Reinforcement Learning (Machine Learning, SIR)

Matthieu Geist (CentraleSupélec)
matthieu.geist@centralesupelec.fr

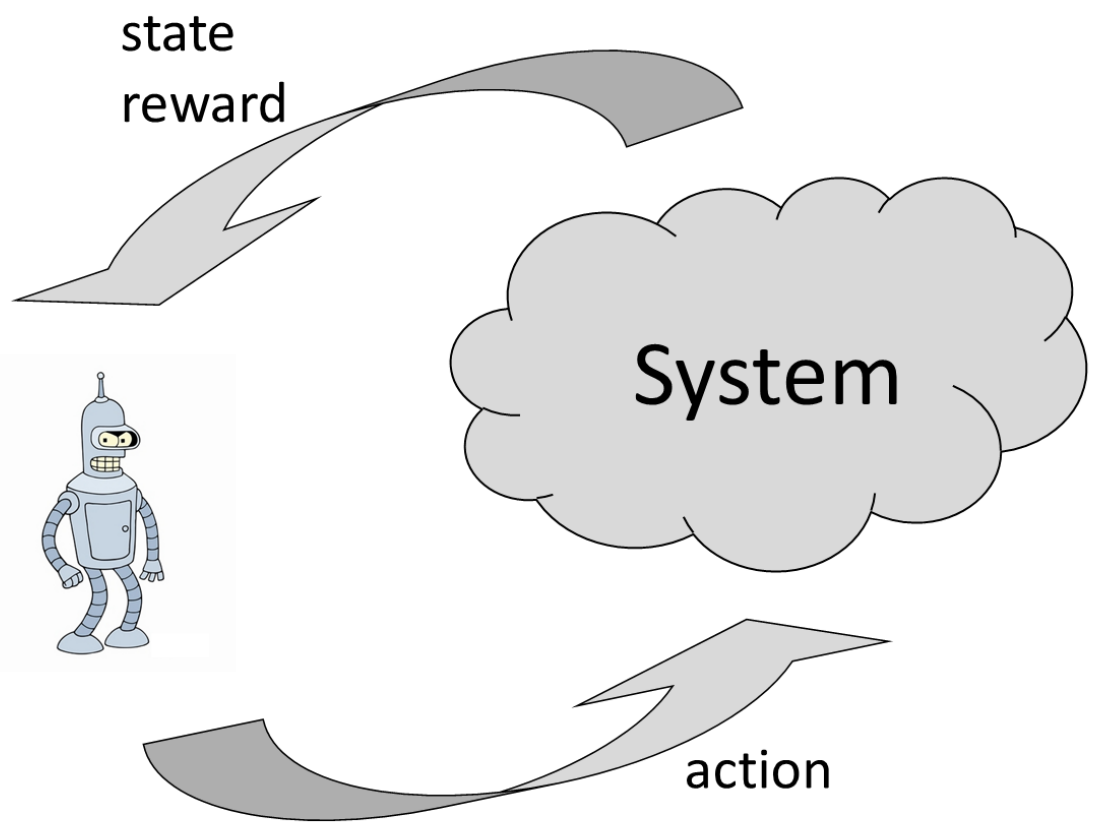
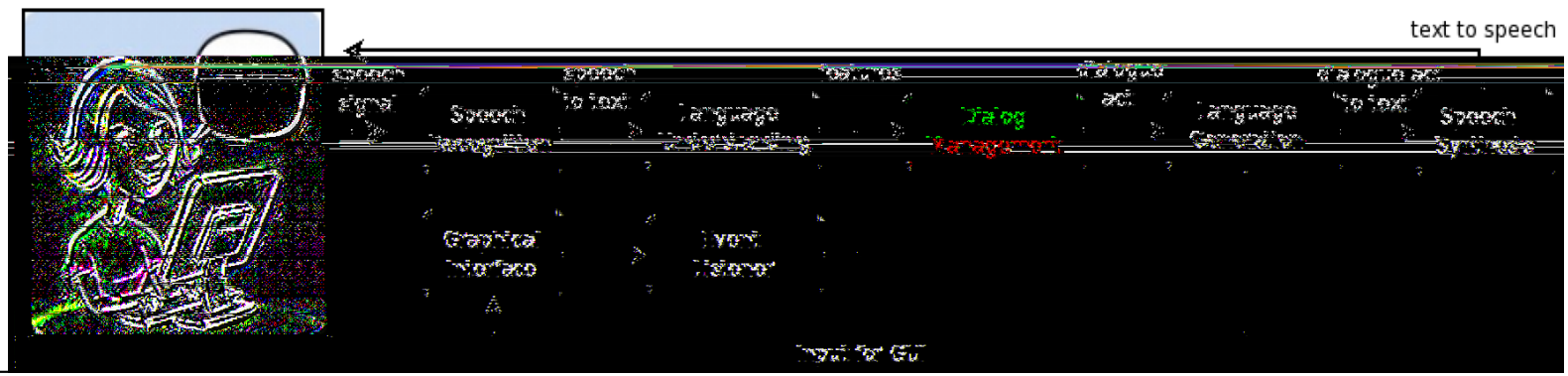


Figure : The perception-action cycle in reinforcement learning.

