University of Pittsburgh
CS1675 Introduction to Machine Learning
Professor Milos Hauskrecht

Handout 11
April 15, 2021

Problem assignment 11

Due: Thursday, April 22, 2021

Problem 1. Reinforcement learning agent

In this assignment we implement and experiment with a simple reinforcement learning agent for learning the optimal policy by directly interacting with the environment. The environment has 4 states labeled {1,2,3,4} and 3 actions labeled {1 2 3}. The environment is stochastic so it may transition into many different states given the current state and the current action.

Part a. You were given basic code that lets you simulate the environment and its interactions with the agent. It consists of the following scripts and functions:

- Script RL_init_model that loads the model of the environment from file model.mat and keeps it stored in the variable model.
- Function [init_state] = RL_reset_environment(model) that places the agent into initial state (selected randomly based on prior)
- Function [new_state reward] = RL_simulate_one_step(current_state, action, model) that simulates the environment and its behavior for one step using the current state and the selected action, and returns the new state and the reward for the transition.

Part b. A policy maps states to actions. In general, a policy may assign a different action to every possible state. Let us assume the policy is defined by a vector with its index representing the state and values representing one of the actions. For example, the vector $(1, 1, 3, 2)^T$ would represent a policy that picks action 1 for state 1, action 1 for state 2, action 3 for state 3, and action 2 for state 4. Write a code that lets you simulate the model and the policy $(1, 1, 3, 2)^T$ for 10 steps and generate a sequence of state-action-reward-trajectories generated by the model and the policy. To first (initial state) state should be selected using the function RL_reset_environment(model). Please report the state-action-reward trajectory for 10 steps.

Part c. Our next goal is to evaluate a policy. Let's assume the quality of any policy for the reminder of this assignment is measured in terms of the infinite horizon expected

discounted reward. Let us also assume the discount factor is $\gamma = 0.95$. Now let us see how we can determine the value of the policy via interaction with our simulated environment. In general, the value of the policy depends on the start state. Let $V^\pi(s)$ denotes the value of the policy (expected discounted reward) starting from state s. Since there are 4 states V^π can be defined by a vector of 4 numbers, each representing the value of the policy for a respective state.

The value of the policy can be estimated from interaction with the environment, that is, by applying the policy to the (simulated) environment and by relying on the reward and the new state information received from the environment. More specifically, we can do this by iterative re-estimation process that keeps an estimate of the vector $\hat{V}^\pi$, and updates its $\hat{V}^\pi(s)$ component whenever we get into state s. Briefly, we update $\hat{V}^\pi(s)$ as follows:

$$\hat{V}^\pi(s) \leftarrow (1 - \alpha(n(s)))\hat{V}^\pi(s) + \alpha(n(s)) \left[r + \gamma \hat{V}^\pi(s') \right]$$

where r is the reward received from the environment after applying the action $\pi(s)$ to state s, and where s' is the new (next) state. The parameter $\alpha(n(s))$ is the learning rate that depends on $n(s)$, the number of times the state s has been visited before.

Implement the above online policy evaluation algorithm for estimating $\hat{V}^\pi$ from interactions with the environment and run it for 15,000 steps. Set the estimate of $\hat{V}^\pi(s)$ for all states s to 0 at the beginning. Please reset the environment every 200 steps to its initial state (do not reset $\hat{V}^\pi$ at that time!!!). Please use:

$$\alpha(n(s)) = \frac{1}{(1 + n(s))^{0.6}}$$

as the learning rate.

Please record the actual values of $\hat{V}^\pi$ first at the beginning (when values are 0), after every 100 steps, and also at the end (after 15,000 steps). Plot the changes in the $\hat{V}^\pi$ values over 15,000 steps in a graph and include the graph in the report. By inspecting the graph do you think $\hat{V}^\pi$ estimates have converged? Explain. If you think the values have not converged keep adding more simulation steps in increments of 15,000, and repeat the analysis.

Part d. Pick a policy π' on your own. Repeat Part c. for the new policy π' and compare the results for π and π' (at 15,000 steps). Which policy is better? π or π'? Explain why?

Part e. Now let us consider and implement the Q-learning algorithm that aims to estimate the optimal action-value functions $\hat{Q}(s, a)$ by directly interacting with the environment. The Q function update is defined as:

$$\hat{Q}(s, a) \leftarrow (1 - \alpha(n(s, a)))\hat{Q}(s, a) + \alpha(n(s, a)) \left[r + \gamma \max_{a'} \hat{Q}(s', a') \right]$$

where r is the reward received from the environment after applying action a to state s, and where s' is the new (next) state. The parameter $\alpha(n(s, a))$ is the learning rate that depends on $n(s, a)$, the number of times action a has been applied to state s.

Implement and submit the above Q-learning algorithm in a matlab script file called Q_learning. Please initialize the learning process by setting the estimates of $\hat{Q}(s,a)$ for all states s and actions a to 0 at the beginning. To decide what action a to chose in state s please use ϵ greedy strategy. In particular, implement the strategy with $\epsilon = 0.4$, that is, when arriving to state s, pick the action with the highest $\hat{Q}(s,a)$ value with probability 0.6 and any of the remaining actions with probability 0.2. Please use:

$$\alpha(n(s,a)) = \frac{1}{(1+n(s,a))^{0.6}}$$

as the learning rate. Similarly to Parts c and d run the above Q-learning algorithm for 15,000 steps. Reset the environment every 200 steps to its initial state (do not reset $\hat{Q}(s,a)$ estimates!!!).

At the end of 15,000 simulation steps compute the policy $\pi^*(s) = \arg\max_a \hat{Q}(s,a)$ and $V^*(s) = \max_a \hat{Q}(s,a)$. Is the learned policy better than the policies you have tried in Part c and d? Support your answer by comparing V^* and $\hat{V}^\pi$ functions and their values.