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:

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

(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 (2) 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:

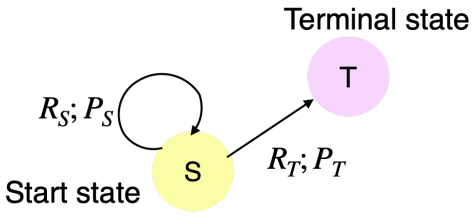


Figure 1: A simple 2-state MRP.

- 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:

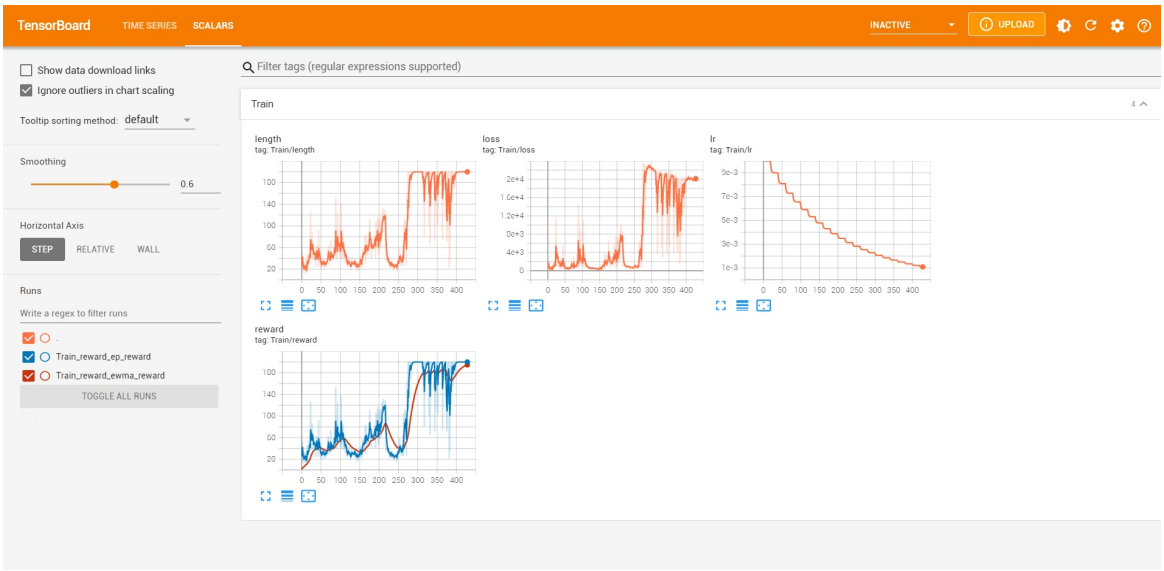


Figure 2: An example of the Tensorboard record.

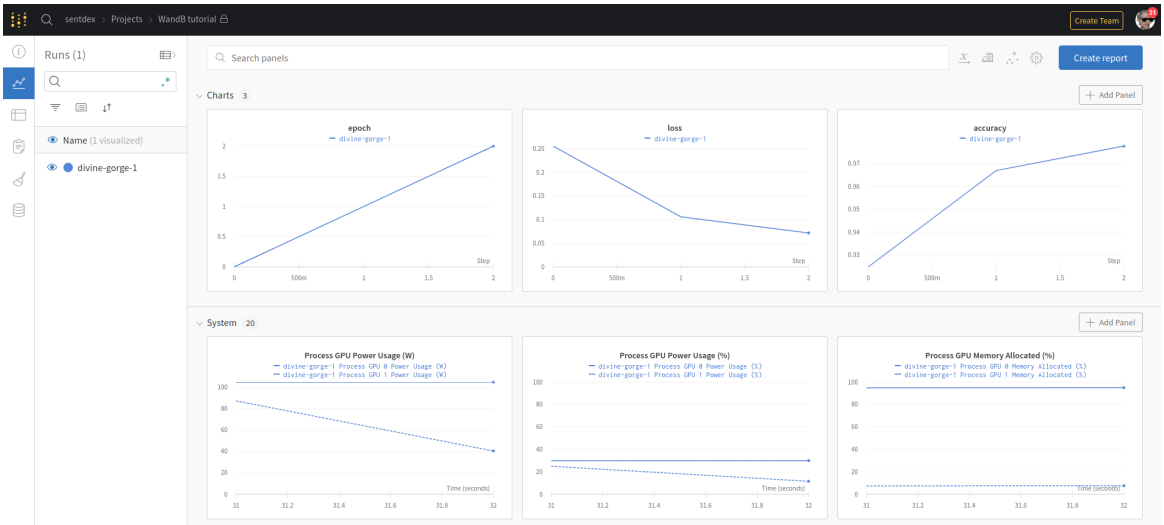
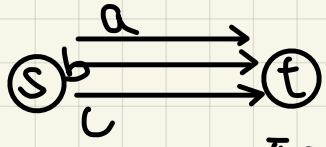


Figure 3: An example of the Weight and Biases record.

Problem 1.



$Q^{\pi_\theta}(s,a)=100,\ Q^{\pi_\theta}(s,b)=98,\ Q^{\pi_\theta}(s,c)=95$

$$\pi_\theta(a|s)=\frac{e^0}{e^0+e^{\ln 5}+e^{\ln 4}}=\frac{1}{10}=0.1$$

$$\pi_\theta(b|s)=\frac{e^{\ln 5}}{e^0+e^{\ln 5}+e^{\ln 4}}=\frac{5}{10}=0.5$$

$$\pi_\theta(b|s)=\frac{e^{\ln 4}}{e^0+e^{\ln 5}+e^{\ln 4}}=\frac{4}{10}=0.4$$

(a) mean

$$\log\pi(a|s)=\log\frac{e^{\theta(s,a)}}{e^{\theta(s,a)}+e^{\theta(s,b)}+e^{\theta(s,c)}}$$

$$=\theta(s,a)-\log\left(e^{\theta(s,a)}+e^{\theta(s,b)}+e^{\theta(s,c)}\right)$$

$$\nabla_\theta\log\pi_\theta(a|s)=\begin{bmatrix}\frac{\partial\log\pi_\theta(a|s)}{\partial\theta(s,a)}\\ \frac{\partial\log\pi_\theta(a|s)}{\partial\theta(s,b)}\\ \frac{\partial\log\pi_\theta(a|s)}{\partial\theta(s,c)}\end{bmatrix}=\begin{pmatrix}1-\pi_\theta(a|s)\\ -\pi_\theta(b|s)\\ -\pi_\theta(c|s)\end{pmatrix}=\begin{bmatrix}0.9\\ -0.5\\ -0.4\end{bmatrix}$$

$$\nabla_\theta\log\pi_\theta(b|s)=\begin{bmatrix}\frac{\partial\log\pi_\theta(b|s)}{\partial\theta(s,a)}\\ \frac{\partial\log\pi_\theta(b|s)}{\partial\theta(s,b)}\\ \frac{\partial\log\pi_\theta(b|s)}{\partial\theta(s,c)}\end{bmatrix}=\begin{pmatrix}-\pi_\theta(a|s)\\ 1-\pi_\theta(b|s)\\ -\pi_\theta(c|s)\end{pmatrix}=\begin{bmatrix}-0.1\\ 0.5\\ -0.4\end{bmatrix}$$

$$\nabla_\theta\log\pi_\theta(c|s)=\begin{bmatrix}\frac{\partial\log\pi_\theta(c|s)}{\partial\theta(s,a)}\\ \frac{\partial\log\pi_\theta(c|s)}{\partial\theta(s,b)}\\ \frac{\partial\log\pi_\theta(c|s)}{\partial\theta(s,c)}\end{bmatrix}=\begin{pmatrix}-\pi_\theta(a|s)\\ -\pi_\theta(b|s)\\ 1-\pi_\theta(c|s)\end{pmatrix}=\begin{bmatrix}-0.1\\ -0.5\\ 0.6\end{bmatrix}$$

