1 Reinforcement Learning

1.1 Markov Decision Process

Environment Setup (may contain spoilers for Shrek 1)

Lord Farquaad is hoping to evict all fairytale creatures from his kingdom of Duloc, and has one final ogre to evict: Shrek. Unfortunately all his previous attempts to catch the crafty ogre have fallen short, and he turns to you, with your knowledge of Markov Decision Processes (MDP's) to help him catch Shrek once and for all.

Consider the following MDP environment where the agent is Lord Farquaad:

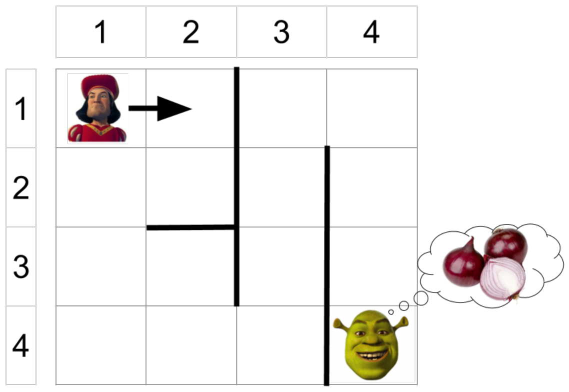


Figure 1: Kingdom of Duloc, circa 2001

Here's how we will define this MDP:

- S **(state space):** a set of states the agent can be in. In this case, the agent (Farquaad) can be in any location (row, col) and also in any orientation $\in \{N, E, S, W\}$. Therefore, state is represented by a three-tuple (row, col, dir), and S = all possible of such tuples. Farquaad's start state is $(1, 1, E)$.
- A **(action space):** a set of actions that the agent can take. Here, we will have just three actions: turn right, turn left, and move forward (turning does not change *row* or *col*, just *dir*). So our action space is $\{R, L, M\}$. Note that Farquaad is debilitatingly short, so he cannot travel through (or over) the walls. Moving forward when facing a wall results in no change in state (but counts as an action).
- $R(s, a)$ **(reward function):** In this scenario, Farquaad gets a reward of 5 by moving into the swamp (the cell containing Shrek), and a reward of 0 otherwise.
- $p(s'|s, a)$ **(transition probabilities):** We'll use a deterministic environment, so this will bee 1 if s' is reachable from s and by taking a, and 0 if not.

1. What are $|S|$ and $|A|$ (size of state space and size of action space)?

   $|S| = 4$ rows $\times$ 4 columns $\times$ 4 orientations $= 64$
   $|A| = |\{R, L, M\}| = 3$

2. Why is it called a "Markov" decision process? (Hint: what is the assumption made with p?)

   $p(s'|s, a)$ assumes that s' is determined only by s and a (and not any other previous states or actions).

3. What are the following transition probabilities?

$$p((1, 1, N)|(1, 1, N), M) =$$
$$p((1, 1, N)|(1, 1, E), L) =$$
$$p((2, 1, S)|(1, 1, S), M) =$$
$$p((2, 1, E)|(1, 1, S), M) =$$

$$p((1, 1, N)|(1, 1, N), M) = 1$$
$$p((1, 1, N)|(1, 1, E), L) = 1$$
$$p((2, 1, S)|(1, 1, S), M) = 1$$
$$p((2, 1, E)|(1, 1, S), M) = 0$$

4. Given a start position of $(1, 1, E)$ and a discount factor of $\gamma = 0.5$, what is the expected discounted future reward from $a = R$? For $a = L$? (Fix $\gamma = 0.5$ for following problems).

   For $a = R$ we get $R_R = 5 * (\frac{1}{2})^{16}$ (it takes 17 moves for Farquaad to get to Shrek, starting with $R, M, M, M, L...$)

   For $a = L$, this is a bad move, and we need another move to get back to our original orientation, from which we can go with our optimal policy. So the reward here is:

   $R_L = (\frac{1}{2})^2 * R_R = 5 * (\frac{1}{2})^{18}$

5. What is the optimal action from each state, given that orientation is fixed at E? (if there are multiple options, choose any)

R	R	M	R
R	R	L	R
M	R	L	R
M	M	L	-

(some have multiple options, I just chose one of the possible ones)

6. Farquaad's chief strategist (Vector from Despicable Me) suggests that having $\gamma = 0.9$ will result in a different set of optimal policies. Is he right? Why or why not?

   Vector is wrong. While the reward quantity will be different, the set of optimal policies does not change. (it is now $5 * (\frac{9}{10})^{16}$) (one can only assume that Lord Farquaad and Vector would be in kahoots: both are extremely nefarious!)

7. Vector then suggests the following setup: $R(s, a) = 0$ when moving into the swamp, and $R(s, a) = -1$ otherwise. Will this result in a different set of optimal policies? Why or why not?

