Navigating to the Best Policy in Markov Decision Processes

Aymen Al Marjani[1], Aurélien Garivier[1], Alexandre Proutiere[2]

October 18, 2021

[1]ENS de Lyon

[2]KTH Royal Institute of Technology

Outline

Introduction

Main Results

Novelties in Algorithm design

Conclusion

Introduction

Motivation

- Pac-Man needs to learn an optimal policy that maximizes his long-term reward.
- Pac-Man doesn't have access to a simulator.
- So Pac-Man needs to navigate through the unknown maze to collect observations.

Infinite horizon discounted MDPs

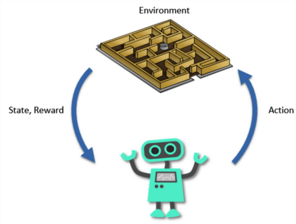
1. $\mathcal{S}, \mathcal{A}$: **Finite** state and action spaces.

$$\mathcal{M} =< \mathcal{S}, \mathcal{A}, p_{\mathcal{M}}, q_{\mathcal{M}}, \gamma >$$

Infinite horizon discounted MDPs

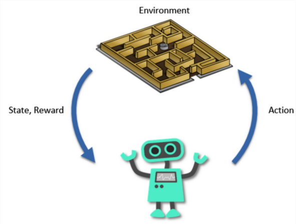
$$\mathcal{M} =< \mathcal{S}, \mathcal{A}, p_{\mathcal{M}}, q_{\mathcal{M}}, \gamma >$$

1. $\mathcal{S}, \mathcal{A}$: **Finite** state and action spaces.
2. After playing action a at state s the agent:
   - receives reward $R(s, a) \sim q_{\mathcal{M}}(.|s, a)$.
   - makes transition to $s' \sim p_{\mathcal{M}}(.|s, a)$.



Infinite horizon discounted MDPs

$$\mathcal{M} =< \mathcal{S}, \mathcal{A}, p_{\mathcal{M}}, q_{\mathcal{M}}, \gamma >$$



1. $\mathcal{S}, \mathcal{A}$: **Finite** state and action spaces.
2. After playing action a at state s the agent:
   - receives reward $R(s, a) \sim q_{\mathcal{M}}(.|s, a)$.
   - makes transition to $s' \sim p_{\mathcal{M}}(.|s, a)$.
   - For simplicity, we assume q with support in $[0, 1]$.

Best Policy Identification

$\mathcal{M} =< \mathcal{S}, \mathcal{A}, p_{\mathcal{M}}, q_{\mathcal{M}}, \gamma >$

- $\gamma \in [0, 1)$ is the discount factor.

Best Policy Identification

$\mathcal{M} =< \mathcal{S}, \mathcal{A}, p_{\mathcal{M}}, q_{\mathcal{M}}, \gamma >$

- $\gamma \in [0, 1)$ is the discount factor.
- Collect observations to identify a policy $\pi : \mathcal{S} \to \mathcal{A}$ maximizing the total discounted reward:

$$\pi^\star_{\mathcal{M}} \in \arg\max_{\pi} \; V^\pi_{\mathcal{M}}(s) = \mathbb{E}_{\mathcal{M}}\left[\sum_{t=0}^{\infty} \gamma^t R(s^\pi_t, \pi(s^\pi_t)) \middle| \; s_0 = s \right]$$

Best Policy Identification

$\mathcal{M} =< \mathcal{S}, \mathcal{A}, p_{\mathcal{M}}, q_{\mathcal{M}}, \gamma >$

- $\gamma \in [0, 1)$ is the discount factor.
- Collect observations to identify a policy $\pi : \mathcal{S} \to \mathcal{A}$ maximizing the total discounted reward:

$$\pi^\star_{\mathcal{M}} \in \arg\max_{\pi} \; V^{\pi}_{\mathcal{M}}(s) = \mathbb{E}_{\mathcal{M}}\left[\sum_{t=0}^{\infty} \gamma^t R(s^{\pi}_t, \pi(s^{\pi}_t)) \middle| \, s_0 = s \right]$$

- We assume that: (1) $\pi^\star \triangleq \pi^\star_{\mathcal{M}}$ is unique; (2) $\mathcal{M}$ is communicating.