$\because \theta(s,a)=\theta(s,b)=\theta(s,c)=0 \Rightarrow \pi_\theta(a|s)=\pi(b|s)=\pi(c|s)=\frac{1}{3}$

$$E[\hat{\nabla}V] = E_{\tau\sim p_{\pi_\theta}}\left[\sum_{t=0}^{2} \gamma^t Q^{\pi_\theta}(s_t,a_t)\nabla_\theta \log \pi_\theta(a_t|s_t)\right] \text{ by (P2)}$$

$$= 100\times\begin{bmatrix}0.9\\-0.5\\-0.4\end{bmatrix}\times 0.1 + 98\begin{bmatrix}-0.1\\0.5\\-0.4\end{bmatrix}\times 0.5 + 95\times\begin{bmatrix}-0.1\\-0.5\\0.6\end{bmatrix}\times 0.4$$

$$= \begin{bmatrix}0.3\\0.5\\-0.8\end{bmatrix} \square$$

(a) ② covariance matrix

$$Var[\hat{\nabla}V] = E[(\hat{\nabla}V - E[\hat{\nabla}V])(\hat{\nabla}V - E[\hat{\nabla}V])^T]$$

$$= \left(100\times\begin{bmatrix}0.9\\-0.5\\-0.4\end{bmatrix} - \begin{bmatrix}0.3\\0.5\\-0.8\end{bmatrix}\right)\times\left(100\times\begin{bmatrix}0.9\\-0.5\\-0.4\end{bmatrix} - \begin{bmatrix}0.3\\0.5\\-0.8\end{bmatrix}\right)^T\times 0.1$$

$$+\left(98\times\begin{bmatrix}-0.1\\0.5\\-0.4\end{bmatrix} - \begin{bmatrix}0.3\\0.5\\-0.8\end{bmatrix}\right)\times\left(98\times\begin{bmatrix}-0.1\\0.5\\-0.4\end{bmatrix} - \begin{bmatrix}0.3\\0.5\\-0.8\end{bmatrix}\right)^T\times 0.5$$

$$+\left(95\times\begin{bmatrix}-0.1\\-0.5\\0.6\end{bmatrix} - \begin{bmatrix}0.3\\0.5\\-0.8\end{bmatrix}\right)\times\left(95\times\begin{bmatrix}-0.1\\-0.5\\0.6\end{bmatrix} - \begin{bmatrix}0.3\\0.5\\-0.8\end{bmatrix}\right)^T\times 0.4$$

$$= \begin{bmatrix}894.03 & -509.75 & -384.28\\ -509.75 & 2352.75 & -1843\\ -384.28 & -1843 & 2227.28\end{bmatrix} \square$$

(b) ① mean

$$B(s) = \sum_a \pi(a|s)Q(s,a) = 0.1\times 100 + 0.5\times 98 + 0.4\times 95 = 97$$

$$V^{\pi_\theta}(s) = E_{\tau\sim p_{\pi_\theta}}\left[\sum_{t=0}^{2}\gamma^t\left(Q^{\pi_\theta}(s_t,a_t) - B(s_t)\right)\nabla_\theta\log\pi_\theta(a_t|s_t)\right] \text{ by (P2)}$$

$$= 0.1\times(100-97)\times\begin{bmatrix}0.9\\-0.5\\-0.4\end{bmatrix} + 0.5\times(98-97)\times\begin{bmatrix}-0.1\\0.5\\-0.4\end{bmatrix}$$

$$+0.4(95-97)\begin{bmatrix}-0.1\\-0.5\\0.6\end{bmatrix}$$

$$= \begin{bmatrix}0.3\\0.5\\-0.8\end{bmatrix} (= E[\hat{\nabla}V]) \square$$

(b) ② covariance matrix

$$Var[V^{T=3}_{(5)}] = (3\times\begin{bmatrix}0.9\\-0.5\\-0.4\end{bmatrix}-\begin{bmatrix}0.3\\0.5\\-0.8\end{bmatrix})\times(3\times\begin{bmatrix}0.9\\-0.5\\-0.4\end{bmatrix}-\begin{bmatrix}0.3\\0.5\\-0.8\end{bmatrix})^T\times 0.1$$
$$+(1\times\begin{bmatrix}-0.1\\0.5\\-0.4\end{bmatrix}-\begin{bmatrix}0.3\\0.5\\-0.8\end{bmatrix})\times(1\times\begin{bmatrix}-0.1\\0.5\\-0.4\end{bmatrix}-\begin{bmatrix}0.3\\0.5\\-0.8\end{bmatrix})^T\times 0.5$$
$$+(-2\times\begin{bmatrix}-0.1\\-0.5\\0.6\end{bmatrix}-\begin{bmatrix}0.3\\0.5\\-0.8\end{bmatrix})\times(-2\times\begin{bmatrix}-0.1\\-0.5\\0.6\end{bmatrix}-\begin{bmatrix}0.3\\0.5\\-0.8\end{bmatrix})^T\times 0.4$$
$$=\begin{bmatrix}0.66 & -0.5 & -0.16\\ -0.5 & 0.5 & 0\\ -0.16 & 0 & 0.16\end{bmatrix}\square$$

(c) 假設 $B(5)=b$、M: covariance matrix

$$a:\begin{bmatrix}(100-b)\times 0.9-0.3\\(100-b)\times(-0.5)-0.5\\(100-b)\times(-0.4)+0.8\end{bmatrix}\quad b:\begin{bmatrix}(98-b)\times(-0.1)-0.3\\(98-b)\times(0.5)-0.5\\(98-b)\times(-0.4)+0.8\end{bmatrix}$$

$$c:\begin{bmatrix}(95-b)\times(-0.1)-0.3\\(95-b)\times(-0.5)-0.5\\(95-b)\times 0.6+0.8\end{bmatrix}$$

trace of covariance matrix $f(b)$

$$=0.1[((100-b)\times 0.9-0.3)^2+((100-b)\times(-0.5)-0.5)^2+((100-b)\times(-0.4)+0.8)^2]$$
$$+0.5[((98-b)\times(-0.1)-0.3)^2+((98-b)\times(0.5)-0.5)^2+((98-b)\times(-0.4)+0.8)^2]$$
$$+0.4[((95-b)\times(-0.1)-0.3)^2+((95-b)\times(-0.5)-0.5)^2+((95-b)\times 0.6+0.8)^2]$$

$f'(b)=0 \Rightarrow b=97.138$

$\Rightarrow$ optimal $B(5)=97.138$ $\square$

(a) $\frac{\partial V^{\pi_\theta}(\mu)}{\partial \theta} = \begin{bmatrix} \frac{\partial V^{\pi_\theta}(\mu)}{\partial \theta(s_1,a_1)} & \cdots & \frac{\partial V^{\pi_\theta}(\mu)}{\partial \theta(s_1,a_{|A|})} \\ \vdots & & \\ \frac{\partial V^{\pi_\theta}(\mu)}{\partial \theta(s_{|S|},a_1)} & \cdots & \frac{\partial V^{\pi_\theta}(\mu)}{\partial \theta(s_{|S|},a_{|A|})} \end{bmatrix}$

$\Rightarrow \left\| \frac{\partial V^{\pi_\theta}(\mu)}{\partial \theta} \right\|_2 = \left[\sum_s \sum_a \left| \frac{\partial V^{\pi_\theta}(\mu)}{\partial \theta(s,a)} \right|^2 \right]^{1/2}$

$\geq \left[\sum_s \left| \frac{\partial V^{\pi_\theta}(\mu)}{\partial \theta(s,a^*(s))} \right|^2 \right]^{1/2}$

$= \left\{ \left[\sum_s \left| \frac{\partial V^{\pi_\theta}(\mu)}{\partial \theta(s,a^*(s))} \right|^2 \right] \left[\sum_s 1^2 \right] \right\}^{1/2} \frac{1}{\sqrt{|S|}}$

$\geq \left\{ \sum_s \left| \frac{\partial V^{\pi_\theta}(\mu)}{\partial \theta(s,a^*(s))} \right| \right\} \frac{1}{\sqrt{|S|}}$ by Cauchy-Schwarz inequality)

