Federated Reinforcement Learning

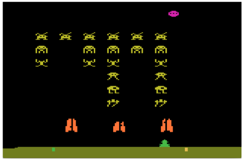
Paul Mangold, École Polytechnique

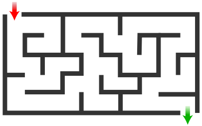
REDEEM Retreat @ Annecy, September 24th 2025

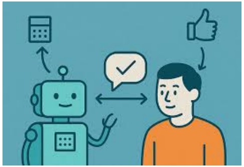
Refresher on Reinforcement Learning

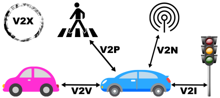
In RL, agent:

- take actions in an environment
- collect reward after their action
- learn to obtain better rewards





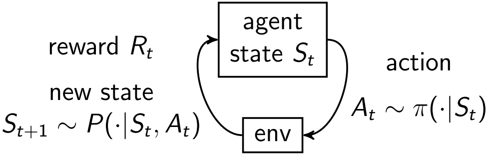




Refresher on Reinforcement Learning

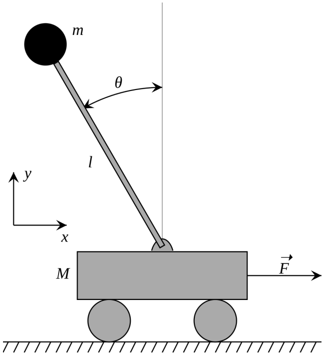
Environment:

- set of states $\mathcal{S}$
- set of actions $\mathcal{A}$
- rewards, typically in $[0, 1]$
- transition $P(\cdot|s, a)$ for $s, a \in \mathcal{S} \times \mathcal{A}$



Goal: learn π to get good rewards

Example: CartPole



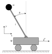
Goal: keep the stick up

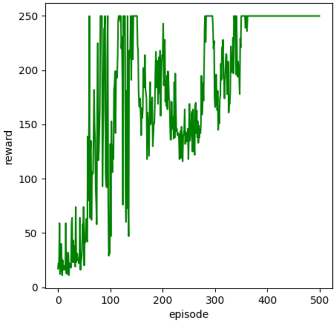
- state: angle of the stick
- reward: 1 if still up, 0 otherwise

Idea: run episodes of length $H = 250$
$\rightarrow$ adapt policy after each episode

Example: CartPole

Cumulative reward, 1 cart





Two Big Questions in Reinforcement Learning

1. **Policy evaluation**: evaluate if a policy is good
2. **Policy optimization**: find a good policy

Two Big Questions in Reinforcement Learning

1. **Policy evaluation**: evaluate if a policy is good

   take a policy π
   goal: approximate the expected sum of reward for each $s \in \mathcal{S}$

   $$V^\pi(s) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t) | S_0 = s\Big]$$

   where $A_t \sim \pi(\cdot|S_t)$ and $S_{t+1} \sim P(\cdot|S_t, A_t)$

2. **Policy optimization**: find a good policy

Two Big Questions in Reinforcement Learning

1. **Policy evaluation**: evaluate if a policy is good

2. **Policy optimization**: find a good policy

   find the best policy (according to value), for all $s \in \mathcal{S}$

$$\pi_\star(\cdot|s) \in \arg\max_\pi V^\pi(s)$$

1. Policy Evaluation: TD Learning

1. Policy Evaluation: TD Learning

State Value function:

$$V^{(\pi)}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t) \,\middle|\, S_0 = s\right]$$

1. Policy Evaluation: TD Learning

State Value function:

$$V^{(\pi)}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t) \;\middle|\; S_0 = s\right]$$

Expanding the first step, we obtain the Bellman equation:

$$\begin{aligned} V^{(\pi)}(s) &= \mathbb{E}[R(s, A_0)] + \gamma \sum_{a \in \mathcal{A}} \pi(a|s) \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R(S_t, A_t)\right] \\ &= \mathbb{E}[R(s, A_0)] + \gamma \sum_{a \in \mathcal{A}} \pi(a|s) \sum_{s' \in \mathcal{S}} P(s'|s, a) V^{(\pi)}(s') \end{aligned}$$

