Beyond Prediction: Reinforcement Learning

John D. Kelleher and Brian Mac Namee and Aoife D'Arcy

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

Intelligent Agents

$$(o_1, a_1, r_1), (o_2, a_2, r_2), (o_3, a_3, r_3), \ldots, (o_e, a_e, r_e) \tag{1}$$

Intelligent Agents

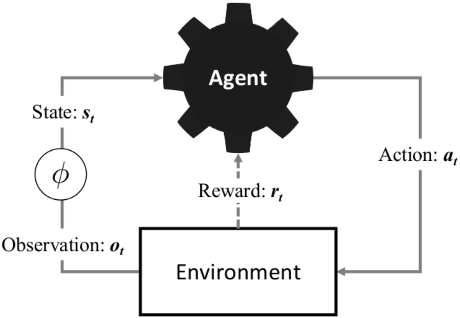


Figure 1: An agent behaving in an environment and the observation, reward, action cycle. The transition from observations of the environment to a state is shown by the state generation function, ϕ.

Intelligent Agents

$$(s_1, a_1, r_1), (s_2, a_2, r_2), (s_3, a_3, r_3), \ldots, (s_e, a_e, r_e) \tag{2}$$

$$G = r_t + r_{t+1} + r_{t+2} + r_{t+3} + \ldots + r_e \tag{3}$$

$$a_t = \pi(s_t) \tag{4}$$

$$P(A_t = a \mid S_t = s) = \pi(S_t = s) \tag{5}$$

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

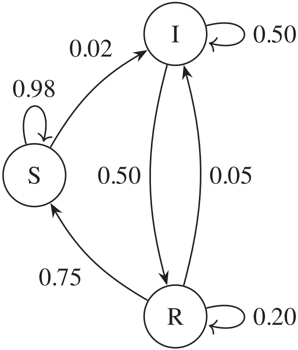
$$Q_\pi(s_t, a_t) = E_\pi[r_t + r_{t+1} + r_{t+2} + r_{t+3} + \ldots + r_e \mid s_t, a_t] \tag{7}$$

$$G_{\gamma} = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \gamma^3 r_{t+3} + \ldots + \gamma^{e-t} r_e \tag{8}$$

$$G_{\gamma=0.1} = r_t + 0.1 \times r_{t+1} + 0.01 \times r_{t+2} + 0.001 \times r_{t+3} + \ldots$$

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

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



(a) S-I-R Markov process

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

(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.

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

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

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

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

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

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	*Twist*	0
2	2♥ 7♣ *K*♣	(19)	8♥	(8)	PH-DH	*Stick*	+1
3	2♥ 7♣ *K*♣	(19)	8♥ *Q*♦	(18)	WIN		
1	4♠ *A*♥	(15)	*Q*♥	(10)	PM-DH	*Twist*	-1
2	4♠ *A*♥ 9♣	(24)	*Q*♥	(10)	BUST		
1	2♦ 4♦	(6)	3♥	(3)	PL-DL	*Twist*	0
2	2♦ 4♦ 3♥	(9)	3♥	(3)	PL-DL	*Twist*	0
3	2♦ 4♦ 3♥ 6♣	(15)	3♥	(3)	PM-DL	*Twist*	0
4	2♦ 4♦ 3♥ 6♣ 6♦	(21)	3♥	(3)	PH-DL	*Stick*	0
5	2♦ 4♦ 3♥ 6♣ 6♦	(21)	3♥ 7♥ *A*♠	(21)	TIE		
1	*Q*♦ *J*♣	(20)	*A*♥	(11)	PH-DH	*Stick*	+1
2	*Q*♦ *J*♣	(20)	*A*♣ 5♣ *Q*♠	(26)	WIN		
1	*A*♦ *A*♥	(22)	2♥	(2)	PH-DL	*Stick*	+2
2	*A*♦ *A*♥	(22)	2♥	(2)	TWENTYTWO		

