Recitation 7 February 27

1 MDPs: Warm-Up

1. What is the Markov Property?

   The **Markov Property** states that action outcomes depend on the current state only (action outcomes do not depend on the past states and actions).

2. What are the Bellman equations, and when are they used?

   The **Bellman Equations** give a definition of "optimal utility" via expectimax recurrence.
   They give a one-step lookahead relationship between utilities at a current time step and the next time step.

3. What is a policy? What is an optimal policy?

   A **policy** is a function that maps states to actions; $\pi(s)$ gives an action for state s.
   An **optimal policy** is a policy that maximizes the expected utility if an agent follows it.

4. How does the discount factor γ affect how the agent finds the optimal policy? Why do we restrict gamma $\gamma < 1$?

   γ determines how much the value of a state should take into account the value of future states that the agent could wind up in.
   A gamma $0 < \gamma < 1$ helps our algorithms converge and to prevents against infinite rewards if our game lasts forever.

5. Fill in the following table explaining the effects of having different gamma values:

γ	**Effect on policy search:**
$\gamma = 0$	
$\gamma = 1$	

γ	**Effect on policy search:**
$\gamma = 0$	A smaller γ indicates smaller "horizon," or a shorter term focus. When γ is 0, we only consider immediate rewards. We do not consider rewards in the future as having any value.
$\gamma = 1$	The higher the discount factor, the more the state would value distant future states. When γ is 1, we begin to act as though rewards at any given point in time are equally valuable.

6. What are the two steps to Policy Iteration?

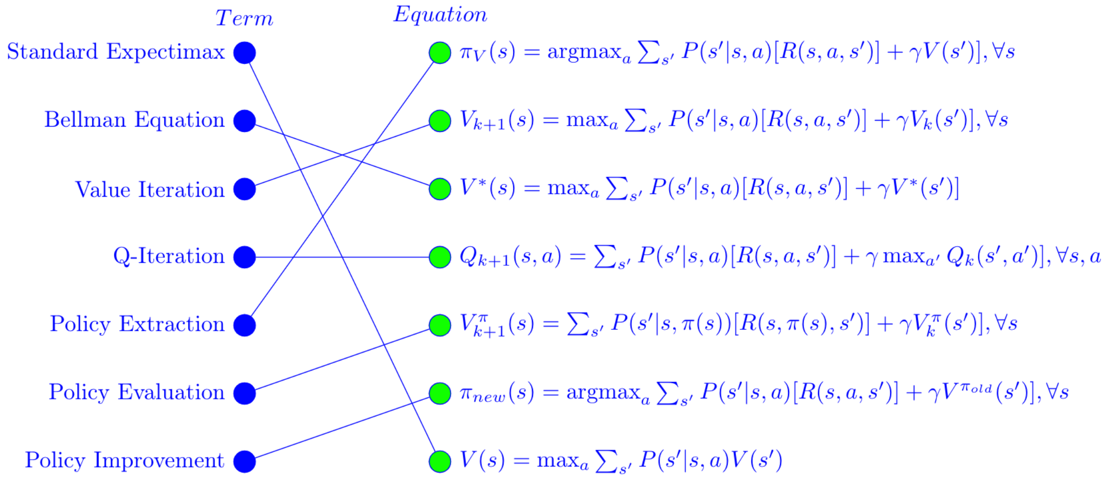
   Policy evaluation and policy improvement.

7. What is the relationship between $V^*(s)$ and $Q^*(s,a)$?

$$V^*(s) = \max_a Q^*(s,a)$$

8. (MDP Notation Review) Draw a line connecting each term in the left column with its corresponding equation in the right column.

