# Optimization Meets Reinforcement Learning

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

**IEEE BigData 2021**

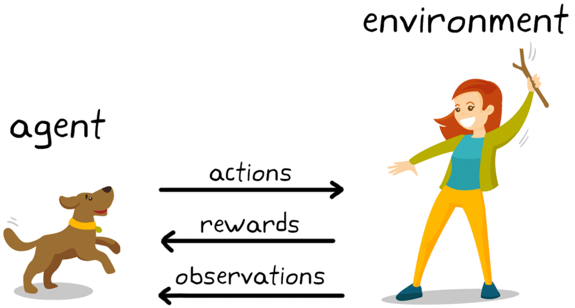
Dec 17, 2021

# Outline

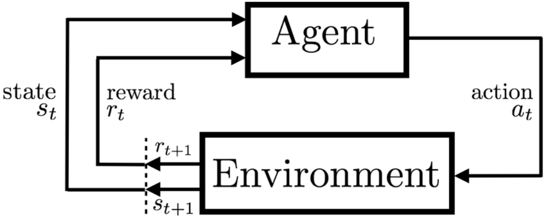
# Outline

# 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

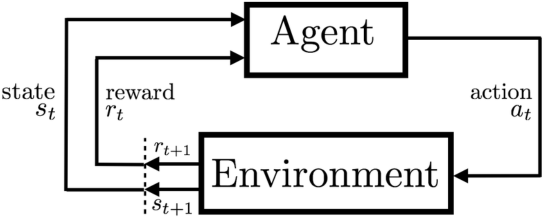


# Markov Decision Process (MDP)




- 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 $s$

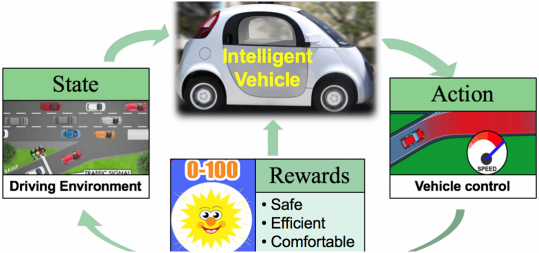
MDP trajectory $\{s_t, a_t, r_t, s_{t+1}\}_{t=0}^{\infty}$ defined by

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

- Randomness: actions, state transitions

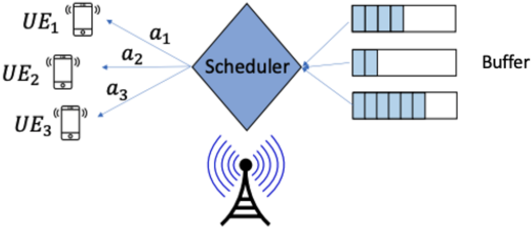
# Application: Autonomous Driving

- 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



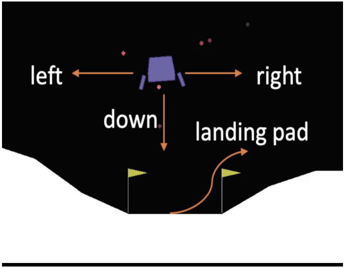
# Application: Wireless Communication

- 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: LunarLander 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 $\gamma \in (0, 1)$

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

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

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

# 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 $\gamma \in (0, 1)$

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

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

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

**RL Goal: find the best policy $\pi^*$**

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

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

Tutorial will not cover all the RL formulations

- Finite-time horizon, Average reward, Regret analysis

# Outline

# Formulation of 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:

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

  - Expected accumulated reward, start with $s$ follow $\pi$.

# Formulation of 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:

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

  - Expected accumulated reward, start with $s$ follow $\pi$.

**Policy Evaluation Problem:**

Given a fixed policy $\pi$, how to evaluate its state value function $V_\pi$?

- Foundation for policy optimization

# Summary of Policy Evaluation Approaches

- 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 Approaches

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

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

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

# 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**):

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

**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

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

  - $\mathrm{T}_\pi$ is contraction $\Rightarrow V_t \to V_\pi$.

