

This gridworld MDP operates like to the one we saw in class. The states are grid squares, identified by their row and column number (row first). The agent always starts in state (1,1), marked with the letter S. There are two terminal goal states, (2,3) with reward +5 and (1,3) with reward -5. Rewards are 0 in non-terminal states. (The reward for a state is received as the agent moves into the state.) The transition function is such that the intended agent movement (North, South, West, or East) happens with probability .8. With probability .1 each, the agent ends up in one of the states perpendicular to the intended direction. If a collision with a wall happens, the agent stays in the same state.

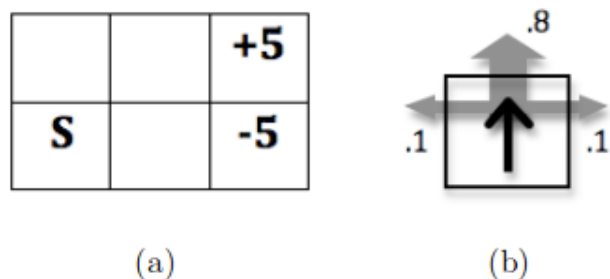


Figure 1: (a) Gridworld MDP. (b) Transition function.

1. Draw the optimal policy for this grid? (5 points)

S =	(1,1)	(1,2)	(1,3)	(2,1)	(2,2)	(2,3)
$\pi^*(S)$ =	Up	Left	NA	Right	Right	NA

2. Suppose the agent knows the transition probabilities. Give the first two rounds of value iteration updates for each state, with a discount of 0.9. (Assume V_0 is 0 everywhere and compute V_i for times $i = 1, 2$). (8 points)

Apply the Bellman backups $V_{i+1}(s) = \max_a (\sum_{s'} T(s, a, s') (R(s, a, s') + \gamma V_i(s')))$ twice. I will show the computations for the max actions. Most of the terms will be zero, which are omitted here for compactness.

S =	(1,1)	(1,2)	(1,3)	(2,1)	(2,2)	(2,3)
$V_0(S)$ =	0	0	0	0	0	0
$V_1(S)$ =	0	0	0	0	$0.8 \times 5.0 = 4.0$	0
$V_2(S)$ =	0	$0.9 \times 0.8 \times 4$ $+ 0.1 \times -5 = 2.38$	0	$0.8 \times 0.9 \times 4.0 = 2.88$	$0.8 \times 5.0 = 4.0$	0

3. Suppose the agent does not know the transition probabilities. What does it need to be able to do (or have available) in order to learn the optimal policy? (5 points)

The agent must be able to explore the world by taking actions and observing the effects.

4. The agent starts with the policy that always chooses to go right, and executes the following three trials: 1) (1,1)–(1,2)–(1,3), 2) (1,1)–(1,2)–(2,2)–(2,3), and 3) (1,1)–(2,1)–(2,2)–(2,3). What are the monte carlo (direct utility) estimates for states (1,1) and (2,2), given these traces? (5 points)

To compute the estimates, average the rewards received in the trajectories that went through the indicated states.

$$V((1,1)) = (-5 + 5 + 5)/3 = 5/3 = 1.666$$

$$V((2,2)) = (5 + 5)/2 = 5$$

5. Using a learning rate of .1 and assuming initial values of 0, what updates does the TD-learning agent make after trials 1 and 2, above? (5 points)

