Policy Mirror Descent Inherently Explores Action Space

Yan Li

Georgia Institute of Technology

ISyE student seminar, 2023

Joint work with George Lan

Markov decision process

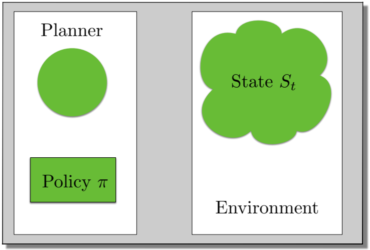
- **Sequential decision making over multiple timesteps ..**

Markov decision process

▷ **Sequential decision making over multiple timesteps ..**

Key elements

- policy π
- state space: $\mathcal{S}$
- action space: $\mathcal{A}$
- cost function c
- transition kernel $\mathbb{P}$

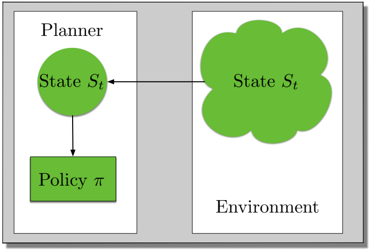


Markov decision process

▷ **Sequential decision making over multiple timesteps ..**

Key elements

- policy π
- state space: $\mathcal{S}$
- action space: $\mathcal{A}$
- cost function c
- transition kernel $\mathbb{P}$

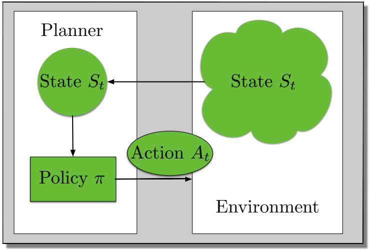


Markov decision process

▷ **Sequential decision making over multiple timesteps ..**

Key elements

- policy π
- state space: $\mathcal{S}$
- action space: $\mathcal{A}$
- cost function c
- transition kernel $\mathbb{P}$



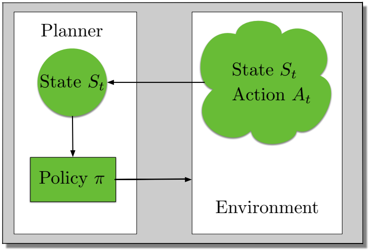
Decision making: A_t follows distribution $\pi(\cdot|S_t)$

Markov decision process

▷ **Sequential decision making over multiple timesteps ..**

Key elements

- policy π
- state space: $\mathcal{S}$
- action space: $\mathcal{A}$
- cost function c
- transition kernel $\mathbb{P}$



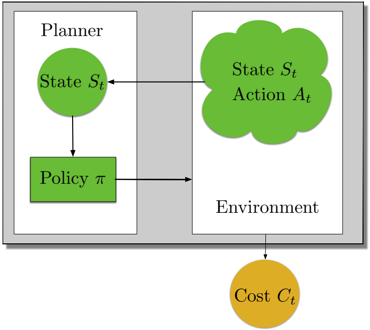
Decision making: A_t follows distribution $\pi(\cdot|S_t)$

Markov decision process

▷ **Sequential decision making over multiple timesteps ..**

Key elements

- policy π
- state space: $\mathcal{S}$
- action space: $\mathcal{A}$
- cost function c
- transition kernel $\mathbb{P}$



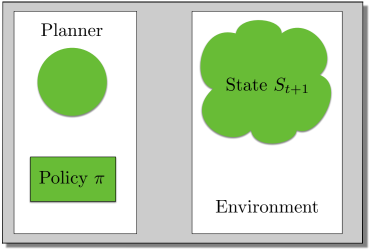
Observing loss: $C_t = c(S_t, A_t) \in [0,1]$

Markov decision process

▷ **Sequential decision making over multiple timesteps ..**

Key elements

- policy π
- state space: $\mathcal{S}$
- action space: $\mathcal{A}$
- cost function c
- transition kernel $\mathbb{P}$



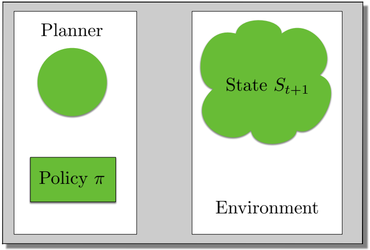
State transition: S_{t+1} follows distribution $\mathbb{P}(\cdot|S_t, A_t)$

Markov decision process

▷ **Sequential decision making over multiple timesteps ..**

Key elements

- policy π
- state space: $\mathcal{S}$
- action space: $\mathcal{A}$
- cost function c
- transition kernel $\mathbb{P}$



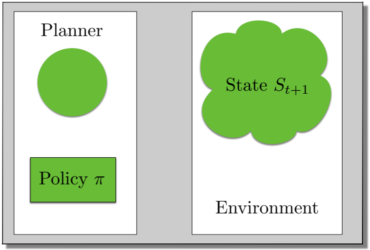
Repeat decision process ..

Markov decision process

▷ **Sequential decision making over multiple timesteps ..**

Key elements

- policy π
- state space: $\mathcal{S}$
- action space: $\mathcal{A}$
- cost function c
- transition kernel $\mathbb{P}$



Trajectory:

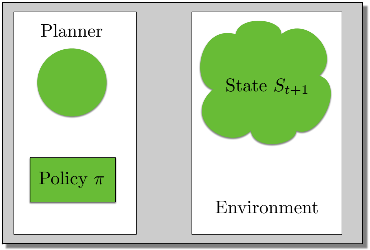
$$\{(S_0, A_0, C_0), (S_1, A_1, C_1), \ldots, (S_t, A_t, C_t), \ldots\}$$

Markov decision process

▷ **Sequential decision making over multiple timesteps ..**

Key elements

- policy π
- state space: $\mathcal{S}$
- action space: $\mathcal{A}$
- cost function c
- transition kernel $\mathbb{P}$



