_i$ as in Theorem 1. Given a total number of iterations T and the overall privacy budgets $\bar{\epsilon}_1, \ldots, \bar{\epsilon}_n > 0$ of each agent, the following privacy parameters minimize the utility loss:*

$$\epsilon_i^*(t) = \frac{\sqrt[3]{C} - 1}{\sqrt[3]{C^T} - 1} \sqrt[3]{C^t} \bar{\epsilon}_i, \quad \forall i \in [\![n]\!], 0 \leq t \leq T-1.$$

Proof. Denote $A_i = \frac{\mu c_i D_{ii} L_0}{m_i}$. As the first part of the upper-bound of the utility loss in Theorem 2 does not

depend on the $\epsilon_i(t)$'s, we need to find the $\epsilon_i(t)$'s which minimize the following quantity:

$$\begin{aligned}\min \sum_{t=0}^{T-1}\sum_{i=1}^{n}\Big(1-\frac{\sigma}{nL_{max}}\Big)^t\big(\mu c_i D_{ii} s_i(t)\big)^2 &= \min \sum_{t=0}^{T-1}\sum_{i=1}^{n}\Big(1-\frac{\sigma}{nL_{max}}\Big)^t\Big(\frac{2\mu c_i D_{ii} L_0}{\epsilon_i(t_i) m_i}\Big)^2 \\ &= \min \sum_{t=0}^{T-1} C^t\Big(\sum_{i=1}^{n}\frac{A_i^2}{\epsilon_i^2(t)}\Big) \\ &= \min \sum_{i=1}^{n}\Big(\sum_{t=0}^{T-1}\frac{A_i^2 C^t}{\epsilon_i^2(t)}\Big), \end{aligned} \tag{12}$$

under the constraints: $\forall i, \epsilon_i(t) \geq 0$ and $\sum_{t=0}^{T-1}\epsilon_i(t) = \bar{\epsilon}_i$. As the agents are independent, we can solve the above optimization problem separately for each agent i to minimize $\sum_{t=0}^{T-1}\frac{A_i^2 C^t}{(\epsilon_i(t))^2}$ under the constraint $\sum_{t=0}^{T-1}\epsilon_i(t) = \bar{\epsilon}_i$. Denote $\epsilon_i(t) = x_{it}$. We have $x_{i0} = \bar{\epsilon}_i - \sum_{t=1}^{T-1} x_{it}$. Replacing x_{i0} in the objective function of Eq. 12, we can write the objective function as follows:

$$\forall i: \quad F_i(x_{i1}, \cdots, x_{iT-1}) = \frac{A_i^2}{(\bar{\epsilon}_i - \sum_{t=1}^{T-1} x_{it})^2} + \sum_{t=1}^{T-1}\frac{A_i^2 C^t}{x_{it}^2}.$$

The problem is thus equivalent to finding $\min F_i(x_{i1}, \cdots, x_{it-1})\ \ \forall i$ under the previous constraints. To find the optimal x_{it} we find set its partial derivative $\frac{\partial F_i}{\partial x_{it}}$ to 0:

$$\frac{\partial F_i}{\partial x_{it}} = \frac{-2A_i^2 C^t}{x_{it}^3} + \frac{2A_i^2}{(\bar{\epsilon}_i - \sum_{t=1}^{T-1} x_{it})^3} = 0.$$

Hence the value of x_{it} satisfies:

$$\forall t \in [1 : T-1] : x_{it} = \sqrt[3]{C^t}\Big(\bar{\epsilon}_i - \sum_{j=1}^{T-1} x_{ij}\Big). \tag{13}$$

For a fixed i, after summing up all x_{it}s, we get:

$$\sum_{j=1}^{T-1} x_{ij} = \Big(\sum_{j=1}^{T-1}\sqrt[3]{C^j}\Big)\Big(\bar{\epsilon}_i - \sum_{j=1}^{T-1} x_{ij}\Big) = \frac{\sqrt[3]{C^T} - \sqrt[3]{C}}{\sqrt[3]{C} - 1}\Big(\bar{\epsilon}_i - \sum_{j=1}^{T-1} x_{ij}\Big).$$

And hence:

$$x_{i0} = \bar{\epsilon}_i - \sum_{j=1}^{T-1} x_{ij} = \frac{\sqrt[3]{C} - 1}{\sqrt[3]{C^T} - 1}\bar{\epsilon}_i.$$

Using Eq. 13, we thus finally get:

$$\forall i, t : \epsilon_i^*(t) = \frac{\sqrt[3]{C} - 1}{\sqrt[3]{C^T} - 1}\sqrt[3]{C^t}\bar{\epsilon}_i$$

□

The noise allocation strategy given by the above lemma optimizes the utility loss, which is an expectation with respect to the clock ticks. Hence $\epsilon_i^*(t)$ gives the amount of noise that an agent should add *in expectation* at a given global iteration t. Note that we have $\sum_{t=0}^{T-1}\epsilon_i^*(t) = \bar{\epsilon}_i$, hence the budget is matched exactly in expectation. For a particular run of the algorithm however, each agent i only wakes up at a subset $\mathcal{T}_i$ of all iterations and only uses its privacy budget at these iterations $t \in \mathcal{T}_i$. Thus, for a single run, the above strategy does not use up the entire privacy budget $\bar{\epsilon}_i$.