The general TD-learning update is (the other form from lecture is also acceptable):

$$V(s) = V(s) + \alpha(r + \gamma V(s') - V(s))$$

After trial 1, all of the updates will be zero, except for:

$$V((1,2)) = 0 + .1(-5 + 0.9 \times 0 - 0) = -0.5$$

After trial 2, the updates will be:

$$V((1,1)) = 0 + .1(0 + 0.9 \times -0.5 - 0) = -0.045$$

$$V((1,2)) = -0.5 + .1(0 + 0.9 \times 0 + 0.5) = -0.45$$

$$V((2,2)) = 0 + .1(5 + 0.9 \times 0 - 0) = 0.5$$

Consider an MDP with states $\{4, 3, 2, 1, 0\}$, where 4 is the starting state. In states $k \geq 1$, you can *walk* (W) and $T(k, W, k-1) = 1$. In states $k \geq 2$, you can also *jump* (J) and $T(k, J, k-2) = T(k, J, k) = 1/2$. State 0 is a terminal state. The reward $R(s, a, s') = (s - s')^2$ for all (s, a, s') . Use a discount of $\gamma = 1/2$.

(a) (3 pt) Compute $V^*(2)$.

Solution:

We compute the value function from state 0 to state 2.

$$\begin{aligned}
 V^*(0) &= 0 \\
 V^*(1) &= \max\{1 + \gamma V^*(0), \frac{1}{2}(1 + \gamma V^*(0)) + \frac{1}{2}\gamma V^*(1)\} \\
 &= \max\{1, \frac{1}{2} + \frac{1}{4}V^*(1)\} \\
 &= \max\{1, \frac{1}{2}\frac{4}{3}\} \\
 &= 1 \\
 V^*(2) &= \max\{1 + \gamma V^*(1), \frac{1}{2}(4 + \gamma V^*(0)) + \frac{1}{2}\gamma V^*(2)\} \\
 &= \max\{\frac{3}{2}, 2 + \frac{1}{4}V^*(2)\} \\
 &= \max\{\frac{3}{2}, 2\frac{4}{3}\} = \frac{8}{3}
 \end{aligned}$$

where $\frac{1}{2}\frac{4}{3}$ comes from supposing that if $\frac{1}{2} + \frac{1}{4}V^*(1)$ were the maximum, then

$$\begin{aligned}
 V^*(1) &= \frac{1}{2} + \frac{1}{4}V^*(1) \\
 \frac{3}{4}V^*(1) &= \frac{1}{2} \\
 V^*(1) &= \frac{1}{2}\frac{4}{3}
 \end{aligned}$$

and similarly for $2\frac{4}{3}$.

(b) (3 pt) Compute $Q^*(4, W)$?

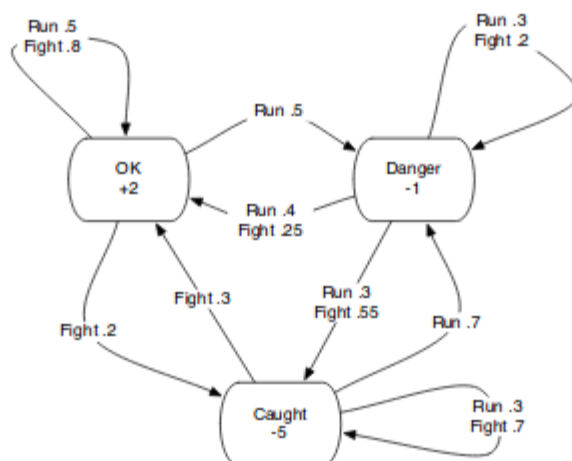
Solution:

$$\begin{aligned}
 V^*(3) &= \frac{1}{2}(4 + \frac{1}{2}V^*(1)) + \frac{1}{2}\frac{1}{2}V^*(3) \\
 &= \frac{9}{4} + \frac{1}{4}V^*(3) \\
 &= 3 \\
 Q^*(4, W) &= 1 + \frac{1}{2}V^*(3) = \frac{5}{2}
 \end{aligned}$$

A boy is being chased around the school yard by bullies and must choose whether to Fight or Run.

- There are three states:
 - Ok (O), where he is fine for the moment.
 - Danger (D), where the bullies are right on his heels.
 - Caught (C), where the bullies catch up with him and administer noogies.
- He begins in state O 75% of the time.
- He begins in state D 25% of the time.

