Optimization Meets Reinforcement Learning

Yingbin Liang, The Ohio State University
Shaofeng Zou, University at Buffalo, SUNY
Yi Zhou, University of Utah

ICASSP 2022

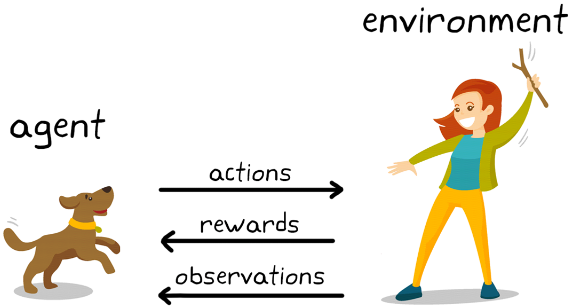
May 22, 2022

Outline

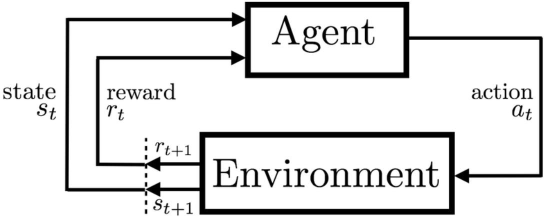
Outline

Reinforcement Learning

- An agent learns to interact with environment in the best way
  - Agent observes state, and takes an action based on a policy
  - Agent receives a reward
  - Environment changes the state
  - Agent finds a policy to maximize reward

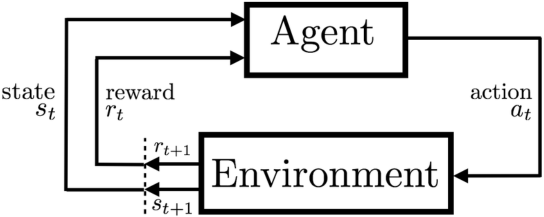


Markov Decision Process (MDP)



- Markov decision process (MDP): $(\mathcal{S}, \mathcal{A}, r, \mathrm{P})$
  - $\mathcal{S}$ and $\mathcal{A}$: state and action spaces
  - $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$: reward function
  - $\mathrm{P}(s'|s, a)$: transition kernel; prob of $s \to s'$ given action a
- Agent's policy $\pi(a|s)$: prob of selecting action a in state s

Markov Decision Process (MDP)



- Markov decision process (MDP): $(\mathcal{S}, \mathcal{A}, r, \mathrm{P})$
  - $\mathcal{S}$ and $\mathcal{A}$: state and action spaces
  - $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$: reward function
  - $\mathrm{P}(s'|s, a)$: transition kernel; prob of $s \to s'$ given action a
- Agent's policy $\pi(a|s)$: prob of selecting action a in state s

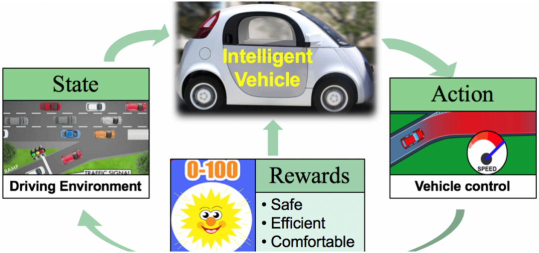
MDP trajectory $\{s_t, a_t, r_t, s_{t+1}\}_{t=0}^{\infty}$ defined by

$$s_0 \xrightarrow{\pi(\cdot|s_0)} a_0 \xrightarrow{\mathrm{P}(\cdot|s_0, a_0)} (s_1, r_0) \xrightarrow{\pi(\cdot|s_1)} a_1 \cdots$$

- Randomness: actions, state transitions

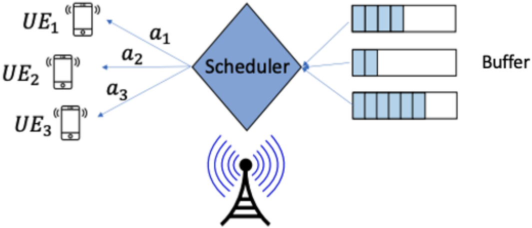
Application: Autonomous Driving

- Collects driving data
- AI agent trained to optimize driving control
- Specification of MDP
  - State: driving environment (distance to nearby cars, weather, etc)
  - Action: turn left/right, accelerate, brake
  - Reward: stay safe, drive smoothly
  - Policy: vehicle control in a state



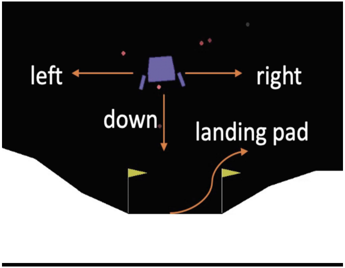
Application: Wireless Communication

- Downlink Scheduling [1]
- Learn optimal scheduling to minimize average queuing delay
- Specification of MDP
  - State: buffer status and channel state
  - Action: assign resource block, determine number of transmitted bits
  - Reward: buffer cost
  - Policy: determine action in a given state



Application: Robotics

- Robotics: Robot Control (left figure)
  - Robot learns the landing environment
  - Robot follows a policy to adjust the landing direction
- Robotics: Arm Manipulation (right figure)
  - Robot learns the warehouse environment
  - Robot follows a policy to manipulate its arm



Formulation of RL

- MDP trajectory $\{s_t, a_t, r_t, s_{t+1}\}_t$ with $r_t := r(s_t, a_t, s_{t+1})$
- Quality of s, a: discount factor $\gamma \in (0, 1)$

$$\text{(State value):} \quad V_\pi(s) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, \pi\Big]$$

$$\text{(State-action value):} \quad Q_\pi(s, a) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a, \pi\Big]$$

- Expected long-term accumulated reward start with s, a

Formulation of RL

- MDP trajectory $\{s_t, a_t, r_t, s_{t+1}\}_t$ with $r_t := r(s_t, a_t, s_{t+1})$
- Quality of s, a: discount factor $\gamma \in (0, 1)$

$$\text{(State value):} \quad V_\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, \pi\right]$$
$$\text{(State-action value):} \quad Q_\pi(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, a_0 = a, \pi\right]$$

- Expected long-term accumulated reward start with s, a

RL Goal: find the best policy π^*

$$\text{(Criterion I):} \quad V_{\pi^*}(s) \geq V_\pi(s), \quad \forall \pi, \forall s$$
$$\text{(Criterion II):} \quad \max_\pi J(\pi) := \mathbb{E}_{s\sim\xi}[V_\pi(s)]$$

Tutorial will not cover all the RL formulations

- Finite-time horizon, Average reward, Regret analysis

Outline

Outline

Policy Evaluation

- Recall Markov Decision Process: $\{s_t, a_t, r_t, s_{t+1}\}_t$

$$s_0 \xrightarrow{\pi(\cdot|s_0)} a_0 \xrightarrow{\mathrm{P}(\cdot|s_0,a_0)} (s_1, r_0) \xrightarrow{\pi(\cdot|s_1)} a_1 \cdots$$

- State value function:

$$V_\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, \pi\right]$$

  - Expected accumulated reward, start with s follow π.

Policy Evaluation

- Recall Markov Decision Process: $\{s_t, a_t, r_t, s_{t+1}\}_t$

$$s_0 \xrightarrow{\pi(\cdot|s_0)} a_0 \xrightarrow{\mathrm{P}(\cdot|s_0,a_0)} (s_1, r_0) \xrightarrow{\pi(\cdot|s_1)} a_1 \cdots$$

- State value function:

$$V_\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, \pi\right]$$

  - Expected accumulated reward, start with s follow π.

Policy Evaluation Problem:

Given a fixed policy π, how to evaluate its state value function V_π?

- Foundation for policy optimization

Summary of Policy Evaluation Algorithms

- Known transition kernel $P(\cdot|s, a)$
  - Solving Bellman equation
- Unknown transition kernel $P(\cdot|s, a)$ (Model-free)
  - On-policy TD learning
  - Off-policy TD learning

Summary of Policy Evaluation Algorithms

- Known transition kernel $P(\cdot|s,a)$
  - Solving Bellman equation
- Unknown transition kernel $P(\cdot|s,a)$ (Model-free)
  - On-policy TD learning
  - Off-policy TD learning

Our focus is model-free approaches.

Known P: Bellman Equation

Transition kernel $\mathsf{P}(\cdot|s,a)$ is known

- By definition of $V_\pi(s)$:

$$
\begin{aligned}
V_\pi(s) &= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots | s_0 = s, \pi] \\
&= \mathbb{E}[r_0 | s_0 = s, \pi] + \gamma \mathbb{E}[r_1 + \gamma r_2 + \cdots | s_0 = s, \pi]
\end{aligned}
$$

Known P: Bellman Equation

Transition kernel $\mathrm{P}(\cdot|s,a)$ is known

- By definition of $V_\pi(s)$:

$$\begin{aligned} V_\pi(s) &= \mathbb{E}[r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots | s_0 = s, \pi] \\ &= \mathbb{E}[r_0 | s_0 = s, \pi] + \gamma \mathbb{E}[r_1 + \gamma r_2 + \cdots | s_0 = s, \pi] \end{aligned}$$

- Note that

$$\begin{aligned} &\mathbb{E}[r_1 + \gamma r_2 + \cdots | s_0 = s, \pi] \\ &= \mathbb{E}_{s_1}\Big[\mathbb{E}[r_1 + \gamma r_2 + \cdots | s_0 = s, s_1 = s', \pi]\Big] \\ &= \mathbb{E}_{s_1}[V_\pi(s_1)] \end{aligned}$$

$$V_\pi(s) = \sum_{a,s'} \mathrm{P}(s'|s,a)\pi(a|s)\Big(r(s,a,s') + \gamma V_\pi(s')\Big)$$

$$V_\pi(s) = \sum_{a,s'} \mathrm{P}(s'|s,a)\pi(a|s)\Big(r(s,a,s') + \gamma V_\pi(s')\Big)$$

- Define Bellman operator

(**Bellman operator**):

$$\mathrm{T}_\pi V_\pi(s) = \sum_{a,s'} \mathrm{P}(s'|s,a)\pi(a|s)\Big(r(s,a,s') + \gamma V_\pi(s')\Big)$$

Bellman Equation for Value Function

$$V_\pi(s) = \mathrm{T}_\pi V_\pi(s)$$

$$V_\pi(s) = \sum_{a,s'} \mathrm{P}(s'|s,a)\pi(a|s)\Big(r(s,a,s') + \gamma V_\pi(s')\Big)$$

- Define Bellman operator

(**Bellman operator**):

$$\mathrm{T}_\pi V_\pi(s) = \sum_{a,s'} \mathrm{P}(s'|s,a)\pi(a|s)\Big(r(s,a,s') + \gamma V_\pi(s')\Big)$$

Bellman Equation for Value Function

$$V_\pi(s) = \mathrm{T}_\pi V_\pi(s)$$

- Linear programming: Directly solve the linear equation
  - High computation complexity
- Value iteration: fixed point update

$$V_{t+1}(s) = \mathrm{T}_\pi V_t(s)$$

  - T_π is contraction $\Rightarrow V_t \to V_\pi$.

Model-Free: On-Policy TD Learning

Model-Free

- Transition kernel $\mathrm{P}(\cdot|s, a)$ is unknown

On-Policy Data

- Collect Markovian data $\{s_t, a_t, r_t, s_{t+1}\}_t$ following target policy π

On-Policy TD(0) Algorithm

- Recall Bellman equation

$$V_\pi(s) = \mathbb{E}[r(s, a, s') + \gamma V_\pi(s')]$$

- Idea: update $V_\pi(s)$ using $r(s, a, s') + \gamma V_\pi(s')$

On-Policy TD(0) Algorithm

- Recall Bellman equation

$$V_\pi(s) = \mathbb{E}[r(s,a,s') + \gamma V_\pi(s')]$$

- Idea: update $V_\pi(s)$ using $r(s,a,s') + \gamma V_\pi(s')$
- Formally: collect $\{s_t, a_t, r_t, s_{t+1}\}_t$ and do

$$V(s_t) = \underbrace{r_{t+1} + \gamma V(s_{t+1})}_{\text{Target (one-step bootstrap)}}, \qquad (*)$$

- TD learning is a damped version of (*): $0 < \eta < 1$,

$$V(s_t) \leftarrow (1-\eta)V(s_t) + \eta\big(r_{t+1} + \gamma V(s_{t+1})\big), \qquad \text{(TD)}$$

TD(0) Algorithm [2]

$$V(s_t) \leftarrow V(s_t) + \eta\big(\underbrace{r_{t+1} + \gamma V(s_{t+1}) - V(s_t)}_{\text{temporal difference}}\big)$$

TD(λ) Algorithm

TD(0) Algorithm

$$V(s_t) \leftarrow V(s_t) + \eta\big(r_{t+1} + \gamma V(s_{t+1}) - V(s_t)\big)$$

- In TD(0), target $r_{t+1} + \gamma V(s_{t+1})$ is one-step bootstrap
- Extension: n-step bootstrap

$$G_t^{(n)} := r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n V(s_{t+n})$$

- Define λ-return: $G_t^\lambda := (1-\lambda)\sum_{n=1}^{\infty} \lambda^{n-1} G_t^{(n)}$.

TD(λ) Algorithm [3]

$$V(s_t) \leftarrow V(s_t) + \eta\big(G_t^\lambda - V(s_t)\big)$$

- Reduce the variance of TD target

Value Function Approximation

- **Curse of dimensionality:** state space is often large or infinite
- **Solution:** approximate V_π using parameterized model V_θ
  - Linear model: $V_\theta(s) := \phi_s^\top \theta$, where ϕ_s is feature vector of s
  - Neural model: $V_\theta(s) := \mathrm{NN}_\theta(s)$, where NN_θ is neural network

Value Function Approximation

- **Curse of dimensionality:** state space is often large or infinite
- **Solution:** approximate V_π using parameterized model V_θ
  - Linear model: $V_\theta(s) := \phi_s^\top \theta$, where ϕ_s is feature vector of s
  - Neural model: $V_\theta(s) := \mathrm{NN}_\theta(s)$, where NN_θ is neural network

TD(0) learning with function approximation

- Initialize model θ_0.
- Observe sample $\{s_t, a_t, r_t, s_{t+1}\}$, define target $G_t = r_t + \gamma V_{\theta_t}(s_{t+1})$
- Define loss $\ell_t(\theta) := \frac{1}{2}(V_\theta(s_t) - G_t)^2$, compute $g_t(\theta_t) = -\frac{\partial \ell_t(\theta)}{\partial \theta}|_{\theta=\theta_t}$
- TD update:

$$\theta_{t+1} = \theta_t + \eta g_t(\theta_t),$$

where $g_t(\theta_t) = (r_t + \gamma V_{\theta_t}(s_{t+1}) - V_{\theta_t}(s_t))\nabla V_{\theta_t}(s_t)$

Analysis of TD(0) with Linear Approximation

TD(0) with linear approximation $V_\theta(s) := \phi_s^\top \theta$

$$\theta_{t+1} = \mathrm{Proj}_R\big(\theta_t + \eta g_t(\theta_t)\big),$$
$$\text{where } g_t(\theta_t) = (r_t + \gamma \phi_{s_{t+1}}^\top \theta_t - \phi_{s_t}^\top \theta_t)\phi_{s_t}$$

- **Challenge:** $g_t(\theta_t)$ is gradient of time-varying function ℓ_t
- **Challenge:** Samples $\{s_t, a_t, r_t, s_{t+1}\}_t$ are Markovian and correlated

Non-exhaustive summary of existing work:

- Asymptotic convergence: [4, 5, 6, 7]
- Non-asymptotic (finite-time) convergence
  - I.I.D. samples: [8]
  - Markovian samples: [9], [10] (will be presented)

Finite-Time Convergence of TD(0)

Key Assumption: Geometric Mixing

State stationary distribution μ. There exist $\kappa > 0$, $\rho \in (0, 1)$ such that

$$\sup_{s \in \mathcal{S}} \mathrm{d}_{TV}\left(\mathsf{P}(s_t | s_0 = s), \mu\right) \leq \kappa \rho^t, \quad \forall t \in \mathbb{N}_0$$

Finite-Time Convergence of TD(0)

Key Assumption: Geometric Mixing

State stationary distribution μ. There exist $\kappa > 0$, $\rho \in (0, 1)$ such that

$$\sup_{s \in \mathcal{S}} \mathrm{d}_{TV}\big(P(s_t | s_0 = s), \mu\big) \le \kappa \rho^t, \quad \forall t \in \mathbb{N}_0$$

- Hold for irreducible and aperiodic Markov chains
- Given s_0 and large t, s_t is almost like being sampled from μ

- Feature matrix $\Phi = [\phi_{s_1}^\top; \dots; \phi_{s_n}^\top]$ full column rank, $V_\theta = \Phi\theta$
- Solution point θ^* satisfies [4]

$$V_{\theta^*} = \Pi_{\mathcal{L}} \mathsf{T}_\pi V_{\theta^*}, \quad \text{where } \mathcal{L} = \{\Phi x | x \in \mathbb{R}^d\}$$

- Feature matrix $\Phi = [\phi_{s_1}^\top; \ldots; \phi_{s_n}^\top]$ full column rank, $V_\theta = \Phi\theta$
- Solution point θ^* satisfies [4]

$$V_{\theta^*} = \Pi_{\mathcal{L}} \mathsf{T}_\pi V_{\theta^*}, \quad \text{where } \mathcal{L} = \{\Phi x | x \in \mathbb{R}^d\}$$

Theorem: finite-time convergence [10]

Set learning rate $\eta \leq \mathcal{O}(\frac{1}{1-\gamma})$. After T iterations,

$$\mathbb{E}\left[\|\theta_T - \theta^*\|^2\right] \leq \mathcal{O}\Big(\exp(-c\eta T)\|\theta_0 - \theta^*\|^2 + \eta \frac{\tau_{\text{mix}}(\eta)}{1-\gamma}\Big),$$

where $\tau_{\text{mix}}(\eta) := \min\{t \mid \kappa\rho^t \leq \eta\}$ is the mixing time of Markov chain.