Performance (value function):

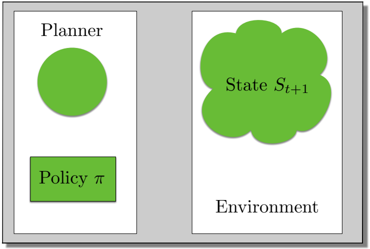
$$V^\pi(s) = \mathbb{E}^\pi\left[\sum_{t=0}^{\infty} \underbrace{\gamma^t C_t}_{\text{discounting future}} \,\middle|\, S_0 = s\right]$$

Markov decision process

▷ **Sequential decision making over multiple timesteps ..**

Key elements

- policy π
- state space: $\mathcal{S}$
- action space: $\mathcal{A}$
- cost function c
- transition kernel $\mathbb{P}$



Planning: finding the optimal policy

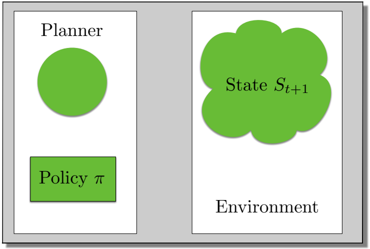
$$\min_{\pi} V^{\pi}(s) \ \forall s \in \mathcal{S}$$

Markov decision process

▷ **Sequential decision making over multiple timesteps ..**

Key elements

- policy π
- state space: $\mathcal{S}$
- action space: $\mathcal{A}$
- cost function c
- transition kernel $\mathbb{P}$



Planning: finding the optimal policy

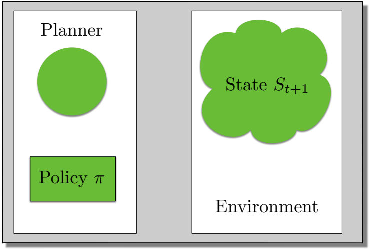
$$\min_{\pi} f_{\nu}(\pi) = \sum_{s \in \mathcal{S}} \nu(s) V^{\pi}(s)$$

Markov decision process

▷ **Sequential decision making over multiple timesteps ..**

Key elements

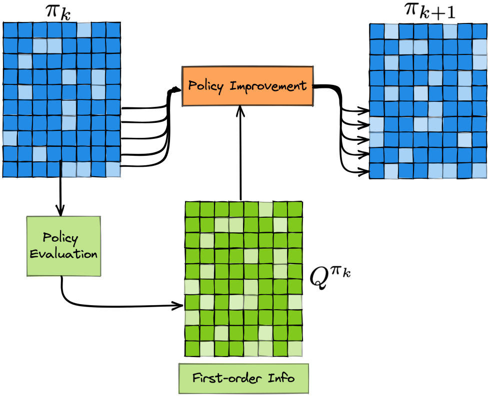
- policy π
- state space: $\mathcal{S}$
- action space: $\mathcal{A}$
- cost function c
- transition kernel $\mathbb{P}$



Planning: finding the optimal policy

$$\min_{\pi} f_{\nu}(\pi) = \sum_{s \in \mathcal{S}} \nu(s) V^{\pi}(s) \quad \Rightarrow \quad \text{Non-convex!}$$

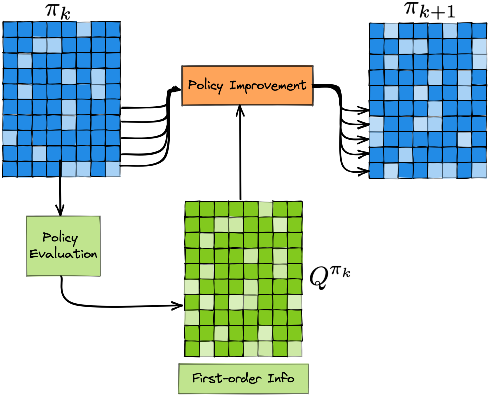
Policy Evaluation and Improvement



First-order policy optimization:

1. $\text{Eval}(\pi_k) \to Q^{\pi_k}$
2. Construct gradient information G_k
3. $\text{Update}(\pi_k, G_k) \to \pi_{k+1}$
4. Repeat ..

Policy Evaluation and Improvement

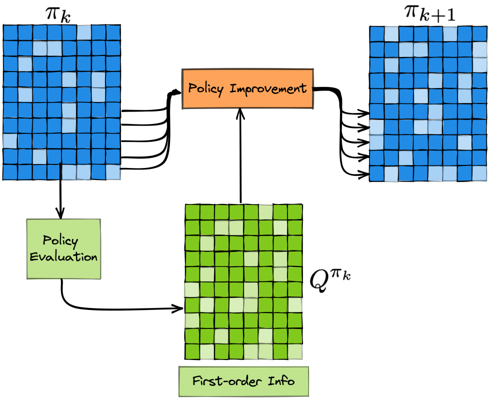


Q-function:

$$Q^{\pi}(s,a) = \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t c(S_t, A_t) \middle| S_0 = s, A_0 = a\right]$$

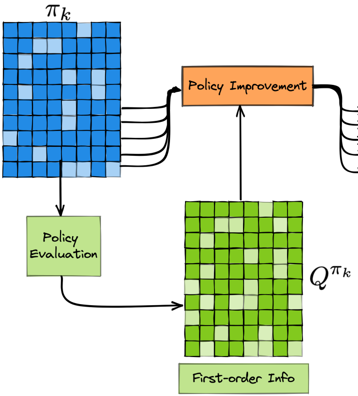
Bellman's equation: Q^{π} solves a linear system involving the transition $\mathbb{P}$

Policy Evaluation and Improvement



⋆ **Challenge:** $\mathbb{P}$ is unknown!