# Model-Free: On-Policy TD Learning

## Model-Free

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

## On-Policy Data

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

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

**TD($\lambda$) 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 $V_\pi$ using parameterized model $V_\theta$
  - Linear model: $V_\theta(s) := \phi_s^\top \theta$, where $\phi_s$ is feature vector of $s$
  - Neural model: $V_\theta(s) := \mathrm{NN}_\theta(s)$, where $\mathrm{NN}_\theta$ is neural network

# Value Function Approximation

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

## TD(0) learning with function approximation

- Initialize model $\theta_0$.
- 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 $\ell_t$
- **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 $\mu$. There exist $\kappa > 0$, $\rho \in (0,1)$ such that

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

# Finite-Time Convergence of TD(0)

**Key Assumption: Geometric Mixing**

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

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

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

- Feature matrix $\Phi = [\phi_{s_1}^\top; \ldots; \phi_{s_n}^\top]$ full column rank, $V_\theta = \Phi\theta$
- Solution point $\theta^*$ 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 $\Phi = [\phi_{s_1}^\top; \dots; \phi_{s_n}^\top]$ full column rank, $V_\theta = \Phi\theta$
- Solution point $\theta^*$ 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 $\eta \le \mathcal{O}(\frac{1}{1-\gamma})$. After $T$ iterations,

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

where $\tau_{\text{mix}}(\eta) := \min\{t \mid \kappa\rho^t \le \eta\}$ 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 $\pi$

- 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 $\pi$

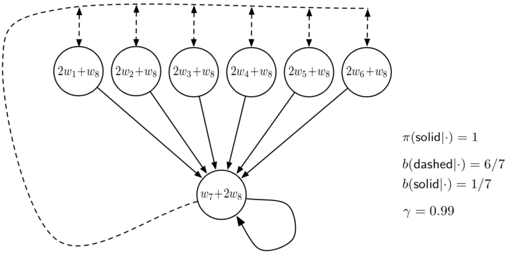
- Limitation: requires executing the target policy
- Limitation: in practice may not have sufficient on-policy data

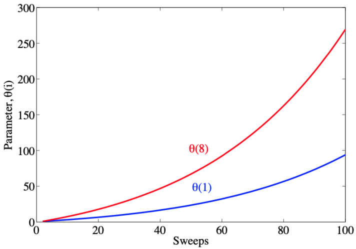
**Off-policy data**

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

# 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 $\theta^*$ satisfies

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

- Data sampled by behavior policy $\pi_b$, stationary distribution $\mu_b$

# Gradient TD for Off-Policy Evaluation

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

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

- Data sampled by behavior policy $\pi_b$, stationary distribution $\mu_b$

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

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

- Error $V_\theta(s) - \Pi_{\mathcal{L}} \mathsf{T}^\pi V_\theta(s)$ 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

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

- $\omega^*(\theta)$ 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

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

- $\omega^*(\theta)$ 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
- $\omega$ update is one-step SGD applied to LMS

# TDC Algorithm

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

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

**TDC algorithm [12]**

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

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

- $\theta$ update is different from GTD2
- $\omega$ 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

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

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

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

# Finite-Time Convergence of TDC

**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,

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

- Need small $\alpha, \beta$ and $\frac{\alpha}{\beta}$
- Small $\frac{\alpha}{\beta}$: $\omega_t$ takes faster update than $\theta_t$, because it needs to approximate the double expectation in $\theta$ 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/State-Action Value Function

- 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 $\pi^*$: 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 $\mathrm{P}$

## 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 $\mathrm{T}$
  - $\|\mathrm{T}V - V^*\|_\infty = \|\mathrm{T}V - \mathrm{T}V^*\|_\infty \leq \gamma\|V - V^*\|_\infty$
  - $\|\underbrace{\mathrm{T}\cdots\mathrm{T}}_{t \text{ times}} V - V^*\|_\infty \leq \gamma^t\|V - V^*\|_\infty \to 0$, as $t \to \infty$

# Policy Iteration

- Assume known reward $r$ and transition kernel P

### Policy Iteration

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

- **Policy improvement theorem**: Let $\pi$ and $\pi'$ be any pair of deterministic policies such that for all $s \in \mathcal{S}$, $Q_\pi(s, \pi'(s)) \geq V_\pi(s)$, then $\pi'$ is no worse than $\pi$: $V_{\pi'}(s) \geq V_\pi(s), \forall s \in \mathcal{S}$
- 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., $\epsilon$-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., $\epsilon$-greedy with $\epsilon = 1/t$)

