Learning from distributed datasets

An introduction with two examples

Pedro Guerreiro and João Xavier
Institute for Systems and Robotics, Instituto Superior Técnico
University of Lisbon

Mathematics, Physics & Machine Learning seminar
June 18, 2020

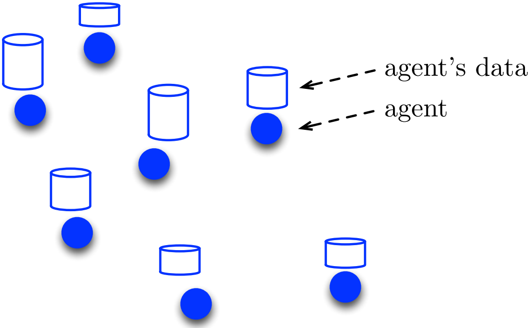
Outline

- What is distributed learning?
- Example 1: distributed convex learning with a roaming token
- Example 2: distributed classification with random meetings

What is distributed learning?

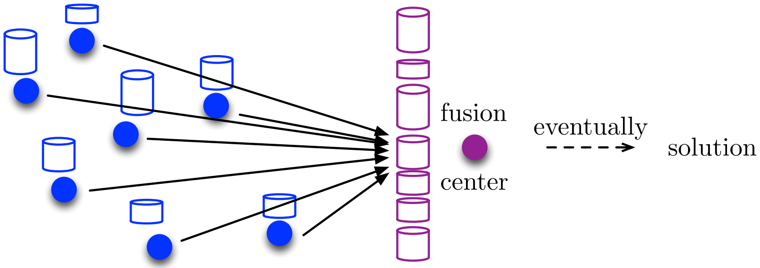
The physical setup

- a team of robots, a wireless sensor network, a swarm of drones, ...



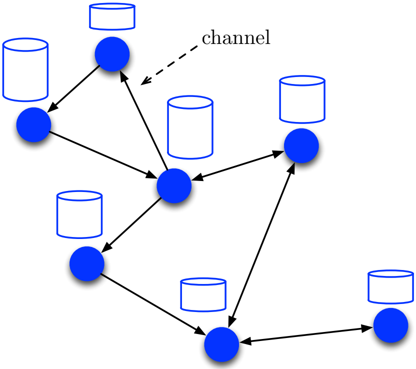
- agents are spatially distributed
- each agent measures data

Centralized learning



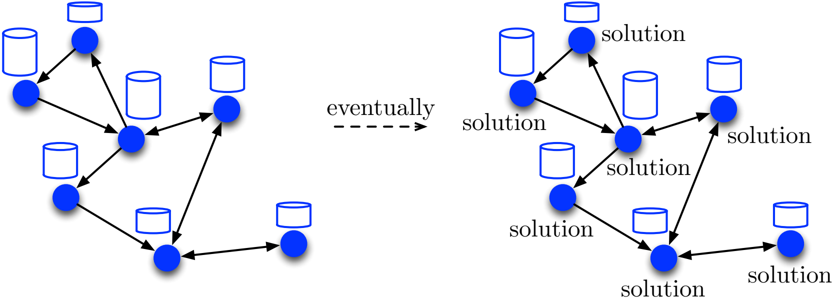
- each agent sends its data to a fusion center
- fusion center computes the solution
- drawbacks: single point of failure, traffic jams, lack of data privacy

Distributed learning



- agents are linked by channels
- a channel can appear and disappear (randomly)
- a channel might be directed

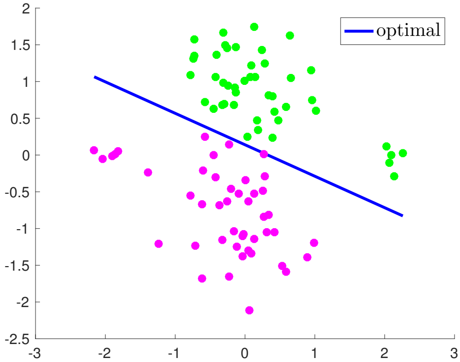
Distributed learning



- agents send messages through channels
- all agents obtain the solution

Example 1: distributed convex learning with a roaming token

- to illustrate, let the learning problem be logistic regression



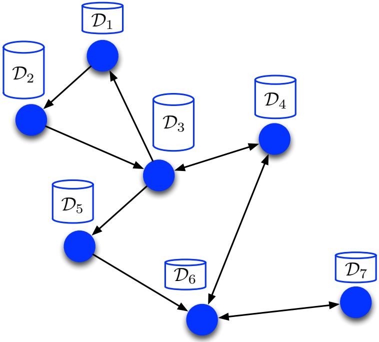
- $\mathcal{D} = \{(x_1, y_1), \ldots, (x_K, y_K)\}$ is dataset (x_k=features, $y_k = \pm 1$)

