Reinforcement Learning

Lecture 8 : Bandit tools for Reinforcement Learning

Emilie Kaufmann

M2 MVA, 2023/2024

From bandits to RL

Solve a multi-armed bandit problem
$\simeq$ maximize rewards in a MDP with one state

The bandit world

- several principles for exploration/exploitation
- efficient algorithms (UCB, Thompson Sampling)
- with regret guarantees
- specific algorithms for best arm identification

RL algorithms so far

- ϵ-greedy exploration
- algorithms with (at most) convergence guarantees, not very sample efficient

vs. (more) efficient algorithms (DeepRL) with no theoretical guarantees

Question : can we be inspired by bandit algorithms to

- propose new RL algorithms
- ... with theoretical guarantees ?

Four RL frameworks

Discounted MDP

$$V^\pi(s) = \mathbb{E}^\pi\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \middle| s_1 = s\right]$$

Episodic MDP

$$V^\pi(s) = \mathbb{E}^\pi\left[\sum_{t=1}^{H} r_t \middle| s_1 = s\right]$$

Average reward MDP

$$V^\pi(s) = \lim_{T\to\infty} \frac{1}{T}\mathbb{E}^\pi\left[\sum_{t=1}^{T} r_t \middle| s_1 = s\right]$$

Goal-oriented MDP

$$V^\pi(s) = \mathbb{E}^\pi\left[\sum_{t=1}^{\tau_g^*} r_t \middle| s_1 = s\right]$$

* g is an absorbing goal state, and τ_g is the time to reach the goal

Four RL frameworks

Discounted MDP

$$V^\pi(s) = \mathbb{E}^\pi\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \middle| s_1 = s\right]$$

Episodic MDP

$$V^\pi(s) = \mathbb{E}^\pi\left[\sum_{t=1}^{H} r_t \middle| s_1 = s\right]$$

Average reward MDP

$$V^\pi(s) = \lim_{T\to\infty} \frac{1}{T}\mathbb{E}^\pi\left[\sum_{t=1}^{T} r_t \middle| s_1 = s\right]$$

Goal-oriented MDP

$$V^\pi(s) = \mathbb{E}^\pi\left[\sum_{t=1}^{\tau_g^*} r_t \middle| s_1 = s\right]$$

* g is an absorbing goal state, and τ_g is the time to reach the goal

Question 1 : How many interactions with the MDPs (or episodes) are needed to find a good policy ?

$$\mathbb{P}\left(V^\star - V^{\widehat{\pi}} \leq \varepsilon\right) \geq 1 - \delta$$

while minimizing the sample complexity

Four RL frameworks

Discounted MDP

$$V^\pi(s) = \mathbb{E}^\pi\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \middle| s_1 = s\right]$$

Episodic MDP

$$V^\pi(s) = \mathbb{E}^\pi\left[\sum_{t=1}^{H} r_t \middle| s_1 = s\right]$$

Average reward MDP

$$V^\pi(s) = \lim_{T\to\infty} \frac{1}{T}\mathbb{E}^\pi\left[\sum_{t=1}^{T} r_t \middle| s_1 = s\right]$$

Goal-oriented MDP

$$V^\pi(s) = \mathbb{E}^\pi\left[\sum_{t=1}^{\tau_g^*} r_t \middle| s_1 = s\right]$$

* g is an absorbing goal state, and τ_g is the time to reach the goal

Question 2 : Can the algorithm learn to behave optimally ?
Can be measured by the number of sub-optimal plays or the regret

$$N = \sum_{t=1}^{\infty} \mathbb{1}\left(V^\star(s_t) - V^{\pi_t}(s_t) > \varepsilon\right)$$

(PAC-MDP framework, discounted MDP, see [Kakade, 2003])

Four RL frameworks

Discounted MDP

$$V^\pi(s) = \mathbb{E}^\pi\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \middle| s_1 = s\right]$$

Episodic MDP

$$V^\pi(s) = \mathbb{E}^\pi\left[\sum_{t=1}^{H} r_t \middle| s_1 = s\right]$$

Average reward MDP

$$V^\pi(s) = \lim_{T\to\infty} \frac{1}{T}\mathbb{E}^\pi\left[\sum_{t=1}^{T} r_t \middle| s_1 = s\right]$$

Goal-oriented MDP

$$V^\pi(s) = \mathbb{E}^\pi\left[\sum_{t=1}^{\tau_g^*} r_t \middle| s_1 = s\right]$$

* g is an absorbing goal state, and τ_g is the time to reach the goal

Question 2 : Can the algorithm learn to behave optimally ?
Can be measured by the number of sub-optimal plays or the regret

$$R_T = TV^\star - \sum_{t=1}^{T} r_t \quad \text{in average-reward MDPs}$$

[Jaksch et al., 2010]

Four RL frameworks

Discounted MDP

$$V^\pi(s) = \mathbb{E}^\pi\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \middle| s_1 = s\right]$$

Episodic MDP

$$V^\pi(s) = \mathbb{E}^\pi\left[\sum_{t=1}^{H} r_t \middle| s_1 = s\right]$$

Average reward MDP

$$V^\pi(s) = \lim_{T\to\infty} \frac{1}{T}\mathbb{E}^\pi\left[\sum_{t=1}^{T} r_t \middle| s_1 = s\right]$$

Goal-oriented MDP

$$V^\pi(s) = \mathbb{E}^\pi\left[\sum_{t=1}^{\tau_g^*} r_t \middle| s_1 = s\right]$$

* g is an absorbing goal state, and τ_g is the time to reach the goal

Question 2 : Can the algorithm learn to behave optimally ?
Can be measured by the number of sub-optimal plays or the regret

$$R_K = \sum_{k=1}^{K} \left(V^\star(s_1^k) - V^{\pi_k}(s_1^k)\right) \qquad \text{in episodic MDPs}$$

regret after K episodes, s_1^k : first state of episode K

[Azar et al., 2017]

Four RL frameworks

Discounted MDP

$$V^\pi(s) = \mathbb{E}^\pi\left[\sum_{t=1}^{\infty} \gamma^{t-1} r_t \middle| s_1 = s\right]$$

Episodic MDP

$$V^\pi(s) = \mathbb{E}^\pi\left[\sum_{t=1}^{H} r_t \middle| s_1 = s\right]$$

Average reward MDP

$$V^\pi(s) = \lim_{T\to\infty} \frac{1}{T}\mathbb{E}^\pi\left[\sum_{t=1}^{T} r_t \middle| s_1 = s\right]$$

Goal-oriented MDP

$$V^\pi(s) = \mathbb{E}^\pi\left[\sum_{t=1}^{\tau_g^*} r_t \middle| s_1 = s\right]$$

* g is an absorbing goal state, and τ_g is the time to reach the goal

Question 2 : Can the algorithm learn to behave optimally ?
Can be measured by the number of sub-optimal plays or the regret

$$R_K = \sum_{k=1}^{K} \left(V^\star(s_1^k) - V^{\pi_k}(s_1^k)\right) \qquad \text{in episodic MDPs}$$

regret after K episodes, s_1^k : first state of episode K

[Azar et al., 2017]

(Reminder :) How to solve an episodic MDP ?

$V^\pi(s) = V_1^\pi(s)$, introducing the value function from step h :

$$V_h^\pi(s) = \mathbb{E}^\pi \left[\sum_{t=h}^{H} r_t \middle| s_h = s \right]$$

Non-stationary policies $\pi = (\pi_1, \dots, \pi_H)$ can be of interest

$\pi_h(s)$: action chosen if we are in state s at step h of the episode

Bellman equations for a (deterministic) policy π

For all $h \in \{1, \dots, H\}$, for all $s \in \mathcal{S}$,

$$V_h^\pi(s) = r(s, \pi_h(s)) + \sum_{s' \in \mathcal{S}} p(s'|s, \pi_h(s)) V_{h+1}^\pi(s'),$$

with $V_{H+1}^\pi(s) = 0$ for all $s \in \mathcal{S}$.

➜ policy evaluation using backwards induction

(Reminder :) How to solve an episodic MDP ?

$V^\star(s) = V_1^\star(s)$, introducing the optimal value function from step h :

$$V_h^\star(s) = \max_\pi \mathbb{E}^\pi \left[\sum_{t=h}^{H} r_t \middle| s_h = s \right]$$

Bellman equations for the optimal policy

For all $h \in \{1, \dots, H\}$, for all $s \in \mathcal{S}$, letting $V_{H+1}^\star = 0$

$$V_h^\star(s) = \max_{a \in \mathcal{A}} \left[r(s,a) + \sum_{s' \in \mathcal{S}} p(s'|s,a) V_{h+1}^\star(s') \right]$$

$$\pi_h^\star(s) = \operatorname*{argmax}_{a \in \mathcal{A}} \left[r(s,a) + \sum_{s' \in \mathcal{S}} p(s'|s,a) V_{h+1}^\star(s') \right]$$

➜ The optimal policy $\pi^\star$ is non-stationnary and can be computed using backwards induction (dynamic programming)

(Reminder :) How to solve an episodic MDP ?

$V^\star(s) = V_1^\star(s)$, introducing the optimal value function from step h :

$$V_h^\star(s) = \max_\pi \mathbb{E}^\pi \left[\sum_{t=h}^{H} r_t \middle| s_h = s \right]$$

Bellman equations for the optimal policy

For all $h \in \{1, \dots, H\}$, for all $s \in \mathcal{S}$, letting $V_{H+1}^\star = 0$

$$\begin{aligned} Q_h^\star(s,a) &= r(s,a) + \sum_{s' \in \mathcal{S}} p(s'|s,a) \left[\max_b Q_{h+1}^\star(s',b) \right] \\ \pi_h^\star(s) &= \operatorname*{argmax}_{a \in \mathcal{A}} Q_h^\star(s,a) \end{aligned}$$

➜ The optimal policy $\pi^\star$ is non-stationnary and can be computed using backwards induction (dynamic programming)

(Reminder :) How to solve an episodic MDP ?