Applications

- playing games (Backgammon, Go, Tetris, Atari...)
- robotics
- autonomous acrobatic helicopter control[1]
- operation research (pricing, vehicle routing...)
- human computer interactions (dialogue management, e-learning...)
- virtually any control problem[2]



[1] http://heli.stanford.edu/

[2] An old list: http://umichrl.pbworks.com/w/page/7597597/Successes_of_Reinforcement_Learning

A Markov Decision Process (MDP) is a tuple $\{\mathcal{S}, \mathcal{A}, P, r, \gamma\}$ where:

- $\mathcal{S}$ is the (finite) state space;
- $\mathcal{A}$ is the (finite) action space;
- $P \in \Delta_{\mathcal{S}}^{\mathcal{S}\times\mathcal{A}}$ is the Markovian transition kernel. The term $P(s'|s,a)$ denotes the probability of transiting in state s' given that action a was chosen in state s;
- $r \in \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$ is the reward function, it associates the reward $r(s,a)$ for taking action a in state s. The reward function is assumed to be uniformly bounded;
- $\gamma \in (0,1)$ is a discount factor that favors shorter term rewards (usually set to a value close to 1).

- Policy:
  - $\pi \in \mathcal{A}^{\mathcal{S}}$;
  - in state s, an agent applying policy π chooses the action $\pi(s)$
- Value function (quantify the quality of a policy):

$$v_\pi(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(S_t, \pi(S_t)) | S_0 = s, S_{t+1} \sim P(.|S_t, \pi(S_t))].$$

- Comparing policies (partial ordering):

$$\pi_1 \geq \pi_2 \Leftrightarrow \forall s \in \mathcal{S}, \quad v_{\pi_1}(s) \geq v_{\pi_2}(s).$$

- Optimal policy:

$$\pi_* \in \underset{\pi \in \mathcal{A}^{\mathcal{S}}}{\operatorname{argmax}}\, v_\pi.$$

Rewriting the Bellman equation

$$v_\pi(s) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(S_t, \pi(S_t)) | S_0 = s, S_{t+1} \sim P(.|S_t, \pi(S_t))]$$

$$= r(s, \pi(s)) + \mathbb{E}[\sum_{t=1}^{\infty} \gamma^t r(S_t, \pi(S_t)) | S_0 = s, S_{t+1} \sim P(.|S_t, \pi(S_t))]$$

$$= r(s, \pi(s)) + \gamma \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(S_{t+1}, \pi(S_{t+1})) | S_0 = s, S_{t+1} \sim P(.|S_t, \pi(S_t))]$$

$$\Leftrightarrow v_\pi(s) = r(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, \pi(s)) v_\pi(s').$$

Rewriting the Bellman equation (cont.)

- Define the stochastic matrix $P_\pi \in \mathbb{R}^{\mathcal{S}\times\mathcal{S}}$ and the vector $r_\pi \in \mathbb{R}^{\mathcal{S}}$ as

$$P_\pi = \big(P(s'|s,\pi(s))\big)_{s,s'\in\mathcal{S}} \text{ and } r_\pi = \big(r(s,\pi(s))\big)_{s\in\mathcal{S}}\,.$$

- Using these notations, we have:

$$v_\pi = r_\pi + \gamma P_\pi v_\pi \Leftrightarrow v_\pi = (I - \gamma P_\pi)^{-1} r_\pi.$$

Bellman evaluation operator

Define the Bellman evaluation operator $T_\pi : \mathbb{R}^{\mathcal{S}} \to \mathbb{R}^{\mathcal{S}}$ as

$$\forall v \in \mathbb{R}^{\mathcal{S}}, \quad T_\pi v = r_\pi + \gamma P_\pi v,$$

or equivalently componentwise

$$\forall s \in \mathcal{S}, \quad [T_\pi v](s) = r(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, \pi(s)) v(s').$$

T_π is a contraction (supremum norm) and v_π is its unique fixed point:

$$v_\pi = T_\pi v_\pi.$$

Optimal value function and policies

Assume that $v_* = v_{\pi_*}$ is known, an optimal policy should be greedy resp. to v_*:

$$\pi_*(s) \in \operatorname*{argmax}_{a \in \mathcal{A}} \left(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) v_*(s') \right).$$

Characterizing v_*:

$$\forall s \in \mathcal{S}, \quad v_*(s) = \max_{a \in \mathcal{A}} \left(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) v_*(s') \right).$$