$= \frac{1}{\sqrt{|S|}} \sum_s \left| \frac{1}{1-\gamma} d_\mu^{\pi_\theta}(s) \pi_\theta(a^*(s)|s) A^{\pi_\theta}(s,a^*(s)) \right|$

(apply PG expression under softmax policies)

$= \frac{1}{1-\gamma} \frac{1}{\sqrt{|S|}} \sum_s d_\mu^{\pi_\theta} \cdot \pi_\theta(a^*(s)|s) \left| A^{\pi_\theta}(s,a^*(s)) \right|$ □

(b) By (2), $V^*(\mu) - V^{\pi_\theta}(\mu) = \frac{1}{1-\gamma} \sum_s \sum_a d_\mu^*(s) \pi^*(a|s) A^{\pi_\theta}(s,a)$

$\Rightarrow (V^*(\mu) - V^{\pi_\theta}(\mu)) \leq \frac{1}{1-\gamma} \sum_s \sum_a d_\mu^*(s) \pi(a^*(s)|s) \left| A^{\pi_\theta}(s,a^*(s)) \right|$

$= \sum_s \frac{d_\mu^*(s)}{d_\mu^{\pi_\theta}(s)} \left| A^{\pi_\theta}(s,a^*(s)) \right| d_\mu^{\pi_\theta}(s)$

$\Rightarrow \left[\min_{s \in S} \pi_\theta(a^*(s)|s) \right] (V^*(\mu) - V^{\pi_\theta}(\mu)) \leq \frac{1}{1-\gamma} \left(\max_s \frac{d_\mu^*(s)}{d_\mu^{\pi_\theta}(s)} \right) \times \left(\min_{s \in S} \pi_\theta(a^*(s)|s) \right) \sum_s \left| A^{\pi_\theta}(s,a^*(s)) \right| d_\mu^{\pi_\theta}(s)$

$\Rightarrow \frac{1}{\sqrt{|S|}} \left\| \frac{d_\mu^*}{d_\mu^{\pi_\theta}} \right\|_\infty^{-1} \min_{s \in S} \pi_\theta(a^*(s)|s) (V^*(\mu) - V^{\pi_\theta}(\mu))$

$\leq \frac{1}{\sqrt{|S|}} \frac{1}{1-\gamma} \sum_s d_\mu^{\pi_\theta}(s) \pi_\theta(a^*(s)|s) \left| A^{\pi_\theta}(s,a^*(s)) \right|$

$\leq \left\| \frac{\partial V^{\pi_\theta}(\mu)}{\partial \theta} \right\|_2$ by (3) □

Problem 3 Property 1.

pf $V(S) = E[G_t \mid S_t = s]$

$= P_T R_T + P_S P_T (R_S + R_T) + P_S^2 P_T (2R_S + R_T) + \cdots$

$S \to T$ $\quad S \to S \to T$ $\quad S \to S \to S \to T$

$= P_T R_T (1 + P_S + P_S^2 + \cdots) + P_T R_S \sum_{k=0}^{\infty} k P_S^k$

$= P_T R_T \frac{1}{1 - P_S} + P_T R_S \frac{P_S}{(1-P_S)^2}$

$1 - P_S = P_T$ $\quad P_T$

$= R_T + \frac{P_S}{P_T} R_S = \frac{R_S}{P_T} R_S + R_T$

$S := \sum_{k=0}^{\infty} k P_S^k$

$S = 0 \cdot P_S^0 + 1 \cdot P_S^1 + 2 P_S^2 + \cdots$

$-) \; P_S S = \quad 0 \cdot P_S^1 + 1 P_S^2 + \cdots$

$(1 - P_S) S = P_S + P_S^2 + \cdots$

$= \frac{P_S}{1 - P_S}$

$\Rightarrow S = \frac{P_S}{(1-P_S)^2}$

Property 2

$E_\tau[\hat{V}_{MC}(S, \tau)] = \sum_\tau P_\tau \hat{V}_{MC}(S, \tau)$

$S \to T, \; S \to S \to T \cdots$

$= P_T (R_T) + P_S P_T ((R_S + R_T) + R_T) \frac{1}{2}$

$+ P_S P_S P_T ((R_S + R_S + R_T) + (R_S + R_T) + R_T) \frac{1}{3} + \cdots$

$= \sum_{k=0}^{\infty} P_T P_S^k \left(\frac{k R_S + (k-1) R_S + \cdots + R_S + (k+1) R_T}{k+1} \right)$

$= \sum_{k=0}^{\infty} P_T P_S^k \frac{\frac{k(k+1)}{2} R_S + (k+1) R_T}{k+1}$

$= \frac{1}{2} P_T R_S \sum_{k=0}^{\infty} P_S^k k + P_T R_T \sum_{k=0}^{\infty} P_S^k$

