Decentralized Stochastic Approximation, Optimization, and Multi-Agent Reinforcement Learning

Justin Romberg, Georgia Tech ECE

CAMDA/TAMIDS Seminar, Texas A& M

~~College Station, Texas~~ Streaming live from Atlanta, Georgia

~~March 16, 2020~~ October 30, 2020

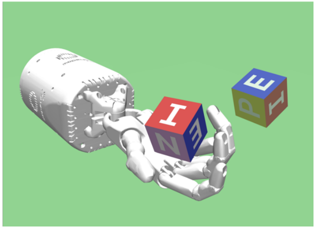
Collaborators

Thinh Doan
Virginia Tech, ECE

Siva Theja Maguluri
Georgia Tech, ISyE

Sihan Zeng
Georgia Tech, ECE

Reinforcement Learning



Ingredients for Distributed RL

Distributed RL is a combination of:

- stochastic approximation
- Markov decision processes
- function representation
- network consensus

Ingredients for Distributed RL

Distributed RL is a combination of:

- stochastic approximation
- Markov decision processes
- function representation
- network consensus
- (complicated probabilistic analysis)

Ingredients for Distributed RL

Distributed RL is a combination of:

- **stochastic approximation**
- Markov decision processes
- function representation
- network consensus
- (complicated probabilistic analysis)

Fixed point iterations

Classical result (Banach fixed point theorem): when $H(\cdot) : \mathbb{R}^N \to \mathbb{R}^N$ is a contraction

$$\|H(\boldsymbol{u}) - H(\boldsymbol{v})\| \ \leq \ \delta\|\boldsymbol{u} - \boldsymbol{v}\|, \quad \delta < 1,$$

then there is a unique **fixed point** $x^\star$ such that

$$\boldsymbol{x}^\star = H(\boldsymbol{x}^\star),$$

and the iteration

$$\boldsymbol{x}_{k+1} = H(\boldsymbol{x}_k),$$

finds it

$$\lim_{k\to\infty} \boldsymbol{x}_k = \boldsymbol{x}^\star.$$

Easy proof

Choose any point $\boldsymbol{x}_0$, then take

$$\boldsymbol{x}_{k+1} = H(\boldsymbol{x}_k)$$

so

$$\begin{aligned} \boldsymbol{x}_{k+1} - \boldsymbol{x}^\star &= H(\boldsymbol{x}_k) - \boldsymbol{x}^\star \\ &= H(\boldsymbol{x}_k) - H(\boldsymbol{x}^\star) \end{aligned}$$

and

$$\begin{aligned} \|\boldsymbol{x}_{k+1} - \boldsymbol{x}^\star\| &= \|H(\boldsymbol{x}_k) - H(\boldsymbol{x}^\star)\| \\ &\leq \delta \|\boldsymbol{x}_k - \boldsymbol{x}^\star\| \\ &\leq \delta^{k+1} \|\boldsymbol{x}_0 - \boldsymbol{x}^\star\|, \end{aligned}$$

so the convergence is **geometric**

Relationship to optimization

Choose any point $\boldsymbol{x}_0$, then take

$$\boldsymbol{x}_{k+1} = H(\boldsymbol{x}_k),$$

then

$$\begin{aligned}\|\boldsymbol{x}_{k+1} - \boldsymbol{x}^\star\| &= \|H(\boldsymbol{x}_k) - H(\boldsymbol{x}^\star)\| \\ &\leq \delta^{k+1}\|\boldsymbol{x}_0 - \boldsymbol{x}^\star\|,\end{aligned}$$

Gradient descent takes

$$H(\boldsymbol{x}) = \boldsymbol{x} - \alpha\nabla f(\boldsymbol{x})$$

for some differentiable f.

Fixed point iterations: Variation

Take

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha(H(\boldsymbol{x}_k) - \boldsymbol{x}_k), \quad 0 < \alpha \leq 1.$$

(More conservative, convex combination of new iterate and old.)

Then again

$$\boldsymbol{x}_{k+1} = (1-\alpha)\boldsymbol{x}_k + \alpha H(\boldsymbol{x}_k)$$

and

$$\begin{aligned} \|\boldsymbol{x}_{k+1} - \boldsymbol{x}^\star\| &\leq (1-\alpha)\|\boldsymbol{x}_k - \boldsymbol{x}^\star\| + \alpha\|H(\boldsymbol{x}_k) - H(\boldsymbol{x}^\star)\| \\ &\leq (1 - \alpha - \delta\alpha)\|\boldsymbol{x}_k - \boldsymbol{x}^\star\|. \end{aligned}$$

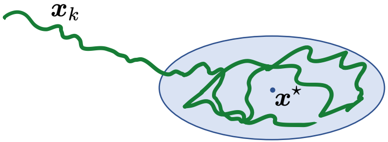
Still converge, albeit a little more slowly for $\alpha < 1$.

What if there is noise?

If our observations of $H(\cdot)$ are *noisy*,

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha\left(H(\boldsymbol{x}_k) - \boldsymbol{x}_k + \boldsymbol{\eta}_k\right), \quad \mathrm{E}[\boldsymbol{\eta}_k] = \mathbf{0},$$

then we don't get convergence for fixed α,



but we do converge to a "ball" around at a geometric rate

Stochastic approximation

If our observations of $H(\cdot)$ are *noisy*,

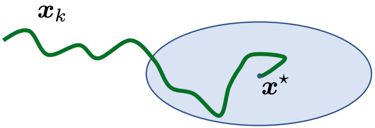
$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \left(H(\boldsymbol{x}_k) - \boldsymbol{x}_k + \boldsymbol{\eta}_k \right), \quad \mathrm{E}[\boldsymbol{\eta}_k] = \mathbf{0},$$

then we need to take $\alpha_k \to 0$ as we approach the solution.

If we take $\{\alpha_k\}$ such that

$$\sum_{k=0}^{\infty} \alpha_k^2 < \infty, \qquad \sum_{k=0}^{\infty} \alpha_k = \infty$$

then we so get (much slower) convergence



Example: $\alpha_k = C/(k+1)$

Ingredients for Distributed RL

Distributed RL is a combination of:

- stochastic approximation
- **Markov decision processes**
- function representation
- network consensus
- (complicated probabilistic analysis)

Markov decision process

At time t,

1. An agent finds itself in a **state** $\boldsymbol{s}_t$
2. It takes action $\boldsymbol{a}_t = \mu(\boldsymbol{s}_t)$
3. It moves to state $\boldsymbol{s}_{t+1}$ according to $\mathrm{P}\left(\boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_t\right)$...
4. ... and receives reward $R(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1})$.

Markov decision process

At time t,

1. An agent finds itself in a **state** $\boldsymbol{s}_t$
2. It takes action $\boldsymbol{a}_t = \mu(\boldsymbol{s}_t)$
3. It moves to state $\boldsymbol{s}_{t+1}$ according to $\mathrm{P}\left(\boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_t\right)$...
4. ... and receives reward $R(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1})$.

