Decentralized Collaborative Learning of Personalized Models over Networks

Paul Vanhaesebrouck, Aurélien Bellet, Marc Tommasi

Decentralized Collaborative Learning of Personalized Models over Networks

Paul Vanhaesebrouck*[1], Aurélien Bellet†[1] and Marc Tommasi†[2]

[1]INRIA
[2]Université de Lille

Abstract

We consider a set of learning agents in a collaborative peer-to-peer network, where each agent learns a *personalized model* according to its own learning objective. The question addressed in this paper is: how can agents improve upon their locally trained model by communicating with other agents that have similar objectives? We introduce and analyze two asynchronous gossip algorithms running in a fully decentralized manner. Our first approach, inspired from label propagation, aims to smooth pre-trained local models over the network while accounting for the confidence that each agent has in its initial model. In our second approach, agents jointly learn and propagate their model by making iterative updates based on both their local dataset and the behavior of their neighbors. Our algorithm to optimize this challenging objective in a decentralized way is based on ADMM.

1 Introduction

Increasing amounts of data are being produced by interconnected devices such as mobile phones, connected objects, sensors, *etc.* For instance, history logs are generated when a smartphone user browses the web, gives product ratings and executes various applications. The currently dominant approach to extract useful information from such data is to collect all users' personal data on a server (or a tightly coupled system hosted in a data center) and apply centralized machine learning and data mining techniques. However, this centralization poses a number of issues, such as the need for users to "surrender" their personal data to the service provider without much control on how the data will be used, while incurring potentially high bandwidth and device battery costs. Even when the learning algorithm can be distributed in a way that keeps data on users' devices, a central entity is often still required for aggregation and coordination (see e.g., McMahan et al., 2016).

In this paper, we envision an alternative setting where many users (agents) with local datasets collaborate to learn models by engaging in a fully decentralized peer-to-peer network. Unlike existing work focusing on problems where agents seek to agree on a global *consensus model* (see e.g., Nedic and Ozdaglar, 2009; Wei and Ozdaglar, 2012; Duchi et al., 2012), we study the case where each agent learns a *personalized model* according to its own learning objective. We assume that the network graph is given and reflects a notion of similarity between agents (two agents are neighbors in the network if they have a similar learning objective), but each agent is only aware of its direct neighbors. An agent can then learn a model from its (typically scarce) personal data but also from interactions with its neighborhood. As a motivating example, consider a decentralized recommender system (Boutet et al., 2013, 2014) in which each user rates a small number of movies on a smartphone application and expects personalized recommendations of new movies. In order to train a reliable recommender for each user, one should rely on the limited user's data but also on information brought by users with similar taste/profile. The peer-to-peer communication graph could be established when some users go the same movie theater or attend the same cultural event, and some similarity weights between users could be computed based on historical data (e.g., counting how many times people have met in such locations).

Our contributions are as follows. After formalizing the problem of interest, we propose two asynchronous and fully decentralized algorithms for collaborative learning of personalized models. They belong to the family of gossip algorithms (Shah, 2009; Dimakis et al., 2010): agents only communicate with a single neighbor at a time, which makes our algorithms suitable for

*first.last@gmail.com
†first.last@inria.fr

deployment in large peer-to-peer real networks. Our first approach, called *model propagation*, is inspired by the graph-based label propagation technique of Zhou et al. (2004). In a first phase, each agent learns a model based on its local data only, without communicating with others. In a second phase, the model parameters are regularized so as to be smooth over the network graph. We introduce some confidence values to account for potential discrepancies in the agents' training set sizes, and derive a novel asynchronous gossip algorithm which is simple and efficient. We prove that this algorithm converges to the optimal solution of the problem. Our second approach, called *collaborative learning*, is more flexible as it interweaves learning and propagation in a single process. Specifically, it optimizes a trade-off between the smoothness of the model parameters over the network on the one hand, and the models' accuracy on the local datasets on the other hand. For this formulation, we propose an asynchronous gossip algorithm based on a decentralized version of Alternating Direction Method of Multipliers (ADMM) (Boyd et al., 2011). Finally, we evaluate the performance of our methods on two synthetic collaborative tasks: mean estimation and linear classification. Our experiments show the superiority of the proposed approaches over baseline strategies, and confirm the efficiency of our decentralized algorithms.

The rest of the paper is organized as follows. Section 2 formally describes the problem of interest and discusses some related work. Our model propagation approach is introduced in Section 3, along with our decentralized algorithm. Section 4 describes our collaborative learning approach, and derives an equivalent formulation which is amenable to optimization using decentralized ADMM. Finally, Section 5 shows our numerical results, and we conclude in Section 6.

2 Preliminaries

2.1 Notations and Problem Setting

We consider a set of n agents $V = [\![n]\!]$ where $[\![n]\!] := \{1, \dots, n\}$. Given a convex loss function $\ell : \mathbb{R}^p \times \mathcal{X} \times \mathcal{Y}$, the goal of agent i is to learn a model $\theta_i \in \mathbb{R}^p$ whose expected loss $\mathbb{E}_{(x_i,y_i)\sim\mu_i} \ell(\theta_i; x_i, y_i)$ is small with respect to an unknown and fixed distribution μ_i over $\mathcal{X} \times \mathcal{Y}$. Each agent i has access to a set of $m_i \geq 0$ i.i.d. training examples $\mathcal{S}_i = \{(x_i^j, y_i^j)\}_{j=1}^{m_i}$ drawn from μ_i. We allow the training set size to vary widely across agents (some may even have no data at all). This is important in practice as some agents may be more "active" than others, may have recently joined the service, *etc.*

In isolation, an agent i can learn a *"solitary" model* θ_i^{sol} by minimizing the loss over its local dataset $\mathcal{S}_i$:

$$\theta_i^{sol} \in \arg\min_{\theta\in\mathbb{R}^p} \mathcal{L}_i(\theta) = \sum_{j=1}^{m_i} \ell(\theta; x_i^j, y_i^j). \qquad (1)$$

The goal for the agents is to improve upon their solitary model by leveraging information from other users in the network. Formally, we consider a weighted connected graph $G = (V, E)$ over the set V of agents, where $E \subseteq V \times V$ is the set of undirected edges. We denote by $W \in \mathbb{R}^{n\times n}$ the symmetric nonnegative weight matrix associated with G, where W_{ij} gives the weight of edge $(i, j) \in E$ and by convention, $W_{ij} = 0$ if $(i, j) \notin E$ or $i = j$. We assume that the weights represent the underlying similarity between the agents' objectives: W_{ij} should tend to be large (resp. small) when the objectives of agents i and j are similar (resp. dissimilar). While we assume in this paper that the weights are given, in practical scenarios one could for instance use some auxiliary information such as users' profiles (when available) and/or prediction disagreement to estimate the weights. For notational convenience, we define the diagonal matrix $D \in \mathbb{R}^{n\times n}$ where $D_{ii} = \sum_{j=1}^{n} W_{ij}$. We will also denote by $\mathcal{N}_i = \{j \neq i : W_{ij} > 0\}$ the set of neighbors of agent i. We assume that the agents only have a local view of the network: they know their neighbors and the associated weights, but not the global topology or how many agents participate in the network.

Our goal is to propose decentralized algorithms for agents to collaboratively improve upon their solitary model by leveraging information from their neighbors.

2.2 Related Work

Several peer-to-peer algorithms have been developed for decentralized averaging (Kempe et al., 2003; Boyd et al., 2006; Colin et al., 2015) and optimization (Nedic and Ozdaglar, 2009; Ram et al., 2010; Duchi et al., 2012; Wei and Ozdaglar, 2012, 2013; Iutzeler et al., 2013; Colin et al., 2016). These approaches solve a *consensus problem* of the form:

$$\min_{\theta\in\mathbb{R}^p} \sum_{i=1}^{n} \mathcal{L}_i(\theta), \qquad (2)$$

resulting in a global solution common to all agents (e.g., a classifier minimizing the prediction error over the union of all datasets). This is unsuitable for our setting, where all agents have personalized objectives.

Our problem is reminiscent of Multi-Task Learning (MTL) (Caruana, 1997), where one jointly learns models for related tasks. Yet, there are several differences with our setting. In MTL, the number of tasks is often small, training sets are well-balanced across tasks, and

all tasks are usually assumed to be positively related (a popular assumption is that all models share a common subspace). Lastly, the algorithms are centralized, aside from the distributed MTL of Wang et al. (2016) which is synchronous and relies on a central server.

