Optimization in Open Networks via Dual Averaging

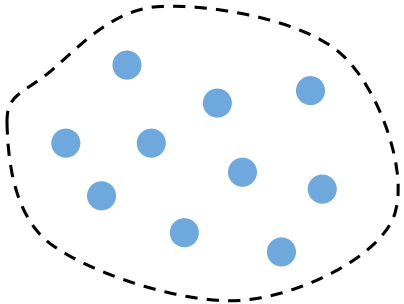
Yu-Guan Hsieh, Franck Iutzeler, Jérôme Malick, and Panayotis Mertikopoulos

CDC 2021

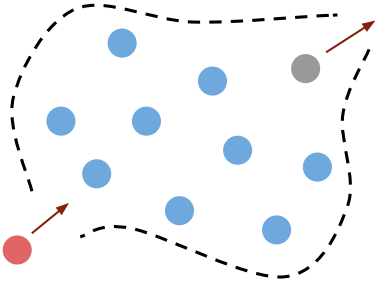
Plan

What is an open network?

An open network is a network that agents can join and leave at any time



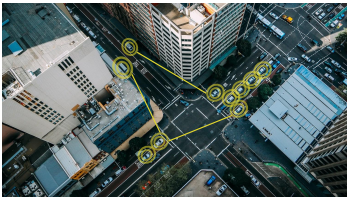
Closed network



Open network

Why do we care about open networks?

- Elastic distributed machine learning
- Volunteer computing
- Vehicular ad-hoc networks (VANETs)



Plan

Model

- Discrete time steps $t = 1, 2, \ldots$
- $\mathcal{V}_t$ denotes the set of agents present at time t
- $m_t = \mathrm{card}(\mathcal{V}_t)$: number of agents that are presented at time t
- $M_t = \sum_{s=1}^{t} m_t$: the sum of m_t over time
- Each agent $i \in \bigcup_{t \in \mathbb{N}} \mathcal{V}_t$ is associated with an individual convex cost function $f^i : \mathbb{R}^d \to \mathbb{R}$

Objectives in stationary environment

- If the underlying problem is **stationary**, we may define the running loss as

$$f_T^{\mathsf{run}}(x) = \frac{1}{M_T} \sum_{t=1}^{T} \sum_{i \in \mathcal{V}_t} f^i(x)$$

- The performance of an algorithm is measured by its average static regret

$$\overline{\mathbf{Loss}}(T) = \frac{1}{M_T} \left(\sum_{t=1}^{T} \sum_{i \in \mathcal{V}_t} f^i(x_t^{\mathsf{ref}}) - \underbrace{\min_{u \in \mathcal{X}} \sum_{t=1}^{T} \sum_{i \in \mathcal{V}_t} f^i(u)}_{\text{minimum running loss}} \right)$$

where $\mathcal{X} \subset \mathbb{R}^d$ is the shared constrained set and x_t^{ref} is a reference point for time t

Objectives in non-stationary environment

- If the underlying problem is **non-stationary**, we may define the instantaneous loss as

$$f_t^{\text{inst}}(x) = \frac{1}{m_t} \sum_{i \in \mathcal{V}_t} f^i(x)$$

- The performance of an algorithm is measured by its ability to track the optimum
  - Maximum tracking error: $\max_{t \geq t_0} \left\{ f_t^{\text{inst}}(x^{\text{ref}}) - \min_{u \in \mathcal{X}} f_t^{\text{inst}}(u) \right\}$
  - Dynamic regret: $\sum_{t=1}^{T} m_t \left\{ f_t^{\text{inst}}(x_t^{\text{ref}}) - \min_{u \in \mathcal{X}} f_t^{\text{inst}}(u) \right\}$

Plan

DAERON: Dual AvERaging for Open Network

Just accumulate gradients from the whole network!