$V^\star(s) = V_1^\star(s)$, introducing the optimal value function from step h :

$$V_h^\star(s) = \max_\pi \mathbb{E}^\pi \left[\sum_{t=h}^{H} r_t \middle| s_h = s \right]$$

Bellman equations for the optimal policy

For all $h \in \{1, \dots, H\}$, for all $s \in \mathcal{S}$, letting $V_{H+1}^\star = 0$

$$\begin{aligned} Q_h^\star(s,a) &= r(s,a) + \sum_{s' \in \mathcal{S}} p(s'|s,a) \left[\max_b Q_{h+1}^\star(s',b) \right] \\ \pi^\star &= \text{greedy}(Q^\star) \end{aligned}$$

➜ The optimal policy $\pi^\star$ is non-stationnary and can be computed using backwards induction (dynamic programming)

Outline

1. Regret and Sample Complexity in RL
2. Regret minimization in episodic MDPs
   - Optimism for Reinforcement Learning
   - Thompson Sampling for Reinforcement Learning
3. Scalable heuristics inspired by those principles
4. Sample complexity
5. (Bandit-based) Monte-Carlo Tree Search
   - UCB for Trees : UCT
   - From UCT to Alpha Zero

Learning in episodic MDPs

For each episode $t \in \{1, \dots, T\}$, an episodic RL algorithm

- starts in some initial state $s_1^t \sim \rho \quad$ (e.g. $s_1^t = s_1$)
- selects a policy π^t (based on observations from past episodes)
- uses this policy to generate an episode of length H :

$$s_1^t, a_1^t, r_1^t, s_2^t, \dots, s_H^t, a_H^t, r_H^t$$

where $a_h^t = \pi_h^t(s_h^t)$ and $(r_h^t, s_{h+1}^t) = \mathrm{step}(s_h^t, a_h^t)$

Definition

The (pseudo)-regret of an episodic RL algorithm $\pi = (\pi^t)_{t \in \mathbb{N}}$ in T episodes is

$$R_T(\pi) = \sum_{t=1}^{T} \left[V^\star(s_1^t) - V^{\pi^t}(s_1^t) \right].$$

Reminder : Minimizing regret in bandits

Small regret requires to introduce the right amount of exploration, which can be done with

- ϵ-greedy

explore uniformly with probability ϵ (or ϵ_t), otherwise trust the estimated model

- Upper Confidence Bounds algorithms

act as if the optimistic model were the true model

- Thompson Sampling

act as if a model sampled from the posterior distribution were the true model

What is wrong with ε-greedy in RL ?

Example : Q-Learning with ε-greedy

➜ ε-greedy exploration

$$a_t = \begin{cases} \operatorname{argmax}_{a \in \mathcal{A}} \hat{Q}_t(s_t, a) & \text{with probability } 1 - \varepsilon_t \\ \sim \mathcal{U}(\mathcal{A}) & \text{with probability } \epsilon_t \end{cases}$$

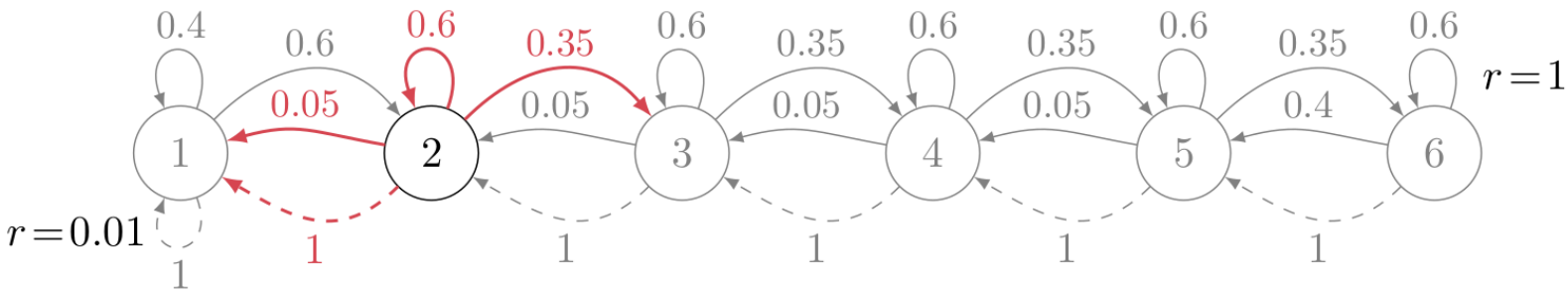
➜ Q-Learning update

$$\hat{Q}_t(s_t, a_t) = \hat{Q}_{t-1}(s_t, a_t) + \alpha_t \left(r_t + \gamma \max_b \hat{Q}_{t-1}(s_t, b) - \hat{Q}_{t-1}(s_t, a_t) \right)$$

⚠ $\hat{Q}_t(s, a)$ is *not* an unbiased estimate of $Q^\star(s, a)$...
(except in the bandit case)

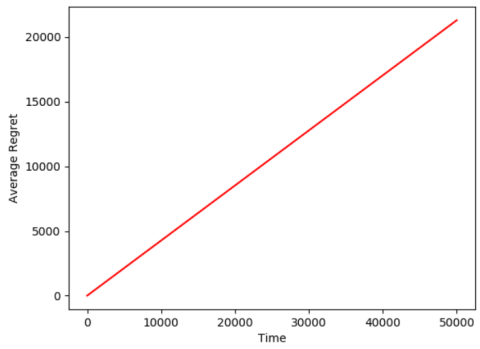
What is wrong with ε-greedy ?

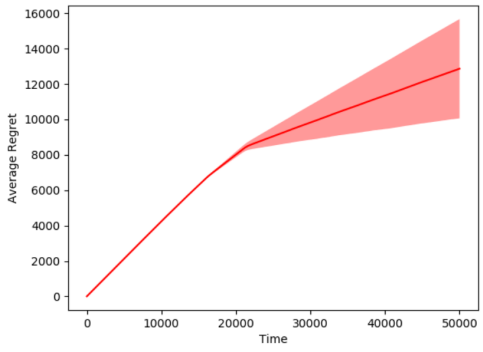
The RiverSwim MDP :

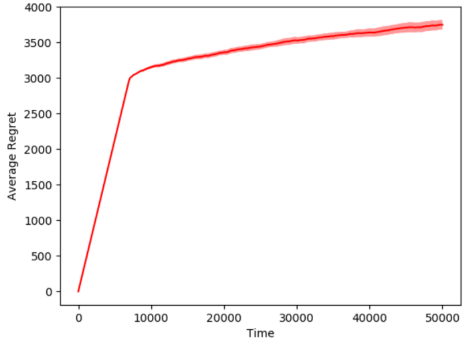


⚠ ε can be hard to tune...

What is wrong with ε-greedy ?







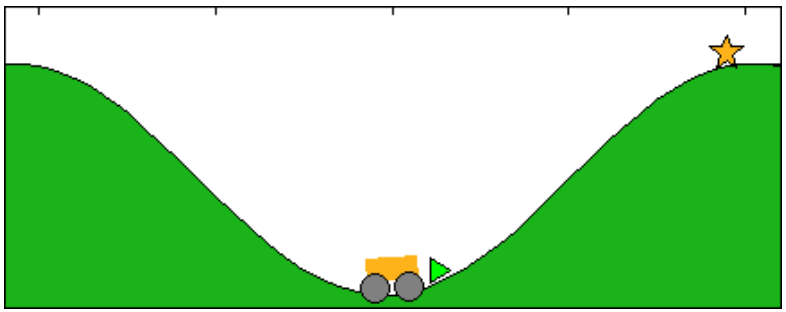
$$\epsilon_t = 0.5 \qquad \epsilon_t = \frac{\epsilon_0}{(N(s_t) - 1000)^{2/3}} \qquad \epsilon_t = \begin{cases} 1 & \text{if } t < 7000 \\ \frac{\epsilon_0}{\sqrt{N(s_t)}} & \text{otherwise} \end{cases}$$

credit : Alessandro Lazaric

⚠ ε-greedy performs undirected exploration

- alternative : model-based methods in which exploration is targeted towards *uncertain regions* of the state/action space

Other sparse rewards environments



Moutain Car

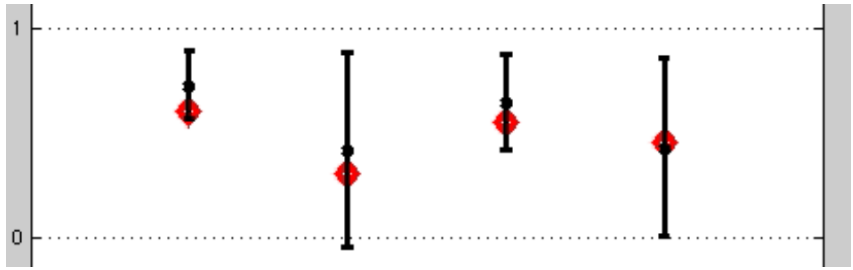
Montezuma Revenge (Atari)

Outline

1. Regret and Sample Complexity in RL
2. Regret minimization in episodic MDPs
   - Optimism for Reinforcement Learning
   - Thompson Sampling for Reinforcement Learning
3. Scalable heuristics inspired by those principles
4. Sample complexity
5. (Bandit-based) Monte-Carlo Tree Search
   - UCB for Trees : UCT
   - From UCT to Alpha Zero