   It will not. While the reward quantity will be different, the set of optimal policies does not change. (Farquaad will still try to minimize the number of steps he takes in order to reach Shrek)

8. Vector now suggests the following setup: $R(s, a) = 5$ when moving into the swamp, and $R(s, a) = 0$ otherwise, but with $\gamma = 1$. Could this result in a different optimal policy? Why or why not?

   This will change the policy, but not in Lord Farquaad's favor. He will no longer be incentivized to reach Shrek quickly (since $\gamma = 1$). The optimal reward from each state is the same (5) and therefore each action from each state is also optimal. Vector really should have taken 10-301/601...

9. Surprise! Elsa from Frozen suddenly shows up. Vector hypnotizes her and forces her to use her powers to turn the ground into ice. Now the environment is now stochastic: since the ground is now slippery, when choosing the action M, with a 0.2 chance, Farquaad will slip and move two squares instead of one. What is the expected future-discounted rewards from $s = (2, 4, S)$?

   Recall that $R_{exp} = max_a E[R(s, a) + \gamma R_{s'}]$

   (notation might be different than in the notes, but conceptually, our reward is the best expected reward we can get from taking any action a from our current state s.)

In this case, our best action is obviously to move forward. So we get

R_{exp} = (expected value of going two steps) + (expected value of going one step)

$E[2_{steps}] = p((4,4,S)|(2,4,S),M) \times R((4,4,S),(2,4,S),M) = 0.2 \times 5 = 1$

$E[1_{step}] = p((4,3,S)|(2,4,S),M) \times (R((4,3,S),(2,4,S),M) + \gamma R_{(4,3,S)})$

where $R_{(4,3,S)}$ is the expected reward from $(4,3,S)$. Since the best reward from here is obtained by choosing $a = M$, and we always end up at Shrek, we get

$E[1_{step}] = 0.8 \times (0 + \gamma \times 5) = 0.8 \times 0.5 \times 5 = 2$

giving us a total expected reward of $R_{exp} = 1 + 2 = 3$

(I will be very disappointed if this is not the plot of Shrek 5)

1.2 Value and Policy Iteration

1. Which of the following environment characteristics would increase the computational complexity per iteration for a value iteration algorithm? Choose all that apply:

   ☐ Large Action Space

   ☐ A Stochastic Transition Function

   ☐ Large State Space

   ☐ Unknown Reward Function

   ☐ None of the Above

   A and C (state space and action space). The computational complexity for value iteration per iteration is $O(|A||S|^2)$

2. Which of the following environment characteristics would increase the computational complexity per iteration for a policy iteration algorithm? Choose all that apply:

   ☐ Large Action Space

   ☐ A Stochastic Transition Function

   ☐ Large State Space

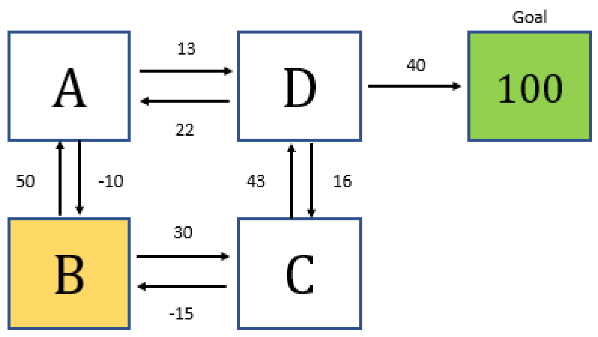
   ☐ Unknown Reward Function

   ☐ None of the Above

   A and C again. The computational complexity for policy iteration per iteration is $O(|A||S|^2 + |S|^3)$

3. In the image below is a representation of the game that you are about to play. There are 5 states: A, B, C, D, and the goal state. The goal state, when reached, gives 100 points as reward. In addition to the goal's points, you also get points by moving to different states. The amount of points you get are shown next to the arrows. You start at state

B. To figure out the best policy, you use asynchronous value iteration with a decay (γ) of 0.9.



(i) When you first start playing the game, what action would you take (up, down, left, right) at state B?

Up

(ii) What is the total reward at state B at this time?

50 (immediate reward of 50, and future reward (value at state A) starts at 0)

(iii) Let's say you keep playing until your total values for each state has converged. What action would you take at state B?

C

(iv) What is the total reward at state B at this time?

174 (30 from the immediate action, and 144 from the future reward (value at state C))

4. Let $V_k(s)$ indicate the value of state s at iteration k in (synchronous) value iteration. What is the relation ship between $V_{k+1}(s)$ and $\sum_{s'} P(s'|s,a)[R(s,a,s')+\gamma V_k(s')]$, for any $a \in$ actions? Please indicate the most restrictive relationship that applies. For example, if $x < y$ always holds, please use $<$ instead of $\leq$. Selecting ? means it's not possible to assign any true relationship. (Select the best choice)

$V_{k+1}(s) \square \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V_k(s')]$

- ◯ $=$
- ◯ $<$
- ◯ $>$
- ◯ $\leq$
- ◯ $\geq$
- ◯ ?