# SARSA with Linear Function Approximation

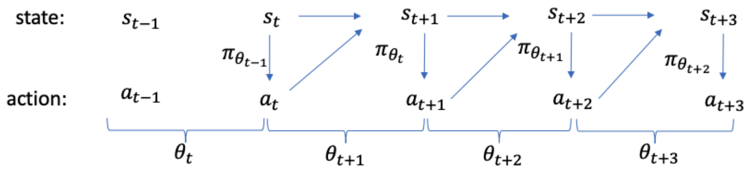
- Large $\mathcal{S}$ and $\mathcal{A}$, unknown $r$ and P

## SARSA

- Initialization: $\theta_0$, $s_0$, $\phi_i$, for $i = 1, 2, ..., N$
- $\pi_{\theta_0} \leftarrow \Gamma(\phi^\top \theta_0)$ (e.g., $\epsilon$-greedy, softmax w.r.t. $\phi^\top \theta_0$)
- Choose $a_0$ according to $\pi_{\theta_0}$
- For $t = 0, 1, 2, \ldots$
  - Observe $s_{t+1}$ and $r(s_t, a_t, s_{t+1})$, choose $a_{t+1}$ according to $\pi_{\theta_t}$
  - $\theta_{t+1} \leftarrow \theta_t + \alpha_t g_t(\theta_t)$
  - **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 $\ell(\theta) = \frac{1}{2}(\underbrace{r(s_t, a_t, s_{t+1}) + \gamma \phi^\top (s_{t+1}, a_{t+1})\theta_t}_{\text{target, one-step bootstrap}} - \phi^\top \theta)^2$
- $\Delta_t$ denotes the temporal difference error at time t: $\Delta_t = \text{target} - \phi^\top (s_t, a_t)\theta_t$,

# SARSA Sample Path




- As $\theta_t$ is updated, $\pi_{\theta_t}$ changes with time
- On-policy algorithm, time-varying policy
- Non-i.i.d. data

# Finite-Sample Analysis [19]

- The limit point $\theta^*$ 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 $\theta^*$ 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 $\theta^*$
- With constant step-size, SARSA converges exponentially fast to a small neighborhood of $\theta^*$

# 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 $\pi_b$, $t = 0$
- For $t = 0, 1, 2, ...$
  - Take action $a_t$ following fixed $\pi_b$, 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):

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

  - $\mu$: stationary distribution induced by behavior policy $\pi_b$
  - $\|Q(\cdot,\cdot)\|_\mu \triangleq \int_{s\in\mathcal{S},a\in\mathcal{A}} d\mu_{s,a} Q(s,a)$
  - $\Pi$: 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]:

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

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 $\pi^*$**

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

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

# 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

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

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

where $c > 0$ is a hyperparameter.

# TRPO/PPO Algorithm

## Trusted Region Policy Optimization (TRPO) [31]

- Update the parameter $w$ under KL constraint

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

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

where $c > 0$ is a hyperparameter.

## Proximal Policy Optimization (PPO) [32]

- Update the parameter $w$ via KL-regularized gradient ascent

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

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 $t$

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

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

# Natural Policy Gradient (NPG) 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 $\nu_{\pi_w}$

# 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 $\nu_{\pi_w}$

## 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 $(s_t,a_t)$, $K \sim \text{Geom}(1-\gamma)$
  - Estimate $\hat{Q}(s_t,a_t)$ by adding rewards over the sample path
  - Sample a length-$K$ trajectory starting at $(s_t)$, $K \sim \text{Geom}(1-\gamma)$
  - 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*

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

- PG converges to a neighborhood of a stationary point at a rate of $\mathcal{O}\left(\frac{1}{T}\right)$.
  - $\alpha$ controls a tradeoff between convergence rate and accuracy
  - Decreasing $\alpha$ 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 $\theta_t$
- 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* $\{1, \cdots, T\}$*.*

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

- 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

# 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 $\theta_t$
- 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* $\{1, \cdots, T\}$.

