COLLABORATIVE MACHINE LEARNING IN LARGE-SCALE PEER-TO-PEER DISTRIBUTED SYSTEMS

Aurélien Bellet (Inria MAGNET)

Joint work with:
M. Tommasi, P. Vanhaesebrouck (Inria, Univ. Lille)
R. Guerraoui, M. Taziki (EPFL)
V. Zantedeschi (Univ. St-Etienne)

4th GDR RSD and ASF Winter School on Distributed Systems and Networks
February 4-8, 2019

A BIT ABOUT MYSELF

- Researcher (CRCN) at Inria since 2015
- Working on **machine learning:** mainly theory and algorithms, also some applications (speech, NLP)
- Working on distributed ML since 2014
- I am not a distributed computing or systems expert

OUTLINE

1. A Formal Introduction to ML

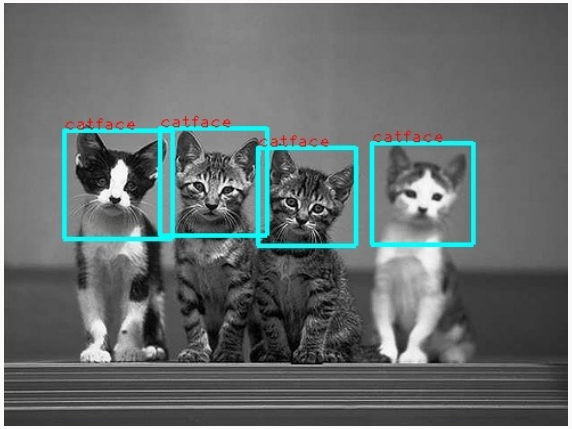
2. P2P, Personalized and Private ML

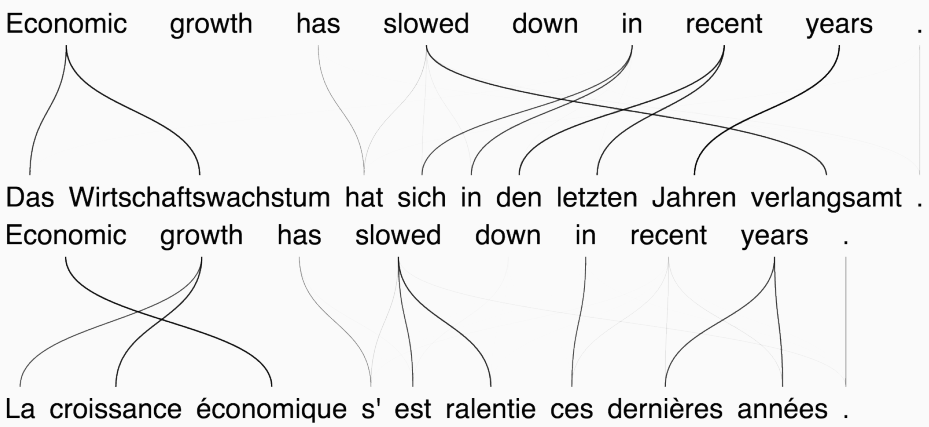
A FORMAL INTRODUCTION TO ML

MACHINE LEARNING

- ML provides a computer with the ability to do certain tasks without being explicitly programmed for it
- This is done by learning from data
- Multidisciplinary field: computer science, statistics, optimization
- ML is fueling the current progress in AI

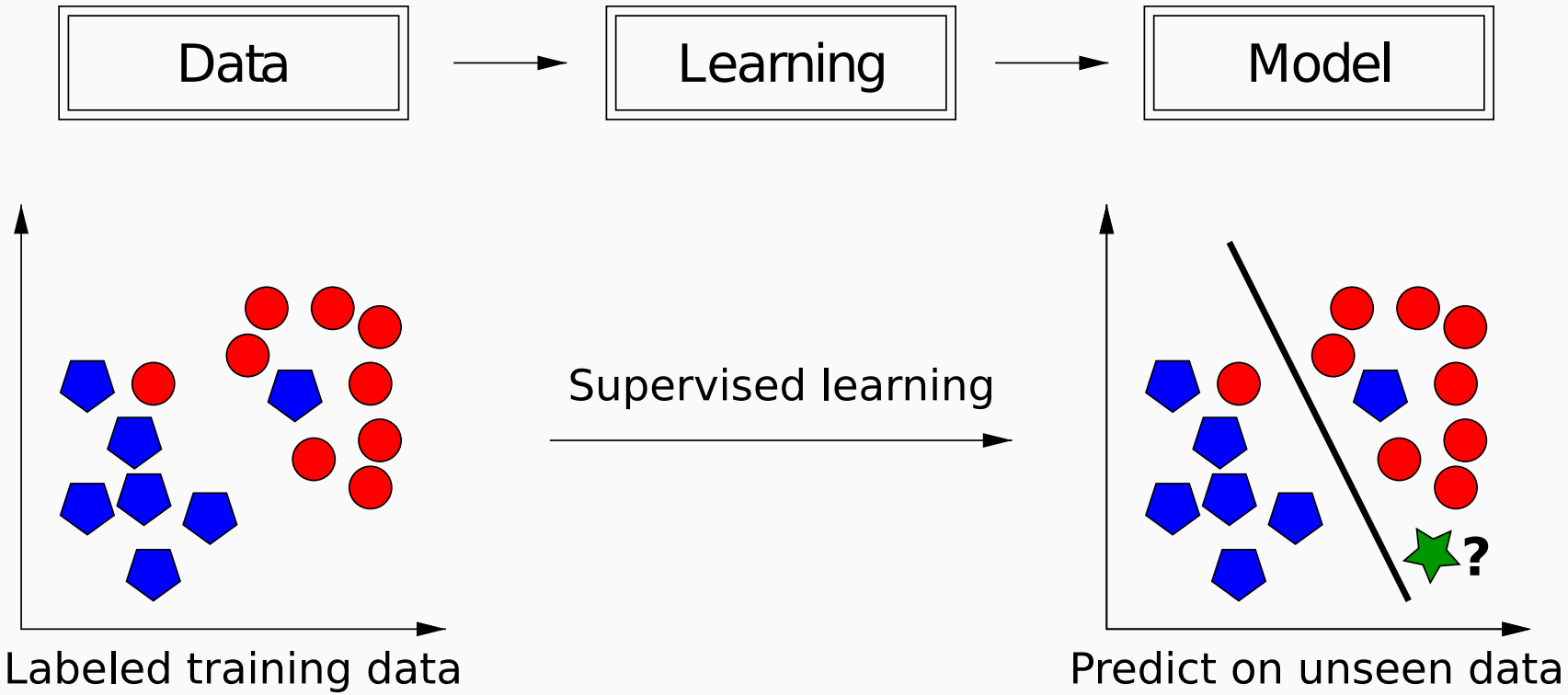
EXAMPLE APPLICATIONS





- Requires large amounts of potentially sensitive, personal data
- Let's not care for now: say we collected data on a central server

