Multi-Agent Reinforcement Learning with Partial Knowledge over Networks

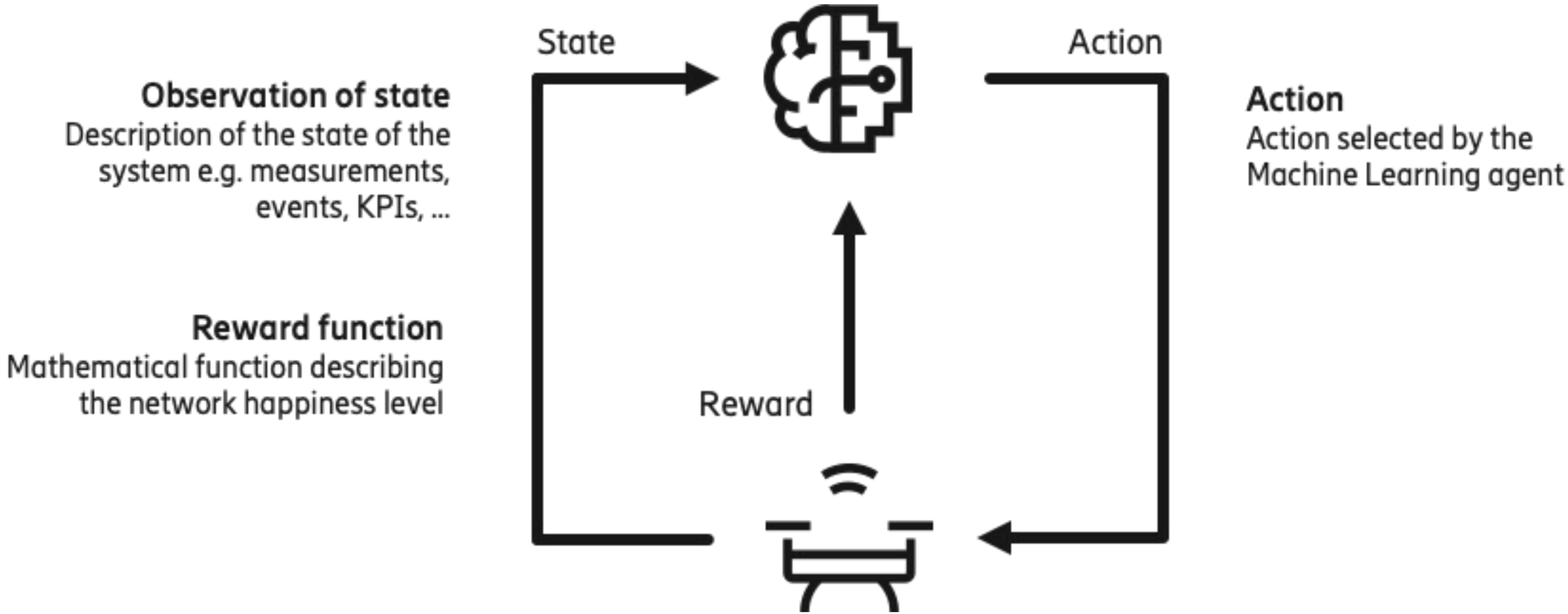
Stefanos Antaris antaris@kth.se
Amaru Cuba Gyllensten amaru.cuba.gyllensten@ri.se
Martin Isaksson martisak@kth.se
Sarit Khirirat sarit@kth.se
Klas Segeljakt klasseg@kth.se

EP3260: Fundamentals of Machine Learning Over Networks

Outline

1. Introduction to Reinforcement Learning
2. Markov Decision Process
3. Multi-agent Markov Decision Process
4. Partially-observable Markov Decision Process
5. Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents (Zhang et.al., 2018)
6. Enterprise Video Streaming

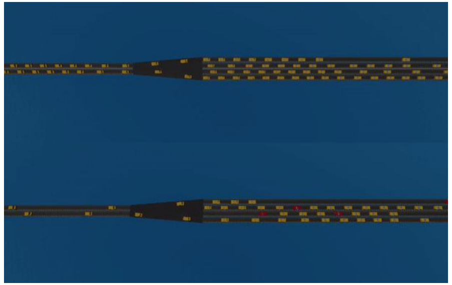
Reinforcement Learning



Reinforcement learning is an area of machine learning inspired by behaviorist psychology, concerned with how software *agents* learn to take *actions* in an *environment* by interacting with it to maximize some notion of cumulative *reward*.

Use-cases for Reinforcement Learning

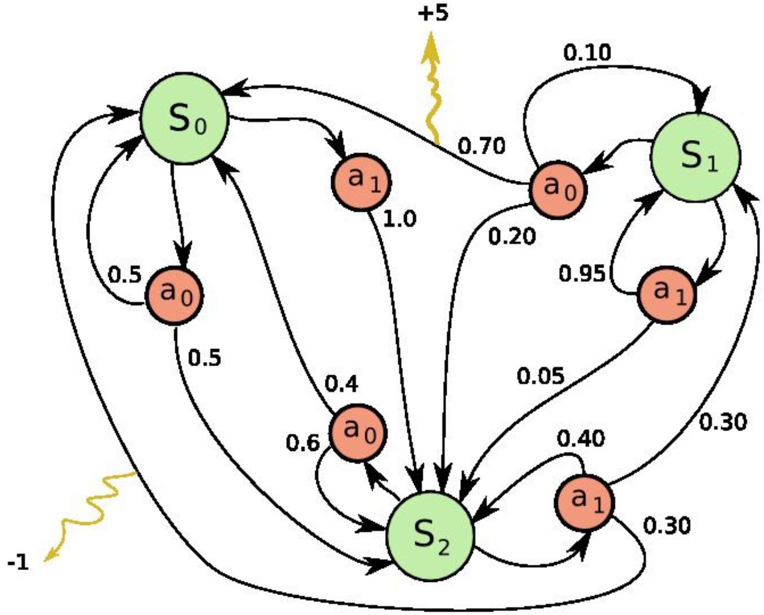
- Resource management in computer clusters
- Traffic Light Control
- Robotics
- Web System Configuration
- Bidding and Advertising
- Games



Markov Decision Process

- At each time step *t*, the process is in some state *s*, and the agent can take an action *a*.

- At the next timestep *t'*, the process responds by (randomly) moving to a new state *s'*.
- The reward *R(a, s, s')* is given.
- Conditional probability distribution of future states of the process depends only on the present state



Source Wikipedia/RL Course by David Silver - Lecture 2: Markov Decision Process

Formal Definition: Markov Decision Process (MDP)

Single agent RL under full observability (Sutton, Barto, 1998, 2018)

$$< \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma >$$

State

Action

Transition probability

Reward

Discount factor

$$\pi(a, s), \text{ where } s \in \mathcal{S}, a \in \mathcal{A}$$

Policy π maps state to the action that gives the highest cumulative reward.

Formal Definition: Markov Decision Process (MDP)

$$< \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma >$$

$$\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \Delta\mathbb{R}$$

$$\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \Delta\mathcal{S}$$

$$\pi : \mathcal{S} \to \Delta\mathcal{A}$$

Reward and State transition functions are unknown. Goal is to find a policy function π that maximizes expected cumulative reward.

Value function and state-action function

Value function for a state S given policy π

Q-function - value function for a state S and immediate action A given policy π

$$v_\pi(s) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s\right], \forall s \in \mathcal{S}$$

$$q_\pi(s, a) = \mathbb{E}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a\right]$$

Partially Observable Markov Decision Process (POMDP)

- The agent cannot directly observe the underlying state
- MDP is to POMDP as Markov Models are to Hidden Markov Models.

$$< \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O} >$$

State, Action, Transition Probability, Reward, Observation, Emission probability

Partially Observable Markov Decision Process (POMDP)

$$< \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O} >$$

$$\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta\mathbb{R}$$

$$\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta\mathcal{S}$$

$$\mathcal{O} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta\Omega$$

$$\pi : \Omega \rightarrow \Delta\mathcal{A}$$

Reward and State transition functions unknown. Goal is to find a policy function π that maximizes expected cumulative reward.
The agent does not directly observe the system state.

Multi agent use-cases

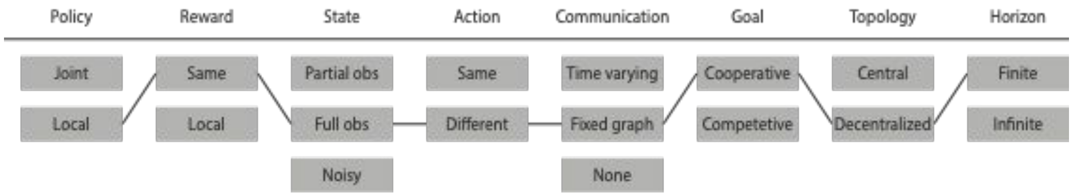
Collaborative multi-agent reinforcement learning

Agents share a common goal

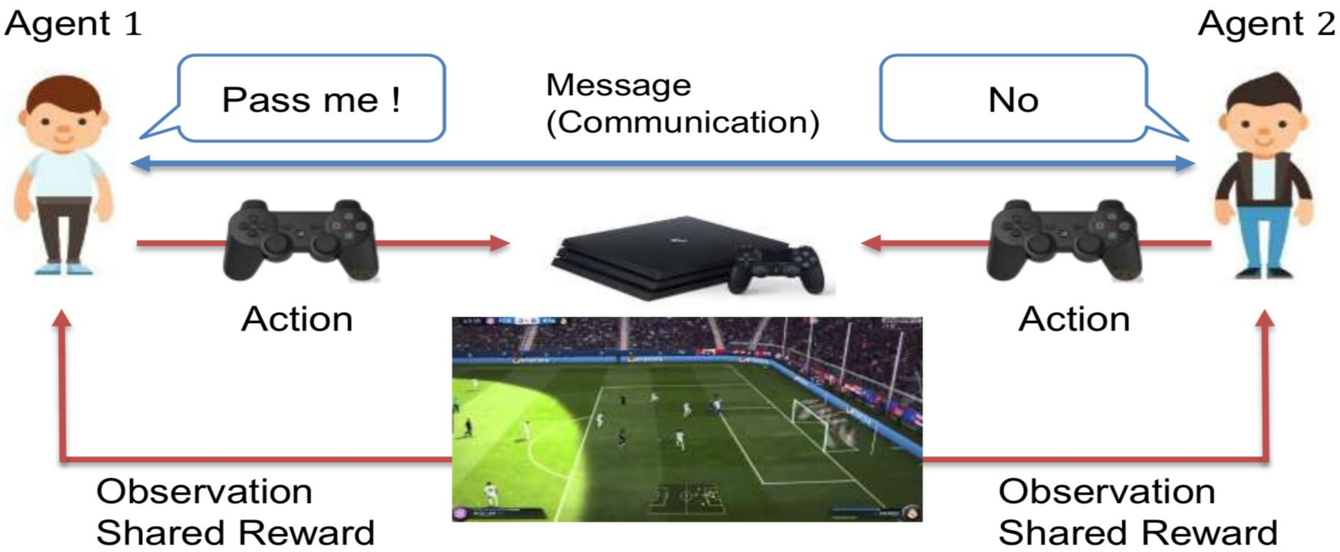
Competitive multi-agent reinforcement learning