Towards an optimistic learning algorithm

- **Reminder** : Optimistic Bandit model



set of possible bandit models $\boldsymbol{\mu} = (\mu_1, \mu_2, \mu_3, \mu_4)$:

$$\mathcal{M}_t = \mathcal{I}_1(t) \times \mathcal{I}_2(t) \times \mathcal{I}_3(t) \times \mathcal{I}_4(t)$$

An optimistic bandit model is

$$\boldsymbol{\mu}_t^+ \in \operatorname*{argmax}_{\boldsymbol{\mu} \in \mathcal{M}_t} \mu^\star$$

➜ the best arm (optimal policy) in $\boldsymbol{\mu}_t^+$ is

$$A_{t+1} = \operatorname*{argmax}_{a \in \mathcal{A}} \mathrm{UCB}_a(t)$$

Towards an optimistic learning algorithm

- **Extension** : Optimistic Markov Decision Process

set of possible MDPs $\boldsymbol{M} = \langle \mathcal{S}, \mathcal{A}, r, p \rangle$:

$$\mathcal{M}_t = \{\langle \mathcal{S}, \mathcal{A}, r, p \rangle : r, p \in \mathcal{B}_t^r \times \mathcal{B}_t^p\}$$

An optimistic Markov Decision Process is

$$\boldsymbol{M}_t^+ \in \operatorname*{argmax}_{\boldsymbol{M} \in \mathcal{M}_t} V^{\star, \boldsymbol{M}}$$

➜ Explore the MDP using the optimistic policy

$$\pi_t^+ : \text{ optimal policy in } \boldsymbol{M}_t^+$$

First proposed for average-rewards MDPs (UCRL [Jaksch et al., 2010])

Towards an optimistic learning algorithm

- **Extension** : Optimistic Markov Decision Process

set of possible MDPs $\boldsymbol{M} = \langle \mathcal{S}, \mathcal{A}, r, p \rangle$:

$$\mathcal{M}_t = \{\langle \mathcal{S}, \mathcal{A}, r, p \rangle : r, p \in \mathcal{B}_t^r \times \mathcal{B}_t^p\}$$

An optimistic Markov Decision Process is

$$\boldsymbol{M}_t^+ \in \operatorname*{argmax}_{\boldsymbol{M} \in \mathcal{M}_t} V^{\star, \boldsymbol{M}}$$

➜ Explore the MDP using the optimistic policy

$$\pi_t^+ : \text{ optimal policy in } \boldsymbol{M}_t^+$$

First proposed for average-rewards MDPs (UCRL [Jaksch et al., 2010])

Challenges

1. How to construct the set $\mathcal{M}_t$ of possible MDPs ?
2. How to numerically compute π_t^+ ?

Optimistic algorithm for episodic MDPs

We are interested in π_t^+ the optimal policy in

$$\boldsymbol{M}_t^+ \in \operatorname*{argmax}_{\boldsymbol{M}\in\mathcal{M}_t} V^{\star,\boldsymbol{M}}(s_1)$$

ie

$$\begin{aligned}\pi_{t,h}^+(s) &= \operatorname*{argmax}_{a\in\mathcal{A}} Q_h^{\star,\boldsymbol{M}_t^+}(s,a) \\ &= \operatorname*{argmax}_{a\in\mathcal{A}} \max_{\boldsymbol{M}\in\mathcal{M}_t} Q_h^{\star,\boldsymbol{M}}(s,a)\end{aligned}$$

Let's relax this a little bit and define

$$\pi_h^t(s) = \operatorname*{argmax}_{a\in\mathcal{A}} \overline{Q}_h^t(s,a)$$

where $\overline{Q}^t$ satisfies

$$\forall \boldsymbol{M}\in\mathcal{M}_t, \forall (h,s,a), \quad Q_h^{\star,\boldsymbol{M}}(s,a) \le \overline{Q}_h^t(s,a)$$

➜ If the true MDP belongs to $\mathcal{M}_t$, π^t is the greedy policy wrt an upper bound on the optimal Q value function

Step 1 : Constructing $\mathcal{M}_t$

$$\mathcal{M}_t = \left\{ \langle \mathcal{S}, \mathcal{A}, r, p \rangle : \forall (s,a) \in \mathcal{S} \times \mathcal{A}, r(s,a) \in \mathcal{B}_t^r(s,a), p(\cdot|s,a) \in \mathcal{B}_t^p(s,a) \right\}$$

Idea : build individual confidence regions

- on the average reward $r(s,a)$: $\mathcal{B}_t^r(s,a) \subseteq \mathbb{R}$
- on the transition probability vector $p(\cdot|s,a)$: $\mathcal{B}_t^p(s,a) \subseteq \Delta(\mathcal{S})$

that rely on the empirical estimates

$$\hat{r}_t(s,a) = \frac{1}{n_t(s,a)} \sum_{i=1}^{n_t(s,a)} r[i] \quad \text{and} \quad \hat{p}_t(s'|s,a) = \frac{n_t(s,a,s')}{n_t(s,a)}$$

$n_t(s,a)$: number of visits of (s,a) before episode t
$n_t(s,a,s')$: number of times s' was the next state when the transition (s,a) was performed before episode t

Goal : $\mathbb{P}_{\boldsymbol{M}}\left(\boldsymbol{M} \in \mathcal{M}_t\right)$ is close to 1

Step 1 : Constructing $\mathcal{M}_t$

$$\mathcal{M}_t = \left\{ \langle \mathcal{S}, \mathcal{A}, r, p \rangle : \forall (s,a) \in \mathcal{S} \times \mathcal{A}, r(s,a) \in \mathcal{B}_t^r(s,a), p(\cdot|s,a) \in \mathcal{B}_t^p(s,a) \right\}$$

Idea : build individual confidence regions

- on the average reward $r(s,a)$: $\mathcal{B}_t^r(s,a) \subseteq \mathbb{R}$

Assuming rewards that are bounded in $[0,1]$,

$$\mathcal{B}_t^r(s,a) = \left[\hat{r}_t(s,a) - \sqrt{\frac{\log(8SA(n_t(s,a))^2/\delta)}{2n_t(s,a)}} ; \hat{r}_t(s,a) + \sqrt{\frac{\log(8SA(n_t(s,a))^2/\delta)}{2n_t(s,a)}} \right]$$

satisfies

$$\mathbb{P}\left(\exists t \in \mathbb{N} : \exists (s,a) \in \mathcal{S} \times \mathcal{A}, r(s,a) \notin \mathcal{B}_t^r(s,a) \right) \leq \frac{\delta}{2}.$$

(Hoeffding inequality + union bounds)

Step 1 : Constructing $\mathcal{M}_t$

$$\mathcal{M}_t = \left\{ \langle \mathcal{S}, \mathcal{A}, r, p \rangle : \forall (s,a) \in \mathcal{S} \times \mathcal{A}, r(s,a) \in \mathcal{B}_t^r(s,a), p(\cdot|s,a) \in \mathcal{B}_t^p(s,a) \right\}$$

Idea : build individual confidence regions

- on the average reward $r(s,a)$: $\mathcal{B}_t^r(s,a) \subseteq \mathbb{R}$

Assuming rewards that are bounded in $[0,1]$,

$$\mathcal{B}_t^r(s,a) = \left[\hat{r}_t(s,a) - \beta_t^r(s,a); \hat{r}_t(s,a) + \beta_t^r(s,a) \right]$$

satisfies

$$\mathbb{P}\left(\exists t \in \mathbb{N} : \exists (s,a) \in \mathcal{S} \times \mathcal{A}, r(s,a) \notin \mathcal{B}_t^r(s,a) \right) \leq \frac{\delta}{2}.$$

(Hoeffding inequality + union bounds)

Step 1 : Constructing $\mathcal{M}_t$

$$\mathcal{M}_t = \left\{ \langle \mathcal{S}, \mathcal{A}, r, p \rangle : \forall (s,a) \in \mathcal{S} \times \mathcal{A}, r(s,a) \in \mathcal{B}_t^r(s,a), p(\cdot|s,a) \in \mathcal{B}_t^p(s,a) \right\}$$

Idea : build individual confidence regions

- on the transition probability vector $p(\cdot|s,a)$: $\mathcal{B}_t^p(s,a) \subseteq \Delta(\mathcal{S})$

$$\mathcal{B}_t^p(s,a) = \left\{ p(\cdot|s,a) \in \Delta(\mathcal{S}) : \|p(\cdot|s,a) - \hat{p}_t(\cdot|s,a)\|_1 \leq C\sqrt{\frac{S \log(n_t(s,a)/\delta)}{n_t(s,a)}} \right\}$$

satisfies

$$\mathbb{P}\Big(\exists t \in \mathbb{N} : \exists (s,a) \in \mathcal{S}, \mathcal{A}, \ p(\cdot|s,a) \notin \mathcal{B}_t^p(s,a) \Big) \leq \frac{\delta}{2}.$$

(Freedman inequality + union bounds)
[Jaksch et al., 2010]

Step 1 : Constructing $\mathcal{M}_t$

$$\mathcal{M}_t = \left\{ \langle \mathcal{S}, \mathcal{A}, r, p \rangle : \forall (s,a) \in \mathcal{S} \times \mathcal{A}, r(s,a) \in \mathcal{B}_t^r(s,a), p(\cdot|s,a) \in \mathcal{B}_t^p(s,a) \right\}$$