Term	*Equation*
Standard Expectimax ●	● $\pi_V(s) = \text{argmax}_a \sum_{s'} P(s'\|s,a)[R(s,a,s')] + \gamma V(s')], \forall s$
Bellman Equation ●	● $V_{k+1}(s) = \max_a \sum_{s'} P(s'\|s,a)[R(s,a,s')] + \gamma V_k(s')], \forall s$
Value Iteration ●	● $V^*(s) = \max_a \sum_{s'} P(s'\|s,a)[R(s,a,s')] + \gamma V^*(s')]$
Q-Iteration ●	● $Q_{k+1}(s,a) = \sum_{s'} P(s'\|s,a)[R(s,a,s')] + \gamma \max_{a'} Q_k(s',a')], \forall s,a$
Policy Extraction ●	● $V^{\pi}_{k+1}(s) = \sum_{s'} P(s'\|s,\pi(s))[R(s,\pi(s),s')] + \gamma V^{\pi}_k(s')], \forall s$
Policy Evaluation ●	● $\pi_{new}(s) = \text{argmax}_a \sum_{s'} P(s'\|s,a)[R(s,a,s')] + \gamma V^{\pi_{old}}(s')], \forall s$
Policy Improvement ●	● $V(s) = \max_a \sum_{s'} P(s'\|s,a)V(s')$

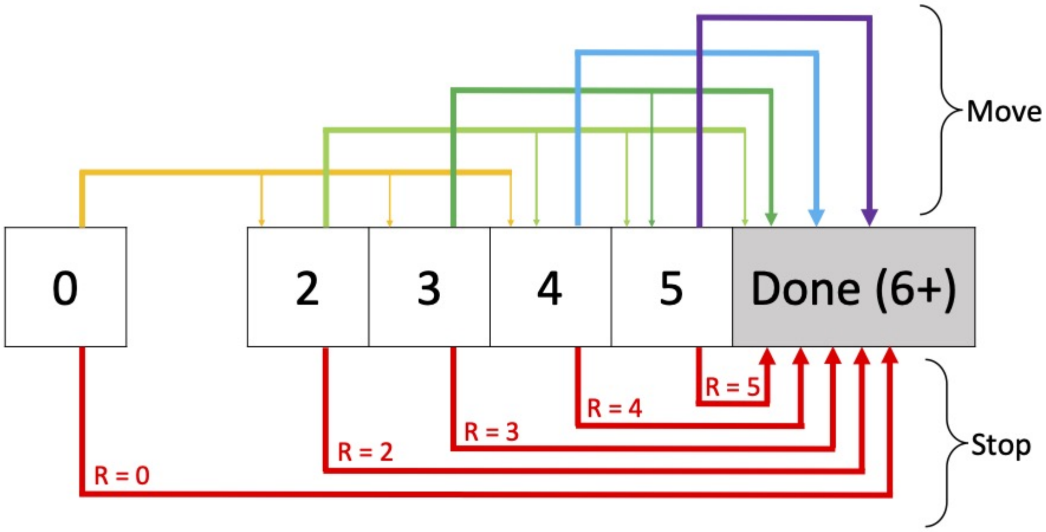

Term
Equation
Standard Expectimax
Bellman Equation
Value Iteration
Q-Iteration
Policy Extraction
Policy Evaluation
Policy Improvement
$\pi_V(s) = \text{argmax}_a \sum_{s'} P(s'|s,a)[R(s,a,s')] + \gamma V(s')], \forall s$
$V_{k+1}(s) = \max_a \sum_{s'} P(s'|s,a)[R(s,a,s')] + \gamma V_k(s')], \forall s$
$V^*(s) = \max_a \sum_{s'} P(s'|s,a)[R(s,a,s')] + \gamma V^*(s')]$
$Q_{k+1}(s,a) = \sum_{s'} P(s'|s,a)[R(s,a,s')] + \gamma \max_{a'} Q_k(s',a')], \forall s,a$
$V^{\pi}_{k+1}(s) = \sum_{s'} P(s'|s,\pi(s))[R(s,\pi(s),s')] + \gamma V^{\pi}_k(s')], \forall s$
$\pi_{new}(s) = \text{argmax}_a \sum_{s'} P(s'|s,a)[R(s,a,s')] + \gamma V^{\pi_{old}}(s')], \forall s$
$V(s) = \max_a \sum_{s'} P(s'|s,a)V(s')$

2 MDPs: Racing

Consider a modification of the racing robot car example seen in lecture. In this game, the car repeatedly moves a random number of spaces that is equally likely to be 2, 3, or 4. The car can either Move or Stop if the total number of spaces moved is less than 6.

