

# Reinforcement Learning

Bojan Božić

TU Dublin

Summer 2021

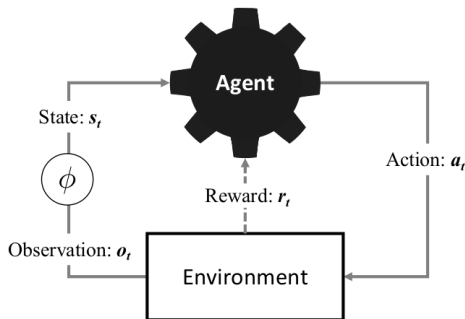
- 1 Big Idea
- 2 Fundamentals
- 3 Standard Approach: Q-Learning, Off-Policy Temporal-Difference Learning
- 4 Extensions and Variations
- 5 Summary
- 6 Further Reading

# Big Idea

- Sarah is a young venture scout in training for her pioneering badge.
- One of the more unusual challenges involved in earning this badge is to learn to cross a stream using a set of stepping-stones while wearing an electronic blindfold.
- The goal is to get across the river in the fewest steps possible without getting wet.
- Before the scout attempts a step, the blindfold is made transparent for 0.5 seconds to give the scout a quick view of their environment so that they make a decision about which direction they will step in and how far.

# Fundamentals

$$(o_1, a_1, r_1), (o_2, a_2, r_2), (o_3, a_3, r_3), \dots, (o_e, a_e, r_e) \quad (1)$$



**Figure 1:** An agent behaving in an environment and the observation, reward, action cycle. The transition from observations of the environment to a state is shown by the state generation function,  $\phi$ .

$$(s_1, a_1, r_1), (s_2, a_2, r_2), (s_3, a_3, r_3), \dots, (s_e, a_e, r_e) \quad (2)$$



$$G = r_t + r_{t+1} + r_{t+2} + r_{t+3} + \dots + r_e \quad (3)$$

$$a_t = \pi(s_t) \quad (4)$$

$$P(A_t = a \mid S_t = s) = \pi(S_t = s) \quad (5)$$

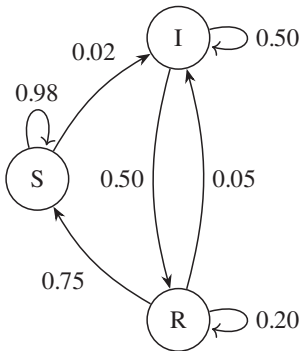
$$V_{\pi}(s_t) = E_{\pi}[r_t + r_{t+1} + r_{t+2} + r_{t+3} + \dots + r_e \mid s_t] \quad (6)$$

$$Q_{\pi}(s_t, a_t) = E_{\pi}[r_t + r_{t+1} + r_{t+2} + r_{t+3} + \dots + r_e \mid s_t, a_t] \quad (7)$$

$$G_{\gamma=0.9} = r_t + 0.9 \times r_{t+1} + 0.81 \times r_{t+2} + 0.729 \times r_{t+3} + \dots$$

$$Q_{\pi}(s_t, a_t) = E_{\pi}[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \dots + \gamma^{3-t} r_e \mid s_t, a_t] \quad (9)$$

## Markov Decision Processes



(a) S-I-R Markov process

$$\mathcal{P} = \begin{matrix} & \begin{matrix} S & I & R \end{matrix} \\ \begin{matrix} S \\ I \\ R \end{matrix} & \begin{bmatrix} 0.98 & 0.02 & 0.00 \\ 0.00 & 0.50 & 0.50 \\ 0.75 & 0.05 & 0.20 \end{bmatrix} \end{matrix}$$

(b) S-I-R transition matrix

**Figure 2:** A simple Markov process to model the evolution of an infectious disease in individuals during an epidemic using the SUSCEPTIBLE-INFECTED-RECOVERED (S-I-R) model.

$$P(S_{t+1} \mid S_t, S_{t-1}, S_{t-2}, \dots) = P(S_{t+1} \mid S_t) \quad (10)$$

$$P(s_1 \rightarrow s_2) = P(S_{t+1} = s_2 \mid S_t = s_1) \quad (11)$$

$$\mathcal{P} = \begin{bmatrix} P(s_1 \rightarrow s_1) & P(s_1 \rightarrow s_2) & \dots & P(s_1 \rightarrow s_n) \\ P(s_2 \rightarrow s_1) & P(s_2 \rightarrow s_2) & \dots & P(s_2 \rightarrow s_n) \\ \vdots & \vdots & \ddots & \vdots \\ P(s_n \rightarrow s_1) & P(s_n \rightarrow s_2) & \dots & P(s_n \rightarrow s_n) \end{bmatrix}$$