- goal is to learn the separating hyperplane

- goal is to learn

$$\theta^\star = \operatorname*{argmin}_\theta \underbrace{\log\left(1 + e^{-y_1\theta^T x_1}\right) + \cdots + \log\left(1 + e^{-y_K\theta^T x_K}\right) + \theta^T P\theta}_{f_\mathcal{D}(\theta)}$$

- ... but with dataset $\mathcal{D}$ split across the agents:

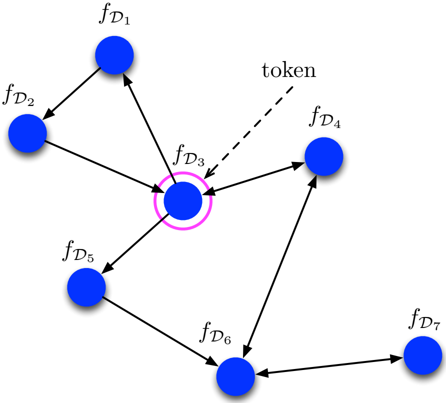


- objective function has the form

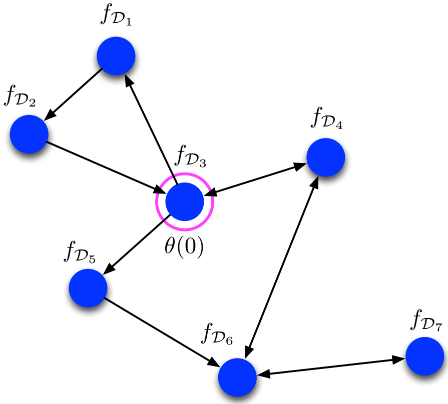
$$f_\mathcal{D} = f_{\mathcal{D}_1} + f_{\mathcal{D}_2} + f_{\mathcal{D}_3} + f_{\mathcal{D}_4} + f_{\mathcal{D}_5} + f_{\mathcal{D}_6} + f_{\mathcal{D}_7}$$

Learning with a random token

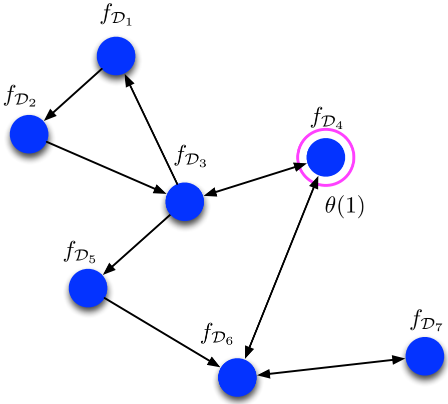
- goal is to learn $\theta^\star = \underset{\theta}{\text{argmin}}\ f_{\mathcal{D}_1}(\theta) + f_{\mathcal{D}_2}(\theta) + \cdots + f_{\mathcal{D}_N}(\theta)$



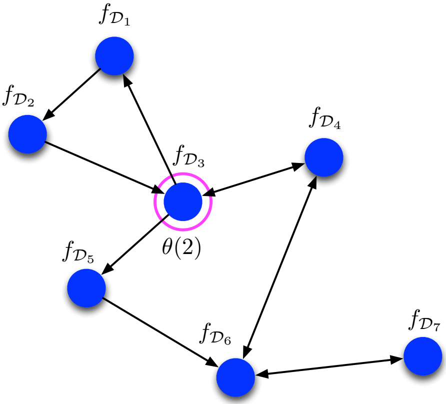
- the token carries a guess of $\theta^\star$
- the token moves randomly across the network
- when the token visits an agent, it updates its guess of $\theta^\star$



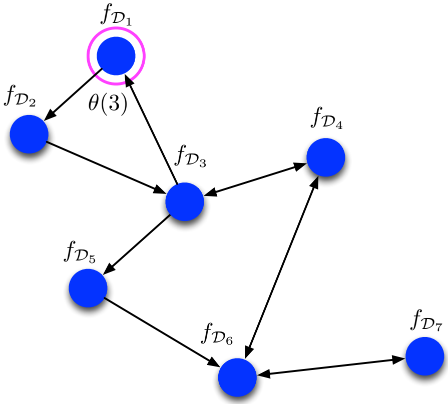
- token starts at agent 3 with some initialization $\theta(0)$