SUPERVISED LEARNING



SUPERVISED LEARNING: NOTATIONS

- Labeled data point $(x, y) \in \mathcal{X} \times \mathcal{Y}$
- $\mathcal{X} \subset \mathbb{R}^q$: representation space (features)
- $\mathcal{Y}$: discrete (classification) or continuous (regression)
- A predictive model is a function $f : \mathcal{X} \to \mathcal{Y}$
- We measure the discrepancy between the prediction $f(x)$ and the true label y using a loss function $\ell(f; x, y)$

SUPERVISED LEARNING: SETUP

- We have access to a training dataset $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^m$ of m labeled points
- A supervised ML algorithm takes $\mathcal{S}$ as input and outputs a model $f : \mathcal{X} \to \mathcal{Y}$
- The learned model f can then be used to predict a label $y \in \mathcal{Y}$ for any (new) data point $x \in \mathcal{X}$

The goal of ML is to generalize to unseen data

$\to$ **need an assumption to relate training data and future data**

SUPERVISED LEARNING: STATISTICAL SETTING

- **Assumption:** all data points $(x, y) \in \mathcal{X} \times \mathcal{Y}$ follows some unknown but fixed distribution μ (specific to the task)
- This is assumed to hold for both training and unseen data
- **Goal**: learn a model f in some model family $\mathcal{F}$ from training data which has small expected loss over μ:

$$R(f) = \mathop{\mathbb{E}}_{(x,y)\sim\mu} \ell(f; x, y)$$

- But μ is unknown, so cannot compute $R(f)$

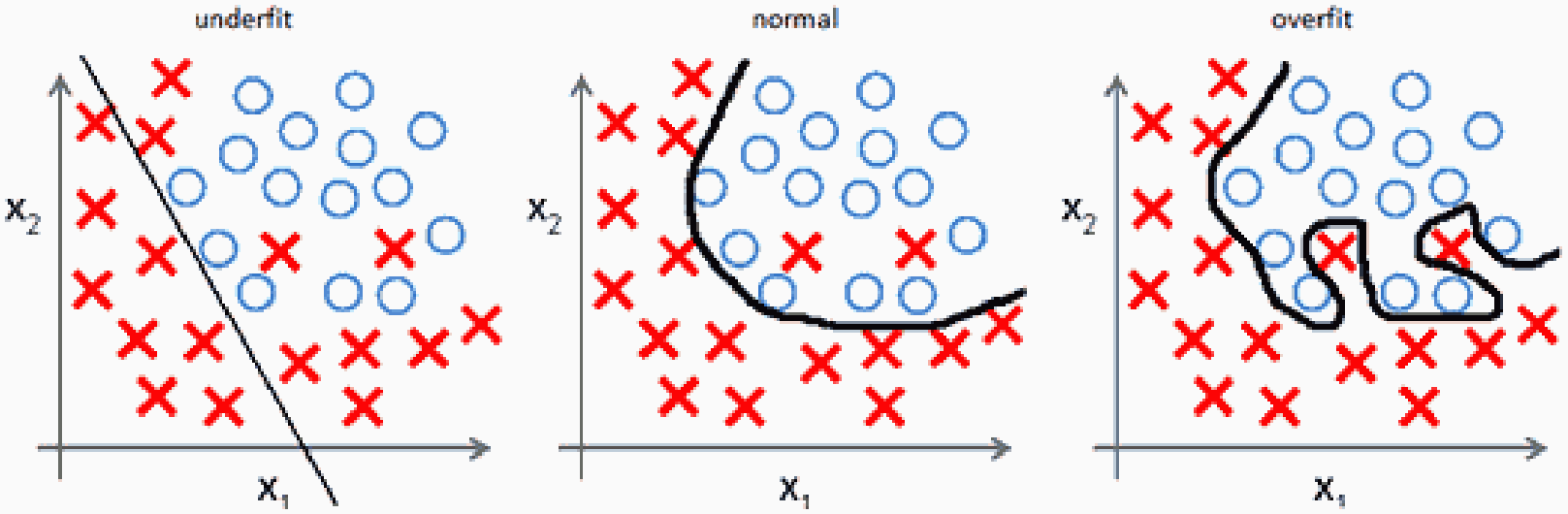
EMPIRICAL RISK MINIMIZATION

- Intuitive idea: minimize average loss on training data

$$\hat{f} \in \arg\min_{f \in \mathcal{F}} \hat{R}(f) = \frac{1}{m} \sum_{i=1}^{m} \ell(f; x_i, y_i)$$

- The hope is that an accurate model on training data will also do well on unseen data
- Why do we care about the model family $\mathcal{F}$? Can't we use a very expressive family which can model any data?