Bellman optimality operator

Define the Bellman optimality operator $T_* : \mathbb{R}^{\mathcal{S}} \to \mathbb{R}^{\mathcal{S}}$ as

$$\forall v \in \mathbb{R}^{\mathcal{S}}, \quad T_* v = \max_{\pi \in \mathcal{A}^{\mathcal{S}}} \left(r_\pi + \gamma P_\pi v \right),$$

or equivalently componentwise

$$\forall s \in \mathcal{S}, \quad [T_* v](s) = \max_{a \in \mathcal{A}} \left(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) v(s') \right).$$

T_* is a contraction (supremum norm) and v_* is its unique fixed point:

$$v_* = T_* v_*.$$

- DP: solve an MDP when the model is known.
- In practice, the model is unknown and one has to rely on data.
- Even so, related *learning* methods are often based on DP.

v_* is the solution of the following linear program:

$$\min_{v \in \mathbb{R}^{\mathcal{S}}} \mathbf{1}^\top v$$
$$\text{subject to } v \geq T_* v.$$

Proof.

$$v \geq T_* v \Rightarrow v \geq v_* \Rightarrow \mathbf{1}^\top v \geq \mathbf{1}^\top v_*$$

and

$$v_* = T_* v_*.$$

Algorithm 1 Linear programming

1: Solve

$$\min_{v \in \mathbb{R}^{\mathcal{S}}} \sum_{s \in \mathcal{S}} v(s)$$

$$\text{subject to } v(s) \geq r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) v(s'), \quad \forall s \in \mathcal{S}, \forall a \in \mathcal{A}$$

and get v_*.

2: **return** the policy π_* defined as

$$\pi_*(s) \in \operatorname*{argmax}_{a \in \mathcal{A}} \left(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) v_*(s') \right).$$

- T_* is a contraction: $\forall u, v \in \mathbb{R}^{\mathcal{S}}$, $\|T_* u - T_* v\|_\infty \leq \|u - v\|_\infty$;
- v_* is its unique fixed point: $T_* v_* = v_*$;
- Banach fixed-point theorem: for any v_0, the sequence

$$v_{k+1} = T_* v_k$$

converges to v_*;

- natural stopping criterion: $\|v_{k+1} - v_k\|_\infty \leq \epsilon$;
- output a greedy policy (resp. to v_k), $\pi_k \in \mathcal{G}(v_k)$

$$\pi \in \mathcal{G}(v) \Leftrightarrow T_\pi v = T_* v$$

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

Algorithm 2 Value iteration

Require: An initial $v_0 \in \mathbb{R}^{\mathcal{S}}$, a stopping criterion ϵ

1: $k = 0$
2: **repeat**
3: **for all** $s \in \mathcal{S}$ **do**
4:

$$v_{k+1}(s) = \max_{a \in \mathcal{A}} \left(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) v_k(s') \right)$$

5: **end for**
6: $k \leftarrow k + 1$
7: **until** $\|v_{k+1} - v_k\|_\infty \leq \epsilon$
8: **return** a policy $\pi_k \in \mathcal{G}(v_k)$:

$$\pi_k(s) \in \operatorname*{argmax}_{a \in \mathcal{A}} \left(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) v_k(s') \right).$$

Quality of the obtained solution?

- Stop iterations if
$$\|v_{k+1} - v_k\|_\infty \leq \epsilon.$$
- Guaranty on the function v_k:
$$\|v_* - v_k\|_\infty \leq \frac{1}{1-\gamma}\epsilon.$$
- Guaranty on the policy π_k:
$$\|v_* - v_{\pi_k}\|_\infty \leq \frac{2\gamma}{(1-\gamma)^2}\epsilon.$$

- Let π be any policy and v_π its value function.
- Let π' be greedy resp. to v_π, $\pi' \in \mathcal{G}(v_\pi)$:

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

- π' is a better policy than π:

$$v_{\pi'} \geq v_\pi$$

- This suggests the following algorithmic scheme, iterate:
  1. policy evaluation: solve $T_{\pi_k} v_{\pi_k} = v_{\pi_k}$;
  2. policy improvement: compute $\pi_{k+1} \in \mathcal{G}(v_{\pi_k})$.

Algorithm 3 Policy iteration

Require: An initial $\pi_0 \in \mathcal{A}^{\mathcal{S}}$

1: $k = 0$
2: **repeat**
3: solve (policy evaluation)

$$v_k(s) = r(s, \pi_k(s)) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, \pi_k(s)) v_k(s'), \quad \forall s \in \mathcal{S}.$$

4: Compute (policy improvement)

$$\pi_{k+1}(s) \in \operatorname*{argmax}_{a \in \mathcal{A}} \left(r(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) v_k(s') \right).$$

5: $k \leftarrow k + 1$
6: **until** $v_{k+1} = v_k$
7: **return** the policy $\pi_{k+1} = \pi_*$

