Personalized and Private Peer-to-Peer Machine Learning

Aurélien Bellet
INRIA

Rachid Guerraoui
EPFL

Mahsa Taziki
EPFL

Marc Tommasi
Université de Lille

Abstract

The rise of connected personal devices together with privacy concerns call for machine learning algorithms capable of leveraging the data of a large number of agents to learn personalized models under strong privacy requirements. In this paper, we introduce an efficient algorithm to address the above problem in a fully decentralized (peer-to-peer) and asynchronous fashion, with provable convergence rate. We show how to make the algorithm differentially private to protect against the disclosure of information about the personal datasets, and formally analyze 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 can significantly improve over models learned in isolation.

1 Introduction

Connected personal devices are now widespread: they can collect and process increasingly large and sensitive user data. As a concrete example, consider the health domain. Smart watches can record cardiac activities, mobile apps encourage users to participate to studies (about depression, concussion, etc.),[1] and recent painless sensors can replace a finger prick for blood glucose testing (Cappon et al., 2017). Such information can be leveraged through machine learning to provide useful personalized services (e.g., personalized treatments) to the user/patient. A common practice is to centralize data from all devices on an external server for batch processing, sometimes without explicit consent from users and with little oversight. While this data concentration is ideal for the utility of the learning process, it raises serious privacy concerns and opens the door to potential misuse (e.g., exploitation for the purpose of recruitment, insurance pricing or granting loans). Therefore, in applications where the data is considered too sensitive to be shared (due to legislation or because the user opts out), one has to learn on each device separately without taking advantage of the multiplicity of data sources (e.g., information from similar users). This preserves privacy but leads to poor accuracy, in particular for new or moderately active users who have not collected much data.

Instead of the above two extreme approaches, our goal is to design a solution allowing a large number of users (agents) to collaborate so as to learn more accurate personalized models while ensuring that their data stay on their local device and that the algorithm does not leak sensitive information to others. We consider a *fully decentralized* solution where agents operate asynchronously and communicate over a network in a peer-to-peer fashion, without any central entity to maintain a global state of the system or even to coordinate the protocol. The network acts as a communication network but also models similarities between users. While a decentralized architecture may be the only available option in some applications (e.g., IoT), it also provides interesting benefits when a more traditional distributed (master/slave) architecture could be used. In particular, peer-to-peer algorithms provide scalability-by-design to large sets of devices thanks to the locality of their updates (Kermarrec and Taïani, 2015). For instance, it was recently shown that fully decentralized learning algorithms can perform better than their distributed counterparts because they avoid a communication bottleneck at the master node (Lian et al., 2017). Finally, a decentralized architecture intrinsically provides some security guarantees as it becomes much more difficult for any party (or any external adversary) to observe the full state of the system.

The problem of decentralized collaborative learning of personal models has been recently considered by Vanhaesebrouck et al. (2017), but they did not consider

[1] See e.g., `https://www.apple.com/researchkit/`

Proceedings of the 21^{st} International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain. PMLR: Volume 84.

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 Dwork and Roth, 2014; Chaudhuri et al., 2011; Bassily et al., 2014, 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 privacy-preserving distributed learning (see e.g., Pathak et al., 2010; Rajkumar and Agarwal, 2012; Shokri and Shmatikov, 2015; Huang et al., 2015) rely on 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 the above 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 to address the problem in the non-private setting. Taking advantage of the structure of the problem, this algorithm has simple updates and provable convergence rates, improving upon the previous work of Vanhaesebrouck et al. (2017). 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 provide a formal analysis of the utility loss due to privacy. Third, we conduct experiments on synthetic and real-world data 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. In Section 2, we describe the problem setting and presents our decentralized algorithm for the non-private case. Section 3 introduces a differentially private version and analyzes the trade-off between utility and privacy. In Section 4, we discuss some related work on decentralized and private learning. Finally, Section 5 is dedicated to numerical experiments. Detailed proofs can be found in the supplementary material.