APPROXIMATION-GENERALIZATION TRADE-OFF



- $\mathcal{F}$ too simple $\rightarrow$ underfitting
- $\mathcal{F}$ too complex $\rightarrow$ overfitting
- Note: the complexity of $\mathcal{F}$ also impacts the algorithmic complexity of the learning procedure

STATISTICAL LEARNING GUARANTEES

- This trade-off is well-explained by statistical learning theory
- One can prove results of the form: for any $f \in \mathcal{F}$, w.p. $1 - \delta$

$$R(f) \leq \hat{R}(f) + \sqrt{\frac{C_{\mathcal{F}} \log(1/\delta)}{m}}$$

  where $C_{\mathcal{F}}$ is a measure of complexity of the model class $\mathcal{F}$
- $C_{\mathcal{F}}$ can simply be $|\mathcal{F}|$ when model family is finite
- Note: regularization can be used to penalize complexity within $\mathcal{F}$

$$\hat{f} \in \arg\min_{f \in \mathcal{F}} \hat{R}(f) + \lambda \Omega(f)$$

ERM EXAMPLE 1: LINEAR REGRESSION

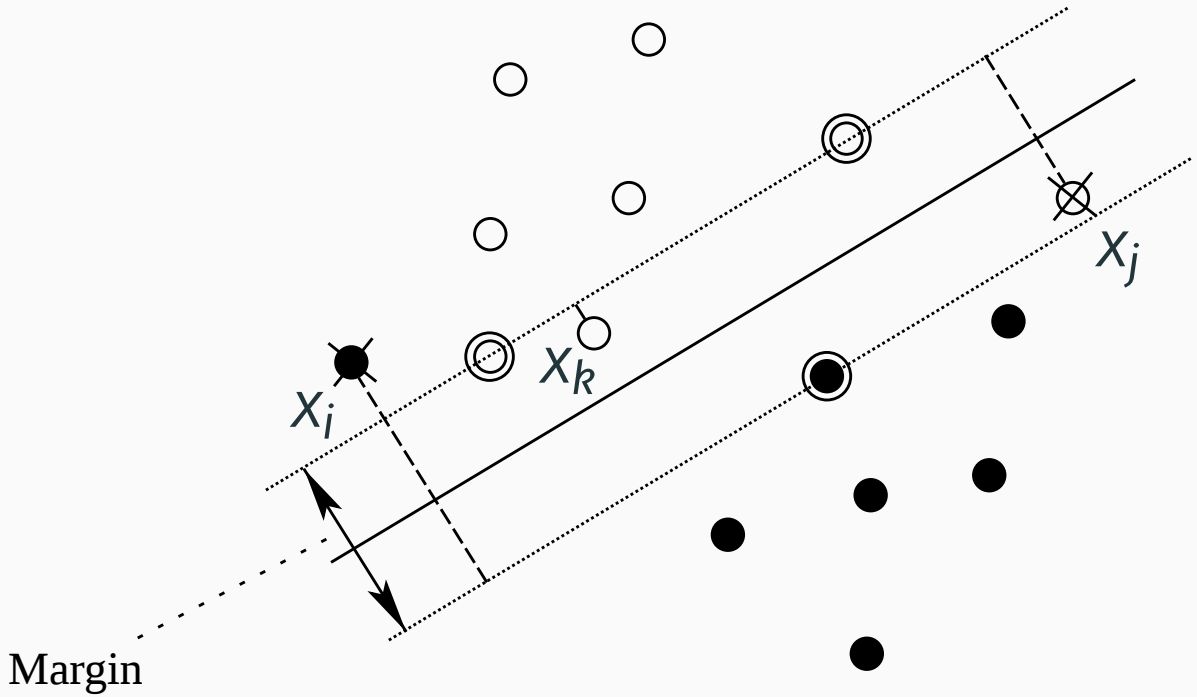
- Real labels space $\mathcal{Y} = \mathbb{R}$
- Linear model $f_\theta(x) = \theta^T x$ parameterized by $\theta \in \mathbb{R}^q$
- Quadratic loss $\ell(f_\theta; x, y) = (y - f_\theta(x))^2$
- ERM problem is a simple least-square problem:

$$\hat{\theta} \in \arg\min_{\theta \in \mathbb{R}^p} \frac{1}{m} \sum_{i=1}^{m} (y_i - f_\theta(x_i))^2, \text{ equivalent to } \hat{\theta} \in \arg\min_{\theta \in \mathbb{R}^p} \|Y - X\theta\|_2^2$$

- Common regularization terms:
  - Squared L2 norm: $\|\theta\|_2^2$
  - L1 norm: $\|\theta\|_1$ (sparsity inducing, cf LASSO)

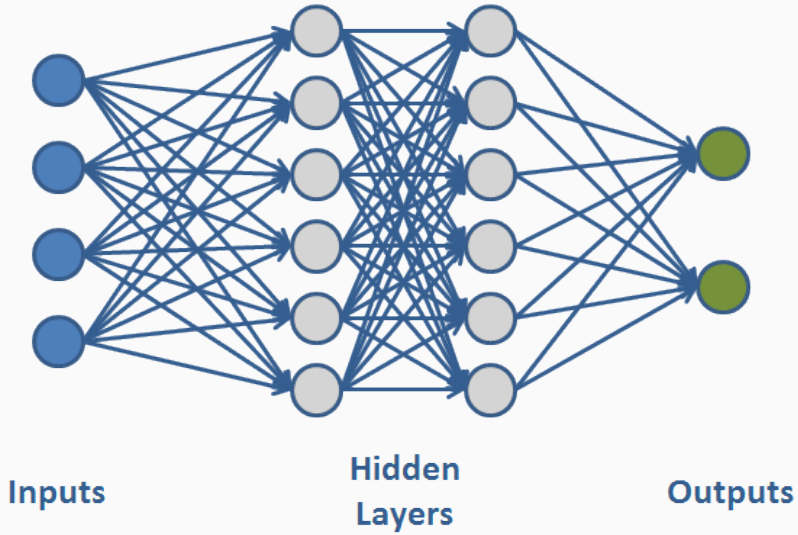
ERM EXAMPLE 2: LINEAR CLASSIFICATION WITH HINGE LOSS

- Binary labels $\mathcal{Y} = \{-1, 1\}$, linear model $f_\theta(x) = \text{sign}[\theta^T x]$
- Hinge loss $\ell(f_\theta; x, y) = \max(0, 1 - y\theta^T x)$ to enforce a safety margin
- ERM problem with L2 regularization is Support Vector Machine



ERM EXAMPLE 3: DEEP NEURAL NETS

- Feed-forward, fully connected DNN:
  - First layer is the input $x_0 = x$
  - Intermediate layers: $x_i = \sigma(W_i x_{i-1})$ with σ nonlinear mapping
  - Last layer: linear model on previous layer + loss