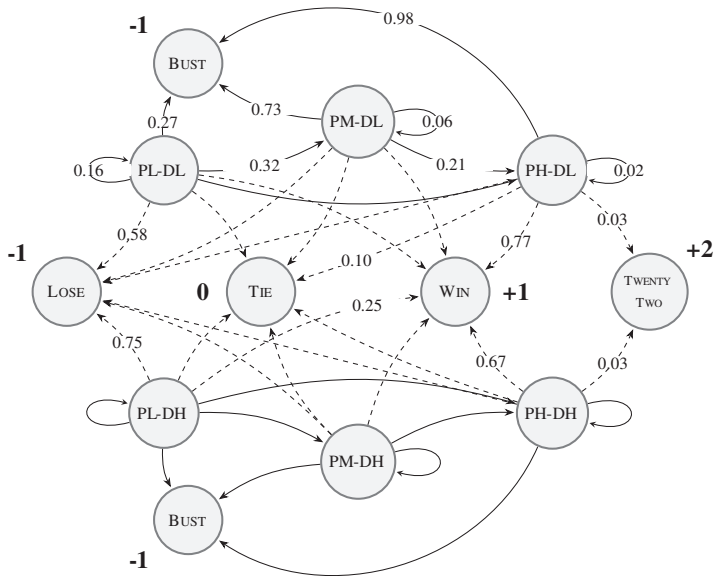
$$P(s_1 \xrightarrow{a} s_2) = P(S_{t+1} = s_2 \mid S_t = s_1, A_t = a) \quad (12)$$

$$R(s_1 \xrightarrow{a} s_2) = E(r_t \mid S_t = s_1, S_{t+1} = s_2, A_t = a) \quad (13)$$



**Table 1:** Some episodes of games played by the TwentyTwos agent showing the cards dealt, as well as the states, actions, and rewards. Note that rewards are shown on the row indicating the action that led to them, not the state that followed that action.

Iter	Player Hand	Dealer Hand	State	Action	Reward
1	2♥ 7♣	(9) 8♥	(8) PL-DH	<i>Twist</i>	0
2	2♥ 7♣ K♣	(19) 8♥	(8) PH-DH	<i>Stick</i>	+1
3	2♥ 7♣ K♣	(19) 8♥ Q♦	(18) WIN		
1	4♠ A♥	(15) Q♥	(10) PM-DH	<i>Twist</i>	-1
2	4♠ A♥ 9♣	(24) Q♥	(10) BUST		
1	2♦ 4♦	(6) 3♥	(3) PL-DL	<i>Twist</i>	0
2	2♦ 4♦ 3♥	(9) 3♥	(3) PL-DL	<i>Twist</i>	0
3	2♦ 4♦ 3♥ 6♣	(15) 3♥	(3) PM-DL	<i>Twist</i>	0
4	2♦ 4♦ 3♥ 6♣ 6♦	(21) 3♥	(3) PH-DL	<i>Stick</i>	0
5	2♦ 4♦ 3♥ 6♣ 6♦	(21) 3♥ 7♥ A♠	(21) TIE		
1	Q♦ J♣	(20) A♥	(11) PH-DH	<i>Stick</i>	+1
2	Q♦ J♣	(20) A♣ 5♣ Q♠	(26) WIN		
1	A♦ A♥	(22) 2♥	(2) PH-DL	<i>Stick</i>	+2
2	A♦ A♥	(22) 2♥	(2) TWENTYTWO		



**Figure 3:** A Markov decision process representation for TwentyTos, a simplified version of the card game Blackjack.

Markov Decision Processes

$$\mathcal{P}^{Twist} = \begin{bmatrix}
P(\text{PL-DL} \xrightarrow{Twist} \text{PL-DL}) & P(\text{PL-DL} \xrightarrow{Twist} \text{PM-DL}) & \dots & P(\text{PL-DL} \xrightarrow{Twist} \text{TWENTYTWO}) \\
P(\text{PM-DL} \xrightarrow{Twist} \text{PL-DL}) & P(\text{PM-DL} \xrightarrow{Twist} \text{PM-DL}) & \dots & P(\text{PM-DL} \xrightarrow{Twist} \text{TWENTYTWO}) \\
\vdots & \vdots & \ddots & \vdots \\
P(\text{TWENTYTWO} \xrightarrow{Twist} \text{PL-DL}) & P(\text{TWENTYTWO} \xrightarrow{Twist} \text{PM-DL}) & \dots & P(\text{TWENTYTWO} \xrightarrow{Twist} \text{TWENTYTWO})
\end{bmatrix}$$
  

