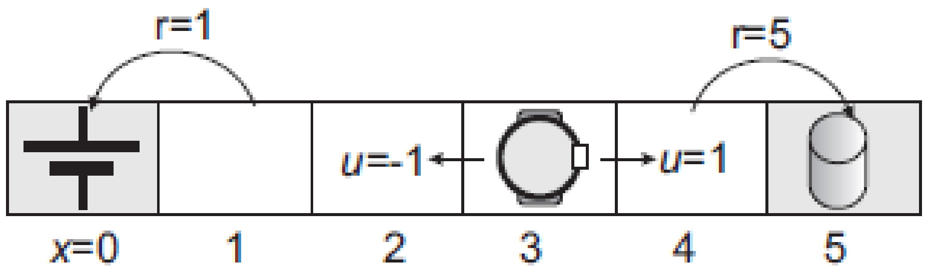
Reinforcement Learning

Mohsen Afsharchi

Deterministic cleaning-robot

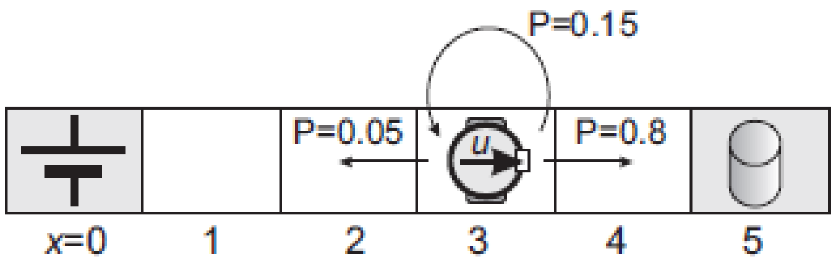


$$f(x,u) = \begin{cases} x+u & \text{if } 1 \le x \le 4 \\ x & \text{if } x = 0 \text{ or } x = 5 \text{ (regardless of } u\text{)} \end{cases}$$

$$\rho(x,u) = \begin{cases} 5 & \text{if } x = 4 \text{ and } u = 1 \\ 1 & \text{if } x = 1 \text{ and } u = -1 \\ 0 & \text{otherwise} \end{cases}$$

$$Q^*(x,u) = \rho(x,u) + \gamma \max_{u'} Q^*(f(x,u),u')$$

Stochastic cleaning-robot

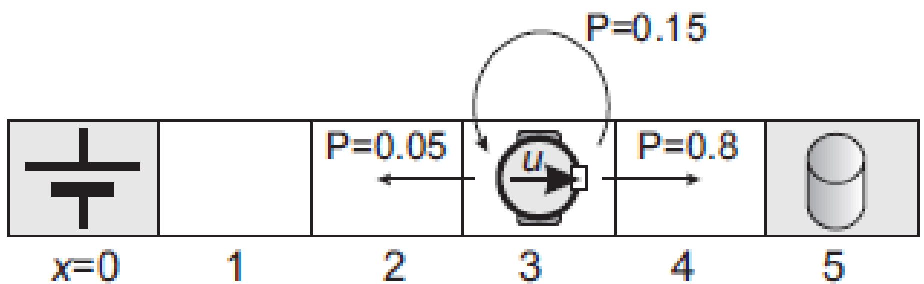


$$\tilde{\rho}(x,u,x') = \begin{cases} 5 & \text{if } x \neq 5 \text{ and } x' = 5 \\ 1 & \text{if } x \neq 0 \text{ and } x' = 0 \\ 0 & \text{otherwise} \end{cases}$$

TABLE 2.1 Dynamics of the stochastic, cleaning-robot MDP.

(x,u)	$\bar{f}(x,u,0)$	$\bar{f}(x,u,1)$	$\bar{f}(x,u,2)$	$\bar{f}(x,u,3)$	$\bar{f}(x,u,4)$	$\bar{f}(x,u,5)$
$(0,-1)$	1	0	0	0	0	0
$(1,-1)$	0.8	0.15	0.05	0	0	0
$(2,-1)$	0	0.8	0.15	0.05	0	0
$(3,-1)$	0	0	0.8	0.15	0.05	0
$(4,-1)$	0	0	0	0.8	0.15	0.05
$(5,-1)$	0	0	0	0	0	1
$(0,1)$	1	0	0	0	0	0
$(1,1)$	0.05	0.15	0.8	0	0	0
$(2,1)$	0	0.05	0.15	0.8	0	0
$(3,1)$	0	0	0.05	0.15	0.8	0
$(4,1)$	0	0	0	0.05	0.15	0.8
$(5,1)$	0	0	0	0	0	1

Optimality in the stochastic setting



$$Q^h(x,u) = \mathrm{E}_{x' \sim \tilde{f}(x,u,\cdot)} \left\{ \tilde{\rho}(x,u,x') + \gamma Q^h(x', h(x')) \right\}$$

$$Q^*(x,u) = \mathrm{E}_{x' \sim \tilde{f}(x,u,\cdot)} \left\{ \tilde{\rho}(x,u,x') + \gamma \max_{u'} Q^*(x', u') \right\}$$

Q-iteration for deterministic MDPs.