- At each time step t, each agent $i \in \mathcal{V}_t$ computes a subgradient $g_t^i \in \partial f^i(x_t^i)$
- The subgradients are transmitted over the network
- $\mathcal{S}_t^i$ is the index set of the subgradients that $i \in \mathcal{V}_t$ has received/computed by time t
- The variable x_t^i is obtained by

$$x_t^i = \Pi_{\mathcal{X}} \left(-\eta \sum_{(j,s) \in \mathcal{S}_t^i} g_s^j \right) \tag{1}$$

where $\Pi_{\mathcal{X}}$ denotes the projection onto $\mathcal{X}$ and $\eta > 0$ is a common learning rate

DAERON: Dual AvERaging for Open Network

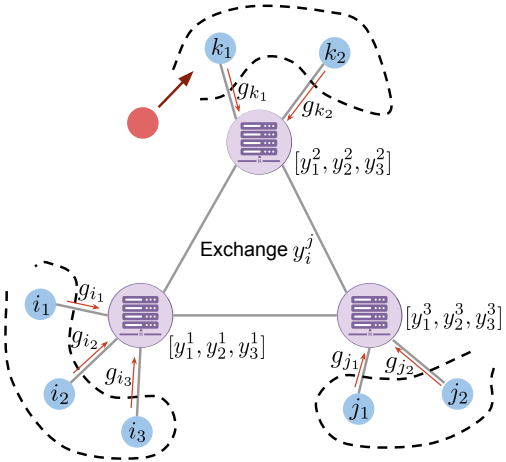
Algorithm DAERON at node i for active period t^{join}-t^{leave}

Parameters: t^{join} time when the agent joins the network; t^{leave} time when the agent leaves the network
Initialize: $\mathcal{S}^i_{t^{\text{join}}} \leftarrow \mathcal{S}^j_{t^{\text{join}}}$ for $j \in \mathcal{V}_{t^{\text{join}}-1} \cap \mathcal{V}_{t^{\text{join}}}$
for $t = t^{\text{join}}, \ldots, t^{\text{leave}}$ **do**
    Generate the prediction x^i_t using (1)
    Compute the local subgradient $g^i_t \in \partial f^i(x^i_t)$
    Send subgradients to other active agents
    Receive subgradients identified by the index set $\mathcal{G}^i_t$
    Update $\mathcal{S}^i_{t+1} \leftarrow \mathcal{S}^i_t \cup \mathcal{G}^i_t \cup \{g^i_t\}$
end for

Practical implementation: the case of semi-centralized network

- y_i^i collects subgradients from agents that are affiliated to node i
- y_i^j is an outdated version of y_i^i and is updated through exchange between the storage nodes
- Storage node i sends $y^i = \sum_j y_j^i$ as accumulated gradients to its affiliated agents

(Time index is omitted throughout)



Convergence of average static regret

Let us define the quadratic mean and the average number of active agents over time

$$m_{\mathsf{QM}} = \sqrt{\frac{1}{T}\sum_{t=1}^{T} m_t^2} \quad \text{and} \quad \bar{m} = \frac{1}{T}\sum_{t=1}^{T} m_t$$

Theorem (Convergence of DAERON)

Assume that

1. *Each f^i is convex and G-Lipschitz and $\mathcal{X}$ is closed and convex*
2. *The delays are bounded by τ*

Then, with suitable learning rate, DAERON ensures $\boxed{\overline{\mathbf{Loss}}(T) = \mathcal{O}\left(\frac{m_{\mathsf{QM}} G\sqrt{\tau}}{\bar{m}\sqrt{T}}\right)}$

Convergence of average static regret

$$\overline{\mathbf{Loss}}(T) = \mathcal{O}\left(\frac{m_{\mathsf{QM}} G\sqrt{\tau}}{\bar{m}\sqrt{T}}\right)$$

- Delays deteriorate the bound by a multiplicative factor of $\sqrt{\tau}$.
- $m_{\mathsf{QM}}/\bar{m} \geq 1$ and this ratio becomes larger if m_t varies greatly across iterations.
- When $\mathcal{V}_t$ is fixed over time, we obtain $\mathcal{O}(\sqrt{\tau/T})$ convergence rate of the usual quantity that we seek to minimize.
- It is possible to derive similar guarantees for *adaptive* learning rates that are both time- and agent-dependent.