3 Model Propagation

In this section, we present our model propagation approach. We first introduce a global optimization problem, and then propose and analyze an asynchronous gossip algorithm to solve it.

3.1 Problem Formulation

In this formulation, we assume that each agent i has learned a solitary model θ_i^{sol} by minimizing its local loss, as in (1). This can be done without any communication between agents. Our goal here consists in adapting these models by making them smoother over the network graph. In order account for the fact that the solitary models were learned on training sets of different sizes, we will use $c_i \in (0, 1]$ to denote the confidence we put in the model θ_i^{sol} of user $i \in \{1, \dots, n\}$. The c_i's should be proportional to the number of training points m_i — one may for instance set $c_i = \frac{m_i}{\max_j m_j}$ (plus some small constant in the case where $m_i = 0$).

Denoting $\Theta = [\theta_1; \dots; \theta_n] \in \mathbb{R}^{n \times p}$, the objective function we aim to minimize is as follows:

$$\mathcal{Q}_{MP}(\Theta) = \frac{1}{2}\Bigg(\sum_{i<j}^{n} W_{ij} \|\theta_i - \theta_j\|^2 + \mu \sum_{i=1}^{n} D_{ii} c_i \|\theta_i - \theta_i^{sol}\|^2 \Bigg), \quad (3)$$

where $\mu > 0$ is a trade-off parameter and $\|\cdot\|$ denotes the Euclidean norm. The first term in the right hand side of (3) is a classic quadratic form used to smooth the models within neighborhoods: the distance between the new models of agents i and j is encouraged to be small when the weight W_{ij} is large. The second term prevents models with large confidence from diverging too much from their original values so that they can propagate useful information to their neighborhood. On the other hand, models with low confidence are allowed large deviations: in the extreme case where agent i has very little or even no data (i.e., c_i is negligible), its model is fully determined by the neighboring models. The presence of D_{ii} in the second term is simply for normalization. We have the following result (the proof is in Appendix A).

Proposition 1 (Closed-form solution)**.** *Let $P = D^{-1}W$ be the stochastic similarity matrix associated with the graph G and $\Theta^{sol} = [\theta_1^{sol}; \dots; \theta_n^{sol}] \in \mathbb{R}^{n \times p}$. The solution $\Theta^\star = \arg\min_{\Theta \in \mathbb{R}^{n \times p}} \mathcal{Q}_{MP}(\Theta)$ is given by*

$$\Theta^\star = \bar{\alpha}(I - \bar{\alpha}(I - C) - \alpha P)^{-1} C \Theta^{sol}, \quad (4)$$

with $\alpha \in (0, 1)$ such that $\mu = (1-\alpha)/\alpha$, and $\bar{\alpha} = 1 - \alpha$.

Our formulation is a generalization of the semi-supervised label propagation technique of (Zhou et al., 2004), which can be recovered by setting $C = I$ (same confidence for all nodes). Note that it is *strictly* more general: we can see from (4) that unless the confidence values are equal for all agents, the confidence information cannot be incorporated by using different solitary models Θ^{sol} or by considering a different graph (because $\frac{\bar{\alpha}}{\alpha}(I - C) - P$ is not stochastic). The asynchronous gossip algorithm we present below thus applies to label propagation for which, to the best of our knowledge, no such algorithm was previously known.

Computing the closed form solution (4) requires the knowledge of the global network and of all solitary models, which are unknown to the agents. Our starting point for the derivation of an asynchronous gossip algorithm is the following iterative form: for any $t \geq 0$,

$$\Theta(t+1) = (\alpha I + \bar{\alpha} C)^{-1} \left(\alpha P \Theta(t) + \bar{\alpha} C \Theta^{sol} \right), \quad (5)$$

The sequence $(\Theta(t))_{t \in \mathbb{N}}$ can be shown to converge to (4) regardless of the choice of initial value $\Theta(0)$, see Appendix B for details. An interesting observation about this recursion is that it can be decomposed into agent-centric updates which only involve neighborhoods. Indeed, for any agent i and any $t \geq 0$:

$$\theta_i(t+1) = \frac{1}{\alpha + \bar{\alpha} c_i} \Big(\alpha \sum_{j \in \mathcal{N}_i} \frac{W_{ij}}{D_{ii}} \theta_j(t) + \bar{\alpha} c_i \theta_i^{sol} \Big).$$

The iteration (5) can thus be understood as a decentralized but synchronous process where, at each step, every agent communicates with all its neighbors to collect their current model parameters and uses this information to update its model. Assuming that the agents do have access to a global clock to synchronize the updates (which is unrealistic in many practical scenarios), synchronization incurs large delays since all agents must finish the update at step t before anyone starts step $t + 1$. The fact that agents must contact all their neighbors at each iteration further hinders the efficiency of the algorithm. To avoid these limitations, we propose below an asynchronous gossip algorithm.

3.2 Asynchronous Gossip Algorithm

In the asynchronous setting, each agent has a *local* clock ticking at the times of a rate 1 Poisson process, and wakes up when it ticks. As local clocks are i.i.d.,

it is equivalent to activating a single node uniformly at random at each time step (Boyd et al., 2006).[1]

The idea behind our algorithm is the following. At any time $t \geq 0$, each agent i will maintain a (possibly outdated) knowledge of its neighbors' models. For mathematical convenience, we will consider a matrix $\widetilde{\Theta}_i(t) \in \mathbb{R}^{n \times p}$ where its i-th line $\widetilde{\Theta}_i^i(t) \in \mathbb{R}^p$ is agent i's model at time t, and for $j \neq i$, its j-th line $\widetilde{\Theta}_i^j(t) \in \mathbb{R}^p$ is agent i's *last knowledge* of the model of agent j. For any $j \notin \mathcal{N}_i \cup \{i\}$ and any $t \geq 0$, we will maintain $\widetilde{\Theta}_i^j(t) = 0$. Let $\widetilde{\Theta} = [\widetilde{\Theta}_1^\top, \ldots, \widetilde{\Theta}_n^\top]^\top \in \mathbb{R}^{n^2 \times p}$ be the horizontal stacking of all the $\widetilde{\Theta}_i$'s.

If agent i wakes up at time step t, two consecutive actions are performed:

- *communication step*: agent i selects a random neighbor $j \in \mathcal{N}_i$ with prob. π_i^j and both agents update their knowledge of each other's model:
$$\widetilde{\Theta}_i^j(t+1) = \widetilde{\Theta}_j^j(t) \text{ and } \widetilde{\Theta}_j^i(t+1) = \widetilde{\Theta}_i^i(t),$$
- *update step*: agents i and j update their own models based on current knowledge. For $l \in \{i, j\}$:
$$\widetilde{\Theta}_l^l(t+1) = (\alpha + \bar{\alpha} c_l)^{-1} \Big(\alpha \sum_{k \in \mathcal{N}_l} \frac{W_{lk}}{D_{ll}} \widetilde{\Theta}_l^k(t+1) + \bar{\alpha} c_l \theta_l^{sol}\Big). \quad (6)$$

All other variables in the network remain unchanged. In the communication step above, π_i^j corresponds to the probability that agent i selects agent j. For any $i \in [\![n]\!]$, we have $\pi_i \in [0,1]^n$ such that $\sum_{j=1}^n \pi_i^j = 1$ and $\pi_i^j > 0$ if and only if $j \in \mathcal{N}_i$.

Our algorithm belongs to the family of *gossip algorithms* as each agent communicates with at most one neighbor at a time. Gossip algorithms are known to be very effective for decentralized computation in peer-to-peer networks (see Dimakis et al., 2010; Shah, 2009). Thanks to its asynchronous updates, our algorithm has the potential to be much faster than a synchronous version when executed in a large peer-to-peer network.

The main result of this section shows that our algorithm converges to a state where all nodes have their optimal model (and those of their neighbors).

Theorem 1 (Convergence)**.** *Let $\widetilde{\Theta}(0) \in \mathbb{R}^{n^2 \times p}$ be some arbitrary initial value and $(\widetilde{\Theta}(t))_{t \in \mathbb{N}}$ be the sequence generated by our algorithm. Let $\Theta^\star = \arg\min_{\Theta \in \mathbb{R}^{n \times p}} \mathcal{Q}_{MP}(\Theta)$ be the optimal solution to model propagation. For any $i \in [\![n]\!]$, we have:*
$$\lim_{t \to \infty} \mathbb{E}\left[\widetilde{\Theta}_i^j(t)\right] = \Theta_j^\star \text{ for } j \in \mathcal{N}_i \cup \{i\}.$$

[1] Our analysis straightforwardly extends to the case where agents have clocks ticking at different rates.

