Reinforcement Learning Approximation and Deep Learning

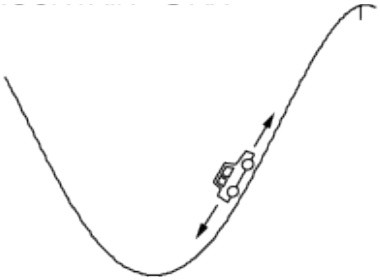
Odalric-Ambrym Maillard

2021-2022

Inria Scool

Continuous space and Function approximation

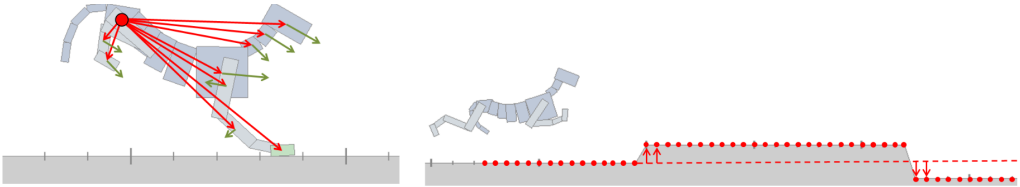
Example: Mountain-car



States: Position $\times$ Speed of car (continuous).
Actions: Acceleration of car.
Transitions: Physics.
Rewards: 1 if reach top of mountain, 0 else.
Optimal policy: First go left to get momentum, then go right.

Continuous state-action space. For other classical tasks, check `https://gym.openai.com/envs/#classic_control`.

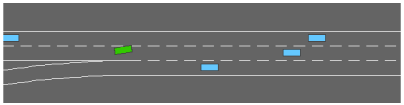
Example: Motion-planning



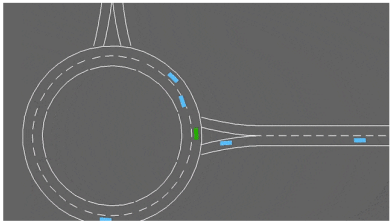
States: relative positions (red points) and velocities.
Actions: acceleration at each joint
Rewards: not falling (height of anchor point).

Example: High-way

Controlling an ego-vehicle in various road environments
`https://github.com/eleurent/highway-env`



The merge-v0 environment.



The roundabout-v0 environment.

Actions: acceleration, changing lane, etc.
Rewards: no collision, speed, etc.

Study safety and robustness (here with partially known behavior of other vehicles).
`http://edouardleurent.com/publication/phd-thesis/`

DP requires:

- the state and action spaces to be small enough;
- the model to be known.

DP requires:

- the state and action spaces to be small enough;
- the model to be known.

Unfortunately:

- the state space can be too large (even continuous) for the value function to be represented exactly,

$$V_\theta(s) = \theta^\top \varphi(s) = \sum_{i=1}^{d} \theta_i \varphi_i(s) \tag{1}$$

DP requires:

- the state and action spaces to be small enough;
- the model to be known.

Unfortunately:

- the state space can be too large (even continuous) for the value function to be represented exactly,

$$V_\theta(s) = \theta^\top \varphi(s) = \sum_{i=1}^{d} \theta_i \varphi_i(s) \tag{1}$$

- the model might be unknown and one has to rely on a dataset

$$\mathcal{D} = \{(s_i, a_i, r_i, s_i')_{1 \leqslant i \leqslant n}\}. \tag{2}$$

  - the dataset can be obtained in multiple ways;

- Computing a **greedy policy** requires **knowing the model**:

$$\pi \in \mathcal{G}V \Leftrightarrow \forall s \in \mathcal{S}, \quad \pi(s) \in \operatorname*{argmax}_{a \in \mathcal{A}} \left(m_a(s) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V(s') \right).$$

- Computing a **greedy policy** requires **knowing the model**:

$$\pi \in \mathcal{G}V \Leftrightarrow \forall s \in \mathcal{S}, \quad \pi(s) \in \operatorname*{argmax}_{a \in \mathcal{A}} \left(m_a(s) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) V(s') \right).$$

- Sampling the optimality operator?
  - Optimality operator:

$$(\mathcal{T}V)(s) = \max_{a \in \mathcal{A}} \mathbb{E}_{s' \sim P(.|s,a)}[m(s,a) + \gamma V(s')];$$

  - with $s'_{i,a} \sim P(.|s_i, a)$, a possible sampled operator:

$$(\widehat{\mathcal{T}}V)(s_i) = \max_{a \in \mathcal{A}} \left(r(s_i,a) + \gamma V(s'_{i,a}) \right);$$

  - it is biased: $\mathbb{E}[(\widehat{\mathcal{T}}V(s_i)|s_i] \neq (\mathcal{T}V)(s_i)$.