If the total spaces moved is 6 or higher, the game automatically ends, and the car receives a reward of 0. When the car Stops, the reward is equal to the total spaces moved (up to 5), and the game ends. There is no reward for the Move action.

Let's formulate this problem as an MDP with the states $\{0, 2, 3, 4, 5, Done\}$.



1. What is the transition function for this MDP? (You should specify discrete values for specific state/action inputs.)

   $T(s, Stop, Done) = 1$, for $s \neq Done$
   $T(0, Move, s') = \frac{1}{3}$ for $s' \in \{2, 3, 4\}$
   $T(2, Move, s') = \frac{1}{3}$ for $s' \in \{4, 5, Done\}$
   $T(3, Move, 5) = \frac{1}{3}$
   $T(3, Move, Done) = \frac{2}{3}$
   $T(4, Move, Done) = 1$
   $T(5, Move, Done) = 1$
   $T(s, a, s') = 0$ otherwise.

2. What is the reward function for this MDP?

   $R(s, Stop, Done) = s, s \leq 5$
   $R(s, a, s') = 0$ otherwise

3. Recall the value iteration update equation:

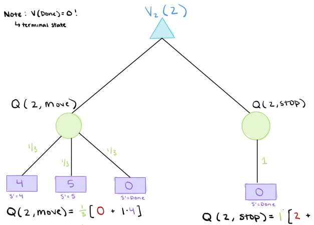
$$V_{k+1}(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V_k(s')]$$

Perform value iteration for 4 iterations with $\gamma = 1$.

States	**0**	**2**	**3**	**4**	**5**	**Done**
V_0						0
V_1						0
V_2						0
V_3						0
V_4						0

States	**0**	**2**	**3**	**4**	**5**	**Done**
V_0	0	0	0	0	0	0
V_1	0	2	3	4	5	0
V_2	3	3	3	4	5	0
V_3	$\frac{10}{3}$	3	3	4	5	0
V_4	$\frac{10}{3}$	3	3	4	5	0

Value iteration tree for $V_2(2)$:



$Q(2, \text{move}) = \frac{1}{3}[0 + 1 \cdot 4] + \frac{1}{3}[0 + 1 \cdot 5] + \frac{1}{3}[0 + 1 \cdot 0] = \frac{4}{3} + \frac{5}{3} = 3$

$Q(2, \text{stop}) = 1[2 + 1 \cdot 0] = 2$

$V_2(2) = \max_{a \in \{\text{move}, \text{stop}\}} \underbrace{Q(2,a)}$

$Q(s,a) = \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V_1(s')]$

$= \max(Q(2,\text{move}), Q(2,\text{stop}))$

$= \max(3, 2)$

$= 3$

4. You should have noticed that value iteration converged above. What is the optimal policy?

States	**0**	**2**	**3**	**4**	**5**
π^*					

States	**0**	**2**	**3**	**4**	**5**
π^*	Move	Move	Stop	Stop	Stop

5. How would our results change with $\gamma = 0.1$?

   By decreasing γ, we focus more and more on immediate rewards. This effectively makes our algorithm more greedy, valuing short-term rewards more than long-term ones.
   For this game, with a discount factor of 0.1, value iteration converges in fewer iterations, but upon a different policy (Move, Stop, Stop, Stop, Stop). For state 2, the algorithm preferred the short-term reward of Stopping over the long-term reward of Moving.
   In the most extreme case with $\gamma = 0$, the values converge immediately and yield an optimal policy of Stop for all states other than state 0, and either Move or Stop at state 0 (because all actions at state 0 lead to an expected utility of 0).

6. Now imagine we changed the rules so that moving to spaces after 5 does not immediately end the game, but rather wraps back around to the beginning states (this is the case where the states exist side by side in a loop).

   For example if you start in state 5, select the action Move, and move 2 spaces, you would end up in state 1 (this is a new state we would need introduce).

   To be clear, the states would now be states $\{0,1,2,3,4,5, \textit{Done}\}$.

   How would this modification change our results? (Assume we again use γ=1)

   This modification essentially adds cycles to our state-space graph. With a discount factor γ=1, cycles in the state-space graph will allow for values to feedback into other states. This feedback would occur for every cycle introduced and each would prolong convergence of value iteration.