$$\mathcal{P}^{Stick} = \begin{bmatrix}
P(\text{PL-DL} \xrightarrow{Stick} \text{PL-DL}) & P(\text{PL-DL} \xrightarrow{Stick} \text{PM-DL}) & \dots & P(\text{PL-DL} \xrightarrow{Stick} \text{TWENTYTWO}) \\
P(\text{PM-DL} \xrightarrow{Stick} \text{PL-DL}) & P(\text{PM-DL} \xrightarrow{Stick} \text{PM-DL}) & \dots & P(\text{PM-DL} \xrightarrow{Stick} \text{TWENTYTWO}) \\
\vdots & \vdots & \ddots & \vdots \\
P(\text{TWENTYTWO} \xrightarrow{Stick} \text{PL-DL}) & P(\text{TWENTYTWO} \xrightarrow{Stick} \text{PM-DL}) & \dots & P(\text{TWENTYTWO} \xrightarrow{Stick} \text{TWENTYTWO})
\end{bmatrix}$$

---

[illegible]

[illegible]

## The Bellman Equations

$$\begin{aligned}
 Q_{\pi}(s_t, a_t) &= E_{\pi} \left[ r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots + \gamma^3 r_{\infty} \mid s_t, a_t \right] \\
 &= E_{\pi} \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t, a_t \right] \tag{14}
 \end{aligned}$$

$$= E_{\pi} \left[ r_t + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t, a_t \right] \tag{15}$$

## The Bellman Equations

$$Q_{\pi}(s_t, a_t) = \sum_{s_{t+1}} P(s_t \xrightarrow{a_t} s_{t+1}) \left[ R(s_t \xrightarrow{a_t} s_{t+1}) + \gamma E_{\pi} \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_{t+1} \right] \right] \quad (16)$$

$$Q_{\pi}(s_t, a_t) = \sum_{s_{t+1}} P(s_t \xrightarrow{a_t} s_{t+1}) \left[ R(s_t \xrightarrow{a_t} s_{t+1}) + \gamma \sum_{a_{t+1}} E_{\pi} \left[ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_{t+1}, a_{t+1} \right] \right] \quad (17)$$

$$Q_{\pi}(s_t, a_t) = \sum_{s_{t+1}} P(s_t \xrightarrow{a_t} s_{t+1}) \left[ R(s_t \xrightarrow{a_t} s_{t+1}) + \gamma \sum_{a_{t+1}} \pi(s_{t+1}, a_{t+1}) Q_{\pi}(s_{t+1}, a_{t+1}) \right] \quad (18)$$

$$Q_{*}(s_t, a_t) = \sum_{s_{t+1}} P(s_t \xrightarrow{a_t} s_{t+1}) \left[ R(s_t \xrightarrow{a_t} s_{t+1}) + \gamma \max_{a_{t+1}} Q_{*}(s_{t+1}, a_{t+1}) \right] \quad (19)$$

**Table 2:** An action-value table for an agent trained to play the card game TwentyTos (the simplified version of Blackjack described in Section ??<sup>[??]</sup>).

State	Action	Value	State	Action	Value	State	Action	Value
PL-DL	<i>Twist</i>	0.039	PH-DL	<i>Twist</i>	−0.666	PM-DH	<i>Twist</i>	−0.668
PL-DL	<i>Stick</i>	−0.623	PH-DL	<i>Stick</i>	0.940	PM-DH	<i>Stick</i>	−0.852
PM-DL	<i>Twist</i>	−0.597	PL-DH	<i>Twist</i>	−0.159	PH-DH	<i>Twist</i>	−0.883
PM-DL	<i>Stick</i>	−0.574	PL-DH	<i>Stick</i>	−0.379	PH-DH	<i>Stick</i>	0.391
BUST	<i>Twist</i>	0.000	TIE	<i>Twist</i>	0.000	WIN	<i>Twist</i>	0.000
BUST	<i>Stick</i>	0.000	TIE	<i>Stick</i>	0.000	WIN	<i>Stick</i>	0.000
LOSE	<i>Twist</i>	0.000				TWENTYTWO	<i>Twist</i>	0.000
LOSE	<i>Stick</i>	0.000				TWENTYTWO	<i>Stick</i>	0.000



$$Q_{\pi}(s_t, a_t) \leftarrow Q_{\pi}(s_t, a_t) + \alpha \underbrace{(G(s_t, a_t) - Q_{\pi}(s_t, a_t))}_{\text{difference between actual and expected returns}} \quad (20)$$

$$Q_{\pi}(s_t, a_t) \leftarrow Q_{\pi}(s_t, a_t) + \alpha \left( \underbrace{r_t + \gamma Q_{\pi}(s_{t+1}, a_{t+1})}_{\text{actual return}} - \underbrace{Q_{\pi}(s_t, a_t)}_{\text{expected return}} \right) \quad (21)$$

# Standard Approach: Q-Learning, Off-Policy Temporal-Difference Learning

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left( r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right) \quad (22)$$

Pseudocode description of the Q-learning algorithm for off-policy temporal-difference learning.