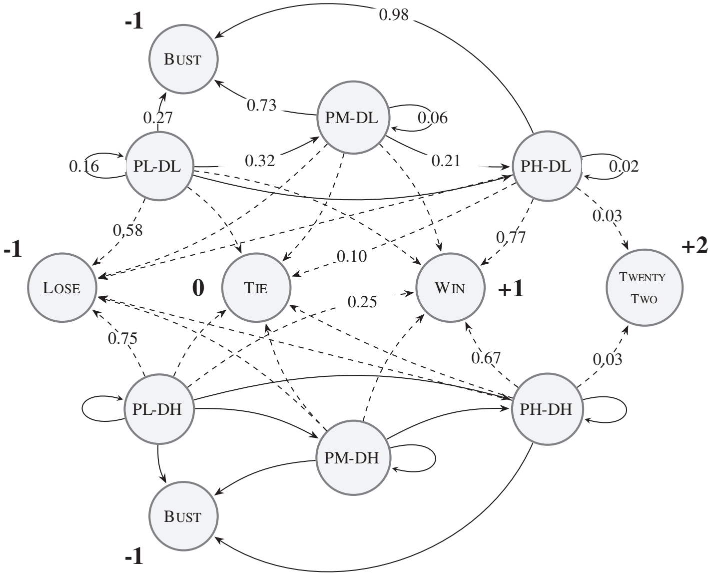


Figure 3: A Markov decision process representation for TwentyTwos, a simplified version of the card game Blackjack.

Markov Decision Processes

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

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

Markov Decision Processes

$$
\mathcal{P}^{Twist} = \begin{array}{c} \\ \textsc{PL-DL} \\ \textsc{PM-DL} \\ \textsc{PH-DL} \\ \textsc{PL-DH} \\ \textsc{PM-DH} \\ \textsc{PH-DH} \\ \textsc{Bust} \\ \textsc{Lose} \\ \textsc{Tie} \\ \textsc{Win} \\ \textsc{TwentyTwo} \end{array}
\begin{array}{c}
\begin{array}{ccccccccccc} \textsc{PL-DL} & \textsc{PM-DL} & \textsc{PH-DL} & \textsc{PL-DH} & \textsc{PM-DH} & \textsc{PH-DH} & \textsc{Bust} & \textsc{Lose} & \textsc{Tie} & \textsc{Win} & \textsc{TwentyTwo} \end{array} \\
\begin{bmatrix}
0.16 & 0.32 & 0.34 & 0.00 & 0.00 & 0.00 & 0.17 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.06 & 0.29 & 0.00 & 0.00 & 0.00 & 0.65 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.08 & 0.00 & 0.00 & 0.00 & 0.92 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.16 & 0.32 & 0.34 & 0.17 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.06 & 0.29 & 0.65 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.08 & 0.92 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00
\end{bmatrix}
\end{array}
$$

Markov Decision Processes

$$
\mathcal{P}^{Stick} = \begin{array}{c} \\ \text{PL-DL} \\ \text{PM-DL} \\ \text{PH-DL} \\ \text{PL-DH} \\ \text{PM-DH} \\ \text{PH-DH} \\ \text{BUST} \\ \text{LOSE} \\ \text{TIE} \\ \text{WIN} \\ \text{TWENTYTWO} \end{array}
\begin{array}{c}
\begin{array}{ccccccccccc} \text{PL-DL} & \text{PM-DL} & \text{PH-DL} & \text{PL-DH} & \text{PM-DH} & \text{PH-DH} & \text{BUST} & \text{LOSE} & \text{TIE} & \text{WIN} & \text{TWENTYTWO} \end{array} \\
\begin{bmatrix}
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.68 & 0.00 & 0.32 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.58 & 0.06 & 0.36 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.20 & 0.10 & 0.68 & 0.03 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.81 & 0.00 & 0.19 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.73 & 0.06 & 0.21 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.22 & 0.15 & 0.60 & 0.03 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 \\
0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 0.00
\end{bmatrix}
\end{array}
$$

$$
\begin{aligned}
Q_\pi(s_t, a_t) &= E_\pi\left[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \ldots + \gamma^3 r_\infty \mid s_t, a_t\right] \\
&= E_\pi\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t, a_t\right] \qquad (14) \\
&= E_\pi\left[r_t + \gamma \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t, a_t\right] \qquad (15)
\end{aligned}
$$

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 TwentyTwos (the simplified version of Blackjack described in Section **??**[??]).

