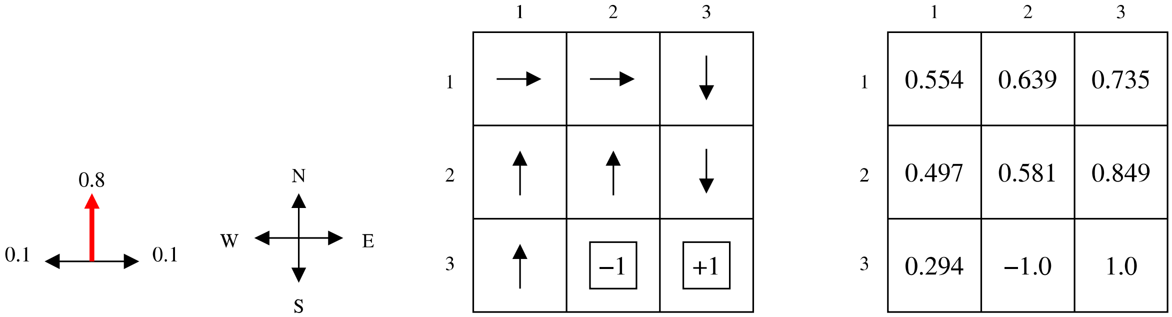
6.825 Reinforcement Learning Examples

TAs: Meg Aycinena and Emma Brunskill

1 Mini Grid World



	1	2	3
1	→	→	↓
2	↑	↑	↓
3	↑	−1	+1

	1	2	3
1	0.554	0.639	0.735
2	0.497	0.581	0.849
3	0.294	−1.0	1.0

(a) Transition model of 3x3 world.

(b) Optimal policy π of 3x3 world.

(c) Utility function of optimal policy of 3x3 world.

Figure 1: A 3x3 grid world.

In the mini grid world shown in Figure 1, there are two terminal states: state (3,2) with a negative reward of -1, and (3,3) with a positive reward of $+1$. According to the transition model shown in Figure 1(a), an action succeeds with probability 0.8, and goes to the left or right of the intended direction with probability 0.1, respectively. The optimal policy π is shown in Figure 1(b), and the correct utility function the optimal policy is shown in Figure 1(c).

2 Passive Learning

We conduct a series of three trials in this environment. We start in the start state (1,1), take actions according to the fixed policy π given in Figure 1(b), and end once a terminal state is reached. The trials are as follows:

Trial	⟨state,reward⟩ series
1	$\langle(1,1),0\rangle \xrightarrow{E} \langle(1,2),0\rangle \xrightarrow{E} \langle(1,3),0\rangle \xrightarrow{S} \langle(2,3),0\rangle \xrightarrow{S} \langle(3,3),1\rangle$
2	$\langle(1,1),0\rangle \xrightarrow{E} \langle(1,2),0\rangle \xrightarrow{S} \langle(2,2),0\rangle \xrightarrow{N} \langle(1,2),0\rangle \xrightarrow{E} \langle(1,3),0\rangle \xrightarrow{S} \langle(2,3),0\rangle \xrightarrow{S} \langle(3,3),1\rangle$
3	$\langle(1,1),0\rangle \xrightarrow{S} \langle(2,1),0\rangle \xrightarrow{N} \langle(1,1),0\rangle \xrightarrow{E} \langle(1,2),0\rangle \xrightarrow{E} \langle(1,3),0\rangle \xrightarrow{S} \langle(2,3),0\rangle \xrightarrow{S} \langle(3,3),1\rangle$

2.1 Passive ADP

2.1.1 Learning the Transition Model

First, we must learn the transition model $Pr(s' \mid s, a)$. This is easy, because the world is fully observable. For simplicity, we use simple ML estimation (counting), but a more robust approach would incorporate Laplacian correction or another Bayesian prior. The ML estimate of the transition function can be computed as:

$$Pr(s' \mid s, a) = \frac{N(s, a, s')}{\sum_{s''} N(s, a, s'')} \tag{1}$$

where $N(s, a, s')$ denotes the number of times the state s' was reached when taking action a in state s.