**Require:** a behavior policy,  $\pi$ , that chooses actions

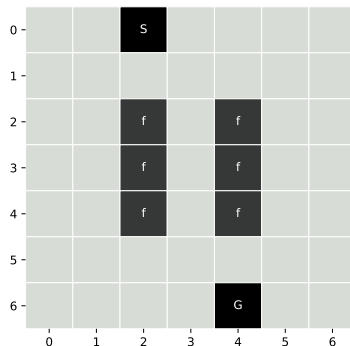
**Require:** an action-value function  $Q$  that performs a lookup into an action-value table with entries for every possible action,  $a$ , and state,  $s$

**Require:** a learning rate,  $\alpha$ , a discount-rate,  $\gamma$ , and a number of episodes to perform

- 1: initialize all entries in the action-value table to random values (except for terminal states which receive a value of 0)
- 2: **for** each episode **do**
- 3:     reset  $s_t$  to the initial agent state
- 4:     **repeat**
- 5:         select an action,  $a_t$ , based on policy,  $\pi$ , current state,  $s_t$ , and action-value function,  $Q$
- 6:         take action  $a_t$  observing reward,  $r_t$ , and new state  $s_{t+1}$
- 7:         update the record in the action-value table for the action,  $a_t$ , just taken in the last state,  $s_t$ , using:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left( r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t) \right) \quad (23)$$

- 8:         let  $s_t = s_{t+1}$
- 9:     **until** agent reaches a terminal state
- 10: **end for**



**Figure 4:** A simple grid world. The start position is annotated with an S and the goal with a G. The squares marked  $f$  denote fire, which is very damaging to an agent.

**Table 3:** A portion of the action-value table for the grid world example at its first initialization.

State	Action	Value	State	Action	Value	State	Action	Value
0-0	<i>up</i>	0.933		...			...	
0-0	<i>down</i>	-0.119	2-0	<i>left</i>	-0.691	6-2	<i>right</i>	0.201
0-0	<i>left</i>	-0.985	2-0	<i>right</i>	0.668	6-3	<i>up</i>	-0.588
0-0	<i>right</i>	0.822	2-1	<i>up</i>	-0.918	6-3	<i>down</i>	0.038
0-1	<i>up</i>	0.879	2-1	<i>down</i>	-0.228	6-3	<i>left</i>	0.859
0-1	<i>down</i>	0.164	2-1	<i>left</i>	-0.301	6-3	<i>right</i>	-0.085
0-1	<i>left</i>	0.343	2-1	<i>right</i>	-0.317	6-4	<i>up</i>	0.000
0-1	<i>right</i>	-0.832	2-2	<i>up</i>	0.633	6-4	<i>down</i>	0.000
0-2	<i>up</i>	0.223	2-2	<i>down</i>	-0.048	6-4	<i>left</i>	0.000
0-2	<i>down</i>	0.582	2-2	<i>left</i>	0.566	6-4	<i>right</i>	0.000
0-2	<i>left</i>	0.672	2-2	<i>right</i>	-0.058	6-5	<i>up</i>	0.321
0-2	<i>right</i>	0.084	2-3	<i>up</i>	0.635	6-5	<i>down</i>	-0.793
0-3	<i>up</i>	-0.308	2-3	<i>down</i>	0.763	6-5	<i>left</i>	-0.267
0-3	<i>down</i>	0.247	2-3	<i>left</i>	-0.121	6-5	<i>right</i>	0.588
0-3	<i>left</i>	0.963	2-3	<i>right</i>	0.562	6-6	<i>up</i>	-0.870
0-3	<i>right</i>	0.455	2-4	<i>up</i>	0.629	6-6	<i>down</i>	-0.720
0-4	<i>up</i>	-0.634	2-4	<i>down</i>	-0.409	6-6	<i>left</i>	0.811
	...			...		6-6	<i>right</i>	0.176

## A Worked Example

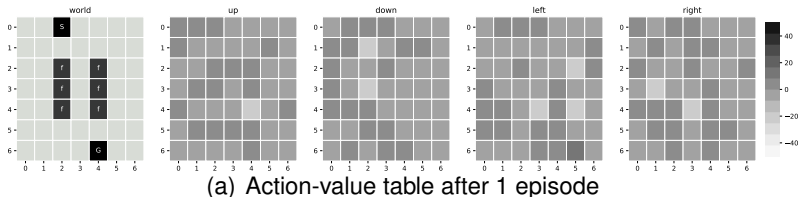
$$\begin{aligned}
 Q(0-3, left) &\leftarrow Q(0-3, left) + \alpha \times (R(0-3, left) + \gamma \times Q(0-2, left) - Q(0-3, left)) \\
 &0.963 + 0.2 \times (-1 + 0.9 \times 0.672 - 0.963) \\
 &0.691
 \end{aligned}$$



