TTIC 31170: Robot Learning and Estimation (Spring 2019)

Problem Set #4 Solutions

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.

Solution:

The value function associated with the optimal policy is a fixed-point of the Bellman equation

$$V^{\pi^*}(s) = \sum_{s'} \mathcal{T}(s, \pi^*(s), s') \Big(\mathcal{R}(s, \pi^*(s), s') + \gamma V^{\pi^*}(s') \Big)$$

If we let $\mathbb{P}$ be an $|\mathcal{S}| \times |\mathcal{S}|$ matrix, where $\mathbb{P}_{ij} = \mathcal{T}(s_i, \pi^*(s_i), s_j)$, $\mathbb{R}$ be a vector of dimension $|\mathcal{S}|$[1], where $\mathbb{R}_i = \mathcal{R}(s_i, \pi^*(s_i), s_j)$, and $\mathbb{V}^{\pi^*}$ be a a vector of dimension $|\mathcal{S}|$, where $\mathbb{V}_i^{\pi^*} = V^{\pi^*}(s_i)$, we can write the equality as

$$\mathbb{V}^{\pi^*} = \mathbb{P} \left(\mathbb{R} + \gamma \mathbb{V}^{\pi^*} \right)$$

Rearranging terms, we have

$$\boxed{\mathbb{V}^{\pi^*} (\mathbb{I} - \gamma \mathbb{P})^{-1} \mathbb{R}}$$

■

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

[1] This assumes that the reward is a function of the subsequent state s' but not the current state s.

Solution:

There are $|\mathcal{S}|$ states and a policy can assign one of $|\mathcal{A}|$ actions to each state. Thus, the total number of policies is $|\mathcal{A}|^{|\mathcal{S}|}$. ■

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

Solution:

Intuitively, given the policy at iteration i, policy evaluation will converge to the corresponding action-value function. The subsequent policy improvement step greedily chooses the immediate action that maximizes the estimate of the action-value function and subsequently follows the corresponding policy. The value of this new policy is guaranteed to be equal to or greater than that of the policy at step i. The value under the two policies is equal only when the policies correspond 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, $\mathcal{A} = \{\mathrm{N}, \mathrm{E}, \mathrm{S}, \mathrm{W}\}$, 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., $P(s_{t+1} = 11 \mid s_t = 6, a = \text{North}) = 1 - p_{\text{noise}}$. Once the agent moves, it is able to observe its current state.

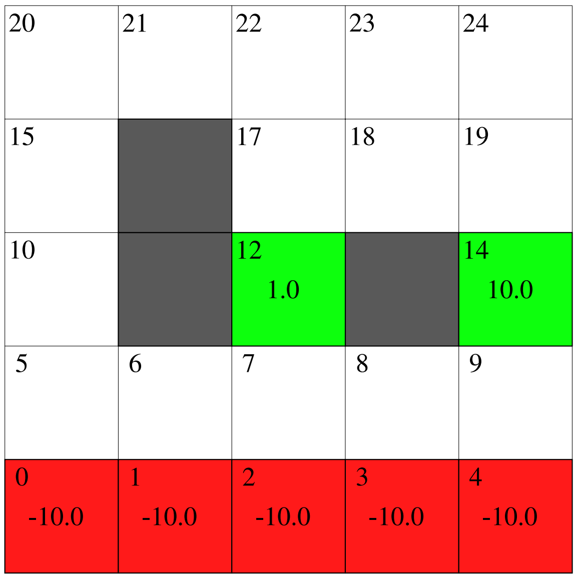
The states rendered as green and red are absorbing states. Executing any action in these states will conclude the episode. Taking an action in any of the states in the bottom row (a cliff) will result in a reward of $R(s_t, s_{t+1} = \text{end}) = -10.0$ for $s_t \in \{0, 1, 2, 3, 4\}$. Any action in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

Solution:

■

(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.

Solution:

Figure 2 compares the different policies and their associated value functions. In the case of $\gamma = 0.1$, the optimal policy is to take an action that leads directly to the closest positive-reward state (e.g., the optimal policy in state 17 is to move down). When $\gamma = 0.99$, however, the optimal policy takes a longer path towards the higher reward state (e.g., moving up to reach state 14 when in state 17). This is due to the fact that a low discount factor results in a preference for near-term rewards relative to larger settings for the discount factor. ■

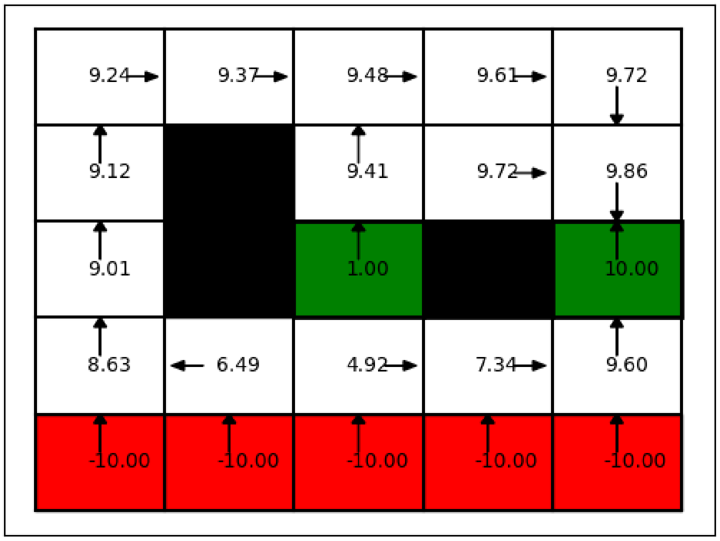


Figure 1: Optimal policy and corresponding value function for noise $= 0.2$, $\gamma = 0.99$, $\epsilon = 0.001$.