- token updates $\theta(1) = \theta(0) - \alpha \nabla f_{\mathcal{D}_4}(\theta(0))$



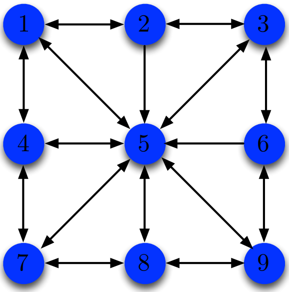
- token updates $\theta(2) = \theta(1) - \alpha \nabla f_{\mathcal{D}_3}(\theta(1))$



- token updates $\theta(3) = \theta(2) - \alpha \nabla f_{\mathcal{D}_1}(\theta(2))$

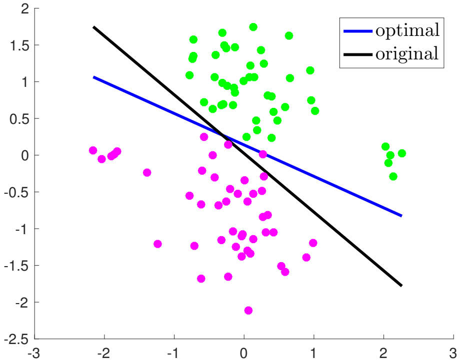
Does this scheme work?

- . . . no
- consider the network



- token chooses neighbor uniformly at random (current node included)

- scheme fails:



- it fails because it minimizes the wrong function:

$$\pi_1 f_{\mathcal{D}_1}(\theta) + \pi_2 f_{\mathcal{D}_2}(\theta) + \cdots + \pi_N f_{\mathcal{D}_N}(\theta)$$

- $(\pi_1, \pi_2, \ldots, \pi_N)$ is the stationary distribution of the random walk

- scheme is minimizing

$$\pi_1 f_{\mathcal{D}_1}(\theta) + \pi_2 f_{\mathcal{D}_2}(\theta) + \cdots + \pi_N f_{\mathcal{D}_N}(\theta),$$

  with $\pi_1, \ldots, \pi_N$ depending on the network topology

- what about unbiasing the scheme by redefining each local function as

$$f_{\mathcal{D}_n} \leftarrow \frac{1}{\pi_n} f_{\mathcal{D}_n}?$$

- this works, but each agent n would need to know π_n

- what about agent n guessing π_n on the fly?

Distributed learning algorithm

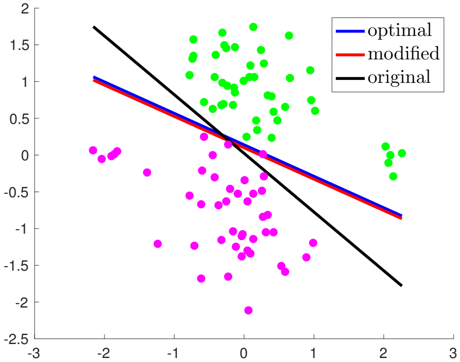
```
t = 0
initialize θ(0)
token starts at some node n(0)
repeat
    t ← t + 1
    token jumps to a random neighbor n(t)
    token updates the parameter
```

$$\theta(t) = \theta(t-1) - \alpha \frac{1}{\pi(n(t),t)} \nabla f_{\mathcal{D}_{n(t)}}\left(\theta(t-1)\right)$$

where $\pi(n,t)$ is fraction of visits to node n, by time t

until some stopping criterion is met

- the algorithm succeeds:



- we can prove: with probability one, (and under suitable convexity),

$$\liminf_{t\to\infty} f_{\mathcal{D}}\left(\theta(t)\right) \leq f_{\mathcal{D}}\left(\theta^{\star}\right) + \mathcal{O}\left(\alpha\right)$$

Blueprint of the proof

- fix an agent n

- define the stopping times $t_1, t_2, t_3, \ldots$, where t_k is (random) time of the kth visit to agent n, along with filtration $\mathcal{F}_{t_1}, \mathcal{F}_{t_2}, \mathcal{F}_{t_3}, \ldots$