2 Peer-to-Peer Personalized Learning with Coordinate Descent

We start by formally describing the learning problem that we address in this paper.

2.1 Problem Setting

We 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 the setting where agent i must learn on its own, a standard approach is to select the model minimizing the local (potentially regularized) empirical loss:

$$\theta_i^{loc} \in \operatorname*{arg\,min}_{\theta\in\mathbb{R}^p} \underbrace{\Big[\frac{1}{m_i} \sum_{j=1}^{m_i} \ell(\theta; x_i^j, y_i^j) + \lambda_i \|\theta\|^2 \Big]}_{\mathcal{L}_i(\theta; \mathcal{S}_i)}, \quad (1)$$

with $\lambda_i \geq 0$. 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 $[\![n]\!] = \{1, \dots, n\}$ and $G = ([\![n]\!], E, W)$ be a weighted connected graph over the set of agents where $E \in [\![n]\!] \times [\![n]\!]$ 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., Evgeniou and Pontil, 2004; Vanhaesebrouck et al., 2017), 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. These pairwise similarity weights may be derived from user profiles (e.g., in the health domain: weight, size, diabetes type, etc.) or directly from the local datasets, and can be computed in a private way (see e.g., Goethals et al., 2004; Alaggan et al., 2011).

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 operates without synchronizing with other agents. Overall, the problem can thus be seen as a multi-task learning problem over a large number of tasks (agents) with imbalanced training sets, and which must be solved in a fully decentralized way.

2.2 Objective Function

Our goal is to jointly learn the models of the agents by leveraging both their local datasets and the similarity information embedded in the network graph.

Following a well-established principle in the multi-task learning literature (Evgeniou and Pontil, 2004; Maurer, 2006; Dhillon et al., 2011), we use graph regularization to favor models that vary smoothly on the graph. Specifically, representing the set of all models $\Theta_i \in \mathbb{R}^p$ as a stacked vector $\Theta = [\Theta_1; \ldots; \Theta_n] \in \mathbb{R}^{np}$, the objective function we wish to minimize is given by

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

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.[2] Minimizing (2) 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" others with their own inaccurate model. Note that the objective (2) includes the two extreme cases of learning purely local models as in (1) (when $\mu \to \infty$) and learning a single global model (for $\mu \to 0$).

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.[3] If we further assume that each $\mathcal{L}_i$ is σ_i^{loc}-strongly convex 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 $\mathcal{Q}_{\mathcal{L}}(\Theta)$ w.r.t. the variables in Θ_i is given by

$$[\nabla\mathcal{Q}_{\mathcal{L}}(\Theta)]_i = D_{ii}(\Theta_i + \mu c_i \nabla\mathcal{L}_i(\Theta_i; \mathcal{S}_i)) - \textstyle\sum_{j\in\mathcal{N}_i} W_{ij}\Theta_j. \tag{3}$$

We define the matrices $U_i \in \mathbb{R}^{np\times p}$, $i \in [\![n]\!]$, such that $(U_1, \ldots, U_n) = I_{np}$. 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 $L_i = D_{ii}(1 + \mu c_i L_i^{loc})$. We denote $L_{min} = \min_i L_i$ and $L_{max} = \max_i L_i$.

2.3 Non-Private Decentralized Algorithm

For ease of presentation, we first present a non-private decentralized algorithm. Note that this is interesting in its own right as the proposed solution improves upon the algorithm previously proposed by Vanhaesebrouck et al. (2017), see Section 4 for a discussion.

Time and communication models. Our goal is to minimize the objective function (2) in a fully decentralized manner. Specifically, we operate in the asynchronous time model (Boyd et al., 2006): each agent has a *local* clock ticking at the times of a rate 1 Poisson process, and wakes up when it ticks. This is in contrast to the synchronous model where agents wake up jointly according to a global clock (and thus need to wait for everyone to finish each round). 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 (which is unknown to the agents). For communication, we rely on a broadcast-based model (Aysal et al., 2009; Nedic, 2011) where agents send messages to all their neighbors at once (without expecting a reply). This model is very appealing in wireless distributed systems, as sending a message to all neighbors has the same cost as sending to a single neighbor.