1. Policy Evaluation: TD Learning

The function $V^{(\pi)}$ satisfies the Bellman equation

$$V^{(\pi)} - R - \gamma P V^{(\pi)} = 0 \qquad (\star)$$

1. Policy Evaluation: TD Learning

The function $V^{(\pi)}$ satisfies the Bellman equation

$$V^{(\pi)} - R - \gamma P V^{(\pi)} = 0 \qquad (\star)$$

Temporal difference learning finds $V^{(\pi)}$ by solving this equation:

- take action $A_t \sim \pi(\cdot|S_t)$
- receive reward $R(S_t, A_t)$ and $S_{t+1} \sim P(\cdot|S_t, A_t)$
- update the current estimate $\hat{V}_t^{(\pi)}$ with the error from $(\star)$

$$\hat{V}_{t+1}^{(\pi)}(S_t) = \hat{V}_t^{(\pi)}(S_t) - \alpha(V_t^{(\pi)}(S_t) - R(S_t, A_t) + \gamma P V_t^{(\pi)}(S_t))$$

$\Rightarrow$ eventually, $\hat{V}_t^{(\pi)}$ converges to $V^{(\pi)}$

2. Policy Optimization: Policy Gradient Method

2. Policy Optimization: Policy Gradient Method

The value function is

$$V^{(\pi)}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t) \;\middle|\; S_0 = s\right]$$
$$= \sum_{t=0}^{\infty} \gamma^t \sum_{s \in \mathcal{S}} \sum_{a \in \mathcal{A}} \mathbb{P}(S_t = s, A_t = a)\mathbb{E}[R(S_t, A_t)]$$

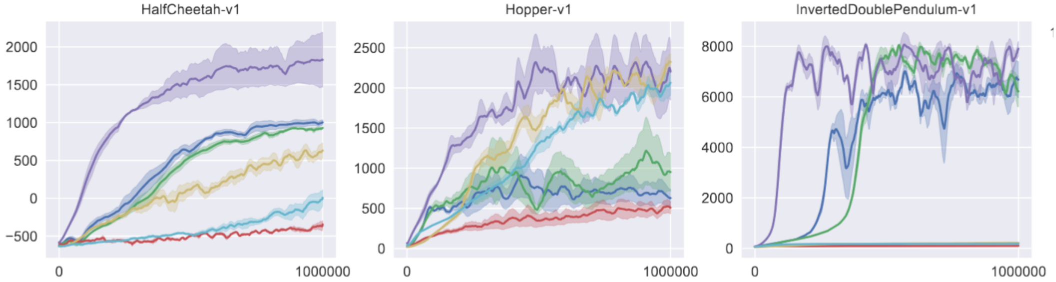
Parameterize the policy π_θ by $\theta \in \mathbb{R}^{SA}$, and update

$$\theta_{t+1} = \theta_t + \alpha \nabla_\theta V^{(\pi_\theta)}$$

$\Rightarrow$ the policy π_{θ_t} converges to an optimal policy $\pi_\star$

The problem of Reinforcement Learning:

All these methods require **a lot** of samples to converge

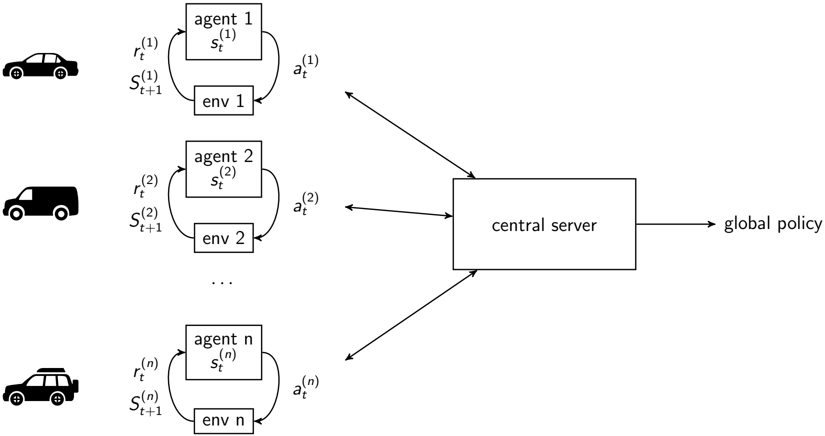


from John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017)

