FULLY DECENTRALIZED JOINT LEARNING OF PERSONALIZED MODELS AND COLLABORATION GRAPHS

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

Workshop “The power of graphs in machine learning and sequential decision-making”
University College London — June 3, 2019
CONTEXT AND MOTIVATION
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
- **Goal**: find a sweet spot between these two extremes
• Coordinator-clients architecture [McMahan et al., 2017]

• Iterates over the following (synchronous) steps:
  • Clients send model updates computed on local data
  • Coordinator aggregates and sends the new model back to clients

• Heavy dependence on coordinator: single point of failure and scalability issues with large number of clients
• Peer-to-peer and asynchronous exchanges along sparse communication graph

• No single point of failure as in classic federated learning

• Scalability-by-design to many devices (see e.g., [Lian et al., 2017])
1. Keep data on the device of the users
2. Learn personalized models in collaborative fashion
3. Learn and leverage a graph of similarities between users
4. Decentralized algorithms to scale to large number of devices

And also (not in this talk):

5. Differential privacy guarantees [Bellet et al., 2018]
6. Low-communication via greedy boosting [Zantedeschi et al., 2019]
PROBLEM FORMULATION
• A set $[n] = \{1, \ldots, n\}$ of users (or agents)

• Each user has a personal distribution over common feature space $\mathcal{X}$ and label space $\mathcal{Y}$ → personal supervised learning task

• User $i$ has local dataset $\mathcal{S}_i = \{(x^i_j, y^i_j)\}_{j=1}^{m_i}$ of size $m_i \geq 0$ drawn from personal distribution

• His/her goal is to learn a model $\theta_i \in \mathbb{R}^p$ which generalizes well to new examples from personal distribution
• Let $\ell : \mathbb{R}^p \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ be a loss function
• In isolation, user $i$ can learn a purely local model by ERM

$$
\theta_{i}^{loc} \in \arg \min_{\theta \in \mathbb{R}^p} \mathcal{L}_i(\theta; S_i) = \frac{1}{m_i} \sum_{j=1}^{m_i} \ell(\theta; x^j_i, y^j_i) + \lambda_i \|\theta\|^2, \text{ with } \lambda_i \geq 0
$$

• Poor generalization when local data is scarce

• **Goal:** improve upon $\theta_{i}^{loc}$ by discovering relationships between personal tasks and leveraging them to learn better personalized models in a fully decentralized setting
DECENTRALIZED SETTING

- **Asynchronous time model**: each user has a local Poisson clock and wakes up when it ticks [Boyd et al., 2006]

- Equivalently: single clock (with counter $t$, unknown to the users) ticking when one of the local clocks ticks

- **Communication model**: semantic overlay over complete graph to restrict communication to pairs of most similar users

- We call this overlay the collaboration graph: undirected, weighted graph $G_w = ([n], w)$ with edge weight $w_{ij} \geq 0$ reflecting similarity between the learning tasks of users $i$ and $j$

- **Our method**: learn sparse collaboration graph and personalized models by optimization of a joint objective
JOINT OPTIMIZATION PROBLEM

- Learn personalized models \( \Theta \in \mathbb{R}^{n \times p} \) and graph weights \( w \in \mathbb{R}^{n(n-1)/2}_{\geq 0} \) as solutions to [Zantedeschi et al., 2019]:

\[
\min_{\Theta \in \mathbb{R}^{n \times p}, \ w \in \mathbb{R}^{n(n-1)/2}_{\geq 0}} \ J(\Theta, w) = \sum_{i=1}^{n} d_i c_i \mathcal{L}_i(\theta_i; S_i) + \frac{\mu}{2} \sum_{i<j} w_{ij} \|\theta_i - \theta_j\|^2 + \lambda g(w),
\]

- \( c_i \in (0, 1] \propto m_i \): confidence of user \( i \), \( d_i = \sum_{j \neq i} w_{ij} \): degree of \( i \)

- Trade-off between having accurate models on local dataset and smoothing models along the graph

- Term \( g(w) \): avoid trivial graphs, encourage desirable properties

- Flexible relationships: \( \mu \) interpolates between learning purely local models and a shared model per connected component
Problem not jointly convex in $\Theta$ and $w$, but is typically bi-convex

Natural approach: alternating optimization over $\Theta$ and $w$

I will present a decentralized algorithm to learn the models given the graph (communication along edges of the graph)