- state-action value function (aka Q-function, quality function)

$$Q_\pi(s,a) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r(S_t, A_t) | S_0 = s, A_0 = a, S_{t+1} \sim \mathbf{p}(.|S_t, A_t), A_{t+1} = \pi(S_{t+1})\Big].$$

- state-action value function (aka Q-function, quality function)

$$Q_\pi(s,a) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r(S_t, A_t) | S_0 = s, A_0 = a, S_{t+1} \sim \mathbf{p}(.|S_t, A_t), A_{t+1} = \pi(S_{t+1})\Big].$$

- Bellman evaluation operator $T_\pi : \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \to \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$
  - definition: $[T_\pi Q](s,a) = r(s,a) + \gamma \sum_{s'\in\mathcal{S}} \mathbf{p}(s'|s,a) Q(s', \pi(s'))$;
  - Q_π is its unique fixed point: $T_\pi Q_\pi = Q_\pi$;
  - link to v_π: $v_\pi(s) = Q_\pi(s, \pi(s))$.

Recap: the Q-function

- state-action value function (aka Q-function, quality function)

$$Q_\pi(s,a) = \mathbb{E}\Big[\sum_{t=0}^{\infty} \gamma^t r(S_t, A_t) | S_0 = s, A_0 = a, S_{t+1} \sim \mathbf{p}(.|S_t, A_t), A_{t+1} = \pi(S_{t+1})\Big].$$

- Bellman evaluation operator $T_\pi : \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \to \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$
  - definition: $[T_\pi Q](s,a) = r(s,a) + \gamma \sum_{s'\in\mathcal{S}} \mathbf{p}(s'|s,a) Q(s', \pi(s'))$;
  - Q_π is its unique fixed point: $T_\pi Q_\pi = Q_\pi$;
  - link to v_π: $v_\pi(s) = Q_\pi(s, \pi(s))$.
- Bellman optimality operator $T_* : \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \to \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$
  - definition: $[T_* Q](s,a) = r(s,a) + \gamma \sum_{s'\in\mathcal{S}} \mathbf{p}(s'|s,a) \max_{a'\in\mathcal{A}} Q(s', a')$;
  - Q_* is its unique fixed point: $Q_* = T_* Q_*$;
  - link to v_*: $v_*(s) = \max_{a\in\mathcal{A}} Q_*(s,a)$.

- Allows acting greedily:
  - resp to $V^\pi = Q_\pi(s, \pi(s))$:

$$\pi' \in \mathcal{G}(V^\pi) \quad \Leftrightarrow \forall s \in \mathcal{S}, \quad \pi'(s) \in \operatorname*{argmax}_{a \in \mathcal{A}} \left(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} \mathbf{p}(s'|s,a) V^\pi(s') \right)$$
$$\Leftrightarrow \forall s \in \mathcal{S}, \quad \pi'(s) \in \operatorname*{argmax}_{a \in \mathcal{A}} Q_\pi(s,a).$$

  - resp. to $V^\star$:

$$\star(s) \in \operatorname*{argmax}_{a \in \mathcal{A}} Q^\star(s,a).$$

  - resp. to any $Q \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}$:

$$\forall Q \in \mathbb{R}^{\mathcal{S} \times \mathcal{A}}, \quad \pi \in \mathcal{G}(Q) \Leftrightarrow \forall s \in \mathcal{S}, \quad \pi(s) \in \operatorname*{argmax}_{a \in \mathcal{A}} Q(s,a).$$

- Allows sampling easily the related operators
  - recall the dataset

$$\mathcal{D} = \{(s_i, a_i, r_i, s_i')_{1 \leqslant i \leqslant n}\}.$$

  - sampled Bellman evaluation operator

$$[\widehat{T}_\pi Q](s_i, a_i) = r_i + \gamma Q(s_i', \pi(s_i'));$$

  - sampled Bellman optimality operator

$$[\widehat{T}_* Q](s_i, a_i) = r_i + \gamma \max_{a' \in \mathcal{A}} Q(s_i', a').$$