   The new feedback for each state would also change the converged value of each state and the optimal policy.

3 MDPs: Policy Iteration

Recall the racing MDP from the prior problem. In this game, the car repeatedly moves a random number of spaces that is equally likely to be 2, 3, or 4. The car can either Move or Stop if the total number of spaces moved is less than 6.
If the total spaces moved is 6 or higher, the game automatically ends, and the car receives a reward of 0. When the car Stops, the reward is equal to the total spaces moved (up to 5), and the game ends. There is no reward for the Move action.
States: $\{0, 2, 3, 4, 5, Done\}$

Transition	Reward
$T(s, Stop, Done) = 1, \quad \forall s \neq Done$	$R(s, Stop, Done) = s, \quad \forall s \leq 5$
$T(0, Move, s') = \frac{1}{3}, \quad \forall s' \in \{2, 3, 4\}$	$R(s, a, s') = 0$ otherwise
$T(2, Move, s') = \frac{1}{3}, \quad \forall s' \in \{4, 5, Done\}$	
$T(3, Move, 5) = \frac{1}{3}$	
$T(3, Move, Done) = \frac{2}{3}$	
$T(4, Move, Done) = 1$	
$T(5, Move, Done) = 1$	
$T(s, a, s') = 0$ otherwise	

Now recall the policy evaluation and policy improvement equations, which together make up policy iteration.

$$\text{Policy Evaluation: } V_{k+1}^{\pi}(s) = \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V_k^{\pi}(s')]$$

$$\text{Policy Improvement: } \pi_{new}(s) \leftarrow \text{argmax}_a \sum_{s'} T(s, a, s')[R(s, a, s') + \gamma V^{\pi_{old}}(s')]$$

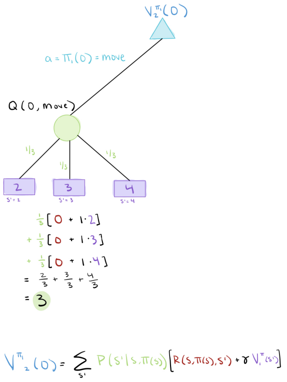
Perform two iterations of policy iteration for one step of this MDP, starting from the fixed policy below. Use the initial $\gamma = 1$.

States	0	2	3	4	5
π_0	Move	Stop	Move	Stop	Move
$V_0^{\pi_0}$					
$V_1^{\pi_0}$					
$V_2^{\pi_0}$					
$V_3^{\pi_0}$					
π_1					
$V_0^{\pi_1}$					
$V_1^{\pi_1}$					
$V_2^{\pi_1}$					
$V_3^{\pi_1}$					
π_2					

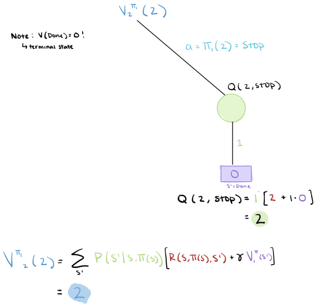
States	0	2	3	4	5
π_0	Move	Stop	Move	Stop	Move
$V_0^{\pi_0}$	0	0	0	0	0
$V_1^{\pi_0}$	0	2	0	4	0
$V_2^{\pi_0}$	2	2	0	4	0
$V_3^{\pi_0}$	2	2	0	4	0
π_1	Move	Stop	Stop	Stop	Stop
$V_0^{\pi_1}$	0	0	0	0	0
$V_1^{\pi_1}$	0	2	3	4	5
$V_2^{\pi_1}$	3	2	3	4	5
$V_3^{\pi_1}$	3	2	3	4	5
π_2	Move	Move	Stop	Stop	Stop

Side Note: Above when evaluating π_1, we started policy evaluation off with all 0s again - this is called a cold start (terminology not super important). We also could have started off instead with the optimal values we'd converged upon last round of policy evaluation ($V_3^{\pi_0}$), which often converges upon the optimal values for π_1 faster.

