UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR ROBOTICS AND COGNITIVE SYSTEMS

Reinforcement Learning (SS2021)
Lecturer: Dr. Christoph Metzner
c.metzner@campus.tu-berlin.de
Teaching Assistant: Honghu Xue
xue@rob.uni-luebeck.de
Submission Deadline: May.26th

Reinforcement Learning - Assignment II (21 pts)

The relevant reference link for this assignment is illustrated in the Literature part below. You need to fetch the python files (zip files) for the programming part. The submission should be a zip file of PDF, related python files and a txt file showing the name of the team members in the group and student matriculation number. **You do not need to repeat the question in the report. The zip file is with the name of "RL_Assignment_index number_group leader name". The final PDF should not exceed 6 pages. Any delayed submission will not be counted for the score. Make sure you don't copy any figures or sentences from the books or from other groups, otherwise you won't get points for that.**

In this assignment, you will learn the concepts of model-free Reinforcement Learning algorithms Monte-Carlo approaches and one-step Temporal-Difference Learning. You will also apply the relevant algorithms in the Gym environments 'Frozenlake-v0' and 'MountainCar-v0'. This assignment also covers some common exploration strategies in RL and its effect on the final performance. Good Luck!

Task I: Theoretical understanding on Monte-Carlo Methods and Temporal Difference Learning

1. What is the key difference between Dynamic Programming (DP) approaches and Monte-Carlo (MC)/ Temporal Difference Learning (TD) approaches, i.e. in which cases can DP approaches not be applied but MC and TD learning can still work? (1 point)
2. Explain what is on-policy learning, off-policy learning in detail. (1.5 points)
3. Explain the stationary policy and non-stationary policy and its relation with the choice of step size α for updating the state value V or Q value. (1.5 points)
4. Apart from a proper choice of the step size α and the discount factor γ, what is another necessary condition to be fulfilled for convergence to an optimal policy in MC and TD approaches in a discrete state-action space? (1 point)
5. Besides the common exploration strategy of (decaying) ϵ-greedy exploration, please explain another exploration strategy, you can choose either one of the following: (i) *Upper-Confidence-Bound Action Selection* (ii) *Boltzmann exploration* (also called *softmax exploration*). Please provide math expressions and explain the term. (1.5 points, for (i), please refer to the book Chapter 2)

Task II: Programming part on MC and on-policy and off-policy one-step TD methods

For those who use the Google Colab, please paste all the code from each '.py' file to the new code block in 'Gym+Pybullet_on_Colab.ipynb'. Don't use 'env.render()' in the program, as it causes failure in rendering the scene in Colab platform. I am seeking for solutions. For google Colab users, you can only see the reward, but unfortunately no animations on your trained policy. You can turn to your team members for animations.

For local users, it is recommended that in the training phase, you don't call the function 'env.render()' so as to save time. In the testing phase, you could render the environment to see the trained result.

1. Implement the algorithm *On-policy first-visit MC control* with ϵ-greedy exploration strategy on the OpenAI Gym environment "frozenlake-v0", where $\epsilon = 0.4$. Show the final learned policy in the same format as in Assignment I. (3 points)
2. Implement the SARSA algorithm using linearly-decaying ϵ-greedy exploration strategy on the adapted "MountainCar-v0" environment with fixed step size ($\alpha = 0.05$). An example of linearly-decaying ϵ-greedy exploration strategy is $\epsilon = 1-(E_c/E_t)$, where E_c and E_t refers to the index of the current episodes and the number of the total training episodes respectively. This means the agent starts exploring the environment with $\epsilon = 100\%$ (full exploration) and exploit more with the training episodes. Show the plot (episodic reward w.r.t number of training episodes) with multiple runs, one run per team member. **Merge all the**

plots of your team in one plot using the template. Wisely choose your axis range for a clear presentation. Hint: you need save the statistics as .npy file and plot again. The episodic reward of a the finally learned policy is around -150. **In the testing phase, please constrain the maximal episode length to** 1000 to avoid infinite loop of interactions. (2 points)

3. Implement the one step Q-learning algorithm with decaying ϵ-greedy exploration strategy on the adapted "MountainCar-v0" environment with fixed step size ($\alpha = 0.05$). Show the plot (episodic reward w.r.t number of training episodes) in the same format as the task of SARSA. The code of this task is very similar to that in SARSA. The episodic reward of the learned policy is around -150. In the testing phase, also constrain the maximal episode length to 1000 to avoid infinite loop of interactions. (2 points)

Task III: Going Deeper

1. Compare the results of the learned policy of the MC approaches with the learned policy of PI/VI in Assignment I. Why can it happen that some states (e.g. state 2) show a different learned policy in MC compared to the one learned with PI/VI approaches (excluding the terminal state of holes)? Analyze the reason for that. (2 points)

2. For task II, question 1, if fixing $\epsilon = 5\%$ for all training episodes and keeping the total number of training episodes unchanged, how is the result of the learned policy compared to the decaying ϵ-greedy exploration strategy given the same number of training episodes? Why is that? (1.5 points)

3. Repeat the task II, subtask 3 only with a different step size setting *'average sample'* ($\alpha = 1/N_{visit}$), where the other settings remain the same. **For this task, you don't need to show the plot. Nor is it necessary to wait until the program finishes.** Explain what you observe for the learned policy with the same number of training episodes of 2000? Theoretically, can the setting of $\alpha = 1/N_{visit}$ reaches convergence to a near optimal policy and why? Does the time to converge to near-optimal action value $Q_{\pi^\star}(s, a)$ depends on the initial Q-value? (2 points)

4. What is the purpose of optimistic initial values? If one wants to perform optimistic initial values on the "MountainCar-v0" environment, which of the following initial values of each state-action pair are considered as valid? And also reason why. (Multiple answers could be correct) A. 0, B. 50 , C. -50 , D. -5 , E. 5 ? (2 points)

Literature

The following reference material is relevant for this assignment.

- **Reinforcement Learning - an introduction, second edition** Chapter 2.7, Chapter 5 (5.8, 5.9 optional), Chapter 6.1 - 6.5, 6.9, Chapter 2.6
  for further understanding but not necessary: Chapter 6.6 - 6.8