Best Policy Identification

$\mathcal{M} =< \mathcal{S}, \mathcal{A}, p_{\mathcal{M}}, q_{\mathcal{M}}, \gamma >$

- $\gamma \in [0, 1)$ is the discount factor.
- Collect observations to identify a policy $\pi : \mathcal{S} \to \mathcal{A}$ maximizing the total discounted reward:

$$\pi^\star_{\mathcal{M}} \in \arg\max_{\pi} \; V^{\pi}_{\mathcal{M}}(s) = \mathbb{E}_{\mathcal{M}}\left[\sum_{t=0}^{\infty} \gamma^t R(s_t^\pi, \pi(s_t^\pi)) \middle| \; s_0 = s \right]$$

- We assume that: (1) $\pi^\star \triangleq \pi^\star_{\mathcal{M}}$ is unique; (2) $\mathcal{M}$ is communicating.
- δ-**PC algorithm:** Return $\widehat{\pi}^\star_\tau$ such that $\mathbb{P}_{\mathcal{M}}(\widehat{\pi}^\star_\tau \neq \pi^\star) \leq \delta$, using minimum number of samples!

Learning: be specific!

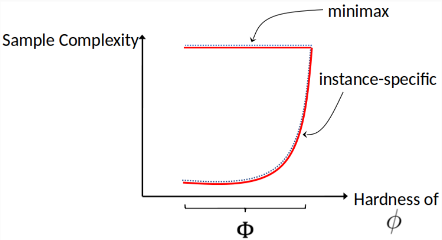
Measures of optimality:

- **Minimax** over a class of MDPs $\mathbb{M}$:

$$\inf_{\mathbb{A}:\delta\text{-PC}} \sup_{\mathcal{M}\in\mathbb{M}} \mathbb{E}_{\mathcal{M},\mathbb{A}}[\tau_\delta]$$

- **Instance-specific:** For ground-truth instance $\mathcal{M}$:

$$\inf_{\mathbb{A}:\delta\text{-PC}} \mathbb{E}_{\mathcal{M},\mathbb{A}}[\tau_\delta]$$



Learning: be specific!

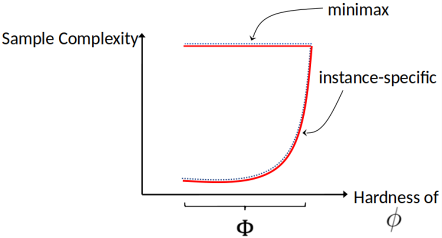
Measures of optimality:

- **Minimax** over a class of MDPs $\mathbb{M}$:

$$\inf_{\mathbb{A}:\delta\text{-PC}} \sup_{\mathcal{M}\in\mathbb{M}} \mathbb{E}_{\mathcal{M},\mathbb{A}}[\tau_\delta]$$

- **Instance-specific:** For ground-truth instance $\mathcal{M}$:

$$\inf_{\mathbb{A}:\delta\text{-PC}} \mathbb{E}_{\mathcal{M},\mathbb{A}}[\tau_\delta]$$



- We seek algorithms that can adapt to the hardness of the instance.

Main Results

Lower Bound: A two-player zero-sum game

- 1st Player (Algorithm) plays a sampling strategy $\omega \in \Sigma$ in the simplex of $\mathbb{R}^{SA}$.

Lower Bound: A two-player zero-sum game

- 1st Player (Algorithm) plays a sampling strategy $\omega \in \Sigma$ in the simplex of $\mathbb{R}^{SA}$.
- 2nd Player (Nature) plays an alternative instance $\mathcal{M}' \in \mathrm{Alt}(\mathcal{M}) = \{\mathcal{M}' : \ \pi^\star \text{ is not optimal in } \mathcal{M}'\}$.