Algorithm. We propose a decentralized coordinate descent algorithm to minimize (2). We initialize the algorithm with an arbitrary set of local models $\Theta(0) = [\Theta_i(0); \ldots; \Theta_n(0)]$. At time step t, an agent i wakes up. Two consecutive actions are then performed by i:

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

$$\begin{aligned}\Theta_i(t+1) &= \Theta_i(t) - (1/L_i)[\nabla\mathcal{Q}_{\mathcal{L}}(\Theta(t))]_i \\ &= (1-\alpha)\Theta_i(t) + \alpha\big(\textstyle\sum_{j\in\mathcal{N}_i} \frac{W_{ij}}{D_{ii}}\Theta_j(t) \\ &\quad - \mu c_i \nabla\mathcal{L}_i(\Theta_i(t); \mathcal{S}_i)\big),\end{aligned} \tag{4}$$

  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 $\mathcal{N}_i$.

The update step (4) 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 $j \in \mathcal{N}_i$. Note that 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

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

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

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).[4] 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.

Convergence analysis. 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 (Wright, 2015). 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 $T > 0$, 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:*

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

Proof. This follows from a slight adaptation of the proof of Wright (2015) (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 3.2) 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 Wright (2015) for details.*

Proposition 1 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 operate asynchronously and in parallel). 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 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 particular agent), and (ii) a low-confidence, poorly connected agent (hence converging slowly).

[4] 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.

3 Differentially Private Algorithm

As described above, the algorithm introduced in the previous section has many interesting properties. However, 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 define our privacy model and introduce an appropriate scheme to make our algorithm differentially private. We study its utility loss and the trade-off between utility and privacy.

3.1 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).[5]

To formally define this privacy model, we rely on the notion of Differential Privacy (DP) (Dwork, 2006), which 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:

$$Pr(\mathcal{M}(\mathcal{S}) \in \mathcal{O}) \leq e^{\epsilon} Pr(\mathcal{M}(\mathcal{S}') \in \mathcal{O}) + \delta, \quad (5)$$

where the probability is over the randomness of the mechanism. At a high level, one can see (5) 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.

[5] 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.

The composition of several DP mechanisms remains DP, albeit a graceful degradation in the parameters (see Dwork et al., 2010; Kairouz et al., 2015, for strong composition results). We refer to Dwork and Roth (2014) for more details on DP.

In our setting, following the notations of (5), 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 preserving privacy.

3.2 Privacy-Preserving Scheme

The privacy-preserving version of our algorithm consists in replacing the update step in (4) by the following one (assuming that at time t agent i wakes up):

$$\begin{aligned}\widetilde{\Theta}_i(t+1) = \quad & (1-\alpha)\widetilde{\Theta}_i(t) + \alpha\big(\textstyle\sum_{j\in\mathcal{N}_i} \frac{W_{ij}}{D_{ii}}\widetilde{\Theta}_j(t) \\ & -\mu c_i(\nabla\mathcal{L}_i(\widetilde{\Theta}_i(t);\mathcal{S}_i) + \eta_i(t))\big),\end{aligned} \tag{6}$$

where $\eta_i(t) \sim \mathit{Laplace}(0, s_i(t))^p \in \mathbb{R}^p$ is a noise vector drawn from a Laplace distribution with finite scale $s_i(t) \geq 0$.[6] The difference with the non-private update is that agent i adds appropriately scaled Laplace noise 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. Note that for full generality, we allow the noise to potentially depend on the global iteration number t, as we will see towards the end of this section that it opens interesting perspectives.

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 shows how to scale the noise at each iteration, $s_i(t_i)$, so as to provide the 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*

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

Remark 2. *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 Kairouz et al., 2015, for a discussion of the trade-offs in the composition of DP mechanisms). 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 classic property of DP, but it is especially appealing in our collaborative formulation as the confidence weights c_i's tune down the importance of agents with small datasets (preventing their noisy information to spread) and give more importance to agents with larger datasets (who propagate useful information). Our next result quantifies how the added noise affects the convergence.