$\frac{P_S''}{(1-P_S)^2} \cdot P_T$ $\quad \frac{1}{1-P_S}$

$= \frac{P_S}{2 P_T} R_S + R_T \quad \square$

$S := \sum_{k=0}^{\infty} k P_S^k = \sum_{k=1}^{\infty} k P_S^k$

$P_S S = \sum_{k=1}^{\infty} (k-1) P_S^k$

$-)$

$(1 - P_S) S = \sum_{k=1}^{\infty} P_S^k$

$= P_S \sum_{k=0}^{\infty} P_S^k$

$= P_S \frac{1}{1 - P_S}$

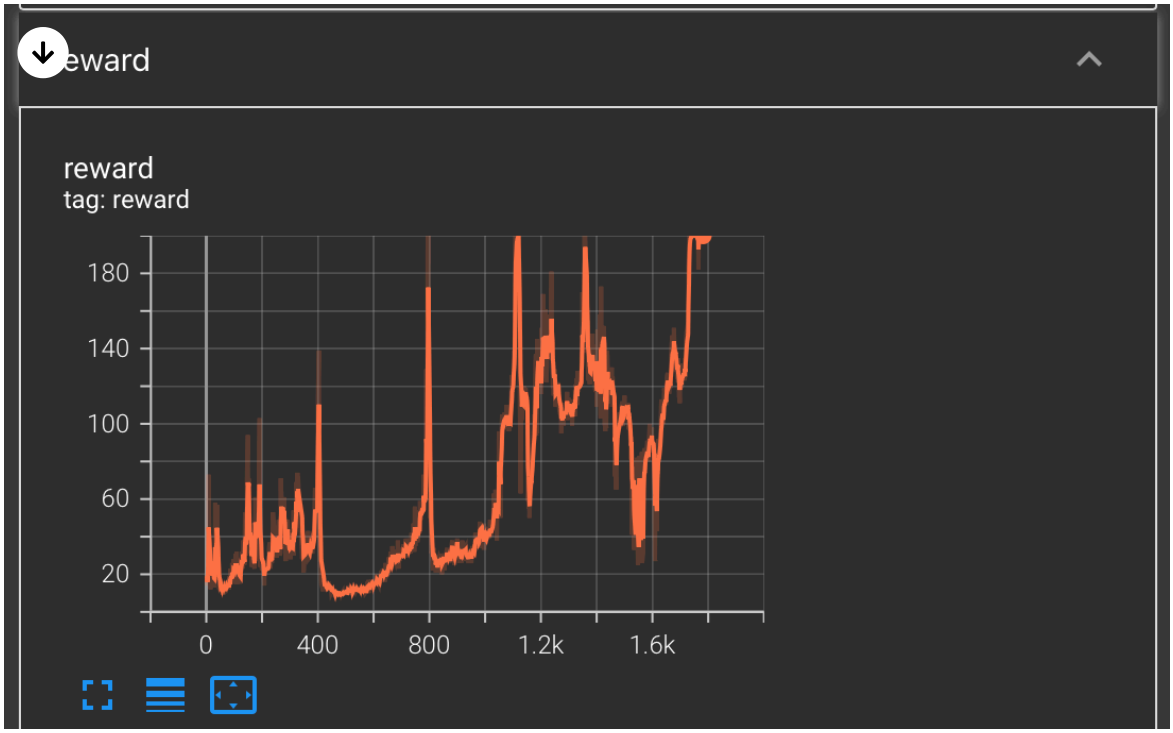
$\Rightarrow S = \frac{P_S}{(1-P_S)^2}$

tags: `2024 年 下學期讀書計畫` `Reinforcement Learning`

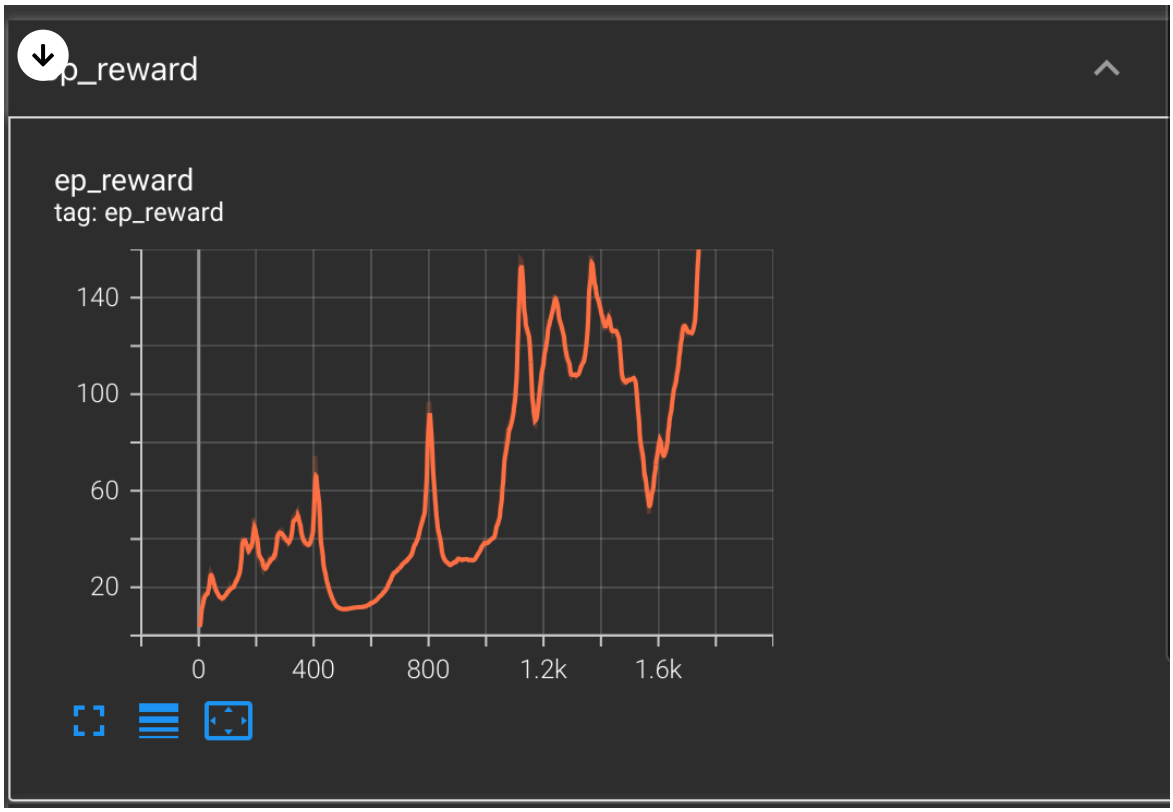
RL Homework 2: Policy Gradient and Model-Free Prediction Implements