Sketch of proof. The first step of the proof is to rewrite the algorithm as an equivalent random iterative process over $\widetilde{\Theta} \in \mathbb{R}^{n^2 \times p}$ of the form:
$$\widetilde{\Theta}(t+1) = A(t)\widetilde{\Theta}(t) + b(t),$$
for any $t \geq 0$. Then, we show that the spectral radius of $\mathbb{E}[A(t)]$ is smaller than 1, which allows us to exhibit the convergence to the desired quantity. The proof can be found in Appendix C. □

4 Collaborative Learning

In the approach presented in the previous section, models are learned locally by each agent and then propagated through the graph. In this section, we allow the agents to simultaneously learn their model and propagate it through the network. In other words, agents iteratively update their models based on both their local dataset and the behavior of their neighbors. While in general this is computationally more costly than merely propagating pre-trained models, we can expect significant improvements in terms of accuracy.

As in the case of model propagation, we first introduce the global objective function and then propose an asynchronous gossip algorithm, which is based on the general paradigm of ADMM (Boyd et al., 2011).

4.1 Problem Formulation

In contrast to model propagation, the objective function to minimize here takes into account the loss of each personal model on the local dataset, rather than simply the distance to the solitary model:
$$\mathcal{Q}_{CL}(\Theta) = \sum_{i<j}^n W_{ij} \|\theta_i - \theta_j\|^2 + \mu \sum_{i=1}^n D_{ii} \mathcal{L}_i(\theta_i), \quad (7)$$
where $\mu > 0$ is a trade-off parameter. The associated optimization problem is $\Theta^\star = \arg\min_{\Theta \in \mathbb{R}^{n \times p}} \mathcal{Q}_{CL}(\Theta)$.

The first term in the right hand side of (7) is the same as in the model propagation objective (3) and tends to favor models that are smooth on the graph. However, while in model propagation enforcing smoothness on the models may potentially translate into a significant decrease of accuracy on the local datasets (even for relatively small changes in parameter values with respect to the solitary models), here the second term prevents this. It allows more flexibility in settings where very different parameter values define models which actually give very similar predictions. Note that the confidence is built in the second term as $\mathcal{L}_i$ is a sum over the local dataset of agent i.

In general, there is no closed-form expression for $\Theta^\star$, but we can solve the problem with a decentralized iterative algorithm, as shown in the rest of this section.

4.2 Asynchronous Gossip Algorithm

We propose an asynchronous decentralized algorithm for minimizing (7) based on the Alternative Direction Method of Multipliers (ADMM). This general method is a popular way to solve consensus problems of the form (2) in the distributed and decentralized settings (see e.g., Boyd et al., 2011; Wei and Ozdaglar, 2012, 2013; Iutzeler et al., 2013). In our setting, we do not seek a consensus in the classic sense of (2) since our goal is to learn a personalized model for each agent. However, we show below that we can reformulate (7) as an equivalent *partial* consensus problem which is amenable to decentralized optimization with ADMM.

Problem reformulation. Let Θ_i be the set of $|\mathcal{N}_i|+1$ variables $\theta_j \in \mathbb{R}^p$ for $j \in \mathcal{N}_i \cup \{i\}$, and denote θ_j by Θ_i^j. This is similar to the notations used in Section 3, except that here we consider Θ_i as living in $\mathbb{R}^{(|\mathcal{N}_i|+1)\times p}$. We now define

$$\mathcal{Q}_{CL}^i(\Theta_i) = \frac{1}{2}\sum_{j\in\mathcal{N}_i} W_{ij}\|\theta_i - \theta_j\|^2 + \mu D_{ii}\mathcal{L}_i(\theta_i),$$

so that we can rewrite our problem (7) as $\min_{\Theta\in\mathbb{R}^{n\times p}} \sum_{i=1}^n \mathcal{Q}_{CL}^i(\Theta_i)$.

In this formulation, the objective functions associated with the agents are dependent as they share some decision variables in Θ. In order to apply decentralized ADMM, we need to decouple the objectives. The idea is to introduce a local copy $\widetilde{\Theta}_i \in \mathbb{R}^{(|\mathcal{N}_i|+1)\times p}$ of the decision variables Θ_i for each agent i and to impose equality constraints on the variables $\widetilde{\Theta}_i^i = \widetilde{\Theta}_j^i$ for all $i \in [\![n]\!], j \in \mathcal{N}_i$. This partial consensus can be seen as requiring that two neighboring agents agree on each other's personalized model. We further introduce 4 secondary variables Z_{ei}^i, Z_{ej}^i, Z_{ei}^j and Z_{ej}^j for each edge $e = (i,j)$, which can be viewed as estimates of the models $\widetilde{\Theta}_i$ and $\widetilde{\Theta}_j$ known by each end of e and will allow an efficient decomposition of the ADMM updates.

Formally, denoting $\widetilde{\Theta} = [\widetilde{\Theta}_1^\top, \ldots, \widetilde{\Theta}_n^\top]^\top \in \mathbb{R}^{(2|E|+n)\times p}$ and $Z \in \mathbb{R}^{4|E|\times p}$, we introduce the formulation

$$\begin{array}{ll} \min\limits_{\substack{\widetilde{\Theta}\in\mathbb{R}^{(2|E|+n)\times p}\\ Z\in\mathcal{C}_E}} & \sum\limits_{i=1}^n \mathcal{Q}_{CL}^i(\widetilde{\Theta}_i) \\ \text{s.t. } \forall e=(i,j)\in E, & \begin{cases} Z_{ei}^i = \widetilde{\Theta}_i^i,\ Z_{ei}^j = \widetilde{\Theta}_i^j \\ Z_{ej}^j = \widetilde{\Theta}_j^j,\ Z_{ej}^i = \widetilde{\Theta}_j^i, \end{cases} \end{array} \tag{8}$$

where $\mathcal{C}_E = \{Z \in \mathbb{R}^{4|E|\times p} \mid Z_{ei}^i = Z_{ej}^i, Z_{ej}^j = Z_{ei}^j \text{ for all } e=(i,j)\in E\}$. It is easy to see that Problem (8) is equivalent to the original problem (7) in the following sense: the minimizer $\widetilde{\Theta}^\star$ of (8) satisfies $(\widetilde{\Theta}^\star)_i^j = \Theta_j^\star$ for all $i \in [\![n]\!]$ and $j \in \mathcal{N}_i \cup \{i\}$. Further observe that the set of constraints involving $\widetilde{\Theta}$ can be written $D\widetilde{\Theta} + HZ = 0$ where $H = -I$ of dimension $4|E|\times 4|E|$ is diagonal invertible and D of dimension $4|E|\times(2|E|+n)$ contains exactly one entry of 1 in each row. The assumptions of Wei and Ozdaglar (2013) are thus met and we can apply asynchronous decentralized ADMM.

Before presenting the algorithm, we derive the augmented Lagrangian associated with Problem (8). Let Λ_{ei}^j be dual variables associated with constraints involving $\widetilde{\Theta}$ in (8). For convenience, we denote by $Z_i \in \mathbb{R}^{2|\mathcal{N}_i|}$ the set of secondary variables $\{\{Z_{ei}^i\} \cup \{Z_{ei}^j\}\}_{e=(i,j)\in E}$ associated with agent i. Similarly, we denote by $\Lambda_i \in \mathbb{R}^{2|\mathcal{N}_i|}$ the set of dual variables $\{\{\Lambda_{ei}^i\} \cup \{\Lambda_{ei}^j\}\}_{e=(i,j)\in E}$. The augmented Lagrangian is given by:

$$L_\rho(\widetilde{\Theta}, Z, \Lambda) = \sum_{i=1}^n L_\rho^i(\widetilde{\Theta}_i, Z_i, \Lambda_i),$$

where $\rho > 0$ is a penalty parameter, $Z \in \mathcal{C}_E$ and

$$L_\rho^i(\widetilde{\Theta}_i, Z_i, \Lambda_i) = \mathcal{Q}_{CL}^i(\widetilde{\Theta}_i) + \sum_{j:e=(i,j)\in E} \Big[\Lambda_{ei}^i(\widetilde{\Theta}_i^i - Z_{ei}^i) + \Lambda_{ei}^j(\widetilde{\Theta}_i^j - Z_{ei}^j) + \frac{\rho}{2}\left(\|\widetilde{\Theta}_i^i - Z_{ei}^i\|^2 + \|\widetilde{\Theta}_i^j - Z_{ei}^j\|^2\right)\Big].$$

Algorithm. ADMM consists in approximately minimizing the augmented Lagrangian $L_\rho(\widetilde{\Theta}, Z, \Lambda)$ by alternating minimization with respect to the primal variable $\widetilde{\Theta}$ and the secondary variable Z, together with an iterative update of the dual variable Λ.