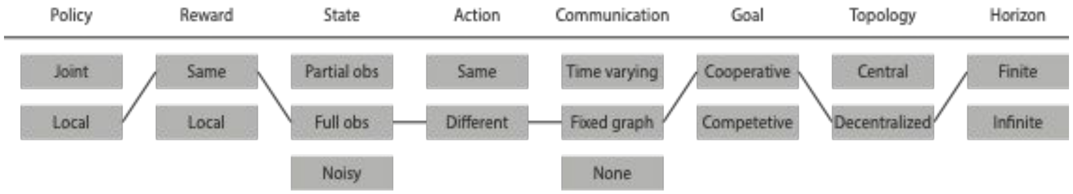
Agents compete against each other

...

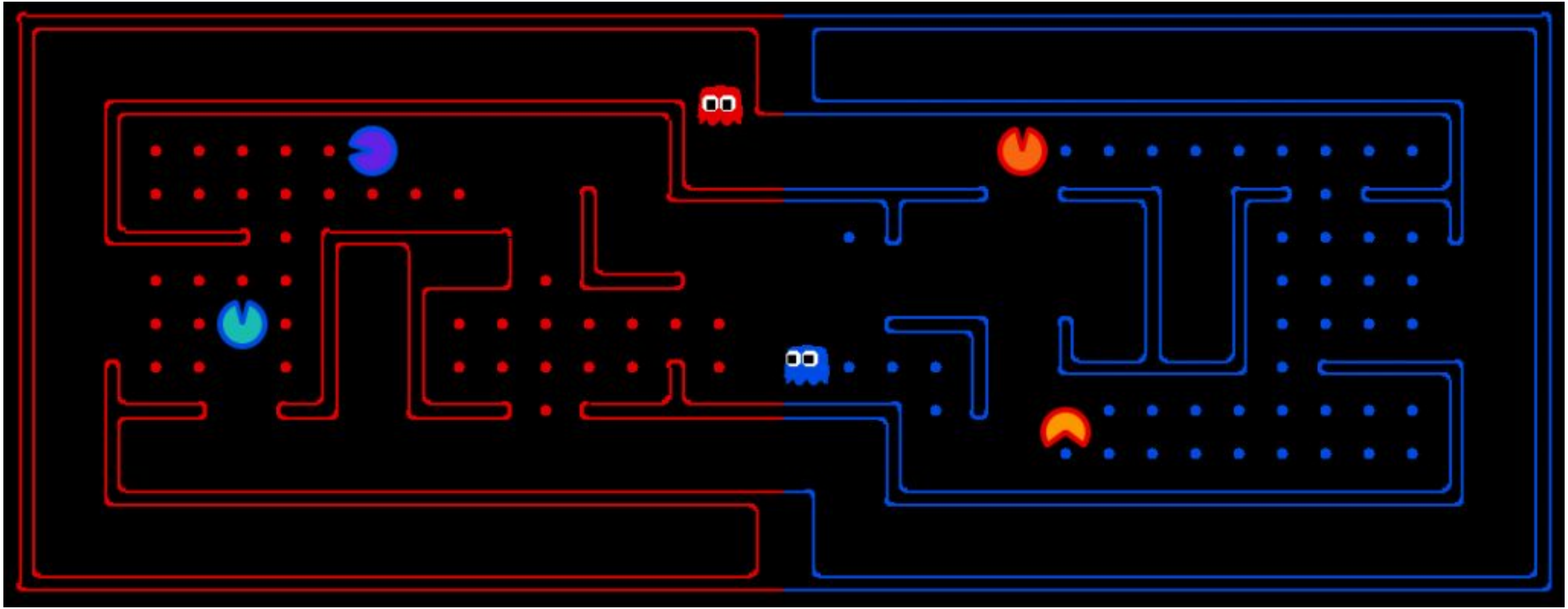


Collaborative Multi-agent Reinforcement Learning





Competitive Multi-Agent Reinforcement Learning



Multi Agent Markov Decision Process (MAMDP)

Multi agent RL under full state observability.

$$< \mathcal{I}, \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma >$$

Agents

State

Joint Action

Transition probability

Joint reward

Discount factor

Multi Agent Markov Decision Process (MAMDP)

$$< \mathcal{I}, \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma >$$

$$\mathcal{A} = \prod_i \mathcal{A}_i$$

$$\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta\mathbb{R}^N$$

$$\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta\mathcal{S}$$

$$\pi_i : \mathcal{S} \rightarrow \Delta\mathcal{A}_i$$

Reward and State transition functions are unknown. Goal is to find a policy function π that maximizes expected aggregate reward for all agents.
Each each agent only observes its own reward.

Multi Agent Partially Observable Markov Decision Process

$$< \mathcal{I}, \mathcal{S}, \mathbf{A}, \mathcal{T}, \mathcal{R}, \mathbf{\Omega}, \mathcal{O}, \gamma >$$

Discount factor

Agents

Observations probability

Action

State

Reward

Transition probability

Observations

Multi-agent RL under partial observability (Bernstein, et. al., 2002)

Multi Agent Partially Observable Markov Decision Process

$$< \mathcal{I}, \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O}, \gamma >$$

$$\mathcal{A} = \prod_i \mathcal{A}_i$$

$$\Omega = \prod_i \Omega_i$$

$$\mathcal{O} : \mathcal{S} \times \mathcal{A} \rightarrow \Delta\Omega$$

$$\pi_i : \Omega_i \rightarrow \Delta\mathcal{A}_i$$

Reward and State transition functions unknown. Goal is to find a policy function π which maximizes expected cumulative reward.
The policy function depends only on each agents observations.

Optimization for Reinforcement Learning

Value optimization, e.g. classical gradient-based algorithms

Policy optimization, e.g. policy gradient

Classical Optimization Algorithms on RL

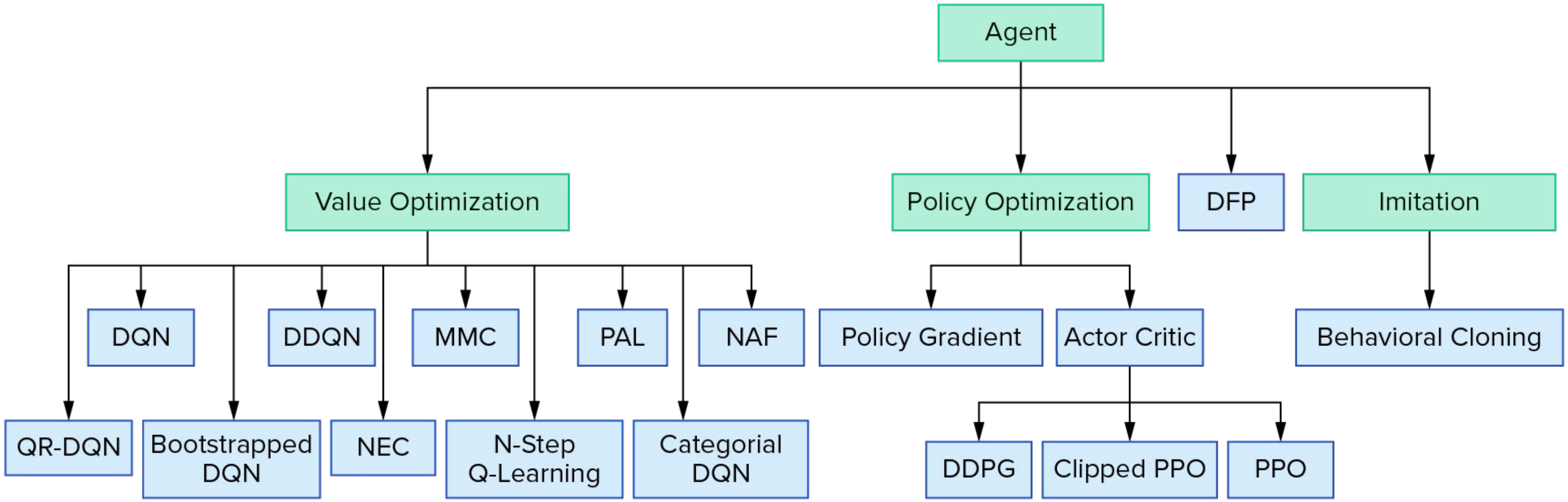
Value optimization

- Apply optimization algorithms to problems based on Bellman optimality condition