Long-term reward of **policy** μ:

$$V_\mu(\boldsymbol{s}) = \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t R(\boldsymbol{s}_t, \mu(\boldsymbol{s}_t), \boldsymbol{s}_{t+1}) \;\middle|\; \boldsymbol{s}_0 = \boldsymbol{s}\right]$$

Markov decision process

At time t,

1. An agent finds itself in a **state** $\boldsymbol{s}_t$
2. It takes action $\boldsymbol{a}_t = \mu(\boldsymbol{s}_t)$
3. It moves to state $\boldsymbol{s}_{t+1}$ according to $\mathrm{P}\left(\boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_t\right)$...
4. ... and receives reward $R(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1})$.

Bellman equation: V_μ obeys

$$V_\mu(\boldsymbol{s}) = \underbrace{\sum_{\boldsymbol{z} \in \mathcal{S}} \mathrm{P}\left(\boldsymbol{z} | \boldsymbol{s}, \mu(\boldsymbol{s})\right) \left[R(\boldsymbol{s}, \mu(\boldsymbol{s}), \boldsymbol{z}) + \gamma V_\mu(\boldsymbol{z})\right]}_{\boldsymbol{b}_\mu + \gamma \boldsymbol{P}_\mu \boldsymbol{V}_\mu}$$

This is a *fixed point equation* for V_μ

Markov decision process

At time t,

1. An agent finds itself in a **state** $\boldsymbol{s}_t$
2. It takes action $\boldsymbol{a}_t = \mu(\boldsymbol{s}_t)$
3. It moves to state $\boldsymbol{s}_{t+1}$ according to $\mathrm{P}\left(\boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_t\right)$...
4. ... and receives reward $R(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1})$.

State-action value function (Q function):

$$Q_\mu(\boldsymbol{s}, \boldsymbol{a}) = \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t R(\boldsymbol{s}_t, \mu(\boldsymbol{s}_t) \boldsymbol{s}_{t+1}) \;\middle|\; \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}\right]$$

Markov decision process

At time t,

1. An agent finds itself in a **state** $\boldsymbol{s}_t$
2. It takes action $\boldsymbol{a}_t = \mu(\boldsymbol{s}_t)$
3. It moves to state $\boldsymbol{s}_{t+1}$ according to $\mathrm{P}\left(\boldsymbol{s}_{t+1} | \boldsymbol{s}_t, \boldsymbol{a}_t\right)$...
4. ... and receives reward $R(\boldsymbol{s}_t, \boldsymbol{a}_t, \boldsymbol{s}_{t+1})$.

State-action value for the optimal policy obeys

$$Q^\star(\boldsymbol{s}, \boldsymbol{a}) = \mathrm{E}\left[R(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s}') + \gamma \max_{a'} Q^\star(\boldsymbol{s}', \boldsymbol{a}') \;\middle|\; \boldsymbol{s}_0 = \boldsymbol{s}, \boldsymbol{a}_0 = \boldsymbol{a}\right]$$

and we take $\mu^\star(\boldsymbol{s}) = \arg\max_{\boldsymbol{a}} Q^\star(\boldsymbol{s}, \boldsymbol{a})$ ...

... this is another *fixed point equation*

Stochastic approximation for policy evaluation

Fixed point iteration for finding $V_\mu(\boldsymbol{s})$:

$$V_{t+1}(\boldsymbol{s}) = V_t(\boldsymbol{s}) + \alpha \underbrace{\left(\sum_{\boldsymbol{z}} \mathrm{P}\left(\boldsymbol{z}|\boldsymbol{s}\right)\left[R(\boldsymbol{s},\boldsymbol{z}) + \gamma V_t(\boldsymbol{z})\right] - V_t(\boldsymbol{s})\right)}_{H(\boldsymbol{V}_t) - \boldsymbol{V}_t}$$

Stochastic approximation for policy evaluation

Fixed point iteration for finding $V_\mu(\boldsymbol{s})$:

$$V_{t+1}(\boldsymbol{s}) = V_t(\boldsymbol{s}) + \alpha \underbrace{\left(\sum_{\boldsymbol{z}} \mathrm{P}\left(\boldsymbol{z}|\boldsymbol{s}\right) \left[R(\boldsymbol{s},\boldsymbol{z}) + \gamma V_t(\boldsymbol{z}) \right] - V_t(\boldsymbol{s}) \right)}_{H(\boldsymbol{V}_t) - \boldsymbol{V}_t}$$

In practice, we don't have the model $\mathrm{P}\left(\boldsymbol{z}|\boldsymbol{s}\right)$, only observed data $\{(\boldsymbol{s}_t, \boldsymbol{s}_{t+1})\}$

Stochastic approximation for policy evaluation

Fixed point iteration for finding $V_\mu(\boldsymbol{s})$:

$$V_{t+1}(\boldsymbol{s}) = V_t(\boldsymbol{s}) + \alpha \underbrace{\left(\sum_{\boldsymbol{z}} \mathrm{P}\left(\boldsymbol{z}|\boldsymbol{s}\right) \left[R(\boldsymbol{s}, \boldsymbol{z}) + \gamma V_t(\boldsymbol{z})\right] - V_t(\boldsymbol{s}) \right)}_{H(\boldsymbol{V}_t) - \boldsymbol{V}_t}$$

Stochastic approximation iteration

$$V_{t+1}(\boldsymbol{s}_t) = V_t(\boldsymbol{s}_t) + \alpha_t \left(R(\boldsymbol{s}_t, \boldsymbol{s}_{t+1}) + \gamma V_t(\boldsymbol{s}_{t+1}) - V_t(\boldsymbol{s}_t) \right)$$

The “noise” is that $\boldsymbol{s}_{t+1}$ is sampled, rather than averaged over

Stochastic approximation for policy evaluation

Fixed point iteration for finding $V_\mu(\boldsymbol{s})$:

$$V_{t+1}(\boldsymbol{s}) = V_t(\boldsymbol{s}) + \alpha \underbrace{\left(\sum_{\boldsymbol{z}} \mathrm{P}\left(\boldsymbol{z}|\boldsymbol{s}\right) \left[R(\boldsymbol{s}, \boldsymbol{z}) + \gamma V_t(\boldsymbol{z}) \right] - V_t(\boldsymbol{s}) \right)}_{H(\boldsymbol{V}_t) - \boldsymbol{V}_t}$$

Stochastic approximation iteration

$$V_{t+1}(\boldsymbol{s}_t) = V_t(\boldsymbol{s}_t) + \alpha_t \underbrace{\left(R(\boldsymbol{s}_t, \boldsymbol{s}_{t+1}) + \gamma V_t(\boldsymbol{s}_{t+1}) - V_t(\boldsymbol{s}_t) \right)}_{H(\boldsymbol{V}_t) - \boldsymbol{V}_t + \boldsymbol{\eta}_t}$$