We first briefly discuss how to instantiate the initial values $\widetilde{\Theta}(0)$, $Z(0)$ and $\Lambda(0)$. The only constraint on these initial values is to have $Z(0) \in \mathcal{C}_E$, so a simple option is to initialize all variables to 0. That said, it is typically advantageous to use a warm-start strategy. For instance, each agent i can send its solitary model θ_i^{sol} to its neighbors, and then set $\widetilde{\Theta}_i^i = \theta_i^{sol}$, $\widetilde{\Theta}_i^j = \theta_j^{sol}$ for all $j \in \mathcal{N}_i$, $Z_{ei}^i = \widetilde{\Theta}_i^i$, $Z_{ei}^j = \widetilde{\Theta}_i^j$ for all $e = (i,j) \in E$, and $\Lambda(0) = 0$. Alternatively, one can initialize the algorithm with the model propagation solution obtained using the method of Section 3.

Recall from Section 3.2 that in the asynchronous setting, a single agent wakes up at each time step and selects one of its neighbors. Assume that agent i wakes up at some iteration $t \geq 0$ and selects $j \in \mathcal{N}_i$. Denoting $e = (i,j)$, the iteration goes as follows:

1. Agent i updates its primal variables:

$$\widetilde{\Theta}_i(t+1) = \operatorname*{arg\,min}_{\Theta\in\mathbb{R}^{(|\mathcal{N}_i|+1)\times p}} L_\rho^i(\Theta, Z_i(t), \Lambda_i(t)),$$

and sends $\widetilde{\Theta}_i^i(t+1), \widetilde{\Theta}_i^j(t+1), \Lambda_{ei}^i(t), \Lambda_{ei}^j(t)$ to agent j. Agent j executes the same steps w.r.t. i.

2. Using $\widetilde{\Theta}_j^j(t+1), \widetilde{\Theta}_j^i(t+1), \Lambda_{ej}^j(t), \Lambda_{ej}^i(t)$ received from j, agent i updates its secondary variables:

$$Z_{ei}^i(t+1) = \frac{1}{2}\Big[\frac{1}{\rho}\big(\Lambda_{ei}^i(t) + \Lambda_{ej}^i(t)\big) + \widetilde{\Theta}_i^i(t+1) + \widetilde{\Theta}_j^i(t+1)\Big],$$

$$Z_{ei}^j(t+1) = \frac{1}{2}\Big[\frac{1}{\rho}\big(\Lambda_{ej}^j(t) + \Lambda_{ei}^j(t)\big) + \widetilde{\Theta}_j^j(t+1) + \widetilde{\Theta}_i^j(t+1)\Big].$$

Agent j updates its secondary variables symmetrically, so by construction we have $Z(t+1) \in \mathcal{C}_E$.

3. Agent i updates its dual variables:

$$\Lambda_{ei}^i(t+1) = \Lambda_{ei}^i(t) + \rho\big(\widetilde{\Theta}_i^i(t+1) - Z_{ei}^i(t+1)\big),$$
$$\Lambda_{ei}^j(t+1) = \Lambda_{ei}^j(t) + \rho\big(\widetilde{\Theta}_i^j(t+1) - Z_{ei}^j(t+1)\big).$$

Agent j updates its dual variables symmetrically.

All other variables in the network remain unchanged.

Step 1 has a simple solution for some loss functions commonly used in machine learning (such as quadratic and L_1 loss), and when it is not the case ADMM is typically robust to approximate solutions to the corresponding subproblems (obtained for instance after a few steps of gradient descent), see Boyd et al. (2011) for examples and further practical considerations. Asynchronous ADMM converges almost surely to an optimal solution at a rate of $O(1/t)$ for convex objective functions (see Wei and Ozdaglar, 2013).

5 Experiments

In this section, we provide numerical experiments to evaluate the performance of our decentralized algorithms with respect to accuracy, convergence rate and the amount of communication. To this end, we introduce two novel collaborative tasks: mean estimation and linear classification.

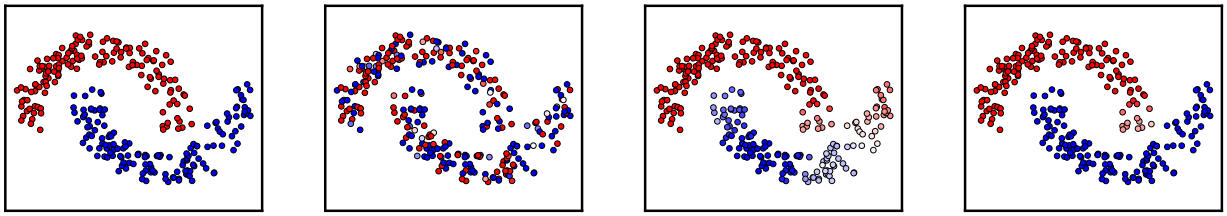
5.1 Collaborative Mean Estimation

We first introduce a simple task in which the goal of each agent is to estimate the mean of a 1D distribution. To this end, we adapt the two intertwining moons dataset popular in semi-supervised learning (Zhou et al., 2004). We consider a set of 300 agents, together with auxiliary information about each agent i in the form of a vector $v_i \in \mathbb{R}^2$. The true distribution μ_i of an agent i is either $\mathcal{N}(1, 40)$ or $\mathcal{N}(-1, 40)$ depending on whether v_i belongs to the upper or lower moon, see Figure 1(a). Each agent i receives m_i samples $x_i^1, \dots, x_i^{m_i} \in \mathbb{R}$ from its distribution μ_i. Its solitary model is then given by $\theta_i^{sol} = \frac{1}{m_i}\sum_{j=1}^{m_i} x_i^j$, which corresponds to the use of the quadratic loss function $\ell(\theta; x_i) = \|\theta - x_i\|^2$. Finally, the graph over agents is the complete graph where the weight between agents i and j is given by a Gaussian kernel on the agents' auxiliary information $W_{ij} = \exp(-\|v_i - v_j\|^2/2\sigma^2)$, with $\sigma = 0.1$ for appropriate scaling. In all experiments, the parameter α of model propagation was set to 0.99, which gave the best results on a held-out set of random problem instances. We first use this mean estimation task to illustrate the importance of considering confidence values in our model propagation formulation, and then to evaluate the efficiency of our asynchronous decentralized algorithm.

Relevance of confidence values. Our goal here is to show that introducing confidence values into the model propagation approach can significantly improve the overall accuracy, especially when the agents receive unbalanced amounts of data. In this experiment, we only compare model propagation with and without confidence values, so we compute the optimal solutions directly using the closed-form solution (4).

We generate several problem instances with varying standard deviation for the confidence values c_i's. More precisely, we sample c_i for each agent i from a uniform distribution centered at $1/2$ with width $\epsilon \in [0, 1]$. The number of samples m_i given to agent i is then set to $m_i = \lceil c_i \cdot 100 \rceil$. The larger ϵ, the more variance in the size of the local datasets. Figures 1(b)-1(d) give a visualization of the models before and after propagation on a problem instance for the hardest setting $\epsilon = 1$. Figure 2 (left-middle) shows results averaged over 1000 random problem instances for several values of ϵ. As expected, when the local dataset sizes are well-balanced (small ϵ), model propagation performs the same with or without the use of confidence values. Indeed, both have similar L_2 error with respect to the target mean, and the win ratio is about 0.5. However, the performance gap in favor of using confidence values increases sharply with ϵ. For $\epsilon = 1$, the win ratio in favor of using confidence values is about 0.85. Strikingly, the error of model propagation with confidence values remains constant as ϵ increases. These results empirically confirm the relevance of introducing confidence values into the objective function.