Idea : build individual confidence regions

- on the transition probability vector $p(\cdot|s,a)$: $\mathcal{B}_t^p(s,a) \subseteq \Delta(\mathcal{S})$

$$\mathcal{B}_t^p(s,a) = \left\{ p(\cdot|s,a) \in \Delta(\mathcal{S}) : \|p(\cdot|s,a) - \hat{p}_t(\cdot|s,a)\|_1 \leq \beta_t^p(s,a) \right\}$$

satisfies

$$\mathbb{P}\Big(\exists t \in \mathbb{N} : \exists (s,a) \in \mathcal{S}, \mathcal{A}, \ p(\cdot|s,a) \notin \mathcal{B}_t^p(s,a) \Big) \leq \frac{\delta}{2}.$$

(Freedman inequality + union bounds)
[Jaksch et al., 2010]

Step 1 : Constructing $\mathcal{M}_t$

$$\mathcal{M}_t = \left\{ \langle \mathcal{S}, \mathcal{A}, r, p \rangle : \forall (s,a) \in \mathcal{S} \times \mathcal{A}, r(s,a) \in \mathcal{B}_t^r(s,a), p(\cdot|s,a) \in \mathcal{B}_t^p(s,a) \right\}$$

$$\begin{aligned} \mathcal{B}_t^r(s,a) &= \left[\hat{r}_t(s,a) - \beta_t^r(s,a); \hat{r}_t(s,a) + \beta_t^r(s,a) \right] \\ \mathcal{B}_t^p(s,a) &= \left\{ p(\cdot|s,a) \in \Delta(\mathcal{S}) : \|p(\cdot|s,a) - \hat{p}_t(\cdot|s,a)\|_1 \leq \beta_t^p(s,a) \right\} \end{aligned}$$

with exploration bonuses :

$$\begin{aligned} \beta_t^r(s,a) &\propto \sqrt{\frac{\log(n_t(s,a)/\delta)}{n_t(s,a)}} \\ \beta_t^p(s,a) &\propto \sqrt{\frac{S \log(n_t(s,a)/\delta)}{n_t(s,a)}} \end{aligned}$$

$$\mathbb{P}_{\boldsymbol{M}} \left(\forall t \in \mathbb{N}, \boldsymbol{M} \in \mathcal{M}_t \right) \geq 1 - \delta$$

Step 2 : Finding the upper bound

For some appropriate exploration bonus $\beta_t(s,a)$ the function $\overline{Q}^t$ defined inductively by $\overline{Q}^t_{H+1} = 0$, and for all $h \in [H]$, $(s,a) \in \mathcal{S} \times \mathcal{A}$,

$$\begin{aligned} \overline{Q}^t_h(s,a) &= \hat{r}_t(s,a) + \beta_t(s,a) + \sum_{s' \in \mathcal{S}} \hat{p}_t(s'|s,a) \max_b \overline{V}^t_{h+1}(s') \\ \overline{V}^t_h(s) &= \min\left[H-h; \max_b \overline{Q}^t_h(s,b)\right], \end{aligned}$$

satisfies $\forall \boldsymbol{M} \in \mathcal{M}_t,\ Q^{\star,\boldsymbol{M}}_h(s,a) \leq \overline{Q}_h(s,a)$.

Proof : (in class)

$$\beta_t(s,a) = \beta^r_t(s,a) + H\beta^p_t(s,a) \simeq CH\sqrt{\frac{S\log(n_t(s,a))}{n_t(s,a)}}$$

➜ $\beta_t(s,a)$ scales in $1/\sqrt{n_t(s,a)}$ where $n_t(s,a)$ is the number of previous visits to (s,a).

Optimistic algorithms

An optimistic algorithm uses in episode t the policy $\pi_t = \text{greedy}(\overline{Q}^t)$ with $\overline{Q}_h^t(s,a)$ an upper confidence bound on $Q_h^\star(s,a)$.

Different choices of $\mathcal{M}_t$ and/or $\overline{Q}^t$ lead to different optimistic algorithms. The one we described is close to the UCB-VI algorithm proposed by [Azar et al., 2017] (with Hoeffding bonuses).

UCB-VI

The exporation policy in episode t is greedy wrt to

$$\begin{aligned}
\overline{Q}_h^t(s,a) &= \hat{r}_t(s,a) + \beta_t(s,a) + \sum_{s' \in \mathcal{S}} \hat{p}_t(s'|s,a) \max_b \overline{V}_{h+1}^t(s') \\
\overline{V}_h^t(s) &= \min\left[H-h; \max_b \overline{Q}_h^t(s,b)\right],
\end{aligned}$$

for some exploration bonus $\beta_t(s,a)$.

Theoretical guarantees

Theorem [Azar et al., 2017]

UCB-VI (with a slightly more complicated bonus) satisfies

$$\mathbb{P}\left(R_T(\text{UCB-VI}) = \mathcal{O}\left(\sqrt{H^2SAT\log(T/\delta)}\right)\right) \geq 1-\delta.$$

(Worse case) Lower bound

For every algorithm $\mathcal{A}$, there exists an MDP $\boldsymbol{M}$ for which

$$\mathbb{E}_{\boldsymbol{M}}[R_T(\mathcal{A})] \geq c\sqrt{H^2SAT}$$

for some constant c.

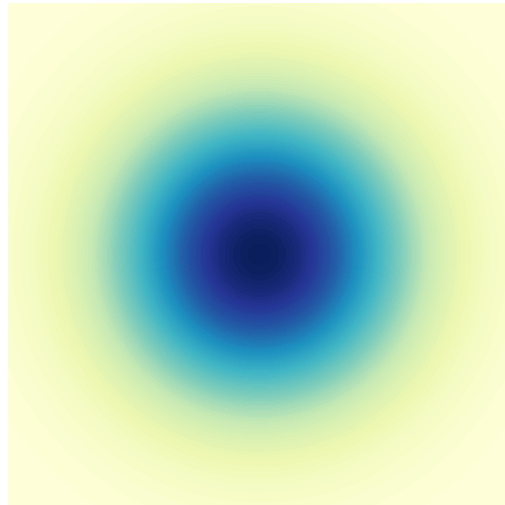
- UCB-VI is nearly optimal in the worse-case.
- Problem-dependent results also exist for (more complex) optimistic algorithms, see e.g. [Simchowitz and Jamieson, 2019]

Outline

1. Regret and Sample Complexity in RL
2. Regret minimization in episodic MDPs
   - Optimism for Reinforcement Learning
   - Thompson Sampling for Reinforcement Learning
3. Scalable heuristics inspired by those principles
4. Sample complexity
5. (Bandit-based) Monte-Carlo Tree Search
   - UCB for Trees : UCT
   - From UCT to Alpha Zero

Posterior Sampling for RL

Bayesian assumption : $\boldsymbol{M}$ is drawn from some prior distribution ν_0.



$\nu_t \in \Delta(\mathcal{M})$: posterior distribution over the set of MDPs

Optimism	**Posterior Sampling**
Set of possible MDPs	Posterior distribution over MDPs
Compute the optimistic MDP	Sample from the posterior distribution

Prior and posterior for MDPs

In the tabular case, there are natural conjuguate priors.

Rewards

- If the rewards are binary, use a uniform prior $r(s,a) \sim \mathcal{U}([0,1])$ $\rightarrow$ Beta posterior (see previous class)
- Otherwise, either use an appropriate parameteric prior, or a binarization trick $r_t \in [0,1] \rightarrow \widetilde{r}_t = \mathbb{1}(U_t \leq r_t)$

Transitions

- Dirichlet prior : $p(\cdot|s,a) \sim \mathrm{Dir}(1,\dots,1)$ (S ones)
- The posterior is

$$p(\cdot|s,a) \sim \mathrm{Dir}\left((1+n_t(s,a,s'))_{s' \in \mathcal{S}}\right)$$

Posterior Sampling for Episodic RL

Algorithm 1: PSRL

Input : Prior distribution ν_0

for $t = 1, 2, \dots$ **do**
- $s_1 \sim \rho$ \\ get the starting state of episode t
- Sample $\widetilde{M}_t \sim \nu_{t-1}$ \\ sample an MDP from the current posterior distribution
- Compute $\widetilde{\pi}^t$ an optimal policy for $\widetilde{M}_t$ \\ backwards induction
- **for** $h = 1, \dots, H$ **do**
  - $a_h = \widetilde{\pi}^t_h(s_h)$ \\ choose next action according to $\widetilde{\pi}^t$
  - $r_h, s_{h+1} = \text{step}(s_h, a_h)$
- **end**
- Update ν_t based on ν_{t-1} and $\{(s_h, a_h, r_h, s_{h+1})\}_{h=1}^H$

end

[Strens, 2000, Osband et al., 2013]

Theoretical guarantees

- A **Bayesian** analysis of PSRL = regret integrated over the prior

Theorem [Osband et al., 2013]

$\mathbb{E}[R_T(\text{PSRL})] = \int \mathbb{E}_{\boldsymbol{M}}[R_T(\text{PSRL})] d\nu_0(\boldsymbol{M}) = \mathcal{O}(S\sqrt{H^2AT})$

- **Frequentist guarantees**

Only variants of PSRL have been analyzed in the frequentist setting :

➜ posterior inflation (augmenting the variance of the posterior)

➜ drawing J samples from the posterior and taking the best one (which is some form of optimism)

Theorem [Tiapkin et al., 2022]

There is a tuning of Optimistic PSRL for which
$\mathbb{P}\left(R_T(\text{OPSRL}) = \mathcal{O}\left(\sqrt{H^3SAT}\right)\right) \geq 1-\delta.$

Outline

1. Regret and Sample Complexity in RL
2. Regret minimization in episodic MDPs
   - Optimism for Reinforcement Learning
   - Thompson Sampling for Reinforcement Learning
3. Scalable heuristics inspired by those principles
4. Sample complexity
5. (Bandit-based) Monte-Carlo Tree Search
   - UCB for Trees : UCT
   - From UCT to Alpha Zero

Limitations of optimistic approaches

An important message from optimistic approaches :

➜ Do not only trust the estimated MDP $\hat{M}_t$, but take into account the uncertainty in the underlying estimate

$$\mathcal{B}_t^r(s,a) = \left[\hat{r}_t(s,a) - \beta_t^r(s,a); \hat{r}_t(s,a) + \beta_t^r(s,a)\right]$$

$$\mathcal{B}_t^p(s,a) = \left\{p(\cdot|s,a) \in \Delta(\mathcal{S}) : \|p(\cdot|s,a) - \hat{p}_t(\cdot|s,a)\|_1 \leq \beta_t^p(s,a)\right\}$$

expressed by exploration bonuses scaling in $\sqrt{\frac{1}{n_t(s,a)}}$ where $n_t(s,a)$ is the count (=number of visits) of (s,a).

