COMP-579: Reinforcement Learning - Assignment 2

Posted Tuesday, January 30, 2024
Due Thursday, February 15, 2024

The assignment contains a coding section and a math section. Please submit a PDF/IPYNB file containing your solutions to the problems, and the code you used to produce the results.

1. **Coding: Tabular RL [70 points]**

   In this problem, compare the performance of SARSA and expected SARSA on the Taxi Problem from the Gym environment suite:

   `https://gymnasium.farama.org/environments/toy_text/taxi/`

   You can choose to start with the template notebook below. If you are not familiar with the Gym API, play around with it first. With all the functions ready, running all the experiments should not take more than 1 hour.

   `https://colab.research.google.com/drive/1St75TEHP2n5RsuuHHQeW0IpCct7t_Fj6?usp=sharing`

   Using a **tabular** representation of the state space, test different temperatures and learning rates, and measure the return overtime. The agent's exploration should be softmax (Boltzmann).

   For each experiment (hyperparamter combination), do 10 independent trials. Each trial consists of 500 segments. In each segment, there are 10 episodes of training, followed by 1 testing episode where you run the optimal policy so far (i.e. pick actions greedily). In other words, there are 5500 episodes in each trial. Pick at least 3 settings of the temperature parameter used in the exploration and at least 3 settings of the learning rate. Plot:

   - One graph that shows the effect of the parameters on the final ***training*** performance.

     The x-axis shows the different parameters (e.g. learning rate), and the y-axis shows the return of the agent (averaged over the last 10 training episodes and the 10 runs); note that this will typically end up as an upside-down U.

     For each algorithm, the graph should have at least 3 lines (e.g. 3 temperature values) corresponding to the choices of the other hyperparameter. There should be at least 6 lines in the graph, and at least 18 different points. Also plot the uncertainty using shading (e.g. using the min and the max return of the 10 runs)
   - The same graph that instead shows the effect of the parameters on the final ***testing*** performance. The y-axis should now show the return during the final testing episode, averaged over the 10 runs.
   - Learning curves (mean and standard deviation computed based on the 10 runs) for the best parameter setting for each algorithm. X-axis shows the segment, Y-axis shows return overtime.

   Feel free to create other figures to help you analyze the results. Write a small report that describes your experiment, your choices of parameters, and the conclusions you draw from the graphs.

2. **Math: Bellman equation and dynamic programming [20 points]** Suppose we are in an MDP and the reward function has the structure $r(s,a) = \alpha r_1(s,a) + \beta r_2(s,a)$ where α, β are real numbers. In this question, we investigate whether we could solve the problems defined by reward functions r_1 and r_2 independently and then somehow combine them linearly to solve the problem defined by r. Assume the discounted setting with a given discount γ.

   (a) [10 points] Suppose you are given the action-value functions q_1^π and q_2^π corresponding to the action-value function of an arbitrary, fixed policy π under the two reward functions. Using the Bellman equation, show if it is possible or not to combine these value functions in a simple manner (linear) to obtain q^π corresponding to reward function r.

   (b) [10 points] Suppose you are given the optimal action-value functions q_1^* and q_2^*. Using the Bellman equation, explain if it is possible or not to combine these value functions in a simple manner (linear) to obtain q^* which optimizes reward function r.

3. **Math: An alternative learning algorithm [10 points]** Consider a learning algorithm which attempts to learn a Q-function, but instead of using the usual Q-learning target $R + \gamma \max_a Q(s', a)$, it uses as target a mixture of

$$R + \gamma((1-\alpha)\max_a Q(s',a) + \alpha \sum_a \pi(s',a)Q(s',a))$$

   where $\alpha \in (0,1)$ is a hyper-parameter.

   Assume that π is an ϵ-greedy policy derived from Q, and the episodes used for training are collected using π only.

   (a) [5 points] Recall that an on-policy control algorithm estimates $q_\pi(s,a)$ for the ***current*** behaviour policy π and for all states s and actions a. Is this algorithm on-policy or off-policy? Justify your answer.

   (b) [5 points] For different values of α, how would you expect this algorithm to perform compared to Q-learning and SARSA? Include bias, variance, and maximization bias in your discussion.

   (c) [5 points] Bonus question: try this algorithm on the Taxi Problem in Question 1, and compare it to the other algorithms. Are the results consistent with your hypothesis?