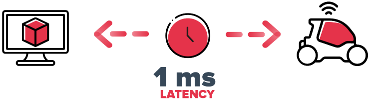
$$J(\pi^*) - \mathbb{E}\big[J(\pi_{w_{\hat{T}}})\big] \leq \mathcal{O}\left(\tfrac{1}{T}\right) + \mathcal{O}\left(\tfrac{1}{\sqrt{B}}\right) + (1 - \mathcal{O}(\lambda_{A_\pi}\beta))^{T_c/2} + \mathcal{O}\left(\tfrac{1}{\sqrt{M}}\right)$$
$$+ \mathcal{O}(\sqrt{\zeta^{\text{critic}}_{\text{approx}}})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

- 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

# Outline

# Topic 1: Safe 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 $R$:

$$V_R^\pi(\rho) := \mathbb{E}\left[\sum_{t=0}^\infty \gamma^t R(s_t, a_t, s_{t+1}) \middle| S_0 \sim \rho\right]$$

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

$$V_C^\pi(\rho) := \mathbb{E}\left[\sum_{t=0}^\infty \gamma^t C(s_t, a_t, s_{t+1}) \middle| S_0 \sim \rho\right]$$

**Goal of CMDP**

$$\max_\pi \quad V_R^\pi(\rho) \quad \text{subject to} \quad V_C^\pi(\rho) \leq c \tag{P}$$

# Primal-Dual Approach: e.g. CPO [46], PDO [47]

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

$$\mathcal{L}(\pi, \lambda) = V_R^\pi(\rho) + \lambda(V_C^\pi(\rho) - c).$$

- Dual function: $d_\lambda := \max_\pi \mathcal{L}(\pi, \lambda)$
  - $d_\lambda$ provides an upper bound on value of (P) for any $\lambda > 0$
- Dual problem:

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

- Duality gap: $\Delta = D^* - P^*$
  - Zero duality gap [44, 45]
  - (P) can be equivalently solved by solving (D)

# Primal-Dual Approach

## Primal-Dual Algorithm

- For $t = 0, 1, ..., T$
  - Compute $\pi_{t+1}$ based on $\mathcal{L}(\pi, \lambda_t)$ and $\pi_t$. Example methods:
    - Dual descent [45]: $\pi_{t+1} = \arg\max_\pi \mathcal{L}(\pi, \lambda_t)$ using some RL oracle
    - Natural policy gradient [48]: $\pi_{t+1} = \pi_t + \eta F_\rho(\pi_t)^\dagger \cdot \nabla_\pi \mathcal{L}(\pi_t, \lambda_t)$
  - Compute the dual ascent step $\lambda_{k+1} = (\lambda_k - \eta(V^{\pi_{t+1}}(\rho) - c))_+$.

# Primal-Dual Approach

## Primal-Dual Algorithm

- For $t = 0, 1, ..., T$
  - Compute $\pi_{t+1}$ based on $\mathcal{L}(\pi, \lambda_t)$ and $\pi_t$. Example methods:
    - Dual descent [45]: $\pi_{t+1} = \arg\max_\pi \mathcal{L}(\pi, \lambda_t)$ using some RL oracle
    - Natural policy gradient [48]: $\pi_{t+1} = \pi_t + \eta F_\rho(\pi_t)^\dagger \cdot \nabla_\pi \mathcal{L}(\pi_t, \lambda_t)$
  - Compute the dual ascent step $\lambda_{k+1} = (\lambda_k - \eta(V^{\pi_{t+1}}(\rho) - c))_+$.

- Performance metric:
  - Let $\pi^*$ denote the optimal solution to primal problem P
  - Optimality gap: $V_R^{\pi^*}(\rho) - V_R^{\pi}(\rho)$.
  - Constraint violation: $(V_C^{\pi}(\rho) - c)_+$.