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

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

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

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

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

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

Require: a behavior policy, π, 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, α, a discount-rate, γ, and a number of episodes to perform

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

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

        let $s_t = s_{t+1}$
    until agent reaches a terminal state
end for

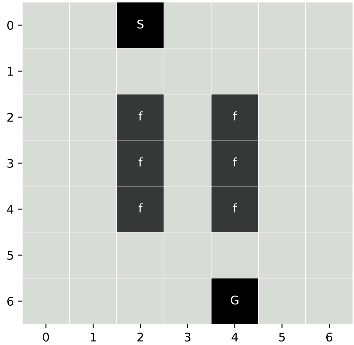


Figure 4: A simple grid world. The start position is annotated with an *S* and the goal with a *G*. The squares marked *f* denote fire, which is very damaging to an agent.

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

State	Action	Value
0-0	*up*	0.933
0-0	*down*	−0.119
0-0	*left*	−0.985
0-0	*right*	0.822
0-1	*up*	0.879
0-1	*down*	0.164
0-1	*left*	0.343
0-1	*right*	−0.832
0-2	*up*	0.223
0-2	*down*	0.582
0-2	*left*	0.672
0-2	*right*	0.084
0-3	*up*	−0.308
0-3	*down*	0.247
0-3	*left*	0.963
0-3	*right*	0.455
0-4	*up*	−0.634
	...	

State	Action	Value
	...	
2-0	*left*	−0.691
2-0	*right*	0.668
2-1	*up*	−0.918
2-1	*down*	−0.228
2-1	*left*	−0.301
2-1	*right*	−0.317
2-2	*up*	0.633
2-2	*down*	−0.048
2-2	*left*	0.566
2-2	*right*	−0.058
2-3	*up*	0.635
2-3	*down*	0.763
2-3	*left*	−0.121
2-3	*right*	0.562
2-4	*up*	0.629
2-4	*down*	−0.409
	...	

State	Action	Value
	...	
6-2	*right*	0.201
6-3	*up*	−0.588
6-3	*down*	0.038
6-3	*left*	0.859
6-3	*right*	−0.085
6-4	*up*	0.000
6-4	*down*	0.000
6-4	*left*	0.000
6-4	*right*	0.000
6-5	*up*	0.321
6-5	*down*	−0.793
6-5	*left*	−0.267
6-5	*right*	0.588
6-6	*up*	−0.870
6-6	*down*	−0.720
6-6	*left*	0.811
6-6	*right*	0.176

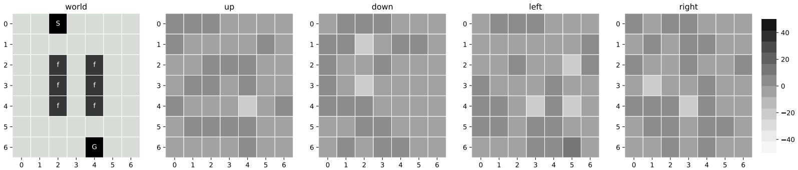
A Worked Example

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

A Worked Example

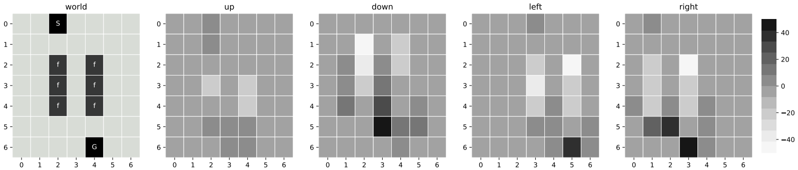
$$
\begin{aligned}
Q(6\text{-}5, \textit{left}) \leftarrow & Q(6\text{-}5, \textit{left}) + \alpha \times (R(6\text{-}5, \textit{left}) + \gamma \times Q(6\text{-}4, \textit{up}) - Q(6\text{-}5, \textit{left})) \\
& -0.267 + 0.2 \times (50 + 0.9 \times 0 - -0.267) \\
& 9.786
\end{aligned}
$$

A Worked Example



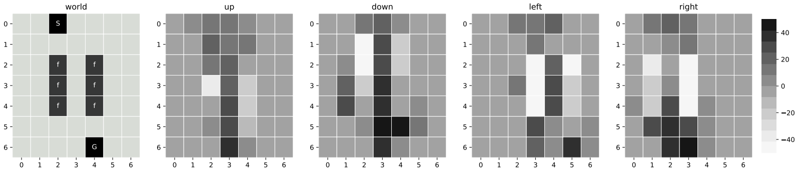
(a) Action-value table after 1 episode

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.