ALGORITHM 2.1 Q-iteration for deterministic MDPs.

Input: dynamics f, reward function ρ, discount factor γ

1: initialize Q-function, e.g., $Q_0 \leftarrow 0$
2: **repeat** at every iteration $\ell = 0, 1, 2, \dots$
3:     **for** every (x, u) **do**
4:         $Q_{\ell+1}(x,u) \leftarrow \rho(x,u) + \gamma \max_{u'} Q_\ell(f(x,u),u')$
5:     **end for**
6: **until** $Q_{\ell+1} = Q_\ell$

Output: $Q^* = Q_\ell$

Q-iteration for stochastic MDPs.

ALGORITHM 2.2 Q-iteration for stochastic MDPs with countable state spaces.

Input: dynamics $\bar{f}$, reward function $\tilde{\rho}$, discount factor γ

initialize Q-function, e.g., $Q_0 \leftarrow 0$

repeat at every iteration $\ell = 0, 1, 2, \ldots$

  for every (x, u) **do**

    $Q_{\ell+1}(x, u) \leftarrow \sum_{x'} \bar{f}(x, u, x')\left[\tilde{\rho}(x, u, x') + \gamma \max_{u'} Q_\ell(x', u')\right]$

  end for

until $Q_{\ell+1} = Q_\ell$

Output: $Q^* = Q_\ell$

Policy Iteration

ALGORITHM 2.4 Policy iteration with Q-functions.

1: initialize policy h_0
2: **repeat** at every iteration $\ell = 0, 1, 2, \ldots$
3: find Q^{h_ℓ}, the Q-function of h_ℓ ▷ policy evaluation
4: $h_{\ell+1}(x) \in \arg\max_u Q^{h_\ell}(x, u)$ ▷ policy improvement
5: **until** $h_{\ell+1} = h_\ell$

Output: $h^* = h_\ell$, $Q^* = Q^{h_\ell}$

Model-free value iteration

$$Q_{k+1}(x_k,u_k) = Q_k(x_k,u_k) + \alpha_k[r_{k+1} + \gamma \max_{u'} Q_k(x_{k+1},u') - Q_k(x_k,u_k)]$$

As the number of transitions k approaches infinity, Q-learning asymptotically converges to Q^* if the state and action spaces are discrete and finite, and under the following conditions

- The sum $\sum_{k=0}^{\infty} \alpha_k^2$ produces a finite value, whereas the sum $\sum_{k=0}^{\infty} \alpha_k$ produces an infinite value.
- All the state-action pairs are (asymptotically) visited infinitely often.

Exploration vs Exploitation

The second condition can be satisfied if, among other things, the controller has a nonzero probability of selecting any action in every encountered state; this is called exploration. The controller also has to exploit its current knowledge in order to obtain good performance, e.g., by selecting greedy actions in the current Q-function. This is a typical illustration of the exploration-exploitation trade-off in online RL. A classical way to balance exploration with exploitation in Q-learning is ε-greedy exploration, which selects actions according to:

$$u_k = \begin{cases} u \in \arg\max_{\bar{u}} Q_k(x_k, \bar{u}) & \text{with probability } 1 - \varepsilon_k \\ \text{a uniformly random action in } U & \text{with probability } \varepsilon_k \end{cases} \tag{2.32}$$

where $\varepsilon_k \in (0,1)$ is the exploration probability at step k. Another option is to use Boltzmann exploration (Sutton and Barto, 1998, Section 2.3), which at step k selects an action u with probability:

$$\mathrm{P}\,(u \,|\, x_k) = \frac{e^{Q_k(x_k,u)/\tau_k}}{\sum_{\bar{u}} e^{Q_k(x_k,\bar{u})/\tau_k}} \tag{2.33}$$

where the temperature $\tau_k \geq 0$ controls the randomness of the exploration. When $\tau_k \rightarrow$

Q-Learning

ALGORITHM 2.3 Q-learning with ε-greedy exploration.