a. reinforce

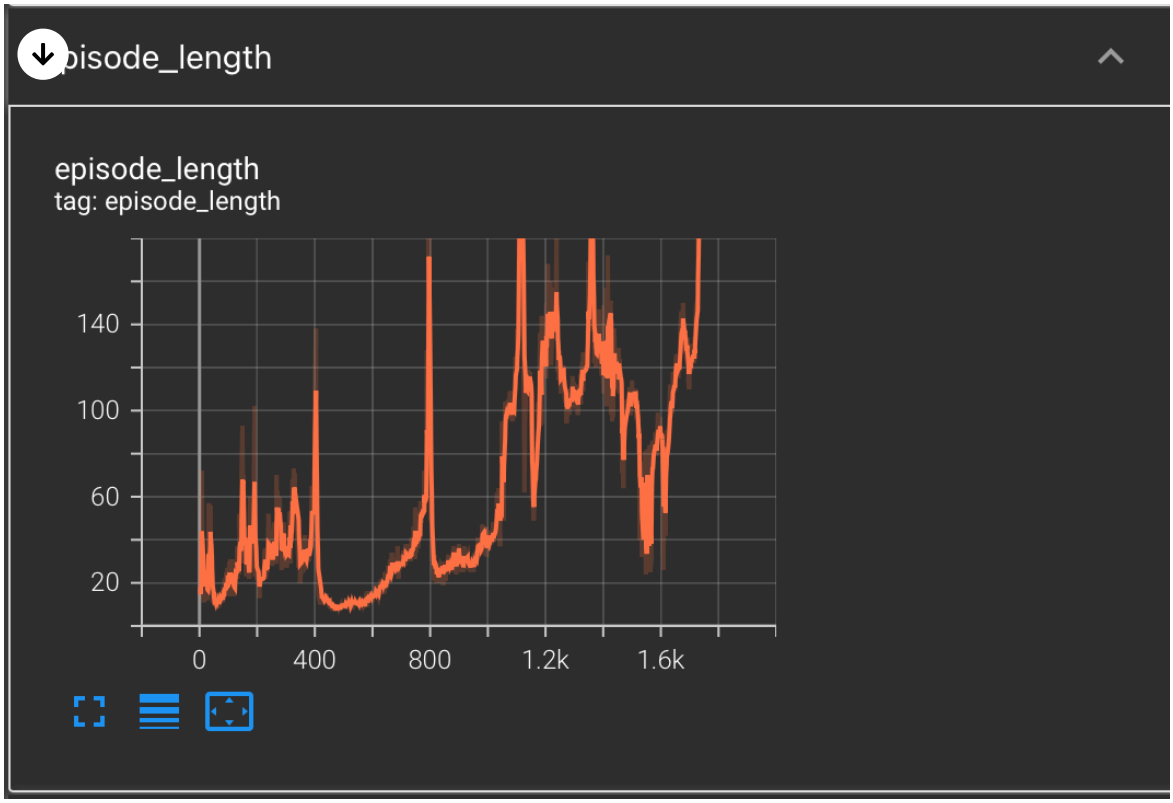
Reward



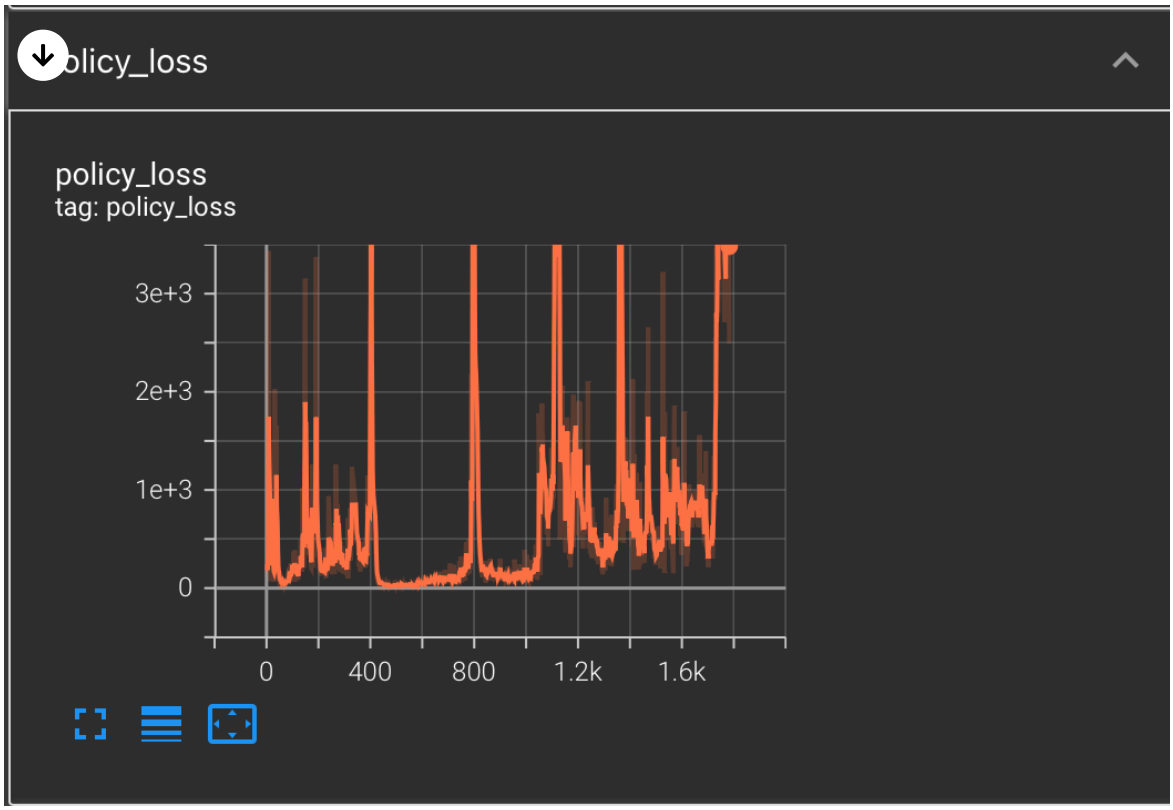
EWMA_Reward



Episode_Length



Policy_Loss



Training

```
Episode 1280    length: 110     reward: 111.0    ewma reward: 114.7
Episode 1300    length: 113     reward: 114.0    ewma reward: 108.04299783454267
Episode 1320    length: 111     reward: 112.0    ewma reward: 108.38507274343799
Episode 1340    length: 110     reward: 111.0    ewma reward: 115.46871298604162
Episode 1360    length: 199     reward: 200.0    ewma reward: 150.15650219078668
Episode 1380    length: 123     reward: 124.0    ewma reward: 142.53687328152634
Episode 1400    length: 115     reward: 116.0    ewma reward: 133.70775063689908
Episode 1420    length: 148     reward: 149.0    ewma reward: 130.68581763377045
Episode 1440    length: 123     reward: 124.0    ewma reward: 126.38510579481752
Episode 1460    length: 113     reward: 114.0    ewma reward: 123.57180326822677
Episode 1480    length: 105     reward: 106.0    ewma reward: 104.74062851805668
Episode 1500    length: 114     reward: 115.0    ewma reward: 106.19831233649252
Episode 1520    length: 79      reward: 80.0     ewma reward: 103.92600956409947
Episode 1540    length: 65      reward: 66.0     ewma reward: 75.46988866584486
Episode 1560    length: 27      reward: 28.0     ewma reward: 56.50736797011812
Episode 1580    length: 79      reward: 80.0     ewma reward: 64.79774758218277
Episode 1600    length: 97      reward: 98.0     ewma reward: 80.18196910618765
Episode 1620    length: 76      reward: 77.0     ewma reward: 74.58474108708093
Episode 1640    length: 105     reward: 106.0    ewma reward: 91.73164984057057
Episode 1660    length: 115     reward: 116.0    ewma reward: 107.20063017517944
Episode 1680    length: 142     reward: 143.0    ewma reward: 127.17076174241835
Episode 1700    length: 121     reward: 122.0    ewma reward: 125.929479784132
Episode 1720    length: 134     reward: 135.0    ewma reward: 126.74746693761534
Episode 1740    length: 199     reward: 200.0    ewma reward: 163.1986590602493
Episode 1760    length: 199     reward: 200.0    ewma reward: 186.80723734734212
Episode 1780    length: 199     reward: 200.0    ewma reward: 194.58183998630864
Solved! Running reward is now 195.11011058764353 and the last episode runs to 199 time steps!
Episode 1       Reward: 200.0
Episode 2       Reward: 200.0
Episode 3       Reward: 200.0
Episode 4       Reward: 200.0
Episode 5       Reward: 200.0
Episode 6       Reward: 200.0
Episode 7       Reward: 200.0
Episode 8       Reward: 200.0
Episode 9       Reward: 200.0
Episode 10      Reward: 200.0
```

Hyperparameter

hyperparameter_1	value
learning_rate	0.01
hidden_size	128

b. reinforce baseline

我們利用以下兩張圖片的結論 $B(s) \approx V^{\theta_\pi}(s)$ 挑選我們的 baseline function

↓. ne variance of REINFORCE PG with baseline can be written as:

$$\mathbb{V}\left[(G_0 - B(s_0))\frac{\partial}{\partial \theta_i}\log \pi_\theta(a_0 \mid s_0)\right] \quad \text{(With baseline)}$$

$$= \sum_s P(s_0 = s)\left(\mathbb{E}[(G_0 - B(s))^2(\frac{\partial}{\partial \theta_i}\log \pi_\theta(a_0 \mid s))^2 \mid s]\right) - \left(\sum_s P(s_0 = s)\mathbb{E}[(G_0 - B(s))\frac{\partial}{\partial \theta_i}\log \pi_\theta(a_0 \mid s_0) \mid s]\right)^2$$

$$= \sum_s P(s_0 = s)\left(\sum_a \pi_\theta(a \mid s)(\mathbb{E}[(G_0 - B(s))^2(\frac{\partial}{\partial \theta_i}\log \pi_\theta(a \mid s))^2 \mid s, a]\right) - \left(\sum_s P(s_0 = s)\sum_a \pi_\theta(a \mid s)\mathbb{E}[(G_0 - B(s))\frac{\partial}{\partial \theta_i}\log \pi_\theta(a \mid s) \mid s, a]\right)^2$$

$$= \sum_s P(s_0 = s)\left(\sum_a \pi_\theta(a \mid s)\left(\frac{\partial}{\partial \theta_i}\log \pi_\theta(a \mid s)\right)^2 \mathbb{E}[(G_0 - B(s))^2 \mid s, a]\right) - \left(\sum_s P(s_0 = s)\sum_a \pi_\theta(a \mid s)\frac{\partial}{\partial \theta_i}\log \pi_\theta(a \mid s)\mathbb{E}[G_0 \mid s, a]\right)^2$$

10

Quantifying Variance Reduction By $B(s)$

$$\mathbb{V}\left[G_0 \frac{\partial}{\partial \theta_i} \log \pi_\theta(a_0 \mid s_0)\right] - \mathbb{V}\left[(G_0 - B(s_0)) \frac{\partial}{\partial \theta_i} \log \pi_\theta(a_0 \mid s_0)\right]$$

$$= \sum_s P(s_0 = s) \left(\sum_a \underbrace{\pi_\theta(a \mid s)\left(\frac{\partial}{\partial \theta_i} \log \pi_\theta(a \mid s)\right)^2}_{:= c_a} \left(\mathbb{E}[G_0^2 \mid s, a] - \mathbb{E}[(G_0 - B(s))^2 \mid s, a]\right) \right)$$

$$= \sum_s P(s_0 = s) \sum_a c_a \left(\mathbb{E}[2B(s)G_0 - B(s)^2 \mid s, a]\right)$$

