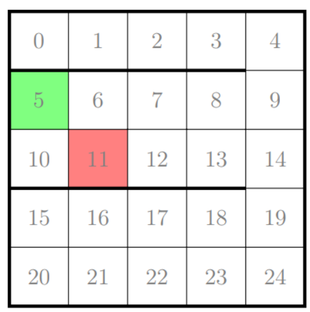
CS 67800, Spring 2017/18

Problem Set 6: Reinforcement Learning

Submission Date : Sunday 8/7/18, 23:59

---

1. **Grid Policies**
   Consider the following grid environment. Starting from any unshaded square, you can move up, down, left, or right. Actions are deterministic and always succeed (e.g. going left from state 1 goes to state 0) unless they will cause the agent to run into a wall. The thicker edges indicate walls, and attempting to move in the direction of a wall results in staying in the same square. Taking any action from the green target square (no. 5) earns a reward of +5 and ends the episode. Taking any action from the red square of death (no. 11) earns a reward of -5 and ends the episode. Otherwise, each move is associated with some reward $r \in \{-1, 0, 1\}$. Assume the discount factor is $\gamma = 1$, unless otherwise specified.



   (a) Define the reward r for actions taken in each unshaded state (using the same reward for all actions in each sate) that would cause the optimal policy to return the *shortest path* to the green target square (no. 5). Try to use the simplest possible reward.

   (b) Using r from part (a), find the optimal value function for each square.

   (c) Does setting $\gamma = 0.8$ change the optimal policy? Why or why not?

   (d) All transitions are even better now: each transition now has an extra reward of 1 in addition to the reward you defined in (a). Assume $\gamma = 0.8$ as in part (c). How would the value function change? How would the policy change? Explain why.

   Answer:

   (a) Let all rewards be -1.

   (b) Optimal values:

-4	-3	-2	-1	0
5	4	3	2	1
4	-5	2	1	0
-5	-4	-3	-2	-1
-6	-5	-4	-3	-2

(c) No, changing γ changes the value function but not the relative order.

(d) The value function would change but the policy would not.

2. **Bellman Equation for Optimal State-Value Function**
In class we saw the recursive equation for the optimal action-value function Q^*. Similarly, derive (step-by-step) the recursive equation for the optimal state-value function V^*.

Answer:

$$
\begin{aligned}
V^*(s) &= \max_a Q^*(s,a) \\
&= \max_a \max_\pi \mathbb{E}_\pi\Big[\sum_{t=0}^{T} \gamma^t R_t \mid S_0 = s, A_0 = a\Big] \\
&= \max_a \max_\pi \Big[r(s,a) + \sum_{s'} p(s' \mid s,a)\mathbb{E}_\pi\big[\sum_{t=1}^{T} \gamma^t R_t \mid S_1 = s', S_0 = s, A_0 = a\big]\Big] \\
&= \max_a \Big[r(s,a) + \sum_{s'} p(s' \mid s,a) \max_\pi \mathbb{E}_\pi\big[\sum_{t=1}^{T} \gamma^t R_t \mid S_1 = s'\big]\Big] \\
&= \max_a \Big[r(s,a) + \gamma \sum_{s'} p(s' \mid s,a) \max_\pi \mathbb{E}_\pi\big[\sum_{t=0}^{T} \gamma^t R_t \mid S_0 = s'\big]\Big] \\
&= \max_a \Big[r(s,a) + \gamma \sum_{s'} p(s' \mid s,a) \max_\pi V^\pi(s')\Big] \\
&= \max_a \Big[r(s,a) + \gamma \sum_{s'} p(s' \mid s,a) V^*(s')\Big]
\end{aligned}
$$

Where $\mathbb{E}_\pi$ stands for the expectation over environment behavior when we follow policy π.

3. **Convergence of Value Iteration**
In class we saw how an iterative approach for policy evaluation converges by showing that the update operator is a *contraction.* Use the same technique to show that the *value iteration* algorithm converges.