The "noise" is that $\boldsymbol{s}_{t+1}$ is sampled, rather than averaged over

Stochastic approximation for policy evaluation

Fixed point iteration for finding $V_\mu(\boldsymbol{s})$:

$$V_{t+1}(\boldsymbol{s}) = V_t(\boldsymbol{s}) + \alpha \underbrace{\left(\sum_{\boldsymbol{z}} \mathrm{P}\left(\boldsymbol{z}|\boldsymbol{s}\right) \left[R(\boldsymbol{s},\boldsymbol{z}) + \gamma V_t(\boldsymbol{z})\right] - V_t(\boldsymbol{s}) \right)}_{H(\boldsymbol{V}_t) - \boldsymbol{V}_t}$$

Stochastic approximation iteration

$$V_{t+1}(\boldsymbol{s}_t) = V_t(\boldsymbol{s}_t) + \alpha_t \underbrace{\left(R(\boldsymbol{s}_t, \boldsymbol{s}_{t+1}) + \gamma V_t(\boldsymbol{s}_{t+1}) - V_t(\boldsymbol{s}_t)\right)}_{H(\boldsymbol{V}_t) - \boldsymbol{V}_t + \boldsymbol{\eta}_t}$$

The "noise" is that $\boldsymbol{s}_{t+1}$ is sampled, rather than averaged over

This is different from stochastic gradient descent, since $H(\cdot)$ is in general not a gradient map

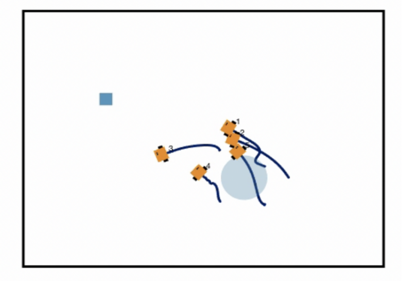
Ingredients for Distributed RL

Distributed RL is a combination of:

- stochastic approximation
- Markov decision processes
- **function representation**
- network consensus
- (complicated probabilistic analysis)

Function approximation

State space can be large (or even infinite) ...



... we need a natural way to parameterize/simplify

Linear function approximation

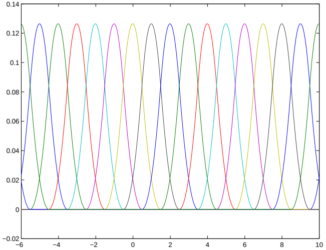
Simple (but powerful) model: linear representation

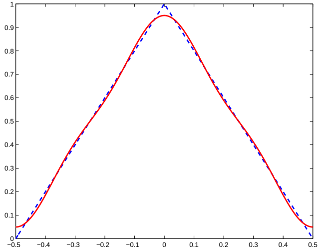
$$V(\boldsymbol{s};\boldsymbol{\theta}) = \sum_{k=1}^{K} \theta_k \phi_k(\boldsymbol{s}) = \boldsymbol{\phi}(\boldsymbol{s})^{\mathrm{T}}\boldsymbol{\theta}, \quad \boldsymbol{\phi}(\boldsymbol{s}) = \begin{bmatrix} \phi_1(\boldsymbol{s}) \\ \vdots \\ \phi_K(\boldsymbol{s}) \end{bmatrix}$$

Linear function approximation

Simple (but powerful) model: linear representation

$$V(\boldsymbol{s};\boldsymbol{\theta}) = \sum_{k=1}^{K} \theta_k \phi_k(\boldsymbol{s}) = \boldsymbol{\phi}(\boldsymbol{s})^{\mathrm{T}}\boldsymbol{\theta}, \quad \boldsymbol{\phi}(\boldsymbol{s}) = \begin{bmatrix} \phi_1(\boldsymbol{s}) \\ \vdots \\ \phi_K(\boldsymbol{s}) \end{bmatrix}$$





Policy evaluation with function approximation

Bellman equation:

$$V(\boldsymbol{s}) = \sum_{\boldsymbol{z} \in \mathcal{S}} \mathrm{P}\left(\boldsymbol{z} | \boldsymbol{s}\right) \left[R(\boldsymbol{s}, \mu(\boldsymbol{s}), \boldsymbol{z}) + \gamma V(\boldsymbol{z})\right]$$

Linear approximation:

$$V(\boldsymbol{s}; \boldsymbol{\theta}) = \sum_{k=1}^{K} \theta_k \phi_k(\boldsymbol{s}) = \boldsymbol{\phi}(\boldsymbol{s})^{\mathrm{T}} \boldsymbol{\theta}$$

These can conflict ....

Policy evaluation with function approximation

Bellman equation:

$$V(\boldsymbol{s}) = \sum_{\boldsymbol{z}\in\mathcal{S}} \mathrm{P}\left(\boldsymbol{z}|\boldsymbol{s}\right)\left[R(\boldsymbol{s},\mu(\boldsymbol{s}),\boldsymbol{z}) + \gamma V(\boldsymbol{z})\right]$$

Linear approximation:

$$V(\boldsymbol{s};\boldsymbol{\theta}) = \sum_{k=1}^{K} \theta_k \phi_k(\boldsymbol{s}) = \boldsymbol{\phi}(\boldsymbol{s})^{\mathrm{T}}\boldsymbol{\theta}$$

These can conflict ....

... but the following iterations

$$\begin{aligned}\boldsymbol{\theta}_{t+1} &= \boldsymbol{\theta}_t + \alpha_t\left(R(\boldsymbol{s}_t,\boldsymbol{s}_{t+1}) + \gamma V(\boldsymbol{s}_{t+1};\boldsymbol{\theta}_t) - V(\boldsymbol{s}_t;\boldsymbol{\theta}_t)\right)\nabla_\theta V(\boldsymbol{s}_t,\boldsymbol{\theta}_t)\\ &= \boldsymbol{\theta}_t + \alpha_t\left(R(\boldsymbol{s}_t,\boldsymbol{s}_{t+1}) + \gamma\boldsymbol{\phi}(\boldsymbol{s}_{t+1})^{\mathrm{T}}\boldsymbol{\theta}_t - \boldsymbol{\phi}(\boldsymbol{s}_t)^{\mathrm{T}}\boldsymbol{\theta}_t\right)\boldsymbol{\phi}(\boldsymbol{s}_t)\end{aligned}$$

converge to a "near optimal" $\boldsymbol{\theta}^\star$

Tsitsiklis and Roy, '97

Ingredients for Distributed RL

Distributed RL is a combination of:

- stochastic approximation
- Markov decision processes
- function representation
- **network consensus**
- (complicated probabilistic analysis)

Network consensus

- Each node in a network has a number $x(i)$
- We want each node to agree on the average

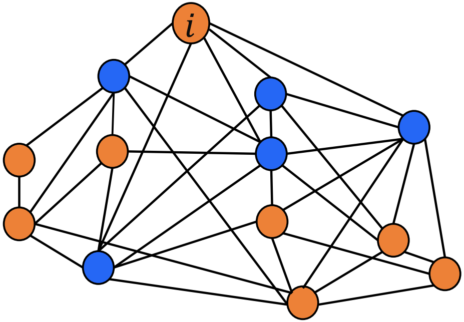
$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x(i) = \mathbf{1}^{\mathrm{T}}\boldsymbol{x}$$