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

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

This result shows that the 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 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, this additive error converges to a finite number as $T \to \infty$. The number of iterations T rules the trade-off between the two terms. We give more details in the supplementary material and study this numerically in Section 5.

[6] We use the convention $\mathit{Laplace}(0,0) = 0$ w.p. 1.

In practical scenarios, each agent i has an 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. A simple and practical strategy is to distribute the budget equally across the T_i iterations. Yet, 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 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 optimize the utility loss while ensuring the budget $\bar{\epsilon}_i$ is matched exactly:*

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

The above noise allocation policy requires agents to know in advance the schedule and 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 3. *Theorem 2 implies that a good warm start point $\Theta(0)$ is beneficial. However, $\Theta(0)$ must be DP. In the supplementary material, we show how to generate a private warm start based on propagating perturbed versions of purely local models in the network.*

4 Related Work

Decentralized ML. Most of the work in fully 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 (Nedic and Ozdaglar, 2009; Ram et al., 2010; Duchi et al., 2012; Wei and Ozdaglar, 2012; Colin et al., 2016). In contrast, we tackle the case where agents have distinct objectives.

The work of Vanhaesebrouck et al. (2017) recently studied the problem of decentralized learning of personalized models and is hence more closely related to our approach, but they did not consider any privacy constraints. At the cost of introducing many auxiliary variables, they cast the objective function as a partial consensus problem over the network which can be solved using a decentralized gossip ADMM algorithm (Wei and Ozdaglar, 2013). Our contributions extend over this previous work in several respects: (i) we propose a simpler block coordinate descent algorithm with linear convergence rate, which also proves to be much faster in practice (see Section 5), (ii) we design a differentially private algorithm and provide an analysis of the resulting privacy/utility trade-off, and (iii) we present an evaluation of our approach on real data (in addition to synthetic experiments).

DP in distributed learning. Differential Privacy has been mostly considered in the context where a "trusted curator" has access to all data. Existing DP schemes for learning in this setting typically rely on the addition of noise to the learned model (output perturbation) or to the objective function itself (objective perturbation), see for instance Chaudhuri et al. (2011).

The private multi-party setting, in which sensitive datasets are distributed across multiple data owners, is known to be harder (McGregor et al., 2010) and has been less studied in spite of its relevance for many applications. Local DP (Duchi et al., 2012; Kairouz et al., 2016), consisting in locally perturbing the data points themselves before releasing them, often achieves poor accuracy (especially when local datasets are small). In the master/slave setting, DP algorithms have been introduced to learn a private global model, either by aggregating models trained locally by each party (Pathak et al., 2010; Hamm et al., 2016) or by perturbing the gradients and/or the objective in a distributed gradient descent algorithm (Song et al., 2013; Rajkumar and Agarwal, 2012; Shokri and Shmatikov, 2015). Some of these approaches rely on the assumption that local datasets are drawn from the same global distribution. The work of Huang et al. (2015) considers the decentralized setting, using a privacy model similar to ours. However, they still learn a single global model and it is not clear how their algorithm and analysis can be adapted to our multi-task problem. Moreover, their approach is synchronous, relies on additional assumptions (e.g., bounded second derivatives) and does not have established convergence rates.