E

1.3 Q-Learning

1. For the following true/false, circle one answer and provide a one-sentence explanation:

   (i) One advantage that Q-learning has over Value and Policy iteration is that it can account for non-deterministic policies.

   Circle one: True False

   False. All three methods can account for non-deterministic policies

   (ii) You can apply Value or Policy iteration to any problem that Q-learning can be applied to.

   Circle one: True False

   False. Unlike the others, Q-learning doesn't need to know the transition probabilities (p(s' | s, a)), or the reward function (r(s,a)) to train. This is its biggest advantage.

   (iii) Q-learning is guaranteed to converge to the true value Q* for a greedy policy.

   Circle one: True False

   False. Q-learning converges only if every state will be explored infinitely. Thus, purely exploiting policies (e.g. greedy policies) will not necessarily converge to Q*, but rather to a local optimum.

2. For the following parts of this problem, recall that the update rule for Q-learning is:

$$\mathbf{w} \leftarrow \mathbf{w} - \alpha \left(q(\mathbf{s}, a; \mathbf{w}) - (r + \gamma \max_{a'} q(\mathbf{s}', a'; \mathbf{w}) \right) \nabla_{\mathbf{w}} q(\mathbf{s}, a; \mathbf{w})$$

   (i) From the update rule, let's look at the specific term $X = (r + \gamma \max_{a'} q(\mathbf{s}', a'; \mathbf{w}))$ Describe in English what is the role of X in the weight update.

   Estimate of true total return (Q*(s,a)). This may get multiple answers, so grade accordingly

(ii) Is this update rule synchronous or asynchronous?

Asynchronous

(iii) A common adaptation to Q-learning is to incorporate rewards from more time steps into the term X. Thus, our normal term $r_t + \gamma * max_{a_{t+1}} q(s_{t+1}, a_{t+1}; w)$ would become $r_t + \gamma * r_{t+1} + \gamma^2 \max_{a_{t+2}} q(\mathbf{s}_{t+2}, a_{t+2} : \mathbf{w})$ What are the advantages of using more rewards in this estimation?

Incorporating rewards from multiple time steps allows for a more "realistic" estimate of the true total reward, since a larger percentage of it is from real experience. It can help with stabilizing the training procedure, while still allowing training at each time step (bootstrapping). This type of method is called N-Step Temporal Difference Learning.

3. Let $Q(s, a)$ indicate the estimated Q-value of state-action pair (s, a) at some point during Q-learning. Now your learner gets reward r after taking action a at state s and arrives at state s'. Before updating the Q values based on this experience, what is the relationship between $Q(s, a)$ and $r + \gamma \max_{a'} Q(s', a')$? Please indicate the most restrictive relationship that applies. For example, if $x < y$ always holds, please use $<$ instead of $\leq$. Selecting ? means it's not possible to assign any true relationship. (Select the best choice)

$Q(s, a) \ \square \ r + \gamma \max_{a'} Q(s', a')$

- ○ $=$
- ○ $<$
- ○ $>$
- ○ $\leq$
- ○ $\geq$
- ○ ?

F

4. During standard (not approximate) Q-learning, you get reward r after taking action $North$ from state A and arriving at state B. You compute the sample $r + \gamma Q(B, South)$, where $South = \arg\max_a Q(B, a)$.

Which of the following Q-values are updated during this step? (Select all that apply)

- ○ Q(A, North)
- ○ Q(A, South)
- ○ Q(B, North)
- ○ Q(B, South)
- ○ None of the above

A

5. In general, for Q-Learning (standard/tabular Q-learning, not approximate Q-learning) to converge to the optimal Q-values, which of the following are true?

   True or False: It is necessary that every state-action pair is visited infinitely often.

   ◯ True

   ◯ False

   True or False: It is necessary that the discount γ is less than 0.5.

   ◯ True

   ◯ False

   True or False: It is necessary that actions get chosen according to $\arg\max_a Q(s,a)$.

   ◯ True

   ◯ False

   (1) **True**: In order to ensure convergence in general for Q learning, this has to be true. In practice, we generally care about the policy, which converges well before the values do, so it is not necessary to run it infinitely often. (2) **False**: The discount factor must be greater than 0 and less than 1, not 0.5. (3) **False**: This would actually do rather poorly, because it is purely exploiting based on the Q-values learned thus far, and not exploring other states to try and find a better policy.

6. Run (synchronous) value iteration on this environment for two iterations. Begin by initializing the value for all states, $V_0(s)$, to zero.