\_\_\_\_\_

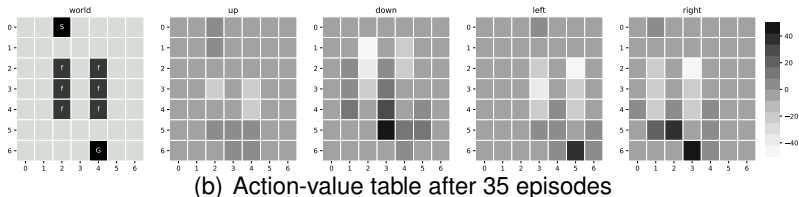
$$Q(6-5, left) \leftarrow Q(6-5, left) + \alpha \times (R(6-5, left) + \gamma \times Q(6-4, up) - Q(6-5, left))$$
$$-0.267 + 0.2 \times (50 + 0.9 \times 0 - -0.267)$$
$$9.786$$

## A Worked Example



**Figure 5:** (a)–(c) The evolution of the entries in the action-value table over episodes of Q-learning off-policy temporal-difference learning across the grid world. (d) The cumulative reward earned from each episode. (e) An illustration of the target policy learned by the agent after 350 episodes. The arrows show the direction with the highest entry in the action-value table for each state. (f) The path the agent will take from the start state to the goal state when greedily following the target policy.

## A Worked Example

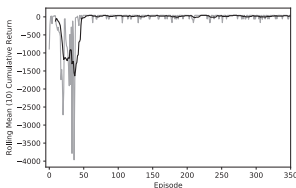


**Figure 6:** (a)–(c) The evolution of the entries in the action-value table over episodes of Q-learning off-policy temporal-difference learning across the grid world. (d) The cumulative reward earned from each episode. (e) An illustration of the target policy learned by the agent after 350 episodes. The arrows show the direction with the highest entry in the action-value table for each state. (f) The path the agent will take from the start state to the goal state when greedily following the target policy.

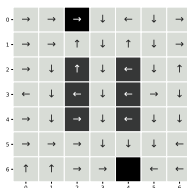


**Figure 7:** (a)–(c) The evolution of the entries in the action-value table over episodes of Q-learning off-policy temporal-difference learning across the grid world. (d) The cumulative reward earned from each episode. (e) An illustration of the target policy learned by the agent after 350 episodes. The arrows show the direction with the highest entry in the action-value table for each state. (f) The path the agent will take from the start state to the goal state when greedily following the target policy.

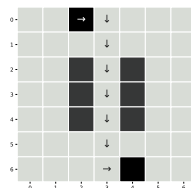
## A Worked Example



(d) Cumulative Reward



(e) Policy



(f) Offline Path

**Figure 8:** (a)–(c) The evolution of the entries in the action-value table over episodes of Q-learning off-policy temporal-difference learning across the grid world. (d) The cumulative reward earned from each episode. (e) An illustration of the target policy learned by the agent after 350 episodes. The arrows show the direction with the highest entry in the action-value table for each state. (f) The path the agent will take from the start state to the goal state when greedily following the target policy.

**Table 4:** A portion of the action-value table for the grid world example after 350 episodes of Q-learning have elapsed.

State	Action	Value	State	Action	Value	State	Action	Value
0-0	<i>up</i>	-1.627	...	...	...	...	...	...
0-0	<i>down</i>	-1.255	2-0	<i>left</i>	-1.583	6-2	<i>right</i>	40.190
0-0	<i>left</i>	-1.655	2-0	<i>right</i>	-1.217	6-3	<i>up</i>	34.375
0-0	<i>right</i>	-1.000	2-1	<i>up</i>	-1.493	6-3	<i>down</i>	40.206
0-1	<i>up</i>	1.302	2-1	<i>down</i>	4.132	6-3	<i>left</i>	24.784
0-1	<i>down</i>	-1.900	2-1	<i>left</i>	-1.643	6-3	<i>right</i>	50.000
0-1	<i>left</i>	-1.900	2-1	<i>right</i>	-36.301	6-4	<i>up</i>	0.000
0-1	<i>right</i>	15.173	2-2	<i>up</i>	13.247	6-4	<i>down</i>	0.000
0-2	<i>up</i>	13.299	2-2	<i>down</i>	-46.862	6-4	<i>left</i>	0.000
0-2	<i>down</i>	12.009	2-2	<i>left</i>	-0.858	6-4	<i>right</i>	0.000
0-2	<i>left</i>	8.858	2-2	<i>right</i>	-1.157	6-5	<i>up</i>	-0.353
0-2	<i>right</i>	18.698	2-3	<i>up</i>	16.973	6-5	<i>down</i>	-0.793
0-3	<i>up</i>	13.921	2-3	<i>down</i>	29.366	6-5	<i>left</i>	36.823
0-3	<i>down</i>	21.886	2-3	<i>left</i>	-88.492	6-5	<i>right</i>	-0.342
0-3	<i>left</i>	15.900	2-3	<i>right</i>	-77.447	6-6	<i>up</i>	-0.870
0-3	<i>right</i>	13.846	2-4	<i>up</i>	-1.016	6-6	<i>down</i>	-0.720
0-4	<i>up</i>	1.637	2-4	<i>down</i>	-20.255	6-6	<i>left</i>	1.008
...	...	...	...	...	...	6-6	<i>right</i>	-0.802