- Specialized networks: CNNs (images), LTSMs/RNNs (sequences)...
- High model complexity, but can still generalize well in practice!
- A lot of ongoing work to better understand this theoretically

SOLVING ERM PROBLEMS: GRADIENT-BASED METHODS

- Assume $\mathcal{F} = \{f_\theta : \theta \in \mathbb{R}^p\}$, the ERM problem is

$$\min_{\theta \in \mathbb{R}^p} \hat{R}(\theta) = \frac{1}{m} \sum_{i=1}^{m} \ell(\theta; x_i, y_i)$$

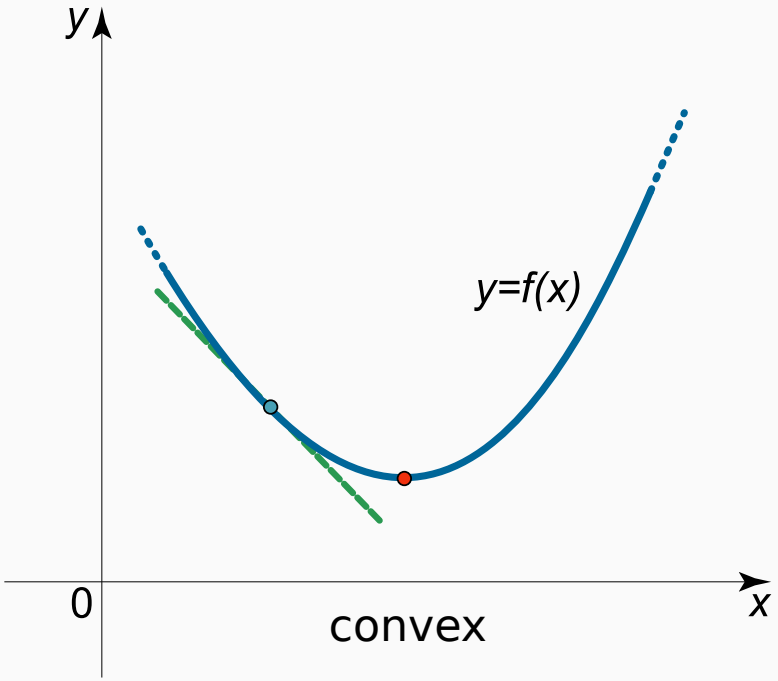
- We typically work with loss functions that are differentiable in θ
- The workhorse of ML is first-order optimization methods: iteratively refine θ based on (an estimate of) the gradient

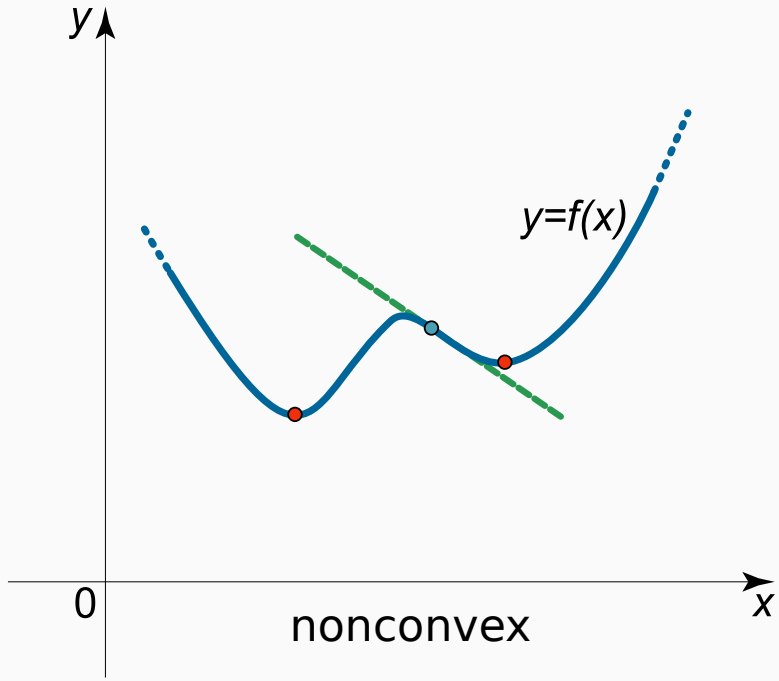
$$\nabla \hat{R}(\theta) = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \ell(\theta; x_i, y_i)$$

Solving ERM problems: gradient-based methods

- Gradient Descent (GD):
  - Initialize to some $\theta(0) \in \mathbb{R}^p$
  - For $t = 0, \ldots, T$: update $\theta(t+1) = \theta(t) - \gamma \nabla \hat{R}(\theta(t))$
- Stochastic Gradient Descent (SGD):
  - Initialize to some $\theta(0) \in \mathbb{R}^p$
  - For $t = 0, \ldots, T$: pick random index $i_t \in \{1, \ldots, m\}$ and update $\theta(t+1) = \theta(t) - \gamma \nabla_\theta \ell(\theta; x_{i_t}, y_{i_t})$
- γ is the step size (or learning rate) to be tuned
- In ML, we typically do not care about high-precision solutions
- For large datasets, SGD has much cheaper iterations and converges faster to a solution with reasonable precision

SOLVING ERM PROBLEMS: GRADIENT-BASED METHODS





- For convex objective functions, gradient-based methods will converge to the global minimum (under appropriate step size)
- Nonconvex case: convergence only to a local minimum
- Convergence rate depends on properties of the objective
- Note: optimization for ML is a very active topic

P2P, PERSONALIZED AND PRIVATE ML

LEARNING ON CONNECTED DEVICES DATA

- Connected devices are widespread and collect increasingly personal data
- Ex: browsing logs, health, speech, accelerometer, geolocation
- Great opportunity to provide personalized services
- Two classic strategies:
  - Centralize data from all devices: limited user control, privacy and security issues, communication/infrastructure costs
  - Learn on each device separately: poor utility for many users
- **Our goal**: show that decentralized collaborative learning gives a sweet spot between these two extremes