- A faster mixing implies smaller convergence error

TD Learning for Off-Policy Evaluation

- Previous TD(0) uses on-policy data

On-Policy Data

Collect Markovian data $\{s_t, a_t, r_t, s_{t+1}\}_t$ following target policy π

- Limitation: requires executing the target policy
- Limitation: in practice may not have sufficient on-policy data

TD Learning for Off-Policy Evaluation

- Previous TD(0) uses on-policy data

On-Policy Data

Collect Markovian data $\{s_t, a_t, r_t, s_{t+1}\}_t$ following target policy π

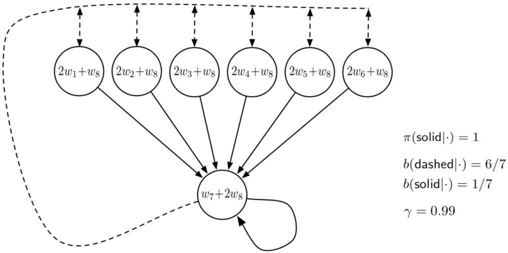
- Limitation: requires executing the target policy
- Limitation: in practice may not have sufficient on-policy data

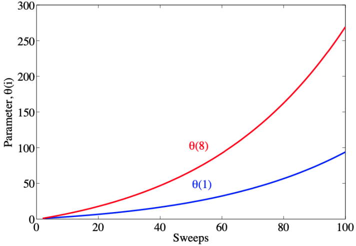
Off-policy data

Collect Markovian data $\{s_t, a_t, r_t, s_{t+1}\}_t$ following behavior policy π_b. The goal is to evaluate V_π of the target policy π.

Divergence of Off-Policy TD(0)

Key message: TD(0) with linear approximation may diverge in the off-policy setting [11]





- Zero reward, function approximation

$$V(s) = 2\theta(s) + \theta_0, \quad s = 1, \ldots, 6$$
$$V(7) = \theta(7) + 2\theta_0$$

- Under certain initialization, parameter diverges

Gradient TD for Off-Policy Evaluation

- Recall $V_\theta(s) = \phi_s^\top \theta$. Optimal θ^* satisfies

$$V_{\theta^*} = \Pi_{\mathcal{L}} \mathsf{T}^\pi V_{\theta^*}$$

- Data sampled by behavior policy π_b, stationary distribution μ_b

Gradient TD for Off-Policy Evaluation

- Recall $V_\theta(s) = \phi_s^\top \theta$. Optimal θ^* satisfies

$$V_{\theta^*} = \Pi_{\mathcal{L}} \mathrm{T}^\pi V_{\theta^*}$$

- Data sampled by behavior policy π_b, stationary distribution μ_b

Mean-square projected Bellman error (MSPBE) [12]

$$\text{(MSPBE): } J(\theta) := \mathbb{E}_{s \sim \mu_b} \left[V_\theta(s) - \Pi_{\mathcal{L}} \mathrm{T}^\pi V_\theta(s) \right]^2$$

- Error $V_\theta(s) - \Pi_{\mathcal{L}} \mathrm{T}^\pi V_\theta(s)$ based on target policy
- $\mathbb{E}_{s \sim \mu_b}$: stationary state distribution induced by behavior policy

Idea of Importance Sampling

- Denote TD error $\delta_t(\theta) = r_t + \gamma\phi_{s_{t+1}}^\top\theta - \phi_{s_t}^\top\theta$
- MSPBE can be rewritten as

$$J(\theta) = \mathbb{E}_{\mu_b,\pi}[\delta_t(\theta)\phi_{s_t}]^\top \mathbb{E}_{\mu_b}[\phi_{s_t}\phi_{s_t}^\top]^{-1}\mathbb{E}_{\mu_b,\pi}[\delta_t(\theta)\phi_{s_t}]$$

Importance Sampling Lemma

$$\mathbb{E}_{\mu_b,\pi}[\delta_t(\theta)\phi_{s_t}] = \mathbb{E}_{\mu_b,\pi_b}\left[\frac{\pi(a_t|s_t)}{\pi_b(a_t|s_t)}\delta_t(\theta)\phi_{s_t}\right],$$

where $\rho_t = \frac{\pi(a_t|s_t)}{\pi_b(a_t|s_t)}$ is the importance sampling ratio. Then, we have

$$-\frac{1}{2}\nabla J(\theta) = \mathbb{E}\left[\rho_t(\phi_{s_t} - \gamma\phi_{s_{t+1}})\phi_{s_t}^\top\right]\mathbb{E}\left[\phi_{s_t}\phi_{s_t}^\top\right]^{-1}\mathbb{E}[\rho_t\delta_t(\theta)\phi_{s_t}]$$

GTD2 Algorithm

$$-\frac{1}{2}\nabla J(\theta) = \mathbb{E}\big[\rho_t(\phi_{s_t} - \gamma\phi_{s_{t+1}})\phi_{s_t}^\top\big] \underbrace{\mathbb{E}\big[\phi_{s_t}\phi_{s_t}^\top\big]^{-1}\mathbb{E}[\rho_t\delta_t(\theta)\phi_{s_t}]}_{\omega^*(\theta)}$$

- $\omega^*(\theta)$ can be viewed as solution to the LMS

$$\text{(LMS): } \omega^*(\theta) = \operatorname*{argmin}_u \mathbb{E}\big[\phi_{s_t}^\top u - \rho_t\delta_t(\theta)\big]^2$$

GTD2 Algorithm

$$-\frac{1}{2}\nabla J(\theta) = \mathbb{E}\left[\rho_t(\phi_{s_t} - \gamma\phi_{s_{t+1}})\phi_{s_t}^\top\right] \underbrace{\mathbb{E}\left[\phi_{s_t}\phi_{s_t}^\top\right]^{-1}\mathbb{E}[\rho_t\delta_t(\theta)\phi_{s_t}]}_{\omega^*(\theta)}$$

- $\omega^*(\theta)$ can be viewed as solution to the LMS

$$\text{(LMS): } \omega^*(\theta) = \operatorname*{argmin}_u \mathbb{E}\left[\phi_{s_t}^\top u - \rho_t\delta_t(\theta)\right]^2$$

GTD2 algorithm [12]

$$\theta_{t+1} = \theta_t + \alpha_t\rho_t(\phi_{s_t} - \gamma\phi_{s_{t+1}})\phi_{s_t}^\top\omega_t$$
$$\omega_{t+1} = \omega_t + \beta_t(\rho_t\delta_t(\theta_t)\phi_{s_t} - \phi_{s_t}\phi_{s_t}^\top\omega_t)$$

- Two timescale updates
- ω update is one-step SGD applied to LMS

TDC Algorithm

$$-\frac{1}{2}\nabla J(\theta) = \mathbb{E}\left[\rho_t(\phi_{s_t} - \gamma\phi_{s_{t+1}})\phi_{s_t}^\top\right] \underbrace{\mathbb{E}\left[\phi_{s_t}\phi_{s_t}^\top\right]^{-1}\mathbb{E}[\rho_t\delta_t(\theta)\phi_{s_t}]}_{\omega^*(\theta)}$$

$$= \mathbb{E}\left[\rho_t\delta_t(\theta)\phi_{s_t}\right] - \gamma\mathbb{E}\left[\rho_t\phi_{s_{t+1}}\phi_{s_t}^\top\right]\omega^*(\theta)$$

TDC algorithm [12]

$$\theta_{t+1} = \theta_t + \alpha_t\rho_t(\delta_t(\theta_t)\phi_{s_t} - \gamma\phi_{s_{t+1}}\phi_{s_t}^\top\omega_t)$$
$$\omega_{t+1} = \omega_t + \beta_t(\rho_t\delta_t(\theta_t)\phi_{s_t} - \phi_{s_t}\phi_{s_t}^\top\omega_t)$$

- θ update is different from GTD2
- ω update is the same as GTD2

Convergence of TDC with Linear Approximation

TDC with linear approximation

$$\theta_{t+1} = \Pi_{R_\theta}\left(\theta_t + \alpha_t \rho_t(\delta_t(\theta_t)\phi_{s_t} - \gamma \phi_{s_{t+1}} \phi_{s_t}^\top \omega_t)\right)$$
$$\omega_{t+1} = \Pi_{R_\omega}\left(\omega_t + \beta_t(\rho_t \delta_t(\theta_t)\phi_{s_t} - \phi_{s_t}\phi_{s_t}^\top \omega_t)\right)$$

- Challenge: Correlated Markovian samples
- Challenge: Correlated two timescale updates

Non-exhaustive of existing work:

- Asymptotic convergence: [12, 13, 14]
- Non-asymptotic (finite-time) convergence
  - I.I.D. samples: [8]
  - Markovian samples: [15], [16] (will be presented)

Finite-Time Convergence of TDC

Key Assumptions:

- (Geometric mixing): There exist $\kappa > 0$, $\rho \in (0,1)$ such that

$$\sup_{s\in\mathcal{S}} \mathrm{d}_{TV}\big(\mathsf{P}(s_t|s_0 = s), \mu\big) \leq \kappa\rho^t, \quad \forall t \in \mathbb{N}_0$$

- (Non-singularity): The following matrices are non-singular

$$A := \mathbb{E}_{\mu_b}[\rho_{s,a}(\gamma\phi_s\phi_{s'}^\top - \phi_s\phi_s^\top)], \quad C := -\mathbb{E}_{\mu_b}[\phi_s\phi_s^\top]$$

Theorem: finite-time convergence [16]

Set learning rates $\alpha < \frac{1}{|\lambda_{\max}(2A^\top C^{-1}A)|}, \beta < \frac{1}{|\lambda_{\max}(2C)|}$. After T iterations,

$$\mathbb{E}\big[\|\theta_T - \theta^*\|^2\big] \leq \mathcal{O}\Big((1 - c\alpha)^t + \alpha \log \alpha^{-1} + \sqrt{\beta \log \beta^{-1} + \frac{\alpha}{\beta}}\Big)$$

- Need small $\frac{\alpha}{\beta}$: ω_t takes faster update than θ_t, because it needs to approximate the double expectation in θ update

Extension: Mini-batch TDC [17]

Mini-batch TDC with linear approximation

$$\theta_{t+1} = \theta_t + \frac{\alpha_t}{M} \sum_{i=tM}^{(t+1)M-1} \rho_i(\delta_i(\theta_t)\phi_{s_i} - \gamma\phi_{s_{i+1}}\phi_{s_i}^\top\omega_t)$$

$$\omega_{t+1} = \omega_t + \frac{\beta_t}{M} \sum_{i=tM}^{(t+1)M-1} (\rho_i\delta_i(\theta_t)\phi_{s_i} - \phi_{s_i}\phi_{s_i}^\top\omega_t)$$

- No need to use bounded projection
- Allow large constant learning rates
- Reduce variance of two timescale stochastic updates

Outline

Optimal Value Functions

- Recall definition of value and state-action value functions:

$$V_\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) \middle| s_0 = s, \pi\right]$$

$$Q_\pi(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) \middle| s_0 = s, a_0 = a, \pi\right]$$

- Goal: to find an optimal policy that maximizes the value function from any initial state s_0
- Optimal value function:

$$V^*(s) = \sup_\pi V_\pi(s), \forall s \in \mathcal{S}$$

- Optimal state-action value function:

$$Q^*(s, a) = \sup_\pi Q_\pi(s, a), \forall (s, a) \in \mathcal{S} \times \mathcal{A}$$

Bellman Operator and Contraction

- Optimal policy π^*: take action $\arg\max_{a \in \mathcal{A}} Q^*(s, a)$ at state $s \in \mathcal{S}$
- $V^*(s) = \max_{a \in \mathcal{A}} Q^*(s, a), \forall s \in \mathcal{S}$
- The Bellman operator T is defined as

$$(\mathrm{T}V)(s) = \max_{a \in \mathcal{A}} \mathbb{E}_{s' \sim \mathrm{P}(\cdot|s,a)} \left[r(s, a, s') + \gamma V(s') \right]$$

- T is contraction: for any V_1 and V_2

$$\|\mathrm{T}V_1 - \mathrm{T}V_2\|_\infty \leq \gamma \|V_1 - V_2\|_\infty$$

- V^* is the fixed point of T: $V^* = \mathrm{T}V^*$

Value Iteration

- Assume known reward r and transition kernel P

Value Iteration

- Initialize $V(s)$ arbitrarily for any $s \in \mathcal{S}$
- Repeat until convergence
  - $V(s) \leftarrow \max_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} \mathrm{P}(s'|s,a)(r(s,a,s') + \gamma V(s'))$, for all $s \in \mathcal{S}$

- Repeatedly update $V(s)$ using Bellman operator, i.e, $V \leftarrow \mathrm{T}V$
- Convergence can be proved using contraction of T
  - $\|\mathrm{T}V - V^*\|_\infty = \|\mathrm{T}V - \mathrm{T}V^*\|_\infty \le \gamma \|V - V^*\|_\infty$
  - $\|\underbrace{\mathrm{T}\cdots\mathrm{T}}_{t \text{ times}} V - V^*\|_\infty \le \gamma^t \|V - V^*\|_\infty \to 0$, as $t \to \infty$

Policy Iteration

- Assume known reward r and transition kernel P

Policy Iteration

- Initialize π arbitrarily
- Repeat until convergence
  - Evaluate Q_π
  - $\pi'(s) \leftarrow \arg\max_{a \in \mathcal{A}} Q_\pi(s, a)$ for all $s \in \mathcal{S}$
  - $\pi \leftarrow \pi'$

- **Policy improvement theorem**: Let π and π' be any pair of deterministic policies such that for all $s \in \mathcal{S}$, $Q_\pi(s, \pi'(s)) \geq V_\pi(s)$, then π' is no worse than π: $V_{\pi'}(s) \geq V_\pi(s), \forall s \in \mathcal{S}$
- Policy from policy iteration has higher or same value than before

SARSA: On-Policy TD Control

- Finite $\mathcal{S}$ and $\mathcal{A}$, unknown reward r and transition kernel P

SARSA

- Parameter: step size $\alpha \in (0, 1]$
- Initialize $Q(s, a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$ arbitrarily
- Initialize s_0 and a_0, $t = 0$
- Repeat until convergence
  - Observe state s_{t+1}, receive reward $r(s_t, a_t, s_{t+1})$
  - Take action a_{t+1} using target policy derived from Q (e.g., ϵ-greedy)
  - $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(\underbrace{r(s_t, a_t, s_{t+1}) + \gamma Q(s_{t+1}, a_{t+1})}_{\text{target, one-step bootstrap}} - Q(s_t, a_t))$
  - $t \leftarrow t + 1$

- SARSA converges to Q^* if
  - All state-action pairs are visited infinitely often
  - The policy converges to the greedy policy (e.g., ϵ-greedy with $\epsilon = 1/t$)

SARSA with Linear Function Approximation

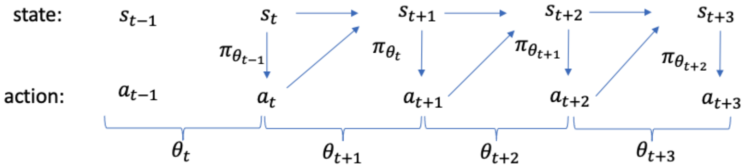
- Large $\mathcal{S}$ and $\mathcal{A}$, unknown r and P

SARSA

- Initialization: θ_0, s_0, ϕ_i, for $i = 1, 2, ..., N$
- $\pi_{\theta_0} \leftarrow \Gamma(\phi^\top \theta_0)$ (e.g., ϵ-greedy, softmax w.r.t. $\phi^\top \theta_0$)
- Choose a_0 according to π_{θ_0}
- For $t = 0, 1, 2, ...$
  - Observe s_{t+1} and $r(s_t, a_t, s_{t+1})$, choose a_{t+1} according to π_{θ_t}
  - $\theta_{t+1} \leftarrow \theta_t + \alpha_t g_t(\theta_t)$
  - **Policy improvement**: $\pi_{\theta_{t+1}} \leftarrow \Gamma(\phi^\top \theta_{t+1})$

- $g_t(\theta_t) = \phi(s_t, a_t)\Delta_t$: gradient of
$$\ell(\theta) = \frac{1}{2}(\underbrace{r(s_t, a_t, s_{t+1}) + \gamma\phi^\top(s_{t+1}, a_{t+1})\theta_t}_{\text{target, one-step bootstrap}} - \phi^\top\theta)^2$$
- Δ_t denotes the temporal difference error at time t:
$\Delta_t = \text{target} - \phi^\top(s_t, a_t)\theta_t$,

SARSA Sample Path



- As θ_t is updated, π_{θ_t} changes with time
- On-policy algorithm, time-varying policy
- Non-i.i.d. data

Finite-Sample Analysis [19]

- The limit point θ^* of the projected SARSA [18]: $A_{\theta^*}\theta^* + b_{\theta^*} = 0$, where $A_{\theta^*} = \mathbb{E}_{\theta^*}[\phi(s,a)(\gamma\phi^T(s',a') - \phi^\top(s,a)]$ and $b_{\theta^*} = \mathbb{E}_{\theta^*}[\phi(s,a)r(s,a,s')]$
- The limiting point θ^* is the one such that $\mathbb{E}_{\theta^*}[g(\theta^*)] = 0$, where $s \sim \mu_{\pi_{\theta^*}}$, $a \sim \pi_{\theta^*}(\cdot|s)$

Theorem

- Finite-sample bound on convergence of SARSA with diminishing step-size: $\mathbb{E}\|\theta_T - \theta^*\|_2^2 \leq \mathcal{O}\left(\frac{\log T}{T}\right)$
- Finite-sample bound on convergence of SARSA with constant step-size: $\mathbb{E}\|\theta_T - \theta^*\|_2^2 \leq \mathcal{O}\left(e^{-cT}\right) + \mathcal{O}(\alpha)$

- With diminishing step-size, SARSA converges exactly to optimal θ^*
- With constant step-size, SARSA converges exponentially fast to a small neighborhood of θ^*

Q-Learning: Off-Policy TD Control

- Finite $\mathcal{S}$ and $\mathcal{A}$, unknown r and P

Q-Learning

- Parameter: step size $\alpha \in (0, 1]$
- Initialize $Q(s, a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}$ arbitrarily
- Initialize s_0, behavior policy π_b, $t = 0$
- For $t = 0, 1, 2, \ldots$
  - Take action a_t following fixed π_b, observe next state s_{t+1}, receive reward $r(s_t, a_t, s_{t+1})$
  - $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(\underbrace{r(s_t, a_t, s_{t+1}) + \gamma \max_{a' \in \mathcal{A}} Q(s_{t+1}, a')}_{\text{target, one-step bootstrap}} - Q(s_t, a_t))$

