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: June.16th

Reinforcement Learning - Assignment III (21 pt + 6 pt)

The reference link for this assignment is illustrated in the Literature part below. Please fetch the python 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 multi-step TD algorithms and eligibility trace in both on-policy and off-policy cases, which is an extension to the previous one-step TD learning. For each plot, please mention the team member who ran the experiment. Best luck!

Task I: Theoretical understanding on multi-step TD learning and Eligibility Trace

1. (1) Explain the difference between one-step TD algorithm and multi-step TD algorithm in terms of the update rule of q-value, considering only the on-policy case. (Provide math formula) (2) Explain the *n-step return* and λ-*return* and their relations (Provide math formula). (2 points)

2. (i) Give formal definition on the the term *importance sampling ratio*(IS ratio) and why do we need that? (ii) How is *importance sampling ratio* related to the Q-value update in n-step SARSA and n-step off-policy learning algorithms (provide the mathematical formulae). (iii) What potential problems will be encountered when introducing IS ratio for update Q-values? (3.5 points)

3. Explain what is bias-variance tradeoff in RL problems, where you need to explain where *bias* and *variance* appears in the context of RL respectively. You could optionally support your answer with graphs. Then compare the bias-variance tradeoff between *one-step TD learning*, *10-step TD learning* and *Monte-Carlo approaches* and reason why is it so. (3 points)

4. State the relation between the choice of λ in λ-return with the target update in *on-policy Monte-Carlo control* and SARSA learning. What is the advantage of eligibility trace over one-step and multi-step TD learning? Does *eligibility-trace* result in a correct value estimate? How are *replacing trace* and *accumulating trace* defined? (Consider only the discrete state-action space) (2.5 points)

5. (i) Considering only the on-policy case, are *forward-view* and *backward-view* of SARSA(λ) or TD(λ) equivalent in off-line updating? Are they equivalent in online updating? (ii) Assuming the agent takes an action a_n in the current state s_n given the previous trajectory in this episode $\tau = \{s_0, a_0, r_1, s_1, a_1, r_2, ..., s_{n-1}, a_{n-1}, r_n, s_n\}$, **explain by providing math formulae** on how $q(s_0, a_0)$ is estimated after taking the action a_n for both forward-view and backward-view respectively. For forward-view, also show how $q(s_0, a_0)$ is estimated before taking the step a_n. Optionally, you could also illustrate with figures for clearer explanation. (iii) What property does *dutch eligibility trace* hold? (3.5 points)

6. Explain the terms *sample update* and *expected update*, *bootstrapping* and *non-bootstrapping*. Mention at least one algorithm for *sample-update* and *non-bootstrapping* method respectively and two algorithms for *expected-update* and *bootstrapping* method respectively. (2 points)

Task II: Programming part on Eligibility Trace

1. Implement the algorithm SARSA(λ) with *replacing trace* on the discretized 'MountainCar-v0' for $\lambda = 0.95$ and step size $\alpha = 0.05$ and $Q_{ini}(s, a) = 0\ \forall s, a$. In the book, the algorithm is illustrated in the case of function approximation for continuous state space. However, in this assignment, we are still in the discrete/tabular case. You can think of discretization as one way of function approximation (non-overlapped *tile coding* with binary features). The approximated q-value $\hat{q}(s, a, w) = w^T x(s, a)$, where $w \in \mathbb{R}^d$ is the weight to be learned and $x(s, a) \in \mathbb{R}^d$ is the encoded feature representation of (s, a) with dimension d.