Scaling for large state action spaces ?

- each state action pair may be visited only very little...
- UCB-VI is quite inefficient in practice for very large state-spaces

Extrinsic versus intrisic reward

UCB-VI is a model-based approach.

- a pure model-based algorithm would estimate the model $(\hat{r}, \hat{p})$ and play the optimal policy in the estimated model
- instead, it plays the optimal policy in a modified model where the reward is replaced by

$$\hat{r}_h^t(s,a) \qquad \underbrace{+ \quad \beta_t(s,a)}_{\substack{\text{add some reward for the} \\ \text{visitation of undervisited states} \\ = \text{intrisic reward}}}$$

More general idea : run any (possibly model-free) RL algorithm replacing the collected (extrinsic) reward r_t by

$$r_t^+ = r_t + r_t^{\mathsf{I}}$$

where r_t^{I} is *some* form of intrinsic reward.

Count-based exploration

General principle

1. Estimate a "proxi" for the number of visits of a state $\widetilde{n}_t(s)$
2. Add an exploration bonus directly to the collected rewards :

$$r_t^+ = r_t + c\sqrt{\frac{1}{\widetilde{n}_t(s_t)}}$$

3. Run any (Deep)RL algorithm on $\mathcal{D} = \bigcup_t \left\{(s_t, a_t, r_t^+, s_{t+1})\right\}$.

Count-based exploration

General principle

1. Estimate a "proxi" for the number of visits of a state $\widetilde{n}_t(s)$
2. Add an exploration bonus directly to the collected rewards :

$$r_t^+ = r_t + c\sqrt{\frac{1}{\widetilde{n}_t(s_t)}}$$

3. Run any (Deep)RL algorithm on $\mathcal{D} = \bigcup_t \left\{(s_t, a_t, r_t^+, s_{t+1})\right\}$.

Example of pseudo-counts :

- use density estimation [Bellemare et al., 2016]
- use a hash function, e.g. $\phi : \mathcal{S} \to \{-1, 1\}^k$ [Tang et al., 2017]
  $n(\phi(s_t)) \leftarrow n(\phi(s_t)) + 1$
- use kernels $n(s_t) = \sum_{s \in \mathcal{H}_t} K(s_t, s)$ [Badia et al., 2020]

Other forms of intrisic rewards

Other forms of intrisic rewards have been proposed in the literature, some of which driven by curiosity.

➜ make the agent willing to discover new parts of the space

Typical example :

- learn to predict the next state (e.g. with a neural network), and use the prediction error as a reward

(e.g., [Burda et al., 2018])

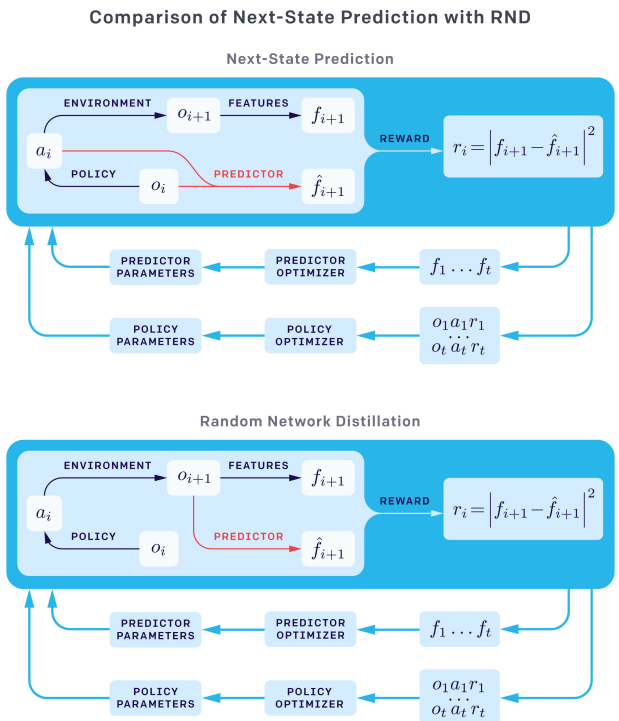
Limitation :

- can attract the agent towards highly stochastic parts of the environment ("noisy TV" issue)

Random Network Distillation

Idea : change the prediction problem to be fixed and deterministic
Draw a random neural network $\widetilde{f}$ at the beginning, and learn to predict $\widetilde{f}(s)$ in each state s, use the error as intrisic reward.

[Burda et al., 2019]

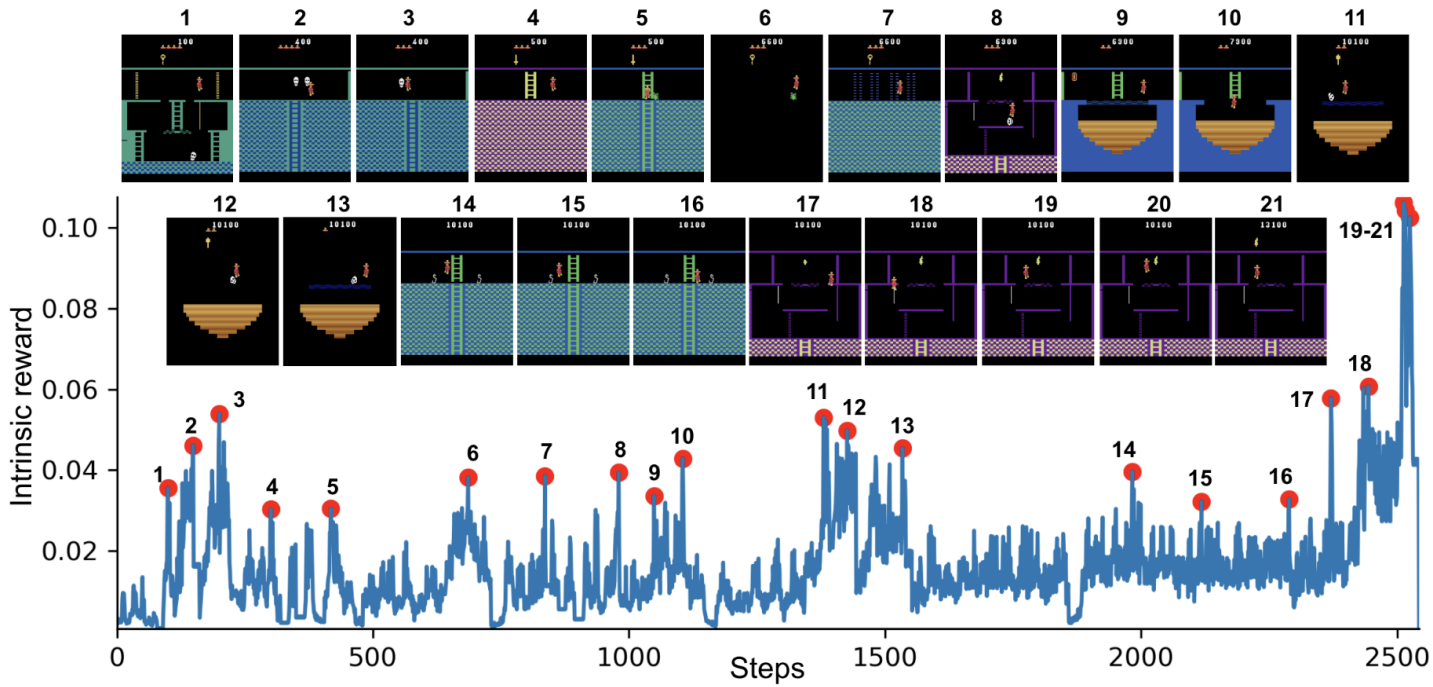


source : OpenAI, see their nice blog article

Random Network Distillation

Other trick : smarter combination of the extrinsic and intrinsic reward, with two heads in the global neural network

- Solving Montezuma !



Never Give Up

Idea : The RND bonus is still vanishing. Couple it with a non-vanishing count-based intrisic reward, taking into account the last episode only.