- use the ergodic theorem for markov chains to show

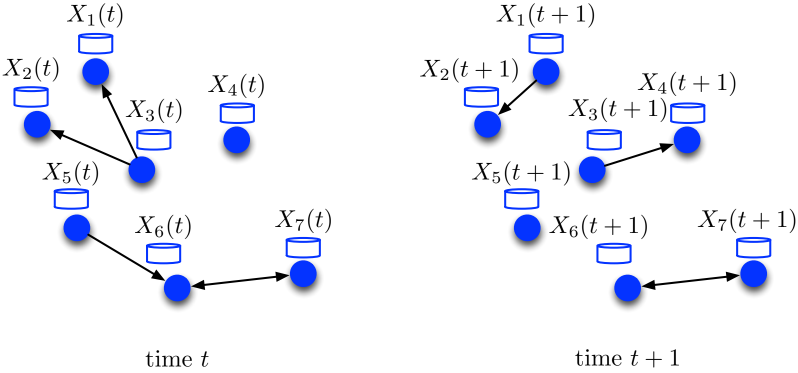
$$\mathbf{E}\left(\left\|\theta(t_{k+1}) - \theta^\star\right\|^2 \mid \mathcal{F}_{t_k}\right) \\ \leq \left\|\theta(t_k) - \theta^\star\right\|^2 - \alpha\left(f\left(\theta(t_k)\right) - f\left(\theta^\star\right)\right) + \mathcal{O}\left(\alpha^2\right)$$

- use martingale arguments to get

$$\liminf_{t\to\infty} f\left(\theta(t)\right) \leq f\left(\theta^\star\right) + \mathcal{O}\left(\alpha\right)$$

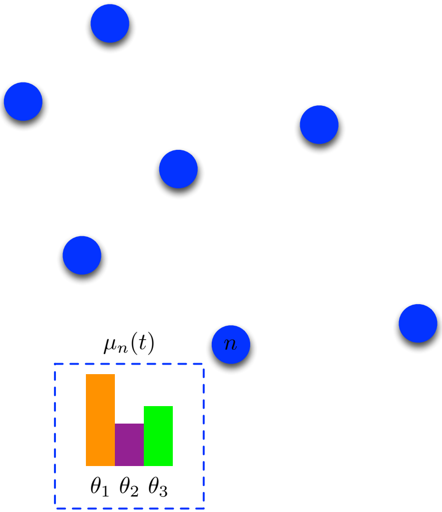
Example 2: distributed classification with random meetings

- each agent n observes a data stream $X_n(1), X_n(2), X_n(3), \ldots$
- streams are samples of unknown random source $\theta^\star \in \{\theta_1, \ldots, \theta_P\}$



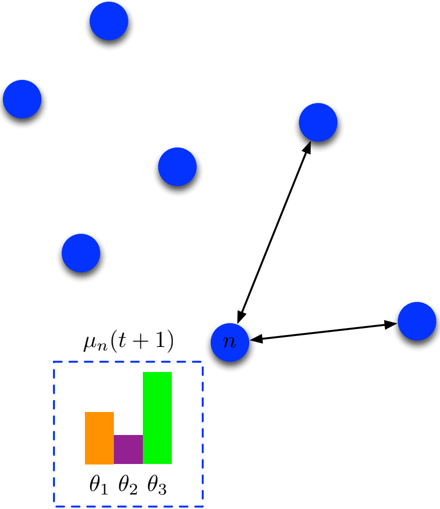
- channels change randomly over time
- agents want to learn which $\theta^\star$ is generating the streams

- we are at time t
- each agent n holds a belief $\mu_n(t) = \left(\mu_n^{\theta_1}(t), \mu_n^{\theta_2}(t), \mu_n^{\theta_3}(t)\right)$



- we move to time $t + 1$

- each agent n meets with available neighbors and updates its belief



Distributed learning algorithm

each agent n initialises $\mu_n(0) = \left(\mu_n^{\theta_1}(0), \ldots, \mu_n^{\theta_P}(0)\right)$ and sets $t = 0$

repeat

local Bayes update: each agent n observes $X_n(t)$ and updates

$$\mu_n^{\theta}(t+1/2) = \frac{P_n^{\theta}\left(X_n(t)\right)\mu_n^{\theta}(t)}{\sum_{\vartheta \in \Theta} P_n^{\vartheta}\left(X_n(t)\right)\mu_n^{\vartheta}(t)}, \quad \text{for } \theta \in \Theta$$

