

ECE 5984 – Deep Reinforcement Learning

Homework Assignment #1: Markov Decision Processes (MDPs)

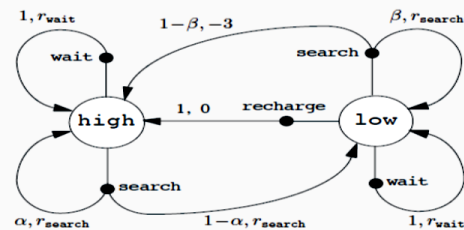
Problem 1: Give a table analogous to that in Example 3.3, but for $p(s', r|s, a)$. It should have columns for s, a, s', r , and $p(s', r|s, a)$, and a row for every 4-tuple for which $p(s', r|s, a) > 0$.

Example 3.3 Recycling Robot

A mobile robot has the job of collecting empty soda cans in an office environment. It has sensors for detecting cans, and an arm and gripper that can pick them up and place them in an onboard bin; it runs on a rechargeable battery. The robot's control system has components for interpreting sensory information, for navigating, and for controlling the arm and gripper. High-level decisions about how to search for cans are made by a reinforcement learning agent based on the current charge level of the battery. To make a simple example, we assume that only two charge levels can be distinguished, comprising a small state set $S = \{\text{high}, \text{low}\}$. In each state, the agent can decide whether to (1) actively **search** for a can for a certain period of time, (2) remain stationary and **wait** for someone to bring it a can, or (3) head back to its home base to **recharge** its battery. When the energy level is **high**, recharging would always be foolish, so we do not include it in the action set for this state. The action sets are then $\mathcal{A}(\text{high}) = \{\text{search}, \text{wait}\}$ and $\mathcal{A}(\text{low}) = \{\text{search}, \text{wait}, \text{recharge}\}$.

The rewards are zero most of the time, but become positive when the robot secures an empty can, or large and negative if the battery runs all the way down. The best way to find cans is to actively search for them, but this runs down the robot's battery, whereas waiting does not. Whenever the robot is searching, the possibility exists that its battery will become depleted. In this case the robot must shut down and wait to be rescued (producing a low reward). If the energy level is **high**, then a period of active search can always be completed without risk of depleting the battery. A period of searching that begins with a **high** energy level leaves the energy level **high** with probability α and reduces it to **low** with probability $1 - \alpha$. On the other hand, a period of searching undertaken when the energy level is **low** leaves it **low** with probability β and depletes the battery with probability $1 - \beta$. In the latter case, the robot must be rescued, and the battery is then recharged back to **high**. Each can collected by the robot counts as a unit reward, whereas a reward of -3 results whenever the robot has to be rescued. Let r_{search} and r_{wait} , with $r_{\text{search}} > r_{\text{wait}}$, respectively denote the expected number of cans the robot will collect (and hence the expected reward) while searching and while waiting. Finally, suppose that no cans can be collected during a run home for recharging, and that no cans can be collected on a step in which the battery is depleted. This system is then a finite MDP, and we can write down the transition probabilities and the expected rewards, with dynamics as indicated in the table on the left:

s	a	s'	$p(s' s, a)$	$r(s, a, s')$
high	search	high	α	r_{search}
high	search	low	$1 - \alpha$	$r_{\text{search}} - 3$
low	search	high	$1 - \beta$	$r_{\text{search}} - 3$
low	search	low	β	r_{search}
high	wait	high	1	r_{wait}
high	wait	low	0	-
low	wait	high	0	-
low	wait	low	1	r_{wait}
low	recharge	high	1	0
low	recharge	low	0	-



Note that there is a row in the table for each possible combination of current state, s , action, $a \in \mathcal{A}(s)$, and next state, s' . Some transitions have zero probability of occurring, so no expected reward is specified for them. Shown on the right is another useful way of

summarizing the dynamics of a finite MDP, as a *transition graph*. There are two kinds of nodes: *state nodes* and *action nodes*. There is a state node for each possible state (a large open circle labeled by the name of the state), and an action node for each state–action pair (a small solid circle labeled by the name of the action and connected by a line to the state node). Starting in state s and taking action a moves you along the line from state node s to action node (s, a) . Then the environment responds with a transition to the next state's node via one of the arrows leaving action node (s, a) . Each arrow corresponds to a triple (s, s', a) , where s' is the next state, and we label the arrow with the transition probability, $p(s'|s, a)$, and the expected reward for that transition, $r(s, a, s')$. Note that the transition probabilities labeling the arrows leaving an action node always sum to 1.

Problem 2: Rewrite the four Bellman equations for the four value functions (v_π, v_*, q_π, q_*) in terms of the three-argument function $p(s'|s, a)$:

$$p(s' | s, a) \doteq \Pr\{S_t = s' \mid S_{t-1} = s, A_{t-1} = a\} = \sum_{r \in \mathcal{R}} p(s', r | s, a)$$

and the two-argument function $r(s, a)$:

$$r(s, a) \doteq \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a] = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r | s, a)$$