We conclude this section by briefly mentioning the recent work of Hitaj et al. (2017) describing an attack against differentially private collaborative deep learning approaches (such as Shokri and Shmatikov, 2015). They show how a malicious participant may actively train a Generative Adversarial Network (GAN) which is able to generate prototypical examples of a class held by another agent. While this does not violate DP, it can still constitute a privacy breach in applications where a class distribution itself is considered pri-

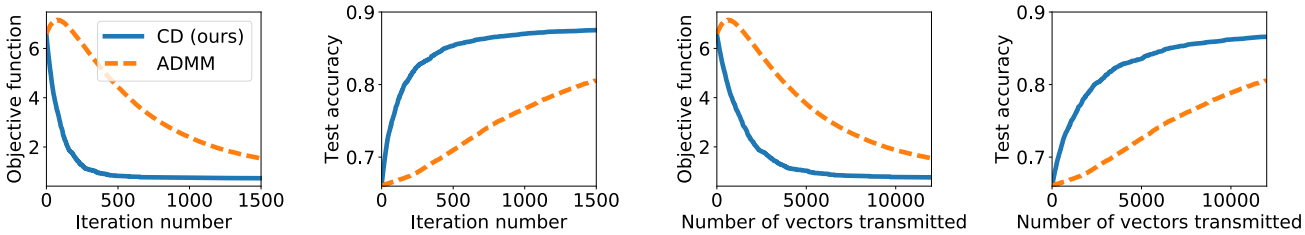
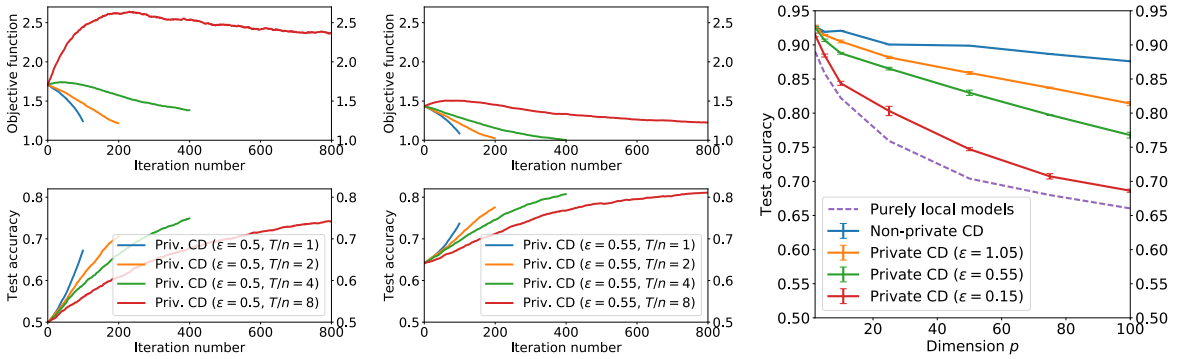


Figure 1: Our Coordinate Descent (CD) algorithm compared to the existing ADMM algorithm.



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

Figure 2: Linear classification 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.

vate. We believe that the key features of our approach, namely the fully decentralized architecture and the graph regularization over personal models, can significantly limit the effectiveness of the above attack. We leave a careful study of this question for future work.

5 Numerical Experiments

5.1 Linear Classification

We first conduct experiments on a linear classification task introduced by Vanhaesebrouck et al. (2017). We briefly recall the setup. Consider a set of $n = 100$ agents. Each of these agents has a target linear separator in $\mathbb{R}^p$ (unknown to the agent). 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 (drawn uniformly between 10 and 100), where each 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. We use 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 points.

Non-private setting: CD vs ADMM. We start by comparing our Coordinate Descent (CD) algorithm to the ADMM algorithm proposed by Vanhaesebrouck et al. (2017) for the non-private setting. Both algorithms are fully decentralized and asynchronous, but our algorithm relies on broadcast (one-way communication from an agent to all neighbors) while the ADMM algorithm is gossip-based (two-way communication between a node and 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 it-

	Purely local models	Non-priv. CD	Priv. $\bar{\epsilon} = 1$	Priv. $\bar{\epsilon} = 0.5$	Priv. $\bar{\epsilon} = 0.1$
Per-user test RMSE	1.2834	0.9502	0.9527	0.9545	0.9855

Table 1: Per-user test RMSE (averaged over users and 5 random runs) on MovieLens-100K.

eration (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 the associated edge is activated. In contrast, our algorithm does not require auxiliary variables.