Policy optimization

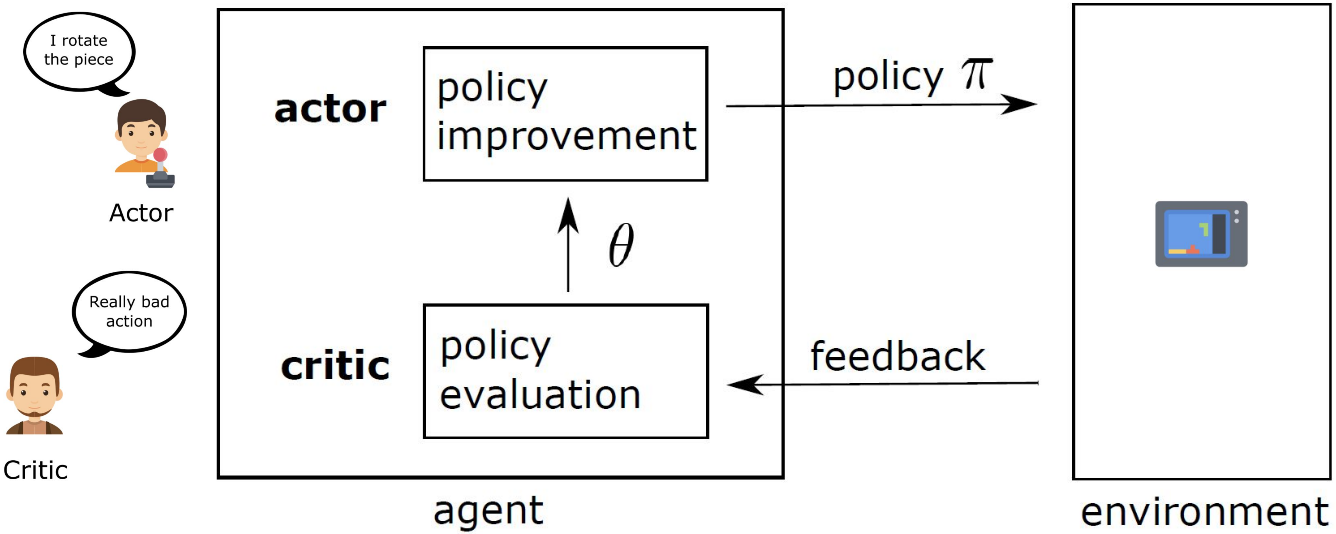
- Apply optimization algorithms directly to problems over reward objectives

Landscape of Reinforcement Learning algorithms



Source https://www.toptal.com/machine-learning/deep-dive-into-reinforcement-learning

Single-Agent Actor Critic Reinforcement Learning



Policy Update: $$\Delta\theta = \alpha * \nabla_\theta * (log\,\pi(S_t, A_t, \theta)) * \cancel{R(t)}$$

New update: $$\Delta\theta = \alpha * \nabla_\theta * (log\,\pi(S_t, A_t, \theta)) * \boxed{Q(S_t, A_t)}$$

Source Simonini “An intro to Advantage Actor Critic methods”

Single-Agent Reinforcement Learning

Problem set-up: An agent determines the policy to maximize long-term reward.

In essence, an agent estimates

$$V^\pi(s) = \max \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s = s_0, \pi\right].$$

The unique solution from the Bellman optimality equation becomes

$$V^\pi = T^\pi V^\pi := R^\pi + \gamma P^\pi V^\pi.$$

Q: How to formulate RL into the optimization problems?

Mean Squared Projected Bellman Error (MSPBE)

Idea: Instead we approximate the value function $V^\pi = \phi(s)^T\theta$, where

- θ is the estimated parameter (which governs the policy), and
- $\phi : S \to \mathbb{R}^d$ Is the feature map induced by the learning model.

To measure the convergence toward the unique Bellman solution, define

$$\mathrm{MSPBE}(\theta) = \frac{1}{2}||V^\pi - \Pi T^\pi V^\pi||^2 + \rho||\theta||^2.$$

By the proper choice of Π,

$$\mathrm{MSPBE}(\theta) = \frac{1}{2}||A\theta - b||^2_{C^{-1}} + \rho||\theta||^2$$

Mean Squared Projected Bellman Error (MSPBE)

$$\mathrm{MSPBE}(\theta) = \frac{1}{2}||A\theta - b||^2_{C^{-1}} + \rho||\theta||^2$$

Here, $A = \mathbb{E}[\phi_t(\phi_t - \gamma\phi_t')^T], b = \mathbb{E}[\phi_t r_t], C = \mathbb{E}[\phi_t\phi_t^T]$, where

- ϕ_t, r_t are the current feature vector and reward
- ϕ_{t+1} are the feature vector at the next state

Usually, we approximate by taking the empirical average, i.e.

$$A \approx \frac{1}{n}\sum_{t=1}^{n}\underbrace{\phi_t(\phi_t - \gamma\phi_t')^T}_{A_t}, b \approx \frac{1}{n}\sum_{t=1}^{n}\underbrace{\phi_t r_t}_{b_t}, C \approx \frac{1}{n}\sum_{t=1}^{n}\underbrace{\phi_t\phi_t^T}_{C_t}.$$

Saddle-point Equivalent Problem to Empirical-MSPBE

Goal: an agent solves an ℓ_2-regularized optimization problem

$$\min_\theta \frac{1}{2}\|A\theta - b\|^2_{C^{-1}} + \rho\|\theta\|^2.$$

By Fenchel duality, $\frac{1}{2}\|y\|^2_{C^{-1}} = \max_x (y^T x - \frac{1}{2}\|x\|^2_C)$, and thus

$$\min_\theta \max_w \frac{1}{n}\sum_{t=1}^n \mathcal{L}_t(w, \theta),$$

where

$$\mathcal{L}_t(w,\theta) = \frac{1}{2} w^T (A_t\theta - b_t) - \frac{1}{2}\|w\|^2_{C_t} + \rho\|\theta\|^2$$

Saddle-point Equivalent Problem to Empirical-MSPBE

$$\min_{\theta} \max_{w} \frac{1}{n} \sum_{t=1}^{n} \mathcal{L}_t(w, \theta),$$

- Primal and negative dual gradients for each loss function can be stacked:

$$G_t(w, \theta) = \begin{bmatrix} \nabla_\theta \mathcal{L}_t(w, \theta) \\ -\nabla_w \mathcal{L}_t(w, \theta) \end{bmatrix}$$

- easily solved by (randomized) gradient-based optimization methods

Saddle-point Equivalent Problem to Empirical-MSPBE

Gradient descent update:

$$\begin{bmatrix} \theta \\ w \end{bmatrix} \leftarrow \begin{bmatrix} \theta \\ w \end{bmatrix} - \begin{bmatrix} \gamma_\theta & 0 \\ 0 & \gamma_w \end{bmatrix} \left(\frac{1}{n} \sum_{t=1}^{n} G_t(w, \theta) \right)$$

SVRG/SAGA update:

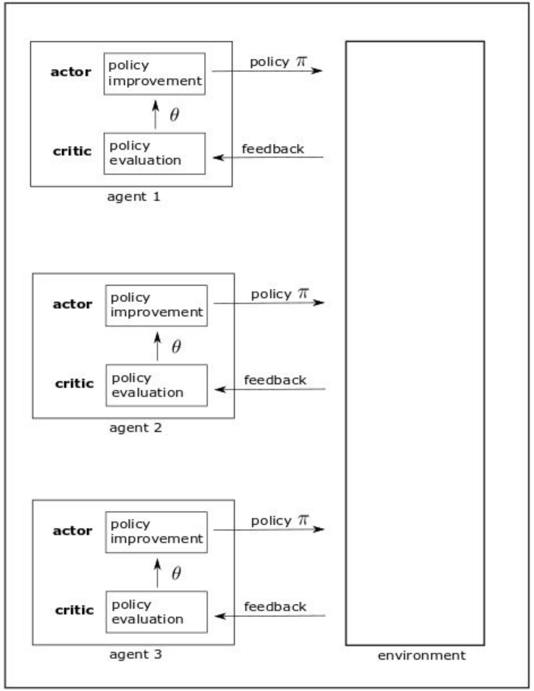
$$\begin{bmatrix} \theta \\ w \end{bmatrix} \leftarrow \begin{bmatrix} \theta \\ w \end{bmatrix} - \begin{bmatrix} \gamma_\theta & 0 \\ 0 & \gamma_w \end{bmatrix} \left(G_t(w, \theta) - G_t(w_f, \theta_f) + \frac{1}{n} \sum_{t=1}^{n} G_t(w_f, \theta_f) \right)$$

Note: Linear convergence guarantees toward the global optimum.

Single-Agent Reinforcement Learning

- Solve RL by (randomized) first-order optimization methods.

Q: Is it possible to extend the formulation to solve multi-agent RL?



Multi-Agent Reinforcement Learning

Goal: A group of N agents collaboratively maximize total collective return.

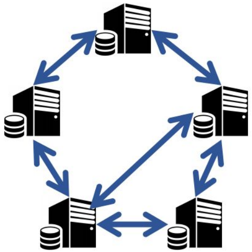
$$V^{\pi}(s) = \max \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s = s_0, \pi\right],$$

where

$$R(s_t, a_t) = \frac{1}{N} \sum_{i=1}^{N} R_i(s_t, a_t)$$

Set-up:

- the states and actions are available to all agents
- The reward is private for each agent



MBPSE for Multi-Agent Reinforcement Learning

By easy computation, the equivalent optimization to multi-agent RL becomes

$$\min \frac{1}{N} \sum_{i=1}^{N} \left(\frac{1}{2} \| \hat{A}\theta - \hat{b}_i \|^2_{\hat{C}^{-1}} + \rho \|\theta\|^2 \right)$$

- $\hat{A}, \hat{C}$ are the same as single-agent RL (states and actions known to all agents)
- $\hat{b}_i = (1/n) \sum_{t=1}^{n} \phi_t r_{t,i}$ where $r_{t,i}$ is the reward known only for agent i

Similar to single-agent RL, we can derive the Saddle-point equivalent problem

Multi-Agent Primal-Dual Optimization Problem

$$\min_{\theta} \max_{w_i, i=1,2,\ldots,N} \frac{1}{n} \frac{1}{N} \sum_{t=1}^{n} \sum_{i=1}^{N} \mathcal{L}_t(w_i, \theta),$$

where

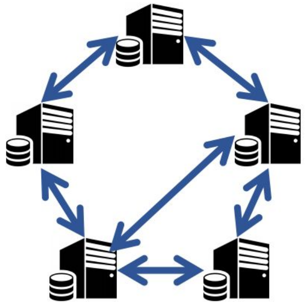
$$\mathcal{L}_t(w_i, \theta) = \frac{1}{2} w_i^T (A_t \theta - b_{t,i}) - \frac{1}{2} \|w_i\|_{C_t}^2 + \rho \|\theta\|^2.$$

Challenges: a decentralized first-order algorithm with full solution accuracy

Gradient Tracking Methods

Consensus average of both solutions and gradients

For each agent do:



$$s_i^k = \sum_{j=1}^{N} W_{ij} s_j^{k-1} + \frac{1}{M}\left[g_i(x_i^k) - g_i(x_i^{k-\tau_i^k})\right]$$

$$x_i^k = \sum_{j=1}^{N} W_{ij} x_j^{k-1} - \gamma s_i^k.$$

Unlike other classical consensus-based algorithms, gradient tracking guarantees

- linear convergence rate for strongly convex optimization toward the **global minimum** with full accuracy
  - for synchronous case $\tau_i^k = 1$ and for asynchronous case $\tau_i^k = \tau$.

