PRIVACY-PRESERVING ALGORITHMS FOR DECENTRALIZED COLLABORATIVE MACHINE LEARNING

Aurélien Bellet (Inria MAGNET)

Joint work with R. Guerraoui, M. Taziki, M. Tommasi and P. Vanhaesebrouck

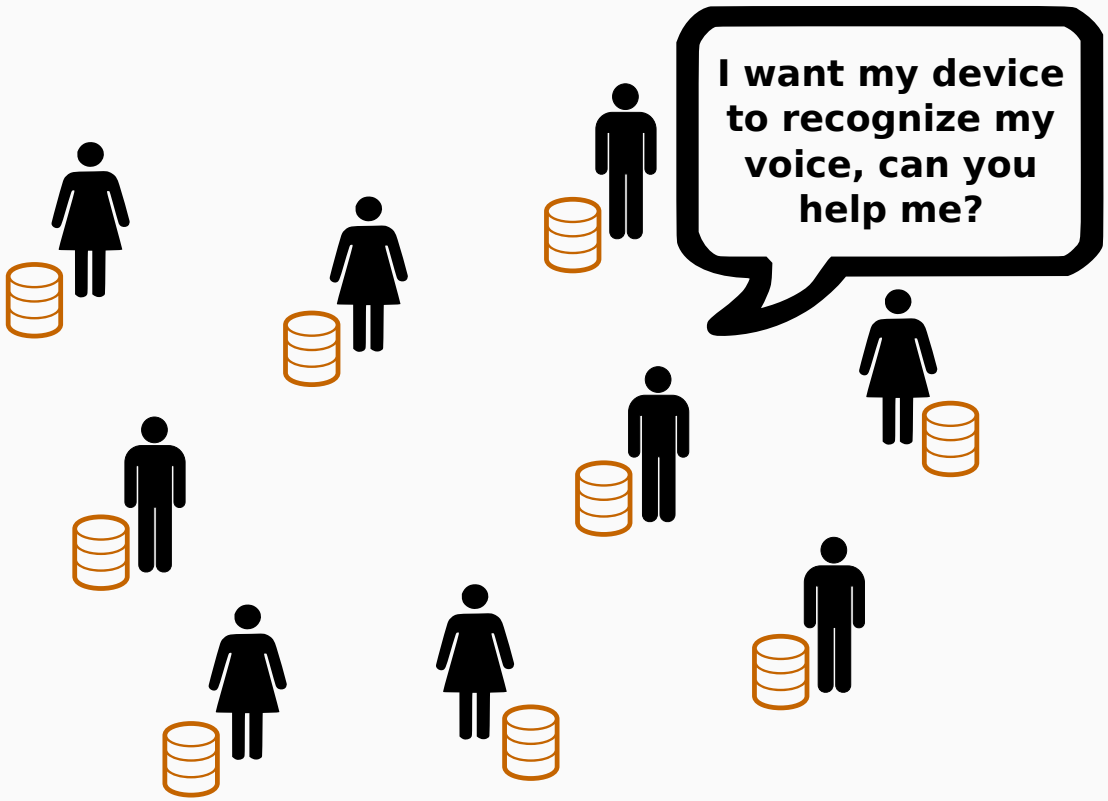
Seminar at the Alan Turing Institute
February 26, 2018

OUTLINE

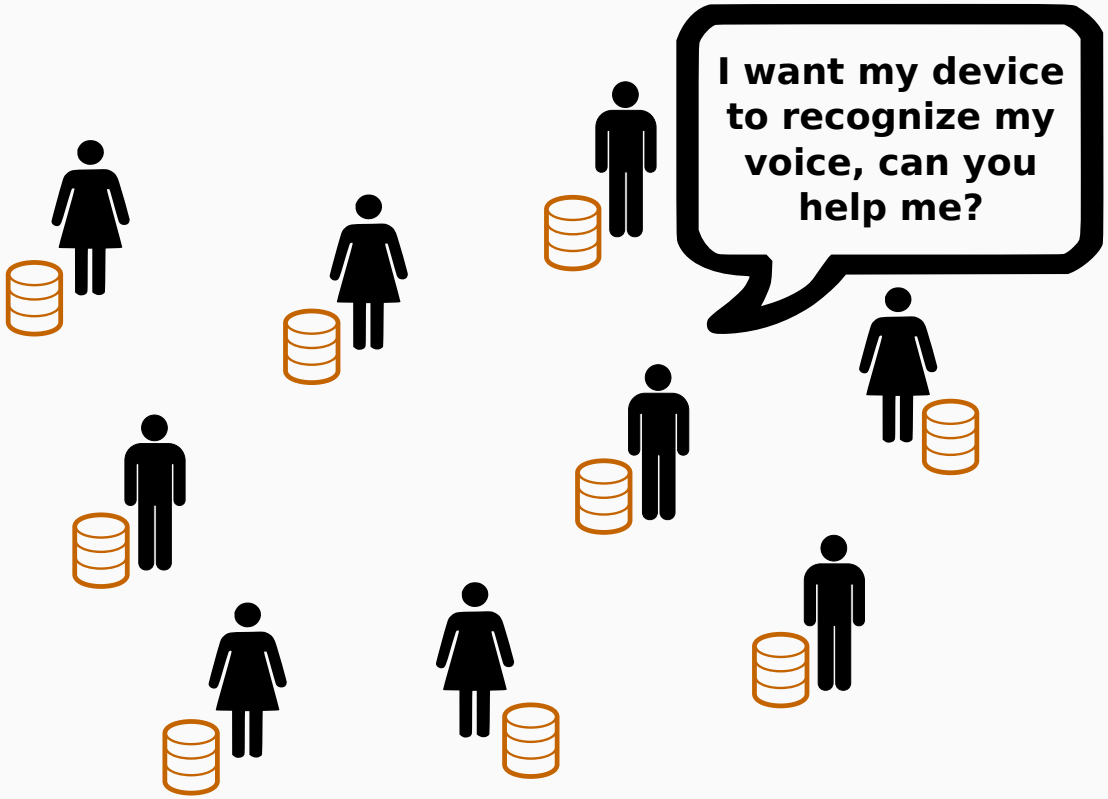
1. Broad context
2. Problem setting
3. Decentralized algorithms
4. Privacy in decentralized learning
5. Illustrative experiments
6. Future work

BROAD CONTEXT

LEARNING FROM CONNECTED DEVICES DATA

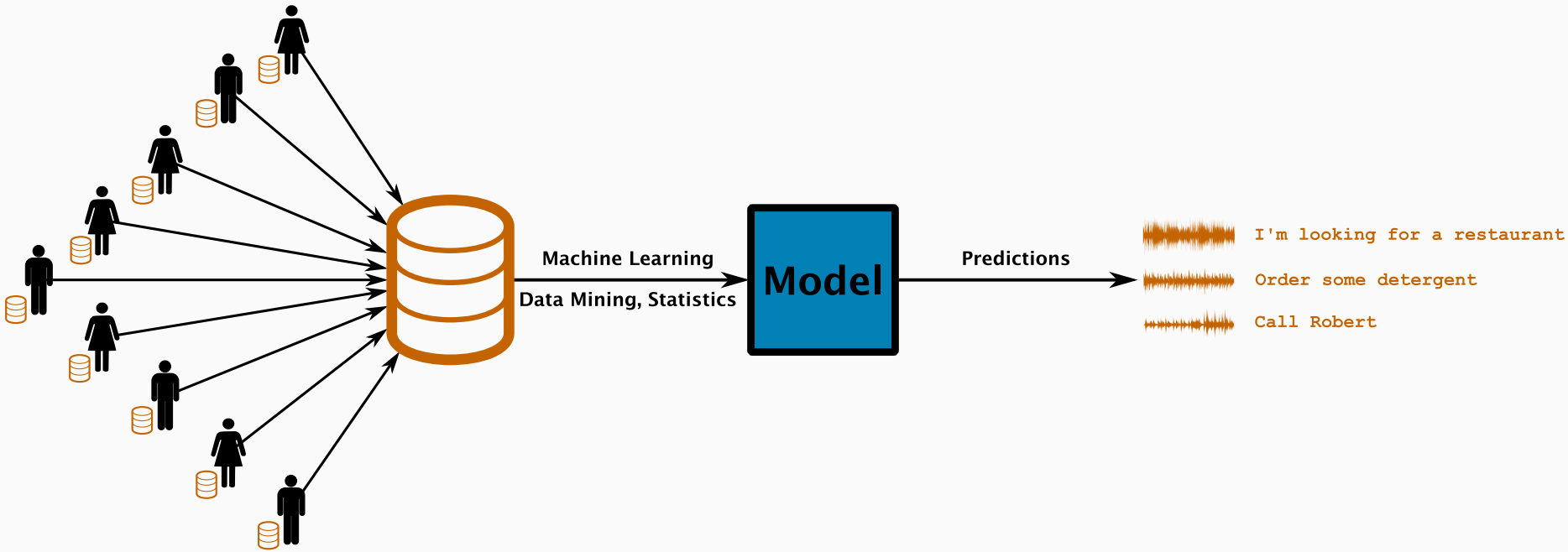


LEARNING FROM CONNECTED DEVICES DATA



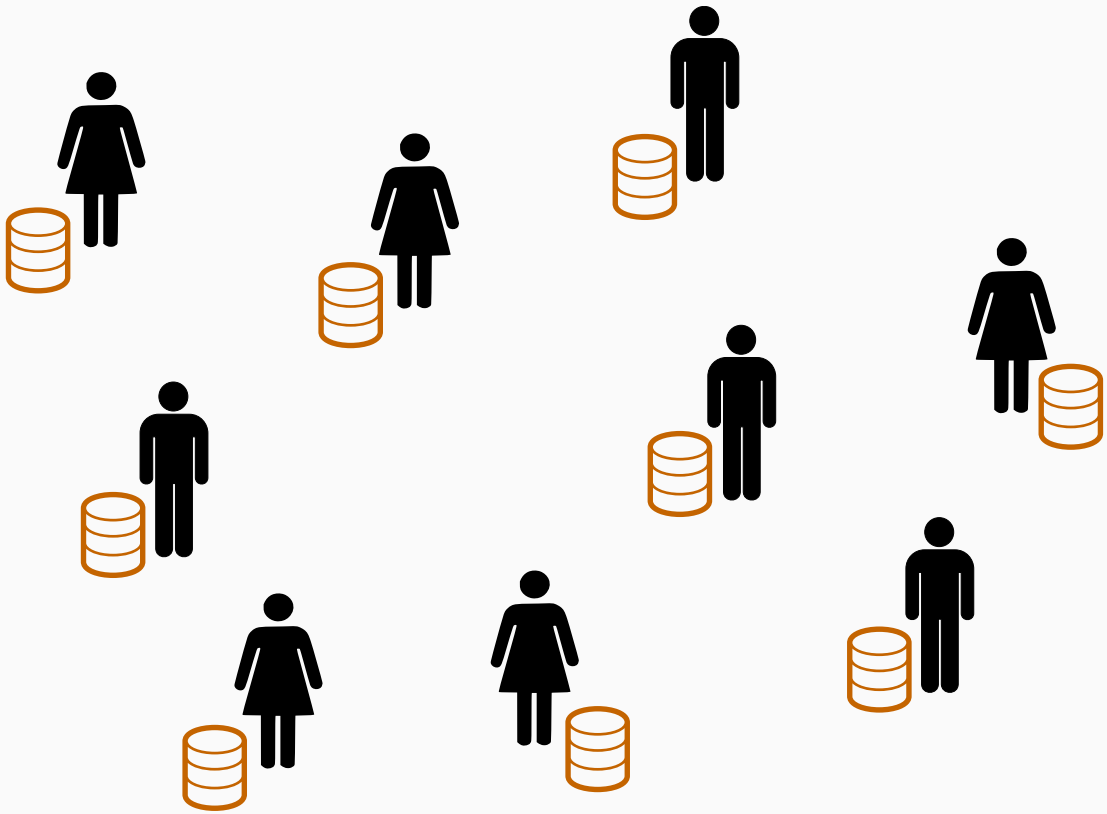
- Other examples of applications
  - Recommend content based on user activity logs
  - Learn personalized treatment from wearable device data

EXTREME APPROACH 1: CENTRALIZED LEARNING



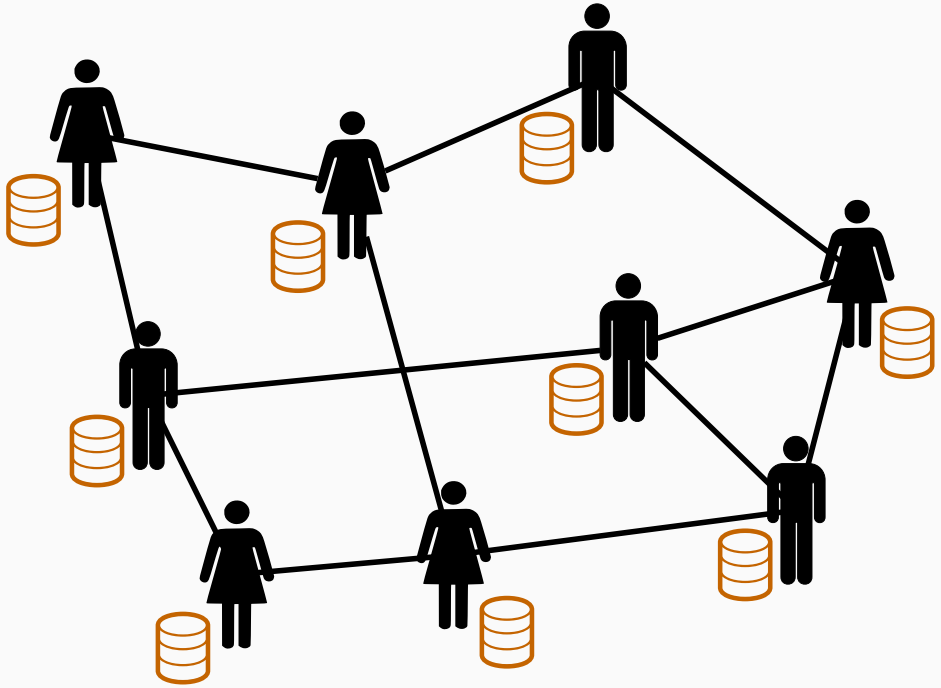
- Best for utility: efficient access and processing
- Lack of user control over its personal data
  - What is collected? Who can access it? How is it used and what for?
- Vulnerability to attacks / subpoenas
  - Yahoo data breach (3B users!), Twitter / Wikileaks court orders
- Costly infrastructure for service provider