The graph of the MDP is given here:



(a) (9 points) Fill out the table with the results of value iteration with a discount factor $\gamma = .9$:

k	$J^k(O)$	$J^k(D)$	$J^k(C)$
1	2	-1	-5
2	2.54	-1.9	-6.98

$$\begin{aligned}
 J^2(O) &= \max(2 + .9((.5 * 2) + (.5 * -1)), 2 + .9((.8 * 2) + (.2 * -5))) \\
 &= \max(2.45, 2.54) \\
 &= 2.54
 \end{aligned}$$

$$\begin{aligned}
 J^2(D) &= \max(-1 + .9((.4 * 2) + (.3 * -1) + (.3 * -5)), -1 + .9((.25 * 2) + (.2 * -1) + (.55 * -1))) \\
 &= \max(-1.9, -3.2050) \\
 &= -1.9
 \end{aligned}$$

$$\begin{aligned}
 J^2(C) &= \max(-5 + .9((.7 * -1) + (.3 * -5)), -5 + .9((.3 * 2) + (.7 * -5))) \\
 &= \max(-6.98, -7.61) \\
 &= -6.98
 \end{aligned}$$

(b) (5 points) At $k = 2$ with $\gamma = .9$ what policy would you select? Is it necessarily true that this is the optimal policy? At $k = 3$ what policy would you select? Is it necessarily true that this is the optimal policy?

- From O choose Fight.
- From D choose Run.
- From C choose Run.

Value iteration that has not converged is not guaranteed to find the optimal policy, so this policy is not necessarily optimal.

2. (a) (3 points) Suppose you have a robot trying to reach a goal and avoid cliffs in a small grid world. It can only move North, South, East, or West, but occasionally fails to move in the intended direction. If you were to model this using an MDP and were trying to solve it optimally, should you use value iteration or policy iteration? Justify your answer in one sentence.

Generally, use value iteration. We have many states and a few actions, and value iteration is generally cheaper than policy iteration.

- (b) (3 points) Now suppose that the robot can teleport to any grid cell but the teleportation causes it to land in neighboring grid cells near the target with some probability. Of you were to model this using an MDP and were trying to solve it optimally should you use value iteration or policy iteration? Justify your answer in one sentence.

Now we should use policy iteration. Instead of 4 actions we now have an action for teleporting to each different square of the grid, which is now a lot of actions; policy iteration is better when we have many many actions.

1. (16 points) Consider a system with two states and two actions. You perform actions and observe the rewards and transitions listed below. Each step lists the current state, reward, action and resulting transition as $S_i; R = r; a_k : S_i \rightarrow S_j$. Perform Q-learning using a learning rate of $\alpha = 0.5$ and a discount factor of $\gamma = 0.5$ for each step. The Q-table entries are initialized to zero.

$S_1 \quad R = -10 \quad a_1 : S_1 \rightarrow S_1$		
Q	S_1	S_2
a_1	-5	0
a_2	0	0

$$Q(a, s) \leftarrow Q(a, s) + \alpha(R(s) + \gamma \max_{a'} [Q(a', s')] - Q(a, s))$$

$$Q(a_1, S_1) \leftarrow 0 + 0.5(-10 + 0.5 \max_{a'(s'=S_1)} [0, 0] - 0)$$

$$= -5$$

$S_1 \quad R = -10 \quad a_2 : S_1 \rightarrow S_2$		
Q	S_1	S_2
a_1	-5	0
a_2	-5	0

$$Q(a_2, S_1) \leftarrow 0 + 0.5(-10 + 0.5 \max_{a'(s'=S_2)} [0, 0] - 0)$$

$$= -5$$

$S_2 \quad R = +20 \quad a_1 : S_2 \rightarrow S_1$		
Q	S_1	S_2
a_1	-5	8.75
a_2	-5	0

$$Q(a_1, S_2) \leftarrow 0 + 0.5(+20 + 0.5 \max_{a'(s'=S_1)} [-5, -5] - 0)$$

$$= 8.75$$

$S_1 \quad R = -10 \quad a_2 : S_1 \rightarrow S_2$		
Q	S_1	S_2
a_1	-5	8.75
a_2	-5.3124	0

$$Q(a_2, S_1) \leftarrow -5 + 0.5(-10 + 0.5 \max_{a'(s'=S_2)} [8.75, 0] - (-5))$$

$$= -5.3125$$

2. (4 points) What is the optimal policy at this point?