Reinforcement Learning (SS2021)
Lecturer: Dr. Christoph Metzner
c.metzner@campus.tu-berlin.de
Teaching Assistant: Honghu Xue
xue@rob.uni-luebeck.de
Submission Deadline: June.16th

The idea is mostly the same as *linear regression* in the lecture of *Probabilistic Machine Learning*, where you first convert your raw input to some (non-linear) feature space, and then find the optimal w so that $\left\|q_\pi(s,a) - \hat{q}(s,a,w)\right\|^2$ is minimal. In our case, $x(s) = [0,0,0,..1,0,0..] \in \mathbb{R}^{21\times16}, x(s,a) \in \mathbb{R}^{21\times16\times3}$, where the value 1 only appears in the index of the bin into which the current state-action pair s, a falls, for other bins/features, the encoded representation is 0. The dimension d of the feature in our case is $21 \times 16 \times 3 = 1008$ (number of discretization bins). Each value in the weight vector $w \in \mathbb{R}^d$ corresponds directly to the q-value of that bins/features. You can also see that the converted feature takes the value of either 0 or 1 among the feature dimensions, which is so-called *binary features*.
Run the algorithm **3-4 times** (One run per team member). Plot all of your episodic rewards **in one plot** according to the plot in Latex template (in mean and variance scheme). **Only in the testing phase, constrain the maximal episodic length to be** 1000 **(i.e.** $T = 1000$**) to avoid endless loop.** The learned cumulative episodic reward should be around -150. (The code is similar to Assignment II.) (1.5 points)

2. In order to examine the effect of choice of λ on the convergence speed as well as the advantage of eligibility-trace over SARSA($\lambda = 0$), the initial Q-values for all state-action pairs are set to be 10000. Here, we change the total number of training episodes to be 5000 while keeping other conditions the same. After training, check the learned q-value of SARSA($\lambda = 0.5$), SARSA($\lambda = 0.75$), SARSA($\lambda = 0.99$) and analyze how is the effect of λ on the convergence speed. (**Hint**: compare the learned q-value with the $Q_{\pi^*}(s,a)$, where π^* is the optimal policy, assuming the agent takes 150 steps from initial state to the goal under π^*.) For this task, you don't need to show the plot of training curve, the program needs to be run until the end for analysis. 5000 episodes takes roughly 25 mins to finish (time differs on the choice of λ). **Distribute one setting to each of the team member to reduce time.** Better to save the q-table as '.npy' file so that analysis can be postponed. (3 points)

Bonus Tasks

1. Provide a categorization of the following algorithms: expected SARSA, n-step Tree Backup algorithm, Watkin's Q(λ), Q(σ) on sample update/expected update, bootstrapping/non-bootstrapping, on-policy/off-policy, requires a model/model-free. (2 points)

2. Implement the off-policy Watkin's Q(λ) algorithm (discrete case) with the same setting as **Task II, subtask 1**, whereas $\lambda = 0.95$. Use the *replacing trace*. Show the plot of episodic rewards w.r.t. training episodes (mean and variance scheme) of episodic rewards w.r.t. training episodes on 3-4 runs (one run per team member). You can modify your code from the above task. (2 points)

3. Implement the True-online SARSA(λ) algorithm (discrete case) with the same setting as **Task II, subtask 1**, whereas $\lambda = 0.95$. Show the plot of episodic rewards w.r.t. training episodes (mean and variance scheme) on 3-4 runs (one run per team member). You can modify your code from the above task. **Also write down the update rule in tabular case for updating Q-values.** (refer to Chapter 12.7 of the book) (2 points)

Literature

The following reference material is relevant for this assignment.

- **Reinforcement Learning - an introduction, second edition** Chapter 7 (7.5-7.6 for bonus task), Chapter 8.5 (Important: Figure 8.6), Chapter 12.1, 12.2, 12.4, 12.6, 12.7, 12.13 (12.5, 12.10 for bonus task), Chapter 6.6 (for bonus task), Chapter 9.5.3, 9.5.4 (Coarse coding and Tile coding, optional)

Reinforcement Learning (SS2021)
Lecturer: Dr. Christoph Metzner
c.metzner@campus.tu-berlin.de
Teaching Assistant: Honghu Xue
xue@rob.uni-luebeck.de
Submission Deadline: June.16th

- Watkin's Q(λ) (only for bonus task):
`http://www.karanmg.net/Computers/reinforcementLearning/finalProject/KaranComparisonOfSarsaWatkins.pdf` , Figure 4