UNIVERSITÄT ZU LÜBECK
INSTITUTE FOR ROBOTICS
AND COGNITIVE SYSTEMS

Reinforcement Learning (SS2020)
Honghu Xue
xue@rob.uni-luebeck.de
Handout Date: 25. June 2020
Submission Deadline: 16. July 2020

Reinforcement Learning - Assignment V (25 pt + 5 pt)

The relevant reference link for this assignment is illustrated in the Literature part below. Please fetch the python files for the programming part.
The goal of this assignment is to understand a completely new but also widely-used RL algorithms, i.e. *Policy-Gradient* (PG) methods. PG methods differ from previously learned value-based methods in that PG approaches directly maps the state to a probability on the available actions via function approximators (i.e. neural networks) without necessarily performing V or Q-value estimation. The policy in value-based methods can only be retrieved from the estimated V or Q-value. In this assignment, you will also learn REINFORCE and *Actor-Critic* algorithms. The programming part covers the implementation of these algorithms in both discrete action space and continuous action space (state space continuous). Please first refer to the Literature and then proceed with the task. Good Luck!

Task I: Theoretical understanding on Policy-Gradient methods

1. Mention at least two advantages of policy-gradient approaches over value-based methods and the drawback of policy-gradient approach. (1 point)
2. (1) What do PG methods try to maximize, write down the formal math definition of that term. (2) Then show the update rule for policy parameters θ mathematically for REINFORCE. Here, you also need to show how the gradient $\nabla J(\theta)$ is defined and related to update rule, where $J(\theta)$ refers to the term in the first sub-question. (3) Finally show what is the trick of likelihood ratio (in book called *eligibility vector*) in deriving REINFORCE algorithm? Show the final update rule for the policy parameters θ after applying likelihood-ratio trick. (3 points)
3. What is the advantage of REINFORCE with baseline over REINFORCE? Why can baseline mitigate that problem? What kind of baselines are considered valid? And show one valid baseline function. (3 points)
4. Explain the term *Causality* in detail as another important technique to reduce the variance for REINFORCE. Support your answer with formulae. State REINFORCE and A2C are on-policy or off-policy algorithms. (2 points, Sergey's lecture)
5. In the case of *actor-critic* methods, how are the things changed compared to REINFORCE? Show the mathematical expression for the update rule of Q *actor-critic.* (refer to book of RL) (2 points)
6. In continuous action space, one way of policy characterization is using Gaussian distribution. First write down how is $\pi(a|s,\theta)$ defined mathematically with θ being the policy parameters and explain the terms appearing in your mathematical expression. How are the mean and variance defined using θ in practice? Show the complete derivation of *exercise 13.4* on page 336 from *Reinforcement Learning - an introduction, second edition.* (2.5 points, book of RL)

Task II: Programming part on Policy-Gradient methods

In this part, you will implement REINFORCE with causality reduction and one variance reduction scheme on top of REINFORCE algorithm, resulting in one new algorithm *Advantage Actor-Critic* (A2C). In this assignment, you'll implement n-step advantage estimate A2C variants for both discrete action space and continuous action space, where we use N-step return as the target value for state estimate. The state space are all continuous.

1. Show the mathematical definition of *advantage.* (1.5 points , Sergey's lecture)
2. Implement the algorithm REINFORCE with causality reduction on 'CartPoleBulletEnv-v1' in 'REINFORCE-single-thread-discrete-new.py'. 'CartPoleBulletEnv-v1' is a discrete action-space environment on pybullet. For discrete action space, one way of policy parameterization is soft-max in action preference mentioned in the book *Reinforcement Learning, an introduction.* **Run the algorithm once per group member, show the scatter plot in subplot form as Assignment IV)**. The finally training reward should reach 200 and the training endures roughly 5 minutes. (Hint: For this task, your local computer supporting Pytorch GPU is powerful enough to run locally. Please check the shape of each Tensor variable.) (5 points)

Reinforcement Learning (SS2020)
Honghu Xue
xue@rob.uni-luebeck.de
Handout Date: 25. June 2020
Submission Deadline: 16. July 2020

3. Implement the algorithm A2C on 'InvertedPendulumBulletEnv-v0' in 'A2C-single-thread-continuous-new.py', a Pybullet environment with continuous action space. For that you will use Gaussian parameterization mentioned in Task I, Q6. The typical A2C algorithm will enable multi-agent settings, where N agents collects the experience in parallel to save time and return all the collected experience to update actor and critic. In our program, we collect N complete episodes for sequentially using only one agent for simplicity, but with the same exact same functionality as in multi-agent scenario. **Run the algorithm once per group, show the scatter plot**. The final training reward should converge to 1000 and the training endures roughly 15 minutes on Colab. (For those who are interested, you could also run the environement on 'LunarLanderContinuous-v2' or '-v3', a variant of 'LunarLander-v2' in continuous action space, the performance gradually improves, but it could take 3 hours to converge to a reasonable result.) (5 points)

Bonus Tasks

1. Give the formal definition of GAE. Support the answer with the formulae. (1 point)

2. Compare the plot of episodic rewards after convergence to 1000 and the plot of value loss to non-bootstrapping mode for Task II, Q2, what do you observe? Then analyze why the value loss finally converges to mostly 0 when bootstrapping mode is enabled for A2C. What will happen if the discount factor γ is set to one? (Hint: you might need the knowledge on animation of the trained policy to answer this question.) (2 points)

3. This task is more exploratory and open. One disadvantage of the current environment 'Inverted Pendulum' is the forced termination after 1000 steps, which could cause potentially wrong value estimate or oscillation in value estimate in *critic* (i.e. predecessor states prior to terminal state and other states similar to this predecessor state traversed in this iteration). Therefore, as a potential trial, we examine the effect of expanding the state dimension with one additional dimension, namely the time dimension, where its value starts from 0 initially and increments uniformly by $\frac{1}{L}$, where L denotes the maximal episodic length of the environment. With such state design, it satisfies the MDP property and policy becomes non-stationary and 'Bootstrapping mode' in the program can be replaced by this variant. You can test this setting by running A2C. Show the plots of both value losses from the critic and episodic rewards for the algorithms respectively and analyze whether incorporating time in state dimension reduces the value loss and stabilize the performance of returns after convergence. (2 points)

Literature

The following reference material is relevant for this assignment. Each reference have some **overlapping content** with the other, so you don't need to go through all of the reference material below, just choose wisely.

- Sergey Levine's online *Deep Reinforcement Learning* lectures (1)'Policy Gradients' `https://www.youtube.com/watch?v=XGmd3wcyDg8&list=PLkFD6_40KJIxJMR-j5A1mkxK26gh_qg37&index=22&t=0s` (2)'Actor-Critic Algorithms' `https://www.youtube.com/watch?v=Tol_jw5hWnI&list=PLkFD6_40KJIxJMR-j5A1mkxK26gh_qg37&index=20` (highly-recommended)
- **Reinforcement Learning - an introduction, second edition** Chapter 13 (13.6 optional)
- David Silver's online *Reinforcement Learning* lecture 7 (optional if you read the book of RL and watch Sergey's lecture)