Policy Evaluation and Improvement



⋆ **Current status of policy gradients:**

An ϵ-optimal policy can be attained using $\mathcal{O}(1/\epsilon^2)$ samples, **IF ...**

"The BIG IF"

Tension between evaluation and optimization

Classical Policy Evaluation

Q-function:

$$Q^{\pi}(s,a) = \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \gamma^{t} c(S_t, A_t) \middle| S_0 = s, A_0 = a\right]$$

Classical Policy Evaluation

Q-function:

$$Q^{\pi}(s,a) = \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t c(S_t, A_t) \middle| S_0 = s, A_0 = a\right]$$

Classical Policy Evaluation

Q-function:

$$Q^{\pi}(s,a) = \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \gamma^{t} c(S_t, A_t) \middle| S_0 = s, A_0 = a\right]$$

$\text{C-Eval}(\pi_k)$ with unknown $\mathbb{P}$:

Classical Policy Evaluation

Q-function:

$$Q^\pi(s,a) = \mathbb{E}^\pi\left[\sum_{t=0}^\infty \gamma^t c(S_t, A_t) \middle| S_0 = s, A_0 = a\right]$$

$\text{C-Eval}(\pi_k)$ **with unknown** $\mathbb{P}$**:**

1. Deploy π_k, generate trajectory:

$$\xi = \{(S_0, A_0, C_0), (S_1, A_1, C_1), \ldots, (S_t, A_t, C_t), \ldots\}$$

Classical Policy Evaluation

Q-function:

$$Q^\pi(s,a) = \mathbb{E}^\pi\left[\sum_{t=0}^{\infty} \gamma^t c(S_t, A_t) \middle| S_0 = s, A_0 = a\right]$$

$\text{C-Eval}(\pi_k)$ **with unknown** $\mathbb{P}$**:**

1. Deploy π_k, generate trajectory:

$$\xi = \{(S_0, A_0, C_0), (S_1, A_1, C_1), \ldots, (S_t, A_t, C_t), \ldots\}$$

2. Apply learning procedure that makes clever use of the trajectories

Classical Policy Evaluation

Q-function:

$$Q^\pi(s,a) = \mathbb{E}^\pi\left[\sum_{t=0}^{\infty} \gamma^t c(S_t, A_t) \middle| S_0 = s, A_0 = a\right]$$

$\text{C-Eval}(\pi_k)$ with unknown $\mathbb{P}$:

1. Deploy π_k, generate trajectory:

$$\xi = \{(S_0, A_0, C_0), (S_1, A_1, C_1), \ldots, (S_t, A_t, C_t), \ldots\}$$

2. Apply learning procedure that makes clever use of the trajectories
   - On-policy Monte-Carlo

Classical Policy Evaluation

Q-function:

$$Q^\pi(s,a) = \mathbb{E}^\pi \left[\sum_{t=0}^{\infty} \gamma^t c(S_t, A_t) \middle| S_0 = s, A_0 = a\right]$$

$\text{C-Eval}(\pi_k)$ **with unknown $\mathbb{P}$:**

1. Deploy π_k, generate trajectory:

$$\xi = \{(S_0, A_0, C_0), (S_1, A_1, C_1), \ldots, (S_t, A_t, C_t), \ldots\}$$

2. Apply learning procedure that makes clever use of the trajectories
    - On-policy Monte-Carlo
    - On-policy temporal-difference (TD)

Classical Policy Evaluation

Fundamental assumption:

$$\pi_k(a|s) > 0, \ \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$$

Classical Policy Evaluation

Fundamental assumption:

$$\pi_k(a|s) > 0, \ \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$$

Recall trajectory:

$$\xi = \{(S_0, A_0, C_0), (S_1, A_1, C_1), \ldots, (S_t, A_t, C_t), \ldots\}$$

where $A_t \sim \pi_k(\cdot|S_t)$

Classical Policy Evaluation

Fundamental assumption:

$$\pi_k(a|s) > 0,\ \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$$

Recall trajectory:

$$\xi = \{(S_0, A_0, C_0), (S_1, A_1, C_1), \ldots, (S_t, A_t, C_t), \ldots\}$$

where $A_t \sim \pi_k(\cdot|S_t)$

- ⋆ Observation: action with zero probability never gets explored

If $\pi_k(a|s) = 0 \quad \Rightarrow \quad (s, a)$ does not appear in ξ

Classical Policy Evaluation

Fundamental assumption:

$$\pi_k(a|s) > 0, \ \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$$

Recall trajectory:

$$\xi = \{(S_0, A_0, C_0), (S_1, A_1, C_1), \ldots, (S_t, A_t, C_t), \ldots\}$$

where $A_t \sim \pi_k(\cdot|S_t)$

⋆ Observation: action with zero probability never gets explored

If $\pi_k(a|s) = 0 \quad \Rightarrow \quad Q^{\pi_k}(s, a)$ not learnable

Perhaps not a big deal? Lets make some hopefully benign assumptions:

$$\underline{\sigma} := \inf_{k \geq 0} \underbrace{\inf_{s \in \mathcal{S}, a \in \mathcal{A}} \pi_k(a|s)}_{\underline{\sigma}_k} > 0$$

Classical Policy Evaluation

Fundamental assumption:

$$\pi_k(a|s) > 0, \ \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$$

Recall trajectory:

$$\xi = \{(S_0, A_0, C_0), (S_1, A_1, C_1), \ldots, (S_t, A_t, C_t), \ldots\}$$

where $A_t \sim \pi_k(\cdot|S_t)$

- ⋆ Observation: action with zero probability never gets explored

If $\pi_k(a|s) = 0 \quad \Rightarrow \quad Q^{\pi_k}(s, a)$ not learnable

Perhaps not a big deal? Lets make some hopefully benign assumptions:

$$\underline{\sigma} := \inf_{k \geq 0} \underbrace{\inf_{s \in \mathcal{S}, a \in \mathcal{A}} \pi_k(a|s)}_{\underline{\sigma}_k} > 0$$