[Badia et al., 2020]

For the state visited at time t, the intrisic reward is

$$r_t^I = f(r_t^{I,\mathrm{RND}}) \times \frac{1}{\sqrt{\sum_{s' \in \mathrm{NN}_k^{\mathrm{episode}}(s_t)} K_g(s_t, s')} + c}$$

where

- $f(r)$ is some clipping function
- $\mathrm{NN}_k^{\mathrm{episode}}(s_t)$ is the set of k nearest neighbors among the states visited within the last episode, according to the kernel function K_g
- K_g is a kernel function depending on the distance between state representations, $K_g(s, s') = h(||g(s) - g(s'))$

Limitations of Posterior Sampling

An important message from posterior sampling :

➜ Adding some noise to the estimated MDP $\hat{M}_t$ is helpful !

$$\begin{aligned} \widetilde{r}_t(s,a) &= \hat{r}_t(s,a) + \epsilon_t(s,a) \\ \widetilde{p}_t(s'|s,a) &= \hat{p}_t(\cdot|s,a) + \epsilon'_t(s,a). \end{aligned}$$

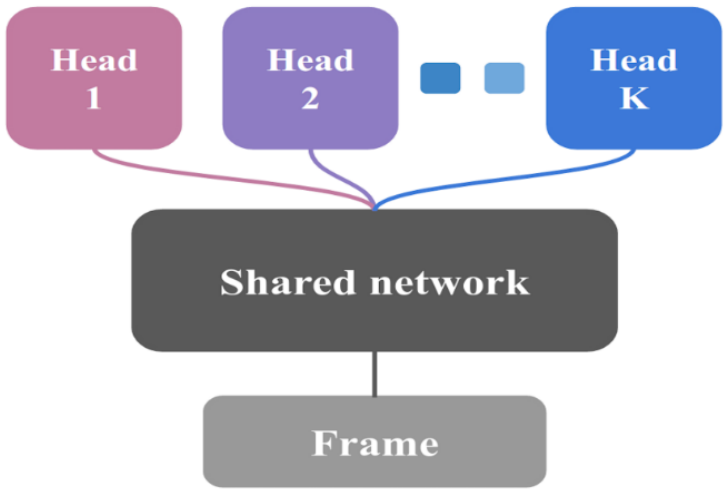
Scaling for large state action spaces ?

- maintaining independent posterior over all state action rewards and transitions can be costly
- more sophisticated prior distributions encoding some structure and the associated posteriors can be hard to sample from

➜ use other type of randomized exploration, not necessarily Bayesian

Bootstrap DQN

Idea : maintain a "distribution of Q-values"
($\neq$ model-based method)



K "boostrap" heads to generate K different estimated Q-values
$Q_1(s,a), \ldots, Q_K(s,a)$

[Osband et al., 2016]

Bootstrap DQN

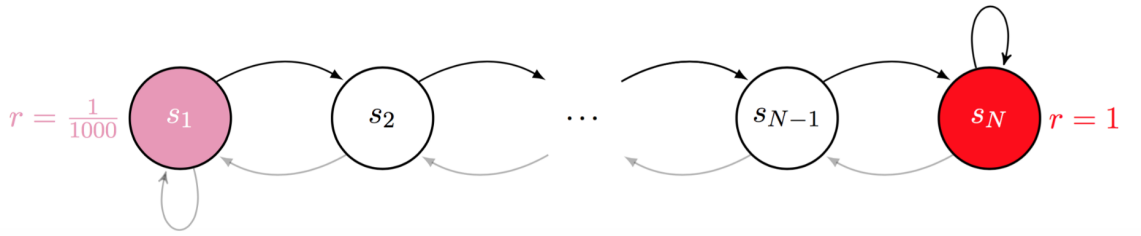
Algorithm 1 Bootstrapped DQN

Input: Value function networks Q with K outputs $\{Q_k\}_{k=1}^K$. Masking distribution M.
Let B be a replay buffer storing experience for training.
for each episode **do**
    Obtain initial state from environment s_0
    Pick a value function to act using $k \sim \text{Uniform}\{1, \dots, K\}$
    for step $t = 1, \dots$ until end of episode **do**
        Pick an action according to $a_t \in \arg\max_a Q_k(s_t, a)$
        Receive state s_{t+1} and reward r_t from environment, having taking action a_t
        Sample bootstrap mask $m_t \sim M$
        Add $(s_t, a_t, r_{t+1}, s_{t+1}, m_t)$ to replay buffer B
    end for
end for

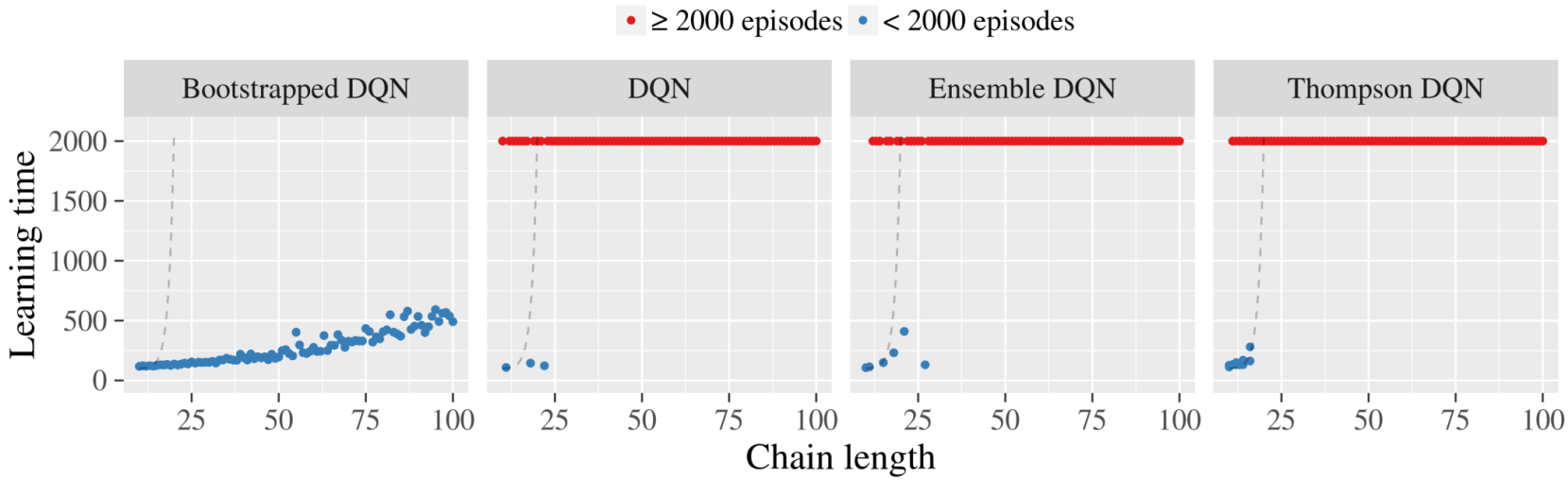
$m_t \in \{0,1\}^k$ determines which heads will be updated with the transition

$$g_t^k = m_t^k \left(r_t + \gamma \max_b Q(s_t, b; \theta_-) - Q_k(s_t, a_t; \theta) \right) \nabla_\theta Q_k(s_t, a_t; \theta)$$

Bootstrap DQN



"Deep Exploration" occurs with Bootstrap DQN



Another possible idea :

➜ add noise in the neural network parameters

(e.g. Noisy Networks [Fortunato et al., 2017])

Outline

1. Regret and Sample Complexity in RL
2. Regret minimization in episodic MDPs
   - Optimism for Reinforcement Learning
   - Thompson Sampling for Reinforcement Learning
3. Scalable heuristics inspired by those principles
4. Sample complexity
5. (Bandit-based) Monte-Carlo Tree Search
   - UCB for Trees : UCT
   - From UCT to Alpha Zero

Best Policy Identification

Best policy identification algorithm :

- π^t : exploratory policy used for episode t
- τ : stopping rule (should we stop exploration ?)
- $\widehat{\pi}$: guess for a good policy

Goal : (ε, δ)-PAC algorithm with small sample complexity τ

$$\mathbb{P}\left(V^\star(s_1) - V^{\widehat{\pi}}(s_1) \leq \varepsilon\right) \geq 1 - \delta.$$

(Worse case) lower bound [Domingues et al., 2021]

For every (ε, δ)-PAC algorithm, there exists an MDP $\boldsymbol{M}$ for which

$$\mathbb{E}_{\boldsymbol{M}}[\tau] \geq c \frac{SAH^2}{\varepsilon^2} \log\left(\frac{1}{\delta}\right)$$

for some constant c.

Matching the lower bound with UCB-VI

Idea 1 : when UCB-VI is run long enough, it starts using good policies
$\rightarrow$ output one of the policies used in the past, chosen at random

[Jin et al., 2018]

Idea 2 : couple UCB-VI with an adaptive stopping rule

$$\tau = \inf\{t \in \mathbb{N} : \overline{V}_1^t(s_1) - \underline{V}_1^t(s_1) \leq \varepsilon\}$$

and output $\widehat{\pi} = \text{greedy}(\underline{Q}^\tau)$ where $\underline{V}^t, \underline{Q}^t$ are lower confidence bounds on the optimal values

[Kaufmann et al., 2021]
[Ménard et al., 2021]

➜ nearly matching the lower bound

Question : Beyond worse-case guarantees ? Best results obtained for algorithms quite different from UCB-VI [Wagenmaker et al., 2022]

Reward-free exploration

- π^t : exploratory policy used in episode t
- τ : stopping rule
- output : database of transitions $\mathcal{D} = \left\{ (s_h^t, a_h^t, s_{h+1}^t)_{\substack{t \leq \tau \\ h \leq H}} \right\}$

Goal : Given any reward function r given after exploration, output a guess $\widehat{\pi}_r$ of the optimal policy in MDP when the reward function is r. (e.g. the optimal policy in the MDP with parameters $(\widehat{p}, r)$)

Formalization : minimize the sample complexity τ while ensuring

$$\mathbb{P}\left(\forall r, V^\star(s_1; r) - V^{\widehat{\pi}_r}(s_1; r) \leq \varepsilon \right) \geq 1 - \delta$$

[Jin et al., 2020]

Using UCB-VI for reward-free exploration

Use a UCB-VI like exploration policy ignoring the rewards, i.e. $\pi^t = \text{greedy}(\overline{Q}^t)$ where

$$\begin{aligned}\overline{Q}_h^t(s,a) &= \widetilde{\beta}_t(s,a) + \sum_{s' \in \mathcal{S}} \hat{p}_t(s'|s,a) \max_b \overline{V}_{h+1}^t(s') \\ \overline{V}_h^t(s) &= \min\left[H-h; \max_b \overline{Q}_h^t(s,b)\right],\end{aligned}$$

and stop when $\max_a \overline{Q}_1^t(s,a) \leq \varepsilon/2$.