# Primal-Dual Approach

## Primal-Dual Algorithm

- For $t = 0, 1, ..., T$
  - Compute $\pi_{t+1}$ based on $\mathcal{L}(\pi, \lambda_t)$ and $\pi_t$. Example methods:
    - Dual descent [45]: $\pi_{t+1} = \arg\max_\pi \mathcal{L}(\pi, \lambda_t)$ using some RL oracle
    - Natural policy gradient [48]: $\pi_{t+1} = \pi_t + \eta F_\rho(\pi_t)^\dagger \cdot \nabla_\pi \mathcal{L}(\pi_t, \lambda_t)$
  - Compute the dual ascent step $\lambda_{k+1} = (\lambda_k - \eta(V^{\pi_{t+1}}(\rho) - c))_+$.

- Performance metric:
  - Let $\pi^*$ denote the optimal solution to primal problem P
  - Optimality gap: $V_R^{\pi^*}(\rho) - V_R^{\pi}(\rho)$.
  - Constraint violation: $(V_C^{\pi}(\rho) - c)_+$.
- Convergence Rate:
  - Duality gap decays at a rate of $\mathcal{O}(1/\sqrt{T})$ [45]
  - Optimality gap decays $\mathcal{O}(1/\sqrt{T})$ and constraint violation decays $\mathcal{O}(1/T^{\frac{1}{4}})$ [48]
- Accelerated primal-dual algorithm: optimality gap and constraint violation decay $\mathcal{O}(1/T)$ [49]

# A Primal Approach: CRPO [50]

- No dual variable is needed, and easier to implement

**Constraint-Rectified Policy Optimization (CRPO)**

- For $t = 0, 1, ..., T - 1$
  - Constraint satisfaction: **If** $V_c^{\pi_t}(\rho) \leq c - \delta$: $\pi_{t+1} \leftarrow$ take one step natural policy gradient update towards minimize $V_C^{\pi_t}(\rho)$
  - Objective improvement: **Else** $\pi_{t+1} \leftarrow$ take one step natural policy gradient update towards maximize $V_R^{\pi_t}(\rho)$

# A Primal Approach: CRPO [50]

- No dual variable is needed, and easier to implement

**Constraint-Rectified Policy Optimization (CRPO)**

- For $t = 0, 1, ..., T - 1$
  - Constraint satisfaction: **If** $V_c^{\pi_t}(\rho) \leq c - \delta$: $\pi_{t+1} \leftarrow$ take one step natural policy gradient update towards minimize $V_C^{\pi_t}(\rho)$
  - Objective improvement: **Else** $\pi_{t+1} \leftarrow$ take one step natural policy gradient update towards maximize $V_R^{\pi_t}(\rho)$

- Optimize policy alternatively between objective improvement and constraint satisfaction
- Optimality gap and constraint violation decay $\mathcal{O}(1/\sqrt{T})$

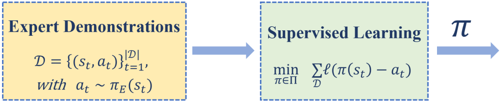
# Topic 2: Imitation Learning

- 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

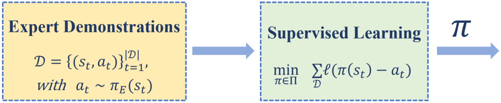
# Two Major Approaches on Imitation Learning

- Behavioral Cloning [51]
  - Directly learns a mapping from state to action based on supervised learning to match expert demonstrations

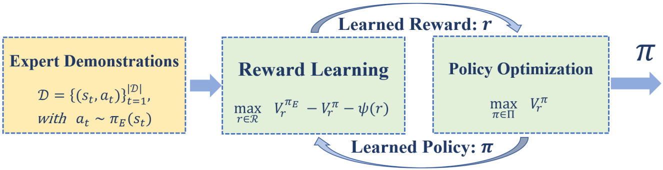


# Two Major Approaches on Imitation Learning

- Behavioral Cloning [51]
  - Directly learns a mapping from state to action based on supervised learning to match expert demonstrations




- Inverse Reinforcement Learning [52, 53]
  - 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 [54]