Asynchronous algorithm. In this second experiment, we compare asynchronous model propagation with the synchronous variant given by (5). We are interested in the average L_2 error of the models as a function of the number of pairwise communications (number of exchanges from one agent to another). Note that a single iteration of the synchronous (resp. asynchronous) algorithm corresponds to $2|E|$ (resp. 2) communications. For the asynchronous algorithm, we set the neighbor selection distribution π_i of agent



(a) Ground models (b) Solitary models (c) MP without confidence (d) MP with confidence

Figure 1: Illustration of the collaborative mean estimation task, where each point represents an agent and its 2D coordinates the associated auxiliary information. Figure 1(a) shows the ground truth models (blue for mean 1 and red for mean -1). Figure 1(b) shows the solitary models (local averages) for an instance where $\epsilon = 1$. Figures 1(c)-1(d) show the models after propagation, without/with the use of confidence values.

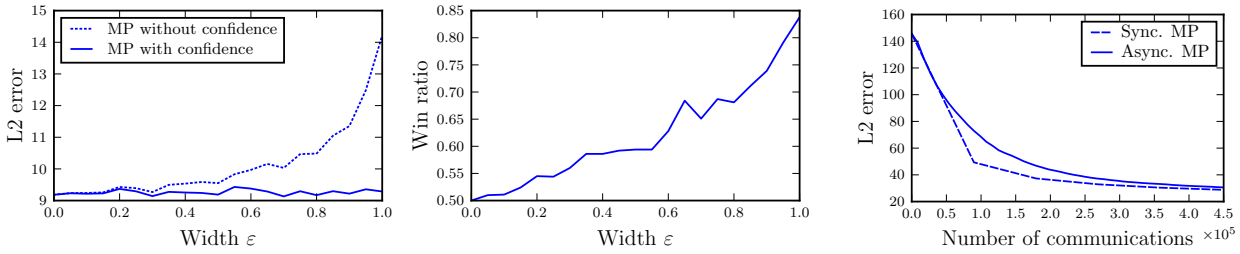


Figure 2: Results on the mean estimation task. (Left-middle) Model propagation with and without confidence values w.r.t. the unbalancedness of the local datasets. The left figure shows the L_2 errors, while the middle one shows the percentage of wins in favor of using confidence values. (Right) L_2 error of the synchronous and asynchronous model propagation algorithms with respect to the number of pairwise communications.

$i \in [\![n]\!]$ to be uniform over the set of neighbors $\mathcal{N}_i$.

Figure 2 (right) shows the results on a problem instance generated as in the previous experiment (with $\varepsilon = 1$). Since the asynchronous algorithm is randomized, we average its results on 100 random runs. We see that our asynchronous algorithm achieves an accuracy/communication trade-off which is almost as good as that of the synchronous one, without requiring any synchronization. It is thus expected to be much faster than the synchronous algorithm on large decentralized networks with communication delays and/or without efficient global synchronization.

5.2 Collaborative Linear Classification

In the previous mean estimation task, the squared distance between two model parameters (i.e., estimated means) translates into the same difference in L_2 error with respect to the target mean. Therefore, our collaborative learning formulation is essentially equivalent to our model propagation approach. To show the benefits that can be brought by collaborative learning, we now consider a linear classification task. Since two linear separators with significantly different parameters can lead to similar predictions on a given dataset, incorporating the local errors into the objective function rather than simply the distances between parameters should lead to more accurate models.

We consider a set of 100 agents whose goal is to perform linear classification in $\mathbb{R}^p$. For ease of visualization, the target (true) model of each agent lies in a 2-dimensional subspace: we represent it as a vector in $\mathbb{R}^p$ with the first two entries drawn from a normal distribution centered at the origin and the remaining ones equal to 0. We consider the similarity graph where the weight between two agents i and j is a Gaussian kernel on the distance between target models, where the distance here refers to the length of the chord of the angle ϕ_{ij} between target models projected on a unit circle. More formally, $W_{i,j} = \exp((\cos(\phi_{i,j}) - 1)/\sigma)$ with $\sigma = 0.1$ for appropriate scaling. Edges with negligible weights are ignored to speed up computation. We refer the reader to Appendix E for a 2D visualization of the target models and the links between them. Every agent receives a random number of training points drawn uniformly between 1 and 20. Each training point (in $\mathbb{R}^p$) is drawn uniformly around the origin, and the binary label is given by the prediction of the target linear separator. We then add some label noise by randomly flipping each label with probability

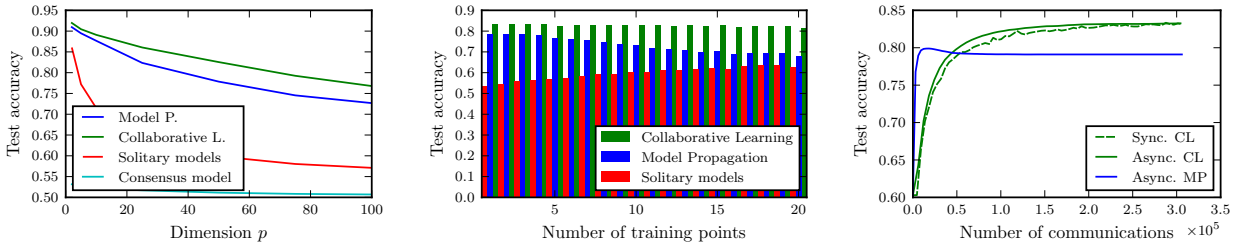


Figure 3: Results on the linear classification task. (Left) Test accuracy of model propagation and collaborative learning with varying feature space dimension. (Middle) Average test accuracy of model propagation and collaborative learning with respect to the number of training points available to the agent (feature dimension $p = 50$). (Right) Test accuracy of synchronous and asynchronous collaborative learning and asynchronous model propagation with respect to the number of pairwise communications (linear classification task, $p = 50$).

0.05. The loss function used by the agents is the hinge loss, given by $\ell(\theta; (x_i, y_i)) = \max\left(0, 1 - y_i \theta^\top x_i\right)$. As in the previous experiment, for each algorithm we tune the value of α on a held-out set of random problem instances. Finally, we will evaluate the quality of the learned model of each agent by computing the accuracy on a separate sample of 100 test points drawn from the same distribution as the training set.

In the following, we use this linear classification task to compare the performance of collaborative learning against model propagation, and to evaluate the efficiency of our asynchronous algorithms.

MP vs. CL. In this first experiment, we compare the accuracy of the models learned by model propagation and collaborative learning with feature space dimension p ranging from 2 to 100. Figure 3 (left) shows the results averaged over 10 randomly generated problem instances for each value of p. As baselines, we also plot the average accuracy of the solitary models and of the global consensus model minimizing (2). The accuracy of all models decreases with the feature space dimension, which comes from the fact that the expected number of training samples remains constant. As expected, the consensus model achieves very poor performance since agents have very different objectives. On the other hand, both model propagation and collaborative learning are able to improve very significantly over the solitary models, even in higher dimensions where on average these initial models barely outperform a random guess. Furthermore, collaborative learning always outperforms model propagation.

We further analyze these results by plotting the accuracy with respect to the size of the local training set (Figure 3, middle). As expected, the accuracy of the solitary models is higher for larger training sets. Furthermore, collaborative learning converges to models which have similar accuracy regardless of the training size, effectively correcting for the initial unbalancedness. While model propagation also performs well, it is consistently outperformed by collaborative learning on all training sizes. This is especially the case for agents with larger training sizes.

Asynchronous algorithms. This second experiment compares our asynchronous collaborative learning algorithm with a synchronous variant also based on ADMM (see Appendix D for details) in terms of number pairwise of communications. Figure 3 (right) shows that our asynchronous algorithm performs as good as its synchronous counterpart and should thus be largely preferred for deployment in real peer-to-peer networks. It is also worth noting that asynchronous model propagation converges an order of magnitude faster than collaborative learning, as it only propagates models that are pre-trained locally. Model propagation can thus provide a valuable warm-start initialization for collaborative learning.

Scalability. We also observe experimentally that the number of iterations needed by our decentralized algorithms to converge scales favorably with the size of the network (see Appendix E for details).

6 Conclusion

We proposed, analyzed and evaluated two asynchronous peer-to-peer algorithms for the novel setting of decentralized collaborative learning of personalized models. In our opinion, this work opens up interesting perspectives. The link between the similarity graph and the generalization performance of the resulting models should be formally analyzed. This could in turn guide the design of generic methods to estimate the graph weights, making our approaches more easily applicable to real-world problems. Other directions of interest include the development of privacy-preserving algorithms as well as extensions to time-evolving networks and sequential arrival of data.