EXTREME APPROACH 2: PURELY LOCAL LEARNING



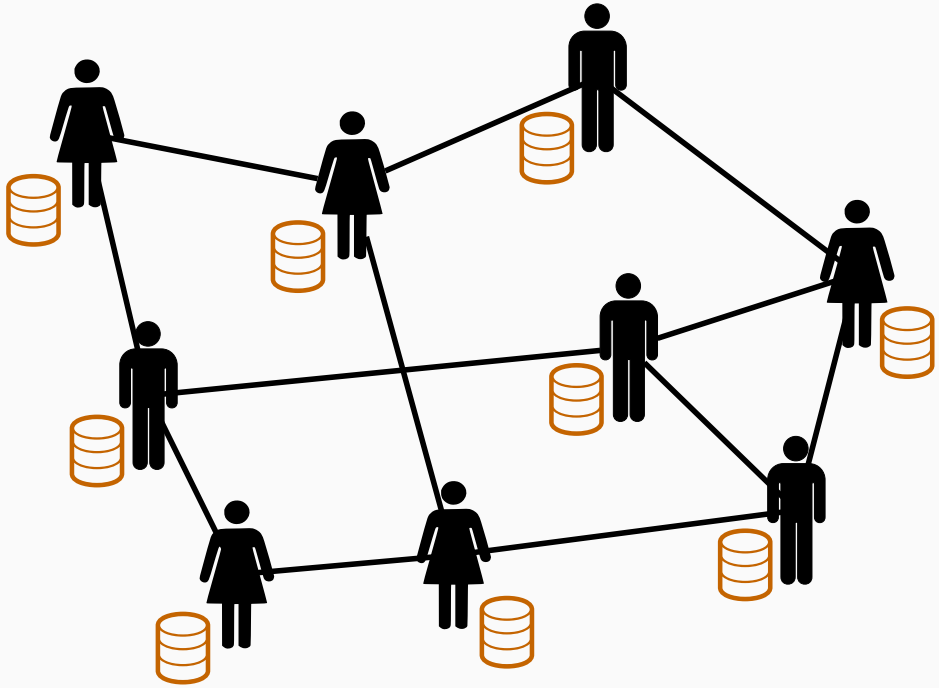
- Best for privacy: no information exchanged between devices
- Bad for utility (especially for users without much data)

OUR APPROACH: FULLY DECENTRALIZED LEARNING



- Personal data stays on user's device
- Peer-to-peer and asynchronous communications
- No single point of failure/entry as in server-client architectures
- Scalability-by-design to many devices through local updates (see e.g., [Lian et al., 2017])

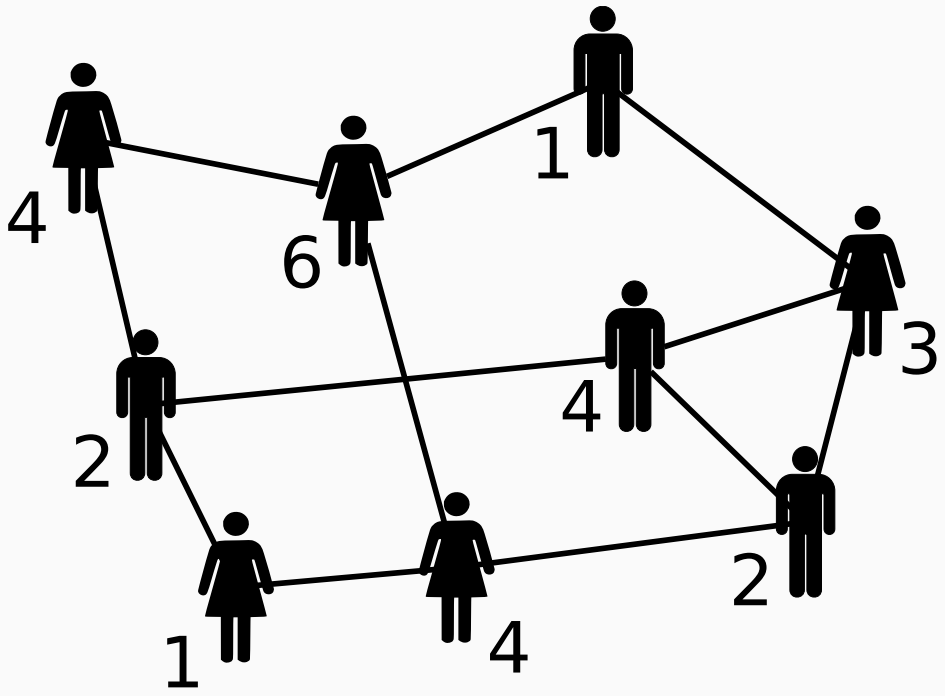
OUR APPROACH: FULLY DECENTRALIZED LEARNING



Some scientific challenges

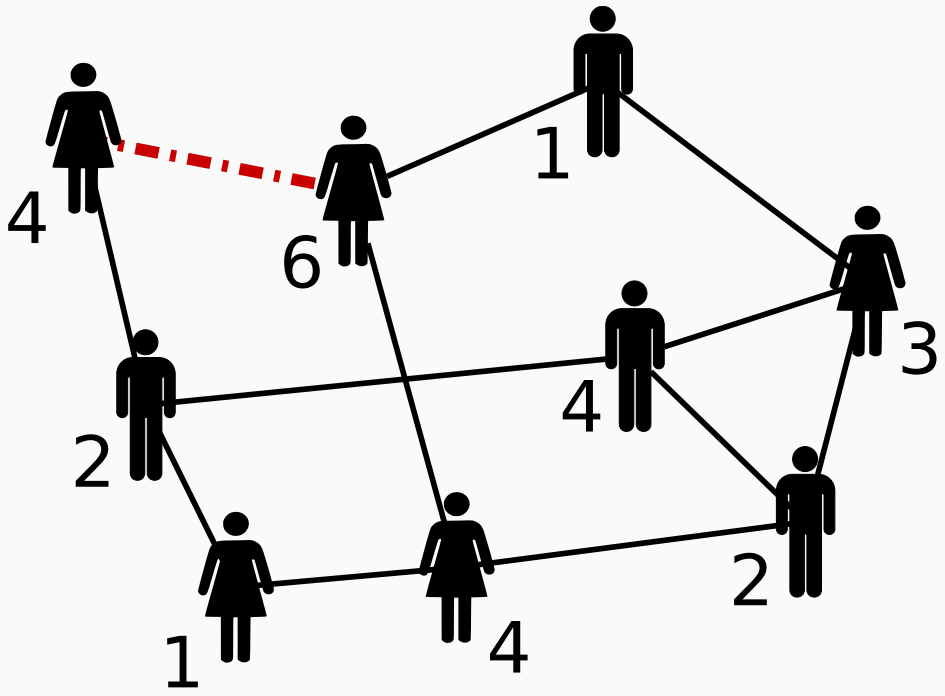
1. How to efficiently learn in a decentralized way under these communication constraints?
2. How to prevent malicious users from inferring sensitive data or manipulating the outcome to their advantage?

KEY PRINCIPLE: GOSSIP ALGORITHM



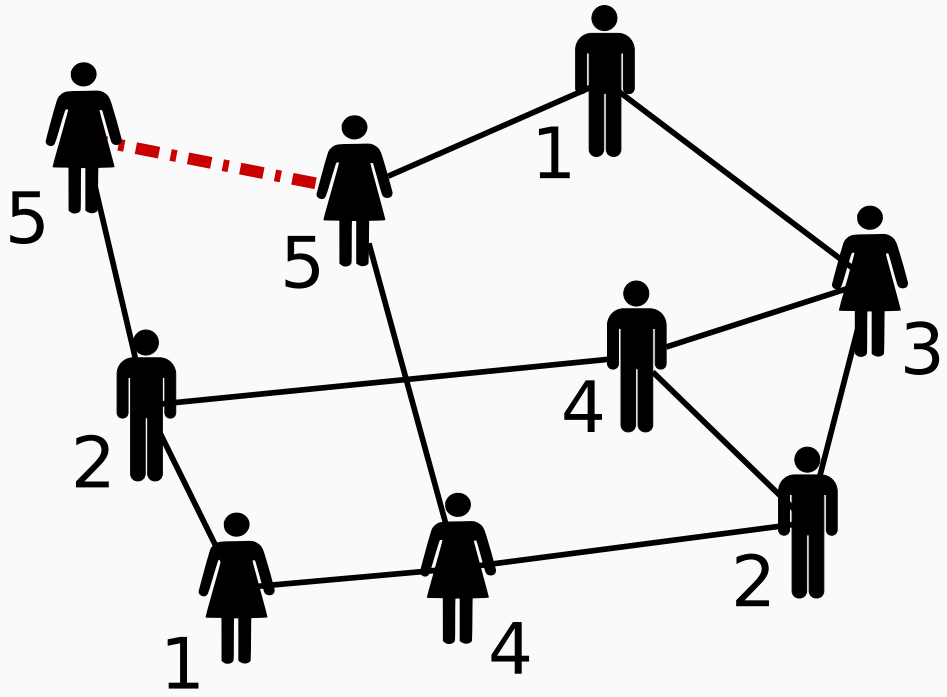
- Users wake up independently and in parallel, select a random neighbor and exchange information
  - Equivalent view: an iteration is a random edge activation
- Simple and asynchronous $\rightarrow$ well suited to large networks

KEY PRINCIPLE: GOSSIP ALGORITHM



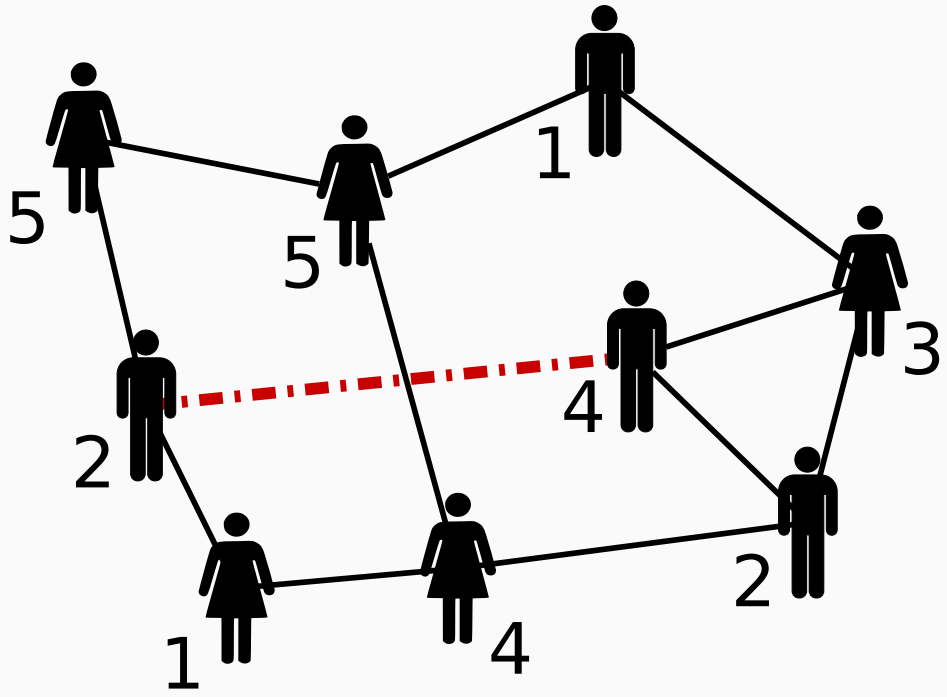
- Users wake up independently and in parallel, select a random neighbor and exchange information
  - Equivalent view: an iteration is a random edge activation
- Simple and asynchronous $\rightarrow$ well suited to large networks