The gradient tracking methods are easily applied for MARL!

(Asynchronous) Gradient Tracking Methods

PD-DistIAG (ST on primal variable, while SAG on dual variable)

for each agent $i \in \{1, \ldots, N\}$ **do**

Update the gradient surrogates by

$$\boldsymbol{s}_i^t = \sum_{j=1}^N W_{ij} \boldsymbol{s}_j^{t-1} + \frac{1}{M}\Big[\nabla_{\boldsymbol{\theta}} J_{i,p_t}(\boldsymbol{\theta}_i^t, \boldsymbol{w}_i^t) - \nabla_{\boldsymbol{\theta}} J_{i,p_t}(\boldsymbol{\theta}_i^{\tau_{p_t}^{t-1}}, \boldsymbol{w}_i^{\tau_{p_t}^{t-1}})\Big] ,$$

$$\boldsymbol{d}_i^t = \boldsymbol{d}_i^{t-1} + \frac{1}{M}\Big[\nabla_{\boldsymbol{w}_i} J_{i,p_t}(\boldsymbol{\theta}_i^t, \boldsymbol{w}_i^t) - \nabla_{\boldsymbol{w}_i} J_{i,p_t}(\boldsymbol{\theta}_i^{\tau_{p_t}^{t-1}}, \boldsymbol{w}_i^{\tau_{p_t}^{t-1}})\Big] ,$$

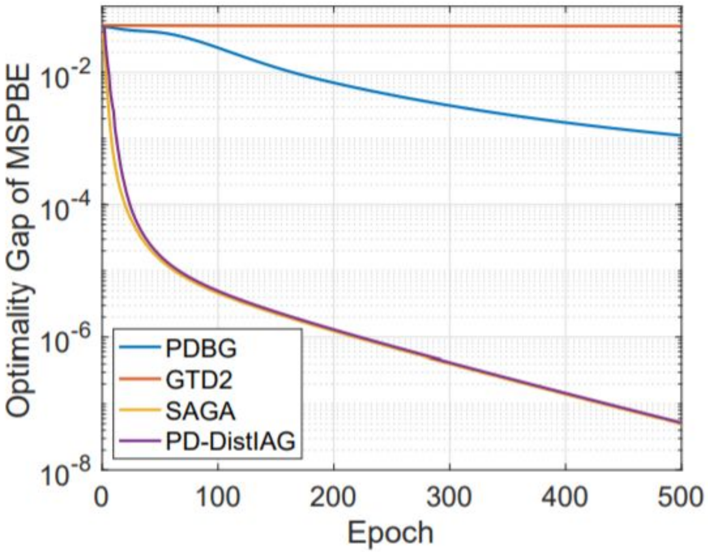
where $\nabla_{\boldsymbol{\theta}} J_{i,p}(\boldsymbol{\theta}_i^0, \boldsymbol{w}_i^0) = \mathbf{0}$ and $\nabla_{\boldsymbol{w}_i} J_{i,p}(\boldsymbol{\theta}_i^0, \boldsymbol{w}_i^0) = \mathbf{0}$ for all $p \in [M]$ for initialization.

Perform primal-dual updates using $\boldsymbol{s}_i^t, \boldsymbol{d}_i^t$ as surrogates for the gradients $w.r.t.$ $\boldsymbol{\theta}$ and $\boldsymbol{w}_i$:

$$\boldsymbol{\theta}_i^{t+1} = \sum_{j=1}^N W_{ij} \boldsymbol{\theta}_j^t - \gamma_1 \boldsymbol{s}_i^t, \quad \boldsymbol{w}_i^{t+1} = \boldsymbol{w}_i^t + \gamma_2 \boldsymbol{d}_i^t \ .$$

Multi-Agent RL on `Mountaincar` Dataset

- Comparisons of PD-DistIAG against other well-known centralized optimization algorithms



- Optimization algorithms outperform traditional RL algorithms.
- PD-DistIAG guarantees comparable convergence rate to centralized methods (or even faster).

Classical Optimization Algorithms on RL

Value optimization

- Popular optimization algorithms can be applied to solve SA and MA RL

Policy optimization

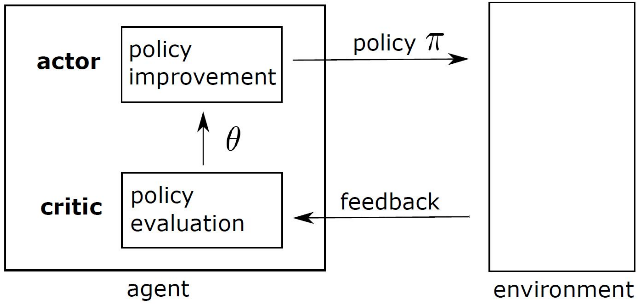
- Apply optimization algorithms directly to problems over reward objectives

Policy Gradient Methods

Problem set-up: An agent determines the policy to maximize long-term reward.

$$V^\pi(s) = \max \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s = s_0, \pi\right].$$

Simplifications:



- parameterized policies $\Pi_\theta = \{\pi_\theta : \theta \in \mathbb{R}^d\}$
- with distribution of a trajectory $p_\theta(\tau)$

Then, we easily apply dual ascent algorithm:

$$\theta \leftarrow \theta + \gamma \nabla V^\pi(s)$$

where $\nabla V^\pi(s) = \mathbb{E}[\nabla \log p_\theta(\tau) \cdot \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)]$

Actor-Critic Algorithms

Limitations of policy gradient methods: high variance.

Solutions: Actor-critic algorithms

$$\theta \leftarrow \theta + \gamma \mathbb{E}\{\nabla \log p_\theta(\tau) \cdot A_t\}$$

where $$A_t = Q_\theta(s_t, a_t) - V^\pi(s_t)$$

$Q_\theta(s_t, a_t)$ is the expected return when taking the current action from the current state

$V^\pi(s_t)$ is the expected return from the current state (which produces the future action)

Actor-Critic Algorithms for MARL

The local advantage function $A_\theta^i : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is defined as

$$A_\theta^i(s, a) = Q_\theta(s, a) - \widetilde{V_\theta^i}(s, a^{-i})$$

where $\widetilde{V_\theta^i}(s, a^{-i}) = \sum_{a^i \in \mathcal{A}^i} \pi_{\theta^i}^i(s, a^i) \cdot Q_\theta(s, a^i, a^{-i})$

Then the gradient of J with respect to θ is given by

$$\begin{aligned} \nabla_{\theta^i} J(\theta) &= \mathbb{E}_{\sim d_\theta, a \sim \pi_\theta}[\nabla_{\theta^i} log \pi_{\theta^i}^i(s, a^i) \cdot A_\theta(s, a)] \\ &= \mathbb{E}_{\sim d_\theta, a \sim \pi_\theta}[\nabla_{\theta^i} log \pi_{\theta^i}^i(s, a^i) \cdot A_\theta^i(s, a)] \end{aligned}$$

Networked actor-critic algorithms (Algorithm 1)

Algorithm 1 The networked actor-critic algorithm based on action-value function

Input: Initial values of the parameters μ_0^i, ω_0^i, $\widetilde{\omega}_0^i$, θ_0^i, $\forall i \in \mathcal{N}$; the initial state s_0 of the MDP, and stepsizes $\{\beta_{\omega,t}\}_{t\geq 0}$ and $\{\beta_{\theta,t}\}_{t\geq 0}$.

Source K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, “Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents,” 2018.

Networked actor-critic algorithms (Algorithm 1)

Algorithm 1 The networked actor-critic algorithm based on action-value function

Input: Initial values of the parameters $\mu_0^i, \omega_0^i, \widetilde{\omega}_0^i, \theta_0^i, \forall i \in \mathcal{N}$; the initial state s_0 of the MDP, and stepsizes $\{\beta_{\omega,t}\}_{t\geq 0}$ and $\{\beta_{\theta,t}\}_{t\geq 0}$.

$\mathcal{N}$ - Set of agents

Critic

μ_t^i - Estimate of averaged return J

ω_t^i - Parameter for approximating the optimal action-value function Q

Actor

θ_t^i - Parameter for approximating the optimal policy π

Source K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, “Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents,” 2018.

Networked actor-critic algorithms (Algorithm 1)

Algorithm 1 The networked actor-critic algorithm based on action-value function

Input: Initial values of the parameters $\mu_0^i, \omega_0^i, \widetilde{\omega}_0^i, \theta_0^i, \forall i \in \mathcal{N}$; the initial state s_0 of the MDP, and stepsizes $\{\beta_{\omega,t}\}_{t\geq 0}$ and $\{\beta_{\theta,t}\}_{t\geq 0}$.
Each agent $i \in \mathcal{N}$ executes action $a_0^i \sim \pi^i_{\theta_0^i}(s_0, \cdot)$ and observes joint actions $a_0 = (a_0^1, \ldots, a_0^N)$.

$\mathcal{N}$ - Set of agents

Critic

μ_t^i - Estimate of averaged return J

ω_t^i - Parameter for approximating the optimal action-value function Q

Actor

θ_t^i - Parameter for approximating the optimal policy π

Source K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, “Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents,” 2018.

Networked actor-critic algorithms (Algorithm 1)

Algorithm 1 The networked actor-critic algorithm based on action-value function

Input: Initial values of the parameters $\mu_0^i, \omega_0^i, \widetilde{\omega}_0^i, \theta_0^i, \forall i \in \mathcal{N}$; the initial state s_0 of the MDP, and stepsizes $\{\beta_{\omega,t}\}_{t\geq 0}$ and $\{\beta_{\theta,t}\}_{t\geq 0}$.
Each agent $i \in \mathcal{N}$ executes action $a_0^i \sim \pi^i_{\theta_0^i}(s_0,\cdot)$ and observes joint actions $a_0 = (a_0^1,\ldots,a_0^N)$.
Initialize the iteration counter $t \leftarrow 0$.
Repeat:

Update the iteration counter $t \leftarrow t+1$.
Until Convergence

$\mathcal{N}$ - Set of agents

Critic

μ_t^i - Estimate of averaged return J

ω_t^i - Parameter for approximating the optimal action-value function Q

Actor

θ_t^i - Parameter for approximating the optimal policy π

Source K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, "Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents," 2018.