References

Boutet, A., Frey, D., Guerraoui, R., Jégou, A., and Kermarrec, A.-M. (2013). WHATSUP: A Decentralized Instant News Recommender. In *Proceedings of the 27th IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, pages 741–752.

Boutet, A., Frey, D., Guerraoui, R., Kermarrec, A.-M., and Patra, R. (2014). Hyrec: leveraging browsers for scalable recommenders. In *Proceedings of the 15th International Middleware Conference*, pages 85–96.

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

Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning*, 3(1):1–122.

Caruana, R. (1997). Multitask Learning. *Machine Learning*, 28(1):41–75.

Colin, I., Bellet, A., Salmon, J., and Clémençon, S. (2015). Extending Gossip Algorithms to Distributed Estimation of U-statistics. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*.

Colin, I., Bellet, A., Salmon, J., and Clémençon, S. (2016). Gossip Dual Averaging for Decentralized Optimization of Pairwise Functions. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*.

Dimakis, A. G., Kar, S., Moura, J. M. F., Rabbat, M. G., and Scaglione, A. (2010). Gossip Algorithms for Distributed Signal Processing. *Proceedings of the IEEE*, 98(11):1847–1864.

Duchi, J. C., Agarwal, A., and Wainwright, M. J. (2012). Dual Averaging for Distributed Optimization: Convergence Analysis and Network Scaling. *IEEE Transactions on Automatic Control*, 57(3):592–606.

Iutzeler, F., Bianchi, P., Ciblat, P., and Hachem, W. (2013). Asynchronous Distributed Optimization using a Randomized Alternating Direction Method of Multipliers. In *Proceedings of the 52nd IEEE Conference on Decision and Control (CDC)*, pages 3671–3676.

Kempe, D., Dobra, A., and Gehrke, J. (2003). Gossip-Based Computation of Aggregate Information. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 482–491.

Li, C.-K., Tsai, M.-C., Wang, K.-Z., and Wong, N.-C. (2015). The spectrum of the product of operators, and the product of their numerical ranges. *Linear Algebra and its Applications*, 469:487 – 499.

McMahan, H. B., Moore, E., Ramage, D., and Agüera y Arcas, B. (2016). Federated Learning of Deep Networks using Model Averaging. Technical report, arXiv:1602.05629.

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

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.

Shah, D. (2009). Gossip Algorithms. *Foundations and Trends in Networking*, 3(1):1–125.

Shi, W., Ling, Q., Yuan, K., Wu, G., and Yin, W. (2014). On the linear convergence of the admm in decentralized consensus optimization. *IEEE Transactions on Signal Processing*, 62(7):1750–1761.

Wang, J., Kolar, M., and Srebro, N. (2016). Distributed Multi-Task Learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 751–760.

Wei, E. and Ozdaglar, A. E. (2012). Distributed Alternating Direction Method of Multipliers. In *Proceedings of the 51th IEEE Conference on Decision and Control (CDC)*, pages 5445–5450.

Wei, E. and Ozdaglar, A. E. (2013). On the O(1/k) Convergence of Asynchronous Distributed Alternating Direction Method of Multipliers. In *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*.

Zhou, D., Bousquet, O., Lal, T. N., Weston, J., and Schölkopf, B. (2004). Learning with local and global consistency. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS)*, volume 16, pages 321–328.

Appendix A Proof of Proposition 1

Proposition 1 (Closed-form solution)**.** *Let $P = D^{-1}W$ be the stochastic similarity matrix associated with the graph G and $\Theta^{sol} = [\theta_1^{sol}; \ldots; \theta_n^{sol}] \in \mathbb{R}^{n \times p}$. The solution $\Theta^\star = \arg\min_{\Theta \in \mathbb{R}^{n \times p}} \mathcal{Q}_{MP}(\Theta)$ is given by*

$$\Theta^\star = \bar{\alpha}(I - \bar{\alpha}(I - C) - \alpha P)^{-1} C \Theta^{sol},$$

with $\alpha \in (0, 1)$ such that $\mu = \bar{\alpha}/\alpha$, and $\bar{\alpha} = 1 - \alpha$.

Proof. We write the objective function in matrix form:

$$\mathcal{Q}_{MP}(\Theta) = \frac{1}{2}\left(\operatorname{tr}\left[\Theta^\top L \Theta\right] + \mu \operatorname{tr}\left[(\Theta - \Theta^{sol})^\top DC(\Theta - \Theta^{sol})\right]\right),$$

where $L = D - W$ is the graph Laplacian matrix and tr denotes the trace of a matrix. As $\mathcal{Q}_{MP}(\Theta)$ is convex and quadratic in Θ, we can find its global minimum by setting its derivative to 0.

$$\begin{aligned}
\nabla \mathcal{Q}_{MP}(\Theta) &= L\Theta + \mu DC(\Theta - \Theta^{sol}) \\
&= L\Theta^* + \mu DC(\Theta^* - \Theta^{sol}) \\
&= (D - W + \mu DC)\Theta^* - \mu DC\Theta^{sol}.
\end{aligned}$$

Hence,

$$\begin{aligned}
\nabla \mathcal{Q}_{MP}(\Theta) = 0 &\Leftrightarrow (I - P + \mu C)\Theta^* - \mu C\Theta^{sol} = 0 \\
&\Leftrightarrow (I - \bar{\alpha}(I - C) - \alpha P)\Theta^* - \bar{\alpha} C\Theta^{sol} = 0,
\end{aligned}$$

with $\mu = \bar{\alpha}/\alpha$. Since P is a stochastic matrix, its eigenvalues are in $[-1, 1]$. Moreover, $(I - C)_{ii} < 1$ for all i, thus $\rho(\bar{\alpha}(I - C) + \alpha P) < 1$ where $\rho(\cdot)$ denotes the spectral radius. Consequently, $I - \bar{\alpha}(I - C) - \alpha P$ is invertible and we get the desired result. □

Appendix B Convergence of the Iterative Form (5)

We can rewrite the equation

$$\Theta(t+1) = (\alpha I + \bar{\alpha} C)^{-1}\left(\alpha P \Theta(t) + \bar{\alpha} C \Theta^{sol}\right), \tag{5}$$

as

$$\Theta(t) = \left((\alpha I + \bar{\alpha} C)^{-1} \alpha P\right)^t \Theta(0) + \sum_{k=0}^{t-1} \left((\alpha I + \bar{\alpha} C)^{-1} \alpha P\right)^k (\alpha I + \bar{\alpha} C)^{-1} \bar{\alpha} C \Theta^{sol}.$$

Since $\frac{\alpha}{(\alpha + \bar{\alpha} c_i)} < 1$ for any $i \in [\![n]\!]$, we have $\rho\left((\alpha I + \bar{\alpha} C)^{-1} \alpha P\right) < 1$ and therefore:

$$\lim_{t \to \infty} \left((\alpha I + \bar{\alpha} C)^{-1} \alpha P\right)^t = 0,$$

hence

$$\begin{aligned}
\lim_{t \to \infty} \Theta(t) &= \left(I - (\alpha I + \bar{\alpha} C)^{-1} \alpha P\right)^{-1} (\alpha I + \bar{\alpha} C)^{-1} \bar{\alpha} C \Theta^{sol} \\
&= (I - \bar{\alpha}(I - C) - \alpha P)^{-1} \bar{\alpha} C \Theta^{sol} \\
&= \Theta^*.
\end{aligned}$$

Appendix C Proof of Theorem 1

Theorem 1 (Convergence)**.** *Let $\widetilde{\Theta}(0) \in \mathbb{R}^{n^2 \times p}$ be some arbitrary initial value and $(\widetilde{\Theta}(t))_{t \in \mathbb{N}}$ be the sequence generated by our model propagation algorithm. Let $\Theta^\star = \arg\min_{\Theta \in \mathbb{R}^{n \times p}} \mathcal{Q}_{MP}(\Theta)$ be the optimal solution to model propagation. For any $i \in [\![n]\!]$, we have:*

$$\lim_{t \to \infty} \mathbb{E}\left[\widetilde{\Theta}_i^j(t)\right] = \Theta_j^\star \text{ for } j \in \mathcal{N}_i \cup \{i\}.$$

Proof. In order to prove the convergence of our algorithm, we need to introduce an equivalent formulation as a random iterative process over $\widetilde{\Theta} \in \mathbb{R}^{n^2 \times p}$, the horizontal stacking of all the $\widetilde{\Theta}_i$'s.