Policy iteration tree for $V_2^{\pi_1}(0)$:


$V_2^{\pi_1}(0)$
$a = \pi_1(0) = \text{move}$
Q(0, move)
1/3
1/3
1/3
2
S' = 2
3
S' = 3
4
S' = 4
$\frac{1}{3}[0 + 1 \cdot 2]$
$+ \frac{1}{3}[0 + 1 \cdot 3]$
$+ \frac{1}{3}[0 + 1 \cdot 4]$
$= \frac{2}{3} + \frac{3}{3} + \frac{4}{3}$
$= 3$
$V^{\pi_1}{}_2(0) = \sum_{s'} P(s'|s,\pi(s))[R(s,\pi(s),s') + \gamma V_1^{\pi}(s')]$

Policy iteration tree for $V_2^{\pi_1}(2)$:


$V_2^{\pi_1}(2)$
Note: V(Done) = 0!
↳ terminal state
$a = \pi_1(2) = \text{Stop}$
Q(2, STOP)
1
0
S' = Done
$Q(2, \text{STOP}) = 1\,[2 + 1 \cdot 0]$
$= 2$
$V^{\pi_1}{}_2(2) = \sum_{s'} P(s'|s,\pi(s))[R(s,\pi(s),s') + \gamma V_1^{\pi}(s')]$
$= 2$

4 MDPs: Conceptual Questions

1. Which MDP methods are optimal (assuming sufficient convergence when iterating)? [Value Iteration, Policy Iteration, Both, Neither]

   Both!

2. What are some limitations of value iteration? What are some limitations of policy iteration?

   Limitations of value iteration:

   (a) each iteration takes $O(|S|^2|A|)$ time, which can be costly depending on the size of S and A;

   (b) values of many states do not change in one iteration, but the process has to continue as long as there is change in some states;

   (c) sometimes the corresponding policy (extract the policy as if the current V_k is V^*) has already converged to optimal, but the values have not converged and therefore we have to continue the value iteration process, which is a waste of time.

   Limitation of policy iteration:

   (a) Each iteration of policy iteration involves policy evaluation.
   Recall that the equation for policy evaluation is $V^{\pi}_{k+1}(s) = \sum_{s'} P(s'|s,\pi(s))[R(s,\pi(s),s')] + \gamma V^{\pi}_{k}(s')], \forall s$
   Since policy evaluation is an iterative process and is only guaranteed to converge as k $\longrightarrow \infty$, we do not know the time of convergence a priori.

   Depending on the conditions, one algorithm may converge faster than the other.

3. When does policy iteration end? Immediately after policy iteration ends (without performing additional computation), do we have the values of the optimal policy?

   Policy iteration ends when the policy converges, i.e., when $\pi_{new} = \pi_{old}$ after running policy improvement. We do have the values for the optimal policy. Since we necessarily ran policy evaluation on the most recent iteration of policy iteration, we have the value function $V^{\pi_{old}}$ corresponding to π_{old}. We know that $\pi_{new} = \pi_{old}$, implying that $V^{\pi_{old}} = V^{\pi_{new}}$. Therefore, we have $V^{\pi_{new}}$, which is the value function corresponding to the optimal policy π_{new}.

4. What changes if during policy iteration, you only run one iteration of Bellman update instead of running it until convergence? Do you still get an optimal policy?

You will get an optimal policy if the termination condition is updated.
Note: policy iteration can converge too early because the policy might not change between two iterations of changing values. For an example, refer to the value iteration demo in Lecture 15, where we run policy extraction after every iteration of value iteration. There are a couple of early iterations where the non-optimal policy doesn't change.

The key observation here is that policy iteration is effectively the same as value iteration, since value iteration involves one step of evaluation as well. In this case, we update values based on our current best policy, and keep doing that until convergence of a policy!

Recall that the equation for value iteration is:

$$V_{k+1}(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V_k(s')] \tag{1}$$

And the equation for policy evaluation is:

$$V_{k+1}^{\pi_i}(s) = \sum_{s'} T(s,\pi(s),s')[R(s,\pi(s),s') + \gamma V_k^{\pi_i}(s')] \tag{2}$$

given some fixed policy π that we update in the policy improvement step, given by:

$$\pi_{i+1}(s) \leftarrow \arg\max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V_k^{\pi_i}(s')] \tag{3}$$

which looks very similar to policy extraction done in value iteration.