$\Longrightarrow$ Focus on this key function

- Allows sampling easily the related operators
  - recall the dataset

$$\mathcal{D} = \{(s_i, a_i, r_i, s_i')_{1 \leqslant i \leqslant n}\}.$$

  - sampled Bellman evaluation operator

$$[\widehat{T}_\pi Q](s_i, a_i) = r_i + \gamma Q(s_i', \pi(s_i'));$$

  - sampled Bellman optimality operator

$$[\widehat{T}_* Q](s_i, a_i) = r_i + \gamma \max_{a' \in \mathcal{A}} Q(s_i', a').$$

  $\Longrightarrow$ Focus on this key function
- Features for the Q-function:
  - linear parametrization of the Q-function $Q \in \mathcal{F}$:

$$\mathcal{F} = \{Q_\theta(s, a) = \theta^\top \varphi(s, a), \theta \in \mathbb{R}^d\}$$

Continuous space and Function approximation

Approximate value iteration

Approximate policy iteration

Policy representation

When $\mathcal{S}, \mathcal{A}$ is huge, we may want to represent in Function spaces: m, p, V, Q or π.
1) Parameterize **Q function**:

$$\mathcal{F} = \{Q_\theta(s,a) = \theta^\top \varphi(s,a), \theta \in \mathbb{R}^d\}$$

When $\mathcal{S}, \mathcal{A}$ is huge, we may want to represent in Function spaces: m, p, V, Q or π.
1) Parameterize **Q function**:

$$\mathcal{F} = \{Q_\theta(s,a) = \theta^\top \varphi(s,a), \theta \in \mathbb{R}^d\}$$

For VI, difficulty is to find $Q_\theta \simeq \mathcal{T} Q_\theta$: **Approximate VI**
For PI, difficulty is to find $Q_\theta \simeq \mathcal{T}_\pi Q_\theta$: **Approximate PI**
Different strategies:

- Minimize $\|Q_\theta - T_\pi Q_\theta\|$: **Bellman residual minimization**.
- Solve $Q_\theta = \Pi_{\mathcal{F}} T_\pi Q_\theta$: **Least-squares Temporal Differences**.

When $\mathcal{S}, \mathcal{A}$ is huge, we may want to represent in Function spaces: m, p, V, Q or π.
1) Parameterize Q **function**:

$$\mathcal{F} = \{Q_\theta(s,a) = \theta^\top \varphi(s,a), \theta \in \mathbb{R}^d\}$$

For VI, difficulty is to find $Q_\theta \simeq \mathcal{T} Q_\theta$: **Approximate VI**
For PI, difficulty is to find $Q_\theta \simeq \mathcal{T}_\pi Q_\theta$: **Approximate PI**
Different strategies:

- Minimize $\|Q_\theta - T_\pi Q_\theta\|$: **Bellman residual minimization**.
- Solve $Q_\theta = \Pi_\mathcal{F} T_\pi Q_\theta$: **Least-squares Temporal Differences**.

2) Parameterize π **function**:

- Solve $\underset{\pi \in \mathcal{F}}{\operatorname{argmin}}(\max_a Q(\cdot, a) - Q(\cdot, \pi))$: **Direct Policy Search**

- Difficulty to apply value iteration ($Q_{k+1} = \mathcal{T} Q_k$):
  - Max operator $\mathcal{T}$ is **unknown**;
  - $Q_k \in \mathcal{F}$ may **not** imply $\mathcal{T} Q_k \in \mathcal{F}$.

- Difficulty to apply value iteration ($Q_{k+1} = \mathcal{T} Q_k$):
  - Max operator $\mathcal{T}$ is **unknown**;
  - $Q_k \in \mathcal{F}$ may **not** imply $\mathcal{T} Q_k \in \mathcal{F}$.
- Example of naive **Approximate value iteration**:
  - writing $Q_k = Q_{\theta_k}$, sampled operator:

$$(\widehat{\mathcal{T}} Q_k)(s_i, a_i) = r_i + \gamma \max_{a' \in \mathcal{A}} Q_k(s_i', a')$$

  - search for $Q \in \mathcal{F}$ being the closest to $\widehat{\mathcal{T}} Q_k$:

$$Q_{k+1} \in \operatorname*{argmin}_{Q_\theta \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \left(Q_\theta(s_i, a_i) - (\widehat{\mathcal{T}} Q_k)(s_i, a_i) \right)^2 .$$

  - summary: $Q_{k+1} = \Pi_{\mathcal{F}} \widehat{\mathcal{T}} Q_k$.