KEY PRINCIPLE: GOSSIP ALGORITHM



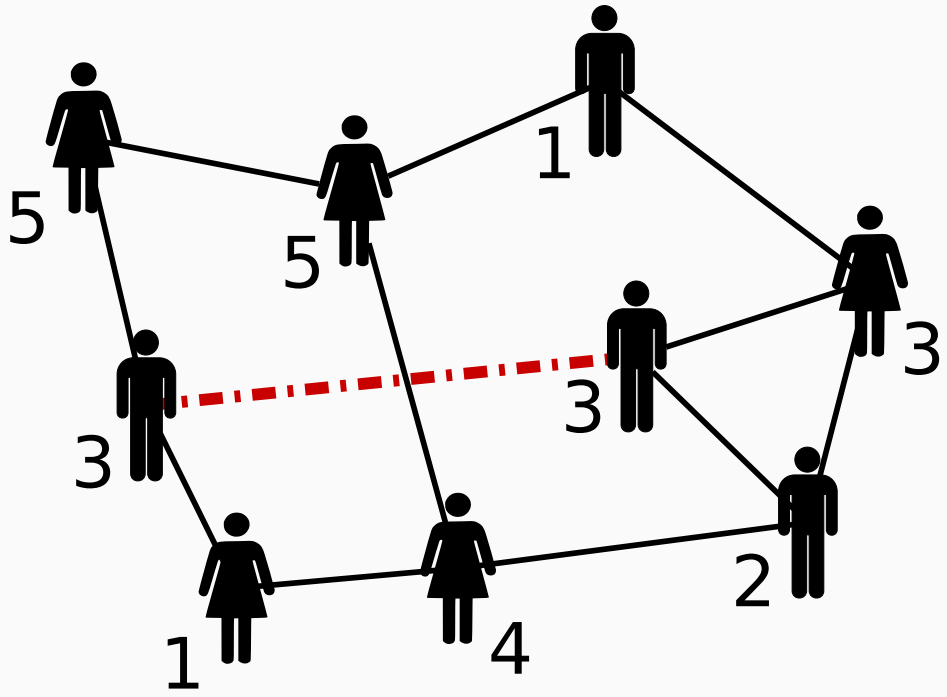
- Users wake up independently and in parallel, select a random neighbor and exchange information
  - Equivalent view: an iteration is a random edge activation
- Simple and asynchronous $\rightarrow$ well suited to large networks

KEY PRINCIPLE: GOSSIP ALGORITHM



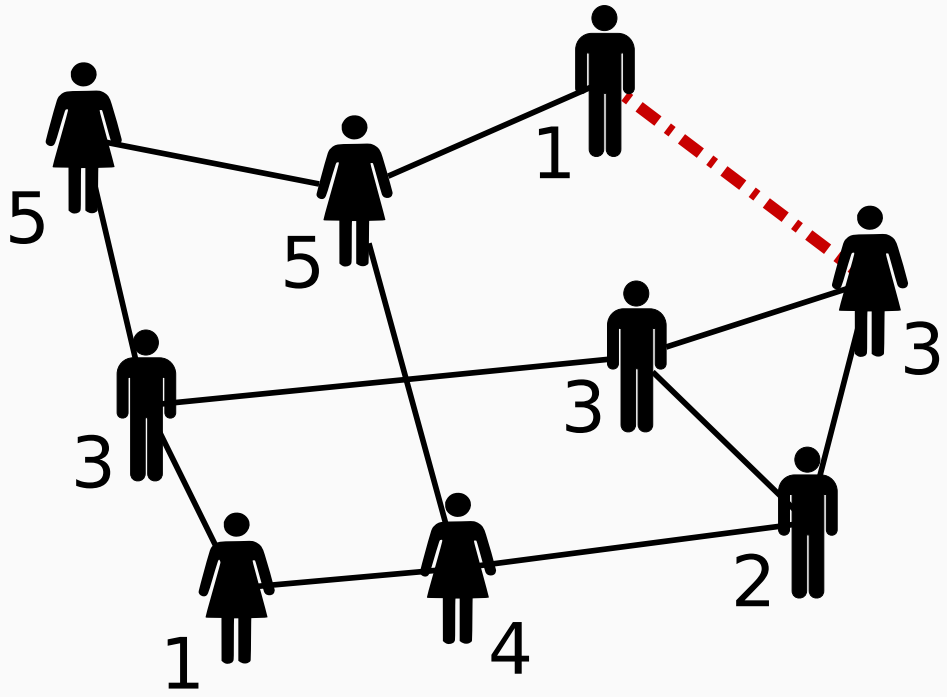
- Users wake up independently and in parallel, select a random neighbor and exchange information
  - Equivalent view: an iteration is a random edge activation
- Simple and asynchronous $\rightarrow$ well suited to large networks

KEY PRINCIPLE: GOSSIP ALGORITHM



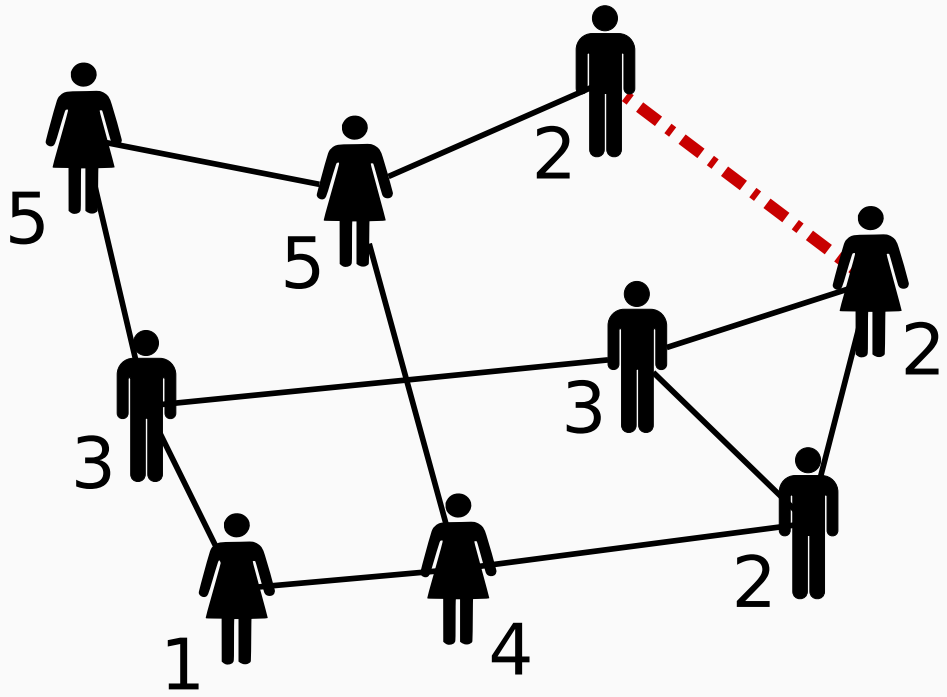
- Users wake up independently and in parallel, select a random neighbor and exchange information
  - Equivalent view: an iteration is a random edge activation
- Simple and asynchronous $\rightarrow$ well suited to large networks

KEY PRINCIPLE: GOSSIP ALGORITHM



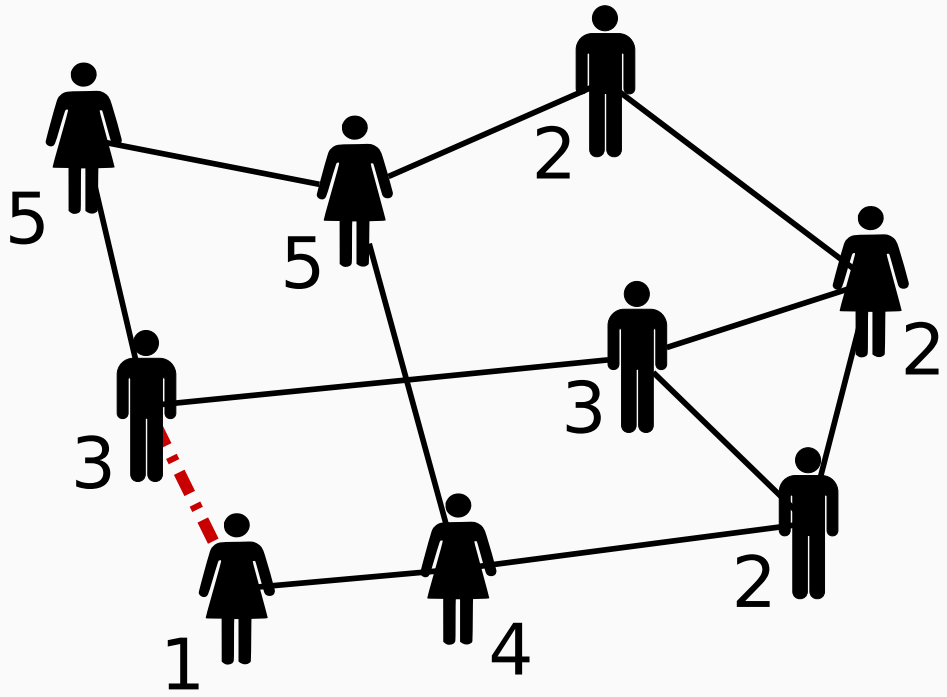
- Users wake up independently and in parallel, select a random neighbor and exchange information
  - Equivalent view: an iteration is a random edge activation
- Simple and asynchronous $\rightarrow$ well suited to large networks

KEY PRINCIPLE: GOSSIP ALGORITHM



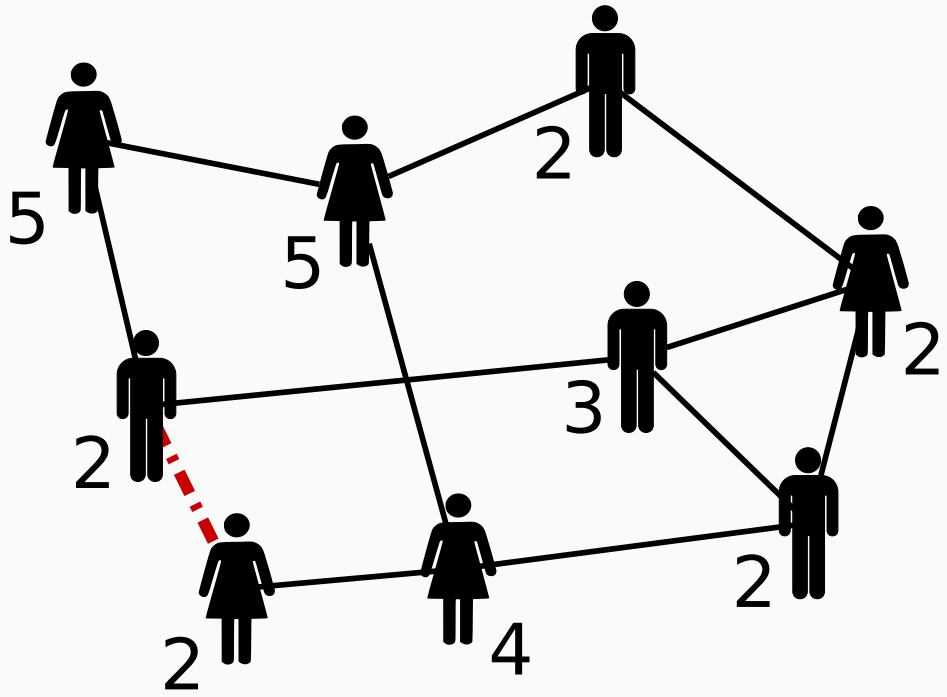
- Users wake up independently and in parallel, select a random neighbor and exchange information
  - Equivalent view: an iteration is a random edge activation
- Simple and asynchronous $\rightarrow$ well suited to large networks

KEY PRINCIPLE: GOSSIP ALGORITHM



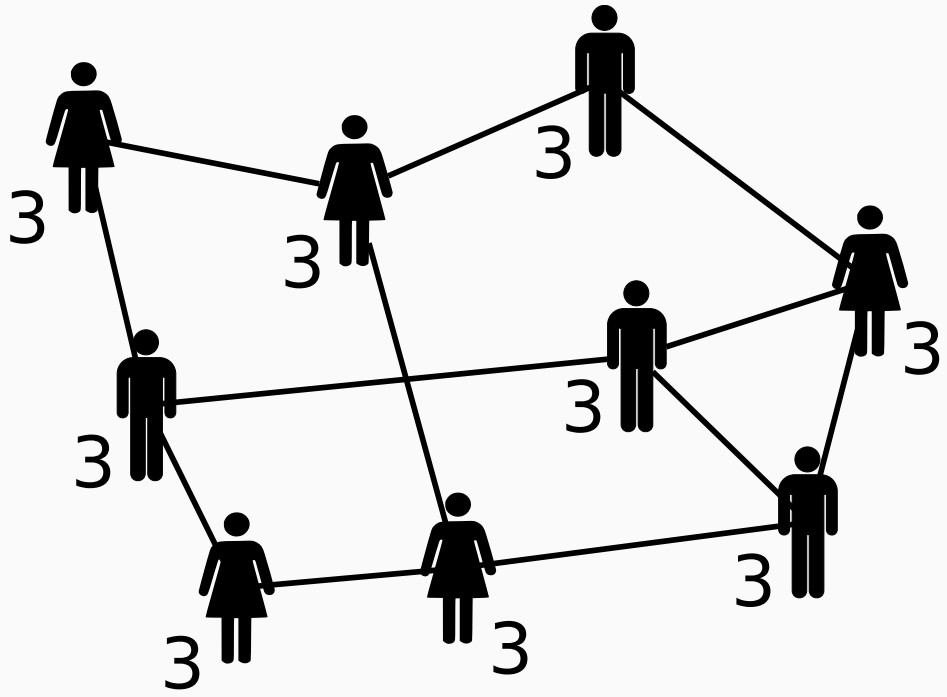
- Users wake up independently and in parallel, select a random neighbor and exchange information
  - Equivalent view: an iteration is a random edge activation
- Simple and asynchronous $\rightarrow$ well suited to large networks