Networked actor-critic algorithms (Algorithm 1)

Algorithm 1 The networked actor-critic algorithm based on action-value function

Input: Initial values of the parameters μ_0^i, ω_0^i, $\widetilde{\omega}_0^i$, θ_0^i, $\forall i \in \mathcal{N}$; the initial state s_0 of the MDP, and stepsizes $\{\beta_{\omega,t}\}_{t\geq 0}$ and $\{\beta_{\theta,t}\}_{t\geq 0}$.
Each agent $i \in \mathcal{N}$ executes action $a_0^i \sim \pi^i_{\theta_0^i}(s_0, \cdot)$ and observes joint actions $a_0 = (a_0^1, \ldots, a_0^N)$.
Initialize the iteration counter $t \leftarrow 0$.
Repeat:
  for all $i \in \mathcal{N}$ **do**
    Observe state s_{t+1}, and reward r_{t+1}^i.
    Update $\mu_{t+1}^i \leftarrow (1-\beta_{\omega,t}) \cdot \mu_t^i + \beta_{\omega,t} \cdot r_{t+1}^i$.
    Select and execute action $a_{t+1}^i \sim \pi^i_{\theta_t^i}(s_{t+1}, \cdot)$.
  end for
  Observe joint actions $a_{t+1} = (a_{t+1}^1, \ldots, a_{t+1}^N)$.

  Update the iteration counter $t \leftarrow t+1$.
Until Convergence

$\mathcal{N}$ - Set of agents

$$\text{Critic}\begin{cases} \mu_t^i \text{ - Estimate of averaged return J} \\ \omega_t^i \text{ - Parameter for approximating the optimal action-value function Q} \end{cases}$$

$$\text{Actor}\begin{cases} \theta_t^i \text{ - Parameter for approximating the optimal policy } \pi \end{cases}$$

Source K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, “Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents,” 2018.

Networked actor-critic algorithms (Algorithm 1)

Algorithm 1 The networked actor-critic algorithm based on action-value function

Input: Initial values of the parameters $\mu_0^i, \omega_0^i, \widetilde{\omega}_0^i, \theta_0^i, \forall i \in \mathcal{N}$; the initial state s_0 of the MDP, and stepsizes $\{\beta_{\omega,t}\}_{t\geq 0}$ and $\{\beta_{\theta,t}\}_{t\geq 0}$.
Each agent $i \in \mathcal{N}$ executes action $a_0^i \sim \pi^i_{\theta_0^i}(s_0, \cdot)$ and observes joint actions $a_0 = (a_0^1, \ldots, a_0^N)$.
Initialize the iteration counter $t \leftarrow 0$.
Repeat:
  for all $i \in \mathcal{N}$ **do**
    Observe state s_{t+1}, and reward r_{t+1}^i.
    Update $\mu_{t+1}^i \leftarrow (1-\beta_{\omega,t}) \cdot \mu_t^i + \beta_{\omega,t} \cdot r_{t+1}^i$.
    Select and execute action $a_{t+1}^i \sim \pi^i_{\theta_t^i}(s_{t+1}, \cdot)$.
  end for
  Observe joint actions $a_{t+1} = (a_{t+1}^1, \ldots, a_{t+1}^N)$.
  for all $i \in \mathcal{N}$ **do**
    Update $\delta_t^i \leftarrow r_{t+1}^i - \mu_t^i + Q_{t+1}(\omega_t^i) - Q_t(\omega_t^i)$.
    Critic step: $\widetilde{\omega}_t^i \leftarrow \omega_t^i + \beta_{\omega,t} \cdot \delta_t^i \cdot \nabla_\omega Q_t(\omega_t^i)$.
  end for
  Update the iteration counter $t \leftarrow t + 1$.
Until Convergence

$\mathcal{N}$ - Set of agents

$$\text{Critic} \begin{cases} \mu_t^i \text{ - Estimate of averaged return J} \\ \delta_t^i \text{ - Action-value temporal difference error} \\ \omega_t^i \text{ - Parameter for approximating the optimal action-value function Q} \end{cases}$$

$$\text{Actor} \begin{cases} \theta_t^i \text{ - Parameter for approximating the optimal policy } \pi \end{cases}$$

Source K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, "Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents," 2018.

Networked actor-critic algorithms (Algorithm 1)

Algorithm 1 The networked actor-critic algorithm based on action-value function

Input: Initial values of the parameters $\mu_0^i, \omega_0^i, \widetilde{\omega}_0^i, \theta_0^i, \forall i \in \mathcal{N}$; the initial state s_0 of the MDP, and stepsizes $\{\beta_{\omega,t}\}_{t\geq 0}$ and $\{\beta_{\theta,t}\}_{t\geq 0}$.
Each agent $i \in \mathcal{N}$ executes action $a_0^i \sim \pi^i_{\theta_0^i}(s_0, \cdot)$ and observes joint actions $a_0 = (a_0^1, \dots, a_0^N)$.
Initialize the iteration counter $t \leftarrow 0$.
Repeat:
  for all $i \in \mathcal{N}$ **do**
    Observe state s_{t+1}, and reward r_{t+1}^i.
    Update $\mu_{t+1}^i \leftarrow (1-\beta_{\omega,t}) \cdot \mu_t^i + \beta_{\omega,t} \cdot r_{t+1}^i$.
    Select and execute action $a_{t+1}^i \sim \pi^i_{\theta_t^i}(s_{t+1}, \cdot)$.
  end for
  Observe joint actions $a_{t+1} = (a_{t+1}^1, \dots, a_{t+1}^N)$.
  for all $i \in \mathcal{N}$ **do**
    Update $\delta_t^i \leftarrow r_{t+1}^i - \mu_t^i + Q_{t+1}(\omega_t^i) - Q_t(\omega_t^i)$.
    Critic step: $\widetilde{\omega}_t^i \leftarrow \omega_t^i + \beta_{\omega,t} \cdot \delta_t^i \cdot \nabla_\omega Q_t(\omega_t^i)$.
    Update $A_t^i \leftarrow Q_t(\omega_t^i) - \sum_{a^i \in \mathcal{A}^i} \pi^i_{\theta_t^i}(s_t, a^i) \cdot Q(s_t, a^i, a^{-i}; \omega_t^i)$, $\psi_t^i \leftarrow \nabla_{\theta^i} \log \pi^i_{\theta_t^i}(s_t, a_t^i)$.
    Actor step: $\theta_{t+1}^i \leftarrow \theta_t^i + \beta_{\theta,t} \cdot A_t^i \cdot \psi_t^i$.
  end for
  Update the iteration counter $t \leftarrow t+1$.
Until Convergence

$\mathcal{N}$ - Set of agents

$\underbrace{}_{\text{Critic}}$

Critic
- μ_t^i - Estimate of averaged return J
- δ_t^i - Action-value temporal difference error
- ω_t^i - Parameter for approximating the optimal action-value function Q

Actor
- A_t^i - Sample of the advantage function
- ψ_t^i - Sample of the score function (policy gradient)
- θ_t^i - Parameter for approximating the optimal policy π

Source K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, "Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents," 2018.

Networked actor-critic algorithms (Algorithm 1)

Algorithm 1 The networked actor-critic algorithm based on action-value function

Input: Initial values of the parameters μ_0^i, ω_0^i, $\widetilde{\omega}_0^i$, θ_0^i, $\forall i \in \mathcal{N}$; the initial state s_0 of the MDP, and stepsizes $\{\beta_{\omega,t}\}_{t\geq 0}$ and $\{\beta_{\theta,t}\}_{t\geq 0}$.
Each agent $i \in \mathcal{N}$ executes action $a_0^i \sim \pi^i_{\theta_0^i}(s_0, \cdot)$ and observes joint actions $a_0 = (a_0^1, \ldots, a_0^N)$.
Initialize the iteration counter $t \leftarrow 0$.
Repeat:
  for all $i \in \mathcal{N}$ **do**
    Observe state s_{t+1}, and reward r_{t+1}^i.
    Update $\mu_{t+1}^i \leftarrow (1-\beta_{\omega,t}) \cdot \mu_t^i + \beta_{\omega,t} \cdot r_{t+1}^i$.
    Select and execute action $a_{t+1}^i \sim \pi^i_{\theta_t^i}(s_{t+1}, \cdot)$.
  end for
  Observe joint actions $a_{t+1} = (a_{t+1}^1, \ldots, a_{t+1}^N)$.
  for all $i \in \mathcal{N}$ **do**
    Update $\delta_t^i \leftarrow r_{t+1}^i - \mu_t^i + Q_{t+1}(\omega_t^i) - Q_t(\omega_t^i)$.
    Critic step: $\widetilde{\omega}_t^i \leftarrow \omega_t^i + \beta_{\omega,t} \cdot \delta_t^i \cdot \nabla_\omega Q_t(\omega_t^i)$.
    Update $A_t^i \leftarrow Q_t(\omega_t^i) - \sum_{a^i \in \mathcal{A}^i} \pi^i_{\theta_t^i}(s_t, a^i) \cdot Q(s_t, a^i, a^{-i}; \omega_t^i)$, $\psi_t^i \leftarrow \nabla_{\theta^i} \log \pi^i_{\theta_t^i}(s_t, a_t^i)$.
    Actor step: $\theta_{t+1}^i \leftarrow \theta_t^i + \beta_{\theta,t} \cdot A_t^i \cdot \psi_t^i$.
    Send $\widetilde{\omega}_t^i$ to the neighbors $\{j \in \mathcal{N} : (i,j) \in \mathcal{E}_t\}$ over the communication network $\mathcal{G}_t$.
  end for
  for all $i \in \mathcal{N}$ **do**
    Consensus step: $\omega_{t+1}^i \leftarrow \sum_{j \in \mathcal{N}} c_t(i,j) \cdot \widetilde{\omega}_t^j$.
  end for
  Update the iteration counter $t \leftarrow t+1$.
Until Convergence

$\mathcal{N}$ - Set of agents

Critic
- μ_t^i - Estimate of averaged return J
- δ_t^i - Action-value temporal difference error
- ω_t^i - Parameter for approximating the optimal action-value function Q

Actor
- A_t^i - Sample of the advantage function
- ψ_t^i - Sample of the score function (policy gradient)
- θ_t^i - Parameter for approximating the optimal policy π

$c_t(i,j)$ - Message weight

Source K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, “Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents,” 2018.