Algorithm 1 Approximate value iteration

Input: A dataset $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)_{1 \leqslant i \leqslant n}\}$, the number K of iterations, a function approximator, an initial state-action value function Q_0.

1: **for** $k = 0$ **to** K **do**

2: Apply the sampled Bellman operator to function Q_k:

$$(\widehat{T} Q_k)(s_i, a_i) = r_i + \gamma \max_{a' \in \mathcal{A}} Q_k(s'_i, a').$$

3: Solve the regression problem with inputs (s_i, a_i) and outputs $(\widehat{T} Q_k)(s_i, a_i)$ to get the Q-function Q_{k+1}

4: **end for**

5: **return** The greedy policy $\pi_{K+1} \in \mathcal{G}(Q_{K+1})$:

$$\forall s \in \mathcal{S}, \quad \pi_{K+1} \in \operatorname*{argmax}_{a \in \mathcal{A}} Q_{K+1}(s, a).$$

- Approximate value iteration:

$$\mathbf{Q_{k+1}} = \mathbb{A}\widehat{\mathbf{T}}\mathbf{Q_k}.$$

- $\mathbb{A}$ is an abstract approximation operator

- Approximate value iteration:

$$\mathbf{Q_{k+1}} = \mathbb{A}\widehat{\mathbf{T}}\mathbf{Q_k}.$$

- $\mathbb{A}$ is an abstract approximation operator
- Problem: $\mathbb{A}\widehat{T}$ should be a **contraction**!

- Approximate value iteration:

$$\mathbf{Q_{k+1}} = \mathbb{A}\widehat{\mathbf{T}}\mathbf{Q_k}.$$

- $\mathbb{A}$ is an abstract approximation operator
- Problem: $\mathbb{A}\widehat{T}$ should be a **contraction**!
  - Otherwise divergence can (will) occur;
  - Not the case for $\Pi\widehat{T}$...
  - Works if "**averagers**" are used for function approximation, such as
    - ensemble of trees
    - notably extremely randomized forests: **fitted-Q**
    - kernel averagers (Nadaraya-Watson)

Working with AVI is tricky due to **max Bellman operator** $\mathcal{T}$.

- Idea: Instead of considering $\mathcal{T}$, consider $\mathcal{T}_\pi$ for a fixed π.

  - Policy iteration:
    1. policy evaluation: solve the fixed-point equation $Q_{\pi_k} = T_{\pi_k} Q_{\pi_k}$;
    2. policy improvement: compute the greedy policy $\pi_{k+1} = \mathcal{G}(Q_{\pi_k})$.
  - **Approximate policy iteration**:
    1. approximate policy evaluation: find a function $Q_k \in \mathcal{F}$ such that $Q_k \approx T_{\pi_k} Q_k$;
    2. policy improvement: compute the greedy policy $\pi_{k+1} = \mathcal{G}(Q_{\pi_k})$.

Algorithm 2 Approximate policy iteration (generic form)

Input: An initial $\pi_0 \in \mathcal{A}^{\mathcal{S}}$ (possibly an initial Q_0 and $\pi_0 \in \mathcal{G}(Q_0)$), number of iterations K

1: **for** $k = 0$ **to** K **do**
2: approximate policy evaluation: find $Q_k \in \mathcal{F}$ such that $Q_k \approx T_{\pi_k} Q_k$.
3: policy improvement: $\pi_{k+1} \in \mathcal{G}(Q_k)$.
4: **end for**
5: **return** the policy π_{K+1}

▷ How to find an **approximate fixed point** of T_π, that is a function $Q_\theta \in \mathcal{F}$ such that $Q_\theta \approx T_\pi Q_\theta$?