- Q-learning converges to Q^* if all state-action pairs are visited infinitely often
- Q-learning sample complexity studies, e.g., [20], [21] and [22]
- Deep Q-learning: use neural network to approximate Q-function [23]

Gradient TD Method for Optimal Control

- Q-learning with function approximation may suffer from divergence issue
- Solution: Greedy-Gradient Q-learning (Greedy-GQ) with linear function approximation [24]
- Consider mean squared projected Bellman error (MSPBE):

$$J(\theta) \triangleq \|\Pi \mathrm{T} Q_\theta - Q_\theta\|_\mu^2$$

  - μ: stationary distribution induced by behavior policy π_b
  - $\|Q(\cdot,\cdot)\|_\mu \triangleq \int_{s\in\mathcal{S},a\in\mathcal{A}} d\mu_{s,a} Q(s,a)$
  - Π: projection operator $\Pi\hat{Q} = \arg\min_{Q\in\mathcal{Q}} \|Q-\hat{Q}\|_\mu$
  - $\mathcal{Q} = \left\{Q_\theta = \phi^\top\theta : \theta \in \mathbb{R}^N\right\}$

Goal: $\min_\theta J(\theta)$

Two Time-Scale Update Rule

- Define $\bar{V}_{s'}(\theta) = \max_{a' \in \mathcal{A}} \theta^\top \phi_{s',a'}$
- TD error: $\delta_{s,a,s'}(\theta) = r(s,a,s') + \gamma \bar{V}_{s'}(\theta) - \theta^\top \phi_{s,a}$
- Let $\hat{\phi}_{s'}(\theta) = \nabla \bar{V}_{s'}(\theta)$. Then gradient of MSPBE is

$$\frac{\nabla J(\theta)}{2} = -\mathbb{E}_\mu[\delta_{s,a,s'}(\theta)\phi_{s,a}] + \gamma \mathbb{E}_\mu[\hat{\phi}_{s'}(\theta)\phi_{s,a}^\top]\omega^*(\theta),$$

  where $\omega^*(\theta) = \mathbb{E}_\mu[\phi_{s,a}\phi_{s,a}^\top]^{-1}\mathbb{E}_\mu[\delta_{s,a,s'}(\theta)\phi_{s,a}]$.
- Double-sampling issue for estimating $\mathbb{E}_\mu[\hat{\phi}_{s'}(\theta)\phi_{s,a}^\top]\omega^*(\theta)$: it involves product of two expectations
- Weight doubling trick [12]:

$$\text{Slow time-scale: } \theta_{t+1} = \theta_t + \alpha(\delta_{t+1}(\theta_t)\phi_t - \gamma(\omega_t^\top \phi_t)\hat{\phi}_{t+1}(\theta_t)),$$

$$\text{Fast time-scale: } \omega_{t+1} = \omega_t + \beta(\delta_{t+1}(\theta_t) - \phi_t^\top \omega_t)\phi_t,$$

Finite-Sample Analysis [25, 26]

Challenges:

- Non-convex objective $J(\theta)$ with two time-scale update rule
- Non-smooth due to max in $\bar{V}_{s'}(\theta) = \max_{a' \in \mathcal{A}} \theta^\top \phi_{s',a'}$
  - Approximate max with a smooth approximation, e.g., softmax
- Biased gradient estimate due to two time-scale update and Markovian noise

Theorem [25]

Finite-sample bound on convergence of Greedy-GQ with linear function approximation: $\mathbb{E}[\|\nabla J(\theta_W)\|^2] = \mathcal{O}\left(\frac{\log T}{\sqrt{T}}\right)$

- Gradient norm converges to 0 implies convergence to stationary points

Variance Reduced Greedy-GQ [28]

- Greedy-GQ update: denote $O_t = (s_t, a_t, r_t, s_{t+1})$

$$\theta_{t+1} = \theta_t - \alpha G_{O_t}(\theta_t, \omega_t), \quad \omega_{t+1} = \omega_t - \beta H_{O_t}(\theta_t, \omega_t)$$

- Variance reduction [27]: reference parameters $\widetilde{\theta}$, $\widetilde{\omega}$

$$\text{(Reference updates) } \widetilde{G} := \frac{1}{M}\sum_{i=1}^{M} G_{O_i}(\widetilde{\theta}, \widetilde{\omega}), \ \widetilde{H} := \frac{1}{M}\sum_{i=1}^{M} H_{O_i}(\widetilde{\theta}, \widetilde{\omega})$$

(Variance-reduced Greedy-GQ):

$$\theta_{t+1} = \theta_t - \alpha\big(G_{O_t}(\theta_t, \omega_t) - G_{O_t}(\widetilde{\theta}, \widetilde{\omega}) + \widetilde{G}\big)$$

$$\omega_{t+1} = \omega_t - \beta\big(H_{O_t}(\theta_t, \omega_t) - H_{O_t}(\widetilde{\theta}, \widetilde{\omega}) + \widetilde{H}\big)$$

- Periodically update $\widetilde{\theta}, \widetilde{\omega}, \widetilde{G}, \widetilde{H}$
- Improved sample complexity

Outline

Formulation of RL

- State value function:

$$V_\pi(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) | s_0 = s, \pi]$$

- State-action value function:

$$Q_\pi(s, a) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1}) | s_0 = s, a_0 = a, \pi]$$

  where $a_t \sim \pi(\cdot | s_t)$ for all $t \geq 0$.

- Average value function:

$$J(\pi) = (1 - \gamma)\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t, s_{t+1})] = \mathbb{E}_{s \sim \xi}[V_\pi(s)]$$

  where $\xi(\cdot)$ denotes initial distribution.

RL Goal: find the best policy π^*

$$\text{(Criterion I):} \quad V_{\pi^*}(s) \geq V_\pi(s), \quad \forall \pi, \forall s$$

$$\text{(Criterion II):} \quad \max_\pi J(\pi) := \mathbb{E}_{s \sim \xi}[V_\pi(s)]$$

Parameterization of Policy

- Central idea:
  - Parameterize the policy as $\{\pi_w, w \in \mathcal{W}\}$
  - $J(\pi) = J(\pi_w) := J(w)$

Goal of Policy-Based RL: $\max_{w \in \mathcal{W}} J(\pi_w) := J(w)$

Parameterization of Policy

- Central idea:
  - Parameterize the policy as $\{\pi_w, w \in \mathcal{W}\}$
  - $J(\pi) = J(\pi_w) := J(w)$

Goal of Policy-Based RL: $\max_{w \in \mathcal{W}} J(\pi_w) := J(w)$

- Example parameterizations of policy
  - Direct parameterization: $\pi_w(a|s) = w_{s,a}$, where $w \in \triangle(\mathcal{A})^{|\mathcal{S}|}$, i.e., $w_{s,a} \geq 0$, and $\sum_{a \in \mathcal{A}} w_{s,a} = 1$ for all (s, a)
  - Tabular softmax parameterization:

$$\pi_w(a|s) = \frac{\exp(w_{s,a})}{\sum_{a' \in \mathcal{A}} \exp(w_{s,a'})}$$

  - Linear softmax parameterization:

$$\pi_w(a|s) \propto \exp(\phi(s,a)^T w)$$

  - Gaussian policy: $\pi_w(a|s) = \mathcal{N}(\phi(s)^T w, \sigma^2)$

Policy Gradient Algorithm

Goal of Policy-Based RL: $\max_{w \in \mathcal{W}} J(\pi_w) := J(w)$

- Policy gradient $\nabla J(w)$ [29]

$$\nabla_w J(w) = \mathbb{E}_{\nu_{\pi_w}}\left[Q_{\pi_w}(s,a) \nabla_w \log \pi_w(a|s)\right]$$

  - Define score function $\psi_w(s,a) := \nabla_w \log \pi_w(a|s)$
  - Visitation distribution: $\nu_\pi(s,a) = (1-\gamma)\sum_{t=0}^{\infty} \gamma^t \mathbb{P}(s_t = s, a_t = a)$
  - Define advantage function: $A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s)$

Policy Gradient Algorithm

Goal of Policy-Based RL: $\max_{w \in \mathcal{W}} J(\pi_w) := J(w)$

- Policy gradient $\nabla J(w)$ [29]

$$\nabla_w J(w) = \mathbb{E}_{\nu_{\pi_w}}\left[Q_{\pi_w}(s,a)\nabla_w \log \pi_w(a|s)\right]$$

  - Define score function $\psi_w(s,a) := \nabla_w \log \pi_w(a|s)$
  - Visitation distribution: $\nu_\pi(s,a) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \mathbb{P}(s_t = s, a_t = a)$
  - Define advantage function: $A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s)$

$$\nabla_w J(w) = \mathbb{E}_{\nu_{\pi_w}}\left[Q_{\pi_w}(s,a)\psi_w(s,a)\right] = \mathbb{E}_{\nu_{\pi_w}}\left[A_{\pi_w}(s,a)\psi_w(s,a)\right]$$

Policy Gradient Algorithm

Goal of Policy-Based RL: $\max_{w \in \mathcal{W}} J(\pi_w) := J(w)$

- Policy gradient $\nabla J(w)$ [29]

$$\nabla_w J(w) = \mathbb{E}_{\nu_{\pi_w}}\left[Q_{\pi_w}(s,a)\nabla_w \log \pi_w(a|s)\right]$$

  - Define score function $\psi_w(s,a) := \nabla_w \log \pi_w(a|s)$
  - Visitation distribution: $\nu_\pi(s,a) = (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \mathbb{P}(s_t = s, a_t = a)$
  - Define advantage function: $A_\pi(s,a) = Q_\pi(s,a) - V_\pi(s)$

$$\nabla_w J(w) = \mathbb{E}_{\nu_{\pi_w}}\left[Q_{\pi_w}(s,a)\psi_w(s,a)\right] = \mathbb{E}_{\nu_{\pi_w}}\left[A_{\pi_w}(s,a)\psi_w(s,a)\right]$$

Policy gradient algorithm [29, 30]

- update the parameter w via gradient ascent

$$w_{t+1} = w_t + \alpha_t \nabla_w J(w_t)$$

where $\alpha_t > 0$ is the stepsize.

TRPO/PPO Algorithm

Trusted Region Policy Optimization (TRPO) [31]

- Update the parameter w under KL constraint

$$w_{t+1} = \underset{w}{\operatorname{argmax}}[J(w_t) + (w - w_t)^T \nabla_w J(w_t)]$$

$$\text{subject to} \quad \mathbb{E}_{\nu(s)}[KL(\pi_{w_t} || \pi_w)] \le c$$

where $c > 0$ is a hyperparameter.

TRPO/PPO Algorithm

Trusted Region Policy Optimization (TRPO) [31]

- Update the parameter w under KL constraint

$$w_{t+1} = \underset{w}{\text{argmax}}[J(w_t) + (w - w_t)^T \nabla_w J(w_t)]$$
$$\text{subject to} \quad \mathbb{E}_{\nu(s)}[KL(\pi_{w_t} || \pi_w)] \le c$$

where $c > 0$ is a hyperparameter.

Proximal Policy Optimization (PPO) [32]

- Update the parameter w via KL-regularized gradient ascent

$$w_{t+1} = \underset{w}{\text{argmax}}[J(w_t) + (w - w_t)^T \nabla_w J(w_t) - \alpha \mathbb{E}_{\nu_w(s)}[KL(\pi_{w_t} || \pi_w)]]$$

where $\alpha > 0$ is a hyperparameter.

Natural Policy Gradient (NPG) Algorithm

- Second-order Taylor approximation to KL distance

$$KL(\pi_{w_t}||\pi_w) \approx \frac{1}{2}(w - w_t)^T F(w)(w - w_t)$$

  - Fisher information matrix $F(w) = \mathbb{E}_{\nu_{\pi_w}}[\nabla_w \log \pi_{w_t} \nabla_w \log \pi_{w_t}^T]$

Natural Policy Gradient (NPG) Algorithm

- Second-order Taylor approximation to KL distance

$$KL(\pi_{w_t}||\pi_w) \approx \frac{1}{2}(w - w_t)^T F(w)(w - w_t)$$

  - Fisher information matrix $F(w) = \mathbb{E}_{\nu_{\pi_w}}[\nabla_w \log \pi_{w_t} \nabla_w \log \pi_{w_t}^T]$

- KL-regularized update: at time t

$$\begin{aligned}
&\underset{w}{\text{argmax}} \left[J(w_t) + (w - w_t)^T \nabla_w J(w_t) - \alpha \mathbb{E}_{\nu_w(s)}[KL(\pi_{w_t}||\pi_w)]\right] \\
&\approx \underset{w}{\text{argmax}} \left[J(w_t) + (w - w_t)^T \nabla_w J(w_t) - \frac{\alpha}{2}(w - w_t)^T F(w_t)(w - w_t)\right] \\
&= w_t + \alpha F(w_t)^\dagger \nabla_w J(w_t)
\end{aligned}$$

where $F(w_t)^\dagger$ denotes the pseudo-inverse of $F(w_t)$.

Natural Policy Gradient (NPG) Algorithm

- Second-order Taylor approximation to KL distance

$$KL(\pi_{w_t}||\pi_w) \approx \frac{1}{2}(w - w_t)^T F(w)(w - w_t)$$

  - Fisher information matrix $F(w) = \mathbb{E}_{\nu_{\pi_w}}[\nabla_w \log \pi_{w_t} \nabla_w \log \pi_{w_t}^T]$
- KL-regularized update: at time t

$$\begin{aligned} &\underset{w}{\operatorname{argmax}} \left[J(w_t) + (w - w_t)^T \nabla_w J(w_t) - \alpha \mathbb{E}_{\nu_w(s)}[KL(\pi_{w_t}||\pi_w)]\right] \\ &\approx \underset{w}{\operatorname{argmax}} \left[J(w_t) + (w - w_t)^T \nabla_w J(w_t) - \frac{\alpha}{2}(w - w_t)^T F(w_t)(w - w_t)\right] \\ &= w_t + \alpha F(w_t)^\dagger \nabla_w J(w_t) \end{aligned}$$

where $F(w_t)^\dagger$ denotes the pseudo-inverse of $F(w_t)$.

Natural Policy Gradient (NPG) [33]

- Update parameter w via KL approximator based regularizer

$$w_{t+1} = w_t + \alpha F(w_t)^\dagger \nabla_w J(w_t)$$

Convergence with Exact Policy Gradient

- Policy gradient
  - Direct and tabular softmax policy: global sublinear convergence [34]
  - Direct policy: global linear convergence via regularized MDP [35]
  - Direct policy: global linear convergence via line search [36]
- TRPO/PPO
  - Direct policy: global sublinear convergence via adaptivity [37]
  - Direct policy: global linear convergence via regularized MDP [35]
  - Direct policy: global convergence via line search [36]
- NPG
  - Tabular softmax policy: global sublinear convergence [34]
  - Tabular softmax policy: global linear convergence via regularized MDP [38]

Policy Gradient Algorithms under Unknown MDP

$$\nabla J(w) = \mathbb{E}_{\nu_{\pi_w}}\left[Q_{\pi_w}(s,a)\psi_w(s,a)\right] = \mathbb{E}_{\nu_{\pi_w}}\left[A_{\pi_w}(s,a)\psi_w(s,a)\right]$$

- Let $\hat{P}(\cdot|s_t, a_t) = \gamma\mathbb{P}(\cdot|s_t, a_t) + (1-\gamma)\xi(\cdot)$ [39]
  - $\xi(\cdot)$: initial distribution
  - Samples drawn from $\hat{P}(\cdot|s_t, a_t)$ converge to visitation distribution ν_{π_w}

Policy Gradient Algorithms under Unknown MDP

$$\nabla J(w) = \mathbb{E}_{\nu_{\pi_w}}\left[Q_{\pi_w}(s,a)\psi_w(s,a)\right] = \mathbb{E}_{\nu_{\pi_w}}\left[A_{\pi_w}(s,a)\psi_w(s,a)\right]$$

- Let $\hat{P}(\cdot|s_t,a_t) = \gamma\mathbb{P}(\cdot|s_t,a_t) + (1-\gamma)\xi(\cdot)$ [39]
  - $\xi(\cdot)$: initial distribution
  - Samples drawn from $\hat{P}(\cdot|s_t,a_t)$ converge to visitation distribution ν_{π_w}

Model-free Policy Gradient

- Sample $s_t \sim \hat{P}(\cdot|s_{t-1},a_{t-1}), a_t \sim \pi_{w_t}(\cdot|s_t)$
- Unbiased estimation of $A_{\pi_{w_t}}(s_t,a_t)$
  - Sample a length-K trajectory starting at (s_t,a_t), $K \sim \mathrm{Geom}(1-\gamma)$
  - Estimate $\hat{Q}(s_t,a_t)$ by adding rewards over the sample path
  - Sample a length-K trajectory starting at (s_t), $K \sim \mathrm{Geom}(1-\gamma)$
  - Estimate $\hat{V}(s_t)$ by adding rewards over the sample path
  - $\hat{A}_{\pi_{w_t}}(s_t,a_t) = \hat{Q}(s_t,a_t) - \hat{V}(s_t)$
- Estimate policy gradient $g_t = \hat{A}_{\pi_{w_t}}(s_t,a_t)\nabla_{w_t}\log(\pi_{w_t}(a_t|s_t))$
- Update $w_{t+1} = w_t + \alpha_t g_t$

Convergence of Model-free PG Algorithms

Theorem ([40])

Consider a general nonlinear *policy* $\{\pi_w : w \in \mathcal{W}\}$. *Under a constant stepsize* $\alpha_t = \alpha$, *the output of model-free PG satisfies*

$$\min_{t\in[T]} \mathbb{E}\left[\|\nabla_{w_t} J(w_t)\|^2\right] \leq \mathcal{O}\left(\frac{1}{\alpha T}\right) + \mathcal{O}(\alpha \log^2 \frac{1}{\alpha}).$$

- PG converges to a neighborhood of a stationary point at a rate of $\mathcal{O}\left(\frac{1}{T}\right)$.
  - α controls a tradeoff between convergence rate and accuracy
  - Decreasing α improves accuracy, but slows down convergence
  - Let $\alpha_t = \frac{1}{\sqrt{T}}$, PG converges with a rate of $\mathcal{O}\left(\frac{\log^2 T}{\sqrt{T}}\right)$

Actor-Critic Algorithms [41]