The *communication step* of agent i with its neighbor j consists in overwriting $\tilde{\Theta}_i^j$ and $\tilde{\Theta}_j^i$ with respectively $\tilde{\Theta}_j^j$ and $\tilde{\Theta}_i^i$. This step will be handled by multiplication with the matrix $O(i,j) \in \mathbb{R}^{n^2 \times n^2}$ defined as

$$O(i,j) = I + e_i^j (e_j^j - e_i^j)^\top + e_j^i (e_i^i - e_j^i)^\top,$$

where for $i, j \in [\![n]\!]$, the vector $e_i^j \in \mathbb{R}^{n^2}$ has 1 as its $(i-1)n + j$-th coordinate and 0 in all others.

The *update step* of node i and j consists in replacing $\widetilde{\Theta}_i^i$ and $\widetilde{\Theta}_j^j$ with respectively the i-th line of $(\alpha I + \bar{\alpha} C)^{-1}\big(\alpha P \widetilde{\Theta}_i + \bar{\alpha} C \Theta^{sol}\big)$ and the j-th line of $(\alpha I + \bar{\alpha} C)^{-1}\big(\alpha P \widetilde{\Theta}_j + \bar{\alpha} C \Theta^{sol}\big)$. This step will be handled by multiplication with the matrix $U(i,j) \in \mathbb{R}^{n^2 \times n^2}$ and addition of the vector $u(i,j) \in \mathbb{R}^{n^2 \times p}$ defined as follows:

$$\begin{aligned} U(i,j) &= I + (e_i^i e_i^{i\top} + e_j^j e_j^{j\top})(M - I) \\ u(i,j) &= (e_i^i e_i^{i\top} + e_j^j e_j^{j\top})(\alpha I + \bar{\alpha} C)^{-1} \bar{\alpha} C \widetilde{\Theta}^{sol}, \end{aligned}$$

where $M \in \mathbb{R}^{n^2 \times n^2}$ is a block diagonal matrix with repetitions of $(\alpha I + \bar{\alpha} C)^{-1} \alpha P$ on the diagonal and $\widetilde{\Theta}^{sol} \in \mathbb{R}^{n^2 \times p}$ is built by stacking horizontally n times the matrix Θ^{sol}.

We can now write down a global iterative process which is equivalent to our model propagation algorithm. For any $t \geq 0$:

$$\tilde{\Theta}(t+1) = A(t)\tilde{\Theta}(t) + b(t)$$

where,

$$\begin{cases} A(t) = I^E U(i,j) O(i,j) \\ b(t) = u(i,j) \end{cases} \qquad \text{w.p. } \frac{\pi_i^j}{n} \text{ for } i, j \in [\![1, n]\!],$$

and I^E is a $n^2 \times n^2$ diagonal matrix with its $(i-1)n + j$-th value equal to 1 if $(i,j) \in E$ or $i = j$ and equal to 0 otherwise. Note that I^E is used simply to simplify our analysis by setting to 0 the lines of $A(t)$ corresponding to non-existing edges (which can be safely ignored).

First, let us write the expected value of $\tilde{\Theta}(t)$ given $\tilde{\Theta}(t-1)$:

$$\mathbb{E}\left[\tilde{\Theta}(t) | \tilde{\Theta}(t-1)\right] = \mathbb{E}[A(t)]\tilde{\Theta}(t-1) + \mathbb{E}[b(t)]. \tag{9}$$

Since the $A(t)$'s and $b(t)$'s are i.i.d., for any $t \geq 0$ we have $\bar{A} = \mathbb{E}[A(t)]$ and $\bar{b} = \mathbb{E}[b(t)]$ where

$$\begin{aligned} \bar{A} &= \frac{1}{n} I^E \sum_{i,j}^{n} \pi_i^j U(i,j) O(i,j), \\ \bar{b} &= \frac{1}{n} \sum_{i,j}^{n} \pi_i^j u(i,j). \end{aligned}$$

In order to prove Theorem 1, we first need to show that $\rho(\bar{A}) < 1$, where $\rho(\bar{A})$ denotes the spectral radius of $\bar{A}$. First, recall that $\rho\left((\alpha I + \bar{\alpha} C)^{-1} \alpha P\right) < 1$ (see Appendix Appendix B). We thus have $\rho(M) < 1$ by construction of M and

$$\lambda(I - M) \subset (0, 2),$$

where $\lambda(\cdot)$ denotes the spectrum of a matrix. Furthermore, from the properties in Li et al. (2015) we know that

$$\lambda\left((e_i^i e_i^{i\top} + e_j^j e_j^{j\top})(I - M)\right) \subset [0, 2],$$

and finally we have:

$$\lambda\left(I + (e_i^i e_i^{i\top} + e_j^j e_j^{j\top})(M - I)\right) = \lambda(U(i,j)) \subset [-1, 1].$$

As we also have $\lambda(O(i,j)) \subset [0,1]$ therefore

$$\lambda\left(U(i,j)O(i,j)\right) \subset [-1,1].$$

Let us first suppose that -1 is an eigenvalue of $U(i,j)O(i,j)$ associated with the eigenvector $\tilde{v}$. From the previous inequalities we deduce that $\tilde{v}$ must be an eigenvector of $O(i,j)$ associated with the eigenvalue $+1$ and an eigenvector of $U(i,j)$ associated with the eigenvalue -1. Then from $\tilde{v} = O(i,j)\tilde{v}$ we have $\tilde{v}_i^j = \tilde{v}_j^j$ and $\tilde{v}_j^i = \tilde{v}_i^i$. From $-\tilde{v} = U(i,j)\tilde{v}$ we can deduce that $\tilde{v}_k^l = 0$ for any $k \neq l$ or $k = l \in [\![n]\!] \backslash \{i,j\}$. Finally we can see that $\tilde{v}_i^i = \tilde{v}_j^j = 0$ and therefore $\tilde{v} = 0$. This proves by contradiction that -1 is not an eigenvalue of $U(i,j)O(i,j)$ and furthermore that -1 is not an eigenvalue of $\bar{A}$.

Let us now suppose that $+1$ is an eigenvalue of $\bar{A}$, associated with the eigenvector $\tilde{v} \in \mathbb{R}^{n^2}$. This would imply that

$$\tilde{v} = \bar{A}\tilde{v} = \frac{1}{n} I^E \sum_{i,j}^{n} \pi_i^j U(i,j)O(i,j)\tilde{v}.$$

This can be expressed line by line as the following set of equations:

$$\begin{aligned}
\sum_{k=1}^{n}(\pi_1^k + \pi_k^1)\tilde{v}_1^1 &= e_1^{1\top}\sum_{k=1}^{n}(\pi_1^k + \pi_k^1)MO(1,k)\tilde{v} \\
(\pi_1^2 + \pi_2^1)\tilde{v}_1^2 &= (\pi_1^2 + \pi_2^1)\tilde{v}_2^2 && \text{if } (1,2) \in E && \text{else } \tilde{v}_1^2 = 0 \\
(\pi_1^3 + \pi_3^1)\tilde{v}_1^3 &= (\pi_1^3 + \pi_3^1)\tilde{v}_3^3 && \text{if } (1,3) \in E && \text{else } \tilde{v}_1^3 = 0 \\
&\vdots \\
(\pi_2^1 + \pi_1^2)\tilde{v}_2^1 &= (\pi_2^1 + \pi_1^2)\tilde{v}_1^1 && \text{if } (2,1) \in E && \text{else } \tilde{v}_2^1 = 0 \\
\sum_{k=1}^{n}(\pi_2^k + \pi_k^2)\tilde{v}_2^2 &= e_2^{2\top}\sum_{k=1}^{n}(\pi_2^k + \pi_k^2)MO(2,k)\tilde{v} \\
(\pi_2^3 + \pi_3^2)\tilde{v}_2^3 &= (\pi_2^3 + \pi_3^2)\tilde{v}_3^3 && \text{if } (2,3) \in E && \text{else } \tilde{v}_2^3 = 0 \\
&\vdots \\
(\pi_n^{n-2} + \pi_{n-2}^n)\tilde{v}_n^{n-2} &= (\pi_n^{n-2} + \pi_{n-2}^n)\tilde{v}_{n-2}^{n-2} && \text{if } (n,n-2) \in E && \text{else } \tilde{v}_n^{n-2} = 0 \\
(\pi_n^{n-1} + \pi_{n-1}^n)\tilde{v}_n^{n-1} &= (\pi_n^{n-1} + \pi_{n-1}^n)\tilde{v}_{n-1}^{n-1} && \text{if } (n,n-1) \in E && \text{else } \tilde{v}_n^{n-1} = 0 \\
\sum_{k=1}^{n}(\pi_n^k + \pi_k^n)\tilde{v}_n^n &= e_n^{n\top}\sum_{k=1}^{n}(\pi_n^k + \pi_k^n)MO(n,k)\tilde{v}
\end{aligned}$$

We can rewrite the above system as

$$\begin{aligned}
\tilde{v}_i^j &= \begin{cases} v_j & \text{if } (i,j) \in E \text{ or } i = j \\ 0 & \text{otherwise,} \end{cases} \\
0 &= \left(I - (\alpha I + \bar{\alpha}C)^{-1}\alpha P\right) v.
\end{aligned} \tag{10}$$

with $v \in \mathbb{R}^{n \times p}$. As seen in Appendix Appendix B, the matrix $I - \bar{\alpha}(I - C) - \alpha P$ is invertible. Consequently $v = 0$ and thus $\tilde{v} = 0$, which proves by contradiction that $+1$ is not an eigenvalue of $\bar{A}$.