KEY PRINCIPLE: GOSSIP ALGORITHM



- Users wake up independently and in parallel, select a random neighbor and exchange information
  - Equivalent view: an iteration is a random edge activation
- Simple and asynchronous $\rightarrow$ well suited to large networks

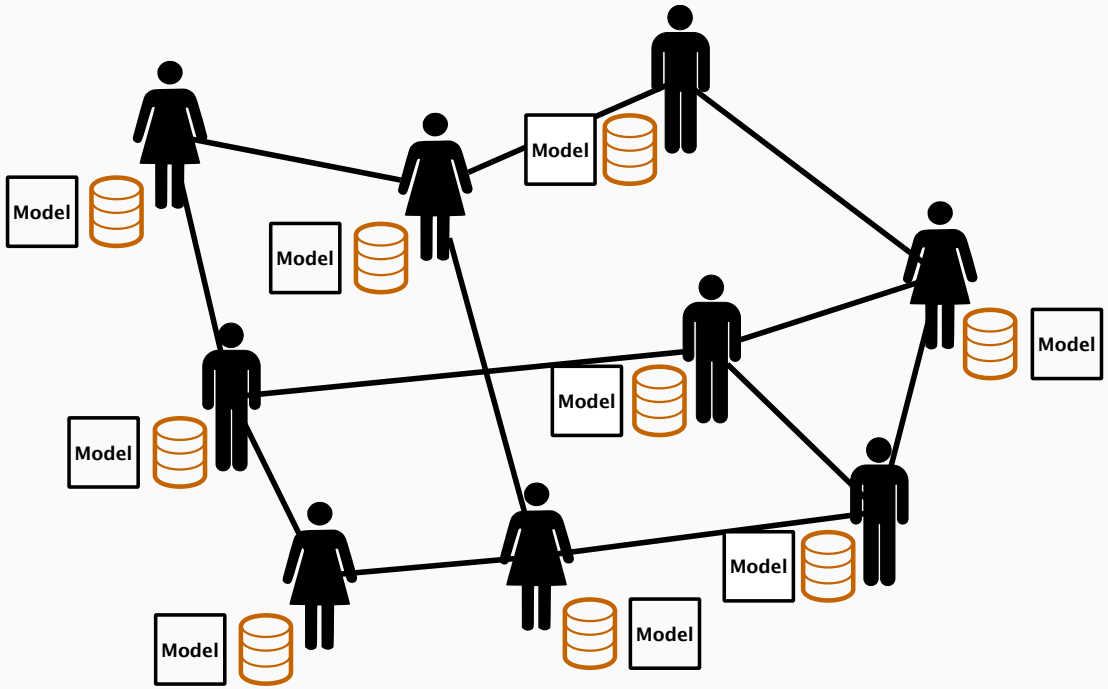
KEY PRINCIPLE: GOSSIP ALGORITHM



- Users wake up independently and in parallel, select a random neighbor and exchange information
  - Equivalent view: an iteration is a random edge activation
- Simple and asynchronous $\rightarrow$ well suited to large networks

EXISTING WORK: CONSENSUS LEARNING

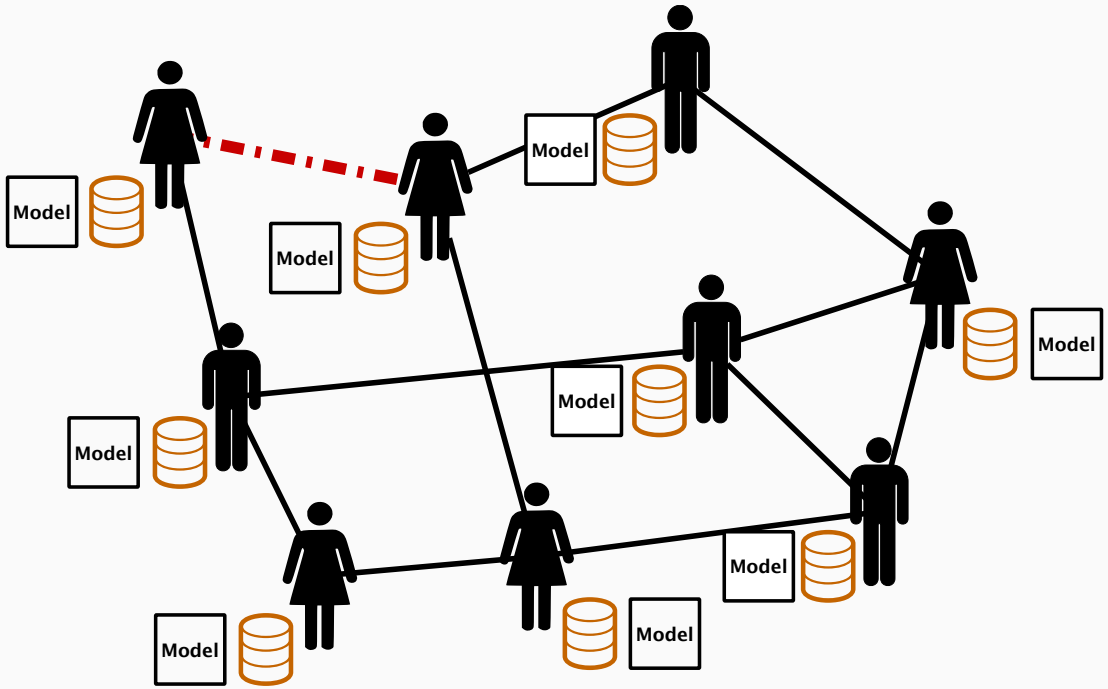
- Gossip algorithms to optimize a loss function over the union of personal datasets [Nedic and Ozdaglar, 2009, Duchi et al., 2012, Wei and Ozdaglar, 2012, Colin et al., 2016, Scaman et al., 2017]



- General idea: at each step
  1. perform a local model update based on personal data
  2. average with neighbor

EXISTING WORK: CONSENSUS LEARNING

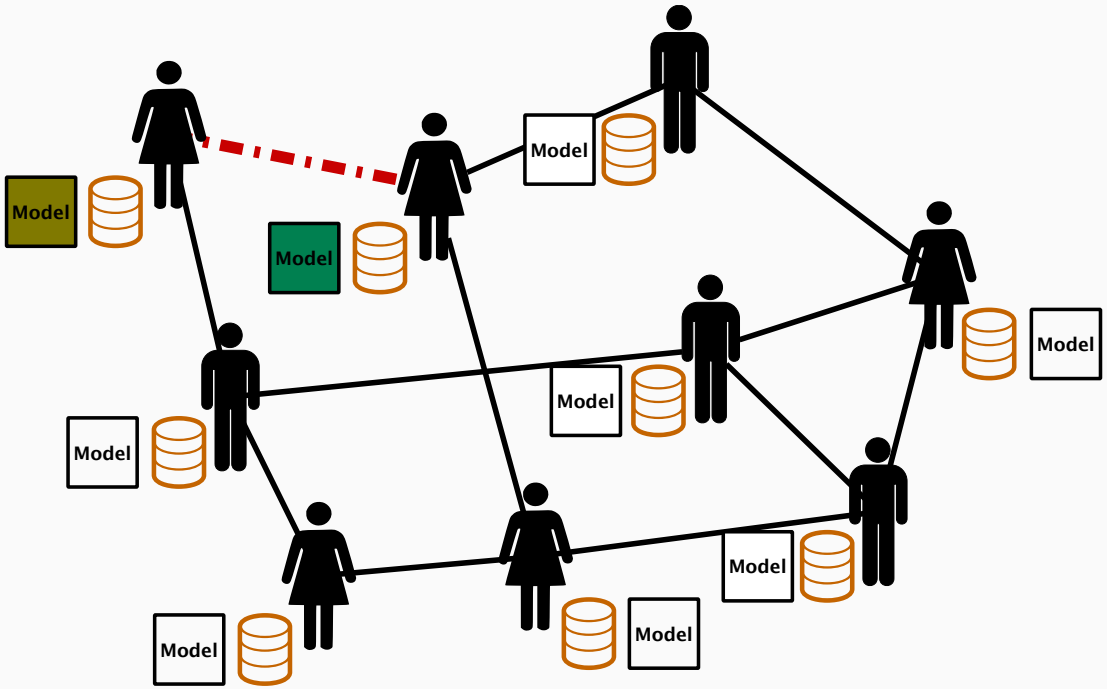
- Gossip algorithms to optimize a loss function over the union of personal datasets [Nedic and Ozdaglar, 2009, Duchi et al., 2012, Wei and Ozdaglar, 2012, Colin et al., 2016, Scaman et al., 2017]



- General idea: at each step
  1. perform a local model update based on personal data
  2. average with neighbor

EXISTING WORK: CONSENSUS LEARNING

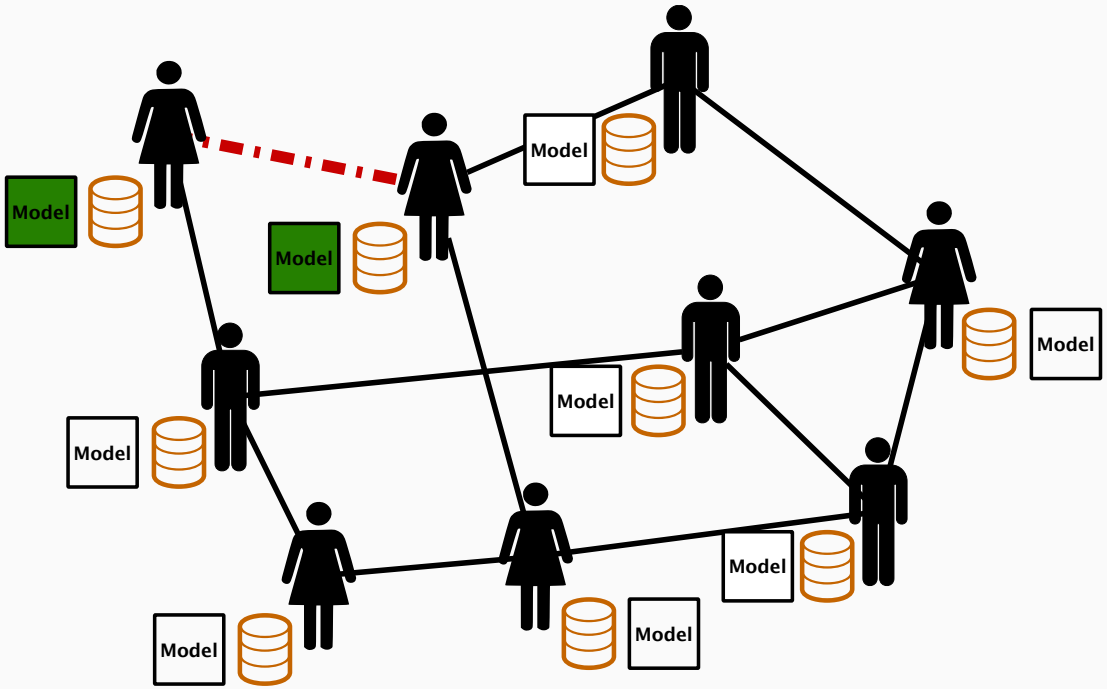
- Gossip algorithms to optimize a loss function over the union of personal datasets [Nedic and Ozdaglar, 2009, Duchi et al., 2012, Wei and Ozdaglar, 2012, Colin et al., 2016, Scaman et al., 2017]



- General idea: at each step
  1. perform a local model update based on personal data
  2. average with neighbor

EXISTING WORK: CONSENSUS LEARNING

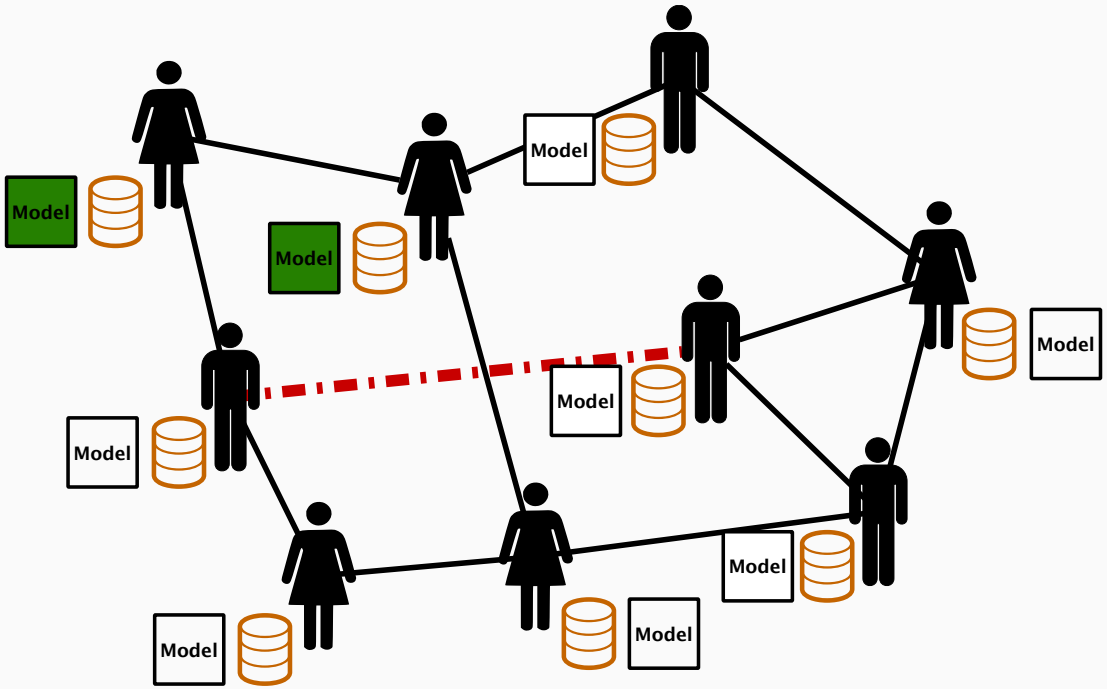
- Gossip algorithms to optimize a loss function over the union of personal datasets [Nedic and Ozdaglar, 2009, Duchi et al., 2012, Wei and Ozdaglar, 2012, Colin et al., 2016, Scaman et al., 2017]