Actor-Critic Algorithm

- Critic
  - Estimates $V_\theta(s)$ by linear function approximation $\phi(s)^\top\theta$
  - Takes T_c length-M minibatch TD learning updates and outputs θ_t
- Actor
  - Approximates $A_{\pi_w}(s,a)$ by temporal difference error $\delta_\theta(s,a,s')$

$$\hat{A}_{\pi_w}(s,a) = \delta_\theta(s,a,s') = r(s,a,s') + \gamma\phi(s')^\top\theta - \phi(s)^\top\theta$$

  - Estimate policy gradient $v_t(\theta_t)$ by averaging $\delta_{\theta_t}(s_t,a_t,s_{t+1})\psi_{w_t}(s_t,a_t)$ over a length-B sample trajectory
  - Updates $w_{t+1} = w_t + \alpha_t v_t(\theta_t)$

Convergence Rate of Actor-Critic Algorithm

Theorem ([42])

Consider a general nonlinear *policy* $\{\pi_w : w \in \mathcal{W}\}$*, and* $\hat{T}$ *is chosen uniformly from* $\{1, \cdots, T\}$*.*

$$\mathbb{E}[\left\|\nabla_w J(w_{\hat{T}})\right\|_2^2] \leq \mathcal{O}\left(\frac{1}{T}\right) + \mathcal{O}\left(\frac{1}{B}\right) + (1 - \mathcal{O}(\lambda_{A_\pi}\beta))^{T_c} + \mathcal{O}\left(\frac{\beta}{M}\right) + \mathcal{O}(\zeta_{\text{approx}}^{\text{critic}}).$$

- Actor has sublinear convergence, and critic has linear convergence
- Actor's bias and variance $\mathcal{O}\left(\frac{1}{B}\right)$; Critic's bias and variance $\mathcal{O}\left(\frac{\beta}{M}\right)$
- Critic's approximation error: $\zeta_{\text{approx}}^{\text{critic}} = \max_{w \in \mathcal{W}} \mathbb{E}_{\nu_w}[|V_{\pi_w}(s) - V_{\theta^*_{\pi_w}}(s)|^2]$
- Actor's mini-batch yields faster convergence rate of $\mathcal{O}(1/T)$ rather than $\mathcal{O}(1/\sqrt{T})$
- This further yields better overall sample complexity

Proof of Convergence

- Let $v_t(\theta)$ denote estimator of $g(\theta, w) = \mathbb{E}_{\nu_w}[A_\theta(s,a)\psi_w(s,a)]$
- Decompose error terms

$$
\begin{aligned}
&\Big(\frac{1}{2}\alpha - L_J\alpha^2\Big)\mathbb{E}[\|\nabla_w J(w_t)\|_2^2 \,|\mathcal{F}_t] \\
&\leq \mathbb{E}[J(w_{t+1})|\mathcal{F}_t] - J(w_t) + 3\Big(\frac{1}{2}\alpha + L_J\alpha^2\Big)\mathbb{E}\Big[\,\big\|v_t(\theta_t) - v_t(\theta^*_{w_t})\big\|_2^2 \\
&\quad + \big\|v_t(\theta^*_{w_t}) - g(\theta^*_{w_t}, w_t)\big\|_2^2 + \big\|g(\theta^*_{w_t}, w_t) - \nabla_w J(w_t)\big\|_2^2 \,\Big|\mathcal{F}_t\Big].
\end{aligned}
$$

- Error due to TD learning

$$
\begin{aligned}
&\mathbb{E}[\big\|v_t(\theta_t) - v_t(\theta^*_{w_t})\big\|_2^2 \,|\mathcal{F}_t] \\
&\leq 4\mathbb{E}[\big\|\theta_t - \theta^*_{w_t}\big\|_2^2 \,|\mathcal{F}_t] \leq (1 - \mathcal{O}(\lambda_{A_\pi}\beta))^{T_c} + \mathcal{O}(\beta/M)
\end{aligned}
$$

Proof of Convergence (Cont.)

- Gradient estimation error under Markovian minibatch sampling

$$\mathbb{E}\left[\left\|v_t(\theta^*_{w_t}) - g(\theta^*_{w_t}, w_t)\right\|_2^2 |\mathcal{F}_t\right] \leq \mathcal{O}\left(\frac{1}{B}\right).$$

- Critic's approximation error

$$\left\|g(\theta^*_{w_t}, w_t) - \nabla_w J(w_t)\right\|_2^2 \leq \mathcal{O}\left(\zeta^{\text{critic}}_{\text{approx}}\right)$$

- Combine error bounds and take summarization over iteration path

$$\begin{aligned}\mathbb{E}[\left\|\nabla_w J(w_{\hat{T}})\right\|_2^2] \leq & \mathcal{O}\left(\tfrac{1}{T}\right) + (1 - \mathcal{O}(\lambda_{A_\pi}\beta))^{T_c} + \mathcal{O}\left(\tfrac{\beta}{M}\right) \\ & + \mathcal{O}\left(\tfrac{1}{B}\right) + \mathcal{O}(\zeta^{\text{critic}}_{\text{approx}}).\end{aligned}$$

Natural Policy Gradient under Unknown MDP

- Natural policy gradient (NPG) [33, 43],

$$w_{t+1} = w_t + \alpha_t F(w_t)^{\dagger} \nabla J(w_t)$$

- Consider $\min_{\theta \in \mathbb{R}^d} L_w(\theta) = \mathbb{E}_{\nu_{\pi_w}}[A_{\pi_w}(s,a) - \psi(s,a)^\top \theta]^2$
  - Minimum norm solution satisfies $\theta_w = F(w)^{\dagger} \nabla J(w)$
- NPG update [34]: $w_{t+1} = w_t + \alpha_t \theta_t$

Natural Policy Gradient under Unknown MDP

- Natural policy gradient (NPG) [33, 43],

$$w_{t+1} = w_t + \alpha_t F(w_t)^\dagger \nabla J(w_t)$$

- Consider $\min_{\theta \in \mathbb{R}^d} L_w(\theta) = \mathbb{E}_{\nu_{\pi_w}}[A_{\pi_w}(s,a) - \psi(s,a)^\top \theta]^2$
  - Minimum norm solution satisfies $\theta_w = F(w)^\dagger \nabla J(w)$
- NPG update [34]: $w_{t+1} = w_t + \alpha_t \theta_t$

Model-free NPG [34]

- At step t, solve least square problem via K iterations
  - Obtain unbiased estimator $\hat{A}_{\pi_{w_t}}(s_k, a_k)$ (same as PG)
  - Update $\theta_{k+1} = \theta_k - \beta \nabla_\theta L_{w_t}(\theta_k)$
- Update $w_{t+1} = w_t + \alpha_t \theta_K$

Natural Policy Gradient under Unknown MDP

- Natural policy gradient (NPG) [33, 43],

$$w_{t+1} = w_t + \alpha_t F(w_t)^\dagger \nabla J(w_t)$$

- Consider $\min_{\theta \in \mathbb{R}^d} L_w(\theta) = \mathbb{E}_{\nu_{\pi_w}}[A_{\pi_w}(s,a) - \psi(s,a)^\top \theta]^2$
  - Minimum norm solution satisfies $\theta_w = F(w)^\dagger \nabla J(w)$
- NPG update [34]: $w_{t+1} = w_t + \alpha_t \theta_t$

Model-free NPG [34]

- At step t, solve least square problem via K iterations
  - Obtain unbiased estimator $\hat{A}_{\pi_{w_t}}(s_k, a_k)$ (same as PG)
  - Update $\theta_{k+1} = \theta_k - \beta \nabla_\theta L_{w_t}(\theta_k)$
- Update $w_{t+1} = w_t + \alpha_t \theta_K$

- NPG with general nonlinear policy converges globally as $\mathcal{O}\left(\frac{1}{\sqrt{T}}\right)$ [34]
- Can achieve $\mathcal{O}\left(\frac{1}{T}\right)$ by self-variance reduction of gradient norm [42]

Natural Actor-Critic Algorithm

$$J(w) = \mathbb{E}_{\nu_{\pi_w}}\left[Q_{\pi_w}(s,a)\psi_w(s,a)\right] = \mathbb{E}_{\nu_{\pi_w}}\left[A_{\pi_w}(s,a)\psi_w(s,a)\right]$$
$$w_{t+1} = w_t + \alpha_t F(w_t)^{\dagger}\nabla J(w_t)$$

Natural Actor-Critic Algorithm

- Critic (same as critic in actor-critic algorithm)
  - Estimates $V_\theta(s)$ by linear function approximation $\phi(s)^\top\theta$
  - Takes T_c length-M minibatch TD learning updates and outputs θ_t
- Actor
  - Computes policy gradient estimator $v_t(\theta_t)$ as in actor-critic algorithm
  - Computes Fisher information estimator $F_t(w_t)$ by averaging over a length-B sample trajectory
  - Updates $w_{t+1} = w_t + \alpha_t F_t(w_t)^{\dagger} v_t(\theta_t)$

Convergence Rate of Natural Actor-Critic Algorithm

Theorem ([42])

Consider a general nonlinear *policy* $\{\pi_w : w \in \mathcal{W}\}$, *and* $\hat{T}$ *is chosen uniformly from* $\{1, \cdots, T\}$.

$$J(\pi^*) - \mathbb{E}\big[J(\pi_{w_{\hat{T}}})\big] \leq \mathcal{O}\left(\tfrac{1}{T}\right) + \mathcal{O}\left(\tfrac{1}{\sqrt{B}}\right) + (1 - \mathcal{O}(\lambda_{A_\pi}\beta))^{T_c/2} + \mathcal{O}\left(\tfrac{1}{\sqrt{M}}\right)$$

$$+ \mathcal{O}(\sqrt{\zeta^{\text{critic}}_{\text{approx}}}) + \mathcal{O}\left(\tfrac{1}{B}\right) + (1 - \mathcal{O}(\lambda_{A_\pi}\beta))^{T_c} + \mathcal{O}\left(\tfrac{\beta}{M}\right) + \mathcal{O}(\zeta^{\text{critic}}_{\text{approx}}) + \mathcal{O}(\sqrt{\zeta^{\text{actor}}_{\text{approx}}})$$

- Actor has sublinear convergence, and critic has linear convergence
- Critic's approx. error: $\zeta^{\text{critic}}_{\text{approx}} = \max_{w \in \mathcal{W}} \mathbb{E}_{\nu_w}[|V_{\pi_w}(s) - V_{\theta^*_{\pi_w}}(s)|^2]$
- Actor's approx. error: $\zeta^{\text{actor}}_{\text{approx}} = \max_{w \in \mathcal{W}} \min_{p \in \mathbb{R}^{d_2}} \mathbb{E}_{\nu_{\pi_w}}\left[\psi_w(s,a)^\top p - A_{\pi_w}(s,a)\right]^2$
- Diminishing variance in actor's update yields a faster convergence rate of $\mathcal{O}(1/T)$ than $\mathcal{O}(1/\sqrt{T})$
- Performance difference lemma [34] of NAC yields global convergence

Extension I: Policy Gradient Algorithm with Adam

PG-AMSGrad [40]

- Sample $s_t \sim \hat{P}(\cdot|s_{t-1}, a_{t-1}), a_t \sim \pi_{w_t}(\cdot|s_t)$
- Estimate Q-function $\hat{Q}_{\pi_{w_t}}(s_t, a_t)$ as in PG
- Estimate policy gradient $g_t = \hat{Q}_{\pi_{w_t}}(s_t, a_t)\nabla_{w_t} \log(\pi_{w_t}(a_t|s_t))$
- $m_t = (1 - \beta_1)m_{t-1} + \beta_1 g_t$ momentum
- $v_t = (1 - \beta_2)\hat{v}_{t-1} + \beta_2 g_t^2$ stepsize adaptation
- $\hat{v}_t = \max(\hat{v}_{t-1}, v_t)$, $\hat{V}_t = diag(\hat{v}_{t,1}, \ldots, \hat{v}_{t,d})$
- Update policy parameter $w_{t+1} = w_t - \alpha_t \hat{V}_t^{-\frac{1}{2}} m_t$

- Convergence rate of PG-AMSGrad [40]
- In practice, PG with Adam converges much faster

Extension II: Off-Policy Policy Gradient Algorithms

- Off-policy policy gradient
  - On-policy sampling with target policy is not possible
  - Off-policy sampling under behavior policy: $(s_i, a_i, s_i') \sim \mathcal{D}$
  - Estimate $\nabla_w J(w)$ with off-policy samples

Actor-critic with distribution correction (AC-DC)

$$g(w) = \hat{\rho}(s,a)\hat{Q}_{\pi_w}(s,a)\nabla_w \log(\pi_w(s,a))$$

where $\hat{\rho}$ and $\hat{Q}_{\pi_w}$ are approximation of $\rho = \nu_{\pi_w}/\mathcal{D}$ and Q_{π_w}, respectively.

- Bias error of AC-DC suffers substantially from estimation errors

$$\Delta_g = \mathbb{E}_{\mathcal{D}}[g(w)] - \nabla_w J(\pi_w) = \Theta(\mathbb{E}[\varepsilon_\rho(s,a) + \varepsilon_Q(s,a)])$$

where $\varepsilon_\rho = \rho - \hat{\rho}$ and $\varepsilon_Q = Q - \hat{Q}$

- Doubly robust off-policy PG estimation [44] reduces bias error

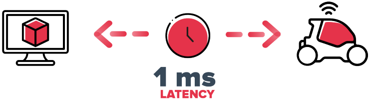
Outline

1. Introduction to Reinforcement Learning and Applications
2. Value-based Algorithms
   - Policy Evaluation
   - Optimal Control
3. Policy Gradient Algorithms
4. **Advanced Topics on RL and Open Directions**
   - Constrained Reinforcement Learning
   - Imitation Learning
   - Multi-Agent Reinforcement Learning
   - Robust Reinforcement Learning

Outline

Topic 1: Constrained Reinforcement Learning

- Practical RL applications involve various safety/resource constraints
  - Left: Power constraint on battery powered devices
  - Right: Safety constraints on autonomous robotics and vehicles
  - Bottom: Delay constraint in communication system



Constrained Markov Decision Process (CMDP)

- Same dynamics as general MDP
- Agent receives reward R and cost C
- Value function w.r.t. reward R:

$$J_R(\pi) := (1-\gamma)\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})\right]$$

- Value function w.r.t. cost C:

$$J_C(\pi) := (1-\gamma)\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t C(s_t, a_t, s_{t+1})\right]$$

Goal of CMDP

$$\max_{\pi} \quad J_R(\pi) \quad \text{subject to} \quad J_C(\pi) \leq c \tag{P}$$

Primal-Dual Approach: e.g. CPO [47], PDO [48]

- Let $\lambda > 0$ be Lagrangian multiplier. Define Lagrangian:

$$\mathcal{L}(\pi, \lambda) = -J_R(\pi) + \lambda(J_C(\pi) - c).$$

- Dual function: $d(\lambda) := \min_\pi \mathcal{L}(\pi, \lambda)$
- Dual problem:

$$D^* = \max_{\lambda \in \mathbb{R}_+} d(\lambda) := \max_{\lambda \in \mathbb{R}_+} \min_\pi \mathcal{L}(\pi, \lambda) \tag{D}$$

- Duality gap: $\Delta = D^* - P^*$, where P^* is the negative solution of (P)
  - Zero duality gap [45, 46]
  - (P) can be equivalently solved by solving (D)

Primal-Dual Approach

Primal-Dual Algorithm

- For $t = 0, 1, \ldots, T$
  - Compute π_{t+1} based on $\mathcal{L}(\pi, \lambda_t)$ and π_t. Example methods:
    - Dual descent [46]: $\pi_{t+1} = \arg\min_\pi \mathcal{L}(\pi, \lambda_t)$ using some RL oracle
    - Natural policy gradient [49]: $\pi_{t+1} = \pi_t - \eta F_\rho(\pi_t)^\dagger \cdot \nabla_\pi \mathcal{L}(\pi_t, \lambda_t)$
  - Compute the dual ascent step $\lambda_{k+1} = (\lambda_k + \eta(J_C(\pi_{t+1}) - c))_+$.

Primal-Dual Approach

Primal-Dual Algorithm

- For $t = 0, 1, ..., T$
  - Compute π_{t+1} based on $\mathcal{L}(\pi, \lambda_t)$ and π_t. Example methods:
    - Dual descent [46]: $\pi_{t+1} = \arg\min_\pi \mathcal{L}(\pi, \lambda_t)$ using some RL oracle
    - Natural policy gradient [49]: $\pi_{t+1} = \pi_t - \eta F_\rho(\pi_t)^\dagger \cdot \nabla_\pi \mathcal{L}(\pi_t, \lambda_t)$
  - Compute the dual ascent step $\lambda_{k+1} = (\lambda_k + \eta(J_C(\pi_{t+1}) - c))_+$.

- Performance metric:
  - Let π^* denote the optimal solution to primal problem P
  - Optimality gap: $J_R(\pi^*) - J_R(\pi)$.
  - Constraint violation: $(J_C(\pi) - c)_+$.

Primal-Dual Approach

Primal-Dual Algorithm

- For $t = 0, 1, ..., T$
  - Compute π_{t+1} based on $\mathcal{L}(\pi, \lambda_t)$ and π_t. Example methods:
    - Dual descent [46]: $\pi_{t+1} = \arg\min_\pi \mathcal{L}(\pi, \lambda_t)$ using some RL oracle
    - Natural policy gradient [49]: $\pi_{t+1} = \pi_t - \eta F_\rho(\pi_t)^\dagger \cdot \nabla_\pi \mathcal{L}(\pi_t, \lambda_t)$
  - Compute the dual ascent step $\lambda_{k+1} = (\lambda_k + \eta(J_C(\pi_{t+1}) - c))_+$.