Node i communicates with its neighbors $\mathcal{N}_i$

- Iterate, take $\boldsymbol{v}_0 = \boldsymbol{x}$, then

$$v_{k+1}(i) = \sum_{j\in\mathcal{N}_i} W_{ij} v_k(i)$$

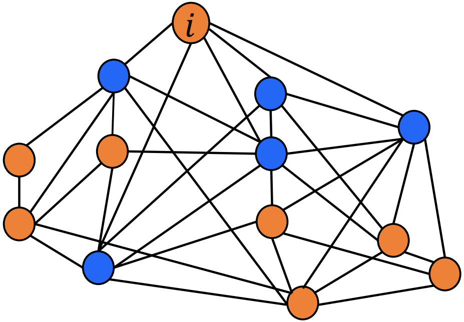
$$\boldsymbol{v}_{k+1} = \boldsymbol{W}\boldsymbol{v}_k, \quad \boldsymbol{W} \text{ doubly stochastic}$$



Network consensus convergence

Nodes reach "consensus" quickly:

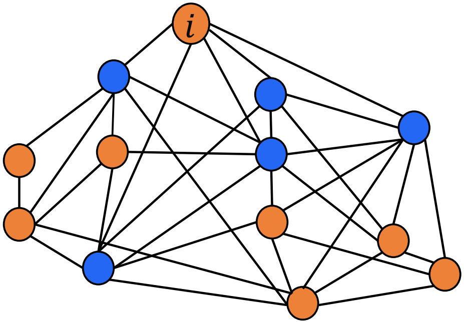
$$
\begin{aligned}
\boldsymbol{v}_{k+1} &= \boldsymbol{W}\boldsymbol{v}_k \\
\boldsymbol{v}_{k+1} - \bar{x}\mathbf{1} &= \boldsymbol{W}\boldsymbol{v}_k - \bar{x}\mathbf{1} \\
&= \boldsymbol{W}(\boldsymbol{v}_k - \bar{x}\mathbf{1}) \\
\|\boldsymbol{v}_{k+1} - \bar{x}\mathbf{1}\| &= \|\boldsymbol{W}(\boldsymbol{v}_k - \bar{x}\mathbf{1})\|
\end{aligned}
$$



Network consensus convergence

Nodes reach "consensus" quickly:

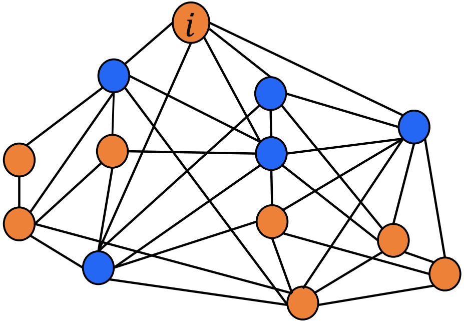
$$
\begin{aligned}
\boldsymbol{v}_{k+1} &= \boldsymbol{W}\boldsymbol{v}_k \\
\boldsymbol{v}_{k+1} - \bar{x}\mathbf{1} &= \boldsymbol{W}\boldsymbol{v}_k - \bar{x}\mathbf{1} \\
&= \boldsymbol{W}(\boldsymbol{v}_k - \bar{x}\mathbf{1}) \\
\|\boldsymbol{v}_{k+1} - \bar{x}\mathbf{1}\| &= \|\boldsymbol{W}(\boldsymbol{v}_k - \bar{x}\mathbf{1})\| \\
&\leq \sigma_2 \|\boldsymbol{v}_k - \bar{x}\mathbf{1}\| \\
&\leq \sigma_2^{k+1} \|\boldsymbol{v}_0 - \bar{x}\mathbf{1}\|
\end{aligned}
$$

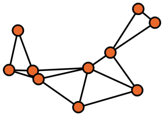


Network consensus convergence

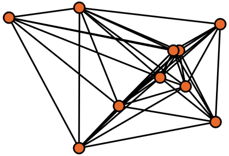
Nodes reach "consensus" quickly:

$$
\begin{aligned}
\boldsymbol{v}_{k+1} &= \boldsymbol{W}\boldsymbol{v}_k \\
\boldsymbol{v}_{k+1} - \bar{x}\mathbf{1} &= \boldsymbol{W}\boldsymbol{v}_k - \bar{x}\mathbf{1} \\
&= \boldsymbol{W}(\boldsymbol{v}_k - \bar{x}\mathbf{1}) \\
\|\boldsymbol{v}_{k+1} - \bar{x}\mathbf{1}\| &= \|\boldsymbol{W}(\boldsymbol{v}_k - \bar{x}\mathbf{1})\| \\
&\leq \sigma_2\|\boldsymbol{v}_k - \bar{x}\mathbf{1}\| \\
&\leq \sigma_2^{k+1}\,\|\boldsymbol{v}_0 - \bar{x}\mathbf{1}\|
\end{aligned}
$$





σ larger

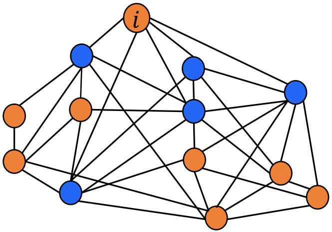


σ smaller

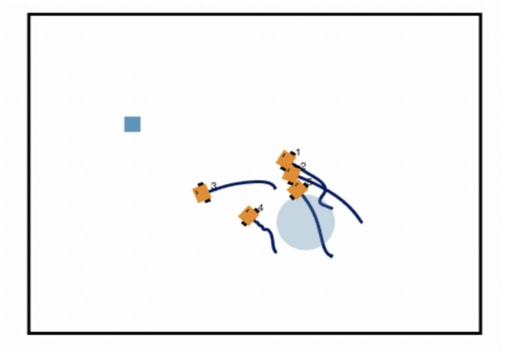
Multi-agent Reinforcement Learning, Scenario 1:
Multiple agents in a single environment, common state, different rewards

What is the value of a particular policy?