- General idea: at each step
  1. perform a local model update based on personal data
  2. average with neighbor

EXISTING WORK: CONSENSUS LEARNING

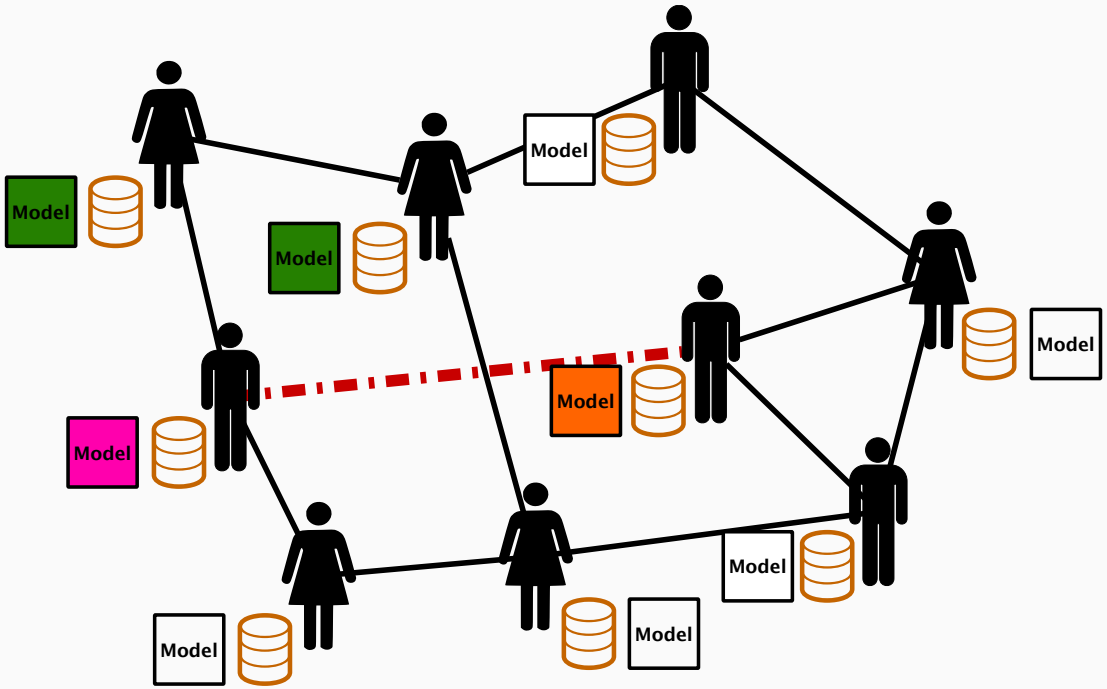
- Gossip algorithms to optimize a loss function over the union of personal datasets [Nedic and Ozdaglar, 2009, Duchi et al., 2012, Wei and Ozdaglar, 2012, Colin et al., 2016, Scaman et al., 2017]



- General idea: at each step
  1. perform a local model update based on personal data
  2. average with neighbor

EXISTING WORK: CONSENSUS LEARNING

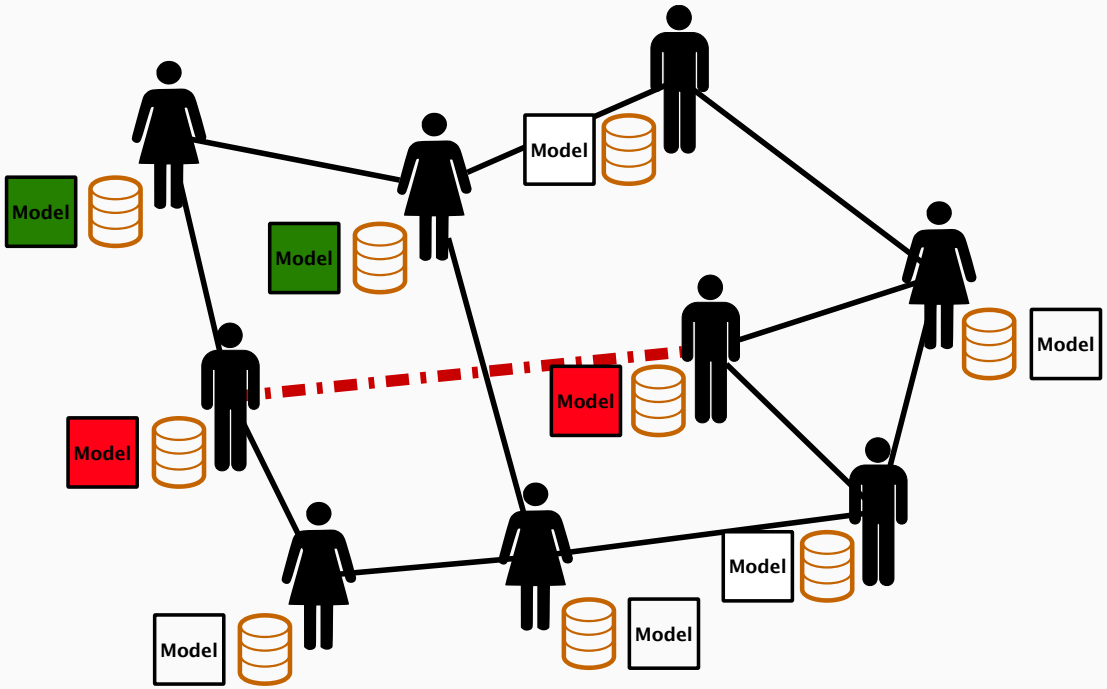
- Gossip algorithms to optimize a loss function over the union of personal datasets [Nedic and Ozdaglar, 2009, Duchi et al., 2012, Wei and Ozdaglar, 2012, Colin et al., 2016, Scaman et al., 2017]



- General idea: at each step
  1. perform a local model update based on personal data
  2. average with neighbor

EXISTING WORK: CONSENSUS LEARNING

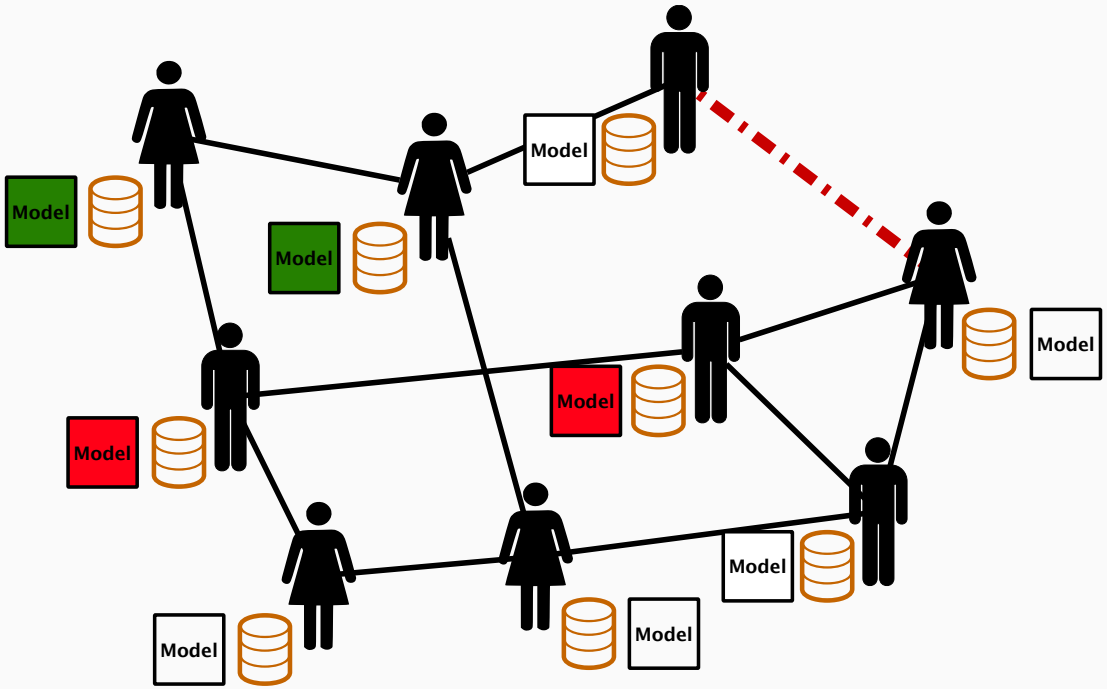
- Gossip algorithms to optimize a loss function over the union of personal datasets [Nedic and Ozdaglar, 2009, Duchi et al., 2012, Wei and Ozdaglar, 2012, Colin et al., 2016, Scaman et al., 2017]



- General idea: at each step
  1. perform a local model update based on personal data
  2. average with neighbor

EXISTING WORK: CONSENSUS LEARNING

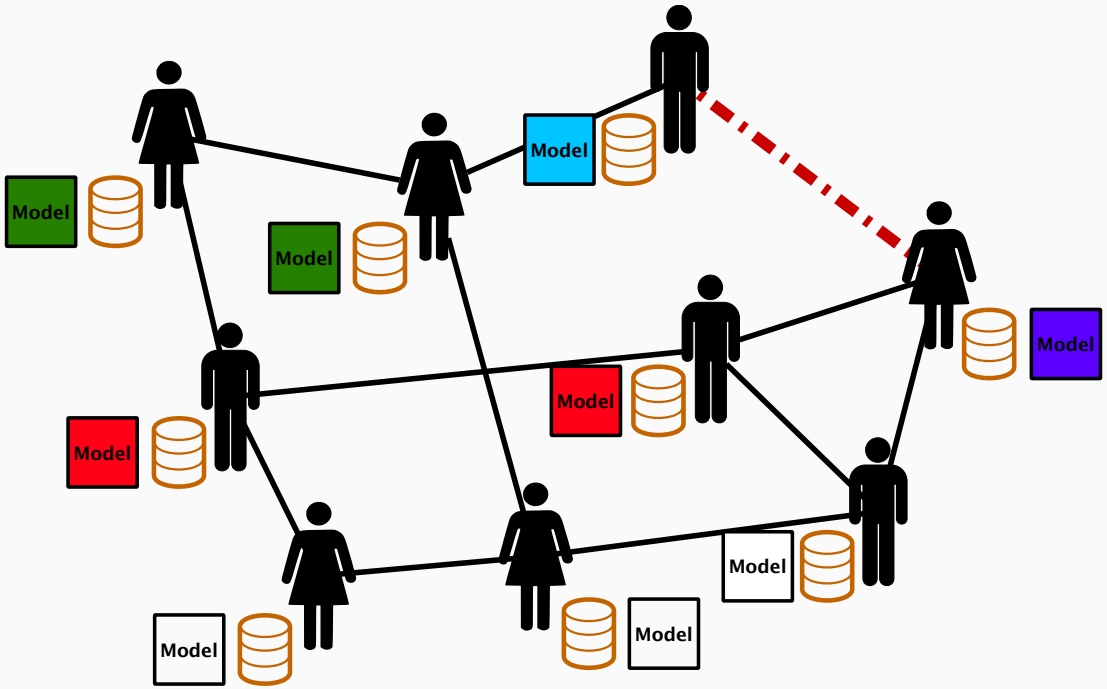
- Gossip algorithms to optimize a loss function over the union of personal datasets [Nedic and Ozdaglar, 2009, Duchi et al., 2012, Wei and Ozdaglar, 2012, Colin et al., 2016, Scaman et al., 2017]



- General idea: at each step
  1. perform a local model update based on personal data
  2. average with neighbor