- Performance metric:
  - Let π^* denote the optimal solution to primal problem P
  - Optimality gap: $J_R(\pi^*) - J_R(\pi)$.
  - Constraint violation: $(J_C(\pi) - c)_+$.
- Convergence Rate:
  - Duality gap decays at a rate of $\mathcal{O}(1/\sqrt{T})$ [46]
  - Optimality gap decays $\mathcal{O}(1/\sqrt{T})$ and constraint violation decays $\mathcal{O}(1/T^{\frac{1}{4}})$ [49]

Accelerated and Regularized Constrained Policy Optimizer (AR-CPO) [50]

- Central idea: solve the minimax problem over the (τ, μ)-regularized Lagrangian via an accelerated dual descent

$$\max_{\lambda \in \mathbb{R}_+^m} \min_{\pi \in \Pi} \mathcal{L}_{\tau,\mu}(\pi, \lambda) := \mathcal{L}(\pi, \lambda) - \tau \mathcal{H}(\pi) - \frac{\mu}{2} \|\lambda\|_2^2$$

Accelerated and Regularized Constrained Policy Optimizer (AR-CPO) [50]

- Central idea: solve the minimax problem over the (τ, μ)-regularized Lagrangian via an accelerated dual descent

$$\max_{\lambda \in \mathbb{R}_+^m} \min_{\pi \in \Pi} \mathcal{L}_{\tau,\mu}(\pi, \lambda) := \mathcal{L}(\pi, \lambda) - \tau \mathcal{H}(\pi) - \frac{\mu}{2} \|\lambda\|_2^2$$

  - Entropy-regularized policy optimizer
    - $\tau \mathcal{H}(\pi)$: smooth dual function $d_{\tau,\mu}(\lambda) := \min_{\pi \in \Pi} \mathcal{L}_{\tau,\mu}(\pi, \lambda)$
    - Examples: RegPO-NPG [33, 38], RegPO-SoftQ [50]

Accelerated and Regularized Constrained Policy Optimizer (AR-CPO) [50]

- Central idea: solve the minimax problem over the (τ, μ)-regularized Lagrangian via an accelerated dual descent

$$\max_{\lambda \in \mathbb{R}_+^m} \min_{\pi \in \Pi} \mathcal{L}_{\tau,\mu}(\pi, \lambda) := \mathcal{L}(\pi, \lambda) - \tau \mathcal{H}(\pi) - \frac{\mu}{2} \|\lambda\|_2^2$$

  - Entropy-regularized policy optimizer
    - ★ $\tau \mathcal{H}(\pi)$: smooth dual function $d_{\tau,\mu}(\lambda) := \min_{\pi \in \Pi} \mathcal{L}_{\tau,\mu}(\pi, \lambda)$
    - ★ Examples: RegPO-NPG [33, 38], RegPO-SoftQ [50]
  - ℓ_2 regularization on λ
    - ★ $\frac{\mu}{2}\|\lambda\|_2^2$: dual function $d_{\tau,\mu}(\lambda)$ becomes strongly concave

Accelerated and Regularized Constrained Policy Optimizer (AR-CPO) [50]

- Central idea: solve the minimax problem over the (τ, μ)-regularized Lagrangian via an accelerated dual descent

$$\max_{\lambda \in \mathbb{R}_+^m} \min_{\pi \in \Pi} \mathcal{L}_{\tau,\mu}(\pi, \lambda) := \mathcal{L}(\pi, \lambda) - \tau \mathcal{H}(\pi) - \frac{\mu}{2} \|\lambda\|_2^2$$

  - Entropy-regularized policy optimizer
    - $\tau \mathcal{H}(\pi)$: smooth dual function $d_{\tau,\mu}(\lambda) := \min_{\pi \in \Pi} \mathcal{L}_{\tau,\mu}(\pi, \lambda)$
    - Examples: RegPO-NPG [33, 38], RegPO-SoftQ [50]
  - ℓ_2 regularization on λ
    - $\frac{\mu}{2}\|\lambda\|_2^2$: dual function $d_{\tau,\mu}(\lambda)$ becomes strongly concave
  - Nesterov's accelerated gradient descent dual optimizer for updating λ
    - Improve the dependence on the condition number

Accelerated and Regularized Constrained Policy Optimizer (AR-CPO) [50]

- Central idea: solve the minimax problem over the (τ, μ)-regularized Lagrangian via an accelerated dual descent

$$\max_{\lambda \in \mathbb{R}_+^m} \min_{\pi \in \Pi} \mathcal{L}_{\tau,\mu}(\pi, \lambda) := \mathcal{L}(\pi, \lambda) - \tau \mathcal{H}(\pi) - \frac{\mu}{2} \|\lambda\|_2^2$$

  - Entropy-regularized policy optimizer
    - $\tau \mathcal{H}(\pi)$: smooth dual function $d_{\tau,\mu}(\lambda) := \min_{\pi \in \Pi} \mathcal{L}_{\tau,\mu}(\pi, \lambda)$
    - Examples: RegPO-NPG [33, 38], RegPO-SoftQ [50]
  - ℓ_2 regularization on λ
    - $\frac{\mu}{2}\|\lambda\|_2^2$: dual function $d_{\tau,\mu}(\lambda)$ becomes strongly concave
  - Nesterov's accelerated gradient descent dual optimizer for updating λ
    - Improve the dependence on the condition number
- Optimality gap and constraint violation decay $\mathcal{O}(1/T)$ [50]

Drawback of Primal-Dual Approach

- Solve a minimax problem over an augmented Lagrangian function

$$\max_{\lambda \in \mathbb{R}_+^m} \min_{\pi \in \Pi} \; -J_R(\pi) + \lambda(J_C(\pi) - c)$$

Drawback of Primal-Dual Approach

- Solve a minimax problem over an augmented Lagrangian function

$$\max_{\lambda \in \mathbb{R}_+^m} \min_{\pi \in \Pi} \; -J_R(\pi) + \lambda(J_C(\pi) - c)$$

- Alternating between policy π and dual variable λ updates
  - If the constraint is violated, $J_C(\pi) - c \geq 0$, λ becomes larger (positive), and policy update will reduce constraint function
  - If the constraint is satisfied, $J_C(\pi) - c \leq 0$, λ decreases gradually to zero so that constraint eventually does not play a role in policy update
  - However, λ can only iteratively increase or decrease, which yields delayed response to enforcing or releasing constraints

Drawback of Primal-Dual Approach

- Solve a minimax problem over an augmented Lagrangian function

$$\max_{\lambda \in \mathbb{R}_+^m} \min_{\pi \in \Pi} \; -J_R(\pi) + \lambda(J_C(\pi) - c)$$

- Alternating between policy π and dual variable λ updates
  - If the constraint is violated, $J_C(\pi) - c \geq 0$, λ becomes larger (positive), and policy update will reduce constraint function
  - If the constraint is satisfied, $J_C(\pi) - c \leq 0$, λ decreases gradually to zero so that constraint eventually does not play a role in policy update
  - However, λ can only iteratively increase or decrease, which yields delayed response to enforcing or releasing constraints

What is more desirable?

- Respond faster if a constraint is satisfied or violated
- Do not introduce a dual variable for easier implementation

A Primal Approach: CRPO [51]

Constraint-Rectified Policy Optimization (CRPO)

- For $t = 0, 1, ..., T-1$
  - Constraint violation: **If** $J_C(\pi_t) \geq c + \delta$: $\pi_{t+1} \leftarrow$ take one step natural policy gradient update towards minimize $J_C(\pi_t)$
  - Objective improvement: **Else** $\pi_{t+1} \leftarrow$ take one step natural policy gradient update towards maximize $J_R(\pi_t)$

A Primal Approach: CRPO [51]

Constraint-Rectified Policy Optimization (CRPO)

- For $t = 0, 1, ..., T-1$
  - Constraint violation: **If** $J_C(\pi_t) \geq c + \delta$: $\pi_{t+1} \leftarrow$ take one step natural policy gradient update towards minimize $J_C(\pi_t)$
  - Objective improvement: **Else** $\pi_{t+1} \leftarrow$ take one step natural policy gradient update towards maximize $J_R(\pi_t)$

- CRPO responds to constraint satisfaction/violation immediately
  - Primal-dual relies on iteration of dual variables, incurring large delay
- CRPO can be implemented as easy as unconstrained optimization
  - Primal-dual requires to update dual variables, which is more complex
- CRPO does not suffer from hyperparameter tuning of learning rates and projection threshold of dual variables
  - Primal-dual approach can be very sensitive to these hyperparamters

Neural Network Function Approximation

- Use neural network to parameterize both value functions and policy
- Define a feature vector $\psi(s, a) \in \mathbb{R}^d$ with $d \geq 2$ for each (s, a)
  - $\|\psi(s, a)\|_2 \leq 1$ for all $(s, a) \in \mathcal{S} \times \mathcal{A}$
- A two-layer neural network $f((s, a); W, b)$ with input $\psi(s, a)$ and width m

$$f((s, a); W, b) = \frac{1}{\sqrt{m}} \sum_{r=1}^{m} b_r \cdot \mathrm{ReLU}(W_r^\top \psi(s, a))$$

  - $b = [b_1, \cdots, b_m]^\top \in \mathbb{R}^m$, and $W = [W_1^\top, \cdots, W_m^\top]^\top \in \mathbb{R}^{md}$
- Initialize $[W_0]_r \sim \mathrm{Unif}\{D_w\}$, where $D_w = \{W : d_1 \leq \|[W]_r\|_2 \leq d_2\}$ and $b_r \sim \mathrm{Unif}[-1, 1]$ independently

Policy Evaluation

- TD learning: at time t, sample $s_{t+1} \sim P(\cdot|s,a)$

$$\hat{\mathcal{T}}_t Q_t(s_t, a_t) = r(s_t, a_t, s_{t+1}) + \gamma Q_t(s_{t+1}, a')$$
$$Q_{t+1} = Q_t + \alpha_t(\hat{\mathcal{T}}_t Q_t - Q_t)$$

- Neural TD learning: neural network parametrization $\theta^R \in \mathbb{R}^{md}$

$$\tilde{\theta}^R = \theta_k^R + \beta\big[R(s,a,s') + \gamma f((s',a');\theta_k^R) - f((s,a);\theta_k^R)\big]\nabla_\theta f((s,a);\theta_k^R)$$
$$\theta_{k+1}^R = \mathrm{argmin}_{\theta\in\boldsymbol{B}} \left\|\theta - \tilde{\theta}^R\right\|_2, \quad \text{where } \boldsymbol{B} = \{\theta \in \mathbb{R}^{md} : \|\theta - \theta_0^i\|_2 \le \Gamma^R\}$$

- Let $\bar{\theta}_K^R = \frac{1}{K}\sum_{k=0}^{K-1}\theta_k^R$ be the average output
- $\bar{Q}_t^R(s,a) = f((s,a),\theta_K^R)$ is an estimator of value function $Q^R_{\pi_{\tau_t W_t}}(s,a)$
- Similarly, $\bar{Q}_t^C(s,a) = f((s,a),\theta_K^C)$ is an estimator of constraint value function $Q^C_{\pi_{\tau_t W_t}}(s,a)$

High-Probability Guarantee for Neural TD

- Consider TD iteration with neural network approximation
  - Let stepsize $\beta = \min\{1/\sqrt{K}, (1-\gamma)/12\}$

High-Probability Guarantee for Neural TD

- Consider TD iteration with neural network approximation
  - Let stepsize $\beta = \min\{1/\sqrt{K}, (1-\gamma)/12\}$

Theorem 1 (High-probability convergence of neural TD)

Under mild regularity conditions and bounded variance, with probability at least $1-\delta$, neural TD learning satisfies

$$\left\| \bar{Q}_t^i(s,a) - Q^i_{\pi_{\tau_t W_t}}(s,a) \right\|^2_{\mu_\pi} \leq \Theta\left(\frac{1}{\sqrt{K}} \sqrt{\log\left(\frac{1}{\delta}\right)} \right) + \Theta\left(\frac{1}{m^{1/4}} \sqrt{\log\left(\frac{K}{\delta}\right)} \right).$$

where $i = R, C$.

High-Probability Guarantee for Neural TD

- Consider TD iteration with neural network approximation
  - Let stepsize $\beta = \min\{1/\sqrt{K}, (1-\gamma)/12\}$

Theorem 1 (High-probability convergence of neural TD)

Under mild regularity conditions and bounded variance, with probability at least $1-\delta$, neural TD learning satisfies

$$\left\| \bar{Q}_t^i(s,a) - Q^i_{\pi_{\tau_t W_t}}(s,a) \right\|^2_{\mu_\pi} \leq \Theta\left(\frac{1}{\sqrt{K}} \sqrt{\log\left(\frac{1}{\delta}\right)} \right) + \Theta\left(\frac{1}{m^{1/4}} \sqrt{\log\left(\frac{K}{\delta}\right)} \right).$$

where $i = R, C$.

- For $K = \Theta(\sqrt{m})$ iterations, $||\bar{Q}_t^i - Q^i_{\pi_{\tau_t W_t}}||_{\mu_\pi} = \mathcal{O}(1/m^{1/8})$

Constraint Estimation

- Sample a batch of state-action pairs $(s_j, a_j) \in \mathcal{B}_t$ from distribution $\xi(\cdot)\pi_{W_t}(\cdot|\cdot)$
- Estimation error of constraint $\left|\bar{J}_C(\theta_t^C) - J_C(\pi_{w_t})\right|$ is small if policy evaluation $\bar{Q}_t^C$ is accurate and concentration of sampling occurs

Constraint Estimation

- Sample a batch of state-action pairs $(s_j, a_j) \in \mathcal{B}_t$ from distribution $\xi(\cdot)\pi_{W_t}(\cdot|\cdot)$
- Estimation error of constraint $\left|\bar{J}_C(\theta_t^C) - J_C(\pi_{w_t})\right|$ is small if policy evaluation $\bar{Q}_t^C$ is accurate and concentration of sampling occurs

Assumption 1 (Concentration of sampling process)

For any parameterized policy π_W, there exists a constant $C_f > 0$ such that for all $k \geq 0$, $\mathbb{E}_{\xi\cdot\pi_W}\left[\exp([\bar{Q}_t^i(s,a) - \mathbb{E}_{\xi\cdot\mu_{\pi_{\tau_t W_t}}}\bar{Q}_t^i(s,a)]^2/C_f^2)\right] \leq 1.$

NPG in CRPO

- Natural policy gradient

$$\bar{\Delta}_t^i = \operatorname*{argmin}_{\theta \in \boldsymbol{B}} \mathbb{E}_{\nu_{\pi_{\tau_t w_t}}} [(\bar{Q}_t^i(s,a) - \psi_{w_t}(s,a)^\top \theta)^2]$$

  - τ_t controls amplitude of w_t; $\tau_t w_t$ serves as parameter of policy
  - $\mathcal{N}_0$: collects all feasible w_t over the algorithm path
  - η: constraint violation level

Algorithm 1 Policy Update for CRPO

1: $\tau_{t+1} = \tau_t + \alpha$
2: **if** $\bar{J}_{C,\mathcal{B}_t} \leq c + \eta$ **then**
3: Add w_t into set $\mathcal{N}_0$
4: $\tau_{t+1} \cdot w_{t+1} = \tau_t \cdot w_t + \alpha \bar{\Delta}_t^R$
5: **else**
6: $\tau_{t+1} \cdot w_{t+1} = \tau_t \cdot w_t - \alpha \bar{\Delta}_t^C$
7: **end if**

Convergence Guarantee of CRPO

- Consider CRPO with neural network approximation
  - Neural TD learning with $K_{\text{in}} = \Theta(\sqrt{m})$ at each iteration
  - Tolerance $\eta = \Theta(m/\sqrt{T} + m^{-1/8})$
  - NPG update learning rate $\alpha = \Theta(1/\sqrt{T})$

Convergence Guarantee of CRPO

- Consider CRPO with neural network approximation
  - Neural TD learning with $K_{\text{in}} = \Theta(\sqrt{m})$ at each iteration
  - Tolerance $\eta = \Theta(m/\sqrt{T} + m^{-1/8})$
  - NPG update learning rate $\alpha = \Theta(1/\sqrt{T})$

Theorem 2 (Convergence Guarantee of CRPO)

With probability at least $1-\delta$, CRPO output satisfies

$$J_R(\pi^*) - \mathbb{E}[J_R(\pi_{\tau_{out} W_{out}})] \le \Theta\left(\frac{1}{\sqrt{T}}\right) + \Theta\left(\frac{1}{m^{1/8}} \log^{\frac{1}{4}}\left(\frac{T\sqrt{m}}{\delta}\right)\right),$$

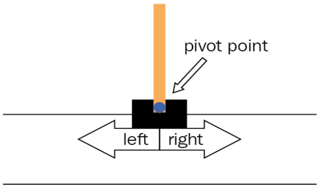
and for all $i = 1, \cdots, p$,

$$\mathbb{E}[J_C(\pi_{\tau_{out} W_{out}})] - c \le \Theta\left(\frac{1}{\sqrt{T}}\right) + \Theta\left(\frac{1}{m^{1/8}} \log^{\frac{1}{4}}\left(\frac{T\sqrt{m}}{\delta}\right)\right).$$

where expectation is on randomness of selecting W_{out} from $\mathcal{N}_0$.

Experiment: CartPole

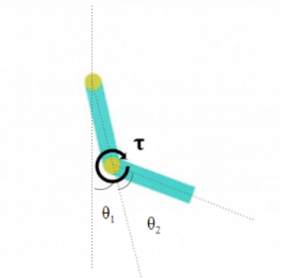
- CartPole setup
  - A pole is attached by an un-actuated joint to a cart
  - The cart moves along a frictionless track over $[-2.4, 2.4]$
  - The pole starts upright
  - Goal: prevent pole from falling over by increasing and reducing cart's velocity.



- MDP environment
  - State space: cart position and velocity, pole angle and angular velocity
  - Action space: push cart to the left, push cart to the right
  - Reward: agent receives a reward $+1$ for every step taken
  - Constraints: agent is penalized with cost $+1$
    - ★ Entering $[-2.4, -2.2]$, $[-1.3, -1.1]$, $[-0.1, 0.1]$, $[1.1, 1.3]$, $[2.2, 2.4]$
    - ★ The angle of pole is larger than 6 degree

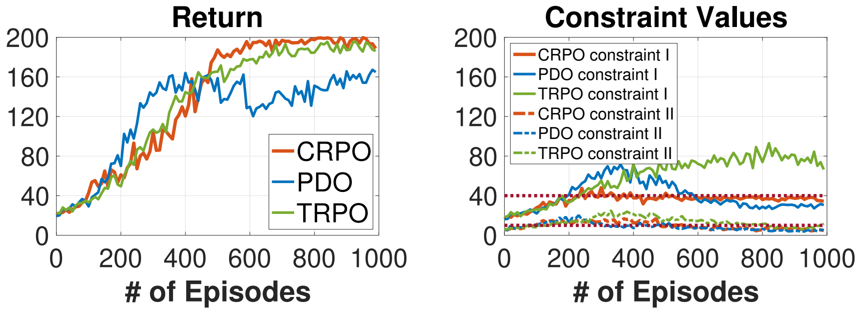
Experiment: Acrobot

- Acrobot setup
  - System includes two joints and two links, where second joint is actuated.
  - Initially, the links are hanging downwards
  - Goal: swing the end of the lower link up to a given height.