The ML estimate of the transition function, using the trials above, is:

	$(1,1)$	$(1,2)$	$(1,3)$	$(2,1)$	$(2,2)$	$(2,3)$	$(3,1)$	$(3,2)$	$(3,3)$
$(1,1), E$	0	0.75	0	0.25	0	0	0	0	0
$(1,2), E$	0	0	0.75	0	0.25	0	0	0	0
$(1,3), S$	0	0	0	0	0	1	0	0	0
$(2,1), N$	1	0	0	0	0	0	0	0	0
$(2,2), N$	0	1	0	0	0	0	0	0	0
$(2,3), S$	0	0	0	0	0	0	0	0	1
$(3,1), E$	?	?	?	?	?	?	?	?	?

Note that we only learn distributions over state-action pairs that are relevant, given the fixed policy. Also, we know nothing about state (3,1) because we have never seen it during our trials.

2.1.2 Learning the Utility Function

Now, we learn the utility function on states, by plugging in our learned transition model and the observed rewards, and then solving the Bellman equations:

$$V^{\pi}(s) = R(s) + \gamma \sum_{s'} Pr(s' \mid s, \pi(s)) V^{\pi}(s') \tag{2}$$

where $\pi(s)$ denotes the action determined by the policy π in state s. Because there is no max operator in this expression, we have a simple set of $|S|$ linear equations in $|S|$ unknowns, which can be solved in closed form. Let the discount factor $\gamma = 0.9$.

$$\begin{aligned}
V^{\pi}((1,1)) &= 0 + 0.9(0.75V^{\pi}((1,2)) + 0.25V^{\pi}((2,1))) \\
V^{\pi}((1,2)) &= 0 + 0.9(0.75V^{\pi}((1,3)) + 0.25V^{\pi}((2,2))) \\
V^{\pi}((1,3)) &= 0 + 0.9(V^{\pi}((2,3))) \\
V^{\pi}((2,1)) &= 0 + 0.9(V^{\pi}((1,1))) \\
V^{\pi}((2,2)) &= 0 + 0.9(V^{\pi}((1,2))) \\
V^{\pi}((2,3)) &= 0 + 0.9(V^{\pi}((3,3))) \\
V^{\pi}((3,3)) &= 1.0
\end{aligned}$$

	1	2	3
1	0.54	0.66	0.81
2	0.48	0.59	0.9
3	?	?	1.0

Figure 2: Learned $V^\pi(s)$, using passive ADP.

In this case, because we never saw states (3,1) or (3,2), we have two fewer equations and unknowns.

Solving this system yields the values shown in Figure 2. They are actually quite a good approximation to the true underlying utilities.

2.2 Passive TD Learning

In passive TD learning, we use each observed transition to adjust the utility values of observed states to agree with the constraint equations given in Equation (2). The update equation is:

$$V^\pi(s) \leftarrow V^\pi(s) + \alpha(N(s))(R(s) + \gamma V^\pi(s') - V^\pi(s)) \tag{3}$$

where $N(s)$ denotes the number of times we have been in state s, and $\alpha(n)$ is a function that increases and n decreases. Here we use $\alpha(n) = \frac{1}{n}$.

Initialize all $V^\pi(s)$ to 0. (This can also be done randomly.) Then, for each trial, we perform the update equation for each $\langle s, a, s' \rangle$ tuple. We also keep a table of $N(s)$ counts.

Trial 1

The initial utilities are all zero, and the only observed reward is in the final state in the trial, state $(3,3)$. Thus, the only non-trivial update while performing the updates for trial 1 is for state $(3,3)$:

$$\begin{aligned} V^\pi((3,3)) &\leftarrow 0 + 1(1 + 0.9(0) - 0) \\ &\leftarrow 1 \end{aligned}$$

The resulting $V^\pi(s)$ is shown in Figure 3(b).

Trial 2

The only non-zero update in this trial is for state $(2,3)$:

$$\begin{aligned} V^\pi((2,3)) &\leftarrow 0 + \frac{1}{2}(0 + 0.9(1) - 0) \\ &\leftarrow 0.45 \end{aligned}$$

The resulting $V^\pi(s)$ is shown in Figure 3(d).