Multi-agent reinforcement learning



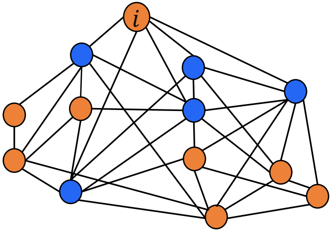
- N agents, communicating on a network
- One environment, common state $\boldsymbol{s}_t \in \mathcal{S}$
  transition probabilities $\mathrm{P}\left(\boldsymbol{s}_{t+1} | \boldsymbol{s}_t\right)$
- Individual actions $\boldsymbol{a}_t^i \in \mathcal{A}^i$
- Individual rewards $R^i(\boldsymbol{s}_t, \boldsymbol{s}_{t+1})$
- evaluate policies $\mu^i : \mathcal{S} \to \mathcal{A}^i$

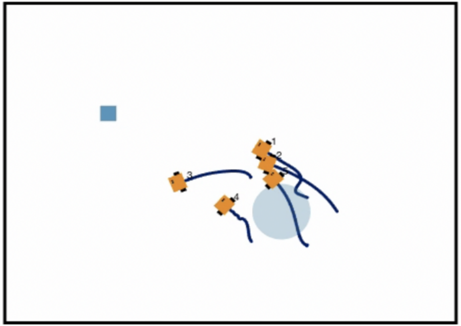


Multi-agent reinforcement learning

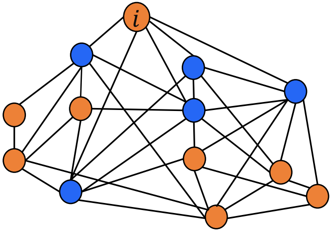
- N agents, communicating on a network
- One environment, common state $\boldsymbol{s}_t \in \mathcal{S}$
  transition probabilities $\mathrm{P}\left(\boldsymbol{s}_{t+1} | \boldsymbol{s}_t\right)$
- Individual actions $\boldsymbol{a}_t^i \in \mathcal{A}^i$
- Individual rewards $R^i(\boldsymbol{s}_t, \boldsymbol{s}_{t+1})$
- compute average cumulative reward

$$V(\boldsymbol{s}) = \mathrm{E}\left[\sum_{t=0}^{\infty} \gamma^t \frac{1}{N} \sum_{i=1}^{N} R^i(\boldsymbol{s}_t, \boldsymbol{s}_{t+1}) \;\middle|\; \boldsymbol{s}_0 = \boldsymbol{s}\right]$$



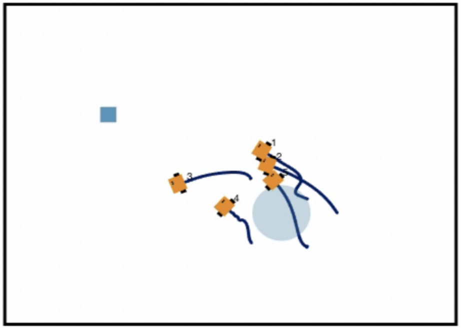


Multi-agent reinforcement learning



- N agents, communicating on a network
- One environment, common state $\boldsymbol{s}_t \in \mathcal{S}$
  transition probabilities $\mathrm{P}\left(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t\right)$
- Individual actions $\boldsymbol{a}_t^i \in \mathcal{A}^i$
- Individual rewards $R^i(\boldsymbol{s}_t, \boldsymbol{s}_{t+1})$
- find $\boldsymbol{V}$ that satisfies

$$V(\boldsymbol{s}) = \sum_{\boldsymbol{z}\in\mathcal{S}} \mathrm{P}\left(\boldsymbol{z}|\boldsymbol{s}\right) \left[\frac{1}{N}\sum_{n=1}^{N} R^i(\boldsymbol{s},\boldsymbol{z}) + \gamma V(\boldsymbol{z})\right]$$



Distributed temporal difference learning

Initialize: Each agent starts at $\boldsymbol{\theta}_0^i$
Iterations: Observe: $\boldsymbol{s}_t$, take action to go to $\boldsymbol{s}_{t+1}$, get reward $R(\boldsymbol{s}_t, \boldsymbol{s}_{t+1})$
Communicate: average estimates from neighbors

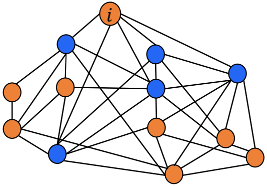
$$\boldsymbol{y}_t^i = \sum_{j \in \mathcal{N}_i} W_{ij} \boldsymbol{\theta}_t^j$$

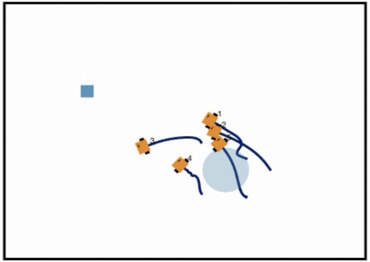
Local updates:

$$\boldsymbol{\theta}_{t+1}^i = \boldsymbol{y}_t^i + \alpha_t d_t^i \boldsymbol{\phi}(\boldsymbol{s}_t),$$

where

$$d_t^i = R^i(\boldsymbol{s}_t, \boldsymbol{s}_{t+1}) + \gamma \boldsymbol{\phi}(\boldsymbol{s}_{t+1})^{\mathrm{T}} \boldsymbol{\theta}_t^i - \boldsymbol{\phi}(\boldsymbol{s}_t)^{\mathrm{T}} \boldsymbol{\theta}_t^i$$





Ingredients for Distributed RL

Distributed RL is a combination of:

- stochastic approximation
- Markov decision processes
- function representation
- network consensus
- **(complicated probabilistic analysis)**

Previous work

Subset of existing results:

- Unified convergence theory: Borkar and Meyn '00
- Convergence rates with "independent noise" (centralized): Thoppe and Borkar '19, Dalal et al '18, Lakshminarayanan and Szepesvari '18
- Convergence rates under Markovian noise (centralized): Bhandari et al COLT '18. Srikant and Ying COLT '19
- Multi-agent RL: Mathkar and Bokar '17, Zhang et al '18, Kar et al '13, Stankovic and Stankovic '16, Macua et al '15

Rate of convergence for distributed TD

- Fixed **step size** $\alpha_t = \alpha$, for small enough α

$$\mathrm{E}\left[\|\boldsymbol{\theta}_t^i - \boldsymbol{\theta}^\star\|\right] \;\leq\; O(\sigma^{t-\tau}) + O(\eta^{t-\tau}) + O(\alpha)$$

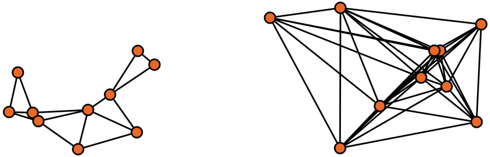
where $\sigma < 1$ is **network connectivity**, $\eta < 1$ are **problem parameters**, and τ is the **mixing time** for the underlying Markov chain

Rate of convergence for distributed TD

- Fixed **step size** $\alpha_t = \alpha$, for small enough α

$$\mathrm{E}\left[\|\boldsymbol{\theta}_t^i - \boldsymbol{\theta}^\star\|\right] \;\leq\; O(\sigma^{t-\tau}) + O(\eta^{t-\tau}) + O(\alpha)$$

where $\sigma < 1$ is **network connectivity**, $\eta < 1$ are **problem parameters**, and τ is the **mixing time** for the underlying Markov chain



