TTIC 31170: Robot Learning and Estimation (Spring 2019)

Problem Set #3

Due Date: June 6, 2019

1 Markov Decision Processes [8 pts]

Given a Markov Decision Process (MDP) $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma, T \rangle$, policy iteration is one algorithm for determining the optimal policy. Policy iteration starts off with a random policy and alternates between a POLICYEVALUATION step, which computes the value function for the current policy, and a POLICYIMPROVEMENT step, which improves the policy based upon this value function. This continues until the change in the value function between successive steps is below a threshold.

(a) **[2pts]** When policy iteration converges to the optimal policy, the corresponding value function will be a fixed point of the Bellman update. Derive an expression for the resulting optimal value as a function of the transition likelihood $\mathcal{T}(s, \pi^*(s), s')$ and reward $\mathcal{R}(s, \pi^*(s), s')$. **Hint**: Treat $V^{\pi*}(s)$ and $\mathcal{R}(s, \pi^*(s), s')$ as column vectors and $\mathcal{T}(s, \pi^*(s), s')$ as a matrix.

(b) **[2pts]** Assuming policies are time-invariant, mapping states to actions, what is the size of the set of possible policies?

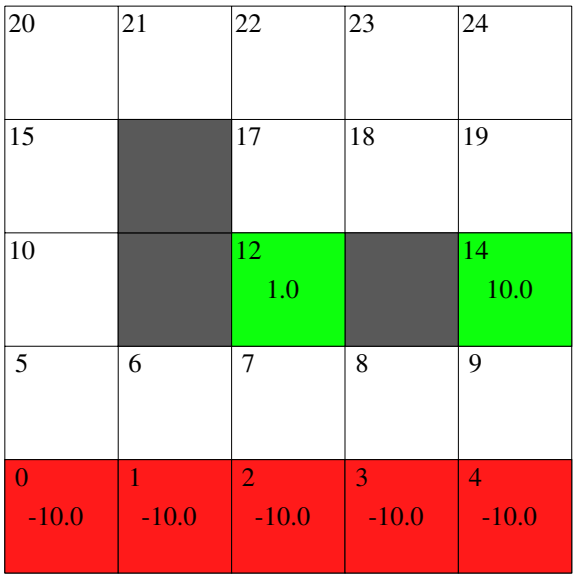
(c) **[4pts]** Provide the sketch of a proof that policy iteration is guaranteed to converge to the optimal policy.

2 Markov Decision Process: Grid World [16 pts]

Consider an agent that navigates in the grid world depicted below. The agent can move in each of the cardinal directions, $\mathcal{A} = \{\text{N}, \text{E}, \text{S}, \text{W}\}$, but the state transition is stochastic: if the agent tries to move North, it will go East or West, each with probability $\frac{p_{\text{noise}}}{2}$; if the agent tries to move East, it will go North or South, each with probability $\frac{p_{\text{noise}}}{2}$; if the agent tries to move South, it will go East or West, each with probability $\frac{p_{\text{noise}}}{2}$; and if the agent tries to move West, it will go North or South, each with probability $\frac{p_{\text{noise}}}{2}$. Black cells denote obstacles and if the agent moves towards an obstacle or the environment boundary, it will stay in its current cell, i.e., $P(s_{t+1} = 11 \,|\, s_t = 6, a = \text{North}) = 1 - p_{\text{noise}}$. Once the agent moves, it is able to observe its current state.

The states rendered as green and red are absorbing states. Executing any action in these states will conclude the episode. Taking an action in any of the states in the bottom row (a cliff) will result in a reward of $R(s_t, s_{t+1} = \text{end}) = -10.0$ for $s_t \in \{0, 1, 2, 3, 4\}$. Any action in states 12 or 14 result in a reward of 1.0 and 10.0, respectively.

The goal is to find the optimal policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$ that maximizes the expected discounted reward for an infinite time horizon. We can model this problem as an MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma, T \rangle$, where we are assuming that $T = \infty$.



(a) **[10pts]** One way to solve for the optimal policy is to use value iteration, whereby we initially assign each state a value of 0.0 and perform a series of Bellman updates (backups) until the updated value function is within ϵ of the previous value function:

$$\|V_{i+1}(s) - V_i(s)\| < \epsilon \quad \text{where } \|U\| = \max_s |U(s)|$$

Included with this problem is a file `GridWorld.py` that provides a Python class for the grid world problem. Implement value iteration within the `ValueIteration(epsilon)` function, which returns the optimal value function, the optimal policy, and the number of iterations necessary for convergence. You can validate your implementation by using the examples from class, where we considered different settings for p_{noise} and γ.

What is included: There are two files included with this problem set, `GridWorld.py` that defines the MDP class, which is the one that you should edit, and `RunMDP.py`, which calls the value iteration function and draws the resulting value function and optimal policy. You can run value iteration as follows:

```
$ python RunMDP.py --noise 0.2 --gamma 0.99 --epsilon 0.001
```

What to hand in: Your `GridWorld.py` and `RunMDP.py` files, along with the figure that depicts the optimal value function and policy for the above parameter settings.

(b) **[2pts]** Run value iteration for the following two parameter settings, $(p_{\text{noise}} = 0.2, \gamma = 0.1, \epsilon = 0.001)$ and $(p_{\text{noise}} = 0.2, \gamma = 0.99, \epsilon = 0.001)$. Provide the resulting figure for each of these settings and explain the differences in the resulting policies.

(c) **[2pts]** Compare the number of iterations necessary for different settings of the discount factor, $(p_{\text{noise}} = 0.2, \gamma = 0.1, \epsilon = 0.001)$ and $(p_{\text{noise}} = 0.2, \gamma = 0.99, \epsilon = 0.001)$. Why are more iterations required for one setting for γ over the other?

Now, consider the effect of an increased noise likelihood, and compare the number of required iterations for $(p_{\text{noise}} = 0.2, \gamma = 0.99, \epsilon = 0.001)$ and $(p_{\text{noise}} = 0.5, \gamma = 0.99, \epsilon = 0.001)$. Explain the difference.

(d) **[2pts]** If sequential backups of the value function are within ϵ according to the above norm, one can show that the error in the value function estimate relative to the optimal value function is bounded as

$$\|V_{i+1}(s) - V^*(s)\| < \frac{2\epsilon\gamma}{1-\gamma}$$

Explain the dependence on γ.

3 Reinforcement Learning [4 pts]

(a) **[4pts]** Consider an agent acting in an environment consisting of two states $\mathcal{S} = \{x, y\}$. We would like to estimate the value function (i.e., $V^\pi(x)$ and $V^\pi(y)$) for a given policy in a model-free way based upon observations of state-reward sequences. In particular, suppose that we observe the following episodes (where absence of an entry for $(s_2, \mathcal{R}_2)$ indicates that the episode terminated)

Episode	$(s_1, \mathcal{R}_1)$, $(s_2, \mathcal{R}_2)$
1	$(x, 0)$, $(y, 0)$
2	$(y, 1)$
3	$(y, 1)$
4	$(y, 1)$
5	$(y, 1)$
6	$(y, 0)$
7	$(y, 1)$
8	$(y, 1)$

What estimate of the value function would you get if you used the TD(0) algorithm? What would you get if you used Monte Carlo prediction? Compare the results in terms of how correct they are for the given episodes and how they may generalize.

4 Time and Collaboration Accounting

(a) **[0.5pts]** Did you work with anyone on this problem set? If so, who?

(b) **[0.5pts]** How long did you spend on this problem set?