Input: discount factor γ,
exploration schedule $\{\varepsilon_k\}_{k=0}^{\infty}$, learning rate schedule $\{\alpha_k\}_{k=0}^{\infty}$

1: initialize Q-function, e.g., $Q_0 \leftarrow 0$
2: measure initial state x_0
3: **for** every time step $k = 0, 1, 2, \ldots$ **do**
4: $$u_k \leftarrow \begin{cases} u \in \arg\max_{\bar{u}} Q_k(x_k, \bar{u}) & \text{with probability } 1 - \varepsilon_k \text{ (exploit)} \\ \text{a uniformly random action in } U & \text{with probability } \varepsilon_k \text{ (explore)} \end{cases}$$
5: apply u_k, measure next state x_{k+1} and reward r_{k+1}
6: $Q_{k+1}(x_k, u_k) \leftarrow Q_k(x_k, u_k) + \alpha_k[r_{k+1} + \gamma \max_{u'} Q_k(x_{k+1}, u') - Q_k(x_k, u_k)]$
7: **end for**

Q-Learning vs Sarsa Learning

$// \, s \, , s' \rightarrow \, states$

$// \, a \, , a' \rightarrow \, actions$

$// \, Q \rightarrow \, state\text{-}action \;\; value$

$// \, \alpha \, , \gamma \rightarrow \, learning \;\; parameters(learning \; rate \, , discount \;\; factor)$

1. Initialize $\boldsymbol{Q}(\boldsymbol{s},\boldsymbol{a})$ arbitrarily
2. Repeat (for each episode)

2.1. Initialize $\boldsymbol{s}$

2.2. Repeat (for each step of episode) until $\boldsymbol{s}$ is terminal

2.2.1. Choose $\boldsymbol{a}$ from $\boldsymbol{s}$ using policy derived from $\boldsymbol{Q}$ (e.g. ϵ-greedy)

2.2.2. Take action $\boldsymbol{a}$ observe reward $\boldsymbol{r}$, state $\boldsymbol{s}'$

2.2.3. $\boldsymbol{Q}(\boldsymbol{s},\boldsymbol{a}) \leftarrow \boldsymbol{Q}(\boldsymbol{s},\boldsymbol{a}) + \alpha [\boldsymbol{r} + \gamma \cdot \boldsymbol{max}_{\boldsymbol{a}'} \boldsymbol{Q}(\boldsymbol{s}',\boldsymbol{a}') - \boldsymbol{Q}(\boldsymbol{s},\boldsymbol{a})]$

2.2.4. $\boldsymbol{s} \leftarrow \boldsymbol{s}'$

Q-Learning vs Sarsa Learning

$// \, s \, , s' \rightarrow \, states$

$// \, a \, , a' \rightarrow \, actions$

$// \, Q \rightarrow \, state\text{-}action \;\, value$

$// \, \alpha \, , \gamma \rightarrow \, learning \;\, parameters \, (learning \; rate \, , discount \;\, factor \,)$

1. Initialize $\boldsymbol{Q}(\boldsymbol{s}, \boldsymbol{a})$ arbitrarily

2. Repeat (for each episode)

2.1. Initialize $\boldsymbol{s}$

2.2. Choose $\boldsymbol{a}$ from $\boldsymbol{s}$ using policy derived from $\boldsymbol{Q}$ (e.g. ϵ-greedy)

2.3. Repeat (for each step of episode) until $\boldsymbol{s}$ is terminal

2.3.1. Take action $\boldsymbol{a}$ observe reward $\boldsymbol{r}$, state $\boldsymbol{s}'$

2.3.2. Choose $\boldsymbol{a}'$ from $\boldsymbol{s}'$ using policy derived from $\boldsymbol{Q}$ (e.g. ϵ-greedy)

2.3.3. $\boldsymbol{Q}(\boldsymbol{s}, \boldsymbol{a}) \leftarrow \boldsymbol{Q}(\boldsymbol{s}, \boldsymbol{a}) + \alpha [\boldsymbol{r} + \gamma \cdot \boldsymbol{Q}(\boldsymbol{s}', \boldsymbol{a}') - \boldsymbol{Q}(\boldsymbol{s}, \boldsymbol{a})]$

2.3.4. $\boldsymbol{s} \leftarrow \boldsymbol{s}'$, $\boldsymbol{a} \leftarrow \boldsymbol{a}'$