Now that we have shown that $\rho(\bar{A}) < 1$, let us write the expected value of $\tilde{\Theta}(t)$ by "unrolling" the recursion (9):

$$\mathbb{E}\left[\tilde{\Theta}(t)\right] = \bar{A}^t\tilde{\Theta}(0) + \sum_{k=0}^{t-1}\bar{A}^k\bar{b}.$$

Let us denote $\tilde{\Theta}^* = \lim_{t\to\infty}\mathbb{E}\left[\tilde{\Theta}(t)\right]$. Because $\rho(\bar{A}) < 1$, we can write

$$\tilde{\Theta}^* = (I - \bar{A})^{-1}\bar{b},$$

and finally

$$(I - \bar{A})\tilde{\Theta}^* = \bar{b}.$$

Similarly as in (10), we can identify $\hat{\Theta} \in \mathbb{R}^{n \times p}$ such that

$$\tilde{\Theta}_i^{j*} = \begin{cases} \hat{\Theta}_j & \text{if } (i,j) \in E \text{ or } i = j \\ 0 & \text{otherwise,} \end{cases}$$
$$\bar{\alpha} C \Theta^{sol} = (I - \bar{\alpha}(I - C) + \alpha P)\hat{\Theta}.$$

Recalling the results from Appendix Appendix A, we have

$$\hat{\Theta} = \bar{\alpha}(I - \bar{\alpha}(I - C) - \alpha P)^{-1} C \Theta^{sol},$$

and we thus have

$$\hat{\Theta} = \Theta^* = \operatorname*{arg\,min}_{\Theta \in \mathbb{R}^{n \times p}} \mathcal{Q}_{MP}(\Theta),$$

and the theorem follows. □

Appendix D Synchronous Decentralized ADMM Algorithm for Collaborative Learning

For completeness, we present here the *synchronous* decentralized ADMM algorithm for collaborative learning. Based on our reformulation of Section 4.2 and following Wei and Ozdaglar (2012), the algorithm to find $\widetilde{\Theta}^\star$ consists in iterating over the following steps, starting at $t = 0$:

1. Every agent $i \in [\![n]\!]$ updates its primal variables:

   $$\widetilde{\Theta}_i(t+1) = \operatorname*{arg\,min}_{\Theta \in \mathbb{R}^{(|\mathcal{N}_i|+1) \times p}} L_\rho^i(\Theta, Z_i(t), \Lambda_i(t)),$$

   and sends $\widetilde{\Theta}_i^i(t+1), \widetilde{\Theta}_i^j(t+1), \Lambda_{ei}^i(t), \Lambda_{ei}^j(t)$ to agent j for all $j \in \mathcal{N}_i$.

2. Using values received by its neighbors, every agent $i \in [\![n]\!]$ updates its secondary variables for all $e = (i,j) \in E$ such that $j \in \mathcal{N}_i$:

   $$Z_{ei}^i(t+1) = \frac{1}{2}\left[\frac{1}{\rho}\left(\Lambda_{ei}^i(t) + \Lambda_{ej}^i(t)\right) + \widetilde{\Theta}_i^i(t+1) + \widetilde{\Theta}_j^i(t+1)\right],$$
   $$Z_{ei}^j(t+1) = \frac{1}{2}\left[\frac{1}{\rho}\left(\Lambda_{ej}^j(t) + \Lambda_{ei}^j(t)\right) + \widetilde{\Theta}_j^j(t+1) + \widetilde{\Theta}_i^j(t+1)\right].$$

   By construction, this update maintains $Z(t+1) \in \mathcal{C}_E$.

3. Every agent $i \in [\![n]\!]$ updates its dual variables for all $e = (i,j) \in E$ such that $j \in \mathcal{N}_i$:

   $$\Lambda_{ei}^i(t+1) = \Lambda_{ei}^i(t) + \rho\big(\widetilde{\Theta}_i^i(t+1) - Z_{ei}^i(t+1)\big),$$
   $$\Lambda_{ei}^j(t+1) = \Lambda_{ei}^j(t) + \rho\big(\widetilde{\Theta}_i^j(t+1) - Z_{ei}^j(t+1)\big).$$

Synchronous ADMM is known to converge to an optimal solution at rate $O(1/t)$ when the objective function is convex (Wei and Ozdaglar, 2012), and at a faster (linear) rate when it is strongly convex (Shi et al., 2014). However, it requires global synchronization across the network, which can be very costly in practice.

Appendix E Additional Experimental Results

Target models in collaborative linear classification For the experiment of Section 5.2, Figure 4 shows the target models of the agents as well as the links between them. We can see that the target models can be very different from an agent to another, and that two agents are linked when there is a small enough (yet non-negligible) angle between their target models.

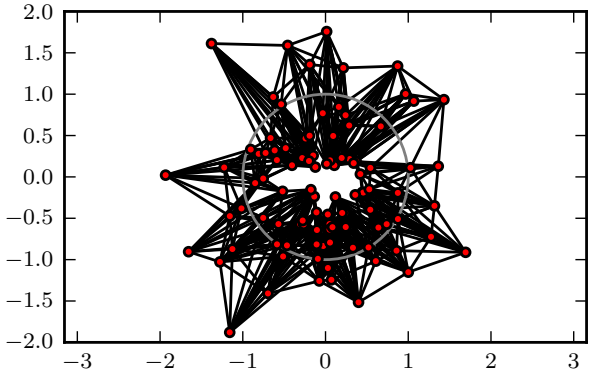


Figure 4: Target models of the agents (represented as points in $\mathbb{R}^2$) for the collaborative linear classification task. Two models are linked together when the angle between them is small, which corresponds to a small Euclidean distance after projection onto the unit circle.

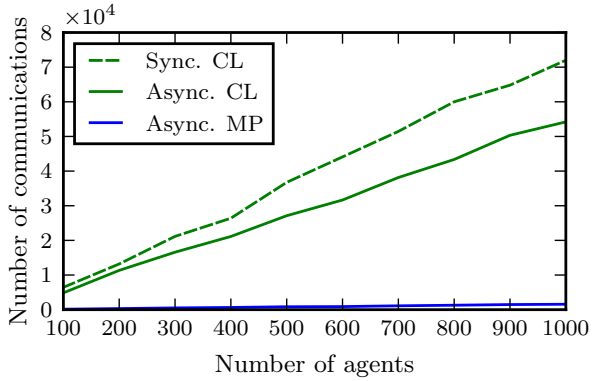


Figure 5: Number of pairwise communications needed to reach 90% of the accuracy of the optimal models with varying number of agents (linear classification task, $p = 50$).

Scalability with respect to the number of nodes In this experiment, we study how the number of iterations needed by our decentralized algorithms to converge to good solutions scale with the size of the network. We focus on the collaborative linear classification task introduced in Section 5.2 with the number n of agents ranging from 100 to 1000. The network is a k-nearest neighbor graph: each agent is linked to the k agents for which the angle similarity introduced in Section 5.2 is largest, and $W_{ij} = 1$ if i and j are neighbors and 0 otherwise.

Figure 5 shows the number of iterations needed by our algorithms to reach 90% of the accuracy of the optimal set of models. We can see that the number of iterations scales linearly with n. In asynchronous gossip algorithms, the number of iterations that can be done in parallel also scales roughly linearly with n, so we can expect our algorithms to scale nicely to very large networks.