I will then present a decentralized algorithm to learn a (sparse) graph given the models (communication through peer sampling)
LEARNING MODELS GIVEN THE GRAPH
• For fixed graph weights, denote $f(\Theta) := J(\Theta, w)$
• Assume local loss $\mathcal{L}_i$ has $L_i^{loc}$-Lipschitz continuous gradient
• Then $\nabla_{\Theta}f$ is $L_i$-Lipschitz w.r.t. block $\Theta_i$ with $L_i = d_i(\mu + c_iL_i^{loc})$
• Can also assume that $\mathcal{L}_i$ is $\sigma_i^{loc}$-strongly convex where $\sigma_i^{loc} > 0$
• Then $f$ is $\sigma$-strongly convex with $\sigma \geq \min_{1 \leq i \leq n}[d_ic_i\sigma_i^{loc}] > 0$
• Denote neighborhood of user $i$ by $N_i = \{j : w_{ij} > 0\}$

• Initialize models $\Theta_i(0) \in \mathbb{R}^{n \times p}$

• At step $t \geq 0$, a random user $i$ wakes up:
  1. user $i$ updates its model based on information from neighbors:

$$\Theta_i(t+1) = \Theta_i(t) - \frac{1}{\mu + c_i \mathcal{L}_{i}} \left( c_i \nabla \mathcal{L}_i(\Theta_i(t); S_i) - \mu \sum_{j \in N_i} \frac{w_{ij}}{d_i} \Theta_j(t) \right)$$

  2. user $i$ sends its updated model $\Theta_i(t+1)$ to its neighborhood $N_i$

• 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 $\Theta(0)$. When $f$ is $\sigma$-strongly convex in $\Theta$, we have:

$$E[f(\Theta(T)) - f^*] \leq \left(1 - \frac{\sigma}{nL_{\text{max}}}\right)^T (f(\Theta(0)) - f^*) ,$$

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

- Essentially follows from coordinate descent analysis [Wright, 2015]
- Can obtain convergence in $O(1/T)$ in convex case
- Can extend analysis to the case where random noise is added to ensure differential privacy [Bellet et al., 2018]
LEARNING THE GRAPH GIVEN MODELS
\[
\min_{\Theta \in \mathbb{R}^{n \times p}, \; w \in \mathbb{R}_{\geq 0}^{n(n-1)/2}} J(\Theta, w) = \sum_{i=1}^{n} d_i c_i \mathcal{L}_i(\theta_i; S_i) + \frac{\mu}{2} \sum_{i<j} w_{ij} \|\theta_i - \theta_j\|^2 + \lambda g(w),
\]

- Our algorithm can deal with weight and degree-separable \( g \)
- Inspired by [Kalofolias, 2016], we can set \( \lambda = \mu \) and define
  \[
g(w) = \beta \|w\|^2 - 1^T \log(d + \delta) \quad \text{(with } \delta \text{ small constant)}
\]
- Log barrier on the degree vector \( d \) to avoid isolated users and \( L_2 \) penalty on weights to control the graph sparsity
- Denoting \( h(w) := J(\Theta, w) \) for fixed \( \Theta \), then \( h \) is strongly convex
We rely on decentralized peer sampling [Jelasity et al., 2007] to allow users to communicate with a set of $\kappa$ random peers

- Initialize weights $w(0)$, set parameter $\kappa \in [n - 1]$

- At each step $t \geq 0$, a random user $i$ wakes up:
  1. Draw a set $\mathcal{K}$ of $\kappa$ users and request their model, loss and degree
  2. Update the associated weights $w(t + 1)_{i, \mathcal{K}} = (w(t + 1)_{ij})_{j \in \mathcal{K}} \in \mathbb{R}^\kappa$:

$$w(t + 1)_{i, \mathcal{K}} \leftarrow \max (0, w(t)_{i, \mathcal{K}} - \frac{1}{L_{\mathcal{K}}} [\nabla h(w(t))]_{i, \mathcal{K}})$$

where $L_{\mathcal{K}} = 2\mu \left( \frac{\kappa + 1}{\delta^2} + \beta \right)$ is the block Lipschitz constant of $\nabla h(w)$

3. Send each updated weight $w(t + 1)_{ij}$ to the associated user $j \in \mathcal{K}$
Theorem ([Zantedeschi et al., 2019])

For any $T > 0$, let $(w(t))_{t=1}^{T}$ be the sequence of iterates generated by the algorithm running for $T$ iterations from an initial point $w(0)$. We have $\mathbb{E}[h(w(T)) - h^*] \leq \rho^T (h(w(0)) - h^*)$ where $\rho$ is given by

$$\rho = 1 - \frac{4}{n(n-1)} \frac{\kappa \beta \delta^2}{\kappa + 1 + 2\beta \delta^2}$$

- Can be seen as an instance of proximal coordinate descent with an overlapping block structure
- $\kappa$ can be used to trade-off between communication cost and convergence speed (more on this in [Zantedeschi et al., 2019])
NUMERICAL EXPERIMENTS
We consider a set of $n = 100$ users and a synthetic linear classification task in $\mathbb{R}^p$ (we use the hinge loss).