# Extensions and Variations

Pseudocode description of the **SARSA** algorithm for on-policy temporal-difference learning.

**Require:** a behavior policy,  $\pi$ , that chooses actions

**Require:** an action-value function  $Q$  that performs a lookup into an action-value table with entries for every possible action,  $a$ , and state,  $s$

**Require:** a learning rate,  $\alpha$ , a discount-rate,  $\gamma$ , and a number of episodes to perform

- 1: initialize all entries in the action-value table to random values (except for terminal states which receive a value of 0)
- 2: **for** each episode **do**
- 3:     reset  $s_t$  to the initial agent state
- 4:     select an action,  $a_t$ , based on policy,  $\pi$ , current state,  $s_t$ , and action-value function,  $Q$
- 5:     **repeat**
- 6:         take action  $a_t$  observing reward,  $r_t$ , and new state,  $s_{t+1}$
- 7:         select the next action,  $a_{t+1}$ , based on policy,  $\pi$ , new state,  $s_{t+1}$ , and action-value function,  $Q$
- 8:         update the record in the action-value table for the action,  $a_t$ , just taken in the last state,  $s_t$ , using:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha (r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

- 9:         let  $s_t = s_{t+1}$  and  $a_t = a_{t+1}$
- 10:     **until** agent reaches terminal state
- 11: **end for**



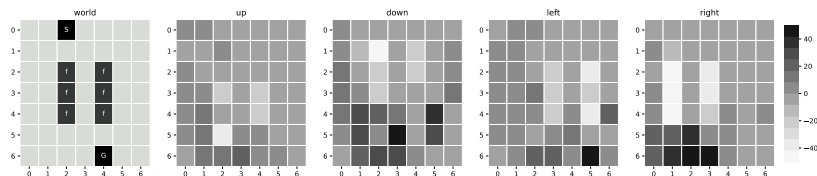
# SARSA, On-Policy Temporal-Difference Learning

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha (r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t))$$

# SARSA, On-Policy Temporal-Difference Learning

$$\begin{aligned}
 Q(0-3, left) &\leftarrow Q(0-3, left) + \alpha \times (R(0-3, left) + \gamma \times Q(0-2, down) - Q(0-3, left)) \\
 &0.963 + 0.2 \times (-1 + 0.9 \times 0.582 - 0.963) \\
 &0.675
 \end{aligned}$$

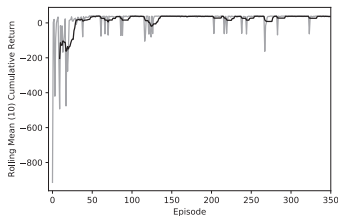
# SARSA, On-Policy Temporal-Difference Learning



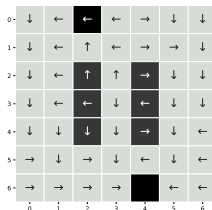
(a) Action-value table after 350 episodes

**Figure 9:** (a) A visualization of the final action-value table for an agent trained using SARSA on-policy temporal-difference learning across the grid world after 350 episodes. (b) The cumulative reward earned from each episode. (c) An illustration of the target policy learned by the agent after 350 episodes. The arrows show the direction with the highest entry in the action-value table for each state. (d) The path the agent will take from the start state to the goal state when greedily following the target policy.

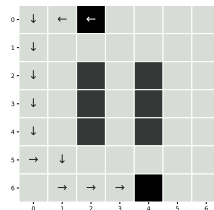
## SARSA, On-Policy Temporal-Difference Learning