# Generative Adversarial Imitation Learning (GAIL)

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

## GAIL Framework [54]

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

- 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

# GAIL Policy Gradient Algorithm

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

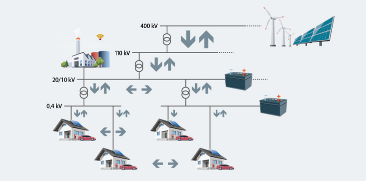
$$\widehat{\nabla}_\alpha F(w, \alpha) = \frac{1}{(1-\gamma)} \left[ \nabla_\alpha r_\alpha(s^E, a^E) - \nabla_\alpha r_\alpha(s^w, a^w) \right] - \nabla_\alpha \psi(\alpha)$$

  - Update $\alpha_{k+1} = \mathrm{Proj}\left( \alpha_k + \beta \widehat{\nabla}_\alpha F(w, \alpha_k) \right)$
- Policy update:
  - Use any policy gradient algorithm to update policy parameter $w$ for reward $r_\alpha(s, a)$

- Convergence rate with global optimality under various conditions [55, 56, 57]
- Convergence rate to stationary point [58]

# Topic 3: Multi-Agent Reinforcement Learning (MARL)

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



# Formulation of MARL

- State value function (of joint policy $\pi$):

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

- Average value function:

$$J(\pi) = (1-\gamma)\mathbb{E}\big[\sum_{t=0}^{\infty} \gamma^t \frac{1}{M} \sum_{m=1}^{M} r_t^{(m)}\big] = \mathbb{E}_\xi[V_\pi(s)]$$

**MARL Problem:**

$$\max_{\{\pi^{(m)}\}_m} J(\pi)$$

- Agents need synchronize info (local states, actions, rewards, etc)
- Tradeoff between communication & computation complexities

# Decentralized Policy Optimization for MARL

- Policy gradient with regard to agent $m$'s parameter $\omega^{(m)}$:

$$\nabla_{\omega^{(m)}} J(\omega_t) \approx \Big[\overline{R}_t + \gamma V(s'_{t+1}) - V(s_t)\Big] \psi_t^{(m)}(a_t^{(m)}|s_t). \qquad (1)$$

  - $V(s)$: learned via standard decentralized TD learning
  - $\psi_t^{(m)}(a_t^{(m)}|s_t)$: locally computed by the agent $m$
  - Challenge 1: need $\overline{R}_t$–average reward over all agents. Sensitive!
  - Challenge 2: How to achieve good communication & computation complexities at the same time?

Solution proposed by [59]:

- Corrupt local rewards using Gaussian with very large variance

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

- Estimate $\overline{R}$ via standard local averaging among all agents

$$\begin{aligned} \overline{R}_0 &= \widetilde{R}^{(m)}, \\ \overline{R}_{\ell+1} &= \textstyle\sum_{m' \in \mathcal{N}_m} W_{m,m'}\ \overline{R}_\ell, \quad \ell = 0, 1, \ldots, T' - 1. \end{aligned}$$

- Further use mini-batch updates to reduce the estimation error

$$\widehat{\nabla}_{\omega^{(m)}} J(\omega_t) = \frac{1}{N} \sum_{i=tN}^{(t+1)N-1} \Big[\overline{R}_i + \gamma V(s'_{i+1}) - V(s_i)\Big] \psi_t^{(m)}(a_i^{(m)} | s_i)$$

  - Can suppress noise with sufficiently large batch size $N$
  - Substantially reduces communication frequency and rounds
  - Helps achieve great sample/computation complexity

# Topic 4: Robust Reinforcement Learning

- Motivations:
  - Possible model deviation between training and test environments, e.g., training is on simulator
  - Adversarial attacks to MDPs
  - These could lead to severe performance degradation
- Robust Markov decision process (MDP): $(\mathcal{S}, \mathcal{A}, r, \mathsf{P})$, where $\mathsf{P} \in \mathcal{P}$, and $\mathcal{P}$ is an uncertainty set of transition kernels
- Robust value function: $\tilde{V}^{\pi}(s) = \inf_{\mathsf{P} \in \mathcal{P}} \mathbb{E}_{\mathsf{P}} \left[\sum_{t=0}^{\infty} \gamma^t r_t | S_0 = s, \pi\right]$
  - Worst-case performance
- Goal: Learn policy robust to model uncertainty