EXISTING WORK: CONSENSUS LEARNING

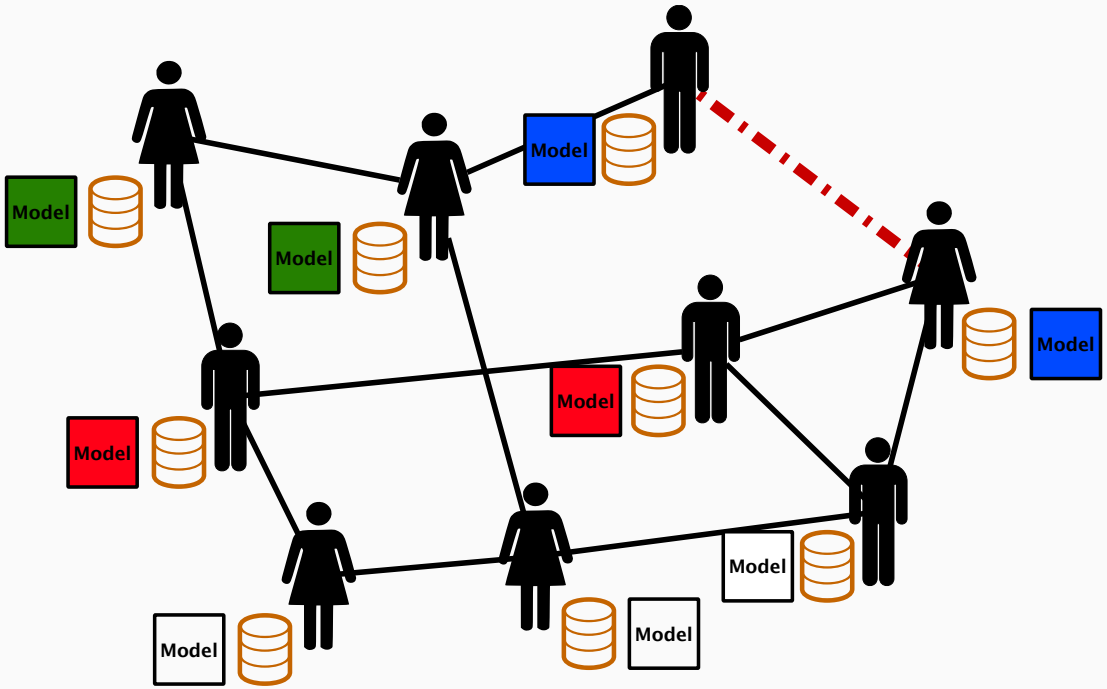
- Gossip algorithms to optimize a loss function over the union of personal datasets [Nedic and Ozdaglar, 2009, Duchi et al., 2012, Wei and Ozdaglar, 2012, Colin et al., 2016, Scaman et al., 2017]



- General idea: at each step
  1. perform a local model update based on personal data
  2. average with neighbor

EXISTING WORK: CONSENSUS LEARNING

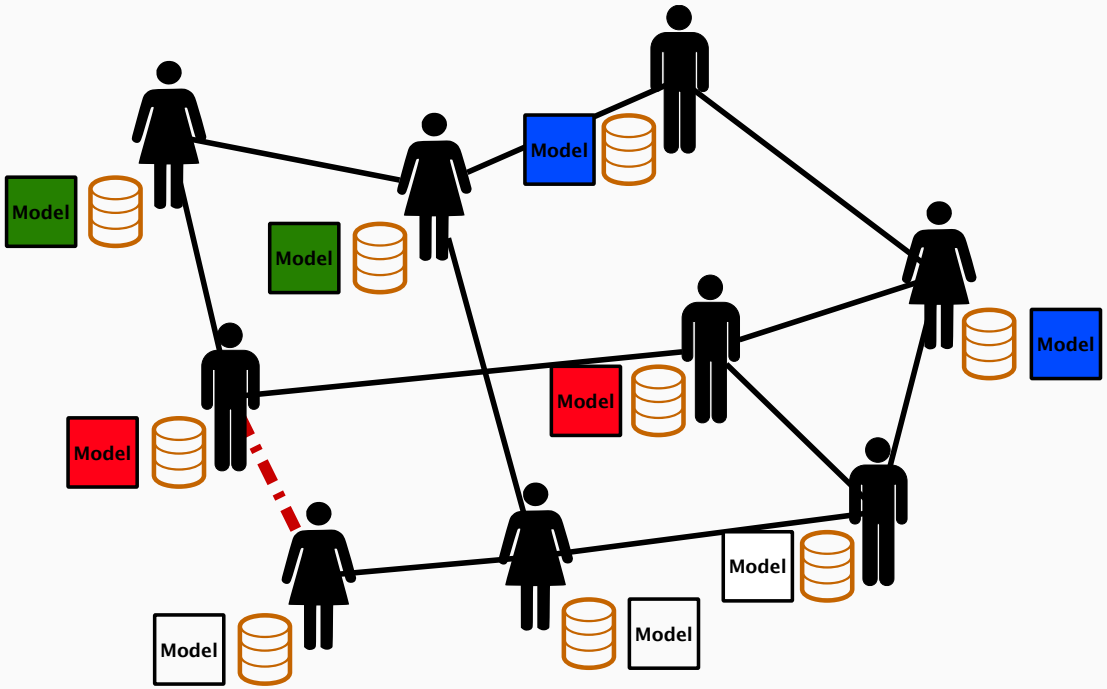
- Gossip algorithms to optimize a loss function over the union of personal datasets [Nedic and Ozdaglar, 2009, Duchi et al., 2012, Wei and Ozdaglar, 2012, Colin et al., 2016, Scaman et al., 2017]



- General idea: at each step
  1. perform a local model update based on personal data
  2. average with neighbor

EXISTING WORK: CONSENSUS LEARNING

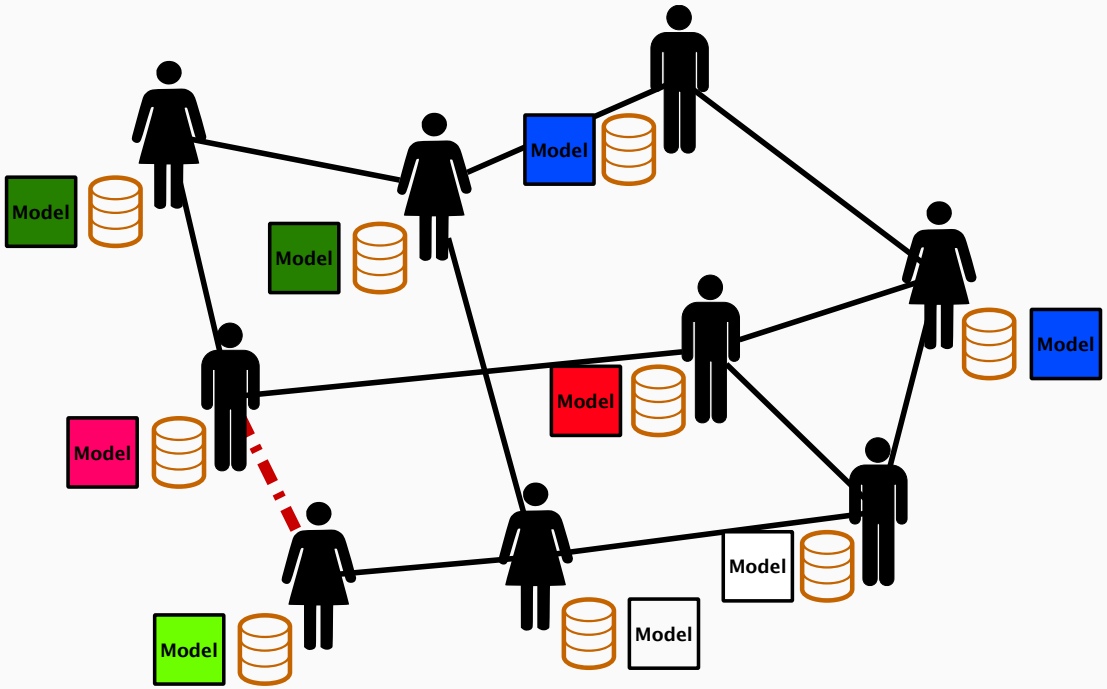
- Gossip algorithms to optimize a loss function over the union of personal datasets [Nedic and Ozdaglar, 2009, Duchi et al., 2012, Wei and Ozdaglar, 2012, Colin et al., 2016, Scaman et al., 2017]



- General idea: at each step
  1. perform a local model update based on personal data
  2. average with neighbor

EXISTING WORK: CONSENSUS LEARNING

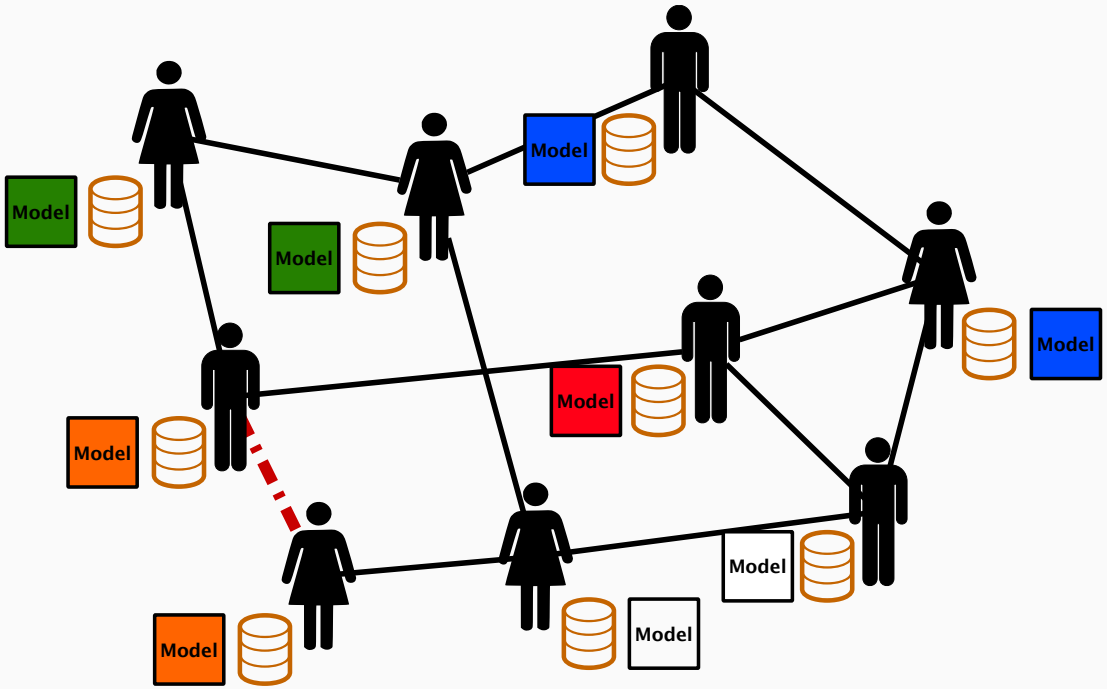
- Gossip algorithms to optimize a loss function over the union of personal datasets [Nedic and Ozdaglar, 2009, Duchi et al., 2012, Wei and Ozdaglar, 2012, Colin et al., 2016, Scaman et al., 2017]



- General idea: at each step
  1. perform a local model update based on personal data
  2. average with neighbor

EXISTING WORK: CONSENSUS LEARNING

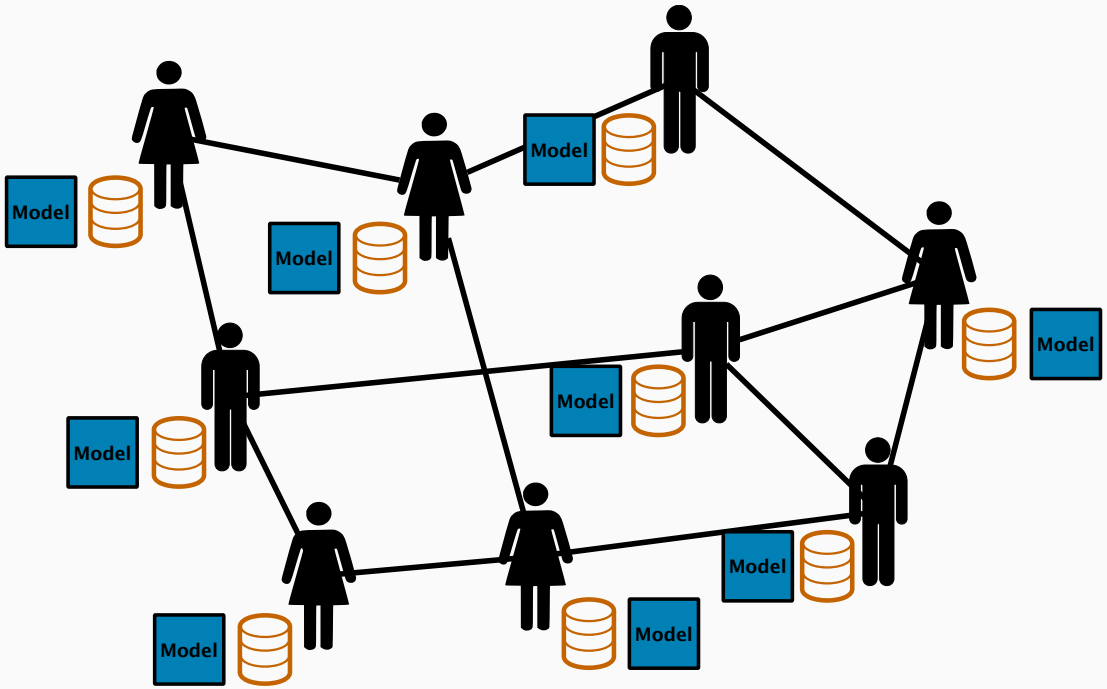
- Gossip algorithms to optimize a loss function over the union of personal datasets [Nedic and Ozdaglar, 2009, Duchi et al., 2012, Wei and Ozdaglar, 2012, Colin et al., 2016, Scaman et al., 2017]



- General idea: at each step
  1. perform a local model update based on personal data
  2. average with neighbor