Lower Bound: A two-player zero-sum game

- 1st Player (Algorithm) plays a sampling strategy $\omega \in \Sigma$ in the simplex of $\mathbb{R}^{SA}$.
- 2nd Player (Nature) plays an alternative instance $\mathcal{M}' \in \mathrm{Alt}(\mathcal{M}) = \{\mathcal{M}' : \ \pi^\star \text{ is not optimal in } \mathcal{M}'\}$.
- The gain of Algorithm (= loss of Nature) is:

$$\sum_{s,a} \omega_{sa} \mathrm{KL}_{\mathcal{M}|\mathcal{M}'}(s,a),$$

where $\mathrm{KL}_{\mathcal{M}|\psi}(s,a) = \mathrm{KL}(q_{\mathcal{M}}(s,a),\ q_\psi(s,a)) + \mathrm{KL}(p_{\mathcal{M}}(s,a),\ p_\psi(s,a))$.

Lower Bound: A two-player zero-sum game

- 1st Player (Algorithm) plays a sampling strategy $\omega \in \Sigma$ in the simplex of $\mathbb{R}^{SA}$.
- 2nd Player (Nature) plays an alternative instance $\mathcal{M}' \in \mathsf{Alt}(\mathcal{M}) = \{\mathcal{M}' : \ \pi^\star \text{ is not optimal in } \mathcal{M}'\}$.
- The gain of Algorithm (= loss of Nature) is:

$$\sum_{s,a} \omega_{sa} \mathrm{KL}_{\mathcal{M}|\mathcal{M}'}(s,a),$$

Algorithm only collects observations from a single trajectory $\implies \omega$ is constrained:

$$\omega \in \Omega(\mathcal{M}) = \left\{\omega \in \Sigma : \ \forall s \in \mathcal{S}, \ \sum_{a} \omega_{sa} = \sum_{s',a'} p_{\mathcal{M}}(s|s',a')\omega_{s'a'}\right\}.$$

Lower Bound: A two-player zero-sum game

- 1st Player (Algorithm) plays a sampling strategy $\omega \in \Sigma$ in the simplex of $\mathbb{R}^{SA}$.
- 2nd Player (Nature) plays an alternative instance $\mathcal{M}' \in \mathrm{Alt}(\mathcal{M}) = \{\mathcal{M}' :\ \pi^\star \text{ is not optimal in } \mathcal{M}'\}$.
- The gain of Algorithm (= loss of Nature) is:

$$\sum_{s,a} \omega_{sa} \mathrm{KL}_{\mathcal{M}|\mathcal{M}'}(s,a),$$

Algorithm only collects observations from a single trajectory $\implies$ ω is constrained:

$$\omega \in \Omega(\mathcal{M}) = \left\{ \omega \in \Sigma :\ \forall s \in \mathcal{S},\ \sum_{a} \omega_{sa} = \sum_{s',a'} p_{\mathcal{M}}(s|s',a')\omega_{s'a'} \right\}.$$

Value of the game:

$$T_o(\mathcal{M})^{-1} = \sup_{\omega \in \Omega(\mathcal{M})} \inf_{\mathcal{M}' \in \mathrm{Alt}(\mathcal{M})} \sum_{s,a} \omega_{sa} \mathrm{KL}_{\mathcal{M}|\mathcal{M}'}(s,a).$$

Lower Bound: A two-player zero-sum game

Proposition 1

The sample complexity of any δ-PC algorithm satisfies: for any $\mathcal{M}$ with a unique optimal policy,

$$\liminf_{\delta \to 0} \frac{\mathbb{E}_{\mathcal{M},\mathbb{A}}[\tau]}{\log(1/\delta)} \geq T_o(\mathcal{M}),$$

$$\text{where } T_o(\mathcal{M})^{-1} = \sup_{\omega \in \Omega(\mathcal{M})} \inf_{\mathcal{M}' \in \mathrm{Alt}(\mathcal{M})} \sum_{s,a} \omega_{sa} \mathrm{KL}_{\mathcal{M}|\mathcal{M}'}(s,a). \tag{1}$$

where:

- $\mathrm{Alt}(\mathcal{M}) = \{\mathcal{M}' : \ \pi^\star \text{ is not optimal in } \mathcal{M}'\}$ = alternative MDPs.
- $\Omega(\mathcal{M}) = \left\{\omega \in \Sigma : \ \forall s \in \mathcal{S}, \ \sum_a \omega_{sa} = \sum_{s',a'} p_{\mathcal{M}}(s|s',a')\omega_{s'a'}\right\}$ the set of navigation-constrained allocations.
- $\mathrm{KL}_{\mathcal{M}|\psi}(s,a) = \mathrm{KL}(q_{\mathcal{M}}(s,a),\ q_\psi(s,a)) + \mathrm{KL}(p_{\mathcal{M}}(s,a),\ p_\psi(s,a))$

Upper Bound

We propose an algorithm, MDP-Navigate-and-Stop (MDP-NaS).