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

# Robust Reinforcement Learning

- **Model-Based Approach** [60, 61]
  - Assume knowledge of uncertainty set
  - Robust value function satisfies robust Bellman equation, which is a contraction
  - Robust value/policy iteration
- **Adversarial Training** [62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75]
  - Reformulate robust RL as a game between agent and nature, where nature chooses transition kernel P
  - Alternatively optimize agent's policy towards maximizing cumulative reward and nature's policy towards minimizing cumulative reward
  - Empirical success, but lack of theoretical robustness guarantee
- **Model-free Approach** [76, 77, 78]
  - Uncertainty set is centered at an unknown MDP from which samples can be taken
  - Online algorithms that can be updated efficiently

# $\varepsilon$-Contamination Model

- $\varepsilon$-contamination uncertainty set:

$$\mathcal{P}_s^a = \{(1-\varepsilon)p_s^a + \varepsilon q\}\,, \text{ for some } 0 \leq \varepsilon \leq 1$$

  With probability $1-\varepsilon$, state transition is perturbed using any arbitrary distribution $q$ over the state space $\mathcal{S}$

- $\varepsilon$-contamination can be related to total-variation/KL divergence defined uncertainty set via Pinsker's inequality

# Robust Q-learning [78]

**Initialization**: $T$, $\tilde{Q}_0(s,a)$ for all $(s,a)$, behavior policy $\pi_b$, $s_0$, step size $\alpha_t$

**For** $t = 0, 1, 2, ..., T-1$

Choose $a_t$ according to $\pi_b(\cdot|s_t)$

Observe $s_{t+1}$ and $r_t$

Update $\tilde{Q}_{t+1}$:

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

$$\tilde{Q}_{t+1}(s_t,a_t) \leftarrow (1-\alpha_t)\tilde{Q}_t(s_t,a_t) + \alpha_t(r_t + \gamma((1-\varepsilon)\tilde{V}_t(s_{t+1}) + \varepsilon \min_{s\in\mathcal{S}} \tilde{V}_t(s))$$

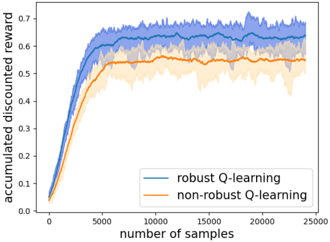
**Output**: $\tilde{Q}_T$

Performance guarantee:

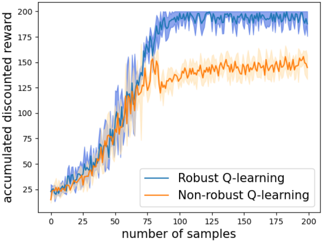
- Robust Q-learning converges to robust solution of $\max_\pi \tilde{V}^\pi$
- Same sample and computational complexity (within a constant factor) as vanilla Q-learning algorithm [78]
- Extension to function approximation also discussed in [78]

# 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

# 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
  - Agents need to jointly achieve a design goal
  - Decentralized algorithms under partial observations of environments
  - Challenges in design: delayed communication; communication depends on network topology
  - Open issues in theory: tradeoff among communications, computations, privacy

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

[45] 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.

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

[47] 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.

[48] 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.

[49] 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.

[50] 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.

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

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

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

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

[55] Q. Cai, M. Hong, Y. Chen, and Z. Wang, "On the global convergence of imitation learning: A case for linear quadratic regulator," *arXiv preprint arXiv:1901.03674*, 2019.

[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] M. Chen, Y. Wang, T. Liu, Z. Yang, X. Li, Z. Wang, and T. Zhao, "On computation and generalization of generative adversarial imitation learning," in *Proc. International Conference on Learning Representations (ICLR)*, 2020.

[59] Anonymous, "Sample and communication-efficient decentralized actor-critic algorithms with finite-time analysis," in *Submitted to The Tenth International Conference on Learning Representations*, 2022.

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

[61] A. Nilim and L. El Ghaoui, "Robustness in Markov decision problems with uncertain transition matrices," in *Proc. Advances in Neural Information Processing Systems (NIPS)*, pp. 839–846, 2004.

[62] 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.

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

[64] 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.

[65] 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.

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

[67] 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.

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

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

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

[71] 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.

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

[73] 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.

[74] 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.

[75] 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.

[76] 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.

[77] 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, 2021.

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