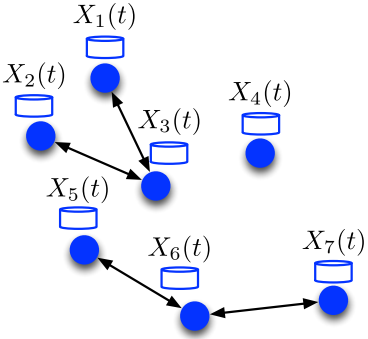
local fusion of beliefs: each agent n receives $\mu_m^{\theta}(t+1/2)$ from its neighbors m and updates

$$\mu_n^{\theta}(t+1) = \frac{\exp\left(\sum_{m=1}^{N} W_{nm}(t) \log \mu_m^{\theta}(t+1/2)\right)}{\sum_{\vartheta \in \Theta} \exp\left(\sum_{m=1}^{N} W_{nm}(t) \log \mu_m^{\vartheta}(t+1/2)\right)}, \quad \text{for } \theta \in \Theta$$

$t \leftarrow t+1$

until some stopping criterion is met

- weight $W_{nm}(t) \neq 0$ only if agent n is linked to agent m at time t
- example:



$$W(t) = \begin{bmatrix} * & 0 & * & 0 & 0 & 0 & 0 \\ 0 & * & * & 0 & 0 & 0 & 0 \\ * & * & * & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & * & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & * & * & 0 \\ 0 & 0 & 0 & 0 & * & * & * \\ 0 & 0 & 0 & 0 & 0 & * & * \end{bmatrix}$$

- assumptions on the random weight matrices $W(t)$:
  - $W_{nm}(t) \geq 0$
  - each row of $W(t)$ sums to one
  - the sequence $(W(t))_{t\geq 0}$ is i.i.d. and independent from $(X_n(t))_{t\geq 0}$
  - $\mathbf{E}(W(t))$ is irreducible and aperiodic

- we can prove: with probability one,

$$\lim_{t \to +\infty} \frac{1}{t} \log \frac{\mu_n^{\theta^\star}(t)}{\mu_n^{\theta}(t)} = \underbrace{\sum_{m=1}^{N} \mathbf{E}\left(\pi_m\right) D_{\mathsf{KL}}\left(P_m^{\theta^\star}, P_m^{\theta}\right)}_{K(\theta^\star, \theta)}, \quad \text{for } \theta \in \Theta$$

where $\pi = (\pi_1, \ldots, \pi_N) > 0$ comes from $\prod_{t=1}^{\infty} W(t) = \mathbf{1}\pi^T$

- interpretation: algorithm works

$$\mu_n^{\theta^\star}(t) \approx e^{K(\theta^\star, \theta)t} \mu_n^{\theta}(t) \text{ for large } t \quad \Rightarrow \quad \left\{ \begin{array}{l} \mu_n^{\theta^\star}(t) \to 1 \\ \mu_n^{\theta}(t) \to 0, \text{ for } \theta \neq \theta^\star \end{array} \right.$$

Blueprint of the proof

- fix $\theta \in \Theta$ and introduce the N dimensional vectors

$$u(t) := \begin{bmatrix} \log \frac{\mu_1^{\theta_\star}(t)}{\mu_1^{\theta}(t)} \\ \vdots \\ \log \frac{\mu_N^{\theta_\star}(t)}{\mu_N^{\theta}(t)} \end{bmatrix} \quad \text{and} \quad l(t) := \begin{bmatrix} \log \frac{P_1^{\theta_\star}(X_1(t))}{P_1^{\theta}(X_1(t))} \\ \vdots \\ \log \frac{P_N^{\theta_\star}(X_N(t))}{P_N^{\theta}(X_N(t))} \end{bmatrix}$$

- note that

$$\frac{1}{t} u(t) = \frac{1}{t} \Phi(t,1) u(0) + \frac{1}{t} \sum_{\tau=0}^{t-1} \Phi(t, t-\tau) l(t-\tau),$$

where

$$\Phi(t,s) = W(t) \cdots W(s)$$

- fix $\epsilon > 0$

- show that there exists a random time $T(t) \leq t$ such that

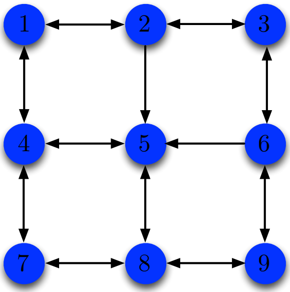
$$\frac{1}{t}\sum_{\tau=0}^{t-1}\Phi(t,t-\tau)l(t-\tau) = \mathbf{1}\left(\frac{1}{t}\sum_{\tau=T(t)}^{t-1} v(t-\tau)^T l(t-\tau)\right) + \mathcal{O}\left(\epsilon\right),$$

where $v(t-\tau)$ is random vector in $\prod_{s=t-\tau}^{\infty} W(s) = \mathbf{1}v(t-\tau)^T$