Networked actor-critic algorithms (Algorithm 2)

Algorithm 2 The networked actor-critic algorithm based on state-value TD-error

Input: Initial values of μ_0^i, $\widetilde{\mu}_0^i$, v_0^i, $\widetilde{v}_0^i$, λ_0^i, $\widetilde{\lambda}_0^i$, θ_0^i, $\forall i \in \mathcal{N}$; the initial state s_0 of the MDP, and stepsizes $\{\beta_{v,t}\}_{t\geq 0}$ and $\{\beta_{\theta,t}\}_{t\geq 0}$.

Critic

- μ_t^i - Estimate of averaged return J
- λ_t^i - Parameter for approximating the **global averaged reward function R**
- v_t^i - Parameter for approximating the optimal **state-value function V**

δ_t^i - **State-value** temporal difference error

Actor

- ψ_t^i - Sample of the score function (policy gradient)
- θ_t^i - Parameter for approximating the optimal policy π

Source K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, “Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents,” 2018.

Networked actor-critic algorithms (Algorithm 2)

Algorithm 2 The networked actor-critic algorithm based on state-value TD-error

Input: Initial values of μ_0^i, $\widetilde{\mu}_0^i$, v_0^i, $\widetilde{v}_0^i$, λ_0^i, $\widetilde{\lambda}_0^i$, θ_0^i, $\forall i \in \mathcal{N}$; the initial state s_0 of the MDP, and stepsizes $\{\beta_{v,t}\}_{t\geq0}$ and $\{\beta_{\theta,t}\}_{t\geq0}$.
Each agent i implements $a_0^i \sim \pi_{\theta_0^i}(s_0, \cdot)$.
Initialize the step counter $t \leftarrow 0$.
Repeat:
  for all $i \in \mathcal{N}$ **do**
    Observe state s_{t+1}, and reward r_{t+1}^i.
  end for
  Update the iteration counter $t \leftarrow t + 1$.
Until Convergence

Critic:
- μ_t^i - Estimate of averaged return J
- λ_t^i - Parameter for approximating the **global averaged reward function R**
- v_t^i - Parameter for approximating the optimal **state-value function V**

δ_t^i - **State-value** temporal difference error

Actor:
- ψ_t^i - Sample of the score function (policy gradient)
- θ_t^i - Parameter for approximating the optimal policy π

Source K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, “Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents,” 2018.

Networked actor-critic algorithms (Algorithm 2)

Algorithm 2 The networked actor-critic algorithm based on state-value TD-error

Input: Initial values of μ_0^i, $\widetilde{\mu}_0^i$, v_0^i, $\widetilde{v}_0^i$, λ_0^i, $\widetilde{\lambda}_0^i$, θ_0^i, $\forall i \in \mathcal{N}$; the initial state s_0 of the MDP, and stepsizes $\{\beta_{v,t}\}_{t\geq 0}$ and $\{\beta_{\theta,t}\}_{t\geq 0}$.
Each agent i implements $a_0^i \sim \pi_{\theta_0^i}(s_0, \cdot)$.
Initialize the step counter $t \leftarrow 0$.
Repeat:
  for all $i \in \mathcal{N}$ **do**
    Observe state s_{t+1}, and reward r_{t+1}^i.
    Update $\widetilde{\mu}_t^i \leftarrow (1-\beta_{v,t}) \cdot \mu_t^i + \beta_{v,t} \cdot r_{t+1}^i$, $\widetilde{\lambda}_t^i \leftarrow \lambda_t^i + \beta_{v,t} \cdot [r_{t+1}^i - \overline{R}_t(\lambda_t^i)] \cdot \nabla_\lambda \overline{R}_t(\lambda_t^i)$.
    Update $\delta_t^i \leftarrow r_{t+1}^i - \mu_t^i + V_{t+1}(v_t^i) - V_t(v_t^i)$
    Critic step: $\widetilde{v}_t^i \leftarrow v_t^i + \beta_{v,t} \cdot \delta_t^i \cdot \nabla_v V_t(v_t^i)$.
  end for
  Update the iteration counter $t \leftarrow t+1$.
Until Convergence

Critic:
- μ_t^i - Estimate of averaged return J
- λ_t^i - Parameter for approximating the **global averaged reward function R**
- v_t^i - Parameter for approximating the optimal **state-value function V**

δ_t^i - **State-value** temporal difference error

Actor:
- ψ_t^i - Sample of the score function (policy gradient)
- θ_t^i - Parameter for approximating the optimal policy π

Source K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, “Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents,” 2018.

Networked actor-critic algorithms (Algorithm 2)

Algorithm 2 The networked actor-critic algorithm based on state-value TD-error

Input: Initial values of μ_0^i, $\widetilde{\mu}_0^i$, v_0^i, $\widetilde{v}_0^i$, λ_0^i, $\widetilde{\lambda}_0^i$, θ_0^i, $\forall i \in \mathcal{N}$; the initial state s_0 of the MDP, and stepsizes $\{\beta_{v,t}\}_{t\geq 0}$ and $\{\beta_{\theta,t}\}_{t\geq 0}$.
Each agent i implements $a_0^i \sim \pi_{\theta_0^i}(s_0, \cdot)$.
Initialize the step counter $t \leftarrow 0$.
Repeat:
  for all $i \in \mathcal{N}$ **do**
    Observe state s_{t+1}, and reward r_{t+1}^i.
    Update $\widetilde{\mu}_t^i \leftarrow (1-\beta_{v,t}) \cdot \mu_t^i + \beta_{v,t} \cdot r_{t+1}^i$, $\quad \widetilde{\lambda}_t^i \leftarrow \lambda_t^i + \beta_{v,t} \cdot [r_{t+1}^i - \overline{R}_t(\lambda_t^i)] \cdot \nabla_\lambda \overline{R}_t(\lambda_t^i)$.
    Update $\delta_t^i \leftarrow r_{t+1}^i - \mu_t^i + V_{t+1}(v_t^i) - V_t(v_t^i)$
    Critic step: $\widetilde{v}_t^i \leftarrow v_t^i + \beta_{v,t} \cdot \delta_t^i \cdot \nabla_v V_t(v_t^i)$.
    Update $\widetilde{\delta}_t^i \leftarrow \overline{R}_t(\lambda_t^i) - \mu_t^i + V_{t+1}(v_t^i) - V_t(v_t^i)$, $\quad \psi_t^i \leftarrow \nabla_{\theta^i} \log \pi_{\theta_t^i}^i(s_t, a_t^i)$.
    Actor step: $\theta_{t+1}^i = \theta_t^i + \beta_{\theta,t} \cdot \widetilde{\delta}_t^i \cdot \psi_t^i$.
  end for
  Update the iteration counter $t \leftarrow t+1$.
Until Convergence

$\text{Critic}\begin{cases} \mu_t^i \text{ - Estimate of averaged return J} \\ \lambda_t^i \text{ - Parameter for approximating the } \textbf{global averaged reward function R} \\ v_t^i \text{ - Parameter for approximating the optimal } \textbf{state-value function V} \end{cases}$

δ_t^i - **State-value** temporal difference error

$\text{Actor}\begin{cases} \psi_t^i \text{ - Sample of the score function (policy gradient)} \\ \theta_t^i \text{ - Parameter for approximating the optimal policy } \pi \end{cases}$

Source K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, “Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents,” 2018.

Networked actor-critic algorithms (Algorithm 2)

Algorithm 2 The networked actor-critic algorithm based on state-value TD-error

Input: Initial values of μ_0^i, $\widetilde{\mu}_0^i$, v_0^i, $\widetilde{v}_0^i$, λ_0^i, $\widetilde{\lambda}_0^i$, θ_0^i, $\forall i \in \mathcal{N}$; the initial state s_0 of the MDP, and stepsizes $\{\beta_{v,t}\}_{t\geq 0}$ and $\{\beta_{\theta,t}\}_{t\geq 0}$.
Each agent i implements $a_0^i \sim \pi_{\theta_0^i}(s_0, \cdot)$.
Initialize the step counter $t \leftarrow 0$.
Repeat:
  for all $i \in \mathcal{N}$ **do**
    Observe state s_{t+1}, and reward r_{t+1}^i.
    Update $\widetilde{\mu}_t^i \leftarrow (1-\beta_{v,t}) \cdot \mu_t^i + \beta_{v,t} \cdot r_{t+1}^i$, $\widetilde{\lambda}_t^i \leftarrow \lambda_t^i + \beta_{v,t} \cdot [r_{t+1}^i - \overline{R}_t(\lambda_t^i)] \cdot \nabla_\lambda \overline{R}_t(\lambda_t^i)$.
    Update $\delta_t^i \leftarrow r_{t+1}^i - \mu_t^i + V_{t+1}(v_t^i) - V_t(v_t^i)$
    Critic step: $\widetilde{v}_t^i \leftarrow v_t^i + \beta_{v,t} \cdot \delta_t^i \cdot \nabla_v V_t(v_t^i)$.
    Update $\widetilde{\delta}_t^i \leftarrow \overline{R}_t(\lambda_t^i) - \mu_t^i + V_{t+1}(v_t^i) - V_t(v_t^i)$, $\psi_t^i \leftarrow \nabla_{\theta^i} \log \pi_{\theta_t^i}^i(s_t, a_t^i)$.
    Actor step: $\theta_{t+1}^i = \theta_t^i + \beta_{\theta,t} \cdot \widetilde{\delta}_t^i \cdot \psi_t^i$.
    Send $\widetilde{\mu}_t^i$, $\widetilde{\lambda}_t^i$, $\widetilde{v}_t^i$ to the neighbors over $\mathcal{G}_t$.
  end for
  for all $i \in \mathcal{N}$ **do**
    Consensus step: $\mu_{t+1}^i \leftarrow \sum_{j\in\mathcal{N}} c_t(i,j) \cdot \widetilde{\mu}_t^j$, $\lambda_{t+1}^i \leftarrow \sum_{j\in\mathcal{N}} c_t(i,j) \cdot \widetilde{\lambda}_t^j$, $v_{t+1}^i \leftarrow \sum_{j\in\mathcal{N}} c_t(i,j) \cdot \widetilde{v}_t^j$.
  end for
  Update the iteration counter $t \leftarrow t+1$.
Until Convergence