Federated Reinforcement Learning

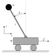
Federated Reinforcement Learning

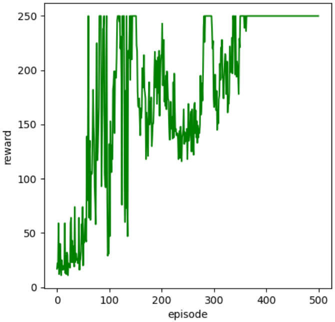
Idea: collaborate to solve these problems together **faster**

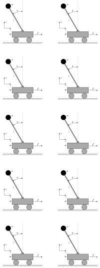


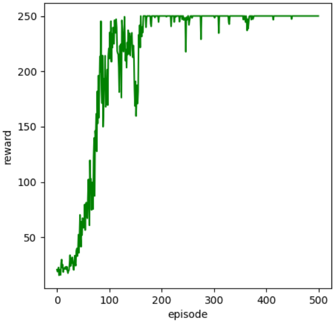
Example: CartPole

Cumulative reward, 1 cart vs. 10 carts









Question:

How does RL benefit from federated learning?

Question:

How does RL benefit from federated learning?

→ Can it accelerate the training?

→ How to handle heterogeneity?

→ How to reduce communications?

Heterogeneity in Reinforcement Learning

Take N agents with transition kernels $P^{(c)}$ and rewards $r^{(c)}$

Two types of heterogeneity, for $c \neq c' \in \{1, \ldots, N\}$

→ transition kernel heterogeneity:

$$\text{for } s, a, s' \in \mathcal{S} \times \mathcal{A} \times \mathcal{S},\ P^{(c)}(s'|s,a) \neq P^{(c')}(s'|s,a)$$

→ rewards heterogeneity

$$\text{for } s, a \in \mathcal{S} \times \mathcal{A} \times \mathcal{S},\ R^{(c)}(s,a) \neq R^{(c')}(s,a)$$

1. Federated Policy Evaluation

1. Federated Policy Evaluation

Federated temporal difference learning method, with shared policy π:

- for each agent $c = 1$ to N
  - take action $A_t^{(c)} \sim \pi(\cdot|S_t^{(c)})$
  - receive reward $R^{(c)}(S_t^{(c)}, A_t^{(c)})$ and $S_{t+1}^{(c)} \sim P^{(c)}(\cdot|S_t^{(c)}, A_t^{(c)})$
  - update the current estimate $\hat{V}_t^{(c,\pi)}$ with the error from $(\star)$

  $$\hat{V}_{t+1}^{(c,\pi)}(S_t^{(c)}) = \bar{V}_t^{(\pi)}(S_t^{(c)}) - \alpha(\bar{V}_t^{(\pi)}(S_t^{(c)}) - R^{(c)}(S_t^{(c)}, A_t^{(c)}) + \gamma P^{(c)} \bar{V}_t^{(\pi)}(S_t^{(c)}))$$
- aggregate $\bar{V}_{t+1}^{(\pi)} = \frac{1}{N} \sum_{c=1}^{N} \hat{V}_t^{(c,\pi)}$

1. Federated Policy Evaluation

Federated temporal difference learning method, with shared policy π:

- for each agent $c = 1$ to N
  - take action $A_t^{(c)} \sim \pi(\cdot|S_t^{(c)})$
  - receive reward $R^{(c)}(S_t^{(c)}, A_t^{(c)})$ and $S_{t+1}^{(c)} \sim P^{(c)}(\cdot|S_t^{(c)}, A_t^{(c)})$
  - update the current estimate $\hat{V}_t^{(c,\pi)}$ with the error from $(\star)$

  $$\hat{V}_{t+1}^{(c,\pi)}(S_t^{(c)}) = \bar{V}_t^{(\pi)}(S_t^{(c)}) - \alpha(\bar{V}_t^{(\pi)}(S_t^{(c)}) - R^{(c)}(S_t^{(c)}, A_t^{(c)}) + \gamma P^{(c)} \bar{V}_t^{(\pi)}(S_t^{(c)}))$$

- aggregate $\bar{V}_{t+1}^{(\pi)} = \frac{1}{N} \sum_{c=1}^{N} \hat{V}_t^{(c,\pi)}$

Theorem: this algorithm converges to a solution of

$$\bar{V}^{(\pi)} - \frac{1}{N} \sum_{c=1}^{N} R^{(c)} - \frac{1}{N} \sum_{c=1}^{N} \gamma P^{(c)} \bar{V}^{(\pi)} = 0$$

1. Federated Policy Evaluation

We show that this algorithm

1. converges even with local training
2. can benefit from control variate to mitigate heterogeneity drift
3. accelerates the learning (N times less samples per agent)

$\Rightarrow$ Problem: the solution to $\bar{V}^{(\pi)} - \frac{1}{N}\sum_{c=1}^{N} R^{(c)} - \frac{1}{N}\sum_{c=1}^{N} \gamma P^{(c)} \bar{V}^{(\pi)} = 0$

...may not be the right value function for each agent

...unless agents are similar enough!