Theorem 1

MDP-NaS is δ-Probably Correct. If the algorithm has access to an optimization oracle that solves the minimization sub-problem of the LB, then

$$\mathbb{E}[\tau_\delta] = \mathcal{O}(T_o(\mathcal{M}))\log(1/\delta).$$

Upper Bound

We propose an algorithm, MDP-Navigate-and-Stop (MDP-NaS).

Theorem 1

MDP-NaS is δ-Probably Correct. If the algorithm has access to an optimization oracle that solves the minimization sub-problem of the LB, then

$$\mathbb{E}[\tau_\delta] = \mathcal{O}(T_o(\mathcal{M}))\log(1/\delta).$$

Otherwise, it's sample complexity is bounded by:

$$\mathcal{O}\left(\inf_{\omega\in\Omega(\mathcal{M})} \max_{(s,a):a\neq\pi^\star(s)} \frac{1+\mathrm{Var}_{p(s,a)}[V^\star_{\mathcal{M}}]}{\omega_{sa}\Delta_{sa}^2} + \frac{1}{\min_s \omega_{s,\pi^\star(s)}\Delta_{\min}^2(1-\gamma)^3}\right)\log(1/\delta)$$

- $\Delta_{sa} = V^\star_{\mathcal{M}}(s) - Q^\star_{\mathcal{M}}(s,a)$: sub-optimality gap.
- $\Delta_{\min} = \min_{s,a\neq\pi^\star(s)} \Delta_{sa}$: minimum gap.
- $\mathrm{Var}_{p(s,a)}[V^\star_{\mathcal{M}}] = \mathbb{V}_{s'\sim p_{\mathcal{M}}(.|s,a)}[V^\star_{\mathcal{M}}(s')]$: variance of next-state value function.

Take-away message

1. Instance specific Lower Bound: Two-player game, similar to the case of generative model but with a restricted strategy set for the algorithm.

2. First Algorithm with problem-specific guarantees in the online setting !

3. Algorithm can be instance-optimal given an optimization oracle that solves the best response problem.

Novelties in Algorithm design

Recipe for instance-optimality in Pure Exploration

- Suppose we know how to compute the optimal allocation vector

$$\boldsymbol{\omega}^\star(\mathcal{M}) = \operatorname*{arg\,max}_{\omega \in \Omega(\mathcal{M})} \inf_{\mathcal{M}' \in \mathrm{Alt}(\mathcal{M})} \sum_{s,a} \omega_{sa} \mathrm{KL}_{\mathcal{M}|\mathcal{M}'}(s, a).$$

Recipe for instance-optimality in Pure Exploration

- Suppose we know how to compute the optimal allocation vector

$$\boldsymbol{\omega}^\star(\mathcal{M}) = \operatorname*{arg\,max}_{\omega \in \Omega(\mathcal{M})} \inf_{\mathcal{M}' \in \mathrm{Alt}(\mathcal{M})} \sum_{s,a} \omega_{sa} \mathrm{KL}_{\mathcal{M}|\mathcal{M}'}(s,a).$$

- All we need is a sampling rule which ensures that for all (s,a):

$$\boxed{N_{sa}(t)/t \underset{t\to\infty}{\longrightarrow} \boldsymbol{\omega}^\star_{sa}(\mathcal{M}).}$$

Recipe for instance-optimality in Pure Exploration

- Suppose we know how to compute the optimal allocation vector

$$\boldsymbol{\omega}^\star(\mathcal{M}) = \operatorname*{arg\,max}_{\omega \in \Omega(\mathcal{M})} \inf_{\mathcal{M}' \in \mathrm{Alt}(\mathcal{M})} \sum_{s,a} \omega_{sa} \mathrm{KL}_{\mathcal{M}|\mathcal{M}'}(s,a).$$

- All we need is a sampling rule which ensures that for all (s,a):

$$\boxed{N_{sa}(t)/t \underset{t\to\infty}{\longrightarrow} \boldsymbol{\omega}^\star_{sa}(\mathcal{M}).}$$