[Kaufmann et al., 2021]

➜ using intrinsic rewards only can help learn good exploratory policies

Outline

1. Regret and Sample Complexity in RL
2. Regret minimization in episodic MDPs
   - Optimism for Reinforcement Learning
   - Thompson Sampling for Reinforcement Learning
3. Scalable heuristics inspired by those principles
4. Sample complexity
5. (Bandit-based) Monte-Carlo Tree Search
   - UCB for Trees : UCT
   - From UCT to Alpha Zero

Monte-Carlo Tree Search

MCTS is a family of methods that adaptively explore the tree of possible next states in a given state s_1, in order to find the best action in s_1.

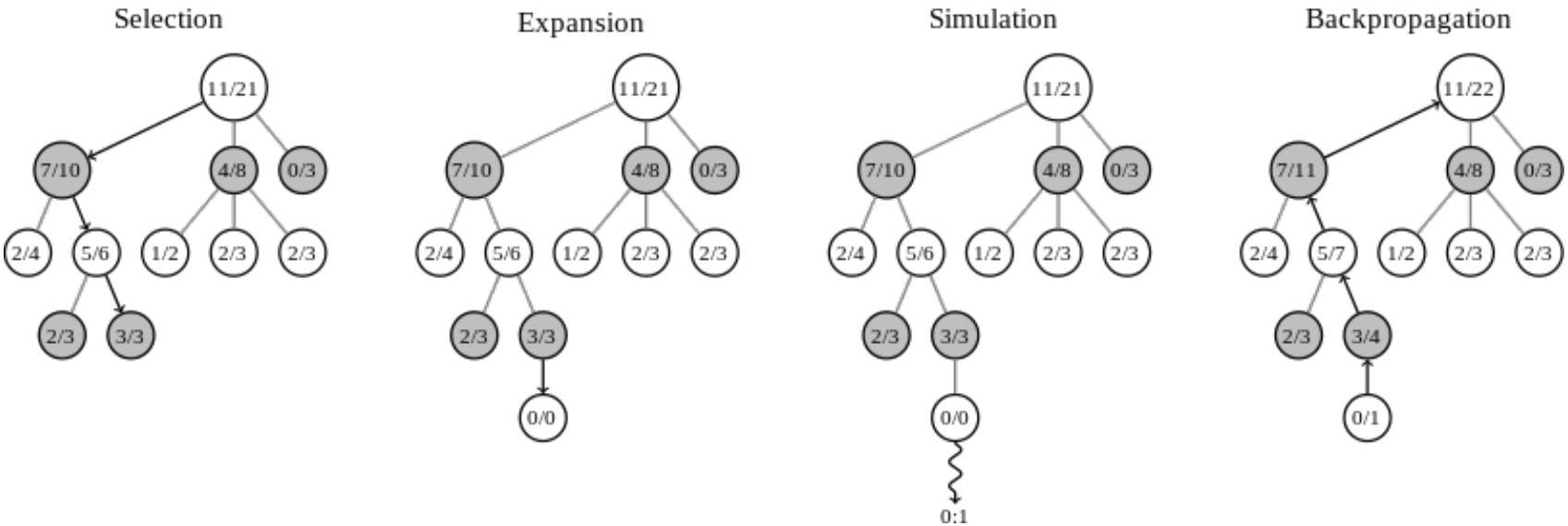


FIGURE – A generic MCTS algorithm for a game

MCTS requires a generative model (to sample trajectories from s_1)

Outline

1. Regret and Sample Complexity in RL
2. Regret minimization in episodic MDPs
   - Optimism for Reinforcement Learning
   - Thompson Sampling for Reinforcement Learning
3. Scalable heuristics inspired by those principles
4. Sample complexity
5. (Bandit-based) Monte-Carlo Tree Search
   - UCB for Trees : UCT
   - From UCT to Alpha Zero

The UCT algorithm

Bandit-Based Monte-Carlo planning : to select a path in the tree, run a bandit algorithm each time a children (next action) needs to be selected

UCT = UCB for Trees [Kocsis and Szepesvári, 2006]

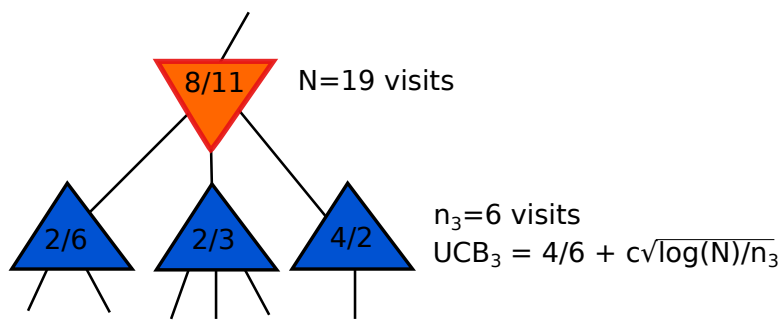
UCT in a Game Tree

In a MAX node s (= root player move), select an action

$$\underset{a \in \mathcal{C}(s)}{\operatorname{argmax}} \ \frac{S(s,a)}{N(s,a)} + c\sqrt{\frac{\log\left(\sum_b N(s,b)\right)}{N(s,a)}}$$

$N(s,a)$: number of visits of (s,a)

$S(s,a)$: number of visits of (s,a) ending with the root player winning



The UCT algorithm

Bandit-Based Monte-Carlo planning : to select a path in the tree, run a bandit algorithm each time a children (next action) needs to be selected

UCT = UCB for Trees [Kocsis and Szepesvári, 2006]

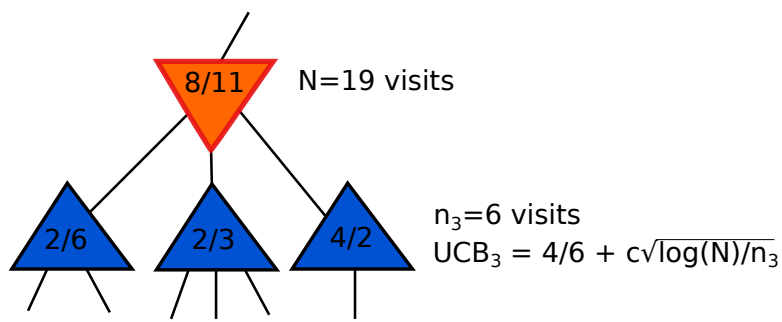
UCT in a Game Tree

In a MIN node s (= adversary move), select an action

$$\underset{a \in \mathcal{C}(s)}{\operatorname{argmin}} \ \frac{S(s,a)}{N(s,a)} - c\sqrt{\frac{\log\left(\sum_b N(s,b)\right)}{N(s,a)}}$$

$N(s,a)$: number of visits of (s,a)

$S(s,a)$: number of visits of (s,a) ending with the root player winning



The UCT algorithm

Bandit-Based Monte-Carlo planning : to select a path in the tree, run a bandit algorithm each time a children (next action) needs to be selected

UCT = UCB for Trees [Kocsis and Szepesvári, 2006]

UCT in a Game Tree

In a MAX node s (= root player move), select an action

$$\underset{a \in \mathcal{C}(s)}{\operatorname{argmax}} \ \frac{S(s,a)}{N(s,a)} + c\sqrt{\frac{\log\left(\sum_b N(s,b)\right)}{N(s,a)}}$$

$N(s,a)$: number of visits of (s,a)
$S(s,a)$: number of visits of (s,a) ending with the root player winning

When a leaf (or some maximal depth) is reached :

- a playout is performed (play the game until the end with a simple heuristic, or produce a random evaluation of the leaf position)
- the outcome of the playout (typically 1/0) is stored in all the nodes visited in the previous trajectory

The UCT algorithm

- first good AIs for Go where based on variants on UCT
- it remains a heuristic (no sample complexity guarantees, parameter c fined-tuned for each application)
- many variants have been proposed

[Browne et al., 2012]

Outline

1. Regret and Sample Complexity in RL
2. Regret minimization in episodic MDPs
   - Optimism for Reinforcement Learning
   - Thompson Sampling for Reinforcement Learning
3. Scalable heuristics inspired by those principles
4. Sample complexity
5. (Bandit-based) Monte-Carlo Tree Search
   - UCB for Trees : UCT
   - From UCT to Alpha Zero

Alpha Zero

AlphaZero learns a good policy by using a MCTS algorithm guided by a neural network

$\neq$ pure play-out based MCTS

Input

A neural network predicting a policy $\boldsymbol{p} \in \Delta(\mathcal{A})$ and a value $v \in \mathbb{R}$ from the current state s : $(\boldsymbol{p}, v) = f_\theta(s)$.

The MCTS algorithm maintains for each visited state/action the counts and cumulated values + a vector of prior action probabilities :

$$\{N(s,a), S(s,a), P(s,a)\}$$

Alpha Zero

AlphaZero learns a good policy by using a MCTS algorithm guided by a neural network

$\neq$ pure play-out based MCTS

Input

A neural network predicting a policy $\boldsymbol{p} \in \Delta(\mathcal{A})$ and a value $v \in \mathbb{R}$ from the current state s : $(\boldsymbol{p}, v) = f_\theta(s)$.