- use the ergodic theorem to get

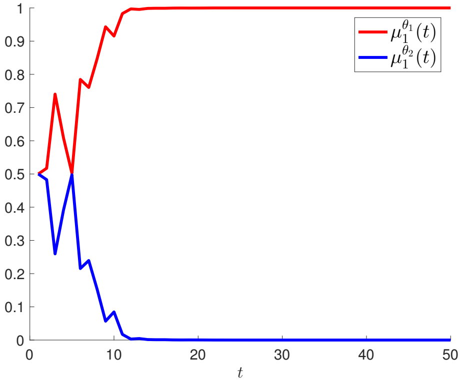
$$\frac{1}{t}\sum_{\tau=T(t)}^{t-1} v(t-\tau)^T l(t-\tau) \to K\left(\theta^{\star},\theta\right)$$

Numerical example

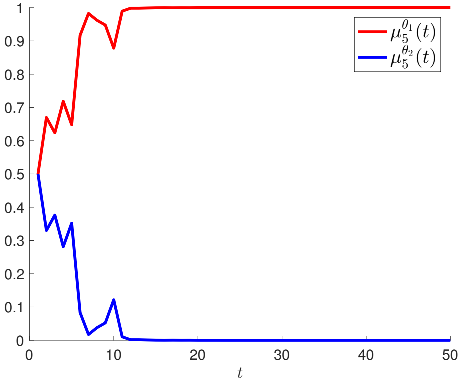


- $\Theta = \{\theta_1, \theta_2\}$ with θ_1 being the active one
- gaussian streams: $X_n(t) \sim \mathcal{N}\left(c_n^\theta, \left(\sigma_n^\theta\right)^2\right)$ for $\theta \in \Theta$
- each channel is (independently) active with probability $p = 0.7$

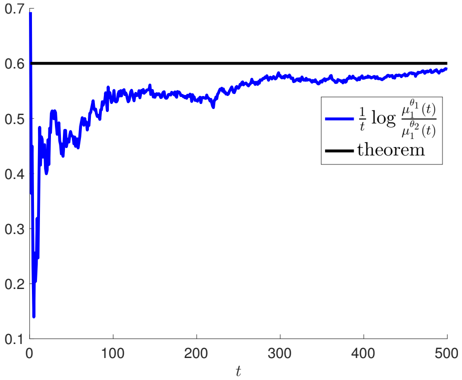
Belief $\mu_1(t) = \left(\mu_1^{\theta_1}(t), \mu_1^{\theta_2}(t)\right)$ at agent 1



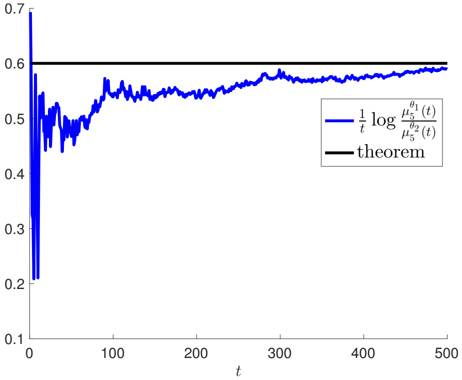
Belief $\mu_2(t) = \left(\mu_5^{\theta_1}(t), \mu_5^{\theta_2}(t)\right)$ at agent 5



Log beliefs ratio $\frac{1}{t}\log\frac{\mu_1^{\theta_1}(t)}{\mu_1^{\theta_2}(t)}$ at agent 1



Log beliefs ratio $\frac{1}{t} \log \frac{\mu_5^{\theta_1}(t)}{\mu_5^{\theta_2}(t)}$ at agent 5



References

- The algorithms on slides 19 and 26 and their proofs of convergence are in the PhD thesis “Distributed Algorithms for Collaborative Learning,” Pedro Guerreiro, Instituto Superior Técnico, Universidade de Lisboa, 2019.

- Algorithm on slide 19 is inspired from Björn Johansson, Maben Rabi, and Mikael Johansson, "A randomized incremental subgradient method for distributed optimization in networked systems," *SIAM Journal on Optimization* 20, no. 3 (2010): 1157-1170.

- Algorithm on slide 26 is inspired from Anusha Lalitha, Tara Javidi, and Anand D. Sarwate, "Social learning and distributed hypothesis testing," *IEEE Transactions on Information Theory* 64, no. 9 (2018): 6161-6179.

Thank you!