- Assume that Q_π is known: simply a **regression** problem. E.g. linear least-squares:

$$\min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left(Q_\pi(s_i, a_i) - Q_\theta(s_i, a_i) \right)^2 .$$

- Assume that Q_π is known: simply a **regression** problem. E.g. linear least-squares:

$$\min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left(Q_\pi(s_i, a_i) - Q_\theta(s_i, a_i) \right)^2 .$$

- Q_π is (obviously) unknown...
- Idea 1: **Monte carlo rollout**?
  - sample a full trajectory starting in s_i where action a_i is chosen first, all subsequent states being sampled according to the system dynamics and actions chosen according to π;
  - let q_i be the associated discounted cumulative reward;
  - unbiased estimate: $\mathbb{E}[q_i | s_i, a_i] = Q_\pi(s_i, a_i)$

- Assume that Q_π is known: simply a **regression** problem. E.g. linear least-squares:

$$\min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left(Q_\pi(s_i, a_i) - Q_\theta(s_i, a_i) \right)^2 .$$

  - Q_π is (obviously) unknown...
  - Idea 1: **Monte carlo rollout**?
    - sample a full trajectory starting in s_i where action a_i is chosen first, all subsequent states being sampled according to the system dynamics and actions chosen according to π;
    - let q_i be the associated discounted cumulative reward;
    - unbiased estimate: $\mathbb{E}[q_i | s_i, a_i] = Q_\pi(s_i, a_i)$
    - replace $Q_\pi(s_i, a_i)$ by the unbiased estimate q_i:

$$\min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left(q_i - Q_\theta(s_i, a_i) \right)^2 .$$

- Drawbacks: requires a simulator, rollouts can be quite noisy.

- Idea 2: **minimize the residual** $\|Q_\theta - T_\pi Q_\theta\|$ for some norm.
- With an ℓ_2-loss, parametric representation and sampled operator:

$$\min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left((\widehat{T}_\pi Q_\theta)(s_i, a_i) - Q_\theta(s_i, a_i) \right)^2$$

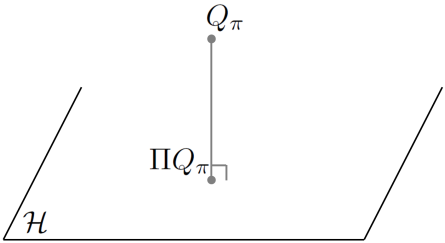
$$= \min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left(r_i + \gamma Q_\theta(s_i', \pi(s_i')) - Q_\theta(s_i, a_i) \right)^2 .$$

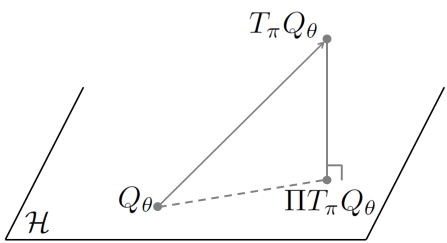
- Idea 2: **minimize the residual** $\|Q_\theta - T_\pi Q_\theta\|$ for some norm.
- With an ℓ_2-loss, parametric representation and sampled operator:

$$\min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left((\widehat{T}_\pi Q_\theta)(s_i, a_i) - Q_\theta(s_i, a_i) \right)^2$$
$$= \min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left(r_i + \gamma Q_\theta(s_i', \pi(s_i')) - Q_\theta(s_i, a_i) \right)^2 .$$

- However, there is a **bias problem**:

$$\begin{aligned} &\mathbb{E}[((\widehat{T}_\pi Q_\theta)(s_i, a_i) - Q_\theta(s_i, a_i))^2 | s_i, a_i] \\ &= ((\mathcal{T}_\pi Q_\theta)(s_i, a_i) - Q_\theta(s_i, a_i))^2 + \mathbb{V}((\widehat{T}_\pi Q_\theta)(s_i, a_i) | s_i, a_i) \\ &\neq ((\mathcal{T}_\pi Q_\theta)(s_i, a_i) - Q_\theta(s_i, a_i))^2. \end{aligned}$$





- Idea 3: Solve $Q_\theta = \Pi T_\pi Q_\theta$. as a **nested optimization** problem:

$$w_\theta = \operatorname*{argmin}_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (r_i + \gamma Q_\theta(s_i', \pi(s_i')) - Q_w(s_i, a_i))^2 \tag{3}$$

$$\theta_n = \operatorname*{argmin}_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (Q_\theta(s_i, a_i) - Q_{w_\theta}(s_i, a_i))^2. \tag{4}$$