The MCTS algorithm maintains for each visited state/action the counts and cumulated values + a vector of prior action probabilities :

$$\{N(s,a), S(s,a), P(s,a)\}$$

Selection step : in some state s, choose the next action to be

$$\underset{a \in \mathcal{C}(s)}{\operatorname{argmax}} \left[\frac{S(s,a)}{N(s,a)} + c \times P(s,a) \frac{\sqrt{N(s)}}{1+N(s,a)} \right]$$

for some (fine-tuned) constant c.

Alpha Zero

AlphaZero learns a good policy by using a MCTS algorithm guided by a neural network

$\neq$ pure play-out based MCTS

Input

A neural network predicting a policy $\boldsymbol{p} \in \Delta(\mathcal{A})$ and a value $v \in \mathbb{R}$ from the current state s : $(\boldsymbol{p}, v) = f_\theta(s)$.

The MCTS algorithm maintains for each visited state/action the counts and cumulated values + a vector of prior action probabilities :

$$\{N(s,a), S(s,a), P(s,a)\}$$

Expansion step : once a leaf s_L is reached, compute $(\boldsymbol{p}, v) = f_\theta(s_L)$.

- Set v to be the value of the leaf
- For all possible next actions b :
  - ➜ initialize the count $N(s_L, b) = 0$
  - ➜ initialize the prior probability $P(s_L, b) = \boldsymbol{p}_b$ (possibly add some noise)

Alpha Zero

AlphaZero learns a good policy by using a MCTS algorithm guided by a neural network

$\neq$ pure play-out based MCTS

Input

A neural network predicting a policy $\boldsymbol{p} \in \Delta(\mathcal{A})$ and a value $v \in \mathbb{R}$ from the current state s : $(\boldsymbol{p}, v) = f_\theta(s)$.

The MCTS algorithm maintains for each visited state/action the counts and cumulated values + a vector of prior action probabilities :

$$\{N(s,a), S(s,a), P(s,a)\}$$

Back-up step : for all ancestor s_t, a_t in the trajectory that end in leaf s_L,

$$\begin{aligned} N(s_t,a_t) &\leftarrow N(s_t,a_t) + 1 \\ S(s_t,a_t) &\leftarrow S(s_t,a_t) + v \end{aligned}$$

Alpha Zero

AlphaZero learns a good policy by using a MCTS algorithm guided by a neural network

$\neq$ pure play-out based MCTS

Input

A neural network predicting a policy $\boldsymbol{p} \in \Delta(\mathcal{A})$ and a value $v \in \mathbb{R}$ from the current state s : $(\boldsymbol{p}, v) = f_\theta(s)$.

The MCTS algorithm maintains for each visited state/action the counts and cumulated values + a vector of prior action probabilities :

$$\{N(s,a), S(s,a), P(s,a)\}$$

Output of the planning algorithm ? select an action a at random according to

$$\pi(a) = \frac{N(s_0,a)^{1/\tau}}{\sum_b N(s_0,b)^{1/\tau}}$$

for some (fine-tuned) temperature τ.

Training the neural network

- In AlphaGo, f_θ was trained on a database of games played by human
- In AlphaZero, the network is trained using only self-play

[Silver et al., 2016, Silver et al., 2017]

Let θ be the current parameter of the network $(\boldsymbol{p}, v) = f_\theta(s_L)$.

1. generate N games where each player uses MCTS(θ) to select the next action a_t (and output a probability over actions π_t)

$$\mathcal{D} = \bigcup_{i=1}^{\text{Nb games}} \left\{ (s_t, \pi_t, \pm r_{T_i}) \right\}_{i=1}^{T_i}$$

   T_i : length of game i, $r_{T_i} \in \{-1, 0, 1\}$ outcome of game i for one player

2. Based on a sub-sample of $\mathcal{D}$, train the neural network using stochastic gradient descent on the loss function

$$L(s, \boldsymbol{\pi}, z; \boldsymbol{p}, v) = (z - v)^2 - \boldsymbol{\pi}^\top \log(\boldsymbol{p}) + c\|\theta\|^2$$

A nice actor-critic architecture

AlphaZero alternates between

- **The actor** : MCTS(θ)
  generates trajectories guided by the network f_θ but still exploring

➜ act as a policy improvement
($N = 25000$ games played, in which the choice of each move uses MCTS with 1600 simulations)

- **The critic** : neural network f_θ
  updates θ based on trajectories followed by the critic

➜ evaluate the actor's policy

Conclusion : Bandits for RL

Bandits tools are useful for Reinforcement Learning :

- UCB-VI, PSRL : bandit-based exploration for tabular MDPs
- ... that can motivate "deeper" heuristics

Bandit tools lead to big success in Monte-Carlo planning

- ... without proper sample complexity guarantees

➜ Unifying theory and practice is a big challenge in RL !

Azar, M. G., Osband, I., and Munos, R. (2017).

Minimax regret bounds for reinforcement learning.

In *Proceedings of the 34th International Conference on Machine Learning, (ICML)*.

Badia, A. P., Sprechmann, P., Vitvitskyi, A., Guo, Z. D., Piot, B., Kapturowski, S., Tieleman, O., Arjovsky, M., Pritzel, A., Bolt, A., and Blundell, C. (2020).

Never give up : Learning directed exploration strategies.

In *ICLR*.

Bellemare, M. G., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. (2016).

Unifying count-based exploration and intrinsic motivation.

In *Advances in Neural Information Processing Systems (NIPS)*.

Browne, C., Powley, E., Whitehouse, D., Lucas, S., Cowling, P., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S. (2012).

A survey of monte carlo tree search methods.

IEEE Transactions on Computational Intelligence and AI in games,, 4(1) :1–49.

Burda, Y., Edwards, H., Pathak, D., Storkey, A. J., Darrell, T., and Efros, A. A. (2018).

Large-scale study of curiosity-driven learning.
arxiv :1808.04355.

Burda, Y., Edwards, H., Storkey, A. J., and Klimov, O. (2019).
Exploration by random network distillation.
In *ICLR*.

Domingues, O. D., Ménard, P., Kaufmann, E., and Valko, M. (2021).
Episodic reinforcement learning in finite mdps : Minimax lower bounds revisited.
In *Algorithmic Learning Theory (ALT)*.

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., Blundell, C., and Legg, S. (2017).
Noisy networks for exploration.
arXiv :1706.10295.

Jaksch, T., Ortner, R., and Auer, P. (2010).
Near-Optimal regret bounds for reinforcement learning.
Journal of Machine Learning Research, 11 :1563–1600.

Jin, C., Allen-Zhu, Z., Bubeck, S., and Jordan, M. I. (2018).
Is Q-learning provably efficient ?
In *Advances in Neural Information Processing Systems (NeurIPS)*.


Jin, C., Krishnamurthy, A., Simchowitz, M., and Yu, T. (2020).
Reward-free exploration for reinforcement learning.
In *International Conference on Machine Learning*, pages 4870–4879. PMLR.


Kakade, S. (2003).
On the Sample Complexity of Reinforcement Learning.
PhD thesis, University College London.


Kaufmann, E., Ménard, P., Domingues, O. D., Jonsson, A., Leurent, E., and Valko, M. (2021).
Adaptive reward-free exploration.
In *Algorithmic Learning Theory (ALT)*.


Kocsis, L. and Szepesvári, C. (2006).
Bandit based monte-carlo planning.
In *Proceedings of the 17th European Conference on Machine Learning*, ECML'06, pages 282–293, Berlin, Heidelberg. Springer-Verlag.


Ménard, P., Domingues, O. D., Jonsson, A., Kaufmann, E., Leurent, E., and Valko, M. (2021).
Fast active learning for pure exploration in reinforcement learning.
In *International Conference on Machine Learning (ICML)*.


Osband, I., Blundell, C., Pritzel, A., and Roy, B. V. (2016).
Deep exploration via bootstrapped DQN.
In *Advances in Neural Information Processing Systems (NIPS)*.


Osband, I., Van Roy, B., and Russo, D. (2013).
(More) Efficient Reinforcement Learning Via Posterior Sampling.
In *Advances in Neural Information Processing Systems*.


Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016).
Mastering the game of go with deep neural networks and tree search.
Nature, 529 :484–489.


Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., van den Driessche, G., Graepel, T., and Hassabis, D. (2017).
Mastering the game of go without human knowledge.
Nature, 550 :354–.


Simchowitz, M. and Jamieson, K. G. (2019).

Non-asymptotic gap-dependent regret bounds for tabular mdps.
In *Advances in Neural Information Processing Systems (NeurIPS)*.

Strens, M. (2000).
A Bayesian Framework for Reinforcement Learning.
In *ICML*.

Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, X., Duan, Y., Schulman, J., Turck, F. D., and Abbeel, P. (2017).
#exploration : A study of count-based exploration for deep reinforcement learning.
In *Advances in Neural Information Processing Systems (NIPS)*.

Tiapkin, D., Belomestny, D., Calandriello, D., Moulines, E., Munos, R., Naumov, A., Rowland, M., Valko, M., and Ménard, P. (2022).
Optimistic posterior sampling for reinforcement learning with few samples and tight guarantees.
In *NeurIPS*.

Wagenmaker, A. J., Simchowitz, M., and Jamieson, K. (2022).
Beyond no regret : Instance-dependent PAC reinforcement learning.
In *Conference On Learning Theory (COLT)*.