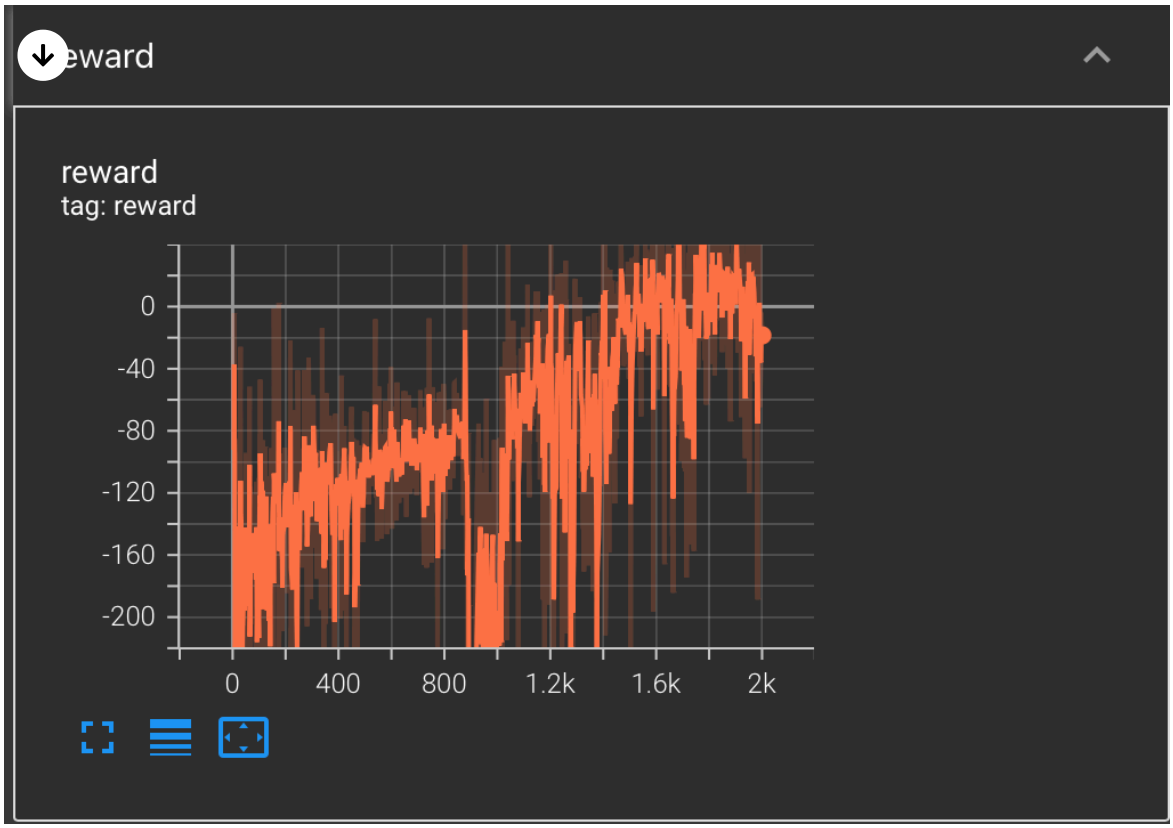
- Suppose $\mathbb{E}[G_0 \mid s, a] \equiv Q^{\pi_\theta}(s, a) \approx V^{\pi_\theta}(s)$, then we may choose $B(s) = V^{\pi_\theta}(s)$
- In practice, $B(s) = V^{\pi_\theta}(s)$ is a popular choice

11

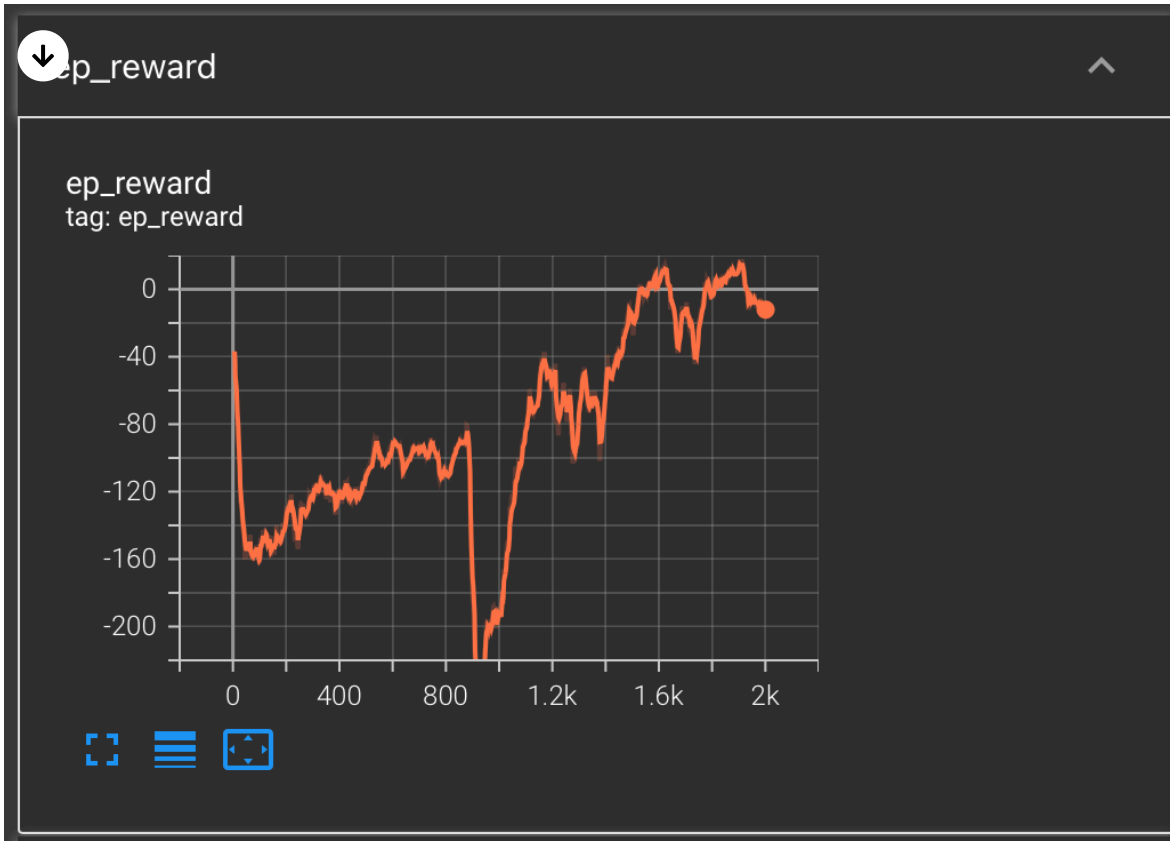
```
def calculate_loss(self, gamma=0.999):
        """
            Calculate the loss (= policy loss + value loss) to perform backpro
            TODO:
                1. Calculate rewards-to-go required by REINFORCE with the help
                2. Calculate the policy loss using the policy gradient
                3. Calculate the value loss using either MSE loss or smooth L1
        """

        # Initialize the lists and variables
        R = 0
        saved_actions = self.saved_actions
        loss = []
        value_losses = []
        returns = []

        ########## YOUR CODE HERE (8-15 lines) ##########
        # Calculate baseline
        baselines = self.get_v_values(gamma)

        # Calculate advantages and returns
        returns = []
        for reward, baseline in zip(reversed(self.rewards), reversed(baselines
            R = reward + gamma * R
            advantage = R - baseline
            returns.insert(0, advantage)

        returns = torch.tensor(returns)

        # Get log probabilities of actions
        log_probs = [action.log_prob for action in saved_actions]
        action_log_probs = torch.stack(log_probs, dim=0)

        # Calculate policy loss with V-value baseline
        policy_loss = -(returns * action_log_probs).sum()

        # Calculate value loss using MSE loss
        values = [action.value for action in saved_actions]
        value_predictions = torch.stack(values, dim=0).squeeze(1)
        value_targets = returns.detach()
        value_loss = F.mse_loss(value_predictions, value_targets)

        # Total loss is the sum of policy loss and value loss
        loss = policy_loss + value_loss

        ########## END OF YOUR CODE ##########

        return loss

    def get_v_values(self, gamma):
        state_values = []
        R = 0

        for action in reversed(self.saved_actions):
            # Assuming action.value contains the value of the next state
            R = action.value + gamma * R  # V^pi_theta(s) = r + gamma * V^pi_t
            state_values.insert(0, R)

        return state_values
```