Private setting. In this experiment, each agent has the same overall privacy budget $\bar{\epsilon}_i = \bar{\epsilon}$. 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 can lead to lower values of the objective with less iterations (as suggested again by Theorem 2). The gain in test accuracy here is significant.

Figure 2(c) shows results for problems of increasing difficulty (by varying the dimension p) with various privacy budgets. 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 ($\bar{\epsilon} = 0.15$), the resulting models significantly outperform the purely local models (a perfectly private baseline). As can be seen in the supplementary material, all agents (irrespective of their dataset size) get an improvement in test accuracy. This improvement is especially large for users with small local datasets, effectively correcting for the imbalance in dataset size. We also show that perturbing the data itself, a.k.a. local DP (Duchi et al., 2012; Kairouz et al., 2016), leads to very inaccurate models.

5.2 Recommendation Task

To illustrate our approach on real-world data, we use MovieLens-100K,[7] a popular benchmark dataset for recommendation systems which consists of 100,000 ratings given by $n = 943$ users over a set of $n_{items} = 1682$ movies. In our setting, each user i corresponds to an agent who only has access to its own ratings $r_{ij_1}, \ldots, r_{ij_{m_i}} \in \mathbb{R}$, where $j_1, \ldots, j_{m_i}$ denote the indices of movies rated by agent i. Note that there is a large imbalance across users: on average, a user has rated 106 movies but the standard deviation is large ($\simeq 100$), leading to extreme values (min=20, max=737). For simplicity, we assume that a common feature representation $\phi_j \in \mathbb{R}^p$ for each movie $j \in [\![n_{items}]\!]$ is known a priori by all agents ($p = 20$ in our experiments). The goal of each agent i is to learn a model $\theta_i \in \mathbb{R}^p$ such that $\theta_i^T \phi_j$ is a good estimate for the rating that i would give to movie j, as measured by the quadratic loss $\ell(\theta; \phi, r) = (\theta^T \phi - r)^2$. This is a very simple model: we emphasize that our goal is not to obtain state-of-the-art results on this dataset but to show that our approach can be used to improve upon purely local models in a privacy-preserving manner. For each agent, we randomly sample 80% of its ratings to serve as training set and use the remaining 20% as test set. The network is obtained by setting $W_{ij} = 1$ if agent i is within the 10-nearest neighbors of agent j (or vice versa) according to the cosine similarity between their training ratings, and $W_{ij} = 0$ otherwise. Due to the lack of space, additional details on the experimental setup are deferred to the supplementary material.

Table 1 shows the test RMSE (averaged over users) for different strategies (for private versions, we use $\bar{\delta}_i = \exp(-5)$ as in the previous experiment). While the purely local models suffer from large error due to data scarcity, our approach can largely outperform this baseline in both the non-private and private settings.

6 Conclusion

We introduced and analyzed an efficient algorithm for personalized and peer-to-peer machine learning under privacy constraints. We argue that this problem is becoming more and more relevant as connected objects become ubiquitous. Further research is needed to address dynamic scenarios (agents join/leave during the execution, data is collected on-line, etc.). We will also explore the use of secure multiparty computation and homomorphic encryption as an alternative/complementary approach to DP in order to provide higher accuracy at the cost of more computation.

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.

[7] `https://grouplens.org/datasets/movielens/100k/`

References

Alaggan, M., Gambs, S., and Kermarrec, A.-M. (2011). Private Similarity Computation in Distributed Systems: from Cryptography to Differential Privacy. In *OPODIS*, pages 357–377.

Aysal, T. C., Yildiz, M. E., Sarwate, A. D., and Scaglione, A. (2009). Broadcast Gossip Algorithms for Consensus. *IEEE Transactions on Signal Processing*, 57(7):2748–2761.

Bassily, R., Smith, A. D., and Thakurta, A. (2014). Private Empirical Risk Minimization: Efficient Algorithms and Tight Error Bounds. In *FOCS*.