DESIRED PROPERTIES OF THE ALGORITHM

1. Keep data on the device of the users
2. Train personalized models in collaborative fashion
3. Learn and leverage similarities between users
4. Scale to a large number of devices
5. Ensure formal privacy guarantees

SOME RELATED WORK

Federated learning [McMahan et al., 2017]

- Coordinator-clients architecture, synchronous
- Does not scale very well with number of clients
- Typically learn a single global model for all users

Decentralized learning [Lian et al., 2017]

- Local exchanges, no coordinator
- Also learn a single global model
- Often no formal privacy guarantee

- We have a set $V = [\![n]\!] = \{1, \dots, n\}$ of n learning agents, assumed to be honest-but-curious
- Agent i has dataset $\mathcal{S}_i = \{(x_i^j, y_i^j)\}_{j=1}^{m_i}$ of size $m_i \geq 0$ drawn from its personal distribution
- In isolation, agent i can learn a purely local model by ERM

$$\theta_i^{loc} \in \arg\min_{\theta \in \mathbb{R}^p} \hat{R}_i(\theta) = \frac{1}{m_i} \sum_{j=1}^{m_i} \ell(\theta; x_i^j, y_i^j) + \lambda_i \|\theta\|^2, \text{ with } \lambda_i \geq 0$$

- **Goal:** improve upon θ_i^{loc} with the help of other agents

- Collaboration graph: sparse nonnegative weights $w \in \mathbb{R}^{K(K-1)/2}$ representing pairwise similarities between agents' objectives
- Each agent i only needs a local view of the network: its neighborhood $\mathcal{N}_i = \{j \neq i : w_{ij} > 0\}$ and the associated weights
- The collaboration graph can be thought of as an overlay over the physical communication network
- We assume for now that the collaboration graph is given

LEARNING THE MODELS: OBJECTIVE FUNCTION

- Find personalized models $\Theta \in \mathbb{R}^{n \times p}$ as solutions to [Vanhaesebrouck et al., 2017]:

$$\min_{\Theta \in \mathbb{R}^{n \times p}} \mathcal{Q}_{CL}(\Theta) = \sum_{i=1}^{n} d_i c_i \hat{R}_i(\theta_i) + \mu \sum_{i<j}^{n} w_{ij} \|\theta_i - \theta_j\|^2$$

- $c_i \in (0, 1] \propto m_i$: confidence of agent i, $d_i = \sum_{j \neq i} w_{ij}$: degree of i
- Trade-off between having accurate models on local dataset and similar models for strongly connected agents
- Hyperparameter μ interpolates between learning purely local models ($\mu \to \infty$) and learning a consensus model ($\mu \to 0$)
- Objective function is convex

DECENTRALIZED ALGORITHM

- **Asynchronous time model**: each agent has a local Poisson clock and wakes up when it ticks
- Equivalently: single clock (with counter t, unknown to the agents) ticking when one of the local clocks ticks
- **1-hop communication model**: the agent who wakes up exchanges messages with its direct neighbors
- Note: we also have gossip algorithms [Vanhaesebrouck et al., 2017]

DECENTRALIZED ALGORITHM

- Initialize models $\Theta_i(0) \in \mathbb{R}^{n \times p}$
- At step $t \geq 0$, a random agent i wakes up:
  1. Agent i updates its model based on information from neighbors:

$$\Theta_i(t+1) = \Theta_i(t) - \gamma \left(c_i \nabla \hat{R}_i(\Theta_i(t)) - \mu \sum_{j \in \mathcal{N}_i} \frac{w_{ij}}{d_i} \Theta_j(t) \right)$$

  2. Agent i sends its updated model $\Theta_i(t+1)$ to its neighborhood $\mathcal{N}_i$
- The update is a trade-off between a local gradient step and a weighted average of neighbors' models

Proposition ([Bellet et al., 2018])

For any $T > 0$, let $(\Theta(t))_{t=1}^{T}$ be the sequence of iterates generated by the algorithm running for T iterations from an initial point $\Theta(0)$. For strongly convex $\mathcal{Q}_{CL}$, we have for some $\rho \in (0, 1)$:

$$\mathbb{E}\left[\mathcal{Q}_{CL}(\Theta(T)) - \mathcal{Q}_{CL}^{\star}\right] \leq \rho^{T}\left(\mathcal{Q}_{CL}(\Theta(0)) - \mathcal{Q}_{CL}^{*}\right)$$

- Fast convergence rate: expected suboptimality gap shrinks by a constant factor ρ at each iteration
- ρ depends on problem-related quantities (we omit the details)

Wait! What about privacy?

WHAT ABOUT PRIVACY?

- In some applications, data may be sensitive and agents may not want to reveal it to anyone else
- In the previous algorithm, agents never communicate their local data but exchange sequences of models computed from data
- Consider an adversary observing all the information sent over the network (but not the internal memory of agents)
- **Goal:** formally guarantee that no/little information about the local dataset is leaked by the algorithm

(ϵ, δ)-Differential Privacy [Dwork, 2006]

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 $\mathcal{S}, \mathcal{S}'$ differing in a single data point and for all sets of possible outputs $\mathcal{O} \subseteq \text{range}(\mathcal{M})$, we have:

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

- Guarantees that the output of $\mathcal{M}$ is almost the same regardless of whether a particular data point was used
- Robust to background knowledge that adversary may have
- Information-theoretic (no computational assumptions)
- Composition property: the combined output of two (ϵ, δ)-DP mechanisms (run on the same dataset) is $(2\epsilon, 2\delta)$-DP

DIFFERENTIALLY PRIVATE ALGORITHM

1. Replace the update of the algorithm by

$$\widetilde{\Theta}_i(t+1) = \widetilde{\Theta}_i(t) - \gamma\Big(c_i(\nabla \hat{R}_i(\widetilde{\Theta}_i(t)) + \eta_i) - \mu \sum_{j \in \mathcal{N}_i} \frac{W_{ij}}{d_i} \widetilde{\Theta}_j(t)\Big),$$

   where $\eta_i \sim Laplace(0, s_i)^p \in \mathbb{R}^p$

2. Agent i then broadcasts noisy iterate $\widetilde{\Theta}_i(t+1)$ to its neighbors

- In our setting, the output of our algorithm is the sequence of agents' models sent over the network

Theorem ([Bellet et al., 2018])

Assume agent i wakes up T_i times and use noise scale $s_i = \frac{L_0}{\epsilon_i m_i}$. Then for any initial point $\widetilde{\Theta}(0)$ independent of $\mathcal{S}_i$, the algorithm is $(\bar{\epsilon}_i, 0)$-DP with $\bar{\epsilon}_i = T_i \epsilon_i$.