EXISTING WORK: CONSENSUS LEARNING

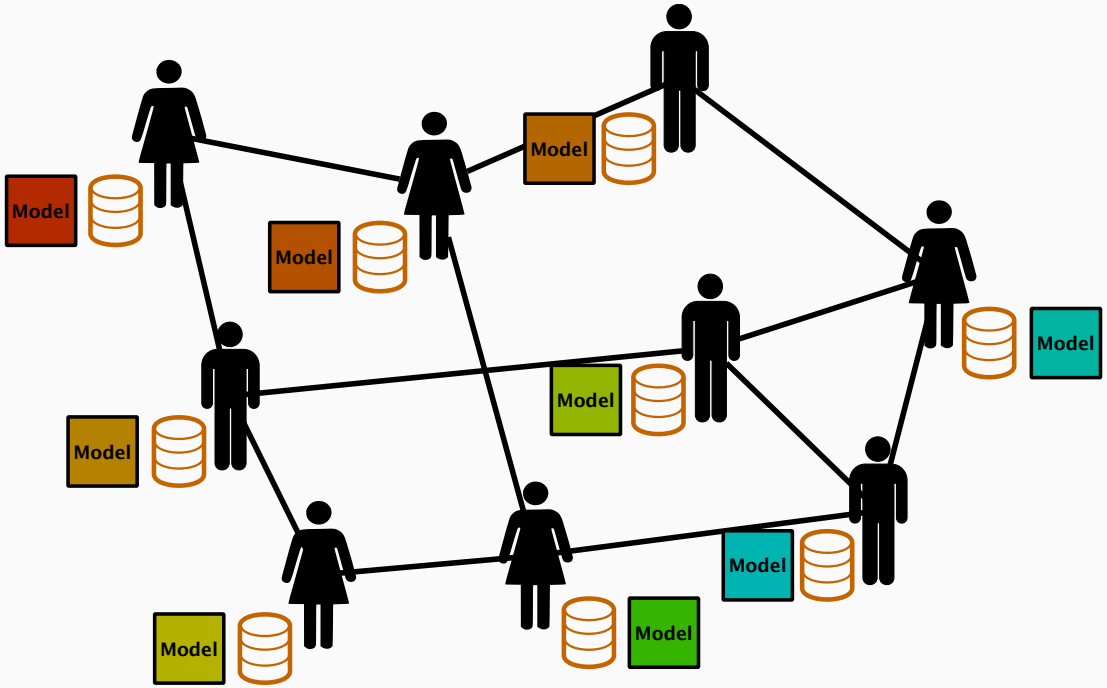
- Gossip algorithms to optimize a loss function over the union of personal datasets [Nedic and Ozdaglar, 2009, Duchi et al., 2012, Wei and Ozdaglar, 2012, Colin et al., 2016, Scaman et al., 2017]



- General idea: at each step
  1. perform a local model update based on personal data
  2. average with neighbor

THIS WORK: PERSONALIZED LEARNING

- Gossip algorithms to learn a personalized model for each user according to its own learning objective



- General idea: trade-off between model accuracy on local data and smoothness with respect to similar users

PROBLEM SETTING

PROBLEM SETTING

- We work in the supervised learning setting: predict label $y \in \mathcal{Y}$ from observation $x \in \mathcal{X}$
- The set of possible prediction models will be indexed by parameters $\theta \in \mathbb{R}^p$
- We use a convex loss function $\ell : \mathbb{R}^p \times \mathcal{X} \times \mathcal{Y}$ to measure the error of a model on a labeled observation
- We have a set $V = [\![n]\!] = \{1, \ldots, n\}$ of n learning agents
- Agent i has dataset $\mathcal{S}_i = \{(x_i^j, y_i^j)\}_{j=1}^{m_i}$ of size $m_i \geq 0$ drawn i.i.d. from its own distribution μ_i over $\mathcal{X} \times \mathcal{Y}$

- Goal of agent i: learn a model $\theta_i \in \mathbb{R}^p$ with small expected loss

$$\mathop{\mathbb{E}}_{(x_i,y_i)\sim\mu_i} \ell(\theta_i; x_i, y_i)$$

- In isolation, agent i can learn a "solitary" model

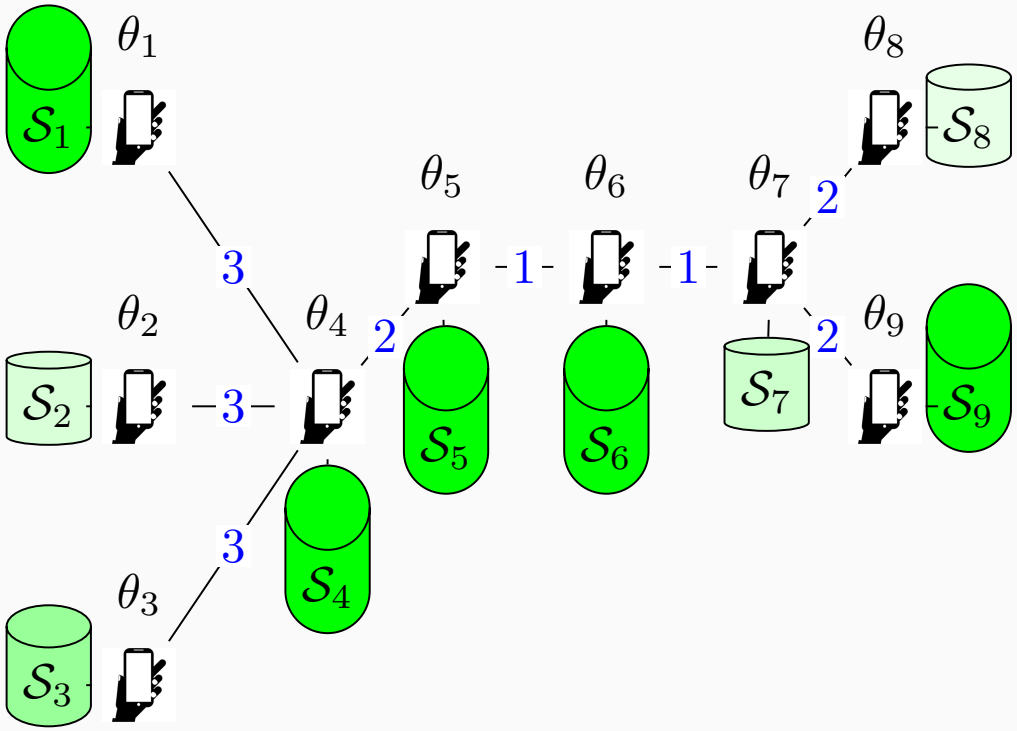
$$\theta_i^{sol} \in \arg\min_{\theta\in\mathbb{R}^p} \mathcal{L}_i(\theta) = \frac{1}{m_i}\sum_{j=1}^{m_i} \ell(\theta; x_i^j, y_i^j) + \lambda_i\|\theta\|^2, \text{ with } \lambda_i \geq 0$$

- **How to improve upon θ_i^{sol} with the help of other users?**

PROBLEM SETTING

- Network: weighted connected graph $G = (V, E)$
- $E \subseteq V \times V$ set of undirected edges
- Weight matrix $W \in \mathbb{R}^{n \times n}$: symmetric, nonnegative, with $W_{ij} = 0$ if $(i, j) \notin E$ or $i = j$
- **Assumption**: network weights are given and represent the underlying similarity between agents
  - Ex: movie recommendation task where the network is set up when users go to the same movie

PROBLEM SETTING



- Agents have only a local view of the network
- They only know their neighborhood $\mathcal{N}_i = \{j \neq i : W_{ij} > 0\}$ and the associated weights

DECENTRALIZED ALGORITHMS

Main idea: smooth the solitary models over the network

- $c_i \in (0, 1]$: confidence in initial model θ_i^{sol}
  - Proportional to the number of training points m_i
- Find new set of models $\Theta \in \mathbb{R}^{n \times p}$ by solving

$$\min_{\Theta \in \mathbb{R}^{n \times p}} \mathcal{Q}_{MP}(\Theta) = \frac{1}{2} \left(\sum_{i<j}^{n} W_{ij} \|\theta_i - \theta_j\|^2 + \mu \sum_{i=1}^{n} D_{ii} c_i \|\theta_i - \theta_i^{sol}\|^2 \right)$$

  - Trade-off between smoothing models within neighborhoods and not diverging too much from confident models
  - Term $D_{ii} = \sum_j W_{ij}$ is just for normalization
- Cannot use closed-form solution (requires global knowledge)

ASYNCHRONOUS GOSSIP ALGORITHM

- Each agent has a local Poisson clock and wakes up when it ticks
- Equivalently: single clock (with counter t) ticking when one of the local clocks ticks
- **Idea of our algorithm**: each agent i maintains a (possibly outdated) knowledge $\widetilde{\Theta}_i(t) \in \mathbb{R}^{n \times p}$ of its neighbors' models
  - $\widetilde{\Theta}_i^i(t) \in \mathbb{R}^p$: agent i's model at time t
  - for $j \neq i$, $\widetilde{\Theta}_i^j(t) \in \mathbb{R}^p$: agent i's *last knowledge* of the model of j
  - For $j \notin \mathcal{N}_i \cup \{i\}$ and any $t > 0$, we maintain $\widetilde{\Theta}_i^j(t) = 0$

ASYNCHRONOUS GOSSIP ALGORITHM

- At step t, some agent i wakes up and two actions are performed
  1. *Communication step*: agent i selects a random neighbor $j \in \mathcal{N}_i$ w.p. π_i^j and both agents update their knowledge of each other:

$$\widetilde{\Theta}_i^j(t+1) = \widetilde{\Theta}_j^j(t) \text{ and } \widetilde{\Theta}_j^i(t+1) = \widetilde{\Theta}_i^i(t),$$

  2. *Update step*: agents i and j update their own models based on current knowledge. For $l \in \{i, j\}$:

$$\widetilde{\Theta}_l^l(t+1) = (\alpha + \bar{\alpha} c_l)^{-1} \Big(\alpha \sum_{k \in \mathcal{N}_l} \frac{W_{lk}}{D_{ll}} \widetilde{\Theta}_l^k(t+1) + \bar{\alpha} c_l \theta_l^{sol} \Big).$$

- All other variables in the network remain unchanged
- For any $i \in [\![n]\!]$, $\pi_i \in [0,1]^n$ must satisfy $\sum_{j=1}^n \pi_i^j = 1$ and $\pi_i^j > 0$ if and only if $j \in \mathcal{N}_i$

Theorem ([Vanhaesebrouck et al., 2017])

Let $\widetilde{\Theta}(0) \in \mathbb{R}^{n^2 \times p}$ be some 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 $i \in [\![n]\!]$,

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

Sketch of proof

- Rewrite algorithm as a random iterative process over $\widetilde{\Theta} \in \mathbb{R}^{n^2 \times p}$:

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

- Show that spectral radius of $\mathbb{E}[A(t)]$ is smaller than 1
- Exhibit convergence to desired quantity

COLLABORATIVE LEARNING: PROBLEM FORMULATION

- Model propagation is very simple but forgets data
- Alternative: learn / propagate models simultaneously by solving

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

- Trade-off between smoothing models within neighborhoods and good accuracy on local datasets
- More flexibility than model propagation in settings where different parameter values may lead to similar predictions
- Can recover the two extreme cases of learning purely local models ($\mu \to \infty$) and learning a single global model ($\mu \to 0$)

ALGORITHM IN THE BROADCAST SETTING

- For simplicity, consider the broadcast communication model
  - The agent which wakes up sends a message to all its neighbors
  - No reply from neighbors
- Assume that local loss $\mathcal{L}_i$ has L_i^{loc}-Lipschitz continuous gradient
- Then $\nabla \mathcal{Q}_{CL}$ is L_i-Lipschitz w.r.t. block Θ_i with $L_i = D_{ii}(1 + \mu c_i L_i^{loc})$
- Can also assume that $\mathcal{L}_i$ is σ_i^{loc}-strongly convex where $\sigma_i^{loc} > 0$
- Then $\mathcal{Q}_{CL}$ is σ-strongly convex with $\sigma \geq \mu \min_{1 \leq i \leq n} [D_{ii} c_i \sigma_i^{loc}] > 0$

- Randomized block coordinate descent algorithm: assume agent i wakes up at step t:
  1. Agent i updates its model based on information from neighbors:

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

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

  2. Agent i sends its updated model $\Theta_i(t+1)$ to its neighborhood $\mathcal{N}_i$

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 $\Theta(0)$. When $\mathcal{Q}_{CL}$ is σ-strongly convex, we have:

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

where $L_{max} = \max_i L_i$.

- Follows from randomized CD analysis [Wright, 2015]
- Can obtain convergence in $O(1/T)$ in convex case

PRIVACY IN DECENTRALIZED LEARNING

PRIVACY SETTING

- In some applications, data may be sensitive and agents may not want to reveal it to anyone else
- In our algorithms, the 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:** how can we guarantee that no/little information about the local dataset is leaked by the algorithm?

(ϵ, δ)-Differential Privacy

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
- Information-theoretic (no computational assumptions)

DIFFERENTIALLY PRIVATE ALGORITHM

1. Replace the update of the algorithm in broadcast setting by

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

   where $\eta_i(t) \sim Laplace(0, s_i(t))^p \in \mathbb{R}^p$
2. Agent i then broadcasts noisy iterate $\widetilde{\Theta}_i(t+1)$ to its neighbors

Theorem ([Bellet et al., 2018])

Let $i \in [\![n]\!]$ and assume

- *$\ell(\cdot; x, y)$ L_0-Lipschitz w.r.t. the L_1-norm for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$*
- *Agent i wakes up on iterations $t_i^1, \ldots, t_i^{T_i}$*
- *For some $\epsilon_i(t_i^k) > 0$, the noise scale is $s_i(t_i^k) = \frac{2L_0}{\epsilon_i(t_i^k) m_i}$*

Then for any initial point $\widetilde{\Theta}(0)$ independent of $\mathcal{S}_i$, the mechanism $\mathcal{M}_i(\mathcal{S}_i)$ is $(\bar{\epsilon}_i, 0)$-DP with $\bar{\epsilon}_i = \sum_{k=1}^{T_i} \epsilon_i(t_i^k)$.