   Write the value of each state after the first ($k = 1$) and the second ($k = 2$) iterations. Write your values as a comma-separated list of 6 numerical expressions in the alphabetical order of the states, specifically $V(A), V(B), V(C), V(D), V(E), V(T)$. Each of the six entries may be a number or an expression that evaluates to a number. Do not include any max operations in your response.

   There is a space below to type any work that you would like us to consider. Showing work is optional. Correct answers will be given full credit, even if no work is shown.

   $V_1(A), V_1(B), V_1(C), V_1(D), V_1(E), V_1(T)$ (Values for 6 states):

   $10, -1, -1, -1, 1, 0$

   $V_2(A), V_2(B), V_2(C), V_2(D), V_2(E), V_2(T)$ (values for 6 states):

$10, 6.8, -2, -0.4, 1, 0$

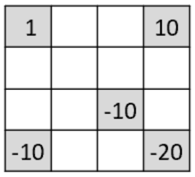
What is the resulting policy after this second iteration? Write your answer as a comma-separated list of three actions representing the policy for states, B, C, and D, in that order. Actions may be Left or Right.

$\pi(B), \pi(C), \pi(D)$ based on V_2 :

Left, Left, Right

Optional work for this problem:

7. Consider a 4x4 Grid World that follows the same rules as Grid World from lecture. Specifically:



- The shaded states have only one action, exit, which leads to a terminal state (not shown) and a reward with the corresponding numerical value printed in that state.
- Leaving any other state gives a living reward, $R(s) = r$.
- The agent will travel in the direction of its chosen action with probability $1 - n$ and will travel in one of the two adjacent directions with probability $n/2$ each.
- If the agent travels into a wall, it will remain in the same state.

Match the MDP setting below with the following optimal policies.

Note: We do not expect you to run value iteration to convergence to compute these policies but rather reason about the effect of different MDP settings.

8. $\gamma = 1.0, n = 0.2, r = 0.1$

A)

1	→	←	10
↓	↑	↑	←
↑	←	-10	↑
-10	↑	←	-20

B)

1	←	→	10
↑	↑	↑	↑
↑	↑	-10	↑
-10	↑	←	-20

C)

1	→	→	10
→	↑	↑	↑
↑	↑	-10	↑
-10	↑	←	-20

D)

1	↑	↑	10
←	↑	↑	←
↑	↑	-10	↑
-10	↑	←	-20

E)

1	←	→	10
↑	↑	↑	↑
↑	←	-10	↑
-10	→	←	-20

F)

1	→	↑	10
↓	↑	↑	↑
↑	←	-10	↑
-10	→	←	-20

○ A

○ B

○ C

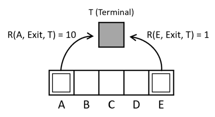
○ D

○ E

○ F

(A). With $\gamma = 1$, the policy will travel to the 10.0 state, and with zero noise, nn, it doesn't have to worry about slipping sideways into negative states.

9. Consider training a robot to navigate the following grid-based MDP environment.



- There are six states, A, B, C, D, E, and a terminal state T.
- Actions from states B, C, and D are Left and Right.
- The only action from states A and E is Exit, which leads deterministically to the terminal state

The reward function is as follows:

- $R(A, Exit, T) = 10$
- $R(E, Exit, T) = 1$
- The reward for any other tuple (s, a, s') equals -1

Assume the discount factor is just 1.

When taking action Left, with 0.8 probability, the robot will successfully move one space to the left, and with 0.2 probability, the robot will move one space in the opposite direction.

When taking action Left, with 0.8 probability, the robot will successfully move one space to the left, and with 0.2 probability, the robot will move one space in the opposite direction.

10. $\gamma = 1.0, n = 0, r = -0.1$

○ A

○ B

○ C

○ D

○ E

○ F

(C). With $\gamma = 1$, the policy will travel to the 10.0 state, and with zero noise, nn, it doesn't have to worry about slipping sideways into negative states.

11. Figure repeated for convenience: $\gamma = 0.1, n = 0.2, r = 0.1$

A)

1	→	←	10
↓	↑	↑	←
↑	←	-10	↑
-10	↑	←	-20

B)

1	←	→	10
↑	↑	↑	↑
↑	↑	-10	↑
-10	↑	←	-20

C)

1	→	→	10
→	↑	↑	↑
↑	↑	-10	↑
-10	↑	←	-20

D)

1	↑	↑	10
←	↑	↑	←
↑	↑	-10	↑
-10	↑	←	-20

E)

1	←	→	10
↑	↑	↑	↑
↑	←	-10	↑
-10	→	←	-20

F)

1	→	↑	10
↓	↑	↑	↑
↑	←	-10	↑
-10	→	←	-20

○ A

○ B

○ C

○ D

○ E

○ F

(E). With low γ, the policy will prefer the closer 1.0 state, but with non-zero noise, it will avoid negative states if at all possible.