Suppose $\{S_t\}$ visits every state (ergodic) and

$\underline{\sigma} > 0 \ \Rightarrow$ Great, we are done! (most prior works)

Classical Policy Evaluation

Perhaps not a big deal? Lets make some hopefully benign assumptions:

$$\underline{\sigma} := \inf_{k \geq 0} \underbrace{\inf_{s \in \mathcal{S}, a \in \mathcal{A}} \pi_k(a|s)}_{\underline{\sigma}_k} > 0$$

Classical Policy Evaluation

Perhaps not a big deal? Lets make some hopefully benign assumptions:

$$\underline{\sigma} := \inf_{k \geq 0} \underbrace{\inf_{s \in \mathcal{S}, a \in \mathcal{A}} \pi_k(a|s)}_{\underline{\sigma}_k} > 0$$

Unfortunately – not benign at all

Classical Policy Evaluation

Perhaps not a big deal? Lets make some hopefully benign assumptions:

"A BIG IF" $$\underline{\sigma} := \inf_{k \geq 0} \underbrace{\inf_{s \in \mathcal{S}, a \in \mathcal{A}} \pi_k(a|s)}_{\underline{\sigma}_k} > 0$$

Unfortunately – not benign at all

Classical Policy Evaluation

Perhaps not a big deal? Lets make some hopefully benign assumptions:

"A BIG IF" $$\underline{\sigma} := \inf_{k\geq 0} \underbrace{\inf_{s\in\mathcal{S}, a\in\mathcal{A}} \pi_k(a|s)}_{\underline{\sigma}_k} > 0$$

Unfortunately – not benign at all

Purpose of planning: structure of optimal policies

Classical Policy Evaluation

Perhaps not a big deal? Lets make some hopefully benign assumptions:

"A BIG IF" $$\underline{\sigma} := \inf_{k \geq 0} \underbrace{\inf_{s \in \mathcal{S}, a \in \mathcal{A}} \pi_k(a|s)}_{\underline{\sigma}_k} > 0$$

Unfortunately – not benign at all

Purpose of planning: structure of optimal policies

Optimal Q-function: $Q^*(s, a) = \min_{\pi} Q^{\pi}(s, a).$

Classical Policy Evaluation

Perhaps not a big deal? Lets make some hopefully benign assumptions:

"A BIG IF" $$\underline{\sigma} := \inf_{k\geq 0} \underbrace{\inf_{s\in\mathcal{S}, a\in\mathcal{A}} \pi_k(a|s)}_{\underline{\sigma}_k} > 0$$

Unfortunately – not benign at all

Purpose of planning: structure of optimal policies

$$\mathcal{A}^*(s) := \operatorname*{Argmin}_{a\in\mathcal{A}} Q^*(s,a) \Rightarrow \boxed{\pi^*(a|s) = 0 \text{ if } a \notin \mathcal{A}^*(s)}$$

Classical Policy Evaluation

Perhaps not a big deal? Lets make some hopefully benign assumptions:

"A BIG IF" $$\underline{\sigma} := \inf_{k \geq 0} \underbrace{\inf_{s \in \mathcal{S}, a \in \mathcal{A}} \pi_k(a|s)}_{\underline{\sigma}_k} > 0$$

Unfortunately – not benign at all

Purpose of planning: structure of optimal policies

$$\mathcal{A}^*(s) := \operatorname*{Argmin}_{a \in \mathcal{A}} Q^*(s,a) \Rightarrow \boxed{\pi^*(a|s) = 0 \text{ if } a \notin \mathcal{A}^*(s)}$$

Tension between policy optimization and evaluation:

$$\text{If } \pi_k \approx \pi^* \Rightarrow \underbrace{\boxed{\underline{\sigma}_k \approx 0}}_{\pi_k \text{ becomes deterministic}} \Rightarrow \boxed{\text{C-Eval}(\pi_k) \text{ fails}} \Rightarrow \boxed{\text{bad } \pi_{k+1}}$$

Preview of our development

Theorem (Li and Lan, '23 – Informal)

An ϵ-optimal policy can be attained by policy gradient methods using $\mathcal{O}(1/\epsilon^2)$ samples, ~~**IF ...**~~

Preview of our development

Theorem (Li and Lan, '23 – Informal)

An ϵ-optimal policy can be attained by policy gradient methods using $\mathcal{O}(1/\epsilon^2)$ samples, ~~IF~~ ...

Some key ingredients:

Preview of our development

Theorem (Li and Lan, '23 – Informal)

An ϵ-optimal policy can be attained by policy gradient methods using $\mathcal{O}(1/\epsilon^2)$ samples, ~~IF~~ ...

Some key ingredients:

1. A 2-year-old dog for policy improvement: stochastic policy mirror descent (Lan, '21)

Preview of our development

Theorem (Li and Lan, '23 – Informal)

An ϵ-optimal policy can be attained by policy gradient methods using $\mathcal{O}(1/\epsilon^2)$ samples, ~~IF~~ ...

Some key ingredients:

1. A 2-year-old dog for policy improvement: stochastic policy mirror descent (Lan, '21)
2. A few new tricks: novel evaluation procedures

Preview of our development

Theorem (Li and Lan, '23 – Informal)

An ϵ-optimal policy can be attained by policy gradient methods using $\mathcal{O}(1/\epsilon^2)$ samples, ~~IF~~ ...