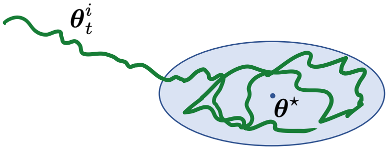
σ larger σ smaller

Rate of convergence for distributed TD

- Fixed **step size** $\alpha_t = \alpha$, for small enough α

$$\mathrm{E}\left[\|\boldsymbol{\theta}_t^i - \boldsymbol{\theta}^\star\|\right] \;\leq\; O(\sigma^{t-\tau}) + O(\eta^{t-\tau}) + O(\alpha)$$

where $\sigma < 1$ is **network connectivity**, $\eta < 1$ are **problem parameters**, and τ is the **mixing time** for the underlying Markov chain



Rate of convergence for distributed TD

- Fixed **step size** $\alpha_t = \alpha$, for small enough α

$$\mathrm{E}\left[\|\boldsymbol{\theta}_t^i - \boldsymbol{\theta}^\star\|\right] \;\leq\; O(\sigma^{t-\tau}) + O(\eta^{t-\tau}) + O(\alpha)$$

  where $\sigma < 1$ is **network connectivity**, $\eta < 1$ are **problem parameters**, and τ is the **mixing time** for the underlying Markov chain

- Time-varying step size $\alpha_t \sim 1/(t+1)$

$$\mathrm{E}\left[\|\boldsymbol{\theta}_t^i - \boldsymbol{\theta}^\star\|\right] \;\leq\; O(\sigma^{t-\tau}) + O\left(\frac{T}{(1-\sigma_2)^2}\frac{\log(t+1)}{t+1}\right)$$

Rate of convergence for distributed TD

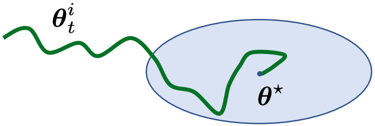
- Fixed **step size** $\alpha_t = \alpha$, for small enough α

$$\mathrm{E}\left[\|\boldsymbol{\theta}_t^i - \boldsymbol{\theta}^\star\|\right] \;\leq\; O(\sigma^{t-\tau}) + O(\eta^{t-\tau}) + O(\alpha)$$

where $\sigma < 1$ is **network connectivity**, $\eta < 1$ are **problem parameters**, and τ is the **mixing time** for the underlying Markov chain

- Time-varying step size $\alpha_t \sim 1/(t+1)$

$$\mathrm{E}\left[\|\boldsymbol{\theta}_t^i - \boldsymbol{\theta}^\star\|\right] \;\leq\; O(\sigma^{t-\tau}) + O\left(\frac{T}{(1-\sigma_2)^2}\frac{\log(t+1)}{t+1}\right)$$



Distributed Stochastic Approximation: General Case

Goal: Find $\boldsymbol{\theta}^\star$ such that $\bar{F}(\boldsymbol{\theta}^\star) = \mathbf{0}$, where

$$\bar{F}(\boldsymbol{\theta}) = \sum_{i=1}^{N} \mathrm{E}[F_i(X_i; \boldsymbol{\theta})],$$

using decentralized communications between agents with access to $F_i(X_i; \boldsymbol{\theta})$.

Using the iteration

$$\boldsymbol{\theta}_i^{k+1} = \sum_{i \in \mathcal{N}(i)} W_{i,j} \boldsymbol{\theta}_j^k + \epsilon F_i(X_i^k, \boldsymbol{\theta}_i^k)$$

gives us

$$\max_j \mathrm{E}\left[\|\boldsymbol{\theta}_i^k - \boldsymbol{\theta}^\star\|_2^2\right] \to O\left(\frac{\epsilon \log(1/\epsilon)}{1 - \sigma_2^2}\right) \quad \text{at a linear rate}$$

when the F_i are Lipschitz, $\bar{F}_i$ are strongly monotone, and the $\{X_i^k\}$ are Markov

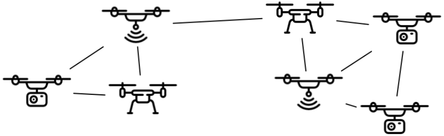
Multi-agent Reinforcement Learning, Scenario 2:
Multiple agents in different environments (dynamics, rewards)

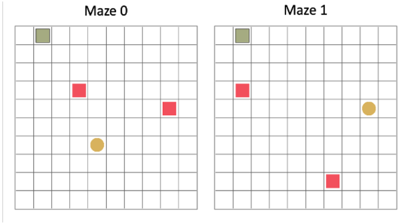
Can we find a jointly optimal policy?

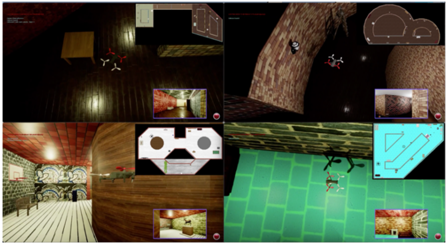
Policy Optimization, Framework

We will set this up as a distributed optimization program with decentralized communications

- One agent explores each environment
- Agent collaborate by sharing their models
- Performance guarantees:
  number of gradient iterations
  sample complexity (future)





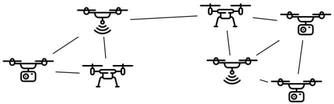


Policy Optimization, Framework

Environments $i = 1, \ldots, N$, each with similar state/action spaces

Key quantities:

- $\pi(\cdot|s)$: policy that maps states into actions
- $r_i(s,a)$: reward function in environment i
- $\rho_i(s)$: initial state distribute in environment i
- $L_i(\pi)$: long-term reward of π in environment i



$$L_i(\pi) = \mathrm{E}\left[\sum_{k=0}^{\infty} \gamma^k r_i(s_i^k, a_i^k)\right], \quad a_i^k \sim \pi(\cdot|s_i^{k-1}), \quad s_i^0 \sim \rho_i$$

We want to solve

$$\underset{\pi}{\text{maximize}} \quad \sum_{i=1}^{N} L_i(\pi)$$

Decentralized Policy Optimization, Challenges

$$\underset{\pi}{\text{maximize}} \; \sum_{i=1}^{N} L_i(\pi) \quad \rightarrow \quad \underset{\theta}{\text{maximize}} \; \sum_{i=1}^{N} L_i(\theta), \quad \pi_\theta(a|s) = \frac{e^{\theta_{s,a}}}{\sum_{a'} e^{\theta_{s,a'}}}$$

- Natural parameterization (softmax) is ill-conditioned at solution

Decentralized Policy Optimization, Challenges

$$\underset{\pi}{\text{maximize}} \sum_{i=1}^{N} L_i(\pi) \quad \to \quad \underset{\theta}{\text{maximize}} \sum_{i=1}^{N} L_i(\theta) - \lambda\, \mathrm{RE}(\theta), \quad \pi_\theta(a|s) = \frac{e^{\theta_{s,a}}}{\sum_{a'} e^{\theta_{s,a'}}}$$