PRIVACY/UTILITY TRADE-OFF

Theorem ([Bellet et al., 2018])

For any $T > 0$, let $(\widetilde{\Theta}(t))_{t=1}^{T}$ be the sequence of iterates generated by T iterations. We have:

$$\mathbb{E}\left[\mathcal{Q}_{CL}(\widetilde{\Theta}(T)) - \mathcal{Q}_{CL}^{\star}\right] \leq \rho^{T}\left(\mathcal{Q}_{CL}(\widetilde{\Theta}(0)) - \mathcal{Q}_{CL}^{\star}\right) + \left(\frac{1}{(1-\rho)Cn}\sum_{i=1}^{n}\left(\mu D_{ii} c_i s_i\right)^2\right)\left(1-\rho^{T}\right)$$

- Second term gives additive error due to noise
- Sweet spot: the less data, the more noise added by the agent, but the least influence in the network
- T rules a trade-off between optimization error and noise error

BACK TO THE COLLABORATION GRAPH

- We have assumed the collaboration graph to be given
- In some applications, it can be constructed based on side information on the agents (e.g., user profiles)
- Unfortunately, this is not always available or reliable
- We propose to learn the collaboration graph from data along with the models

JOINT LEARNING OF MODELS AND COLLABORATION GRAPH

- Joint problem over Θ and w [Zantedeschi et al., 2019]:

$$\min_{\substack{\Theta \in \mathbb{R}^{n \times p} \\ w \in \mathbb{R}_+^{K(K-1)/2}}} \sum_{i=1}^{n} d_i c_i \hat{R}_i(\theta_i) + \frac{\mu}{2} \Big(\sum_{i<j} w_{ij} \|\theta_i - \theta_j\|^2 + \beta \|w\|^2 - 1^T \log(d) \Big)$$

- Log barrier on the degree vector d to avoid isolated agents
- L_2 penalty on weights to tune the graph sparsity
- Assign large weights to agents with similar models, unless their weighted loss is large
- Objective function is nonconvex but each subproblem (fixing one variable and optimizing the other) is convex
- We solve this by alternating optimization on Θ and w (note that we have already seen how to find Θ given w)

DECENTRALIZED ALGORITHM FOR LEARNING THE GRAPH

- We want to find new graph weights w given models Θ
- We thus need agents to communicate beyond their neighbors in the current collaboration graph
- We rely on a peer sampling service [Jelasity et al., 2007], a classic distributed systems primitive allowing an agent to communicate with a random set of peers

DECENTRALIZED ALGORITHM FOR LEARNING THE GRAPH

- Initialize weights $w^{(0)}$, set parameter $\kappa \in \{1, \dots, n-1\}$
- At each step $t \geq 0$, a random agent i wakes up:
  1. Draw κ agents and request their model, loss value and degree
  2. Update associated weights using a gradient descent step
  3. Send each updated weight to the associated agent
- We again show a fast convergence rate, where κ rules a trade-off between communication cost and convergence speed [Zantedeschi et al., 2019]
- Differential privacy still holds: we are only "post-processing" the private models

RELATION TO SELF ORGANIZING OVERLAYS

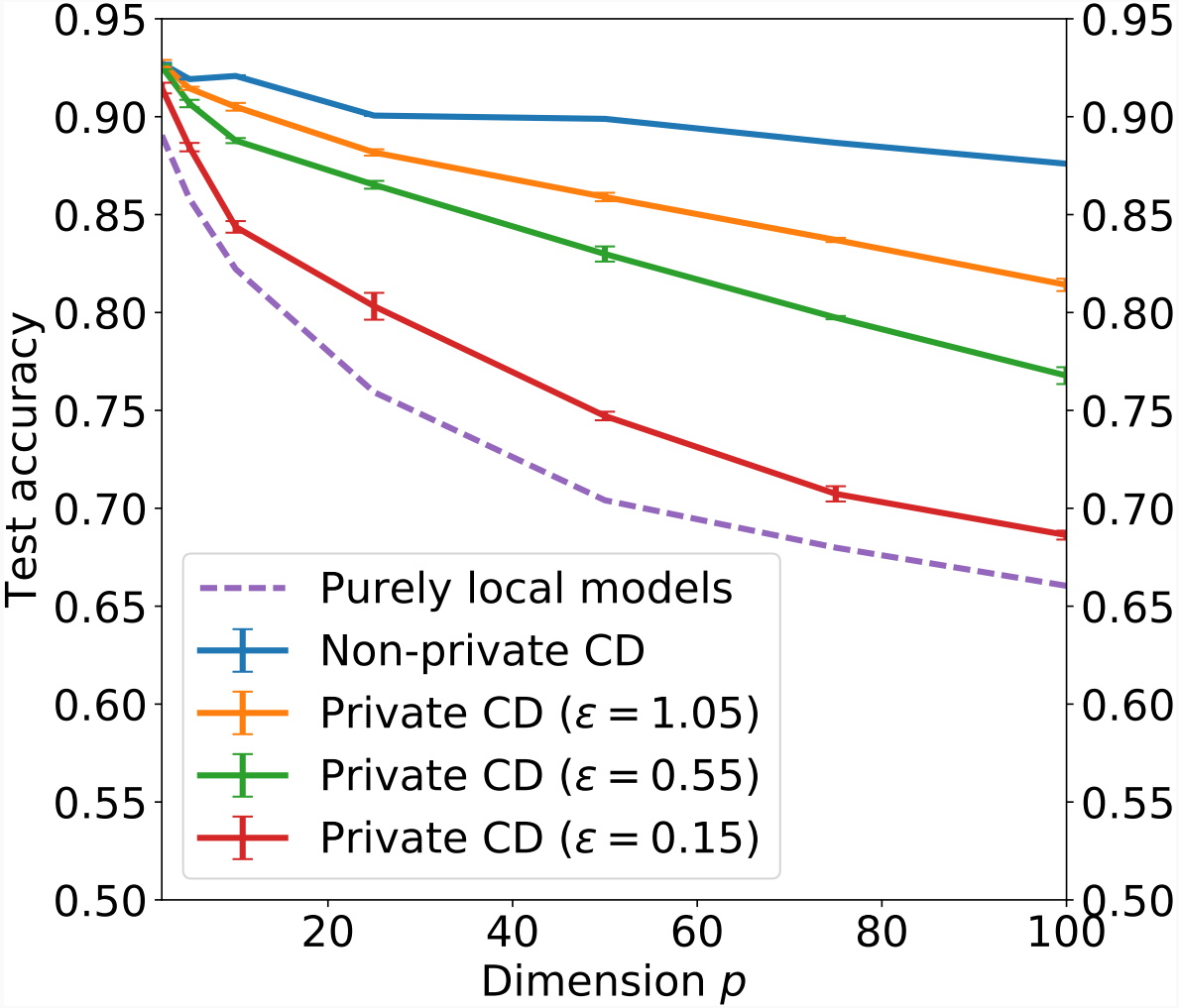
- Self-organizing overlays [Jelasity et al., 2009] are decentralized protocols to autonomously organize a large number of nodes into a predefined topology (cf François's talk)
- Usually one fixes a metric to compute similarity between nodes, and the desired topology (ring, k-NN graph...)
- Our protocol is a sort of self-organizing overlay:
  - The metric is based on (among other things) Euclidean distances between models, although it does not have a "closed form"
  - It can naturally recover from changes in the network
- But topology defined as solution to optimization problem: hence we automatically adapt to the problem