- (3) is ordinary least-squares in w. Solution:

$$w_\theta = \left(\sum_{i=1}^{n} \varphi(s_i, a_i) \varphi(s_i, a_i)^\top \right)^{-1} \sum_{i=1}^{n} \varphi(s_i, a_i)(r_i + \gamma \theta^\top \varphi(s_i', \pi(s_i'))).$$

- (3) is ordinary least-squares in w. Solution:

$$w_\theta = \left(\sum_{i=1}^{n} \varphi(s_i, a_i) \varphi(s_i, a_i)^\top \right)^{-1} \sum_{i=1}^{n} \varphi(s_i, a_i)(r_i + \gamma \theta^\top \varphi(s_i', \pi(s_i'))).$$

- (4) is minimized for $\theta = w_\theta$:

$$\begin{aligned} \theta_n = w_{\theta_n} \Leftrightarrow \theta_n &= \left(\sum_{i=1}^{n} \varphi(s_i, a_i) \varphi(s_i, a_i)^\top \right)^{-1} \sum_{i=1}^{n} \varphi(s_i, a_i)(r_i + \gamma \theta_n^\top \varphi(s_i', \pi(s_i'))) \\ \Leftrightarrow \theta_n &= \left(\sum_{i=1}^{n} \varphi(s_i, a_i) \left(\varphi(s_i, a_i) - \gamma \varphi(s_i', \pi(s_i')) \right)^\top \right)^{-1} \sum_{i=1}^{n} \varphi(s_i, a_i) r_i. \end{aligned}$$

- This API strategy with LSTD is called **LSPI (Least-Squares Policy Iteration)**.

Algorithm 3 Least-squares policy iteration

Input: An initial $\pi_0 \in \mathcal{A}^{\mathcal{S}}$ (possibly an initial Q_0 and $\pi_0 \in \mathcal{G}(Q_0)$), number of iterations K

1: **for** $k = 0$ **to** K **do**

2: approximate policy evaluation:

$$\theta_k = \left(\sum_{i=1}^{n} \varphi(s_i, a_i) \left(\varphi(s_i, a_i) - \gamma \varphi(s_i', \pi_k(s_i')) \right)^\top \right)^{-1} \sum_{i=1}^{n} \varphi(s_i, a_i) r_i.$$

3: policy improvement:

$$\pi_{k+1} \in \mathcal{G}(Q_{\theta_k}).$$

4: **end for**

5: **return** the policy π_{K+1}

Proposition (Williams and Baird, 93)

$$\|V_\pi - \widehat{V}_{BR}\|_\infty \leqslant \frac{1+\gamma}{1-\gamma}\|V_\pi - \widehat{V}_{\text{best}}\|_\infty$$

Proposition (Tsitsiklis and Van Roy, 97)

Let ρ be the stationary sistribution of P_π, then

$$\|V_\pi - \widehat{V}_{LSTD}\|_{2,\rho} \leqslant \frac{1}{1-\gamma}\|V_\pi - \widehat{V}_{\text{best}}\|_{2,\rho}$$

Approximate PI: $\pi_k = \mathcal{G}[v_{k-1}]$ and $v_k = v_{\pi_k} + \varepsilon_k$

Proposition

Assume $\|\varepsilon_k\|_\infty \leqslant \varepsilon$. The loss due to running π_k instead of $\pi_\star$ satisfies

$$\limsup_{k\to\infty} \|V_\star - v_{\pi_k}\|_\infty \leqslant \frac{2\gamma}{(1-\gamma)^2}\varepsilon\,.$$

- Idea: represent the **policy** instead of **Q-function**

- Idea: represent the **policy** instead of **Q-function**
- Motivation: might be easier to learn (we control what it is)
- Let $\mathcal{F} \subset \mathcal{A}^{\mathcal{S}}$ be an hypothesis space of policies. At iteration k, assume $Q_{\pi_k}(s_i, a)$ are known, and solve the **classification problem**

$$\pi_{k+1} \in \operatorname*{argmin}_{\pi \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \left(\max_{a \in \mathcal{A}} Q_{\pi_k}(s_i, a) - Q_{\pi_k}(s_i, \pi(s_i)) \right).$$

- Practically, replace $Q_{\pi_k}(s_i, a)$ by a Monte Carlo rollout.
- Often called **DPI** for **Direct Policy Iteration**.

"*The more applied you go, the stronger theory you need*"

Merci

odalricambrym.maillard@inria.fr
odalricambrymmaillard.wordpress.com