	1	2	3
1	1	1	1
2	0	0	1
3	0	0	1

(a) $N(s)$, after trial 1.

	1	2	3
1	0	0	0
2	0	0	0
3	0	0	1.0

(b) $V^\pi(s)$, after trial 1.

	1	2	3
1	2	3	2
2	0	1	2
3	0	0	2

(c) $N(s)$, after trial 2.

	1	2	3
1	0	0	0
2	0	0	0.45
3	0	0	1.0

(d) $V^\pi(s)$, after trial 2.

	1	2	3
1	4	4	3
2	1	1	3
3	0	0	3

(e) $N(s)$, after trial 3.

	1	2	3
1	0	0	0.135
2	0	0	0.6
3	0	0	1.0

(f) $V^\pi(s)$, after trial 3.

Figure 3: Learned utilities, using passive TD learning.

Trial 3

Now we have two non-zero updates during this trial; first state $(1,3)$, and then another update to state $(2,3)$:

$$
\begin{aligned}
V^\pi((1,3)) &\leftarrow 0 + \frac{1}{3}(0 + 0.9(0.45) - 0) \\
&\leftarrow 0.135 \\
V^\pi((2,3)) &\leftarrow 0.45 + \frac{1}{3}(0 + 0.9(1) - 0.45) \\
&\leftarrow 0.6
\end{aligned}
$$

The resulting $V^\pi(s)$ is shown in Figure 3(f).

From this example, we can see how slowly TD(0) progresses (although it will eventually converge!). The more general purpose TD(λ) algorithm tries to be smarter about how it uses the information from each trial.

3 Active Learning

3.1 Q-Learning

Q-learning is an alternate TD method that learns values on state-action pairs, $Q(s,a)$, instead of utilities on states. The TD update equation for Q-learning is:

$$Q(s,a) \leftarrow Q(s,a) + \alpha(N(s,a))(R(s) + \gamma \max_{a'} Q(s',a') - Q(s,a)) \tag{4}$$

where $N(s,a)$ denotes the number of times we have been in state s and taken action a, and $\alpha(n)$ is a function that increases and n decreases. Again we use $\alpha(n) = \frac{1}{n}$.

Initialize all $Q(s,a)$ to 0. (This can also be done randomly.) Then, for each trial, we perform the update equation for each $\langle s, a, s' \rangle$ tuple. We also keep a table of $N(s,a)$ counts.

We will use the same set of trials as for the previous two examples, for simplicity. However, in general, since Q-learning is an *active* learning algorithm, each trial would have been produced using an exploration function that trades off between exploring the state-action space, and exploiting the current learned model.

Also, in the version of Q-learning presented in Russell and Norvig (page 776), a terminal state cannot have a reward. However, this is just an artifact of their particular formulation, and not inherent to Q-learning. Thus, for consistency with the previous examples, we will simply set the value of $Q(s,a)$ for a terminal state s to its reward $R(s)$, for all values of a.

Trial 1

As we said above, we will learned in this trial that state $(3,3)$ is a terminal state with reward 1. Therefore, we will set the value of the Q-function for $(3,3$ to 1, for all a. And since all other utilities are zero, this is the only non-trivial update while performing updates for trial 1.

$$\begin{aligned} Q((3,3),N) &\leftarrow 1 \\ Q((3,3),E) &\leftarrow 1 \\ Q((3,3),S) &\leftarrow 1 \\ Q((3,3),W) &\leftarrow 1 \end{aligned}$$