(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.

Solution:

With lower discount factors, the myopic nature of the task means that fewer iterations are necessary to converge to the optimal policy and value function than with larger discount factors. In this case, value iteration converges within 5 iterations for $\gamma = 0.1$ and 28 iterations for $\gamma = 0.99$.

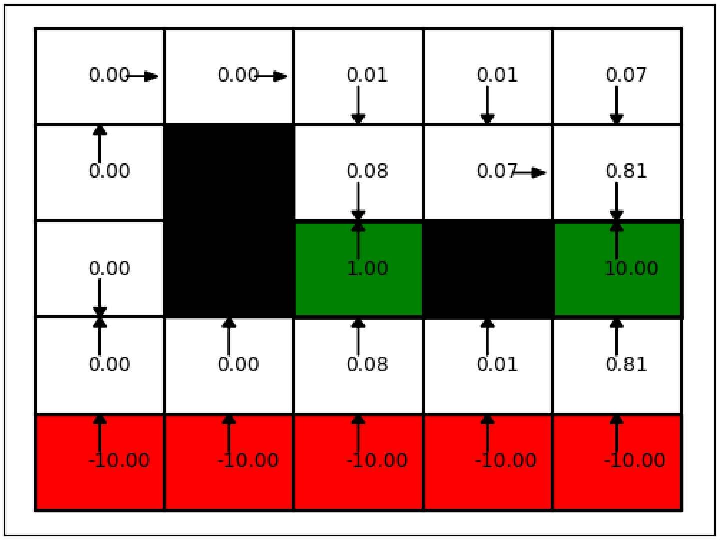
When uncertainty in the transition function is greater, more iterations are required for convergence due to the additional stochasticity. Increasing the noise term from 0.2 to 0.5 results in 56 iterations required for convergence. ■

(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

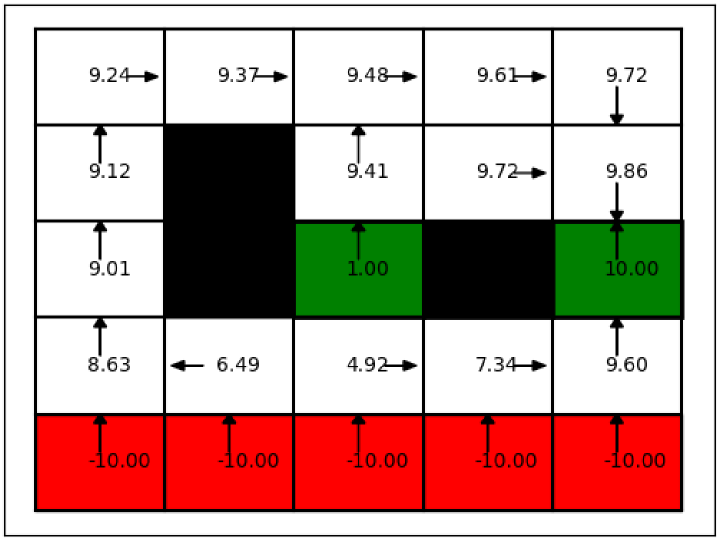
$$\|V_{i+1}(s) - V^*(s)\| < \frac{2\epsilon\gamma}{1-\gamma}$$

Explain the dependence on γ.

Solution:



(a) noise $= 0.2$, $\gamma = 0.1$, $\epsilon = 0.001$



(b) noise $= 0.2$, $\gamma = 0.99$, $\epsilon = 0.001$

Figure 2: Optimal policy and corresponding value function for different discount factors.

Larger discount factors place more emphasis on longer-term rewards, which must be accounted for in the backups. This leads to slower convergence of the value function, as we saw empirically above for value iteration. ■

3 Reinforcement Learning: Monte Carlo vs TD(0) [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.

Solution:

Monte Carlo policy evaluation waits till the end of an episode to update the estimated value of each state. State x is only encountered in the first episode and no reward is received after being in state x, the estimated value of that state will be $V(x) = 0$. Looping over all episodes, the MC-based estimate of the value for state y will be $V(y) = 6/8 = 3/4$, since the total reward received from state y across 8 episodes is 6.

TD(0), in contrast, performs incremental updates. In a batch setting, TD(0) would also estimate the value of state y as $V(y) = 3/4$, but since the agent always transitions to state y when in state x, TD(0) would estimate a value of $V(x) = 3/4$ to state x.

As discussed in class, batch MC converges to the minimum mean square error estimate, i.e., the one that maximizes the likelihood of the returns. TD(0), on the other hand, converges to the value that you would get if you used dynamic programming on the maximum likelihood model estimated from the data (i.e., estimate of the probability of transitioning from x to y would be 1, the probability of the episode terminating from state y would be estimated as being 1, and the estimates of the immediate rewards when transitioning from x and y would be 0 and $3/4$, respectively). $\blacksquare$