535514 Spring 2024: Reinforcement Learning **(Due: 2024/04/19, Friday, 21:00)**

Homework 2: 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+8=24 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)

(c) Let $B(s)$ denote a baseline function and ∇V_B be the corresponding estimated policy gradient (∇V_B is again a random vector). Suppose we say that a baseline function $B(s)$ is *optimal* if it attains the minimum trace of the corresponding covariance matrix of ∇V_B among all possible state-dependent baselines. Please try to find one such optimal $B(s)$.

Problem 2 (Non-Uniform Polyak-Lojacsiewicz Condition in RL) (8+8=16 points)

As described in Lecture 12, let us prove the fundamental Polyak-Lojacsiewicz condition in RL: Let π^* be an optimal policy and let $a^* := \arg\max_a \pi^*(a|s)$ (essentially, a^* is an optimal action). Under softmax policies, we have

$$\left\| \frac{\partial V^{\pi_\theta}(\mu)}{\partial \theta} \right\|_2 \geq \frac{1}{\sqrt{S}} \left\| \frac{d_\mu^{\pi^*}}{d_\mu^{\pi_\theta}} \right\|_\infty^{-1} \cdot \min_{s \in S} \pi_\theta(a^*(s)|s) \cdot [V^*(\mu) - V^{\pi_\theta}(\mu)]. \tag{1}$$

To show this, you would also need the celebrated "Performance difference lemma" as follows: For any two policies π_1 and π_2, we always have

$$V^{\pi_2}(\mu) - V^{\pi_1}(\mu) = \frac{1}{1-\gamma} \mathbb{E}_{s' \sim d_\mu^{\pi_2}} \mathbb{E}_{a' \sim \pi_2(\cdot|s')} \left[A^{\pi_1}(s', a') \right]. \tag{2}$$

(a) To begin with, show the following result:

$$\left\| \frac{\partial V^{\pi_\theta}(\mu)}{\partial \theta} \right\|_2 \geq \frac{1}{1-\gamma} \cdot \frac{1}{\sqrt{S}} \sum_s d_\mu^{\pi_\theta}(s) \cdot \pi_\theta(a^*(s)|s) \cdot \left| A^{\pi_\theta}(s, a^*(s)) \right|. \tag{3}$$

(Hint: You would need to first apply Cauchy-Schwarz inequality and leverage the Policy Gradient expressio under softmax policies. This subproblem shall require about 5-8 lines of proof.)

(b) Next, please use the results in (a) and the Performance difference lemma to conclude that the PL condition in (1) indeed holds. (Hint: Try to handle each term in (3) separately. This subproblem shall require only about 5-8 lines of proof.)

Problem 3 (Monte Carlo Policy Evaluation) (8+8=16 points)

As discussed in Lecture 10, we know that the every-visit Monte Carlo estimate is biased. Let us quickly verify this fact under the simple 2-state MRP (which serves as a reduction from any MRP), as shown below. Specifically, we need to check the following two properties:

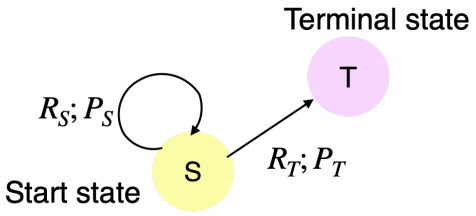

Figure 1: A simple 2-state MRP.

- Property 1: Show that the true value function at state S (denoted by $V(S)$) satisfies that

$$V(S) = \frac{P_S}{P_T} R_S + R_T. \tag{4}$$

- Property 2: Suppose we construct an every-visit MC estimate based on only 1 trajectory τ (denoted by $\hat{V}_{\text{MC}}(S;\tau)$). Then, please show that

$$\mathbb{E}_\tau \left[\hat{V}_{\text{MC}}(S;\tau) \right] = \frac{P_S}{2P_T} R_S + R_T. \tag{5}$$

  (Hint: To begin with, you shall consider all possible trajectories and the corresponding probabilities. Accordingly, you would obtain that $\mathbb{E}_\tau \left[\hat{V}_{\text{MC}}(S;\tau) \right] = \sum_{k=0}^{\infty} P_T P_S^k \left(\frac{R_S + 2R_S + \cdots + kR_S + (k+1)R_T}{k+1} \right)$.)

Problem 4 (Policy Gradient Algorithms With Function Approximation) (20+10+20=50 points)

In this problem, we will implement three policy gradient algorithms with the help of neural function approximators: (1) Vanilla REINFORCE, (2) REINFORCE with value function as the baseline, and (3) REINFORCE with Generalized Advantage Estimation (GAE). For the pseudo code of the algorithms, please see the slides of Lectures 8-11. You may write your code in either PyTorch or TensorFlow (though the sample code presumes PyTorch framework). Moreover, you are recommended to use either Tensorboard or Weight and Biases to keep track of the loss terms and other related quantities of your implementation. If you are a beginner in learning the deep learning framework, please refer to the following tutorials:

- PyTorch: https://pytorch.org/tutorials/
- Tensorboard: https://pytorch.org/tutorials/intermediate/tensorboard_tutorial.html

- Tensorflow: https://www.tensorflow.org/tutorials
- Tensorboard: https://www.tensorflow.org/tensorboard/get_started
- Weight and Biases: https://docs.wandb.ai/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 a baseline, without a baseline, and with GAE) 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 (including the snapshot of the Tensorboard record) 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 (including the snapshot of the Tensorboard record) and document all the hyperparameters of your experiments. (Note: As LunarLander is a slightly 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!)