- Natural parameterization (softmax) is ill-conditioned at solution

Decentralized Policy Optimization, Challenges

$$\underset{\pi}{\text{maximize}} \sum_{i=1}^{N} L_i(\pi) \quad \rightarrow \quad \underset{\theta}{\text{maximize}} \sum_{i=1}^{N} L_i(\theta) - \lambda\,\mathrm{RE}(\theta), \quad \pi_\theta(a|s) = \frac{e^{\theta_{s,a}}}{\sum_{a'} e^{\theta_{s,a'}}}$$

- Natural parameterization (softmax) is ill-conditioned at solution
- Even for a single agent, this problem in nonconvex ...
  ... ability to find global optimum tied to "exploration conditions" (Agarwal et al '19)

Decentralized Policy Optimization, Challenges

$$\underset{\pi}{\text{maximize}} \sum_{i=1}^{N} L_i(\pi) \quad \rightarrow \quad \underset{\theta}{\text{maximize}} \sum_{i=1}^{N} L_i(\theta) - \lambda\,\mathrm{RE}(\theta), \quad \pi_\theta(a|s) = \frac{e^{\theta_{s,a}}}{\sum_{a'} e^{\theta_{s,a'}}}$$

- Natural parameterization (softmax) is ill-conditioned at solution
- Even for a single agent, this problem in nonconvex ...
  ... ability to find global optimum tied to "exploration conditions" (Agarwal et al '19)
- Agents have competing interests (global solution suboptimal for every agent)

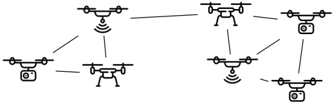
Decentralized Policy Optimization, Challenges

$$\underset{\pi}{\text{maximize}} \sum_{i=1}^{N} L_i(\pi) \quad \rightarrow \quad \underset{\theta}{\text{maximize}} \sum_{i=1}^{N} L_i(\theta) - \lambda \, \mathrm{RE}(\theta), \quad \pi_\theta(a|s) = \frac{e^{\theta_{s,a}}}{\sum_{a'} e^{\theta_{s,a'}}}$$

- Natural parameterization (softmax) is ill-conditioned at solution
- Even for a single agent, this problem in nonconvex ...
  ... ability to find global optimum tied to "exploration conditions" (Agarwal et al '19)
- Agents have competing interests (global solution suboptimal for every agent)
- Gradients can only be computed imperfectly for large or partially specified problems

Algorithm: Decentralized Policy Optimization

$$\underset{\{\theta_i\}}{\text{maximize}} \sum_{i=1}^{N} L_i(\theta_i), \quad \text{subject to} \quad \theta_i = \theta_j, \ (i,j) \in \mathcal{E}$$



- Each agent stores a local version of policy θ_i, initialized to θ_i^0
- At each node, iterate from policy $\pi_{\theta_i^k}$
  - Compute "advantage function" $A(s,a) = Q(s,a) - V(s)$
  - Compute gradient

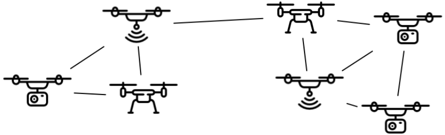
$$\nabla L_i(\theta_i^k) = (\text{complicated function of } \pi_{\theta_i^k} \text{ and } A(s,a))$$

  - Meanwhile, exchange θ_i^k with neighbors
  - Update policy

$$\theta_i^{k+1} = \sum_{j \in \mathcal{N}(i)} W_{i,j} \theta_j^k + \alpha_k \nabla L_i(\theta_i^k)$$

Algorithm: Mathematical Guarantees

$$\theta_i^{k+1} = \sum_{j \in \mathcal{N}(i)} W_{i,j} \theta_j^k + \alpha_k \nabla L_i(\theta_i^k)$$



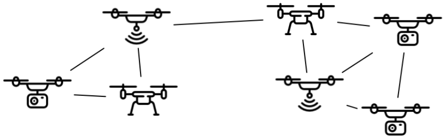
For small enough step sizes α_k, after k iterations we have

$$\left\| \frac{1}{N} \sum_{i=1}^{N} \nabla L_i(\theta_i^k) \right\|^2 \leq O\left(\frac{1}{\sqrt{k}} + \frac{C_g}{k} \right)$$

- Convergence to stationary point (not global max)
- Graph properties expressed in C_g
- Other constants come from λ, N, and MDP properties

Algorithm: Mathematical Guarantees

$$\theta_i^{k+1} = \sum_{j \in \mathcal{N}(i)} W_{i,j}\theta_j^k + \alpha_k \nabla L_i(\theta_i^k)$$



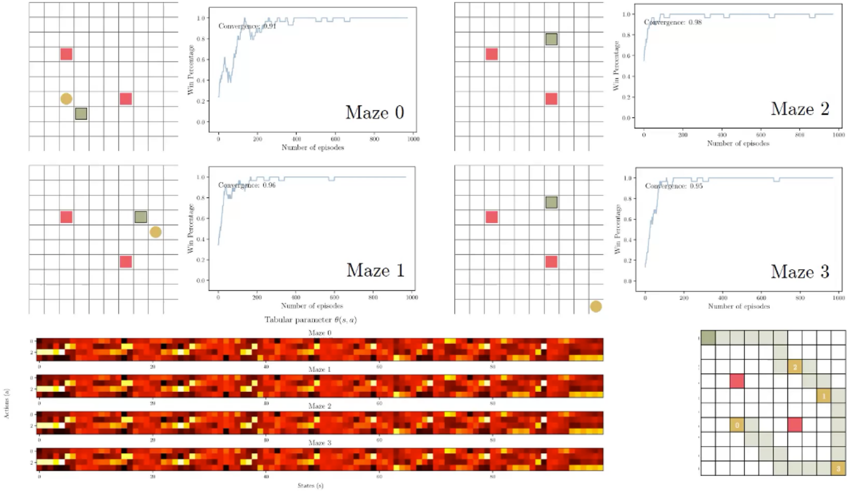
If common states are "equally explored" across environments, then after k iterations

$$\max_j \left\{ \sum_{i=1}^{N} L_i(\theta^*) - L_i(\theta_j^k) \right\} \;\leq\; \epsilon \quad \text{when} \quad k \geq \frac{C}{\epsilon^2}$$