- 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,

$$v(s) = \top(s) = \sum_{i=1}^{d} i \, i(s)$$

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

$$\mathcal{D} = \{(s_i; a_i; r_i; s_i')_{1 \leq i \leq n}\}:$$

    - the dataset can be obtained in multiple ways;
    - the evaluation operator can be sampled (assume here $a_i = (s_i)$),

$$[\hat{T} v](s_i) = r_i + v(s_i')$$

      is unbiased:
      $\mathbb{E}[[\hat{T} v](s_i)|s_i] = \mathbb{E}_{S' \sim P(:|s_i;a_i)}[r_i + v(S')] = [T v](s_i).$

Problems with value functions

- Computing a greedy policy requires knowing the model:

$$\in \mathcal{G}(v) \Leftrightarrow$$

$$\forall s \in \mathcal{S}; \qquad (s) \in \operatorname*{argmax}_{a \in \mathcal{A}} \left(r(s;a) + \sum_{s' \in \mathcal{S}} P(s'|s;a) v(s') \right) :$$

- Sampling the optimality operator?
  - Optimality operator:

$$[T_* v](s) = \max_{a \in \mathcal{A}} \mathbb{E}_{S' \sim P(.|s,a)} [r(s,a) + \gamma v(S')];$$

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

$$[\hat{T}_* v](s_i) = \max_{a \in \mathcal{A}} \left(r(s_i, a) + \gamma v(s'_{i,a}) \right) ;$$

  - it is biased: $\mathbb{E}[[\hat{T}_* v](s_i)|s_i] \neq T_*(s_i)$.

- state-action value function (also called Q-function and quality function)

$$Q^{\pi}(s;a) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(S_t;A_t) | S_0 = s; A_0 = a; S_{t+1} \sim P(:|S_t;A_t); A_{t+1} = \pi(S_{t+1})]:$$

- Bellman evaluation operator $T_\pi : \mathbb{R}^{\mathcal{S}\times\mathcal{A}} \to \mathbb{R}^{\mathcal{S}\times\mathcal{A}}$
  - definition:
    $[T_\pi Q](s,a) = r(s,a) + \gamma \sum_{s'\in\mathcal{S}} P(s'|s,a) Q(s', \pi(s'))$;
  - 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:
    $[T_* Q](s,a) = r(s,a) + \gamma \sum_{s'\in\mathcal{S}} P(s'|s,a) \max_{a'\in\mathcal{A}} Q(s',a')$;
  - 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 $v_\pi = Q_\pi(s, \pi(s))$:

$$'\in \mathcal{G}(v\ ) \Leftrightarrow \forall s \in \mathcal{S};\qquad (s) \in \operatorname*{argmax}_{a\in\mathcal{A}} \left(r(s;a) + \sum_{s'\in\mathcal{S}} P(s'|s;a) v\ (s') \right)$$

$$\Leftrightarrow \forall s \in \mathcal{S};\qquad '(s) \in \operatorname*{argmax}_{a\in\mathcal{A}} Q\ (s;a):$$

  - resp. to v_*:

$$\pi_*(s) \in \operatorname*{argmax}_{a\in\mathcal{A}} Q_*(s,a).$$

  - 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

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

  - sampled Bellman evaluation operator

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

  - sampled Bellman optimality operator

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

- Features for the Q-function:
  - linear param. of the Q-function: $Q_\theta(s, a) = \theta^\top \phi(s, a)$ with

$$\phi(s, a) = \begin{bmatrix} \delta_{a=a_1} \phi(s)^\top & \dots & \delta_{a=a_{|\mathcal{A}|}} \phi(s)^\top \end{bmatrix}^\top.$$

  - other representations are possible.

- Value iteration: $Q_{k+1} = T_* Q_k$
  - T_* cannot be applied, the model being unknown;
  - with large space, $Q_k \in \mathcal{H}$, no reason for $T_* Q_k \in \mathcal{H}$ to hold.
- Approximate value iteration (an introductory example):
  - linear param. for the Q-functions

$$\mathcal{H} = \{Q_\theta(s,a) = \theta^\top \phi(s,a), \theta \in \mathbb{R}^d\};$$

  - writing $Q_k = Q_{\theta_k}$, sampled operator:

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

  - search for $Q \in \mathcal{H}$ being the closest to $\hat{T}_* Q_k$:

$$Q_{k+1} \in \operatorname*{argmin}_{Q_\theta \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} \Big(Q_\theta(s_i, a_i) - [\hat{T}_* Q_k](s_i, a_i) \Big)^2 .$$

  - summary: $Q_{k+1} = \Pi \hat{T}_* Q_k$.