(c) Based on the code for (a), implement the REINFORCE algorithm with the Generalized Advantage Estimation (GAE) as your value prediction. Please run the code on "LunarLander-v2". Note that there is a hyperparameter λ in the GAE algorithm, **please try three different values of** λ and summarize your results of these three choices clearly in the report. For more details about GAE, please refer to the slides of Lectures 9-10 and the original paper (https://arxiv.org/abs/1506.02438). Please briefly explain your implementation of GAE in your report. Also, please save your code in another file named **reinforce_gae.py**. Last but not least, please add comments to your code whenever needed for better readability.

Remark: Regarding the snapshot of the Tensorboard and Weight and Biases records, please feel free to record any value that you aim to know. Here is an example:

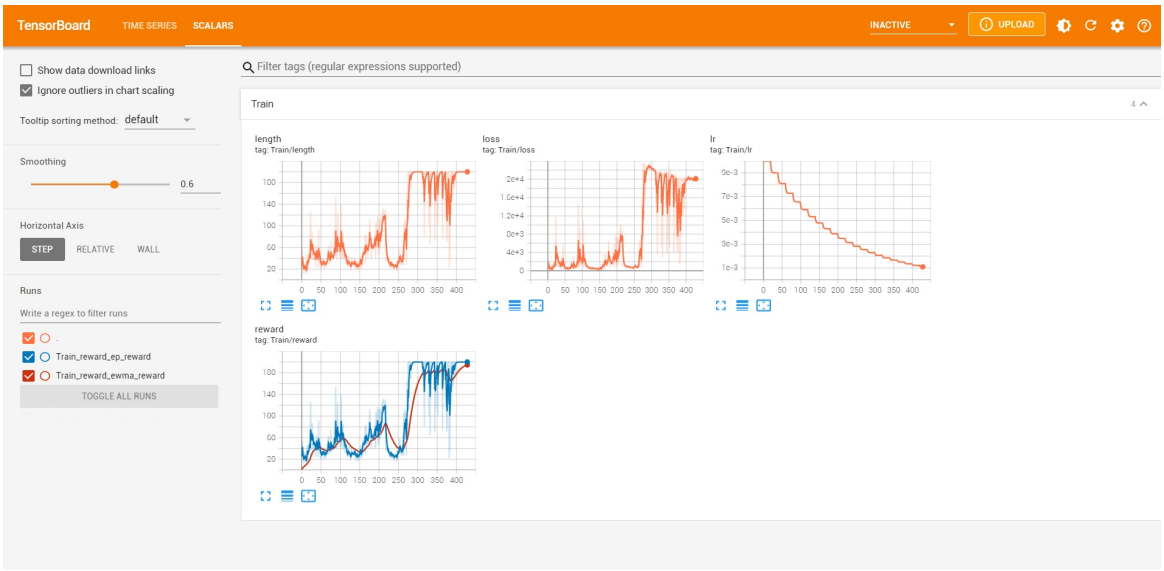

Figure 2: An example of the Tensorboard record.

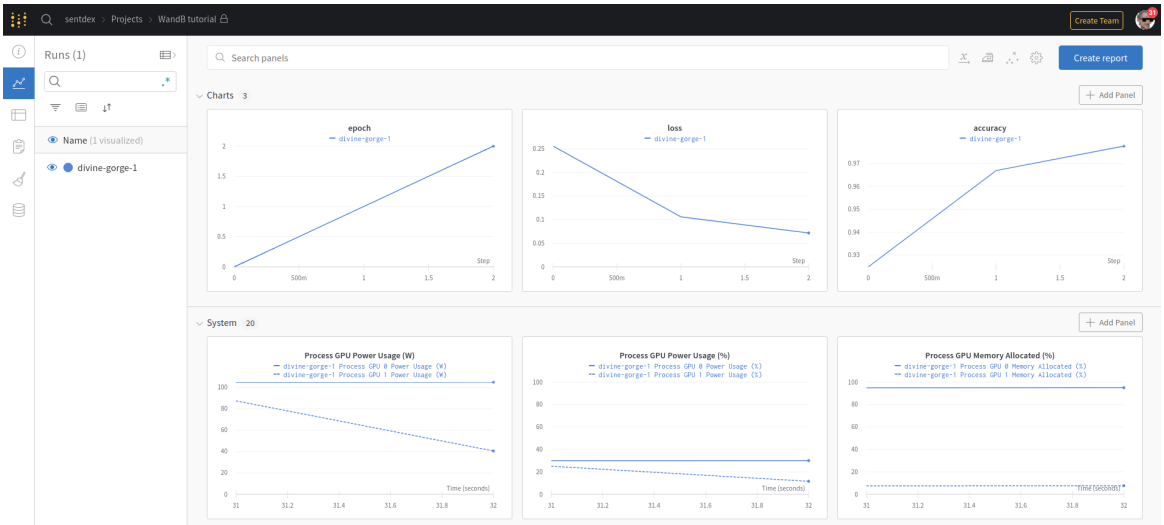

Figure 3: An example of the Weight and Biases record.