Critic

μ_t^i - Estimate of averaged return J

λ_t^i - Parameter for approximating the **global averaged reward function R**

v_t^i - Parameter for approximating the optimal **state-value function V**

δ_t^i - **State-value** temporal difference error

Actor

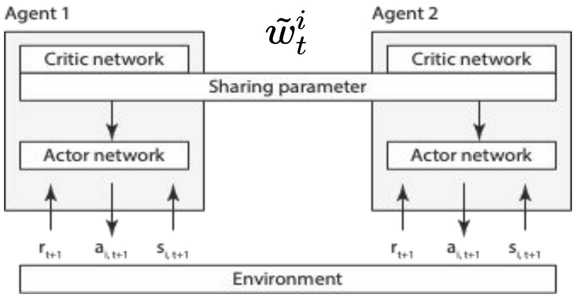
ψ_t^i - Sample of the score function (policy gradient)

θ_t^i - Parameter for approximating the optimal policy π

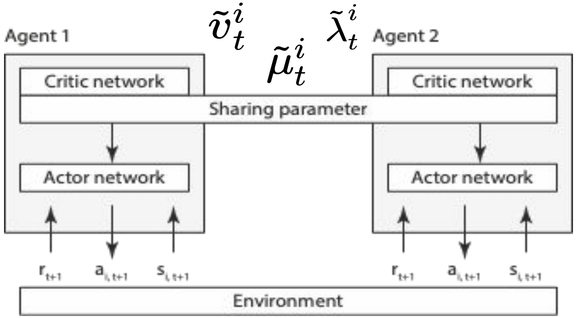
Source K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, “Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents,” 2018.

Algorithms - Consensus requirements

Algorithm 1



Algorithm 2

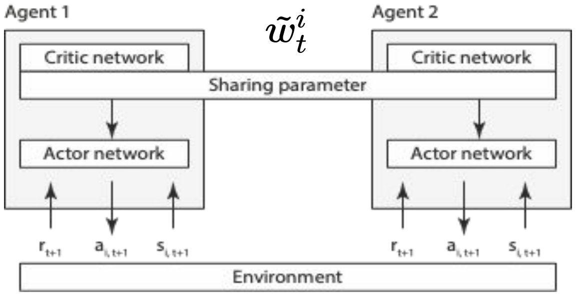


Policy & rewards are private

Algorithms - Consensus requirements

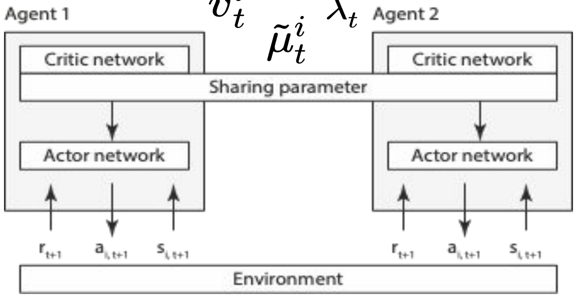
Algorithm 1

$\tilde{w}_t^i$



Algorithm 2

$\tilde{v}_t^i$ $\tilde{\mu}_t^i$ $\tilde{\lambda}_t^i$



Policy & rewards are private

For consensus (e.g. $\omega_{t+1}^i \leftarrow \sum_{j\in\mathcal{N}} c_t(i,j)\cdot\widetilde{\omega}_t^j.$), message-weight can be *Metropolis weights*:

$$c_t(i,j) = \left\{1+\max[d_t(i), d_t(j)]\right\}^{-1}, \ \ \forall (i,j)\in\mathcal{E}_t$$

$$c_t(i,i) = 1 - \sum_{j\in\mathcal{N}_t(i)} c_t(i,j), \ \ \forall i\in\mathcal{N},$$

where

$$d_t(i) = |\mathcal{N}_t(i)|$$

$$\mathcal{N}_t(i) = \{j\in\mathcal{N} : (i,j)\in\mathcal{E}_t\}$$

Numerical results

Linear function approximator

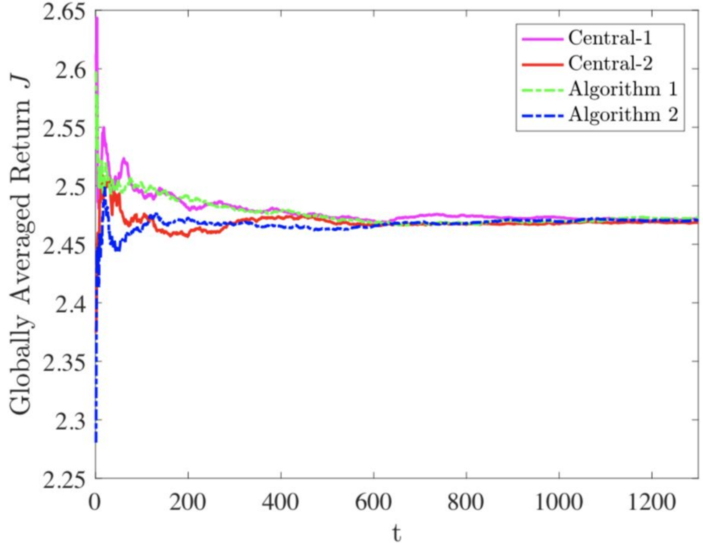


Figure 1: The convergence of globally averaged returns, when linear function approximation is used. We plot the returns achieved by both Algorithm 1 and Algorithm 2, along with their centralized counterparts Central-1 and Central-2.

20 agents, with binary actions.
Graph is randomly generated with connectivity ratio 4/N
There are $|\mathcal{S}| = 20$ states.
Transition matrix is stochastic.
Reward is sampled differently for each agent.

Feature vector is uniformly sampled from [0,1].
Feature vector dimension is 10 for $Q(\omega)$, 5 for $V(v)$, 10 for $\bar{R}(\lambda)$
Stepsizes are $\beta_{\omega,t} = \beta_{v,t} = 1/t^{0.65}$ $\beta_{\theta,t} = 1/t^{0.85}$

Source K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, "Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents," 2018.

Numerical results

Non-linear function approximator

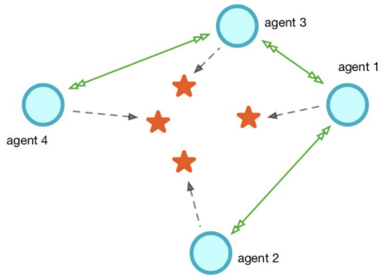


Figure 4: Illustration of the experimental environment for the Cooperative Navigation task we consider, modified from Lowe et al. (2017). In particular, the blue circles represent the agents, the orange stars represent the landmarks, the green arrows represent the communication links between agents, and the gray arrows show the target landmark each agent need to cover.

Modified cooperative navigation use-case.

Agents can observe the global state. Reward is individual, which is a function of distance to target landmark and a penalty depending on distance to other agents.

Target landmark is also individual. Time varying communication graph.

Numerical results

Non-linear function approximator

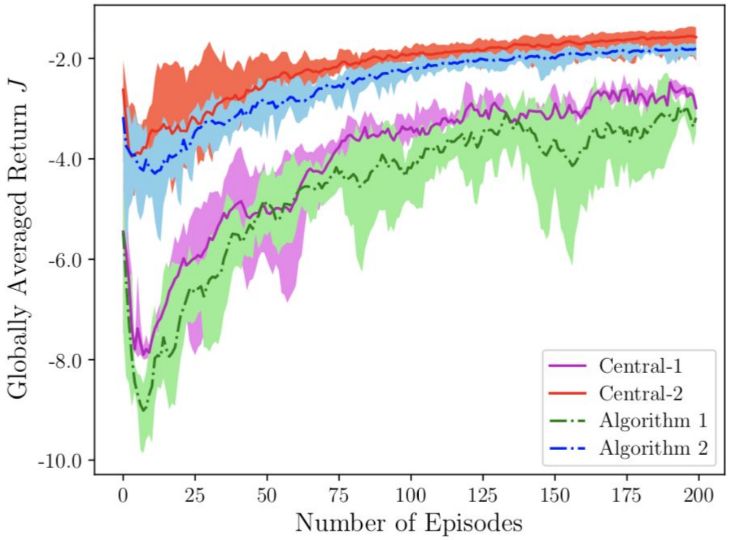


Figure 5: The globally averaged returns for the task of Cooperative Navigation, when neural networks are used for function approximation. We plot the returns achieved by both Algorithm 1 and Algorithm 2, along with their centralized counterparts Central-1 and Central-2.

N=L=10, i.e. 10 agents each with their own target.

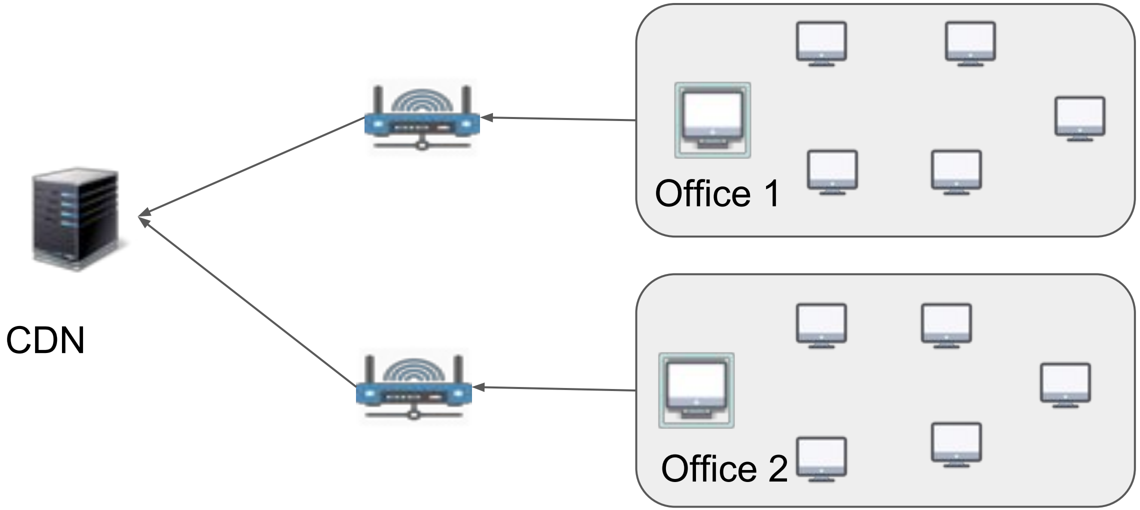
Actions are move (N,W,S,E) or stay.

