CS 234 Winter 2023
Assignment 1
Due: January 20 at 6:00 pm (PST)

For submission instructions, please refer to the website. For all problems, if you use an existing result from either the literature or a textbook to solve the exercise, you need to cite the source.

1 Reward choices [15 pts]

Consider a tabular MDP with $0 < \gamma < 1$ and no terminal states. The agent will act forever. The original optimal value function for this problem is V_1^* and the optimal policy is π_1^*.

(a) Now someone decides to add a small reward bonus c to all transitions in the MDP. This results in a new reward function $\hat{r}(s, a) = r(s, a) + c;\ \forall s, a$ where $r(s, a)$ is the original reward function. What is an expression for the new optimal value function? Can the optimal policy in this new setting change? Why or why not? [5 pts]

(b) Instead of adding a small reward bonus, someone decides to multiply all rewards by an arbitrary constant $c \in \mathbb{R}$. This results in a new reward function $\hat{r}(s, a) = c \times r(s, a);\ \forall s, a$ where $r(s, a)$ is the original reward function. Are there cases in which the new optimal policy is still π_1^* and the resulting value function can be expressed as a function of c and V_1^* (if yes, give the resulting expression and explain for what value(s) of c and why? If not, explain why not)? Are there cases where the optimal policy would change (if yes, provide a value of c and a description of why the optimal policy would change. If not, explain why not)? Is there a choice of c such that all policies are optimal? [5 pts]

(c) If the MDP instead has terminal states that can end an episode, does that change your answer to part (a)? If yes, provide an example MDP where your answers to part (a) and this part would differ. [5 pts]

2 Bellman Residuals and performance bounds [25 pts]

In this problem, we will study Bellman residuals, which we will define below, and how it helps us bound the performance of a policy. But first, some recap and definitions.

Definitions: From lecture, we know that the Bellman backup operator B, defined below is a contraction with the fixed point as V^*, the optimal value function of the MDP. The symbols have their usual meanings. γ is the discount factor and $0 \leq \gamma < 1$. In all parts, $||v||$ is the infinity norm of the vector.

$$(BV)(s) = \max_a [r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a)V(s')]$$

We also saw the contraction operator B^π with the fixed point V^π, which is the Bellman backup operator for a particular policy given below:

$$(B^\pi V)(s) = \mathbb{E}_{a \sim \pi}[r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a)V(s')]$$

In class, we showed that $||BV - BV'|| \leq \gamma ||V - V'||$ for two arbitrary value functions V and V'.

For the rest of this question, you can also assume that $||B^\pi V - B^\pi V'|| \leq \gamma ||V - V'||$. The proof of this is very similar to the proof we saw in class for B.

(a) Prove that the fixed point for B^π is unique. [3 pts]

We can recover a greedy policy π from an arbitrary value function V using the equation below.

$$\pi(s) = \arg\max_a [r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a)V(s')]$$

(b) When π is the greedy policy, explain what is the relationship between B and B^π? [2 pts]

Motivation: It is often helpful to know what the performance will be if we extract a greedy policy from a value function. In the rest of this problem, we will prove a bound on this performance.

Recall that a value function is a $|S|$-dimensional vector where $|S|$ is the number of states of the MDP. When we use the term V in these expressions as an "arbitrary value function", we mean that V is an arbitrary $|S|$-dimensional vector which need not be aligned with the definition of the MDP at all. On the other hand, V^π is a value function that is achieved by some policy π in the MDP. For example, say the MDP has 2 states and only negative immediate rewards. $V = [1, 1]$ would be a valid choice for V even though this value function can never be achieved by any policy π, but we can never have a $V^\pi = [1, 1]$. This distinction between V and V^π is important for this question and more broadly in reinforcement learning.

Why do we care about setting V to vectors than can never be achieved in the MDP? Sometimes algorithms, such as Deep Q-networks, return such vectors. In such situations we may still want to extract a greedy policy π from a provided V and bound the performance of the policy we extracted

aka V^π.

Bellman Residuals: Define the Bellman residual to be $(BV - V)$ and the Bellman error magnitude to be $||BV - V||$. As we will see through the course, this Bellman residual is an important component of several popular RL algorithms such as the Deep Q-networks, referenced above. Intuitively, you can think of it as the difference between what the value function V specifies at a state and the one-step look-ahead along the seemingly best action at the state using the given value function V to evaluate all future states (the BV term).

(c) For what V does the Bellman error magnitude equal 0? [1 pts]

(d) Prove the following statements for an arbitrary value function V and any policy π. [5 pts]
Hint: Try leveraging the triangle inequality by inserting a zero term.

$$||V - V^\pi|| \leq \frac{||V - B^\pi V||}{1 - \gamma}$$

$$||V - V^*|| \leq \frac{||V - BV||}{1 - \gamma}$$

The result you proved in part (d) will be useful in proving a bound on the policy performance in the next few parts. Given the Bellman residual, we will now try to derive a bound on the policy performance, V^π.