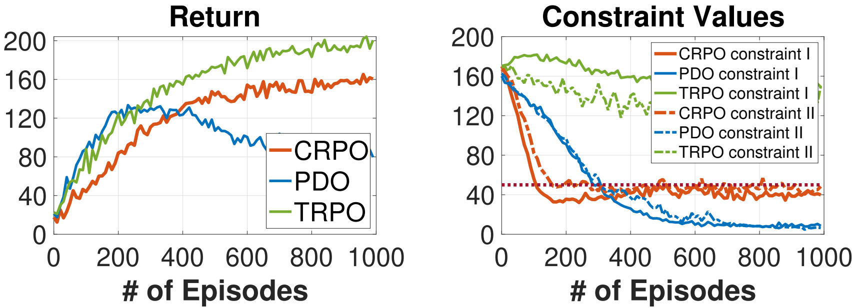
- MDP environment
  - State: two rotational joint angles and the joint angular velocities
  - Action: applying $+1, 0, -1$ torque on the second joint
  - Reward: agent receives a reward $+1$ when the second link is at a height of 0.5
  - Constraints: agent is penalized with cost $+1$
    - ★ Apply a torque $+1$ when the first link swings anticlockwisely
    - ★ Apply a torque $+1$ when the second link swings anticlockwisely with respect to the first link

Comparison of CRPO and Primal-Dual: CartPole



- Convergence
  - CRPO achieves much higher reward
- Constraint violation
  - CRPO tracks constraint thresholds almost exactly, which sufficiently explores boundary of feasible set to optimize reward
  - Primal-Dudal tends to over- or under-enforce the constraints, which results in lower return reward and unstable constraint violation

Comparison of CRPO and Primal-Dual: Acrobot



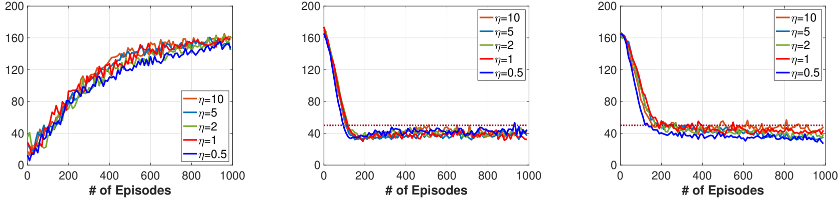
- Convergence
  - CRPO achieves much higher reward
- Constraint violation
  - CRPO drop below thresholds (and thus satisfy the constraints) much faster than that of PDO
  - CRPO tracks constraint thresholds almost exactly, which sufficiently explores boundary of feasible set to optimize reward
  - Primal-Dudal under-enforce constraints, and yields lower reward

Sensitivity to Tuning Parameters

- Primal-dual is very sensitive to stepsize of dual variable's update
  - If stepsize is too small, dual variable updates slowly to enforce constraints
  - If stepsize is too large, algorithm becomes unstable

Sensitivity to Tuning Parameters

- Primal-dual is very sensitive to stepsize of dual variable's update
  - If stepsize is too small, dual variable updates slowly to enforce constraints
  - If stepsize is too large, algorithm becomes unstable
- CRPO is robust with respect to tolerance parameter η



CRPO in Acrobot with η taking different values

Outline

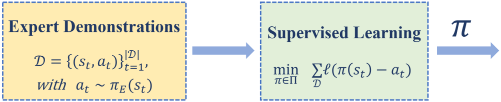
Topic 2: Imitation Learning

- Practical RL applications often encounter:
  - Reward function is unknown
  - Some expert demonstrations are available
  - Goal: find a learner's policy that produces behaviors as close as possible to expert demonstrations
- RL Goal: Learn a desired policy by imitation

Chalodhorn et al., 2007

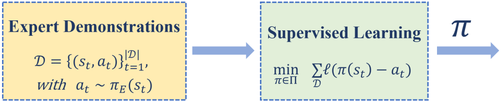
Two Major Approaches on Imitation Learning

- Behavioral Cloning [52]
  - Directly learns a mapping from state to action based on supervised learning to match expert demonstrations

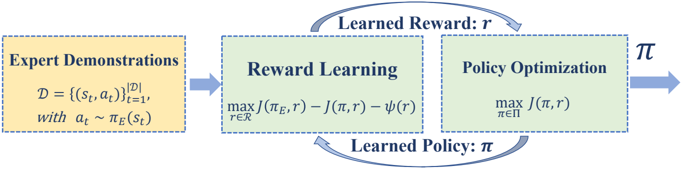


Two Major Approaches on Imitation Learning

- Behavioral Cloning [52]
  - Directly learns a mapping from state to action based on supervised learning to match expert demonstrations



- Inverse Reinforcement Learning [53, 54]
  - First recovers unknown reward function based on expert's trajectories, and then find an optimal policy using such a reward function
  - Generative adversarial imitation learning (GAIL) framework [55]



Generative Adversarial Imitation Learning (GAIL)

- Parameterize reward function as $r_\alpha(s, a)$ where $\alpha \in \Lambda \subset \mathbb{R}^q$
- π_E: expert policy; demonstration samples under π_E are available
- π_w: learner's policy optimized by $w \in \mathcal{W}$
- $J(\pi_E, r_\alpha)$: average value function under expert policy
- $J(\pi_w, r_\alpha)$: average value function under learner's policy
- $\psi(\alpha)$: regularizer of reward parameter

GAIL Framework [55]

$$\min_{w \in \mathcal{W}} \max_{\alpha \in \Lambda} F(w, \alpha) := J(\pi_E, r_\alpha) - J(\pi_w, r_\alpha) - \psi(\alpha)$$

- Maximization: find reward function that best distinguishes between expert's and learner's policies
- Minimization: find learner's policy that matches expert's policy as close as possible

Global Optimum of GAIL

GAIL Framework [55]

$$\min_{w\in\mathcal{W}}\max_{\alpha\in\Lambda} F(w,\alpha) := J(\pi_E, r_\alpha) - J(\pi_w, r_\alpha) - \psi(\alpha)$$

- Define marginal-maximum function $g(w) := \max_{\alpha\in\Lambda} F(w,\alpha)$.
- Let global optimum of GAIL as $w^* = \mathrm{argmin}_{w\in\mathcal{W}}\, g(w)$.
- $\bar{w}$ is ϵ-optimal if $g(\bar{w}) - g(w^*) \le \epsilon$ holds, where $\epsilon \in (0,1)$.

Global Optimum of GAIL

GAIL Framework [55]

$$\min_{w\in\mathcal{W}}\max_{\alpha\in\Lambda} F(w,\alpha) := J(\pi_E, r_\alpha) - J(\pi_w, r_\alpha) - \psi(\alpha)$$

- Define marginal-maximum function $g(w) := \max_{\alpha\in\Lambda} F(w,\alpha)$.
- Let global optimum of GAIL as $w^* = \mathrm{argmin}_{w\in\mathcal{W}}\, g(w)$.
- $\bar{w}$ is ϵ-optimal if $g(\bar{w}) - g(w^*) \le \epsilon$ holds, where $\epsilon \in (0,1)$.

ϵ-optimum of GAIL implies [56]

$$\max_{\alpha\in\Lambda}[J(\pi_E, r_\alpha) - J(\pi_{\bar{w}}, r_\alpha)] \le \max_{\alpha\in\Lambda}\psi(\alpha) + \epsilon.$$

- Properly chosen $\psi(\alpha)$ can guarantee $\pi_{\bar{w}}$ to be sufficiently close to expert policy.

GAIL Policy Gradient Algorithm

- Reward update:
  - Query expert sample $(s^E, a^E) \sim \hat{\mathrm{P}}^{\pi_E}$ and learner's sample $(s^w, a^w) \sim \hat{\mathrm{P}}^{\pi_w}$
  - Estimate stochastic gradient with respect to reward parameter

$$\widehat{\nabla}_\alpha F(w, \alpha) = \left[\nabla_\alpha r_\alpha(s^E, a^E) - \nabla_\alpha r_\alpha(s^w, a^w)\right] - \nabla_\alpha \psi(\alpha)$$

  - Update $\alpha_{k+1} = \mathrm{Proj}\left(\alpha_k + \beta \widehat{\nabla}_\alpha F(w, \alpha_k)\right)$
- Policy update (e.g., by NPG)
  - Estimate natural gradient θ_t via solving

$$\min_{\theta \in R^d} E_{(s,a)\sim\nu_{\pi_w}} \left[A_\alpha^{\pi_w}(s, a) - \nabla_w \log(\pi_w(a|s))^\top \theta\right]^2$$

  - Updated $w_{t+1} = w_t - \eta\theta_t$

Convergence Guarantee of NPG-GAIL

Theorem ([57])

$F(w, \alpha)$ is μ-strongly concave on α. Under other standard assumptions and properly-chosen stepsize, NPG-GAIL converges as

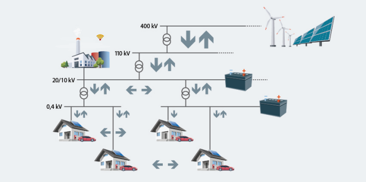
$$\frac{1}{T}\sum_{t=0}^{T-1} E\left[g(w_t)\right] - g(w^*) \leq \mathcal{O}\left(\frac{1}{\sqrt{T}}\right) + \mathcal{O}\left(e^{-K}\right) + \mathcal{O}\left(\frac{1}{B}\right) + \mathcal{O}\left(e^{-T_c}\right) + \mathcal{O}\left(\zeta_{\text{approx}}^{\text{actor}}\right) + \mathcal{O}\left(\lambda\right) + \mathcal{O}\left(\frac{1}{\sqrt{M}}\right)$$

- $\zeta_{\text{approx}}^{\text{actor}}$ is actor approximation error in NPG; K is number of updates of α; B is mini-batch size of α update; T_c is number of updates in value function evaluation in NPG; M is mini-batch size of w update; λ is regularization coefficient in NPG
- NPG-GAIL converges to an $(\epsilon + \mathcal{O}(\zeta_{\text{approx}}^{\text{actor}}))$-accurate *globally* optimal value with an overall sample complexity of $\tilde{\mathcal{O}}\left(\frac{1}{\epsilon^4}\right)$

Outline

Topic 3: Multi-Agent Reinforcement Learning

- RL applications naturally involve multiple agents
  - Left: stock market with numerous investors
  - Middle: multi-drone control
  - Bottom: multi-agent power network



Multi-Agent MDP

- Distributed agents $i = 1, 2, ..., N$;
- Global shared state s;
- Independent policies/actions: $\pi(a|s) = \prod_{i=1}^{N} \pi^i(a^i|s)$;
- Local rewards: $r^i(s, a)$.

Multi-agent MDP trajectory defined by

$$s_0 \xrightarrow{\{\pi^i(\cdot|s_0)\}_{i=1}^{N}} \{a_0^i\}_{i=1}^{N} \xrightarrow{\mathrm{P}(\cdot|s_0,a_0)} (s_1, \{r_0^i\}_{i=1}^{N}) \longrightarrow \cdots$$

Cooperative v.s. Competitive MARL

- Cooperative MARL: Agents cooperate to achieve the same goal;
- Competitive MARL: Agents compete to achieve conflict goals.

Cooperative MARL

- Define global state value function (under joint policy π)

$$V_\pi(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \frac{1}{M} \sum_{m=1}^{M} r_t^{(m)} \middle| s_0 = s, \pi\right]$$

- Agents cooperate to maximize average reward

$$\max_\pi J(\pi) = \mathbb{E}_\xi[V_\pi(s)]$$

- All the agents share the same goal
- Need decentralized synchronization (actions, rewards, etc)
- Study communication & computation complexities

Cooperative MARL: Off-Policy Evaluation

- Given joint policy π, cooperate to evaluate V_π

Decentralized mini-batch TDC [58]

- Agent $i = 1, ..., N$ performs

$$\theta_{t+1}^{i} = \sum_{i' \in \mathcal{N}_i} V_{ii'} \theta_t^{i'} + \frac{\alpha}{n} \sum_{m=tn}^{(t+1)n-1} \rho_m(\delta_m(\theta_t)\phi_{s_m} - \gamma \phi_{s_{m+1}} \phi_{m_t}^{\top} \omega_t)$$

$$\omega_{t+1}^{i} = \sum_{i' \in \mathcal{N}_i} V_{ii'} \omega_t^{i'} + \frac{\beta}{n} \sum_{m=tn}^{(t+1)n-1} (\rho_m \delta_m(\theta_t)\phi_{s_m} - \phi_{s_m} \phi_{s_m}^{\top} \omega_t)$$

- Mini-batch sampling reduces variance and communication frequency
- Local consensus on θ and ω

Cooperative MARL: Off-Policy Evaluation

- Need to estimate global importance sampling ratio $\rho := \prod_{i=1}^{N} \rho^i$
  - Rewrite as $\rho = \exp\left(N \cdot \frac{1}{N}\sum_{i=1}^{N} \ln \rho^i\right)$
  - Synchronize $\frac{1}{N}\sum_{i=1}^{N} \ln \rho^i$ via local averaging

Sample and communication complexities [58]

Choose $\alpha = \mathcal{O}(\frac{1}{\sqrt{N}})$, $\beta = \mathcal{O}(1)$, $n = \mathcal{O}\left(\frac{\sqrt{N}}{\epsilon}\right)$, and run the algorithm for $T = \mathcal{O}\left(\sqrt{N}\ln \epsilon^{-1}\right)$ iterations. Then, for all agents i, the output achieves

$$\mathbb{E}(\|\theta_T^i - \theta^*\|^2) \le \epsilon.$$

The overall sample complexity is $nT = \mathcal{O}(N\epsilon^{-1}\ln \epsilon^{-1})$, and the overall communication complexity is $T = \mathcal{O}\left(\sqrt{N}\ln \epsilon^{-1}\right)$.

Cooperative MARL: Policy Optimization

Decentralized mini-batch actor-critic [59]

- Actor: Agent $i = 1, ..., N$ do

$$\omega_{t+1}^i = \omega_t^i + \alpha \nabla_{\omega^i} J(\omega_t),$$

where the partial policy gradient satisfies

$$\nabla_{\omega^i} J(\omega_t) \approx \left[\bar{r}_t + \gamma V(s'_{t+1}) - V(s_t)\right] \psi_t^i(a_t^i | s_t) \tag{1}$$

- Critic: Agents estimate $V(s)$ via standard decentralized TD

- $\psi_t^i(a_t^i|s_t)$: local score function computed by agent i
- Challenge 1: need $\bar{r}_t$–average reward over all agents. Sensitive!
- Challenge 2: How to achieve low communication & computation complexities at the same time?

- Solve Challenge 1: local averaging over noisy rewards
  - Corrupt local rewards

$$\widetilde{r}^i = r^i\big(1 + \mathcal{N}(0, \sigma^2)\big)$$

  - Estimate $\overline{r}$ via local averaging

$$\overline{r}_0 = \widetilde{r}^i,\ \overline{r}_{t+1} = \textstyle\sum_{i' \in \mathcal{N}_i} W_{ii'}\ \overline{r}_t, \quad t = 0, \ldots, T' - 1.$$

- Solve Challenge 2: Use mini-batch sampling

$$\widehat{\nabla}_{\omega^i} J(\omega_t) = \frac{1}{n} \sum_{m=tn}^{(t+1)n-1} \Big[\overline{r}_m + \gamma V(s'_{m+1}) - V(s_m)\Big] \psi_t^{(i)}(a_m^{(i)} | s_m)$$

  - Suppress reward noise with sufficiently large batch size n
  - Reduces communication frequency

Sample and communication complexity [59]

Choose $\alpha = \mathcal{O}(1), n = \mathcal{O}(\epsilon^{-1})$ and run the algorithm for $T = \mathcal{O}(\epsilon^{-1})$ iterations, the output satisfies $\mathbb{E}(\|\nabla J(\omega_T)\|^2) \le \epsilon$. The overall sample complexity is $\mathcal{O}(\epsilon^{-2} \ln \epsilon^{-1})$, and the overall communication complexity is $\mathcal{O}(\epsilon^{-1} \ln \epsilon^{-1})$.

Competitive MARL

- Define individual state value function for agents $i = 1, ..., N$

$$V_{\pi^i, \pi^{\setminus i}}(s) = \mathbb{E}\big[\sum_{t=0}^{\infty} \gamma^t r_t^i | s_0 = s, \pi\big]$$

- Agents compete to maximize their own reward

$$\max_{\pi^i} V_{\pi^i, \pi^{\setminus i}}(s), \quad \forall s, \forall i = 1, ..., N$$

Nash equilibrium (NE)

- Joint policy π is a NE if for any other policy $\widehat{\pi}$, the following holds.

$$V_{\pi^i, \pi^{\setminus i}}(s) \geq V_{\widehat{\pi}^i, \pi^{\setminus i}}(s), \quad \forall s, \forall i = 1, ..., N$$

Competitive MARL

$$\text{(NE): } V_{\pi^i,\pi^{\setminus i}}(s) \geq V_{\widehat{\pi}^i,\pi^{\setminus i}}(s), \quad \forall s, \forall i = 1, \ldots, N$$

- In general hard to develop efficient algorithms for finding NE
  - Finding NE is PPAD-complete [60]
- However, possible for the following special game

Two-player zero-sum game

- Only two players $N = 2$. Moreover, their rewards sum up to zero, i.e., $r_t^1 + r_t^2 = 0$ for all t.