State dimension is 2(N+L) = 40

Actor and critic neural networks with one hidden layer (24 hidden units) and ReLU.

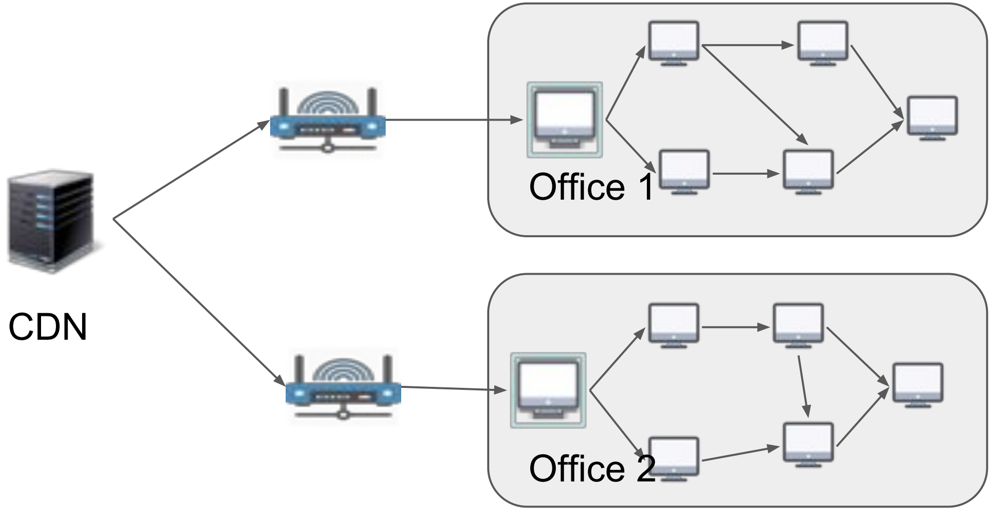
Use Case - Enterprise Video Streaming

1. The future of enterprise communication is **high quality video**
2. Corporate networks can't handle the load

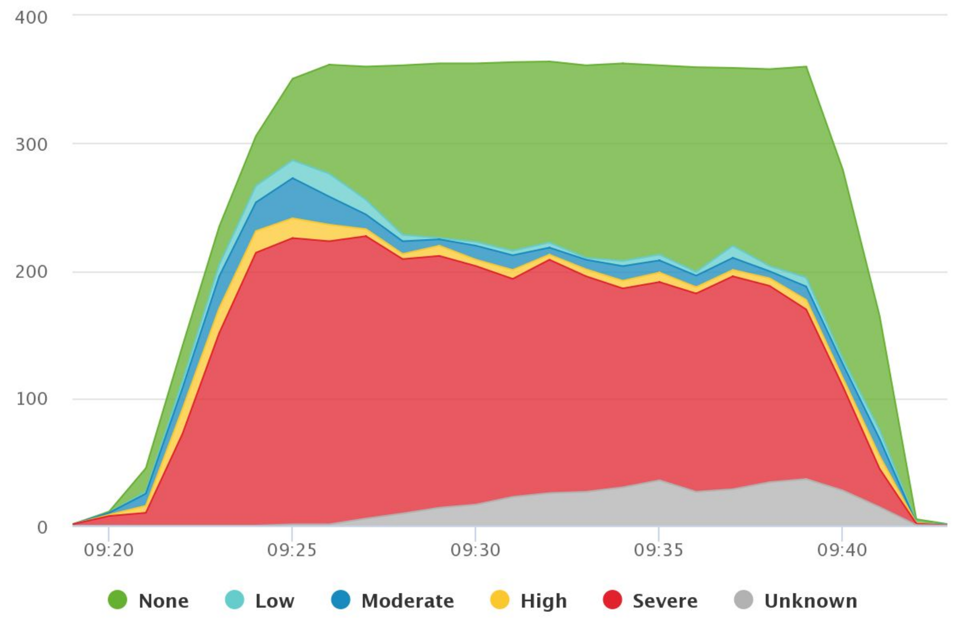


Use Case - Enterprise Video Streaming

- **Why P2P?**
  - More bandwidth inside the office
- **Peer-Assisted Video Streaming**
  - One leader per office
  - Promote P2P within office
  - Control P2P between offices
  - Fallback to CDN iff P2P fails
- **Results**
  - Less requests to CDN
  - **No network congestion**



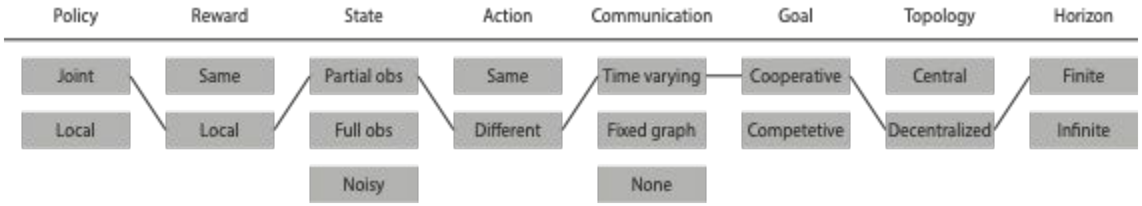
Use case - Enterprise Video Streaming



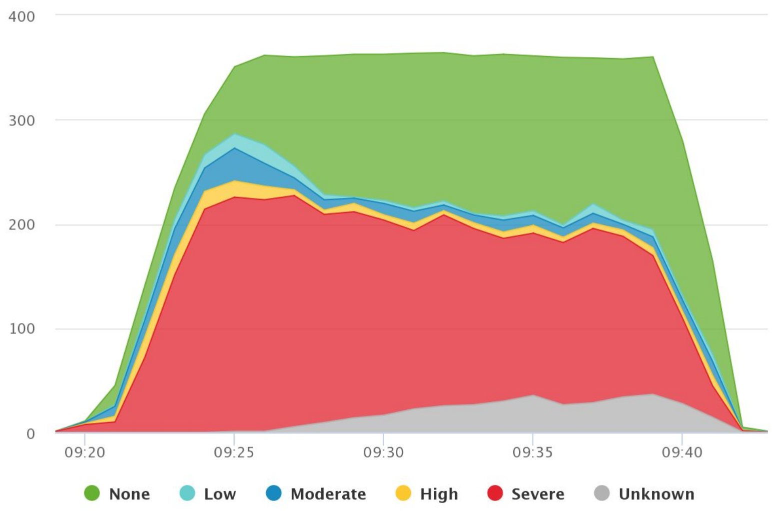
Use Case - Enterprise Video Streaming

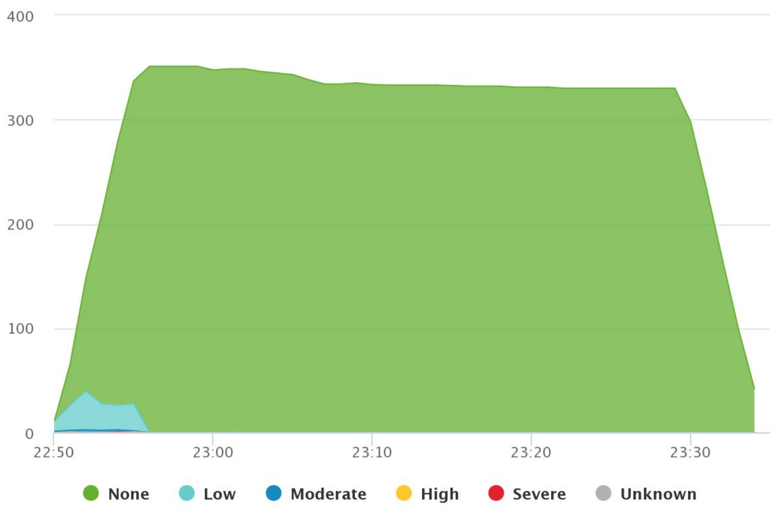
$$< \mathcal{I}, \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \Omega, \mathcal{O}, \gamma >$$

- $\mathcal{I}$ = agents watching the same video
- $\mathcal{S}$ = [Idle, Playing, Paused, Stopped, Buffering]
- $\mathcal{A}$ = [CDN Request, P2P Request, Bitrate up, Bitrate down, Increase Partnership, Decrease Partnership,...]
- $R = max(\sum_j^{\mathcal{I}} (\sum_i^N q_j(i) + \sum_i^N b_j(i) + \sum_i^N s_j(i)))$



Use Case - Enterprise Video Streaming





Further Discussions and Future Works

- Sparse and Delayed rewards
- Self-play
- Scalability
- Network Topology

References

[1] M. Lanctot *et al.*, “A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning,” *Adv. Neural Inf. Process. Syst. 30*, no. Nips, 2017.
[2] K. Zhang, Z. Yang, H. Liu, T. Zhang, and T. Başar, “Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents,” 2018.
[3] H.-T. Wai, Z. Yang, Z. Wang, and M. Hong, “Multi-Agent Reinforcement Learning via Double Averaging Primal-Dual Optimization,” 2018.
[4] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, “Deep Decentralized Multi-task Multi-Agent Reinforcement Learning under Partial Observability,” 2017.
[5] L. Matignon, G. J. Laurent, and N. Le Fort-Piat, “Hysteretic Q-Learning : An algorithm for decentralized reinforcement learning in cooperative multi-agent teams,” *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 64–69, 2007.
[6] S. Kapoor, “Multi-Agent Reinforcement Learning: A Report on Challenges and Approaches,” pp. 1–24, 2018.
[7] Y. Li, “Deep Reinforcement Learning: An Overview,” pp. 1–70, 2017.
[8] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “Deep reinforcement learning: A brief survey,” *IEEE Signal Process. Mag.*, vol. 34, no. 6, pp. 26–38, 2017.
[9] D. Lee, H. Yoon, and N. Hovakimyan, “Primal-Dual Algorithm for Distributed Reinforcement Learning: Distributed GTD2,” 2018.
[10] Du, Simon S., Jianshu Chen, Lihong Li, Lin Xiao, and Dengyong Zhou. "Stochastic variance reduction methods for policy evaluation." In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1049-1058. JMLR. org, 2017.

Thank you!
Questions?