Some key ingredients:

1. A 2-year-old dog for policy improvement: stochastic policy mirror descent (Lan, '21)
2. A few new tricks: novel evaluation procedures
3. Analysis:

Prior development – optimization and evaluation are independent

Preview of our development

Theorem (Li and Lan, '23 – Informal)

An ϵ-optimal policy can be attained by policy gradient methods using $\mathcal{O}(1/\epsilon^2)$ samples, ~~IF ...~~

Some key ingredients:

1. A 2-year-old: stochastic policy mirror descent (Lan, '21)
2. New tricks: novel evaluation procedures
3. Analysis:

Our perspective – jointly consider optimization and evaluation

Stochastic Policy Mirror Descent

* suppose everything is still ok

Policy Improvement

Algorithm SPMD update: $\pi_k \to \pi_{k+1}$

Input: Estimated $\widehat{Q}^{\pi_k}$ from $\mathrm{Eval}(\pi_k)$

Update: For every state $s \in \mathcal{S}$:

$$\pi_{k+1} = \mathrm{argmin}_{p \in \Delta_{\mathcal{A}}} \eta_k \langle \widehat{Q}^{\pi_k}(s, \cdot), p \rangle + \mathcal{D}^p_{\pi_k}(s)$$

Policy Improvement

Algorithm SPMD update: $\pi_k \to \pi_{k+1}$

Input: Estimated $\widehat{Q}^{\pi_k}$ from $\mathrm{Eval}(\pi_k)$

Update: For every state $s \in \mathcal{S}$:

$$\pi_{k+1} = \mathrm{argmin}_{p \in \Delta_{\mathcal{A}}} \eta_k \langle \widehat{Q}^{\pi_k}(s, \cdot), p \rangle + \mathcal{D}^p_{\pi_k}(s)$$

- η_k – stepsize

Policy Improvement

Algorithm SPMD update: $\pi_k \to \pi_{k+1}$

Input: Estimated $\widehat{Q}^{\pi_k}$ from $\mathrm{Eval}(\pi_k)$

Update: For every state $s \in \mathcal{S}$:

$$\pi_{k+1} = \mathrm{argmin}_{p \in \Delta_{\mathcal{A}}} \eta_k \langle \widehat{Q}^{\pi_k}(s, \cdot), p \rangle + \mathcal{D}^p_{\pi_k}(s)$$

- η_k – stepsize
- $\mathcal{D}^p_{\pi_k}(s)$ – Bregman divergence

Policy Improvement

Algorithm SPMD update: $\pi_k \to \pi_{k+1}$

Input: Estimated $\widehat{Q}^{\pi_k}$ from $\mathrm{Eval}(\pi_k)$

Update: For every state $s \in \mathcal{S}$:

$$\pi_{k+1} = \mathrm{argmin}_{p \in \Delta_{\mathcal{A}}} \eta_k \langle \widehat{Q}^{\pi_k}(s, \cdot), p \rangle + \mathcal{D}^p_{\pi_k}(s)$$

- η_k – stepsize
- $\mathcal{D}^p_{\pi_k}(s)$ – Bregman divergence
  1. Projected gradient: $\mathcal{D}^p_{\pi_k}(s) = \|p - \pi_k(\cdot|s)\|_2^2$

Policy Improvement

Algorithm SPMD update: $\pi_k \to \pi_{k+1}$

Input: Estimated $\widehat{Q}^{\pi_k}$ from $\mathrm{Eval}(\pi_k)$

Update: For every state $s \in \mathcal{S}$:

$$\pi_{k+1} = \mathrm{argmin}_{p \in \Delta_{\mathcal{A}}} \eta_k \langle \widehat{Q}^{\pi_k}(s, \cdot), p \rangle + \mathcal{D}^p_{\pi_k}(s)$$

- η_k – stepsize
- $\mathcal{D}^p_{\pi_k}(s)$ – Bregman divergence
  1. Projected gradient: $\mathcal{D}^p_{\pi_k}(s) = \|p - \pi_k(\cdot|s)\|_2^2$
  2. Natural policy gradient: $\mathcal{D}^p_{\pi_k}(s) = \mathrm{KL}(p \| \pi_k(\cdot|s))$:

$$\pi_{k+1}(a|s) \propto \pi_k(a|s) \exp\left(-\eta_k \widehat{Q}^{\pi_k}(s, a)\right)$$

SPMD with Classical Evaluation

Theorem (Lan, '21 – Informal)

1. *Choose a trajectory of length* $\mathcal{O}(\log(1/\epsilon))$ *at each iteration*

SPMD with Classical Evaluation

Theorem (Lan, '21 – Informal)

1. *Choose a trajectory of length* $\mathcal{O}(\log(1/\epsilon))$ *at each iteration*
2. *Apply TD-type evaluation method (CTD, Kotsalis et al., '20)*

SPMD with Classical Evaluation

Theorem (Lan, '21 – Informal)

1. *Choose a trajectory of length* $\mathcal{O}(\log(1/\epsilon))$ *at each iteration*
2. *Apply TD-type evaluation method (CTD, Kotsalis et al., '20)*
3. *Set proper stepsize*

SPMD with Classical Evaluation

Theorem (Lan, '21 – Informal)

1. *Choose a trajectory of length $\mathcal{O}(\log(1/\epsilon))$ at each iteration*
2. *Apply TD-type evaluation method (CTD, Kotsalis et al., '20)*
3. *Set proper stepsize*

Then SPMD returns an ϵ-optimal policy in $\mathcal{O}(1/\epsilon^2)$ iterations

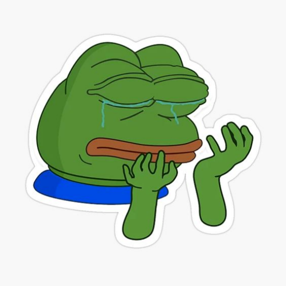
SPMD with Classical Evaluation

Theorem (Lan, '21 – Informal)

1. *Choose a trajectory of length $\mathcal{O}(\log(1/\epsilon))$ at each iteration*
2. *Apply TD-type evaluation method (CTD, Kotsalis et al., '20)*
3. *Set proper stepsize*

Then SPMD returns an ϵ-optimal policy in $\mathcal{O}(1/\epsilon^2)$ iterations

- Requires the "BIG IF": $\underline{\sigma} > 0$



SPMD with New Evaluation Operators

* facing the reality

New Evaluation Procedures

Some prior development:

- Explicit exploration: force policy to explore every action

New Evaluation Procedures

Some prior development:

- Explicit exploration: force policy to explore every action
  - mix with uniform distribution (a.k.a. ϵ-exploration): $\mathcal{O}(1/\epsilon^6)$ (Khodadadian et al., '21)

New Evaluation Procedures

Some prior development:

- Explicit exploration: force policy to explore every action
  - mix with uniform distribution (a.k.a. ϵ-exploration): $\mathcal{O}(1/\epsilon^6)$ (Khodadadian et al., '21)
  - policy perturbation within evaluation (Li et al., '22): $\mathcal{O}(1/\epsilon^2)$

New Evaluation Procedures

Some prior development:

- Explicit exploration: force policy to explore every action
  - mix with uniform distribution (a.k.a. ϵ-exploration): $\mathcal{O}(1/\epsilon^6)$ (Khodadadian et al., '21)
  - policy perturbation within evaluation (Li et al., '22): $\mathcal{O}(1/\epsilon^2)$
- No exploration:

New Evaluation Procedures

Some prior development:

- Explicit exploration: force policy to explore every action
  - mix with uniform distribution (a.k.a. ϵ-exploration): $\mathcal{O}(1/\epsilon^6)$ (Khodadadian et al., '21)
  - policy perturbation within evaluation (Li et al., '22): $\mathcal{O}(1/\epsilon^2)$
- No exploration:
  - weighted policy evaluation (Hu et al., '22): $\mathcal{O}(1/\epsilon^{16})$

New Evaluation Procedures

What can be improved?

- Explicit exploration: force policy to explore every action
  - can be efficient

New Evaluation Procedures

What can be improved?

- Explicit exploration: force policy to explore every action
  - can be efficient
  - need to modify the policy within evaluation

New Evaluation Procedures

What can be improved?

- Explicit exploration: force policy to explore every action
  - can be efficient
  - need to modify the policy within evaluation
  - repeatedly taking high-risk actions

New Evaluation Procedures

What can be improved?

- Explicit exploration: force policy to explore every action
  - can be efficient
  - need to modify the policy within evaluation
  - repeatedly taking high-risk actions
- No exploration:
  - simple, but inefficient

New Evaluation Procedures

What can be improved?

- Explicit exploration: force policy to explore every action
  - can be efficient
  - need to modify the policy within evaluation
  - repeatedly taking high-risk actions
- No exploration:
  - simple, but inefficient

Can we be efficient, and avoid pitfalls above?

New Evaluation Procedures

Algorithm Truncated Monte-Carlo: $\pi_k \to \widehat{Q}^{\pi_k}$

Generate a trajectory of length n

$$\{(S_0, A_0, C_0), (S_1, A_1, C_1), \ldots, (S_{n-1}, A_{n-1}, C_{n-1})\}$$

for every state-action pair (s, a) **do**

$$t(s,a) = \begin{cases} \text{first timestep hitting } (s,a) \text{ before } n \\ n, \text{otherwise} \end{cases}$$

$$\widehat{Q}^{\pi_k}(s,a) = \sum_{t=t(s,a)}^{n-1} \gamma^t C_t$$

if $\pi_k(a|s) \leq \tau$:

$$\widehat{Q}^{\pi_k}(s,a) = \frac{1}{1-\gamma} \qquad \text{[Truncation step]}$$

end for

SPMD with TOMC

Theorem (Li and Lan, '23 – Informal)

1. *Choose* $n = \mathcal{O}(\log(1/\epsilon))$ *at each iteration*

SPMD with TOMC

Theorem (Li and Lan, '23 – Informal)

1. *Choose* $n = \mathcal{O}(\log(1/\epsilon))$ *at each iteration*
2. *Apply TOMC for evaluation with proper* $\tau > 0$

SPMD with TOMC

Theorem (Li and Lan, '23 – Informal)

1. *Choose* $n = \mathcal{O}(\log(1/\epsilon))$ *at each iteration*
2. *Apply TOMC for evaluation with proper* $\tau > 0$
3. *Set proper stepsize*

SPMD with TOMC

Theorem (Li and Lan, '23 – Informal)

1. *Choose* $n = \mathcal{O}(\log(1/\epsilon))$ *at each iteration*
2. *Apply TOMC for evaluation with proper* $\tau > 0$
3. *Set proper stepsize*

Then SPMD returns an ϵ*-optimal policy in* $\mathcal{O}(\mathcal{M}/\epsilon^2)$ *iterations.*

SPMD with TOMC

Theorem (Li and Lan, '23 – Informal)

1. *Choose* $n = \mathcal{O}(\log(1/\epsilon))$ *at each iteration*
2. *Apply TOMC for evaluation with proper* $\tau > 0$
3. *Set proper stepsize*

Then SPMD returns an ϵ*-optimal policy in* $\mathcal{O}(\mathcal{M}/\epsilon^2)$ *iterations.*

Some remarks:

1. $\mathcal{M}$ depend on the divergence $\mathcal{D}^p_{\pi_k}(s)$ in SPMD
   - KL divergence: exponential on $\frac{1}{1-\gamma}$ (effective horizon)
   - Tsallis divergence: polynomial

SPMD with TOMC

Theorem (Li and Lan, '23 – Informal)

1. *Choose* $n = \mathcal{O}(\log(1/\epsilon))$ *at each iteration*
2. *Apply TOMC for evaluation with proper* $\tau > 0$
3. *Set proper stepsize*

Then SPMD returns an ϵ*-optimal policy in* $\mathcal{O}(\mathcal{M}/\epsilon^2)$ *iterations.*

Some remarks:

1. $\mathcal{M}$ depend on the divergence $\mathcal{D}^p_{\pi_k}(s)$ in SPMD
    - KL divergence: exponential on $\frac{1}{1-\gamma}$ (effective horizon)
    - Tsallis divergence: polynomial
2. No changes to the policy, no explicit exploration

SPMD with TOMC

Theorem (Li and Lan, '23 – Informal)

1. *Choose* $n = \mathcal{O}(\log(1/\epsilon))$ *at each iteration*
2. *Apply TOMC for evaluation with proper* $\tau > 0$
3. *Set proper stepsize*

Then SPMD returns an ϵ*-optimal policy in* $\mathcal{O}(\mathcal{M}/\epsilon^2)$ *iterations.*

Some remarks:

1. $\mathcal{M}$ depend on the divergence $\mathcal{D}^p_{\pi_k}(s)$ in SPMD
   - KL divergence: exponential on $\frac{1}{1-\gamma}$ (effective horizon)
   - Tsallis divergence: polynomial
2. No changes to the policy, no explicit exploration
3. Has certain "memory"

$$\pi_k(a|s) \leq \tau \ \Rightarrow \pi_{k+1}(a|s) < \pi_k(a|s) \leq \tau$$

How does SPMD + TOMC work – conceptually?

Let us revisit the tension

If $\pi_k \approx \pi^*$ $\Rightarrow$ $\underbrace{\boxed{\underline{\sigma}_k \approx 0}}_{\pi_k \text{ becomes deterministic}}$ $\Rightarrow$ $\boxed{\text{C-Eval}(\pi_k) \text{ fails}}$ $\Rightarrow$ $\boxed{\text{bad } \pi_{k+1}}$

How does SPMD + TOMC work – conceptually?

Let us revisit the tension

$$\text{If } \pi_k \approx \pi^* \Rightarrow \underbrace{\boxed{\sigma_k \approx 0}}_{\pi_k \text{ becomes deterministic}} \Rightarrow \boxed{\text{C-Eval}(\pi_k) \text{ fails}} \Rightarrow \boxed{\text{bad } \pi_{k+1}}$$

Wait ..

$$\text{If } \pi_k \approx \pi^* \Rightarrow \underbrace{\boxed{\text{We should use this!}}}_{\text{TOMC handles this}} \Rightarrow \boxed{\pi_{k+1} \approx \pi^*}$$

How does SPMD + TOMC work – conceptually?

Let us revisit the tension

$$\text{If } \pi_k \approx \pi^* \Rightarrow \underbrace{\boxed{\sigma_k \approx 0}}_{\pi_k \text{ becomes deterministic}} \Rightarrow \boxed{\text{C-Eval}(\pi_k) \text{ fails}} \Rightarrow \boxed{\text{bad } \pi_{k+1}}$$

Wait ..

$$\text{If } \pi_k \approx \pi^* \Rightarrow \underbrace{\boxed{\text{We should use this!}}}_{\text{TOMC handles this}} \Rightarrow \boxed{\pi_{k+1} \approx \pi^*}$$

Technical challenge

Difficult to detect $\pi_k \approx \pi^*$ (we do not know π^*)

How does SPMD + TOMC work – technically?

Imagine an oracle $\mathbb{O}$

- At every iteration:
  - $\pi_k(a|s) < \tau \xrightarrow{\mathbb{O}} a \notin \mathcal{A}^*(s)$ (i.e., a is non-optimal)

How does SPMD + TOMC work – technically?

Imagine an oracle $\mathbb{O}$

- At every iteration:
  - $\pi_k(a|s) < \tau \xrightarrow{\mathbb{O}} a \notin \mathcal{A}^*(s)$ (i.e., a is non-optimal)

Some observations

How does SPMD + TOMC work – technically?

Imagine an oracle $\mathbb{O}$

- At every iteration:
  - $\pi_k(a|s) < \tau \xrightarrow{\mathbb{O}} a \notin \mathcal{A}^*(s)$ (i.e., a is non-optimal)

Some observations

- $\mathbb{O}$ does not exist (even) for policy iteration (with exact Q^{π_k})

How does SPMD + TOMC work – technically?

Imagine an oracle $\mathbb{O}$

- At every iteration:
  - $\pi_k(a|s) < \tau \xrightarrow{\mathbb{O}} a \notin \mathcal{A}^*(s)$ (i.e., a is non-optimal)

Some observations

- $\mathbb{O}$ does not exist (even) for policy iteration (with exact Q^{π_k})
- $\mathbb{O}$ exists for ~~S~~PMD (with exact Q^{π_k})

How does SPMD + TOMC work – technically?

Imagine an oracle $\mathbb{O}$

- At every iteration:
    - $\pi_k(a|s) < \tau \xrightarrow{\mathbb{O}} a \notin \mathcal{A}^*(s)$ (i.e., a is non-optimal)

Some observations

- $\mathbb{O}$ does not exist (even) for policy iteration (with exact Q^{π_k})
- $\mathbb{O}$ exists for ~~S~~PMD (with exact Q^{π_k})
- If $\mathbb{O}$ exists for SPMD, we consider two cases

How does SPMD + TOMC work – technically?

Imagine an oracle $\mathbb{O}$

- At every iteration:
  - $\pi_k(a|s) < \tau \xrightarrow{\mathbb{O}} a \notin \mathcal{A}^*(s)$ (i.e., a is non-optimal)

Some observations

- $\mathbb{O}$ does not exist (even) for policy iteration (with exact Q^{π_k})
- $\mathbb{O}$ exists for ~~S~~PMD (with exact Q^{π_k})
- If $\mathbb{O}$ exists for SPMD, we consider two cases
  1. If $\pi_k(a|s) \leq \tau$, then $a \notin \mathcal{A}^*(s)$, SPMD + TOMC makes sure

$$\pi_{k+1}(a|s) < \pi_k(a|s)$$

How does SPMD + TOMC work – technically?