- Existence of NE is first proved by Shapely in 1953 [61]
- Can be reformulated as linear programming

Two-Player Zero-Sum Game

- Define $r_t := r_t^1 = -r_t^2$ and the following value function

$$V_{\pi^1,\pi^2}(s) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r_t | s_0 = s, \pi\Big]$$

- Two-player zero-sum game reduces to

Two-player zero-sum game

$$\min_{\pi^2} \max_{\pi^1} V_{\pi^1,\pi^2}(s), \quad \forall s$$

- Perfect duality holds, i.e.,

$$\min_{\pi^2} \max_{\pi^1} V_{\pi^1,\pi^2}(s) = \max_{\pi^1} \min_{\pi^2} V_{\pi^1,\pi^2}(s), \quad \forall s$$

Classic Value Iteration for Zero-Sum Game

$$\min_{\pi^2} \max_{\pi^1} V_{\pi^1,\pi^2}(s), \quad \forall s$$

- Classic value iteration

$$\begin{aligned} Q_k(s, a^1, a^2) &= r(s, a^1, a^2) + \gamma \mathbb{E}[V_k(s')], \\ V_{k+1}(s) &= \min_{\pi^2} \max_{\pi^1} \pi^1(s)^\top Q_k(s) \pi^2(s) \end{aligned}$$

- In one-player case, $V_{k+1}(s) = \arg\max_a Q_k(s, a)$
- Requires transition kernel to compute $\mathbb{E}[V_k(s')]$
- Can be shown to converge to the optimal value function
- Challenge: Need to solve matrix game efficiently

Predictive Update via Entropy Regularization

- Smooth the matrix game via entropy regularization

$$
\begin{aligned}
Q_k(s, a^1, a^2) &= r(s, a^1, a^2) + \gamma \mathbb{E}[V_k(s')], \\
V_{k+1}(s) &= \min_{\pi^2} \max_{\pi^1} \pi^1(s)^\top Q_k(s) \pi^2(s) + \tau \mathcal{H}(\pi^1(s)) - \tau \mathcal{H}(\pi^2(s))
\end{aligned}
$$

- Improves bilinear geometry to strongly convex-strongly concave
- Predictive Update algorithm [38]

$$
\text{(PU):} \quad \begin{cases}
\overline{\pi}^1_{k,t+1}(a^1|s) \propto \pi^1_{k,t}(a^1|s)^{1-\eta\tau} \exp\left(\eta Q^1_{k,t}(s, a^1)\right) \\
\overline{\pi}^2_{k,t+1}(a^2|s) \propto \pi^2_{k,t}(a^2|s)^{1-\eta\tau} \exp\left(-\eta Q^2_{k,t}(s, a^2)\right) \\
\pi^1_{k,t+1}(a^1|s) \propto \pi^1_{k,t}(a^1|s)^{1-\eta\tau} \exp\left(\eta \overline{Q}^1_{k,t+1}(s, a^1)\right) \\
\pi^2_{k,t+1}(a^2|s) \propto \pi^2_{k,t}(a^2|s)^{1-\eta\tau} \exp\left(-\eta \overline{Q}^2_{k,t+1}(s, a^2)\right)
\end{cases},
$$

- Decentralized, symmetric, private

Convergence and Complexity

Iteration Complexity [38]

Set $\eta = \mathcal{O}(\frac{1-\gamma}{2(1+\tau(\ln|\mathcal{A}|+1-\gamma))})$ and $\tau = \mathcal{O}(\frac{(1-\gamma)\epsilon}{\ln|\mathcal{A}|})$, and run the algorithm for $T = \mathcal{O}(\frac{1}{(1-\gamma)^3\epsilon})$ iterations. Then, the output achieves ϵ-NE, i.e.,

$$\max_{\mu} V_{\mu,\pi^2}(s) - \min_{\nu} V_{\pi^1,\nu}(s) \leq \epsilon.$$

- Our recent work proposes a sample-based stochastic version [62]
- Developed Monte Carlo estimators with Markovian samples to estimate $\mathbb{E}[V_k(s')]$, $Q_{k,t}(s,a)$, $\overline{Q}_{k,t}(s,a)$
- Achieve sample complexity $\mathcal{O}(\frac{|\mathcal{A}|}{\epsilon^{5.5}(1-\gamma)^{13.5}})$, improves the SOTA $\mathcal{O}(\frac{|\mathcal{A}|^3|\mathcal{S}|^{10.5}}{\epsilon^8(1-\gamma)^{29.5}})$ [63]

Outline

Topic 4: Robust Reinforcement Learning

- Motivations:
  - Possible model deviation between training and test environments, e.g., training is on simulator, model is from empirical estimate
  - Adversarial attacks to MDPs
  - These could lead to severe performance degradation

Topic 4: Robust Reinforcement Learning

- Motivations:
  - Possible model deviation between training and test environments, e.g., training is on simulator, model is from empirical estimate
  - Adversarial attacks to MDPs
  - These could lead to severe performance degradation
- Robust Markov decision process: $(\mathcal{S}, \mathcal{A}, r, \mathrm{P})$, where $\mathrm{P} \in \mathcal{P}$, and $\mathcal{P}$ is an uncertainty set of transition kernels
  - Reward function r could also be uncertain

Topic 4: Robust Reinforcement Learning

- Motivations:
  - Possible model deviation between training and test environments, e.g., training is on simulator, model is from empirical estimate
  - Adversarial attacks to MDPs
  - These could lead to severe performance degradation
- Robust Markov decision process: $(\mathcal{S}, \mathcal{A}, r, \mathrm{P})$, where $\mathrm{P} \in \mathcal{P}$, and $\mathcal{P}$ is an uncertainty set of transition kernels
  - Reward function r could also be uncertain
- Examples of uncertainty set: let $\hat{p}_s^a$ denote centroid transition kernel, e.g., empirical estimate and simulator
  - Relative entropy: $\mathcal{P}_s^a = \{p : D(p \| \hat{p}_s^a) \leq \varepsilon\}$
  - Total variation: $\mathcal{P}_s^a = \{p : TV(p \| \hat{p}_s^a) \leq \varepsilon\}$
  - $\mathcal{P} = \bigotimes_{s \in \mathcal{S}, a \in \mathcal{A}} \mathcal{P}_s^a$

Robust Reinforcement Learning

- P_t: transition kernel at time t, and $P_t \in \mathcal{P}$
- Dynamic model: P_t for different t are allowed to be different
- Static model: $P_{t_1} = P_{t_2}$, for any $t_1, t_2 \geq 0$

Equivalence

Solutions to dynamic model and static model are equivalent under rectangular uncertainty set [64]

Robust Reinforcement Learning

- P_t: transition kernel at time t, and $\mathrm{P}_t \in \mathcal{P}$
- Dynamic model: P_t for different t are allowed to be different
- Static model: $\mathrm{P}_{t_1} = \mathrm{P}_{t_2}$, for any $t_1, t_2 \geq 0$

Equivalence

Solutions to dynamic model and static model are equivalent under rectangular uncertainty set [64]

- Robust value function:
  $\tilde{V}^\pi(s) = \inf_{\mathrm{P}_t \in \mathcal{P}, t \geq 0} \mathbb{E}_{\mathrm{P}_t, t\geq 0} \left[\sum_{t=0}^{\infty} \gamma^t r_t | S_0 = s, \pi\right]$
- Robust action value function:
  $\tilde{Q}^\pi(s,a) = \inf_{\mathrm{P}_t \in \mathcal{P}, t \geq 0} \mathbb{E}_{\mathrm{P}_t, t\geq 0} \left[\sum_{t=0}^{\infty} \gamma^t r_t | S_0 = s, A_0 = a, \pi\right]$
  - A pessimistic approach that optimizes the worst-case performance
- Goal: Learn policy robust to model uncertainty

$$\tilde{V}^*(s) = \max_\pi \tilde{V}^\pi(s), \forall s \in \mathcal{S}$$

Model-Based Approach

- Assume that uncertainty set $\mathcal{P}$ is *known*
- Robust Bellman operator:

$$\tilde{\mathrm{T}}\tilde{V}(s) = \max_{a \in \mathcal{A}} r(s,a) + \gamma \sigma_{\mathcal{P}_{s,a}}(\tilde{V}),$$

where $\sigma_{\mathcal{P}_{s,a}}(\tilde{V})$ is the support function: $\sigma_{\mathcal{P}_{s,a}}(\tilde{V}) = \sup_{p \in \mathcal{P}_{s,a}} p^\top \tilde{V}$

Theorem (Contraction [65, 64])

$\tilde{\mathrm{T}}$ *is a contraction in* ℓ_∞ *norm, and its unique fixed point is* $\tilde{V}^*$

- $\tilde{V}^*$ can be solved by robust value/policy iteration

Adversarial Training Approach

- Approach 1:
  - Reformulate robust RL as a game between agent and nature, where nature chooses transition kernel $\mathsf{P}_t \in \mathcal{P}, t \geq 0$
    $\max_{\pi} \inf_{\mathsf{P}_t \in \mathcal{P}, t \geq 0} \mathbb{E}_{\mathsf{P}_t, t \geq 0} \left[\sum_{t=0}^{\infty} \gamma^t r_t | S_0 = s, \pi\right]$
  - Alternatively optimize agent's policy towards maximizing reward and nature's policy towards minimizing reward
- Approach 2:
  - Adversarially perturb the state observation

Adversarial Training Approach

- Approach 1:
  - Reformulate robust RL as a game between agent and nature, where nature chooses transition kernel $\mathsf{P}_t \in \mathcal{P}, t \geq 0$
    $\max_\pi \inf_{\mathsf{P}_t \in \mathcal{P}, t \geq 0} \mathbb{E}_{\mathsf{P}_t, t \geq 0} \left[\sum_{t=0}^{\infty} \gamma^t r_t | S_0 = s, \pi \right]$
  - Alternatively optimize agent's policy towards maximizing reward and nature's policy towards minimizing reward
- Approach 2:
  - Adversarially perturb the state observation
- Empirical success, but lack of theoretical convergence and robustness guarantee
- References: [66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79]

Model-free Approach

- Uncertainty set is centered at an unknown MDP from which samples can be taken
- Goal: design principled online robust RL algorithm

Additive Uncertainty Set [80, 81]

- Uncertainty set: $\mathcal{P}_s^a = \{p_s^a + x | x \in \mathcal{U}_s^a\}$
  - p_s^a is simulator transition kernel, from which samples are taken
  - $\mathcal{U}_s^a$ is the confidence region

    e.g., ellipsoid

    $\mathcal{U}_s^a = \{x : x^\top A_s^a x \leq 1, \sum_i x_i = 0, -p_s^a(i) \leq x_i \leq 1 - p_s^a(i)\}$

Additive Uncertainty Set [80, 81]

- Uncertainty set: $\mathcal{P}_s^a = \{p_s^a + x | x \in \mathcal{U}_s^a\}$
  - p_s^a is simulator transition kernel, from which samples are taken
  - $\mathcal{U}_s^a$ is the confidence region
    e.g., ellipsoid
    $\mathcal{U}_s^a = \{x : x^\top A_s^a x \le 1, \sum_i x_i = 0, -p_s^a(i) \le x_i \le 1 - p_s^a(i)\}$
- Robust TD, Q-learning, SARSA [80]
- Robust least squares policy evaluation and robust least squares policy iteration [81]
- Basic idea:
  - a stochastic implementation of robust Bellman operator
  - when calculate support function $\sigma_{\mathcal{P}_{s,a}}(\tilde{V})$, relax $\mathcal{U}_s^a$ to $\hat{\mathcal{U}}_s^a$
    $\hat{\mathcal{U}}_s^a = \{x : x^\top A_s^a x \le 1, \sum_i x_i = 0\}$
  - issue: $p_s^a + x, x \in \hat{\mathcal{U}}_s^a$ may not be a probability distribution anymore
- Converge if discount factor γ is much less than 1 (to offset error caused by relaxation)

ε-Contamination Uncertainty Set

- ε-contamination uncertainty set:

$$\mathcal{P}_s^a = \{(1-\varepsilon)p_s^a + \varepsilon q | q \in \Delta(\mathcal{S})\}, \text{ for some } 0 \le \varepsilon \le 1$$

  where $\Delta(\mathcal{S})$ is the probability simplex on $\mathcal{S}$

- Interpretation: with probability $1-\varepsilon$, state transition is perturbed to any arbitrary distribution $q \in \Delta(\mathcal{S})$
- Algorithm and results can be similarly obtained for case with $\Delta(\mathcal{S})$ replaced by a set that depends on s, a

ε-Contamination Uncertainty Set

- ε-contamination uncertainty set:

$$\mathcal{P}_s^a = \left\{(1-\varepsilon)p_s^a + \varepsilon q | q \in \Delta(\mathcal{S})\right\}, \text{ for some } 0 \le \varepsilon \le 1$$

  where $\Delta(\mathcal{S})$ is the probability simplex on $\mathcal{S}$
- Interpretation: with probability $1-\varepsilon$, state transition is perturbed to any arbitrary distribution $q \in \Delta(\mathcal{S})$
- Algorithm and results can be similarly obtained for case with $\Delta(\mathcal{S})$ replaced by a set that depends on s, a
- ε-Contamination model (Huber in [82]) has been widely used to model distributional uncertainty in the literature
- ε-contamination can be related to total-variation/KL divergence defined uncertainty set via Pinsker's inequality

Robust Q-learning [83]

Initialization: T, $\tilde{Q}_0(s,a)$ for all (s,a), behavior policy π_b, s_0, step size α_t
For $t = 0, 1, 2, \ldots, T-1$
Choose a_t according to $\pi_b(\cdot|s_t)$
Observe s_{t+1} and r_t
Update $\tilde{Q}_{t+1}$:

$$\tilde{V}_t(s) \leftarrow \max_{a \in \mathcal{A}} \tilde{Q}_t(s,a), \forall s \in \mathcal{S}$$

$$\tilde{Q}_{t+1}(s_t,a_t) \leftarrow (1-\alpha_t)\tilde{Q}_t(s_t,a_t) + \alpha_t \underbrace{\left(r_t + \gamma\left((1-\varepsilon)\tilde{V}_t(s_{t+1}) + \varepsilon \min_{s \in \mathcal{S}} \tilde{V}_t(s)\right)\right)}_{\text{target, one-step bootstrap}}$$

Output: $\tilde{Q}_T$

Convergence and Sample Complexity [83]

Theorem (Asymptotic Convergence)

If step sizes α_t satisfy that $\sum_{t=0}^{\infty} \alpha_t = \infty$ and $\sum_{t=0}^{\infty} \alpha_t^2 < \infty$, then $\tilde{Q}_t \to \tilde{Q}^*$ as $t \to \infty$ almost surely.

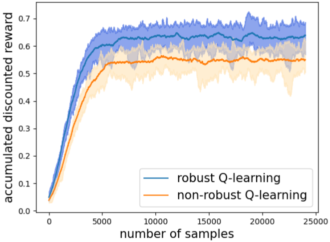
Theorem (Finite-Time Error Bound)

For any ϵ, when $T = \tilde{\mathcal{O}}(\frac{1}{\mu_{\min}(1-\gamma)^5\epsilon^2} + \frac{t_{mix}}{\mu_{\min}(1-\gamma)})$, $\|\tilde{Q}_T - \tilde{Q}^*\| \leq \epsilon$.

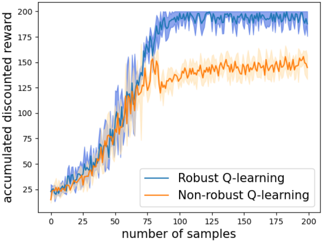
- $t_{\text{mix}} = \min\left\{t : \max_{s\in\mathcal{S}} d_{\text{TV}}(\mu_\pi, P(s_t = \cdot|s_0 = s)) \leq \frac{1}{4}\right\}$ measures the mixing time under behavior policy π_b
- $\mu_{\min} = \min_{(s,a)\in\mathcal{S}\times\mathcal{A}} \mu_{\pi_b}(s, a)$: how many samples are needed to visit every state-action pair sufficiently often
- Robust Q-learning converges to $\tilde{Q}^*$
- Same sample and computational complexity (within a constant factor) as vanilla Q-learning algorithm

Experiments on Robust Q-Learning

- Train Q-learning and robust Q-learning under a perturbed MDP
- Test on real unperturbed environment
- Robust Q-learning achieves higher reward than vanilla Q-learning



(a) FrozenLake



(b) Cartpole

Robust TDC with Linear Function Approximation

- Large state/action space
- Policy evaluation: for any policy π, evaluate its performance under worst-case transition kernel:

$$\tilde{V}^{\pi}(s) = \min_{\mathsf{P}_t \in \mathcal{P}, t \geq 0} \mathbb{E}_{\mathsf{P}_t \in \mathcal{P}, t \geq 0} \left[\sum_{t=0}^{\infty} \gamma^t r_t | S_0 = s, \pi \right]$$

- Linear function approximation: find $V_\theta(s) = \theta^\top \phi(s)$ for a family of base functions $\phi(s) \in \mathbb{R}^N$, such that $V_\theta \approx V^\pi$
- Why no robust TD with function approximation? It may divergence since it is essentially "off-transition-kernel" (similar to off-policy)

Robust TDC with Linear Function Approximation

- Robust Bellman operator (for policy evaluation):

$$\tilde{\mathsf{T}}_\pi V(s) \triangleq \mathbb{E}_{A\sim\pi(\cdot|s)}[r(s,A) + \gamma\sigma_{\mathcal{P}_s^A}(V)]$$

- $\tilde{V}^\pi$ is the fixed point of $\tilde{\mathsf{T}}_\pi$
- Minimize the mean squared projected robust Bellman error (MSPRBE)

$$\min_\theta \mathrm{MSPRBE}(\theta) = \left\| \prod \tilde{\mathsf{T}}_\pi V_\theta - V_\theta \right\|_{\mu_\pi}^2$$

- $\min_{s'\in\mathcal{S}} V(s')$ is non-differentiable and brings difficulties in algorithm design and analysis
- Use a smoothed robust Bellman operator to approximate robust Bellman operator

Robust TDC with Linear Function Approximation

- Use LogSumExp to approximate min: $\mathrm{LSE}(V) = -\frac{\log\left(\sum_s e^{-\rho V(s)}\right)}{\rho}$
- Smoothed Robust Bellman operator: $\hat{\mathrm{T}}_\pi V(s) = \mathbb{E}_{A\sim\pi(\cdot|s)}\left[r(s,A) + \gamma(1-R)\sum_{s'\in\mathcal{S}} p^A_{s,s'} V(s') + \gamma R \cdot \mathrm{LSE}(V)\right]$

Theorem (Contraction [83])

$\hat{\mathrm{T}}_\pi$ *is a contraction and has a unique fixed point (denoted by* $\hat{V}^\pi$*). Moreover,* $\hat{V}^\pi \to \tilde{V}^\pi$ *as* $\rho \to \infty$.