2. Federated Policy Optimization

2. Federated Policy Optimization

What about federated policy gradient?

$$\theta_{t+1} = \theta_t + \frac{\alpha}{N} \sum_{c=1}^{N} \nabla_\theta V^{(c, \pi_{\theta_t})}$$

2. Federated Policy Optimization

What about federated policy gradient?

$$\theta_{t+1} = \theta_t + \frac{\alpha}{N} \sum_{c=1}^{N} \nabla_\theta V^{(c,\pi_{\theta_t})}$$

Some remarks about regularity: each $V^{(c,\pi_\theta)}$ is:

- L-smooth for some $L > 0$
- satisfies a non-uniform Łojasiewicz property for $\mu : \mathbb{R}^p \to \mathbb{R}$:

$$\|\nabla_\pi V^{(c,\pi_\theta)}\|^2 \geq 2\mu(\theta)\big(V^{(c,\star)} - V^{(c,\pi_\theta)}\big)^2 \qquad (\star)$$

Problem: due to heterogeneity, $\frac{1}{N}\sum_{c=1}^{N} V^{c,\pi_\theta)}$ does not satisfy $(\star)$

2. Federated Policy Optimization

With $\mu = \min_t \mu(\theta_t)$, we prove that

$$\frac{1}{N}\sum_{c=1}^{N} V^{(c,\star)} - \mathbb{E}V^{(c,\pi_t)} \lesssim \frac{L}{\mu T}\frac{1}{N}\sum_{c=1}^{N}(V^{(c,\star)} - V^{(c,\pi_{\theta_0})}) + \frac{\eta^{1/2}}{\mu^{1/2}N^{1/2}} + \frac{\zeta^{1/2}}{\mu^{1/2}}$$

where $\zeta \neq 0$ if agents are heterogeneous

On the Impact of Heterogeneity on Federated RL

We can measure heterogeneity by

- $\rightarrow$ transition heterogeneity: $\epsilon_P = \sup_{c \neq c', s, a \in \mathcal{S} \times \mathcal{A}} \| P^{(c)}(\cdot|s,a) - P^{(c')}(\cdot|s,a) \|_{TV}$
- $\rightarrow$ rewards heterogeneity $\epsilon_r = \sup_{c \neq c', s, a \in \mathcal{S} \times \mathcal{A}} |R^{(c)}(s,a) - R^{(c')}(s,a)|$

Federated error is always of order $\epsilon_P + \epsilon_r$

This is due to the fact that objectives are fundamentally mis-aligned

Conclusion

Federated reinforcement learning is still at its beginning

In this talk, we studied

- a federated TD learning algorithm
- a federated policy gradient algorithm

Contrary to classical FL, there is no “analogy with centralized”
$\rightarrow$ we necessarily pay heterogeneity somewhere...

Perspectives

Contrary to classical FL, there is no “analogy with centralized”
$\rightarrow$ we necessarily pay heterogeneity somewhere...

But there is hope:

- in homogeneous cases, everything works
- under heterogeneity... we should personalize!

In fact, maybe it is the same in all federated learning :)

Thank you!

Works related to this talk:

- Safwan Labbi et al. "On Global Convergence Rates for Federated Policy Gradient under Heterogeneous Environment". In: *arXiv* (2025)
- Safwan Labbi et al. "Federated ucbvi: Communication-efficient federated regret minimization with heterogeneous agents". In: *AISTATS* (2024)
- Lorenzo Mancini et al. "Joint Channel Selection using FedDRL in V2X". In: *MECOM*. 2024
- Paul Mangold et al. "Scafflsa: Taming heterogeneity in federated linear stochastic approximation and td learning". In: *NeurIPS* (2024)

Thanks to my collaborators on these projects:

Safwan Labbi, Lorenzo Mancini, Eric Moulines, Daniil Tiapkin