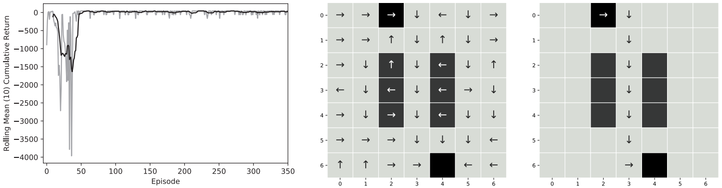
(b) Action-value table after 35 episodes

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.



(c) Action-value table after 350 episodes

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.



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

Extensions and Variations

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

Require: a behavior policy, π, 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, α, a discount-rate, γ, and a number of episodes to perform

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

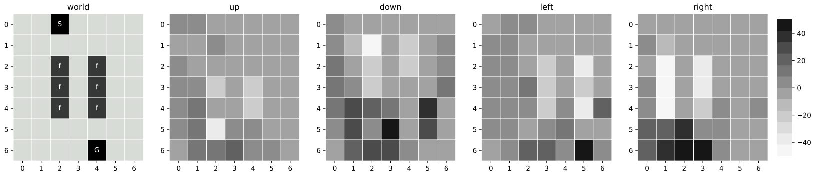
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left(r_t + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right)$$

        let $s_t = s_{t+1}$ and $a_t = a_{t+1}$
    until agent reaches terminal state
end for

SARSA, On-Policy Temporal-Difference Learning

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

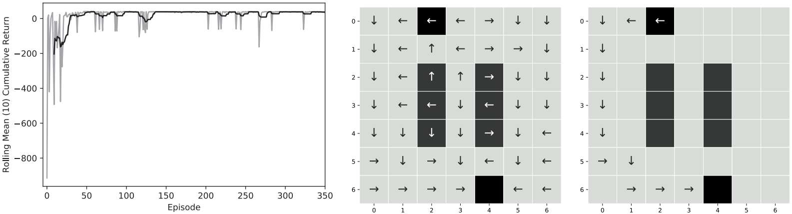
$$
\begin{aligned}
\mathbb{Q}(0\text{-}3, \textit{left}) \leftarrow & \mathbb{Q}(0\text{-}3, \textit{left}) + \alpha \times (R(0\text{-}3, \textit{left}) + \gamma \times \mathbb{Q}(0\text{-}2, \textit{down}) - \mathbb{Q}(0\text{-}3, \textit{left})) \\
& 0.963 + 0.2 \times (-1 + 0.9 \times 0.582 - 0.963)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



(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.

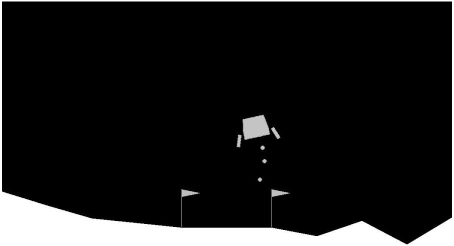


Figure 11: The Lunar Lander environment. The aim of the game is to control the spaceship starting from the top of the world and attempting to land on the landing pad.

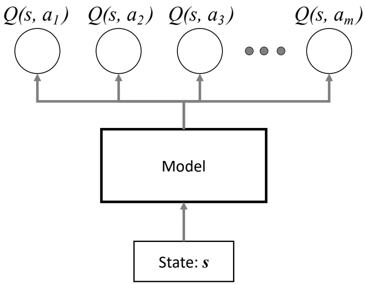


Figure 12: Framing the action-value function as a prediction problem.

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

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

$$\mathcal{L}(Q_{\mathbb{M}_{\mathbf{W}}}(s_t)) = \left(t_i - Q_{\mathbb{M}_{\mathbf{W}}}(s_t, a_t)\right)^2 \tag{26}$$

$$= \left(r_t + \gamma \max_{a_{t+1}} Q_{\mathbb{M}_{\mathbf{W}}}(s_{t+1}, a_{t+1}) - Q_{\mathbb{M}_{\mathbf{W}}}(s_t, a_t)\right)^2 \tag{27}$$

$$\frac{\partial \mathcal{L}(Q_{\mathbb{M}_{\mathbf{W}}}(s_t, a_t))}{\partial \mathbf{W}} = \left(r_t + \gamma \max_{a_{t+1}} Q_{\mathbb{M}_{\mathbf{W}}}(s_{t+1}, a_{t+1}) - Q_{\mathbb{M}_{\mathbf{W}}}(s_t, a_t)\right) \frac{\partial Q_{\mathbb{M}_{\mathbf{W}}}(s_t, a_t)}{\partial \mathbf{W}} \tag{28}$$