Abstraction

- Approximate value iteration:

$$Q_{k+1} = \mathfrak{A}\hat{T}_* Q_k.$$

- $\mathfrak{A}$ is an abstract approximation operator
- $\mathfrak{A}\hat{T}_*$ should be a contraction!
  - otherwise divergence can (will) occur;
  - not the case for $\Pi\hat{T}_*$...
    - do not implement the introductory example!
  - true if averagers are used for function approximation, such as
    - ensemble of trees
    - notably extremely randomized forests: fitted-Q
    - kernel averagers (Nadaraya-Watson)

Algorithm 4 Approximate value iteration

Require: A dataset $\mathcal{D} = \{(s_i, a_i, r_i, s_i')_{1 \leq i \leq n}\}$, 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:

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

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

4: **end for**

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

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

- 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 $Q_k \in \mathcal{H}$ 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 5 Approximate policy iteration

Require: 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 $Q_k \in \mathcal{H}$ 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}

- Problem: how to find an approximate fixed point of T_π, that is a function $Q_\theta \in \mathcal{H}$ such that $Q_\theta \approx T_\pi Q_\theta$?

Monte Carlo rollouts

- Approx. fixed point of T $\Leftrightarrow$ approx Q .
- Assume that Q is known, simply a regression problem. For example, linear least-squares:

$$\min_{\in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left(Q\ (s_i; a_i) - Q\ (s_i; a_i)\right)^2 :$$

- Q is (obviously) unknown... 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 all subsequent actions being chosen according to ;
  - write q_i the associated discounted cumulative reward;
  - unbiased estimate: $\mathbb{E}[q_i | s_i; a_i] = Q\ (s_i; a_i)$
- replace $Q\ (s_i; a_i)$ by the unbiased estimate q_i:

$$\min_{\in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left(q_i - Q\ (s_i; a_i)\right)^2 :$$

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

Residual approach

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

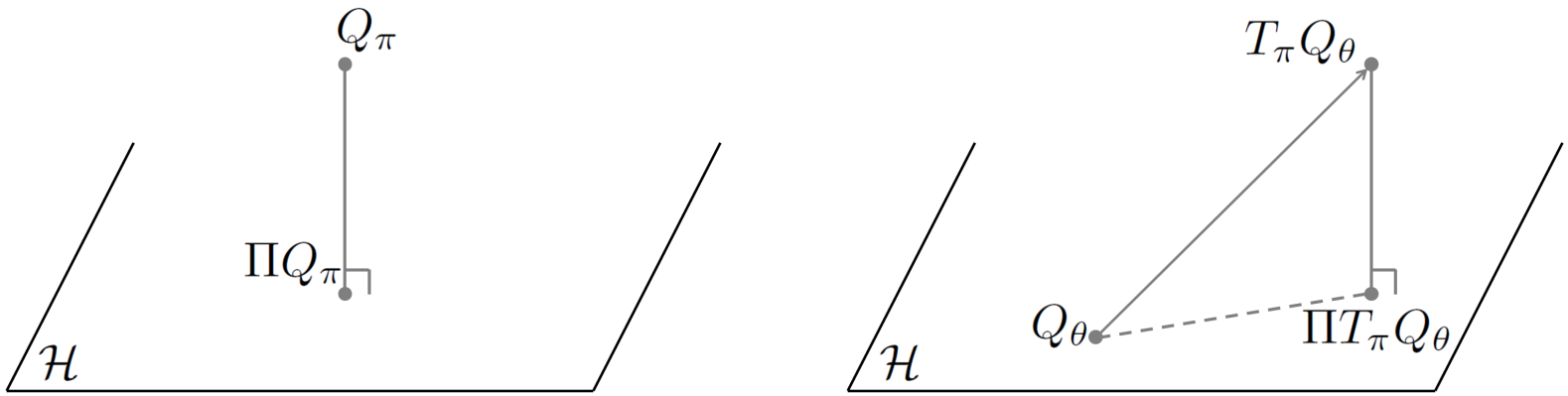
$$\min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left([\hat{T}_\pi Q_\theta](s_i, a_i) - Q_\theta(s_i, a_i) \right)^2$$

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

- However, there is a bias problem:

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

Least-Squares temporal Differences



- Idea: solve

$$Q_\theta = \Pi T_\pi Q_\theta.$$

- As a nested optimization problem:

$$\begin{cases} w_\theta = \text{argmin}_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n (r_i + \gamma Q_\theta(s_i', \pi(s_i')) - Q_w(s_i, a_i))^2 \\ \theta_n = \text{argmin}_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n (Q_\theta(s_i, a_i) - Q_w\ (s_i, a_i))^2 \end{cases}.$$

Least-Squares temporal Differences (cont.)