Each user is associated with an (unknown) target linear model.

Each user $i$ receives a random number $m_i$ of samples with label given by the prediction of target model (plus noise).

We can build an “oracle” collaboration graph based on the angle between target models (note: this is cheating!)
EXPERIMENTS: COLLABORATIVE LINEAR CLASSIFICATION

- Results when using the oracle graph
• We show that the learned topology adapts to the problem, unlike classic heuristics (e.g., $k$-NN graph)
• Below we approximately recover the cluster structure, and prediction accuracy is close to that of the oracle graph
**EXPERIMENTS: REAL DATASETS**

- Real datasets that are naturally collected at the user level
- Number of users $n$ from 23 to 190, no task similarity available
- Linear models, and nonlinear ensembles [Zantedeschi et al., 2019]
- Our approach **clearly outperforms** global and local models
- Greedily trained nonlinear ensembles achieve better accuracy under communication budget (see [Zantedeschi et al., 2019])

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Global-lin</th>
<th>Local-lin</th>
<th>Ours-lin</th>
<th>Global-nonlin</th>
<th>Local-nonlin</th>
<th>Ours-nonlin</th>
</tr>
</thead>
<tbody>
<tr>
<td>HARWS</td>
<td>93.64</td>
<td>92.69</td>
<td><strong>96.31</strong></td>
<td>94.34</td>
<td>93.16</td>
<td>95.70</td>
</tr>
<tr>
<td>VEHICLE</td>
<td>87.11</td>
<td>90.38</td>
<td><strong>91.37</strong></td>
<td>88.02</td>
<td>90.59</td>
<td>90.81</td>
</tr>
<tr>
<td>COMPUTER</td>
<td>62.18</td>
<td>60.68</td>
<td>69.08</td>
<td><strong>69.16</strong></td>
<td>66.61</td>
<td><strong>72.09</strong></td>
</tr>
<tr>
<td>SCHOOL</td>
<td>57.06</td>
<td>70.43</td>
<td><strong>71.92</strong></td>
<td>69.16</td>
<td>66.61</td>
<td><strong>72.22</strong></td>
</tr>
</tbody>
</table>

*bold blue = best, regular blue = second best*
FUTURE WORK
• Extend analysis to nonconvex setting (deep neural nets)
• Use the graph to smooth predictions rather than model parameters
• Learn graph weights as statistical estimates of some distance between data distributions and prove generalization guarantees
• Dynamic setting: data arrives sequentially, users join/leave
• Robustness to malicious parties [Dellenbach et al., 2018]
Thank you for your attention!
Questions?


How to learn a graph from smooth signals.
In AISTATS.

Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for
Decentralized Parallel Stochastic Gradient Descent.
In NIPS.

(2017).
Communication-efficient learning of deep networks from decentralized data.
In AISTATS.

Coordinate descent algorithms.

Communication-efficient and decentralized multi-task boosting while learning the
collaboration graph.
• In some applications, data may be sensitive and users may not want to reveal it to anyone else

• In the previous algorithm, users 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 users)

• **Goal:** formally guarantee that no/little information about the local dataset is leaked by the algorithm
### $(\epsilon, \delta)$-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 $(\epsilon, \delta)$-differentially private if for all datasets $S, S'$ differing in a single data point and for all sets of possible outputs $\mathcal{O} \subseteq \text{range}(\mathcal{M})$, we have:

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

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

\[
\tilde{\Theta}_i(t+1) = \tilde{\Theta}_i(t) - \frac{1}{\mu + c_iL^i_{loc}} \left( c_i(\nabla \mathcal{L}_i(\tilde{\Theta}_i(t); S_i) + \eta_i) - \mu \sum_{j \in N_i} \frac{w_{ij}}{d_i} \tilde{\Theta}_j(t) \right),
\]

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

2. user \( i \) then broadcasts noisy iterate \( \tilde{\Theta}_i(t+1) \) to its neighbors
In our setting, the output of our algorithm is the sequence of users’ models sent over the network.

**Theorem ([Bellet et al., 2018])**

Assume user $i$ wakes up $T_i$ times and use noise scale $s_i = \frac{L_0}{\epsilon_i m_i}$. Then for any initial point $\Theta(0)$ independent of $S_i$, the algorithm is $(\bar{\epsilon}_i, 0)$-DP with $\bar{\epsilon}_i = T_i \epsilon_i$. 
Theorem ([Bellet et al., 2018])

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

$$
\mathbb{E} \left[ (\tilde{\Theta}(T)) - * \right] \leq \rho^T \left( (\tilde{\Theta}(0)) - * \right) + \left( \frac{1}{(1 - \rho)Cn} \sum_{i=1}^{n} (d_i c_i s_i)^2 \right) \left( 1 - \rho^T \right)
$$

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