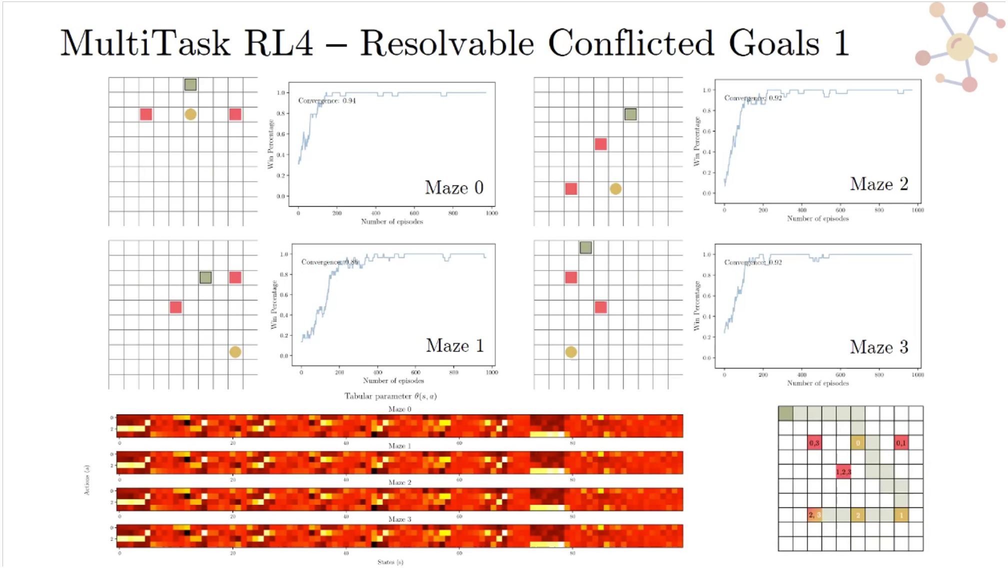
- Convergence to **global optimum**
- Requires careful choice of regularization parameter λ
- "Equal exploration" hard to verify
- Can make this stochastic, but not with finite-sample guarantee

Simulation: GridWorld

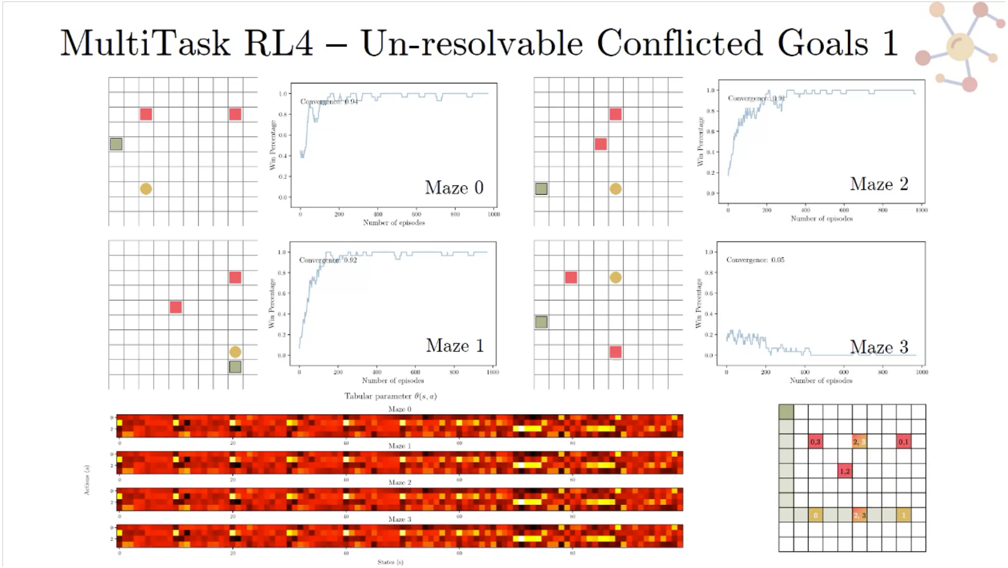
MultiTask RL4 – Unconflicted Goals



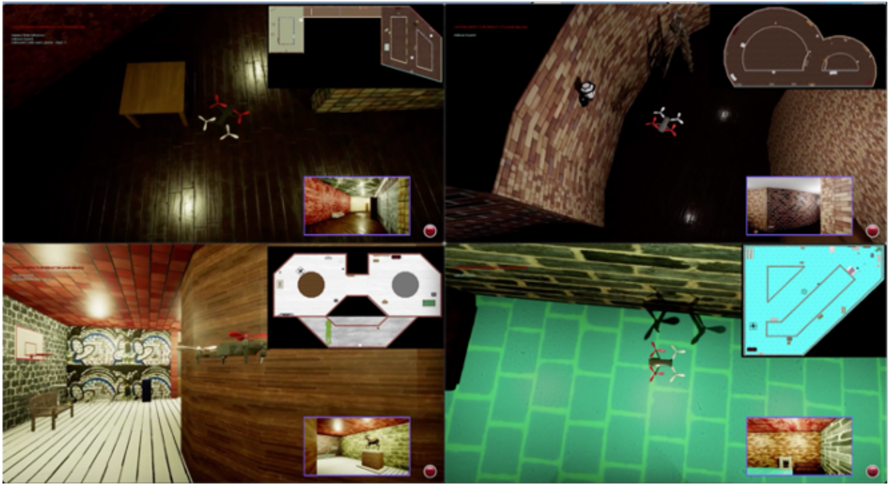
Simulation: GridWorld



Simulation: GridWorld



Simulation: Drones in D-PEDRA



Simulation: Drones in D-PEDRA

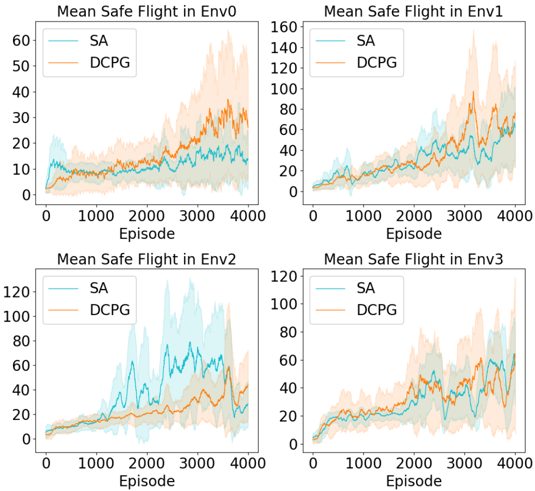


Table 1: MSF of the learned policy

Policy	Env0	Env1	Env2	Env3	Sum
SA-0	15.9	4.5	4.1	3.6	28.1
SA-1	3.0	55.4	9.7	8.1	76.2
SA-2	1.5	0.8	21.1	2.0	25.4
SA-3	2.3	0.8	8.6	40.1	51.8
DCPG	**25.2**	**67.9**	**40.5**	**61.8**	**195.4**
Random	2.5	3.9	4.7	3.7	14.8

Interesting, unexplained result:

Learning a joint policy is easier than learning individual policies

Thank you!

References:

T. T. Doan, S. M. Maguluri, and J. Romberg, “Finite-time performance of distributed temporal difference learning with linear function approximation,” submitted January 2020, arXiv:1907.12530.

T. T. Doan, S. M. Maguluri, and J. Romberg, “Finite-time analysis of distributed TD(0) with linear function approximation for multi-agent reinforcement learning,” ICML 2019.

S. Zeng, T. T. Doan, and J. Romberg, “Finite-time analysis of decentralized stochastic approximation with applications in multi-agent and multi-task learning,” submitted October 2020, arxiv:2010.15088 .

S. Zeng, et al, “ A decentralized policy gradient approach to multi-task reinforcement learning,” submitted October 2020, arxiv:2006.04338.