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

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

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

        perform an iteration of stochastic gradient descent using a single training instance $< s_t, t >$
    until agent reaches terminal state
end for

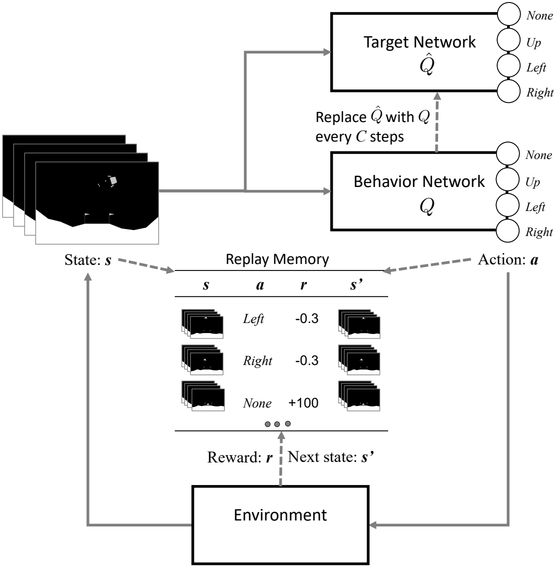


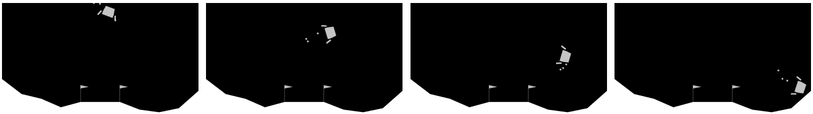
Figure 13: An illustration of the DQN algorithm including experience replay and target network freezing.

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

initialize replay memory $\mathcal{D}$ with N steps based on random actions
initialize weights, $\mathbf{W}$ in behavior action-value function network, $Q_{\mathbb{M}}$, to random values
initialize weights, $\widehat{\mathbf{W}}$ in target action-value function network, $\widehat{Q_{\mathbb{M}}}$ to $\mathbf{W}$
for each episode **do**
    reset s_t to the initial agent state
    repeat
        select action, a_t, based on agent's policy, π, the current state, s_t, and behavior network output, $Q_{\mathbb{M}}(s_t, a_t)$
        take action a_t and observe the resulting reward, r_t, and new state, s_{t+1}
        add tuple $\langle s = s_t, a = a_t, r = r_t, s' = s_{t+1} \rangle$ as a new instance in $\mathcal{D}$
        randomly select a mini-batch of b instances from $\mathcal{D}$ to give $\mathcal{D}_b$
        generate target feature values for each instance, $\langle s_i, a_i, r_i, s'_i \rangle$ in $\mathcal{D}_b$ as:

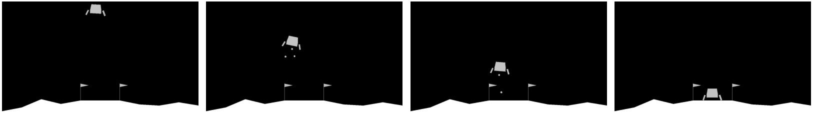
$$t_i = r_i + \gamma \max_{a'} \widehat{Q_{\mathbb{M}}} \left(s'_i, a' \right)$$

        perform an iteration of mini-batch gradient descent using $\mathcal{D}_b$
        every C steps let $\widehat{Q_{\mathbb{M}}} = Q_{\mathbb{M}}$
    until agent reaches terminal state
end for



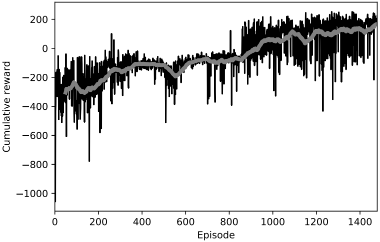
(a) Poor performance in the Lunar Lander environment early in the learning process.

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.



(b) Good performance in the Lunar Lander environment after 30,000 learning steps.

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.



(c) Changing return during training.

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: (**???**).
- Key early foundation setting work: (**?????**).
- Sutton and Barto's textbook has remained the definitive work on reinforcement learning (**?**).
- For a broader discussion on the challenges of defining reward functions: (**??**).
- For more recent advances at the junction of reinforcement learning and deep learning: (**??**).