- Opt. problem (linear param. assumed):

$$\begin{cases} w = \text{argmin}_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (r_i + Q(s_i'; \pi(s_i')) - Q_w(s_i; a_i))^2 \\ \theta_n = \text{argmin}_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} (Q(s_i; a_i) - Q_{w_\theta}(s_i; a_i))^2 \end{cases}.$$

- First Eq., linear least-squares in w:

$$w = \left(\sum_{i=1}^{n} \phi(s_i; a_i) \phi(s_i; a_i)^\top \right)^{-1} \sum_{i=1}^{n} \phi(s_i; a_i)(r_i + \gamma \phi^\top(s_i'; \pi(s_i'))).$$

- Second Eq. minimized for $\theta = w$:

$$\theta_n = w_n \Leftrightarrow \theta_n = \left(\sum_{i=1}^{n} \phi(s_i; a_i) \phi(s_i; a_i)^\top \right)^{-1} \sum_{i=1}^{n} \phi(s_i; a_i)(r_i + \gamma \theta_n^\top \phi(s_i'; \pi(s_i')))$$

$$\Leftrightarrow \theta_n = \left(\sum_{i=1}^{n} \phi(s_i; a_i) (\phi(s_i; a_i) - \gamma \phi(s_i'; \pi(s_i')))^\top \right)^{-1} \sum_{i=1}^{n} \phi(s_i; a_i) r_i.$$

- API with LSTD named LSPI (Least-Squares Policy Iteration).

Least-Squares temporal Differences (cont.)

Algorithm 6 Least-squares policy iteration

Require: 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:

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

3: policy improvement:

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

4: **end for**

5: **return** the policy π_{K+1}

Approximating the policy

- Idea: instead of generalizing the Q-function, generalize the policy
- Motivation: a policy might be easier to learn than a Q function
- At iteration k, let $\mathcal{F} \subset \mathcal{A}^{\mathcal{S}}$ be an hypothesis space of policies, assume that $Q_{\pi_k}(s_i, a)$ are known, and solve the cost-sensitive multi-class 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

- For ADP, we assumed that the dataset was provided.
- What about online learning?
  - requires an online learner;
  - dilemma between exploration and exploitation.

SARSA

- Goal: online estimation of Q , for a given .
- Assume a linear param., and that Q is known, the risk of interest is

$$\min_{\in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left(Q \ (s_i; a_i) - Q \ (s_i; a_i) \right)^2 :$$

- Minimize it with a stochastic gradient descent:

$$\begin{aligned} _{i+1} &= \ _i - \frac{_i}{2} \nabla \left(Q \ (s_i; a_i) - Q \ (s_i; a_i) \right)^2 \\ &= \ _i + \ _i \ (s_i; a_i) \left(Q \ (s_i; a_i) - \ _i^\top \ (s_i; a_i) \right) ; \end{aligned}$$

- $Q \ (s_i; a_i)$ is unknown, bootstrap it ($s_{i+1} \sim P(:|s_i; a_i)$ and $a_{i+1} = \ (s_{i+1})$)

$$Q \ (s_i; a_i) \to [\hat{T} \ Q \ _i](s_i; a_i) = r_i + \ Q \ _i(s_{i+1}; a_{i+1}):$$

- Replace $Q \ (s_i; a_i)$ by its estimate in the update rule

$$\begin{aligned} _{i+1} &= \ _i + \ _i \ (s_i; a_i) (r_i + \ Q \ _i(s_{i+1}; a_{i+1}) - Q \ _i(s_i; a_i)) \\ &= \ _i + \ _i \ (s_i; a_i) \left(r_i + \ _i^\top \ (s_{i+1}; a_{i+1}) - \ _i^\top \ (s_i; a_i) \right) : \end{aligned}$$

- This is called a temporal difference algorithm.

SARSA (cont.)

Algorithm 7 SARSA

Require: An initial parameter vector θ_0, the initial state s_0, an initial action a_0, the learning rates $(\alpha_i)_{i \geq 0}$

1: i = 0
2: **while true do**
3: Apply action a_i in state s_i
4: Get the reward r_i and observe the new state s_{i+1}
5: **Choose the action a_{i+1} to be applied in state s_{i+1}**
6: Update the parameter vector of the Q-function according to the transition $(s_i; a_i; r_i; s_{i+1}; a_{i+1})$;

$$\theta_{i+1} = \theta_i + \alpha_i \, \nabla_\theta Q_\theta(s_i; a_i) \left(r_i + \gamma \, \theta_i^\top \, \nabla_\theta Q_\theta(s_{i+1}; a_{i+1}) - \theta_i^\top \, \nabla_\theta Q_\theta(s_i; a_i) \right)$$

7: $i \leftarrow i + 1$
8: **end while**

- Remark: on-policy algorithm.