- Multi-armed bandits and MDPs with a generative model (simulator):

    1. Forced exploration: sample (s_{t+1}, a_{t+1}) from $\{(s,a):\ N_{sa}(t) \leq \sqrt{t}\}$ if not empty.

Recipe for instance-optimality in Pure Exploration

- Suppose we know how to compute the optimal allocation vector

$$\boldsymbol{\omega}^\star(\mathcal{M}) = \underset{\omega \in \Omega(\mathcal{M})}{\arg\max} \inf_{\mathcal{M}' \in \mathrm{Alt}(\mathcal{M})} \sum_{s,a} \omega_{sa} \mathrm{KL}_{\mathcal{M}|\mathcal{M}'}(s,a).$$

- All we need is a sampling rule which ensures that for all (s, a):

$$\boxed{N_{sa}(t)/t \underset{t \to \infty}{\longrightarrow} \boldsymbol{\omega}^\star_{sa}(\mathcal{M}).}$$

- Multi-armed bandits and MDPs with a generative model (simulator):

    1. Forced exploration: sample (s_{t+1}, a_{t+1}) from $\{(s,a): \ N_{sa}(t) \le \sqrt{t}\}$ if not empty.
    2. Tracking: Otherwise sample $(s_{t+1}, a_{t+1}) \in \arg\min_{s,a} N_{sa}(t)/t - \boldsymbol{\omega}^\star_{sa}(\widehat{\mathcal{M}}_t)$.

Recipe for instance-optimality in Pure Exploration

- Suppose we know how to compute the optimal allocation vector

$$\boldsymbol{\omega}^\star(\mathcal{M}) = \operatorname*{arg\,max}_{\omega \in \Omega(\mathcal{M})} \inf_{\mathcal{M}' \in \mathrm{Alt}(\mathcal{M})} \sum_{s,a} \omega_{sa} \mathrm{KL}_{\mathcal{M}|\mathcal{M}'}(s,a).$$

- All we need is a sampling rule which ensures that for all (s,a):

$$\boxed{N_{sa}(t)/t \underset{t\to\infty}{\longrightarrow} \boldsymbol{\omega}^\star_{sa}(\mathcal{M}).}$$

- Multi-armed bandits and MDPs with a generative model (simulator):

1. Forced exploration: sample (s_{t+1}, a_{t+1}) from $\{(s,a):\ N_{sa}(t) \leq \sqrt{t}\}$ if not empty. Ensures consistency of $\boldsymbol{\omega}^\star(\widehat{\mathcal{M}}_t)$!
2. Tracking: Otherwise sample $(s_{t+1}, a_{t+1}) \in \arg\min_{s,a} N_{sa}(t)/t - \boldsymbol{\omega}^\star_{sa}(\widehat{\mathcal{M}}_t)$. Ensures efficient sampling strategy on the long run.

D-Navigation: A novel sampling rule

- We can't choose the next state the online setting ! $\implies$ Tracking is no longer applicable.

D-Navigation: A novel sampling rule

- We can't choose the next state the online setting ! $\implies$ Tracking is no longer applicable.
- Define oracle policy:

$$\forall (s,a) \in \mathcal{S} \times \mathcal{A}, \quad \pi^o(\mathcal{M})(a|s) \triangleq \frac{\omega^\star_{sa}(\mathcal{M})}{\sum_{b\in\mathcal{A}} \omega^\star_{sb}(\mathcal{M})}.$$

$\boldsymbol{\omega}^\star$ is the stationary distribution of the Markov Chain P_{π^o}.

D-Navigation: A novel sampling rule

- We can't choose the next state the online setting ! $\implies$ Tracking is no longer applicable.
- Define oracle policy:

$$\forall (s,a) \in \mathcal{S} \times \mathcal{A}, \quad \pi^{o}(\mathcal{M})(a|s) \triangleq \frac{\omega^{\star}_{sa}(\mathcal{M})}{\sum_{b\in\mathcal{A}} \omega^{\star}_{sb}(\mathcal{M})}.$$

  $\boldsymbol{\omega}^{\star}$ is the stationary distribution of the Markov Chain $P_{\pi^{o}}$.