(b) Cumulative Reward

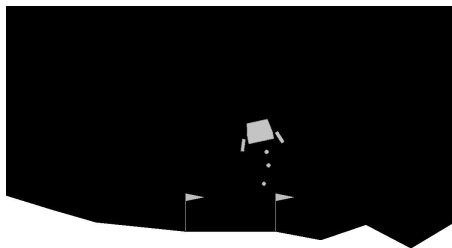


(c) Policy

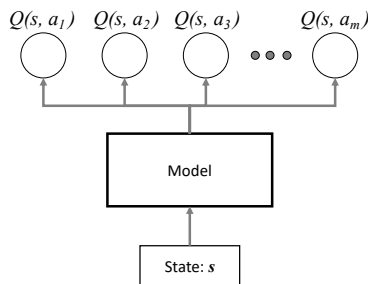


(d) Offline Path

**Figure 10:** (a) A visualization of the final action-value table for an agent trained using SARSA on-policy temporal-difference learning across the grid world after 350 episodes. (b) The cumulative reward earned from each episode. (c) An illustration of the target policy learned by the agent after 350 episodes. The arrows show the direction with the highest entry in the action-value table for each state. (d) The path the agent will take from the start state to the goal state when greedily following the target policy.



**Figure 11:** The Lunar Lander environment. The aim of the game is to control the spaceship starting from the top of the world and attempting to land on the landing pad.



**Figure 12:** Framing the action-value function as a prediction problem.

$$\mathbb{M}(s_t, a_t) \approx Q_{\pi}(s_t, a_t) \quad (24)$$

$$\mathbb{M}(s_t) \approx Q_{\pi}(s_t, a_t) \quad (25)$$

(26)

(27)

(28)

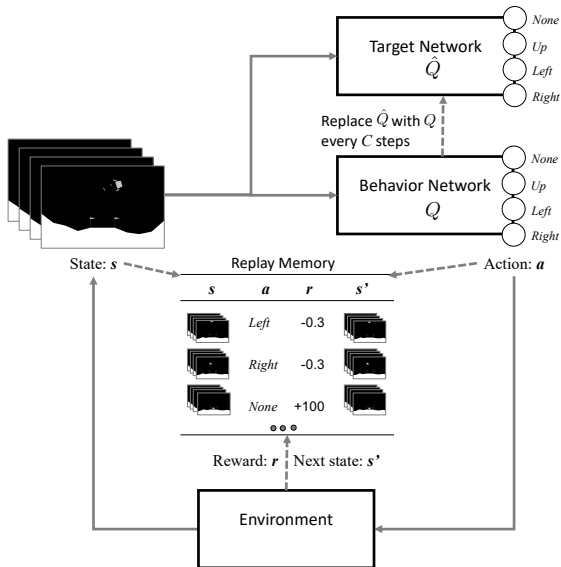


Pseudocode description of the **naive neural Q-learning** algorithm.

- 1: initialize weights,  $\mathbf{W}$ , in action-value function network,  $Q_{\mathbb{M}}$ , to random values
- 2: **for** each episode **do**
- 3:     reset  $s_t$  to the initial agent state
- 4:     **repeat**
- 5:         select action,  $a_t$ , based on policy,  $\pi$ , the current state,  $s_t$ , and action-value network output,  $Q_{\mathbb{M}}(s_t, a_t)$
- 6:         take action  $a_t$  and observing reward,  $r_t$ , and new state,  $s_{t+1}$
- 7:         generate a target feature

$$t = r_t + \gamma \max_{a_{t+1}} Q_{\mathbb{M}}(s_{t+1}, a_{t+1})$$

- 8:         perform an iteration of stochastic gradient descent using a single training instance  $\langle s_t, t \rangle$
- 9:     **until** agent reaches terminal state
- 10: **end for**



**Figure 13:** An illustration of the DQN algorithm including experience replay and target network freezing.

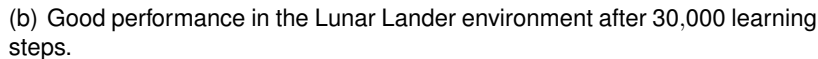
Pseudocode description of the **deep Q network** (DQN) algorithm.

- 1: initialize replay memory  $\mathcal{D}$  with  $N$  steps based on random actions
- 2: initialize weights,  $\mathbf{W}$  in behavior action-value function network,  $Q_{\mathcal{M}}$ , to random values
- 3: initialize weights,  $\widehat{\mathbf{W}}$  in target action-value function network,  $\widehat{Q}_{\mathcal{M}}$  to  $\mathbf{W}$
- 4: **for** each episode **do**
- 5:     reset  $s_t$  to the initial agent state
- 6:     **repeat**
- 7:         select action,  $a_t$ , based on agent's policy,  $\pi$ , the current state,  $s_t$ , and behavior network output,  $Q_{\mathcal{M}}(s_t, a_t)$
- 8:         take action  $a_t$  and observe the resulting reward,  $r_t$ , and new state,  $s_{t+1}$
- 9:         add tuple  $\langle s = s_t, a = a_t, r = r_t, s' = s_{t+1} \rangle$  as a new instance in  $\mathcal{D}$
- 10:         randomly select a mini-batch of  $b$  instances from  $\mathcal{D}$  to give  $\mathcal{D}_b$
- 11:         generate target feature values for each instance,  $\langle s_i, a_i, r_i, s'_i \rangle$  in  $\mathcal{D}_b$
- 12:         as:
$$t_i = r_i + \gamma \max_{a'} \widehat{Q}_{\mathcal{M}}(s'_i, a')$$
- 13:         perform an iteration of mini-batch gradient descent using  $\mathcal{D}_b$
- 14:         every  $C$  steps let  $\widehat{Q}_{\mathcal{M}} = Q_{\mathcal{M}}$
- 15:     **until** agent reaches terminal state
- 16: **end for**