Q-learning

- Goal: direct online estimation of Q_*.
- Assume a linear param., and that Q_* is known, the risk of interest is

$$\min_{\in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \left(Q_*(s_i; a_i) - Q\ (s_i; a_i)\right)^2 :$$

- Minimize it with a stochastic gradient descent:

$$\begin{aligned} {}_{i+1} &= {}_i - \frac{i}{2} \nabla \left(Q_*(s_i; a_i) - Q\ (s_i; a_i)\right)^2 \\ &= {}_i + {}_i\ (s_i; a_i) \left(Q_*(s_i; a_i) - {}_i^\top\ (s_i; a_i)\right) ; \end{aligned}$$

- $Q_*(s_i; a_i)$ is unknown, bootstrap it $(s_{i+1} \sim P(:|s_i; a_i))$

$$Q_*(s_i; a_i) \to [\hat{T}_* Q\ {}_i](s_i; a_i) = r_i + \max_{a \in \mathcal{A}} Q\ {}_i(s_{i+1}; a):$$

- Replace $Q_*(s_i; a_i)$ by its estimate in the update rule

$$\begin{aligned} {}_{i+1} &= {}_i + {}_i\ (s_i; a_i) \left(r_i + \max_{a \in \mathcal{A}} Q\ {}_i(s_{i+1}; a) - Q\ {}_i(s_i; a_i)\right) \\ &= {}_i + {}_i\ (s_i; a_i) \left(r_i + \max_{a \in \mathcal{A}} ({}_i^\top\ (s_{i+1}; a)) - {}_i^\top\ (s_i; a_i)\right) : \end{aligned}$$

Q-learning (cont.)

Algorithm 8 Q-learning

Require: An initial parameter vector θ_0, the initial state s_0, the learning rates $(\alpha_i)_{i \geq 0}$

1: i = 0
2: **while true do**
3: **Choose the action a_i to be applied in state s_i**
4: Apply action a_i in state s_i
5: Get the reward r_i and observe the new state s_{i+1}
6: Update the parameter vector of the Q-function according to the transition $(s_i; a_i; r_i; s_{i+1})$;

$$\theta_{i+1} = \theta_i + \alpha_i \, \nabla_\theta Q_{\theta_i}(s_i; a_i) \left(r_i + \max_{a \in \mathcal{A}} \left(\theta_i^\top \phi(s_{i+1}; a) \right) - \theta_i^\top \phi(s_i; a_i) \right)$$

7: $i \leftarrow i + 1$
8: **end while**

- Remark: off-policy algorithm.

- With SARSA or Q-learning, what action should be applied?
  - acting always greedily is not wise;
  - dilemma between exploration and exploitation.
- ϵ-greedy policy:

$$\pi_\epsilon(s) = \begin{cases} \text{argmax}_{a \in \mathcal{A}} Q_\theta(s, a) & \text{with probability } 1 - \epsilon \\ \text{a random action} & \text{with probability } \epsilon \end{cases}.$$

- Softmax (stochastic) policy (τ is the temperature param.):

$$\pi_\tau(a|s) = \frac{e^{\frac{1}{\tau} Q(s,a)}}{\sum_{a' \in \mathcal{A}} e^{\frac{1}{\tau} Q(s,a')}}.$$

- Other schemes exist.

- What about continuous actions (max, argmax)?
- Policy search: parameterize the policy, search directly in the policy space.
- We'll use stochastic policies:
  - stochastic policy: $\pi \in \Delta_{\mathcal{A}}^{\mathcal{S}}$ associates to each state s a conditional probability over actions $\pi(.|s)$
  - All things already defined extend naturaly to stochastic policies:

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left(r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) v_\pi(s) \right),$$

$$v_\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) Q_\pi(s,a),$$

$$Q_\pi(s,a) = r(s,a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s,a) v_\pi(s).$$

- Exemples of parameterized policies:
  - discrete actions

$$\pi_\theta(a|s) = \frac{e^{\theta^\top \phi(s,a)}}{\sum_{a' \in \mathcal{A}} e^{\theta^\top \phi(s,a')}},$$

  - continuous (1d here) actions

$$\pi_\theta(a|s) \propto e^{-\frac{1}{2}\left(\frac{a-\theta^\top \phi(s)}{\sigma}\right)^2}.$$

- The policy search problem:
  - let $\nu \in \Delta_{\mathcal{S}}$ be a user-defined distribution over states;
  - solve

$$max_{\theta \in \mathbb{R}^d} J(\theta) \text{ with } J(\theta) = \sum_{s \in \mathcal{S}} \nu(s) v_{\pi_\theta}(s) = \mathbb{E}_{S \sim \nu}[v_{\pi_\theta}(S)].$$

- Difference with (A)DP:
  - DP: find a policy that maximizes the value *for every state*;
  - Policy search: find a policy that maximizes the value *in average*.