- 這邊有加上判斷 episode 次數超過 2000 就卡掉
- 所以不是真的跑到要求

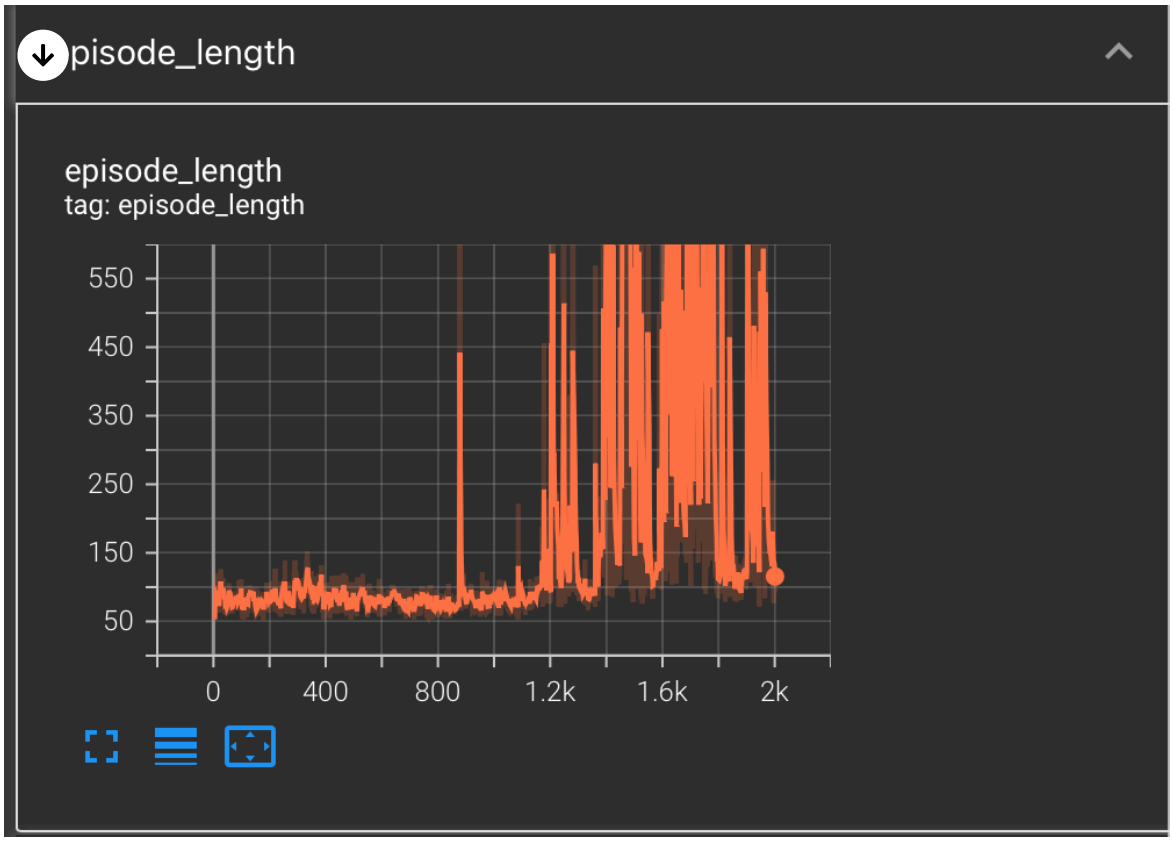
Reward



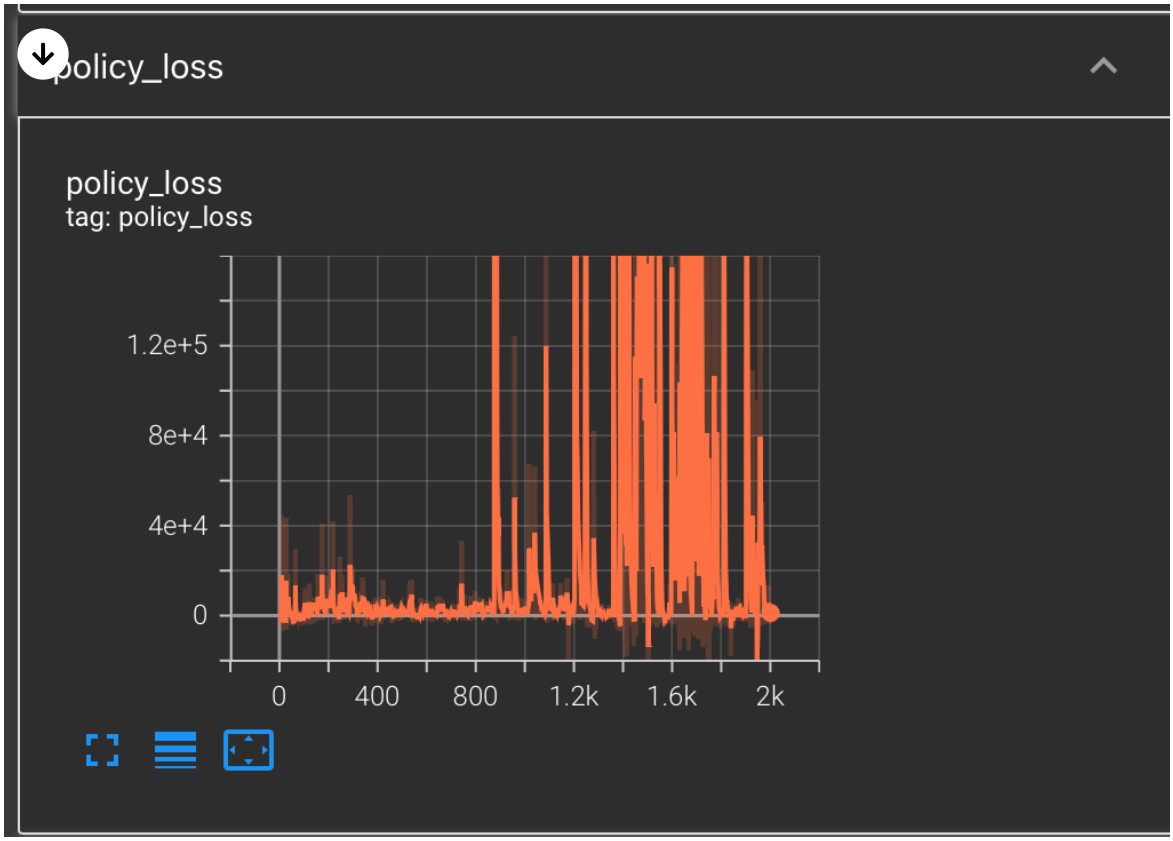
EWMA_Reward



Episode_Length



Policy_Loss



Training

```
Episode 1520    length: 142     reward: -6.838910653244909      ewma reward: -6.7273257922688
Episode 1540    length: 120     reward: -5.380564289640461      ewma reward: -0.5377983278301
Episode 1560    length: 119     reward: 1.461623725587586       ewma reward: 3.14260703776590
Episode 1580    length: 152     reward: 34.81830601055708       ewma reward: 7.03498572026662
Episode 1600    length: 113     reward: 4.046959958339045       ewma reward: 6.19124600066677
Episode 1620    length: 156     reward: -26.276192906992023     ewma reward: 9.12964970160801
Episode 1640    length: 173     reward: -59.309993787762835     ewma reward: -1.2306483767513
Episode 1660    length: 999     reward: 53.78162365949972       ewma reward: -13.087592451205
Episode 1680    length: 176     reward: 15.898343308786352      ewma reward: -25.429994724678
Episode 1700    length: 999     reward: -14.013114693577428     ewma reward: -13.561396711515
Episode 1720    length: 138     reward: -64.84797235301987      ewma reward: -24.709567128956
Episode 1740    length: 999     reward: -34.666218840992336     ewma reward: -35.273647751098
Episode 1760    length: 130     reward: -38.7680188869668       ewma reward: -11.611972595533
Episode 1780    length: 163     reward: 39.69725116444994       ewma reward: 2.93883525578899
Episode 1800    length: 132     reward: 57.43289997864076       ewma reward: -0.7321955244822
Episode 1820    length: 113     reward: 6.134986596934951       ewma reward: 4.50008307479650
Episode 1840    length: 82      reward: 10.224368049741017      ewma reward: 8.73637864352512
Episode 1860    length: 127     reward: 14.50527759694232       ewma reward: 10.1521124442174
Episode 1880    length: 82      reward: -73.95997941492504      ewma reward: 8.56753085637269
Episode 1900    length: 115     reward: -20.350003301917013     ewma reward: 12.9510024211470
Episode 1920    length: 999     reward: 24.081872661693758      ewma reward: 5.99018289389370
Episode 1940    length: 105     reward: -18.18721871699148      ewma reward: -5.5770317245936
Episode 1960    length: 130     reward: -89.16052269383081      ewma reward: -7.0442197591596
Episode 1980    length: 93      reward: 30.461035001325797      ewma reward: -3.6499338153366
Episode 2000    length: 120     reward: -15.051329955840473     ewma reward: -12.365464661017
Solved! Running reward is now -12.565417942675053 and the last episode runs to 92 time steps!
Episode 1       Reward: -81.29795442491276
Episode 2       Reward: -43.534461142386206
Episode 3       Reward: -166.38640362046084
Episode 4       Reward: -2.9735699551213344
Episode 5       Reward: -8.62361110330744
Episode 6       Reward: 131.92094295153407
Episode 7       Reward: 99.14846399644294
Episode 8       Reward: -5.15241338220433
Episode 9       Reward: -48.309868228121815
Episode 10      Reward: -19.70331522102245
```