- Play $a_t \sim \pi_t(.|s_t)$ where:

$$\pi_t = \varepsilon_t \pi_u + (1-\varepsilon_t)\pi^{o}(\widehat{\mathcal{M}}_t).$$

D-Navigation: A novel sampling rule

- We can't choose the next state the online setting ! $\implies$ Tracking is no longer applicable.
- Define oracle policy:

$$\forall (s,a) \in \mathcal{S} \times \mathcal{A}, \quad \pi^o(\mathcal{M})(a|s) \triangleq \frac{\omega^\star_{sa}(\mathcal{M})}{\sum_{b\in\mathcal{A}} \omega^\star_{sb}(\mathcal{M})}.$$

  $\boldsymbol{\omega}^\star$ is the stationary distribution of the Markov Chain P_{π^o}.
- Play $a_t \sim \pi_t(.|s_t)$ where:

$$\pi_t = \varepsilon_t \pi_u + (1-\varepsilon_t)\pi^o(\widehat{\mathcal{M}}_t)$$

  (π_u is the uniform policy)

D-Navigation: A novel sampling rule

- We can't choose the next state the online setting ! $\implies$ Tracking is no longer applicable.
- Define oracle policy:

$$\forall (s,a) \in \mathcal{S} \times \mathcal{A}, \quad \pi^o(\mathcal{M})(a|s) \triangleq \frac{\omega^\star_{sa}(\mathcal{M})}{\sum_{b \in \mathcal{A}} \omega^\star_{sb}(\mathcal{M})}.$$

  $\boldsymbol{\omega}^\star$ is the stationary distribution of the Markov Chain P_{π^o}.

- Play $a_t \sim \pi_t(.|s_t)$ where:

$$\pi_t = \varepsilon_t \pi_u + (1 - \varepsilon_t)\pi^o(\widehat{\mathcal{M}}_t).$$

  ε_t is an exploration parameter that needs careful tuning (check the paper !)

D-Navigation: Why does it work ?

- $(s_0, a_0, \dots, s_t, a_t, \dots) = (s_t, a_t)_{t \geq 0}$ is the realization of Non-homogeneous (and history dependent !) Markov Chain with values in $\mathcal{S} \times \mathcal{A}$ with kernels $(P_{\pi_t})_{t \geq 1}$.
- Denote by $(\omega_t)_{t \geq 1}$ the stationary distributions.
- Using forced exploration: $\pi_t \underset{t \to \infty}{\longrightarrow} \pi^o(\mathcal{M})$ a.s, hence

$$\omega_t \underset{t \to \infty}{\longrightarrow} \boldsymbol{\omega}^\star(\mathcal{M}).$$

- $N_{sa}(t)/t \underset{t \to \infty}{\longrightarrow} \boldsymbol{\omega}^\star_{sa}(\mathcal{M})$ is the result of an ergodic theorem.

D-Navigation: Why does it work ?

Theorem 2 (Propositions 12, 8 in the paper)

Denote by P_t the kernel of π_t and by ω_t its stationary distribution. Assume that:

- There exists two constants C_t and ρ_t such that for all $n \geq 1$:

  $$\left\| P_t^n - W_t \right\|_\infty \leq C_t \rho_t^n$$

  where W_t is a rank-one matrix whose rows are equal to $\omega_t^\intercal$.
- UNIFORM SPEED: Define $L_t = C_t(1-\rho_t)^{-1}$. Then $\limsup\limits_{t\to\infty} L_t < \infty$ a.s.
- STABILITY: $\mathrm{TV}(P_{t+1}, P_t) \underset{t\to\infty}{\longrightarrow} 0$ a.s.

Then for all (s, a):

$$N_{sa}(t)/t \underset{t\to\infty}{\longrightarrow} \boldsymbol{\omega}^\star_{sa}.$$

Conclusion

Conclusion

1. First Algorithm with problem-specific sample complexity in the online setting !
2. Algorithm can be instance-optimal given an optimization oracle.
3. Although tracking is not possible, achieving some target oracle allocation is still possible through adaptive control of the trajectory, and a powerful ergodic theorem !
4. First step towards understanding problem-specific ε-optimal policy identification.

Thanks !