- Follows from sensitivity analysis of the update
- Sweet spot in collaborative learning: the less data, the more noise added by the agent, but the least influence in the network
- Can be improved by applying strong composition theorems

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. For σ-strongly convex $\mathcal{Q}_{CL}$, we have:

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

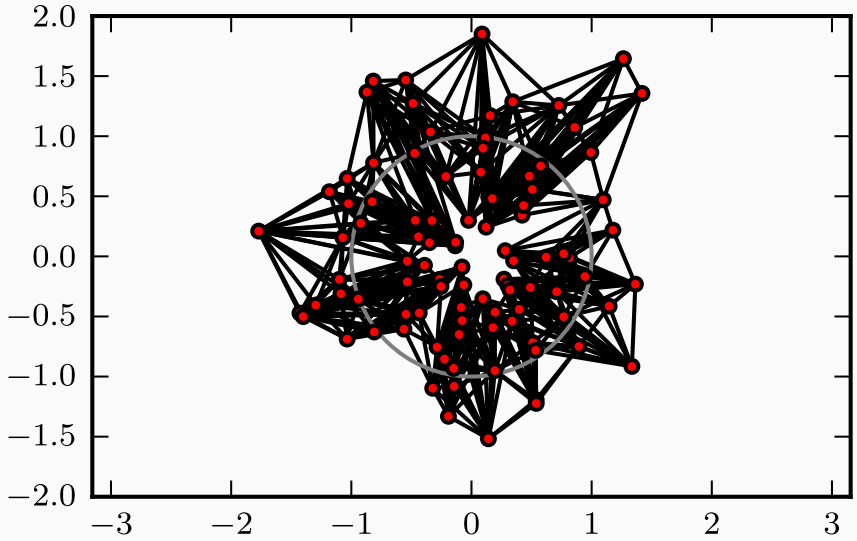
where $L_{min} = \min_{1 \leq i \leq n} L_i$.

- Second term gives additive error due to noise
- When noise scale of each agent constant across iterations, this additive error converges to a finite number as $T \to \infty$
- More results on how to scale noise in the paper

ILLUSTRATIVE EXPERIMENTS

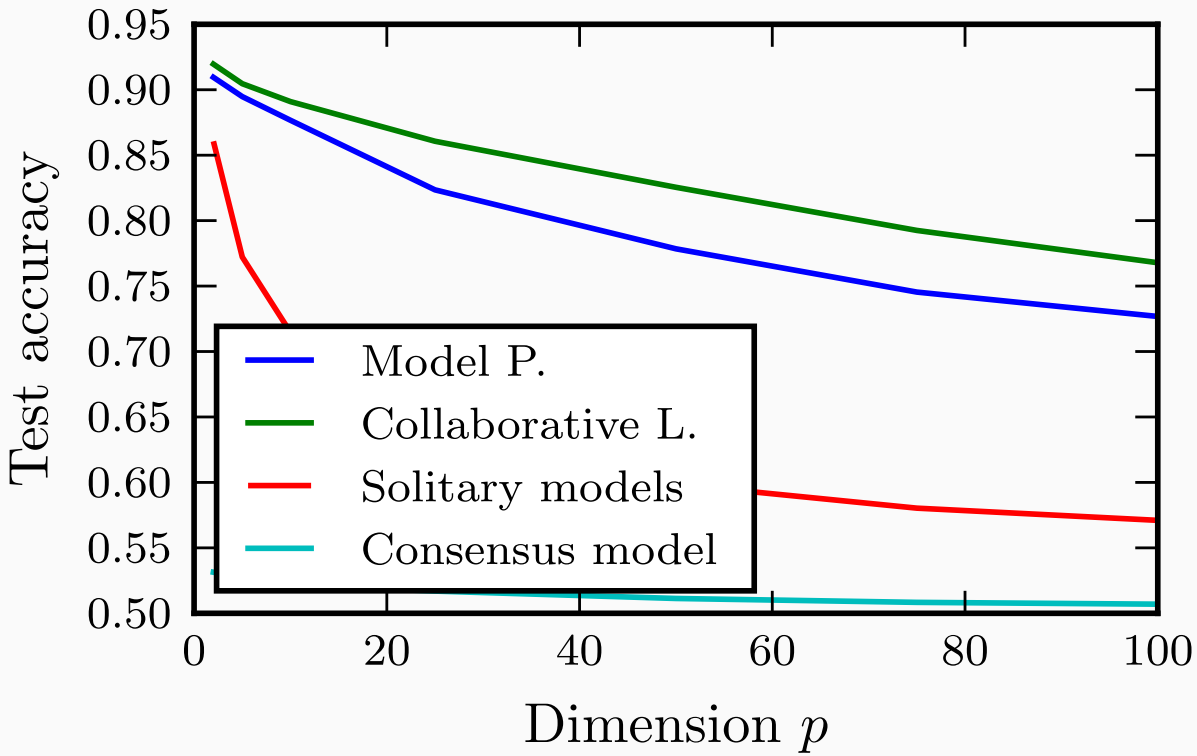
COLLABORATIVE LINEAR CLASSIFICATION

- We consider a set of $n = 100$ agents and a linear classification task in $\mathbb{R}^p$ (with hinge loss)
- Target models lie in a 2D subspace, network weights based on the angle between true models
- Each agent i receives a random number m_i of samples with label given by the prediction of target model (plus noise)



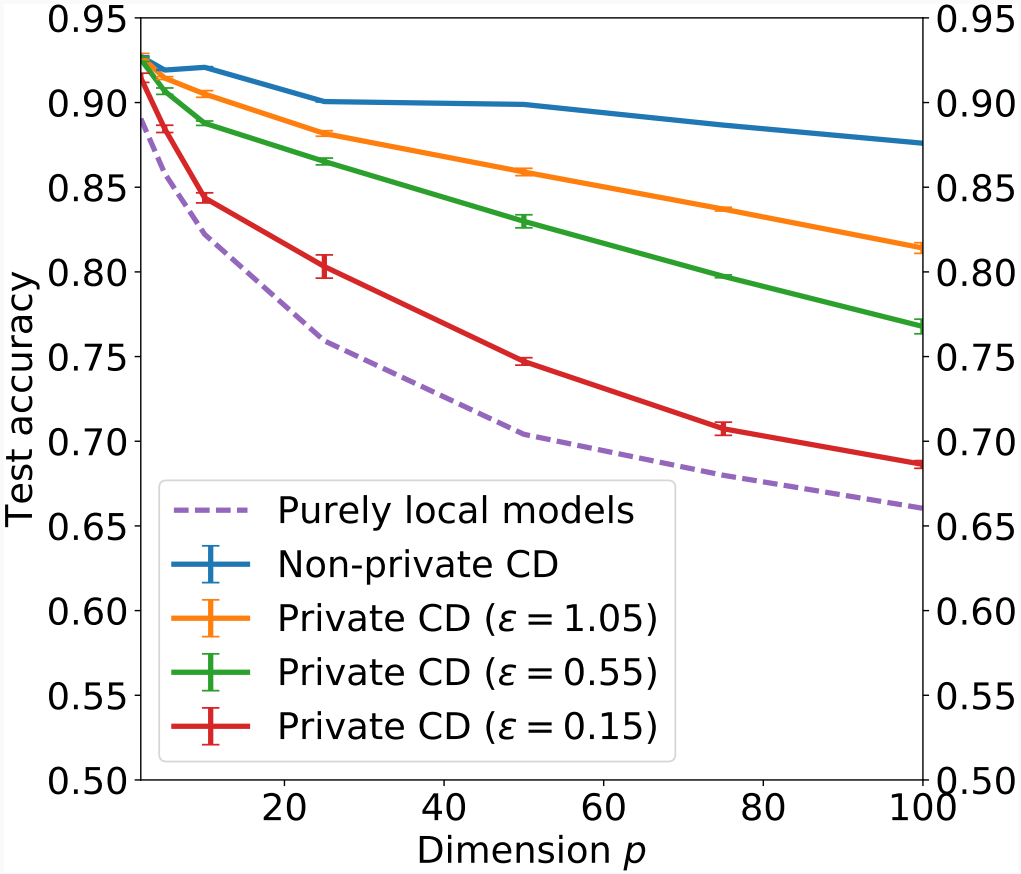
COLLABORATIVE LINEAR CLASSIFICATION: NON-PRIVATE SETTING

- Both CL and MP provide great improvements over local models
- CL consistently outperforms MP by significant margin



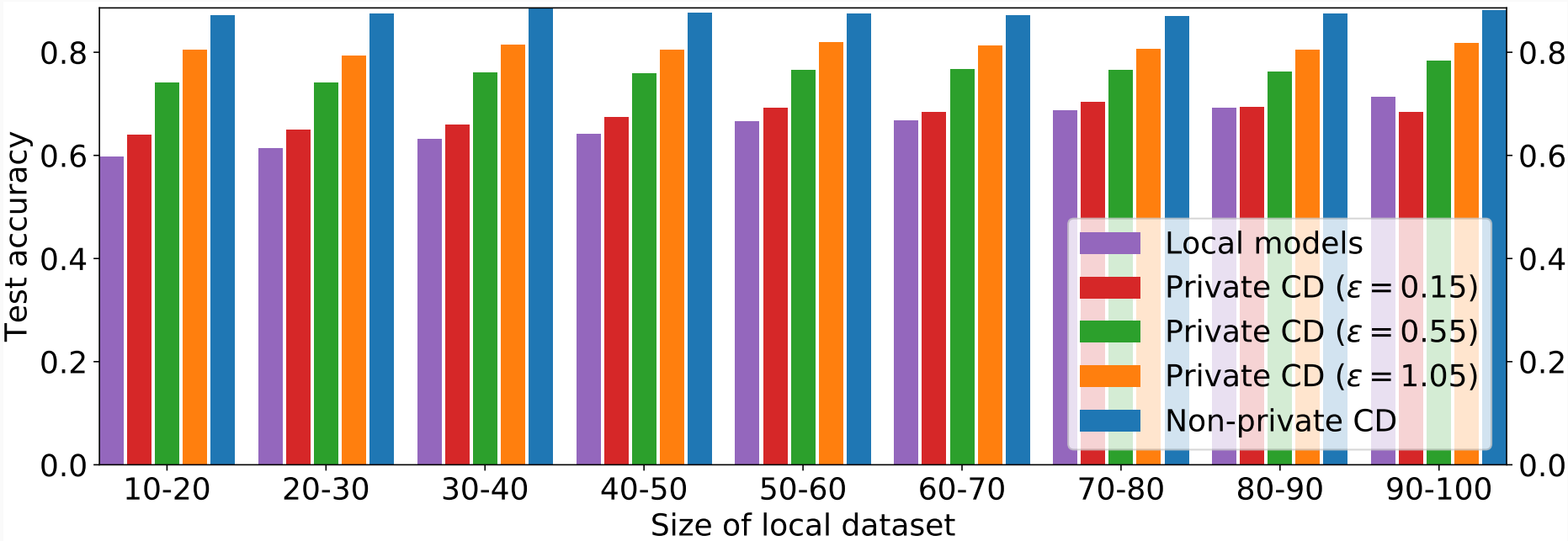
COLLABORATIVE LINEAR CLASSIFICATION: PRIVATE SETTING

- The private variant outperforms the purely local models for "reasonable" values of ϵ



COLLABORATIVE LINEAR CLASSIFICATION: PRIVATE SETTING

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



MOVIE RECOMMENDATION

- MovieLens-100K: 100,000 ratings given by $n = 943$ users over $1,682$ movies
- Each user has access only to its own ratings (80% train, 20% test)
- For simplicity, assume known features $\phi_j \in \mathbb{R}^p$ for each movie j
- Quadratic loss: $\ell(\theta; \phi, r) = (\theta^T \phi - r)^2$
- Network: 10-NN graph with cosine similarity on training ratings
- Error: RMSE averaged over users

	Purely local	Non-priv. CD	Priv. $\bar{\epsilon} = 1$	Priv. $\bar{\epsilon} = 0.5$	Priv. $\bar{\epsilon} = 0.1$
Test error	1.2834	0.9502	0.9527	0.9545	0.9855

FUTURE WORK

FUTURE WORK

Theory

- Statistical generalization bounds
- Generic methods to estimate/learn graph weights
- Online learning: data arrive sequentially, agents may join/leave

Applications and practical use

- More real-world experiments (e.g., activity recognition)
- Extend to nonconvex case (e.g., deep nets)
- Decentralized discovery of similar peers
- Cryptographic tools to achieve better accuracy (at the cost of more computation)

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*.

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

[Duchi et al., 2012] 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.

[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*.

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

[Scaman et al., 2017] Scaman, K., Bach, F., Bubeck, S., Lee, Y. T., and Massoulié, L. (2017).
Optimal algorithms for smooth and strongly convex distributed optimization in networks.
Technical report, arXiv:1702.08704.

References II

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

[Wei and Ozdaglar, 2012] Wei, E. and Ozdaglar, A. E. (2012).
Distributed Alternating Direction Method of Multipliers.
In *CDC*, pages 5445–5450.

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