Imagine an oracle $\mathbb{O}$

- At every iteration:
  - $\pi_k(a|s) < \tau \xrightarrow{\mathbb{O}} a \notin \mathcal{A}^*(s)$ (i.e., a is non-optimal)

Some observations

- $\mathbb{O}$ does not exist (even) for policy iteration (with exact Q^{π_k})
- $\mathbb{O}$ exists for ~~S~~PMD (with exact Q^{π_k})
- If $\mathbb{O}$ exists for SPMD, we consider two cases
  1. If $\pi_k(a|s) \le \tau$, then $a \notin \mathcal{A}^*(s)$, SPMD + TOMC makes sure
  $$\pi_{k+1}(a|s) < \pi_k(a|s)$$
  2. If $\pi_k(a|s) > \tau$, then a is explored by π_k, and
  $Q^{\pi_k}(s,a)$ can be learned well

How does SPMD + TOMC work – technically?

Imagine an oracle $\mathbb{O}$

- At every iteration:
  - $\pi_k(a|s) < \tau \xrightarrow{\mathbb{O}} a \notin \mathcal{A}^*(s)$ (i.e., a is non-optimal)

Some observations

- $\mathbb{O}$ does not exist (even) for policy iteration (with exact Q^{π_k})
- $\mathbb{O}$ exists for ~~S~~PMD (with exact Q^{π_k})
- If $\mathbb{O}$ exists for SPMD, we consider two cases
  1. If $\pi_k(a|s) \leq \tau$, then $a \notin \mathcal{A}^*(s)$, SPMD + TOMC makes sure
     $$\pi_{k+1}(a|s) < \pi_k(a|s)$$
  2. If $\pi_k(a|s) > \tau$, then a is explored by π_k, and
     $Q^{\pi_k}(s,a)$ can be learned well
  3. **Power of $\mathbb{O}$**:

> "We can learn every action that still matters"

Putting our Thoughts Together

Constructing the oracle $\mathbb{O}$

- $\mathbb{O}$ should be robust in the presence of noise

Putting our Thoughts Together

Constructing the oracle $\mathbb{O}$

- $\mathbb{O}$ should be robust in the presence of noise
- $\mathbb{O}$ should be (approximately) correct at every iteration

Putting our Thoughts Together

Constructing the oracle $\mathbb{O}$

- $\mathbb{O}$ should be robust in the presence of noise
- $\mathbb{O}$ should be (approximately) correct at every iteration
  1. If using inductive argument, $\mathcal{O}(1/\epsilon^4)$ samples

Putting our Thoughts Together

Constructing the oracle $\mathbb{O}$

- $\mathbb{O}$ should be robust in the presence of noise
- $\mathbb{O}$ should be (approximately) correct at every iteration
  1. If using inductive argument, $\mathcal{O}(1/\epsilon^4)$ samples
  2. A more refined probabilistic argument, $\mathcal{O}(1/\epsilon^2)$ samples

Putting our Thoughts Together

Constructing the oracle $\mathbb{O}$

- $\mathbb{O}$ should be robust in the presence of noise
- $\mathbb{O}$ should be (approximately) correct at every iteration
  1. If using inductive argument, $\mathcal{O}(1/\epsilon^4)$ samples
  2. A more refined probabilistic argument, $\mathcal{O}(1/\epsilon^2)$ samples
- Construction $\mathbb{O}$ requires interaction of optimization and evaluation

Putting our Thoughts Together

Constructing the oracle $\mathbb{O}$

- $\mathbb{O}$ should be robust in the presence of noise
- $\mathbb{O}$ should be (approximately) correct at every iteration
  1. If using inductive argument, $\mathcal{O}(1/\epsilon^4)$ samples
  2. A more refined probabilistic argument, $\mathcal{O}(1/\epsilon^2)$ samples
- Construction $\mathbb{O}$ requires interaction of optimization and evaluation
  1. Recall PI does not have such an $\mathbb{O}$

Putting our Thoughts Together

Constructing the oracle $\mathbb{O}$

- $\mathbb{O}$ should be robust in the presence of noise
- $\mathbb{O}$ should be (approximately) correct at every iteration
  1. If using inductive argument, $\mathcal{O}(1/\epsilon^4)$ samples
  2. A more refined probabilistic argument, $\mathcal{O}(1/\epsilon^2)$ samples
- Construction $\mathbb{O}$ requires interaction of optimization and evaluation
  1. Recall PI does not have such an $\mathbb{O}$
- More details in the paper

Putting our Thoughts Together

Constructing the oracle $\mathbb{O}$

- $\mathbb{O}$ should be robust in the presence of noise
- $\mathbb{O}$ should be (approximately) correct at every iteration
  1. If using inductive argument, $\mathcal{O}(1/\epsilon^4)$ samples
  2. A more refined probabilistic argument, $\mathcal{O}(1/\epsilon^2)$ samples
- Construction $\mathbb{O}$ requires interaction of optimization and evaluation
  1. Recall PI does not have such an $\mathbb{O}$
- More details in the paper
  - ⋆ An alternative evaluation procedure

Putting our Thoughts Together

Constructing the oracle $\mathbb{O}$

- $\mathbb{O}$ should be robust in the presence of noise
- $\mathbb{O}$ should be (approximately) correct at every iteration
  1. If using inductive argument, $\mathcal{O}(1/\epsilon^4)$ samples
  2. A more refined probabilistic argument, $\mathcal{O}(1/\epsilon^2)$ samples
- Construction $\mathbb{O}$ requires interaction of optimization and evaluation
  1. Recall PI does not have such an $\mathbb{O}$
- More details in the paper
  - ⋆ An alternative evaluation procedure

Presentation based on Preprint

- Li, Y., & Lan, G. (2023). Policy Mirror Descent Inherently Explores Action Space. arXiv preprint arXiv:2303.04386.