If we instead condition on a particular schedule, i.e. each agent i knows $\mathcal{T}_i$ in advance, we can appropriately renormalize the privacy parameters to make sure that the agents truly utilize their entire privacy budget. The overall privacy parameter for agent i with optimal noise allocation as in Lemma 3 given $\mathcal{T}_i$ is as follows:

$$\sum_{t \in \mathcal{T}_i} \epsilon_i^*(t) = \lambda_{\mathcal{T}_i}(i) \cdot \bar{\epsilon}_i.$$

Hence we can set

$$\sum_{t \in \mathcal{T}_i} \epsilon_i(t) = \sum_{t \in \mathcal{T}_i} \frac{\epsilon_i^*(t)}{\lambda_{\mathcal{T}_i}(i)} = \bar{\epsilon}_i.$$

This shows that the privacy parameters defined in Proposition 2 fulfills the overall privacy budget of an agent i during the iterations that agent i wakes up. Furthermore, for $\epsilon_i^*(t)$ defined in Lemma 3, $\sum_{t=0}^{T-1} \epsilon_i^*(t) = \bar{\epsilon}_i \quad \forall i \in [\![n]\!]$. We then conclude that $\lambda_{\mathcal{T}_i}(i) \leq 1$ as $\mathcal{T}_i \subseteq [\![T]\!]$. The additional noise to provide $\epsilon_i(t)$ differential privacy (as in Proposition 3) is thus lower than the additional noise to provide $\epsilon_i^*(t)$ differential privacy in Lemma 3.

B Propagation of (Private) Local Models

In this section, we give some details on an interesting special case of our framework. The idea is to smooth models that are pre-trained locally by each agent. Formally, the objective function to minimize is as follows:

$$\mathcal{Q}_{MP}(\Theta) = \frac{1}{2}\Bigg(\sum_{i<j}^{n} W_{ij} \|\Theta_i - \Theta_j\|^2 + \mu \sum_{i=1}^{n} D_{ii} c_i \|\Theta_i - \Theta_i^{loc}\|^2 \Bigg), \tag{14}$$

which is a special case of (3) when we set $\mathcal{L}_i(\Theta_i; \mathcal{S}_i) = \frac{1}{2}\|\Theta_i - \Theta_i^{loc}\|^2$ with $\Theta_i^{loc} \in \arg\min_{\theta \in \mathbb{R}^p} \sum_{j=1}^{m_i} \ell(\theta; x_i^j, y_i^j) + \lambda_i \|\theta\|^2$. Each $\mathcal{L}_i$ is 1-strongly convex in Θ_i with 1-Lipschitz continuous gradient, hence we have $L_i = D_{ii}(1+\mu c_i)$ and $\sigma \geq \mu \min_{1 \leq i \leq n}[D_{ii} c_i]$.

Developing the update step (5) for this special case, we get:

$$\Theta_i(t+1) = \frac{1}{1+\mu c_i}\Bigg(\sum_{j \in \mathcal{N}_i} \frac{W_{ij}}{D_{ii}} \Theta_j(t) + \mu c_i \Theta_i^{loc} \Bigg). \tag{15}$$

Because the objective function (14) is quadratic, (15) corresponds to the *exact minimizer* of $\mathcal{Q}_{MP}$ along the block coordinate direction Θ_i. Hence $\Theta_i(t+1)$ does not depend on $\Theta_i(t)$, but only on the solitary and neighboring models.

It turns out that we recover the update rule proposed by [25] specifically for model propagation. Our block coordinate algorithm thus generalizes their approach to general local loss functions and allows to obtain convergence rate instead of only asymptotic convergence.

Private setting In the above objective, the interaction with each local dataset $\mathcal{S}_i$ is only through the pre-trained model Θ_i^{loc} learned by minimizing the (regularized) empirical risk as denoted in (1). Therefore, if we generate a DP version $\widetilde{\Theta}_i^{loc}$ of Θ_i^{loc}, we can run the non-private coordinate descent algorithm (15) using $\widetilde{\Theta}_i^{loc}$ instead of Θ_i^{loc} without degrading the privacy guarantee. We can thus avoid the dependency on the number of iterations of the coordinate descent algorithm. Several well-documented methods for an agent to generate a DP version of its local model Θ_i^{loc} exist in the literature, under mild assumptions on the loss function and regularizer. One may for instance use output or objective perturbation [4].