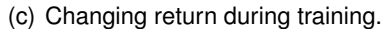


(a) Poor performance in the Lunar Lander environment early in the learning process.

**Figure 14:** (a) Frames from an episode early in the training process in which the agent performs poorly. (b) Frames from an episode near the end of the learning process where the agent is starting to be very effective. (c) Changing episode returns during DQN training. The gray line shows a 50-episode moving average to better highlight the trend.



**Figure 15:** (a) Frames from an episode early in the training process in which the agent performs poorly. (b) Frames from an episode near the end of the learning process where the agent is starting to be very effective. (c) Changing episode returns during DQN training. The gray line shows a 50-episode moving average to better highlight the trend.



**Figure 16:** (a) Frames from an episode early in the training process in which the agent performs poorly. (b) Frames from an episode near the end of the learning process where the agent is starting to be very effective. (c) Changing episode returns during DQN training. The gray line shows a 50-episode moving average to better highlight the trend.

# Summary

- In a reinforcement learning scenario an **agent** inhabiting an **environment** attempts to achieve a **goal** by taking a sequence of **actions** to move it between **states**.
- On completion of each action the agent receives an immediate scalar **reward** indicating whether the outcome of the action was positive or negative and to what degree.
- To choose which action to take in a given state the agent uses a **policy**.
- Policies rely on being able to assess the **expected return** of taking an action in a particular state, and an **action-value function** is used to calculate this.



- The learning approaches described here are **value-based** and **model-free**.
- **Temporal-difference learning**, and its **Q-learning (off-policy)** and **SARSA (on-policy)** variants, is an important tabular methods for reinforcement learning.
- **Deep Q networks** are an approximate approach to temporal difference learning based on deep neural networks.
- One overarching point about reinforcement learning that is worth mentioning is that it comes at the cost of hugely increased computation.

# Further Reading

- Key texts on intelligent agents: (Wooldridge and Jennings, 1995; Wooldridge, 2009; Mac Namee, 2009).
- Key early foundation setting work: (Howard, 1960; Bellman, 1957a,b; Michie, 1961, 1963).
- Sutton and Barto's textbook has remained the definitive work on reinforcement learning (Sutton and Barto, 2018).
- For a broader discussion on the challenges of defining reward functions: (Asimov, 1950; Bostrom, 2003).
- For more recent advances at the junction of reinforcement learning and deep learning: (Sejnowski, 2018; Sutton and Barto, 2018).

- 1 Big Idea
- 2 Fundamentals
- 3 Standard Approach: Q-Learning, Off-Policy Temporal-Difference Learning
- 4 Extensions and Variations
- 5 Summary
- 6 Further Reading

Asimov, Isaac. 1950. *I, robot*. Gnome Press.

Bellman, R. E. 1957a. *Dynamic programming*. Princeton University Press.

Bellman, R. E. 1957b. A markov decision process. *Journal of Mathematical Mechanics* 6: 679–684.

Bostrom, Nick. 2003. Ethical issues in advanced artificial intelligence. In *Science fiction and philosophy: From time travel to superintelligence*, 277–284. Wiley-Blackwell.

Howard, R. 1960. *Dynamic programming and Markov processes*. MIT Press.

Mac Namee, Brian. 2009. Agent based modeling in computer graphics and games. In *Encyclopedia of complexity and systems science*, ed. R. A. Meyers. Dublin Institute of Technology.

Michie, D. 1961. Trial and error. In *Science survey, part 2*, eds. S. A. Barnett and A. McLaren, 129–145. Penguin.

- Michie, D. 1963. Experiments on the mechanisation of game learning. *Computer Journal* 1: 232–263.
- Sejnowski, Terrence J. 2018. *The deep learning revolution*. MIT Press.
- Sutton, Richard S., and Andrew G. Barto. 2018. *Reinforcement learning: An introduction*. MIT Press.
- Wooldridge, Michael. 2009. *An introduction to multiagent systems*. Wiley.
- Wooldridge, Michael, and Nicholas R. Jennings. 1995. Intelligent agents: Theory and practice. *The Knowledge Engineering Review* 10 (2): 115–152.