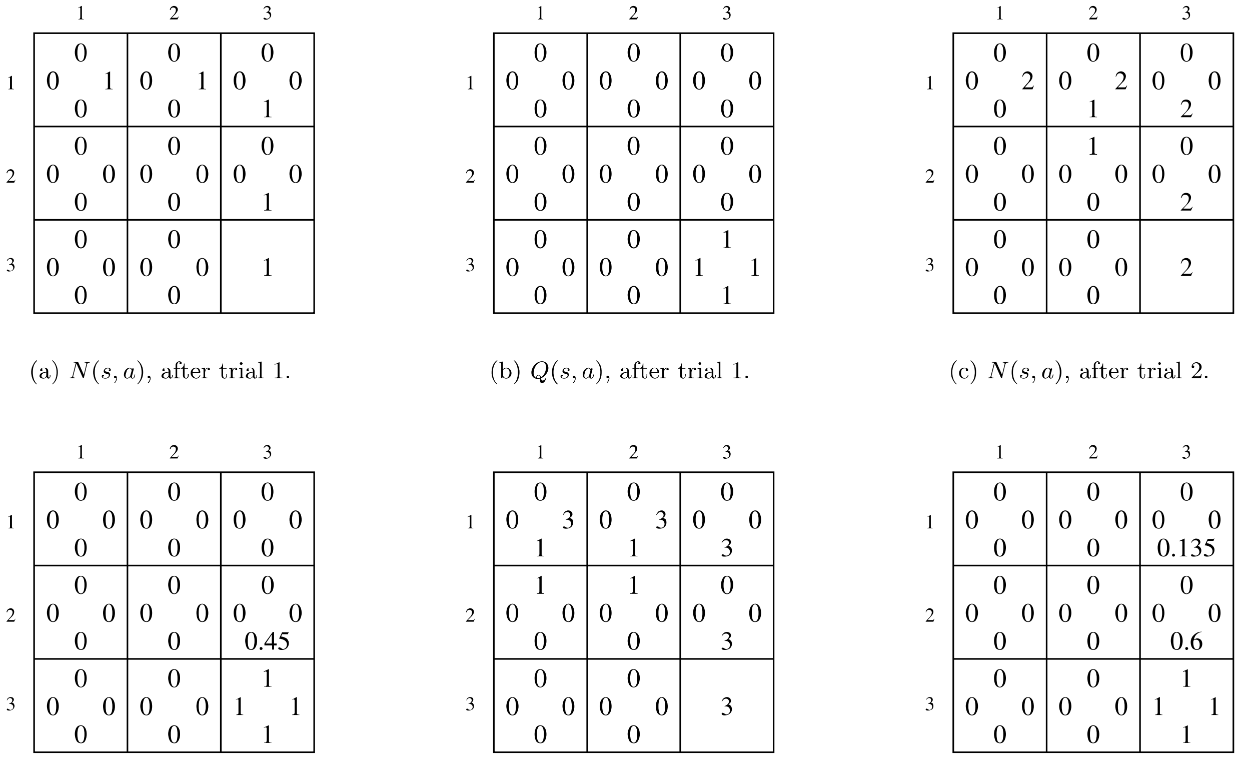
The resulting $Q(s,a)$ is shown in Figure 4(b).

Trial 2

The only non-zero update in this trial is for state-action pair $\langle (2,3), S \rangle$:

$$\begin{aligned} Q((2,3),S) &\leftarrow 0 + \frac{1}{2}(0 + 0.9(1) - 0) \\ &\leftarrow 0.45 \end{aligned}$$

The resulting $Q(s,a)$ is shown in Figure 4(d).



(a) $N(s,a)$, after trial 1.

(b) $Q(s,a)$, after trial 1.

(c) $N(s,a)$, after trial 2.

(d) $Q(s,a)$, after trial 2.

(e) $N(s,a)$, after trial 3.

(f) $Q(s,a)$, after trial 3.

Figure 4: Learned utilities, using Q-learning.

Trial 3

As in passive TD learning, now we have two non-zero updates during this trial; first state $(1,3)$, and then another update to state $(2,3)$:

$$\begin{aligned}
Q((1,3),S) &\leftarrow 0 + \frac{1}{3}(0 + 0.9(0.45) - 0) \\
&\leftarrow 0.135 \\
Q((2,3),S) &\leftarrow 0.45 + \frac{1}{3}(0 + 0.9(1) - 0.45) \\
&\leftarrow 0.6
\end{aligned}$$

The resulting $Q(s,a)$ is shown in Figure 4(f).