EXPERIMENTS: COLLABORATIVE LINEAR CLASSIFICATION

- We consider a set of $n = 100$ agents and a synthetic linear classification task in $\mathbb{R}^p$ (we use the hinge loss)
- Each agent is associated with an (unknown) target linear model
- Each agent i receives a random number m_i of samples with label given by the prediction of target model (plus noise)
- We can build a "ground-truth" collaboration graph based on the angle between target models (note: this is cheating!)

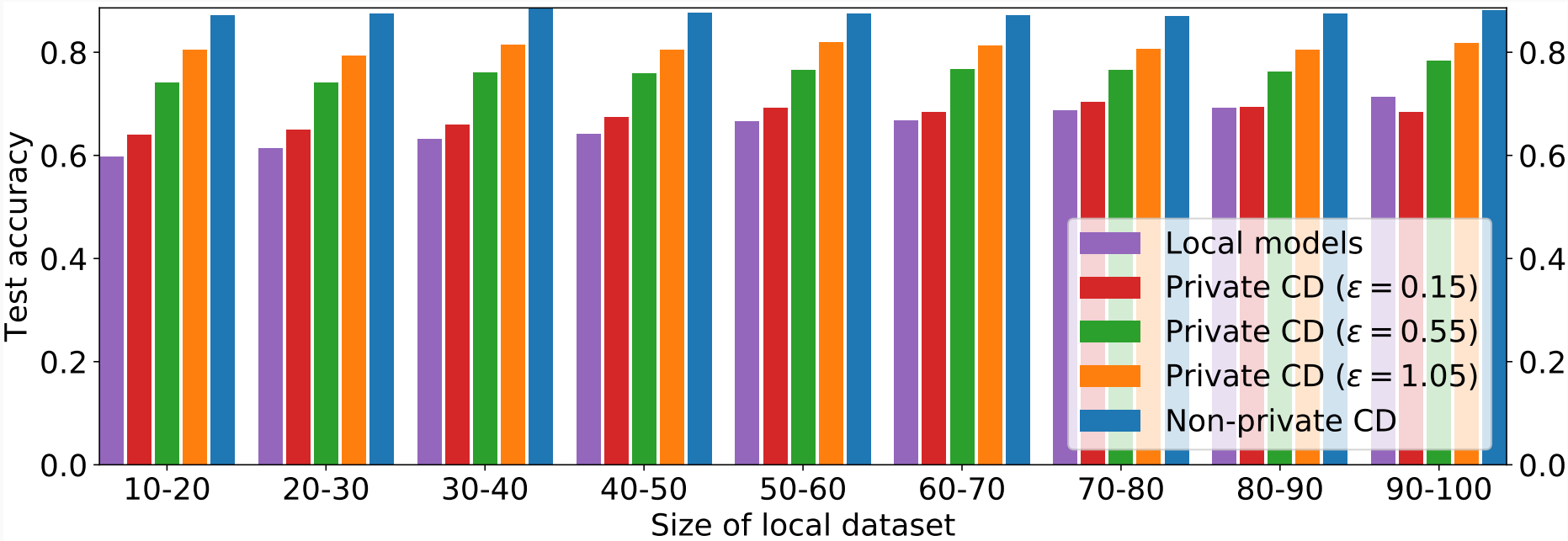
EXPERIMENTS: COLLABORATIVE LINEAR CLASSIFICATION

- Results when using the ground-truth graph



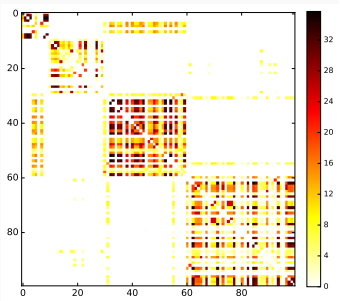
EXPERIMENTS: COLLABORATIVE LINEAR CLASSIFICATION

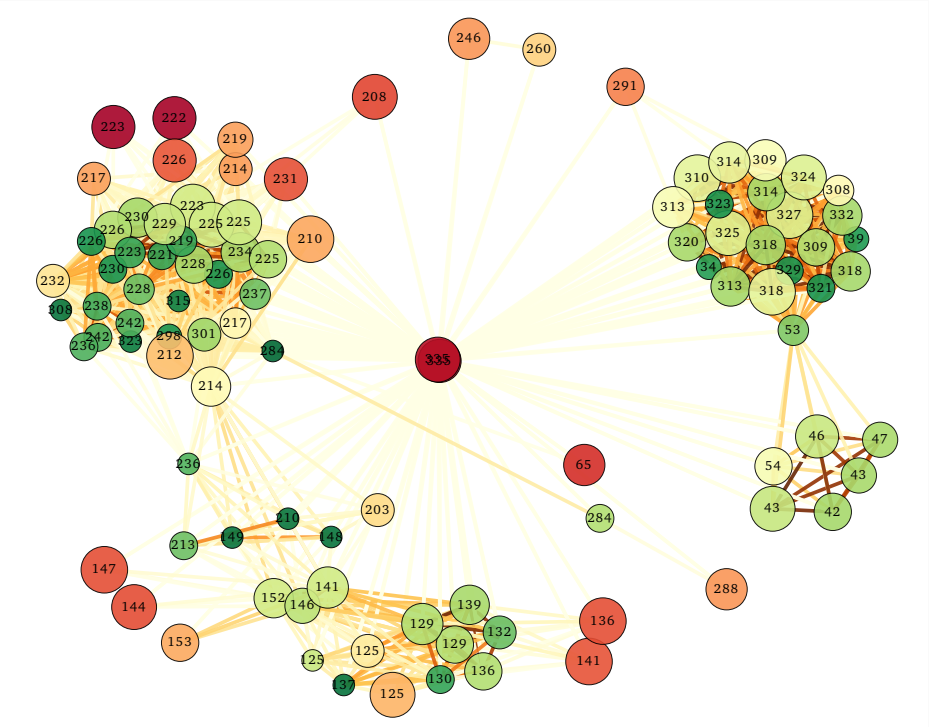
- All agents benefit even in private setting
- Agents with small local datasets get a stronger boost



EXPERIMENTS: COLLABORATIVE LINEAR CLASSIFICATION

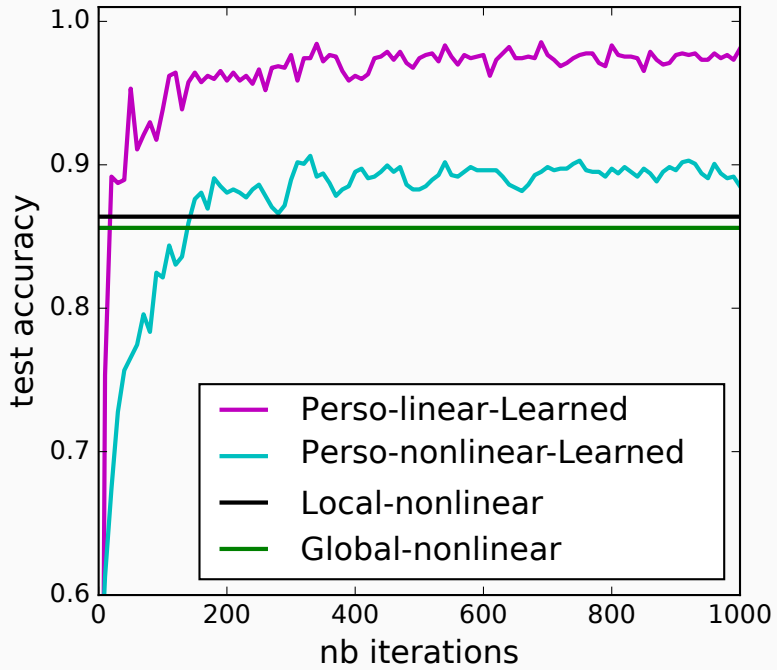
- We show that the learned topology adapts to the problem, unlike classic heuristics (e.g., k-NN graph)
- For instance, we can approximately recover a cluster structure over agents' tasks
- Prediction accuracy with learned graph is close to that obtained with the ground-truth graph





EXPERIMENTS: ACTIVITY RECOGNITION ON SMARTPHONES

- Use a public dataset with $n = 30$ agents
- Simple classification problem: walking upstairs vs downstairs
- Learn linear and nonlinear ensembles [Zantedeschi et al., 2019]
- 3-12 training points per agent, 561 features derived from sensors
- No agent similarity information available



Some future work

- Derive statistical generalization guarantees
- Reduce noise needed for DP: secure aggregation of models, privacy amplification schemes
- Dynamic setting: data arrive sequentially, agents may join/leave