$$\pi(S_1) = a_1$$

$$\pi(S_2) = a_1$$

Assume we are an agent in a 3x2 gridworld, as shown in the below figure. We start at the bottom left node (1) and finish in the top right node (6). When node 6 is reached, we receive a reward of +10 and return to the start for a new episode. On all other actions that not lead to state 6, the reward is -1.

4	5	finish 6
start 1	2	3

In each state we have four possible actions: up, down, left and right. For each action we move deterministically in the specific direction on the grid. Assume that we cannot take actions that bring us outside the grid.

The current estimates of $Q(s, a)$ are given in the below table:

Q(1,up)=4			Q(1,right)=3
Q(2,up)=6		Q(2,left)=3	Q(2,right)=8
Q(3,up)=9		Q(3,left)=7	
	Q(4,down)=2		Q(4,right)=5
	Q(5,down)=6	Q(5,left)=5	Q(5,right)=8

Question a (1p) Since we have full environmental knowledge, we can apply Bellman's equation to further update the Q estimates (i.e. dynamic programming). We take a **greedy** policy and $\gamma = 0.9$. For convenience, the Q-specification of Bellman's equation is given (where s' and a' denote the next state and action, respectively):

$$Q(s, a) = \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma \sum_{a'} \pi(s', a') Q(s', a')] \quad (1)$$

Perform a single update of $Q(3, \text{left})$.

Answer: $Q(3, \text{left}) = 1 * [-1 + 0.9 * (1 * 8)] = 6.2$

Note that because the environment is deterministic, we only need to sum over one s' (which has $P_{3,2}^{left} = 1$), and because the policy is greedy, we only sum over one next action a' (which has probability $\pi(2, right) = 1$).

Question b (1p) We now no longer assume a model of the environment. The above table was rather created through temporal difference learning, where we sample through the state-space. Why is it not smart to take the greedy policy now (from the start)? What should be balanced here?

Answer:

- (0.5p) In the sampling-based/learning setting, the greedy policy may lead to suboptimal results, since we do not ensure exploration. Maybe there are parts of the state-space with even better rewards, but we just have not visited them yet.
- (0.5p) What we should balance is exploration versus exploitation. The greedy policy only considers exploitation.

Question c (1p) Why were we using Q-values? What is the advantage of learning state-action values (Q) compared to state values (V)? (Hint: consider action selection)

Answer: The advantage is in the action selection. With Q-values, we can directly see the value of each available action, and use these in e.g. ϵ -greedy or softmax action selection. On the contrary, when we only have V-estimates, we need to 1) do extra calculations, and 2) have a transition model ($P_{ss'}^a$), so at each decision moment we can calculate the $Q(s, a) = \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma V(s')]$ for all available actions a . Especially when controlling a real-time system, your decisions have to be fast.

Question d (1p) We decide to switch to softmax exploration:

$$\pi(s, a) = \frac{e^{Q(s, a)}}{\sum_b e^{Q(s, b)}} \quad (2)$$

We are currently in node 2. Give the probability that we will move right on the next step (you can write the equation with correct numeric values, but you may skip calculating the resulting decimal number).

Answer: $\pi(2, right) = \frac{e^8}{e^8 + e^6 + e^3}$

Dont forget to sum over all available actions in the denominator, including the one you are calculating the probability for. I see some of you omitting the e^8 term in the denominator in this example. However, then the probability distribution over all actions will not sum up to 1.