Recall that Theorem 2 emphasizes that it is beneficial to have a good warm start point $\Theta(0)$ for the general algorithm: the smaller $\mathcal{Q}_{\mathcal{L}}(\Theta(0)) - \mathcal{Q}_{\mathcal{L}}^{\star}$, the less iterations needed to decrease the optimization error to the

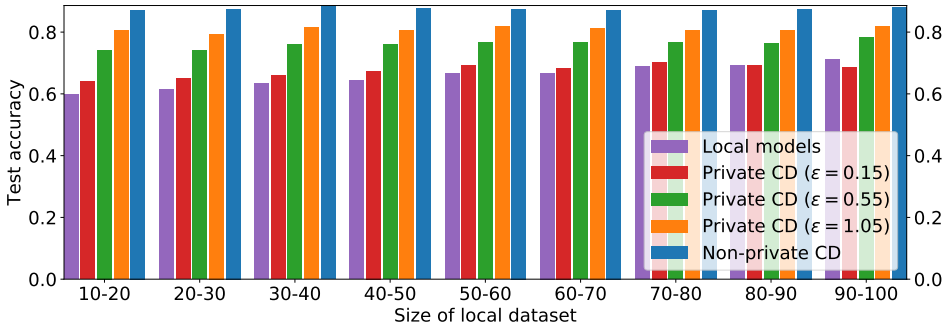


Figure 3: Final test accuracy (averaged over 5 runs) per local dataset size for dimension $p = 100$ and several privacy regimes. Best seen in color.

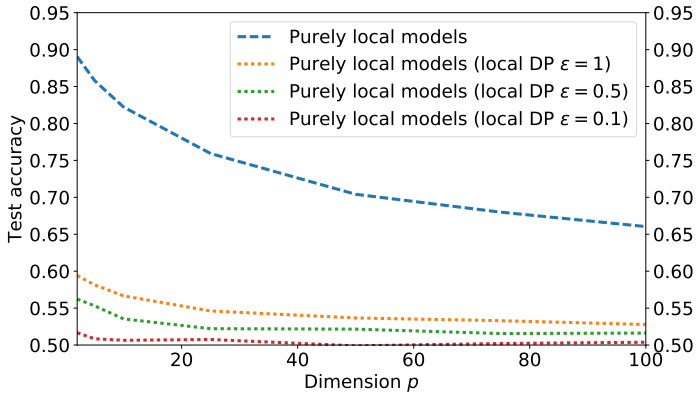


Figure 4: Test accuracy (averaged over 5 runs) of purely local models learned from perturbed data (local DP) w.r.t. to data dimension, for different privacy regimes. Best seen in color.

desired precision and hence the less noise added due to privacy. However, to ensure the overall privacy of the algorithm, $\Theta(0)$ must be also be differentially private. In light of the discussion above, we can use the private model propagation solution as a good private warm start.

C Additional Experimental Results

In this section, we present additional experimental results which complement those displayed in the main text.

Test accuracy w.r.t. size of local dataset Figure 3 shows the test accuracy of our algorithm under different private regimes depending on the size of the local dataset of an agent. First, we can see that all agents (irrespective of their dataset size) benefit from private collaborative learning, in the sense that their final accuracy is larger than that of their purely local model. This is important as it gives an incentive to all agents (including those with larger datasets) to collaborate. Second, the algorithm effectively corrects for the imbalance in dataset size: agents with less data generally get a larger boost in accuracy and can almost catch up with better-endowed agents (which also get an improvement, albeit smaller).

Local Differential Privacy baseline As mentioned in Section 2.2, local Differential Privacy [6, 15] consists in adding noise to each data point itself (proportional to the sensitivity of its features) so that the resulting perturbed point does not reveal too much about the original point. Local DP can be used trivially in the distributed/decentralized setting as it is purely local. However it is also a very conservative approach as it is agnostic to the type of analysis done on the data (the perturbed dataset can be released publicly). Figure 4 shows the accuracy of purely local models learned after applying $(\epsilon, 0)$-local DP to each local dataset. As expected, the loss in accuracy compared to purely local models learned from unperturbed data is huge. We were not able to improve significantly over these models by applying our collaborative learning algorithm on the perturbed data, confirming that it contains mostly random noise. This confirms the relevance of our private collaborative algorithm based on perturbing the updates rather than the data itself.