(e) Let V be an arbitrary value function and π be the greedy policy extracted from V. Let $\varepsilon = ||BV - V||$ be the Bellman error magnitude for V. Prove the following for any state s. [5 pts]
Hint: Try to use the results from part (d).

$$V^\pi(s) \geq V^*(s) - \frac{2\varepsilon}{1 - \gamma}$$

A little bit more notation: define $V \leq V'$ if $\forall s$, $V(s) \leq V'(s)$.

What if our algorithm returns a V that satisfies $V^* \leq V$, i.e., it returns a value function that is better than the optimal value function of the MDP. Once again, remember that V can be any vector, not necessarily achievable in the MDP but we would still like to bound the performance of V^π where π is extracted from said V. We will show that if this condition is met, then we can achieve an even tighter bound on policy performance.

(f) Using the same notation and setup as part (e), if $V^* \leq V$, show the following holds for any state s. [5 pts]
Hint: Recall that $\forall \pi$, $V^\pi \leq V^*$. (why?)

$$V^\pi(s) \geq V^*(s) - \frac{\varepsilon}{1 - \gamma}$$

(g) It's not easy to show that the condition $V^* \leq V$ holds because we often don't know V^* of the MDP. Show that if $BV \leq V$ then $V^* \leq V$. Note that this sufficient condition is much easier to check and does not require knowledge of V^*. [4 pts]
Hint: Try to apply induction. What is $\lim_{n \to \infty} B^n V$?

Intuition: A useful way to interpret the results from parts (e) (and (f)) is based on the observation that a constant immediate reward of r at every time-step leads to an overall discounted reward of $r + \gamma r + \gamma^2 r + \ldots = \frac{r}{1-\gamma}$. Thus, the above results say that a state value function V with Bellman error magnitude ε yields a greedy policy whose reward per step (on average), differs from optimal by at most 2ε. So, if we develop an algorithm that reduces the Bellman residual, we're also able to bound the performance of the policy extracted from the value function outputted by that algorithm, which is very useful!

Challenge: It is possible to make the bounds from parts (e) and (f) tighter. Try to prove the following if you're interested. **This part will not be graded.**

Let V be an arbitrary value function and π be the greedy policy extracted from V. Let $\varepsilon = ||BV - V||$ be the Bellman error magnitude for V. Prove the following for any state s:

$$V^{\pi}(s) \geq V^{*}(s) - \frac{2\gamma\varepsilon}{1-\gamma}$$

Further, if $V^{*} \leq V$, prove for any state s

$$V^{\pi}(s) \geq V^{*}(s) - \frac{\gamma\varepsilon}{1-\gamma}$$

3 Frozen Lake MDP [25 pts]

Now you will implement value iteration and policy iteration for the Frozen Lake environment from Gymnasium and originally from OpenAI Gym. We have provided custom versions of this environment in the starter code.

Setup: This assignment needs to be completed with Python3.8+. We have provided a `requirements.txt` file that contains all the associated Python libraries you will need to complete this assignment. You can go through the file and install all the libraries by running `pip install -r requirements.txt` in the terminal. We highly recommend that you do this in a virtual environment dedicated for this assignment to avoid any dependency conflicts. **While programming, please ignore any `DeprecationWarning` you may see**.

Submission: There is a `Makefile` provided that will help you submit the assignment. Please run `make clean` followed by `make submit` in the terminal and submit the resulting zip file on Gradescope.

(a) **(coding)** Read through `vi_and_pi.py` and implement `policy_evaluation`, `policy_improvement` and `policy_iteration`. The stopping tolerance (defined as $\max_s |V_{old}(s) - V_{new}(s)|$) is tol = 10^{-3} . Use $\gamma = 0.9$. Return the optimal value function and the optimal policy. [10pts]

(b) **(coding)** Implement `value_iteration` in `vi_and_pi.py`. The stopping tolerance is tol = 10^{-3} . Use $\gamma = 0.9$. Return the optimal value function and the optimal policy. [10 pts]

(c) **(written)** Run both methods on the Deterministic-4x4-FrozenLake-v0 and Stochastic-4x4-FrozenLake-v0 environments. In the second environment, the dynamics of the world are stochastic. How does stochasticity affect the number of iterations required, and the resulting policy, quantitatively and qualitatively? [5 pts]

Sanity Check: For the Deterministic-4x4-FrozenLake-v0 environment, the values for the first three states are `0.59, 0.656, 0.729`. The value functions from VI and PI should be within error tolerance 10^{-3} of these values. You can use this to verify your implementation. For grading purposes, we shall test your implementation against other hidden test cases as well.