- Goal: minimize smoothed mean squared projected robust Bellman error (SMSPRBE):

$$\min_\theta J(\theta) := \min_\theta \left\| \prod \hat{\mathrm{T}}_\pi V_\theta - V_\theta \right\|^2_{\mu_\pi}$$

Robust TDC with Linear Function Approximation

Input: T, α, β, ρ, ϕ_i for $i = 1, ..., N$, projection radius K
Initialization: θ_0, w_0, s_0
Choose $W \sim \text{Uniform}(0, 1, ..., T-1)$
For $t = 0, 1, 2, ..., W-1$

Take action according to $\pi(\cdot|s_t)$ and observe s_{t+1} and c_t

$\phi_t \leftarrow \phi_{s_t}$

$\delta_t(\theta_t) \leftarrow r_t + \gamma(1-R)V_{\theta_t}(s_{t+1}) - \gamma R \frac{\log(\sum_s e^{-\rho\theta^\top \phi_s})}{\rho} - V_{\theta_t}(s_t)$

$\theta_{t+1} \leftarrow$

$$\prod_K \left(\theta_t + \alpha\left(\delta_t(\theta_t)\phi_t - \gamma\left((1-R)\phi_{t+1} + R\sum_{s\in\mathcal{S}}\left(\frac{e^{-\rho V_\theta(s)}\phi_s}{\sum_{j\in\mathcal{S}} e^{-\rho V_\theta(j)}}\right)\right)\phi_t^\top \omega_t\right)\right)$$

$\omega_{t+1} \leftarrow \prod_K(\omega_t + \beta(\delta_t(\theta_t) - \phi_t^\top \omega_t)\phi_t)$

Output: θ_W

Results on Robust TDC

Theorem 3 (Robust TDC [83])

Define step-sizes: $\beta = \mathcal{O}\left(\frac{1}{T^b}\right)$, $\alpha = \mathcal{O}\left(\frac{1}{T^a}\right)$, *where* $\frac{1}{2} < a \leq 1$ *and* $0 < b \leq a$. *Then*

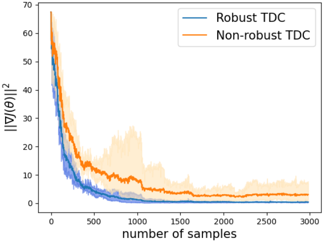
$$\mathbb{E}[\|\nabla J(\theta_W)\|^2] = \mathcal{O}\left(\frac{1}{T\alpha} + \alpha \log(1/\alpha) + \frac{1}{T\beta} + \beta \log(1/\beta)\right).$$

If $a = b = 0.5$, *then*

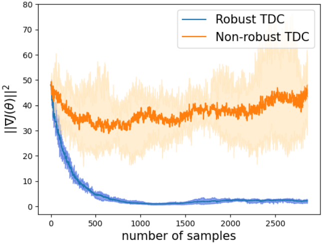
$$\mathbb{E}[\|\nabla J(\theta_W)\|^2] = \mathcal{O}\left(\frac{\log T}{\sqrt{T}}\right).$$

Experiments on Robust TDC

- Train TDC and robust TDC under a perturbed MDP
- Test on real unperturbed environment
- Robust TDC converges to stationary points faster than TDC
- TDC may even diverge



(a)



(b)

Open Problems in Reinforcement Learning

- Multi-task reinforcement learning
  - Tasks can share similar but different transition kernels
  - Meta-learning can be applied to achieve sampling efficiency
  - Open issues in theory: characterization of sample complexity improvement due to meta-learning
- Off-policy/Offline reinforcement learning
  - No access to online interaction with environment, but access only to a given set of data samples
  - Dataset has limited coverage over state-action space, and is sampled under behavior policy, not target policy
  - Open issues in design: how to design desirable algorithms to address overestimation and distribution shift
  - Open issues in theory: what is the minimum requirement to achieve polynomial sample complexity efficiency

Open Problems (Cont.)

- Partially observable MDP
  - No access to full state information
  - Optimal policy is not stationary
  - Markovian structure does not hold anymore
  - Open issues in design: how to design efficient model-free and model-based methods
  - Open issues in theory: how to characterize sample complexity
- Multi-agent RL
  - Multiple agents interact collaboratively or competitively
  - Decentralized algorithms under partial observations of environments
  - Challenges in design: delayed communication; communication depends on network topology; curse of dimensionality
  - Open issues in theory: tradeoff among communications, computations, privacy; equilibrium; sample complexity

Questions?

References

[1] N. Sharma, S. Zhang, S. R. S. Venkata, F. Malandra, N. Mastronarde, and J. Chakareski, “Deep reinforcement learning for delay-sensitive lte downlink scheduling,” in *IEEE Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 1–6, IEEE, 2020.

[2] R. S. Sutton, “Learning to predict by the methods of temporal differences,” *Machine learning*, vol. 3, no. 1, pp. 9–44, 1988.

[3] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*.
2018.

[4] J. N. Tsitsiklis and B. Van Roy, “An analysis of temporal-difference learning with function approximation,” *IEEE transactions on automatic control*, vol. 42, no. 5, pp. 674–690, 1997.

[5] V. S. Borkar, *Stochastic approximation: a dynamical systems viewpoint*, vol. 48.
2009.

[6] A. Benveniste, P. Priouret, and M. Métivier, *Adaptive Algorithms and Stochastic Approximations*.
Springer-Verlag, 1990.

[7] V. Tadić, "On the convergence of temporal-difference learning with linear function approximation," *Machine learning*, vol. 42, no. 3, pp. 241–267, 2001.

[8] G. Dalal, B. Szörényi, G. Thoppe, and S. Mannor, "Finite sample analyses for td (0) with function approximation," in *Proc. Association for the Advancement of Artificial Intelligence (AAAI)*, vol. 32, 2018.

[9] R. Srikant and L. Ying, "Finite-time error bounds for linear stochastic approximation andtd learning," in *Proc. Conference on Learning Theory (COLT)*, pp. 2803–2830, 2019.

[10] J. Bhandari, D. Russo, and R. Singal, "A finite time analysis of temporal difference learning with linear function approximation," in *Proc. Conference on Learning Theory (COLT)*, vol. 75, pp. 1691–1692, 2018.

[11] L. Baird, "Residual algorithms: Reinforcement learning with function approximation," in *Proc. International Conference on Machine Learning (ICML)*, pp. 30–37, 1995.

[12] R. S. Sutton, H. R. Maei, D. Precup, S. Bhatnagar, D. Silver, C. Szepesvári, and E. Wiewiora, "Fast gradient-descent methods for temporal-difference learning with linear function approximation," in *Proc. International Conference on Machine Learning (ICML)*, pp. 993–1000, 2009.

[13] H. R. Maei, *Gradient temporal-difference learning algorithms*. PhD thesis, University of Alberta, 2011.

[14] H. Yu, "On convergence of some gradient-based temporal-differences algorithms for off-policy learning," *arXiv1712.09652*, 2018.

[15] M. Kaledin, E. Moulines, A. Naumov, V. Tadic, and H.-T. Wai, "Finite time analysis of linear two-timescale stochastic approximation with markovian noise," in *Proc. Conference on Learning Theory (COLT)*, pp. 2144–2203, 2020.

[16] T. Xu, S. Zou, and Y. Liang, "Two time-scale off-policy td learning: Non-asymptotic analysis over markovian samples," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pp. 10634–10644, 2019.

[17] T. Xu and Y. Liang, "Sample complexity bounds for two timescale value-based reinforcement learning algorithms," in *Proc. International Conference on Artificial Intelligence and Statistics*, vol. 130, pp. 811–819, 13–15 Apr 2021.

[18] F. S. Melo, S. P. Meyn, and M. I. Ribeiro, "An analysis of reinforcement learning with function approximation," in *Proc. International Conference on Machine Learning (ICML)*, pp. 664–671, ACM, 2008.

[19] S. Zou, T. Xu, and Y. Liang, "Finite-sample analysis for sarsa with linear function approximation," in *Proc. Advances in Neural Information Processing Systems*, pp. 8665–8675, 2019.

[20] G. Li, C. Cai, Y. Chen, Y. Gu, Y. Wei, and Y. Chi, "Is q-learning minimax optimal? a tight sample complexity analysis," *arXiv preprint arXiv:2102.06548*, 2021.

[21] M. J. Wainwright, "Variance-reduced q-learning is minimax optimal," *arXiv preprint arXiv:1906.04697*, 2019.

[22] G. Li, Y. Wei, Y. Chi, Y. Gu, and Y. Chen, "Sample complexity of asynchronous q-learning: Sharper analysis and variance reduction," *arXiv preprint arXiv:2006.03041*, 2020.

[23] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, and G. Ostrovski, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529–533, 2015.

[24] H. R. Maei, C. Szepesvári, S. Bhatnagar, and R. S. Sutton, "Toward off-policy learning control with function approximation," in *Proc. International Conference on Machine Learning (ICML)*, 2010.

[25] Y. Wang and S. Zou, "Finite-sample analysis of Greedy-GQ with linear function approximation under Markovian noise," in *Proc. International Conference on Uncertainty in Artificial Intelligence (UAI)*, vol. 124, pp. 11–20, 2020.

[26] T. Xu and Y. Liang, "Sample complexity bounds for two timescale value-based reinforcement learning algorithms," *ArXiv:2011.05053*, 2020.

[27] R. Johnson and T. Zhang, "Accelerating stochastic gradient descent using predictive variance reduction," in *Proc. Advances in Neural Information Processing Systems*, vol. 26, 2013.

[28] S. Ma, Z. Chen, Y. Zhou, and S. Zou, "Greedy-GQ with variance reduction: Finite-time analysis and improved complexity," in *Proc. International Conference on Learning Representations (ICLR)*, 2021.

[29] R. S. Sutton, D. A. McAllester, S. P. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," in *Proc. Advances in Neural Information Processing Systems (NIPS)*, pp. 1057–1063, 2000.

[30] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine Learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[31] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *The 32nd International Conference on Machine Learning (ICML)*, pp. 1889–1897, 2015.

[32] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[33] S. M. Kakade, "A natural policy gradient," in *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1531–1538, 2002.

[34] A. Agarwal, S. M. Kakade, J. D. Lee, and G. Mahajan, "Optimality and approximation with policy gradient methods in Markov decision processes," *arXiv preprint arXiv:1908.00261*, 2019.

[35] G. Lan, "Policy mirror descent for reinforcement learning: Linear convergence, new sampling complexity, and generalized problem classes," *ArXiv:2102.00135*, 2021.

[36] J. Bhandari and D. Russo, "Global optimality guarantees for policy gradient methods," *arXiv preprint arXiv:1906.01786*, 2019.

[37] L. Shani, Y. Efroni, and S. Mannor, "Adaptive trust region policy optimization: Global convergence and faster rates for regularized MDPs," *arXiv preprint arXiv:1909.02769*, 2019.

[38] S. Cen, C. Cheng, Y. Chen, Y. Wei, and Y. Chi, "Fast global convergence of natural policy gradient methods with entropy regularization," *arXiv:2007.06558*, 2020.

[39] V. Konda, "Actor-critic algorithms (ph.d. thesis)," *Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology*, 2002.

[40] H. Xiong, T. Xu, YingbinLiang, and W. Zhang, "Non-asymptotic convergence of Adam-type reinforcement learning algorithms under

Markovian sampling," in *Proc. AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

[41] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, pp. 1008–1014, 2000.

[42] T. Xu, Z. Wang, and Y. Liang, "Improving sample complexity bounds for (natural) actor-critic algorithms," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, also available as *arXiv preprint arXiv:2004.12956*, 2020.

[43] S.-I. Amari, "Natural gradient works efficiently in learning," *Neural Computation*, vol. 10, no. 2, pp. 251–276, 1998.

[44] T. Xu, Z. Yang, Z. Wang, and Y. Liang, "Doubly robust off-policy actor-critic: Convergence and optimality," in *Proc. International Conference on Machine Learning (ICML)*, 2021.

[45] E. Altman, *Constrained Markov Decision Processes*, vol. 7. CRC Press, 1999.

[46] S. Paternain, L. F. Chamon, M. Calvo-Fullana, and A. Ribeiro, "Constrained reinforcement learning has zero duality gap," in *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2019.

[47] J. Achiam, D. Held, A. Tamar, and P. Abbeel, "Constrained policy optimization," in *Proc. International Conference on Machine Learning (ICML)*, pp. 22–31, 2017.

[48] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone, "Risk-constrained reinforcement learning with percentile risk criteria," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6070–6120, 2017.

[49] D. Ding, K. Zhang, T. Basar, and M. Jovanovic, "Natural policy gradient primal-dual method for constrained markov decision processes," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020.

[50] T. Li, Z. Guan, S. Zou, T. Xu, Y. Liang, and G. Lan, "Faster algorithm and sharper analysis for constrained Markov decision process," *arXiv preprint arXiv:2110.10351*, 2021.

[51] T. Xu, Y. Liang, and G. Lan, "CRPO: A new approach for safe reinforcement learning with convergence guarantee," in *Proc. International Conference on Machine Learning (ICML)*, 2021.

[52] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation.," *Neural Computation*, vol. 3, no. 1, pp. 88–97, 1991.

[53] S. Russell, "Learning agents for uncertain environments," in *Proc. Eleventh Annual Conference on Computational Learning Theory*, 1998.

[54] A. Y. Ng and S. Russell, "Algorithms for inverse reinforcement learning," in *Proc. International Conference on Machine Learning (ICML)*, 2000.

[55] J. Ho and S. Ermon, "Generative adversarial imitation learning," in *Proc. Advances in Neural Information Processing Systems (NIPS)*, 2016.

[56] Y. Zhang, Q. Cai, Z. Yang, and Z. Wang, "Generative adversarial imitation learning with neural networks: Global optimality and convergence rate," *arXiv preprint arXiv:2003.03709*, 2020.

[57] Z. Guan, T. Xu, and Y. Liang, "When will generative adversarial imitation learning algorithms attain global convergence," in *Proc. International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021.

[58] Z. Chen, Y. Zhou, and R. Chen, "Multi-agent off-policy td learning: Finite-time analysis with near-optimal sample complexity and communication complexity," *ArXiv:2103.13147*, 2021.

[59] Z. Chen, Y. Zhou, R. Chen, and S. Zou, "Sample and communication-efficient decentralized actor-critic algorithms with finite-time analysis," *arXiv:2109.03699*, 2021.

[60] X. Deng, Y. Li, D. Mguni, J. Wang, and Y. Yang, "On the complexity of computing markov perfect equilibrium in general-sum stochastic games," *Electronic Colloquium on Computational Complexity (ECCC)*, vol. 28, p. 128, 2021.

[61] L. S. Shapley, "Stochastic games*," *Proceedings of the National Academy of Sciences*, vol. 39, no. 10, pp. 1095–1100, 1953.

[62] Z. Chen, S. Ma, and Y. Zhou, "Sample efficient stochastic policy extragradient algorithm for zero-sum markov game," in *International Conference on Learning Representations*, 2022.

[63] C.-Y. Wei, C.-W. Lee, M. Zhang, and H. Luo, "Last-iterate convergence of decentralized optimistic gradient descent/ascent in infinite-horizon competitive markov games," in *Proc. Conference on Learning Theory (COLT)*, 2021.

[64] G. N. Iyengar, "Robust dynamic programming," *Mathematics of Operations Research*, vol. 30, no. 2, pp. 257–280, 2005.

[65] A. Nilim and L. El Ghaoui, "Robustness in Markov decision problems with uncertain transition matrices.," in *NIPS*, pp. 839–846, Citeseer, 2003.

[66] E. Vinitsky, Y. Du, K. Parvate, K. Jang, P. Abbeel, and A. Bayen, "Robust reinforcement learning using adversarial populations," *arXiv preprint arXiv:2008.01825*, 2020.

[67] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta, "Robust adversarial reinforcement learning," in *International Conference on Machine Learning*, pp. 2817–2826, PMLR, 2017.

[68] M. A. Abdullah, H. Ren, H. B. Ammar, V. Milenkovic, R. Luo, M. Zhang, and J. Wang, "Wasserstein robust reinforcement learning," *arXiv preprint arXiv:1907.13196*, 2019.

[69] L. Hou, L. Pang, X. Hong, Y. Lan, Z. Ma, and D. Yin, "Robust reinforcement learning with wasserstein constraint," *arXiv preprint arXiv:2006.00945*, 2020.

[70] A. Rajeswaran, S. Ghotra, B. Ravindran, and S. Levine, "Epopt: Learning robust neural network policies using model ensembles," *arXiv preprint arXiv:1610.01283*, 2016.

[71] C. G. Atkeson and J. Morimoto, "Nonparametric representation of policies and value functions: A trajectory-based approach," in *Proc. Advances in Neural Information Processing Systems (NIPS)*, pp. 1643–1650, 2003.

[72] J. Morimoto and K. Doya, "Robust reinforcement learning," *Neural computation*, vol. 17, no. 2, pp. 335–359, 2005.

[73] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," *arXiv preprint arXiv:1702.02284*, 2017.

[74] J. Kos and D. Song, "Delving into adversarial attacks on deep policies," *arXiv preprint arXiv:1705.06452*, 2017.

[75] Y.-C. Lin, Z.-W. Hong, Y.-H. Liao, M.-L. Shih, M.-Y. Liu, and M. Sun, "Tactics of adversarial attack on deep reinforcement learning agents," *arXiv preprint arXiv:1703.06748*, 2017.

[76] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary, "Robust deep reinforcement learning with adversarial attacks," *arXiv preprint arXiv:1712.03632*, 2017.

[77] A. Mandlekar, Y. Zhu, A. Garg, L. Fei-Fei, and S. Savarese, "Adversarially robust policy learning: Active construction of physically-plausible perturbations," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 3932–3939, IEEE, 2017.

[78] S. H. Lim, H. Xu, and S. Mannor, "Reinforcement learning in robust Markov decision processes," *Proc. Advances in Neural Information Processing Systems (NIPS)*, vol. 26, pp. 701–709, 2013.

[79] K. Zhang, B. Hu, and T. Basar, "On the stability and convergence of robust adversarial reinforcement learning: A case study on linear quadratic systems," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[80] A. Roy, H. Xu, and S. Pokutta, "Reinforcement learning under model mismatch," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pp. 3046–3055, 2017.

[81] K. P. Badrinath and D. Kalathil, "Robust reinforcement learning using least squares policy iteration with provable performance guarantees," in *International Conference on Machine Learning*, pp. 511–520, PMLR, 2021.

[82] P. J. Huber, "A robust version of the probability ratio test," *Ann. Math. Statist.*, vol. 36, pp. 1753–1758, 1965.

[83] Y. Wang and S. Zou, "Online robust reinforcement learning with model uncertainty," in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2021.