Boyd, S., Ghosh, A., Prabhakar, B., and Shah, D. (2006). Randomized gossip algorithms. *IEEE/ACM Transactions on Networking*, 14(SI):2508–2530.

Cappon, G., Acciaroli, G., Vettoretti, M., Facchinetti, A., and Sparacino, G. (2017). Wearable continuous glucose monitoring sensors: A revolution in diabetes treatment. *Electronics*, 6(3).

Chaudhuri, K., Monteleoni, C., and Sarwate, A. D. (2011). Differentially Private Empirical Risk Minimization. *Journal of Machine Learning Research*, 12:1069–1109.

Colin, I., Bellet, A., Salmon, J., and Clémençon, S. (2016). Gossip Dual Averaging for Decentralized Optimization of Pairwise Functions. In *ICML*.

Dhillon, P. S., Sellamanickam, S., and Selvaraj, S. K. (2011). Semi-supervised multi-task learning of structured prediction models for web information extraction. In *CIKM*, pages 957–966.

Duchi, J. C., Jordan, M. I., and Wainwright, M. J. (2012). Privacy Aware Learning. In *NIPS*.

Dwork, C. (2006). Differential Privacy. In *ICALP*, volume 2.

Dwork, C. and Roth, A. (2014). The Algorithmic Foundations of Differential Privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3–4):211–407.

Dwork, C., Rothblum, G. N., and Vadhan, S. (2010). Boosting and Differential Privacy. In *FOCS*.

Evgeniou, T. and Pontil, M. (2004). Regularized multi-task learning. In *KDD*.

Goethals, B., Laur, S., Lipmaa, H., and Mielikäinen, T. (2004). On Private Scalar Product Computation for Privacy-Preserving Data Mining. In *ICISC*.

Hamm, J., Cao, Y., and Belkin, M. (2016). Learning privately from multiparty data. In *ICML*.

Hitaj, B., Ateniese, G., and Perez-Cruz, F. (2017). Deep Models Under the GAN: Information Leakage from Collaborative Deep Learning. In *CCS*.

Huang, Z., Mitra, S., and Vaidya, N. (2015). Differentially Private Distributed Optimization. *ICDCN*.

Kairouz, P., Oh, S., and Viswanath, P. (2015). The Composition Theorem for Differential Privacy. In *ICML*.

Kairouz, P., Oh, S., and Viswanath, P. (2016). Extremal Mechanisms for Local Differential Privacy. *Journal of Machine Learning Research*, 17:1–51.

Kermarrec, A.-M. and Taïani, F. (2015). Want to scale in centralized systems? Think P2P. *Journal of Internet Services and Applications*, 6(1):16:1–16:12.

Lian, X., Zhang, C., Zhang, H., Hsieh, C.-J., Zhang, W., and Liu, J. (2017). Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent. In *NIPS*.

Maurer, A. (2006). The Rademacher Complexity of Linear Transformation Classes. In *COLT*.

McGregor, A., Mironov, I., Pitassi, T., Reingold, O., Talwar, K., and Vadhan, S. (2010). The Limits of Two-Party Differential Privacy. In *FOCS*.

Nedic, A. (2011). Asynchronous Broadcast-Based Convex Optimization Over a Network. *IEEE Transactions on Automatic Control*, 56(6):1337–1351.

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

Pathak, M. A., Rane, S., and Raj, B. (2010). Multiparty Differential Privacy via Aggregation of Locally Trained Classifiers. In *NIPS*.

Rajkumar, A. and Agarwal, S. (2012). A Differentially Private Stochastic Gradient Descent Algorithm for Multiparty Classification. In *AISTATS*.

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

Shokri, R. and Shmatikov, V. (2015). Privacy-Preserving Deep Learning. In *CCS*.

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

Vanhaesebrouck, P., Bellet, A., and Tommasi, M. (2017). Decentralized Collaborative Learning of Personalized Models over Networks. In *AISTATS*.

Wei, E. and Ozdaglar, A. E. (2012). Distributed Alternating Direction Method of Multipliers. In *CDC*.

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

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