IOC5259 Spring 2022: Reinforcement Learning **(Due: 2022/04/14, Thursday, 21:00)**

Homework 1–Part II: Policy Gradient and Model-Free Prediction

Submission Guidelines: Your deliverables shall consist of 2 separate files – (i) A PDF file: Please compile all your write-ups into one .pdf file (photos/scanned copies are acceptable; please make sure that the electronic files are of good quality and reader-friendly); (ii) A zip file: Please compress all your source code into one .zip file. Please submit your deliverables via E3.

Problem 1 (Baseline for Variance Reduction) (8+8=16 points)

Consider an example similar to that in the slides of Lecture 8 for explaining the baseline. Suppose there are only 1 non-terminal starting state (denoted by s) and 3 actions (denoted by a, b, c) in the MDP of interest. After any one of the action is applied at the starting state s, the MDP would evolve from s to the terminal state, with probability 1. Moreover, consider the following setting:

- The rewards are deterministic, and the reward function is defined as $r(s,a) = 100$, $r(s,b) = 98$, and $r(s,c) = 95$. Moreover, there is no terminal reward.
- We consider a softmax policy with parameters $\theta_a, \theta_b, \theta_c$ such that $\pi_\theta(\cdot|s) = \exp(\theta_\cdot)/(\exp(\theta_a) + \exp(\theta_b) + \exp(\theta_c))$. Moreover, currently the parameters are $\theta_a = 0$, $\theta_b = \ln 5$, $\theta_c = \ln 4$.
- We would like to combine PG with SGD. At each policy update, we would construct an unbiased estimate $\widehat{\nabla} V$ of the true policy gradient $\nabla_\theta V^{\pi_\theta}$ by sampling one trajectory (Note: $\widehat{\nabla} V$ is a random vector. In this example, each trajectory has only one time step, and $s_0 = s$, a_0 is either a, b, or c, and s_1 is the terminal state).

(a) What are the mean vector of $\widehat{\nabla} V$ (denoted by $\mathbb{E}[\widehat{\nabla} V]$) and the covariance matrix of $\widehat{\nabla} V$ (i.e., $\mathbb{E}[(\widehat{\nabla} V - \mathbb{E}[\widehat{\nabla} V])(\widehat{\nabla} V - \mathbb{E}[\widehat{\nabla} V])^\intercal]$)?

(b) Suppose we leverage the value function $V^{\pi_\theta}(s)$ as the baseline and denote by $\tilde{\nabla} V$ the corresponding estimated policy gradient. Then, what are the mean vector and the covariance matrix of $\tilde{\nabla} V$? (Note: $\tilde{\nabla} V$ is also a random vector)

Problem 2 (Policy Gradient) (8 points)

Show the following useful property discussed in Lecture 6: for any function $f : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$,

$$\mathbb{E}_{\tau \sim P_\mu^{\pi_\theta}} \left[\sum_{t=0}^{\infty} \gamma^t f(s_t, a_t) \right] = \frac{1}{1-\gamma} \mathbb{E}_{s \sim d_\mu^{\pi_\theta}} \mathbb{E}_{a \sim \pi_\theta(\cdot|s)} \big[f(s,a) \big] \tag{1}$$

(Hint: It might be slightly easier to go from the RHS to LHS. Specifically, you may first expand the RHS of (1) into a sum of $f(s,a)$ over s and a and then apply the definition of $d_\mu^{\pi_\theta}$, which involves a sum of probability over t. Next, try to reorganize the triple summation into the form of the LHS of (1))

Problem 3 (Stochastic Approximation) (10 points)

In this problem, let us rigorously prove the convergence of stochastic approximation (SA) update for the example discussed in Lecture 11:

- Let $F : \mathbb{R} \to \mathbb{R}$ be a one-dimensional real-valued function.
- Suppose that $F(\theta)$ is L-smooth and that there exists $\delta > 0$ such that $F'(\theta) \geq \delta$, for all $\theta \in \mathbb{R}$.

- Our goal is to find the root of $F(\theta)$ (i.e., the θ that satisfies $F(\theta) = 0$) through noisy measurements $Y(\theta; \varepsilon) = F(\theta) + \varepsilon$, where ε is an additive noise with $\mathbb{E}[\varepsilon] = 0$.
- The stochastic approximation update proceeds iteratively. Let θ_k denote the parameter at the end of the k-th iteration. In each iteration $k+1$, the SA update is designed as

$$\theta_{k+1} = \theta_k - \alpha_k Y(\theta_k; \varepsilon_k), \tag{2}$$

  where α_k is the learning rate and $\{\varepsilon_k\}$ form a sequence of i.i.d. zero-mean noises with finite variance σ^2.

Please show that if $\sum_{k=0}^{\infty} \alpha_k = \infty$ and $\sum_{k=0}^{\infty} {\alpha_k}^2 < \infty$, then we have $F(\theta_k) \to 0$, as $k \to \infty$.

Problem 4 (Policy Gradient Algorithms With Function Approximation) (14+12=26 points)

In this problem, we will implement two policy gradient algorithms with the help of neural function approximators: (1) Vanilla REINFORCE and (2) REINFORCE with value function as the baseline For the pseudo code of the algorithms, please see the slides of Lecture 8 and Lecture 9. You may write your code in either PyTorch or TensorFlow (though the sample code presumes PyTorch framework). If you are a beginner in learning the deep learning framework, please refer to the following tutorials:

- PyTorch: https://pytorch.org/tutorials/
- Tensorflow: https://www.tensorflow.org/tutorials

For the deliverables, please submit the following:

- Technical report: Please summarize all your experimental results in 1 single report (and please be brief)
- All your source code
- Your well-trained models (REINFORCE with and without a baseline) saved in either .pth files or .ckpt files

(a) We start by solving the simple "CartPole-v0" problem (https://gym.openai.com/envs/CartPole-v0/) using the vanilla REINFORCE algorithm. Read through **reinforce.py** and then implement the member functions of the class **Policy** and the function **train**. Please briefly summarize your results in the report and document all the hyperparameters (e.g. learning rates and NN architecture) of your experiments.

(b) Based on the code for (a), implement the REINFORCE algorithm with a baseline and solve the "LunarLander-v2" problem, which has slightly higher state and action space dimensionality (https://gym.openai.com/envs/LunarLander-v2/). Note that you are allowed to **design a baseline function on your own** (e.g., the baseline can either be a handcrafted state-dependent function, the value function, or some function learned directly from the trajectories). Please clearly explain your choice of the baseline function in the report. Save your code in another file named **reinforce_baseline.py**. Please add comments to your code whenever needed for better readability. Again, please briefly summarize your results in the report and document all the hyperparameters of your experiments. (Note: As LunarLander is a more challenging environment than CartPole, it might require some efforts to tune the hyperparameters, e.g., learning rates or learning rate scheduler. You could either do grid search or even use some more advanced techniques such as the evolutionary methods. As the main purpose of this homework is to help you get familiar with RL implementation, there is NO hard requirement on the obtained returns of this LunarLander task. Just try your best and enjoy!)