CS 234 Spring 2017
Assignment 1
Due: April 19, 1:30 pm

1 Bellman Operator Properties

Consider an MDP $M = (S, A, R, P, \gamma)$ where $|S| < \infty$ and $|A| < \infty$. Recall the Bellman operator T is defined such that:

$$(TV)(s) = \max_{a \in A} \left\{ R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V(s') \right\}, \forall s \in S\,.$$

For a policy π, the Bellman operator associated with π, T_π, is defined such that:

$$(T_\pi V)(s) = \mathbb{E}_\pi \left[R(s,\pi(s)) + \gamma \sum_{s' \in S} P(s'|s,\pi(s)) V(s') \right], \forall s \in S\,.$$

We consider the following properties of these operators. Note $=$ and $\leq$ are element-wise comparisons. For example, $V \leq V'$ if $V(s) \leq V'(s)$ for all $s \in S$.

(a) Let V and V' be two value functions. Prove for any policy π, $V \leq V' \Rightarrow T_\pi V \leq T_\pi V'$. Also, prove $V \leq V' \Rightarrow TV \leq TV'$.

(b) Let c be a constant and $\vec{1}$ be the all-ones vector. Prove for any policy π, $T_\pi(V + c\vec{1}) = T_\pi V + \gamma c\vec{1}$. Also, prove $T(V + c\vec{1}) = TV + \gamma c\vec{1}$.

(c) For $\alpha, \beta \geq 0$ and $\alpha + \beta = 1$, prove $T(\alpha V + \beta V') \leq \alpha TV + \beta TV'$.

(d) Prove or disprove the following statement: For all V, $TV \leq V$.

2 Value Iteration

(a) After each iteration of value iteration you can extract the current best policy (given the current value function, by taking an argmax instead of max). If the best policy at step k is the same as the best policy at step $k+1$ when doing value iteration, can the policy ever change again (with further iterations of value iteration)? Prove it cannot or disprove with a counterexample.

(b) Consider an MDP problem with finite state space and finite action set. Empirically we often halt value iteration after a fixed number of steps for computational reasons, yielding an approximately optimal value function $\tilde{V}$. If we then extract the best policy for $\tilde{V}$ (by taking the argmax of one more backup) we will get a policy. Let's assume we know that the maximum difference across any state is (for extra understanding, think about how the difference in $|BV - V|$ allows us to ensure this)

$$|V^*(s) - \tilde{V}(s)| \leq \epsilon, \text{for all } s$$

Then prove that the resulting best policy extracted from $\tilde{V}$ will also have a performance that is close to the performance of the optimal policy. In particular, for a greedy policy $\pi_{\tilde{V}}$ derived from $\tilde{V}$, define the loss function $L_{\tilde{V}}$ as follows:

$$L_{\tilde{V}}(s) = V^*(s) - V_{\pi_{\tilde{V}}}(s), \text{ for all } s$$

where V^* is the optimal value function and $V_{\pi_{\tilde{V}}}$ is the value function associated with $\pi_{\tilde{V}}$. Prove

$$L_{\tilde{V}}(s) \leq \frac{2\gamma\epsilon}{1-\gamma}$$

If you use outside sources you must write up your own solution and cite the source.

3 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 21 goes to state 20) 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. 11) earns a reward of $+5$ and ends the episode. Taking any action from the red square of death (no. 15) 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 $\gamma = 1$ unless otherwise specified.

20	21	22	23	24
15	16	17	18	19
10	11	12	13	14
5	6	7	8	9
0	1	2	3	4

(a) Define the reward r for all states (except state 15 and state 11 whose rewards are specified above) that would cause the optimal policy to return the shortest path to the green target square (no. 11).

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

(c) Does setting $\gamma = 0.5$ change the optimal policy? Why or why not?

(d) The green target state is now even more awesome! Its reward is $+20$. Assuming the rest of the rewards for all states are the same as you defined in part (a), does the value function change and why? Does the policy change?

4 Frozen Lake MDP

Now you will implement value iteration and policy iteration for the Frozen Lake environment from OpenAI Gym (https://gym.openai.com/envs/FrozenLake-v0). We have provided custom versions of this environment in the starter code.

(a) **(coding)** Implement policy 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.

(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.

(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?

5 Frozen Lake Reinforcement Learning

Continuing with only the `Stochastic-4x4-FrozenLake-v0` environment, you will now do full reinforcement learning using model-based and model-free methods! This problem is considered solved when 100 consecutive trials achieve an average score of 0.78, so you can use this to check the performance of your agent.

(a) **(coding)** Implement Q-learning with ϵ-greedy exploration in `model_free_learning.py`. Return an array of the Q values for each state-action pair.

(b) **(coding)** Implement SARSA with ϵ-greedy exploration in `model_free_learning.py`. Return an array of the Q values for each state-action pair.

(c) **(coding, written)** Plot the running average score of the Q-learning agent for the first 1000 training episodes. The x-axis should be the episode number, and the y-axis should be the average score over all training episodes so far. Include the plot in your writeup.

(d) **(coding, written)** Now implement model-based learning in `model_based_learning.py` where after each episode, you train an MDP model using the experience from that episode and then use value iteration on your updated model of the environment to compute a current best policy. Explore using ϵ-greedy. Plot performance on the same axis as part (c). Compare and contrast Q-learning and model-based learning, including at least one benefit of each. For model-based learning, your agent does not always have to win the environment, but it should consistently score an average of 0.5 or higher.