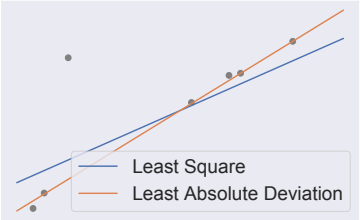
Plan

Decentralized least absolute deviation (LAD) regression

Given a data set evenly distributed on m nodes $(a_{ik}, b_{ik})_{i,k\in[m]\times[n]}$ with a_{ik} in $\mathbb{R}^d$ and $b_{ik} \in \mathbb{R}$, LAD solves

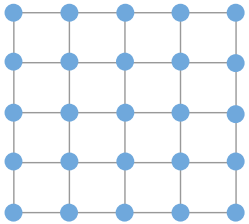
$$\min_{x\in\mathbb{R}^d}\left\{f(x) \coloneqq \frac{1}{m}\sum_{i=1}^{m}\frac{1}{n}\sum_{k=1}^{n}|a_{ik}^{\top}x - b_{ik}|\right\}$$

- $m = 64$, $n = 200$, $d = 20$
- Data are first generated from an underlying model and then some proportions are corrupted. This makes the problem more heterogeneous and thus enhances the importance of agent coordination.

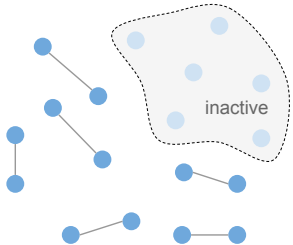


Network description

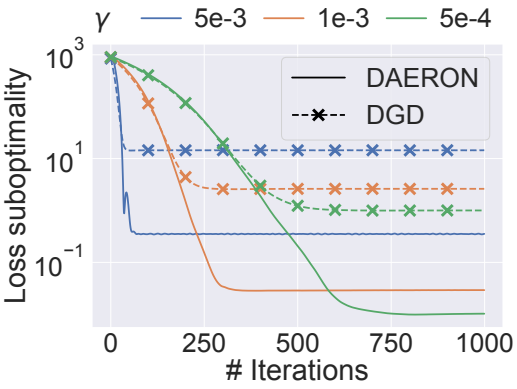
- **Static network:** Both the network composition and the topology (8×8 grid graph) are fixed over time. Adjacent nodes exchange gradients at each iteration.



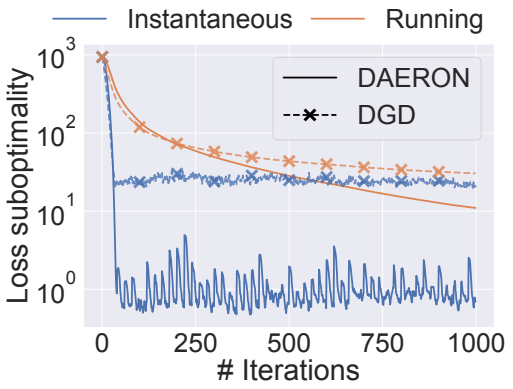
- **Open network:** Every 20 iterations, each agent changes its activation status with probability $p = 0.05$. Active nodes are randomly paired with each other at each iteration.



Results



Static network



Open network

DGD stands for Decentralized Subgradient Descent

Conclusion

- Paradigm shift: from closed network to open network
- Simple strategy may work: just aggregate your gradients
- New components need to be incorporated into the analysis: tools from online learning, time-varying optimization

H., Iutzeler, F., Malick, J., & Mertikopoulos, P. (2020). Multi-agent online optimization with delays: Asynchronicity, adaptivity, and optimism. arXiv preprint arXiv:2012.11579.