Plenty of cool ML problems for distributed folks to work on!

- More realistic assumptions: message collision/drop, concurrency
- Robustness to malicious / byzantine behavior
- Secure, robust distributed systems to run private ML algorithms

QUICK AD: APVP 2019

- 10th Atelier sur la Protection de la Vie Privée
- Co-organizing with P. Bourhis, W. Rudametkin, M. Tommasi
- Multidisciplinary workshop on privacy (mostly in French)
- **July 9-11** (location to be confirmed)
- CFP to present your work (can already be published)
- Announcements soon!

Thank you for your attention!

Questions?

REFERENCES I

[Bellet et al., 2018] Bellet, A., Guerraoui, R., Taziki, M., and Tommasi, M. (2018).
Personalized and Private Peer-to-Peer Machine Learning.
In *AISTATS*.

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

[Jelasity et al., 2009] Jelasity, M., Montresor, A., and Babaoglu, Ö. (2009).
T-Man: Gossip-based fast overlay topology construction.
Computer Networks, 53(13):2321–2339.

[Jelasity et al., 2007] Jelasity, M., Voulgaris, S., Guerraoui, R., Kermarrec, A.-M., and van Steen, M. (2007).
Gossip-based peer sampling.
ACM Trans. Comput. Syst., 25(3).

[Lian et al., 2017] 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*.

References II

[McMahan et al., 2017] McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and Agüera y Arcas, B. (2017).
Communication-efficient learning of deep networks from decentralized data.
In *AISTATS*.

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

[Zantedeschi et al., 2019] Zantedeschi, V., Bellet, A., and Tommasi, M. (2019).
Communication-efficient and decentralized multi-task boosting while learning the collaboration graph.
Technical report, arXiv:1901.08460.