- natural approach, gradient ascent:

$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta).$$

- Gradient $\nabla_\theta J(\theta)$?
- Define $d_{\nu,\pi} \in \Delta_{\mathcal{S}}$, the γ-weighted occupancy measure:

$$d_{\nu,\pi} = (1-\gamma)\nu^\top (I - \gamma P_\pi)^{-1}.$$

Theorem (Policy gradient)

Let π_θ be such that $\pi_\theta(a|s) > 0$, for all s, a. We have

$$\begin{aligned}\nabla_\theta J(\theta) &= \frac{1}{1-\gamma} \sum_{s \in \mathcal{S}} d_{\nu,\pi(s)} \sum_{a \in \mathcal{A}} \pi_\theta(a|s) Q_\pi\ (s,a) \nabla_\theta \ln \pi_\theta(a|s) \\ &= \frac{1}{1-\gamma} \mathbb{E}_{S \sim d\ ;\ ,A \sim \pi\ (.|S)} [Q_\pi\ (S,A) \nabla_\theta \ln \pi_\theta(A|S)].\end{aligned}$$

- Policy gradient:

$$\nabla_\theta J(\theta) = \frac{1}{1-\gamma} \mathbb{E}_{S \sim d\ ;\ ,A \sim \pi\ (.|S)}[Q_\pi\ (S,A) \nabla_\theta \ln \pi_\theta(A|S)].$$

- Q_π can be estimated pointwise using Monte Carlo rollouts.
- Policy search is called an actor method (DPI too).
- Can we replace Q_π by an approximation $Q_w \in \mathcal{H}$, without changing the gradient?
- If so, the resulting approach is called an actor critic method.

Policy gradient with a critic

Theorem (Policy gradient with a critic)

If the parametrization of the state-action value function is compatible, in the sense that

$$\forall (s,a) \in \mathcal{S} \times \mathcal{A}, \quad \nabla_\theta \ln \pi_\theta(a|s) = \nabla_w Q_w(s,a),$$

and if Q_w is a local optimum of the risk based on the ℓ_2-loss, with state-action distribution given by $d_{\nu,\pi}$, and with the target function being Q_π, that is

$$\nabla_w \mathbb{E}_{d\ ;} \left[\left(Q_\pi\ (S,A) - Q_w(S,A) \right)^2 \right] = 0,$$

then the gradient satisfies

$$\nabla_\theta J(\theta) = \mathbb{E}_{d\ ;} \left[Q_w(S,A) \nabla_\theta \ln \pi_\theta(A|S) \right].$$

Policy gradient with a critic (cont.)

- Exemple of compatible approximation.
- Softmax policy: $\pi_\theta(a|s) = \frac{e^{\theta^\top \phi(s,a)}}{\sum_{a' \in \mathcal{A}} e^{\theta^\top \phi(s,a')}}$.
- Gradient: $\nabla_\theta \ln \pi_\theta(a|s) = \phi(s,a) - \sum_{a' \in \mathcal{A}} \pi_\theta(a'|s)\phi(s,a')$.
- Compatible approximation:
  $Q_w(s,a) = w^\top(\phi(s,a) - \sum_{a' \in \mathcal{A}} \pi_\theta(a'|s)\phi(s,a'))$.
- Not a Q-function as $\sum_{a \in \mathcal{A}} \pi_\theta(a|s) Q_w(s,a) = 0$
- More an advantage function: $A_\pi(s,a) = Q_\pi(s,a) - v_\pi(s)$.
- Yet, for any $v \in \mathbb{R}^{\mathcal{S}}$, $\mathbb{E}_{d_{\nu,\pi}}[v(s)\nabla_\theta \ln \pi_\theta(a|s)] = 0$
- As the term $w^\top \sum_{a' \in \mathcal{A}} \pi_\theta(a'|s)\phi(s,a')$ does not depend on a, a compatible approximation is

$$Q_w(s,a) = \theta^\top \phi(s,a).$$

Natural policy gradient

- Natural gradient:
  - gradient premultiplied by the inverse of the Fisher information matrix
  - instead of following the steepest direction in the parameter space, it follows the steepest direction with respect to Fisher metric
  - tends to be much more efficient empirically
- in our case, the natural gradient $\tilde{\nabla}$ is

$$\tilde{\nabla}_\theta J(\theta) = F(\theta)^{-1} \nabla_\theta J(\theta)$$

$$\text{with } F(\theta) = \mathbb{E}_{d\,;}\, [\nabla_\theta \ln \pi_\theta(A|S)(\nabla_\theta \ln \pi_\theta(A|S))^\top]$$

Natural policy gradient (cont.)

Theorem (Natural policy gradient with a critic)

If the parametrization of the state-action value function is compatible, in the sense that

$$\forall (s,a) \in \mathcal{S} \times \mathcal{A}, \quad \nabla_\theta \ln \pi_\theta(a|s) = \nabla_w Q_w(s,a),$$

and if Q_w is a local optimum of the risk based on the ℓ_2-loss, with state-action distribution given by $d_{\nu,\pi}$, and with the target function being Q_π, that is

$$\nabla_w \mathbb{E}_{d\ ;}\ [(Q_\pi\ (S,A) - Q_w(S,A))^2] = 0,$$

then the natural gradient satisfies

$$\tilde{\nabla}_\theta J(\theta) = w.$$