Hyperparameter

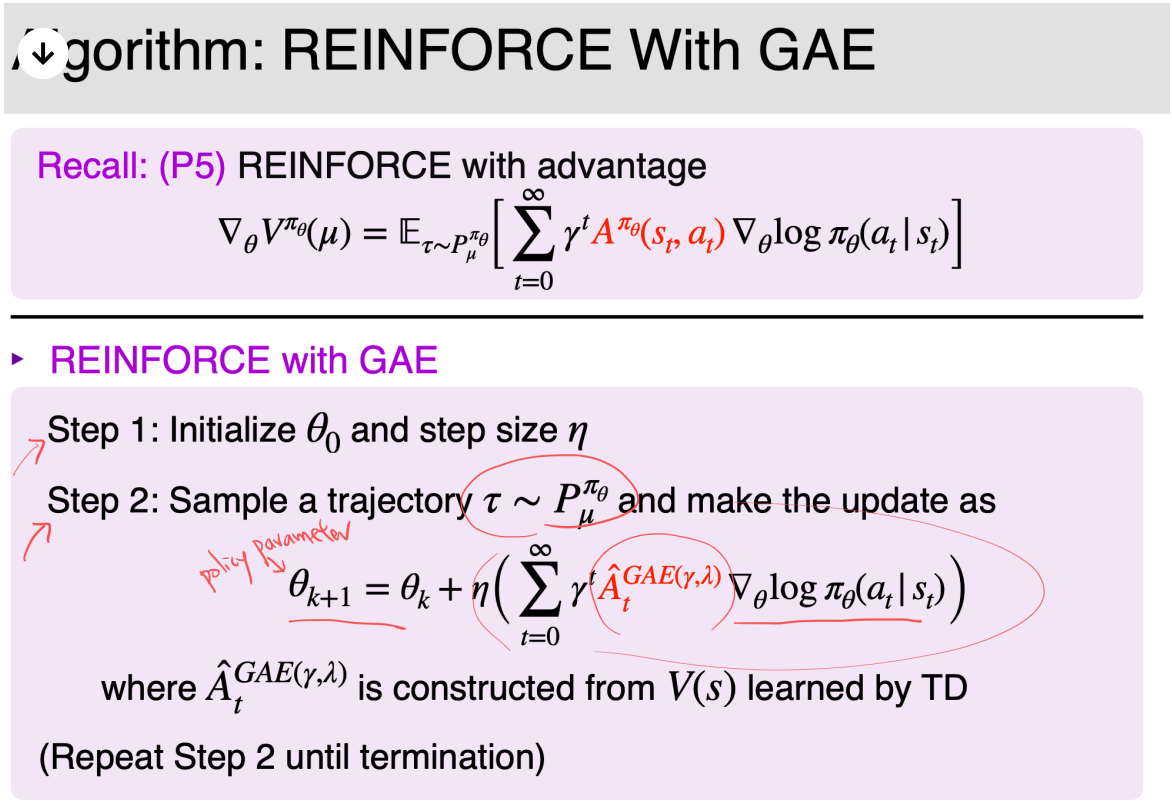
hyperparameter_1	value
learning_rate	0.0015

```
self.discrete = isinstance(env.action_space, gym.spaces.Discrete)
self.observation_dim = env.observation_space.shape[0]
self.action_dim = env.action_space.n if self.discrete else env.action_space.shape[0]
self.hidden_size = 128
self.double()
```

```
if __name__ == '__main__':
    # For reproducibility, fix the random seed
    random_seed = 10
    lr = 0.001
    env = gym.make("LunarLander-v2")
    env.seed(random_seed)
    torch.manual_seed(random_seed)
    train(lr)
    test(f'LunarLander_reinforce_baseline={lr}.pth')
```

c. reinforce gae lambda=0.99

實做概念



我們利用這張投影片的 gradient 公式計算 gradient

```
# get probility and value
log_probs = [action.log_prob for action in saved_actions]
values = [action.value for action  in saved_actions]

# calculate (P4) advantages = E[advantages * log_probs(a|s)] / (1 - gamma)
advantages = GAE(gamma, lambda_, None)(self.rewards, values)
advantages = advantages.detach()

action_log_probs = torch.stack(log_probs, dim=0)
values = torch.stack(values, dim=0)[:,0]

policy_losses = -(advantages * action_log_probs).sum() / (1 - gamma)
```

接著按流程計算 GAE function

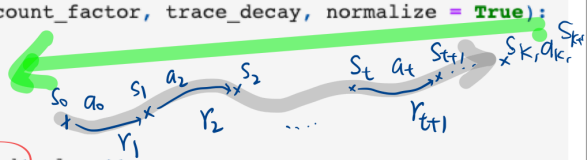
Some Discussions on GAE

1. Do we need to wait until the end of a trajectory to construct GAE?

Yes!

2. How to efficiently calculate GAE for different t of the same trajectory?

```
def calculate_advantages(rewards, values, discount_factor, trace_decay, normalize = True):

    advantages = []
    advantage = 0
    next_value = 0

    for r, v in zip(reversed(rewards), reversed(values)):
        td_error = r + next_value * discount_factor - v
        advantage = td_error + advantage * discount_factor * trace_decay
        next_value = v
        advantages.insert(0, advantage)

    advantages = torch.tensor(advantages)
```



Estimate $A^{\pi}(s_0,a_0), A^{\pi}(s_1,a_1), \ldots$

$A^{\pi}(s_t,a_t), \ldots$

3. Where does $V(s)$ in GAE come from?

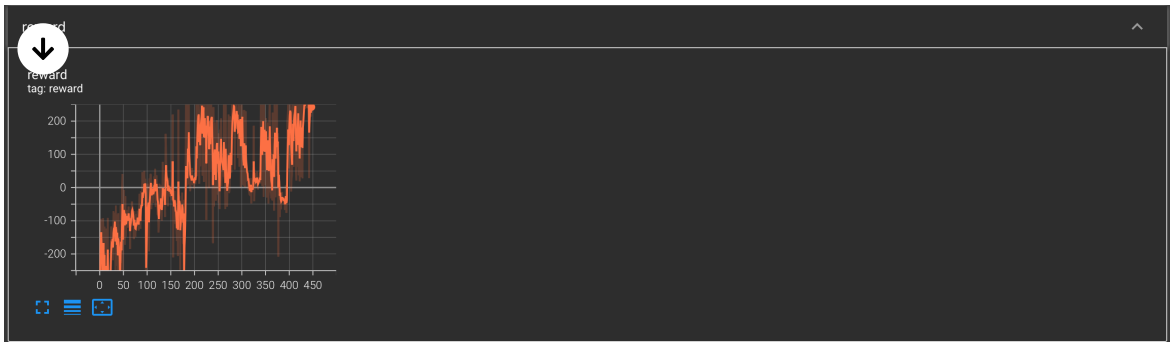
Any model-free prediction methods!

$\delta_K = r_{K+1} + \gamma V(s_{K+1}) - V(s_K)$

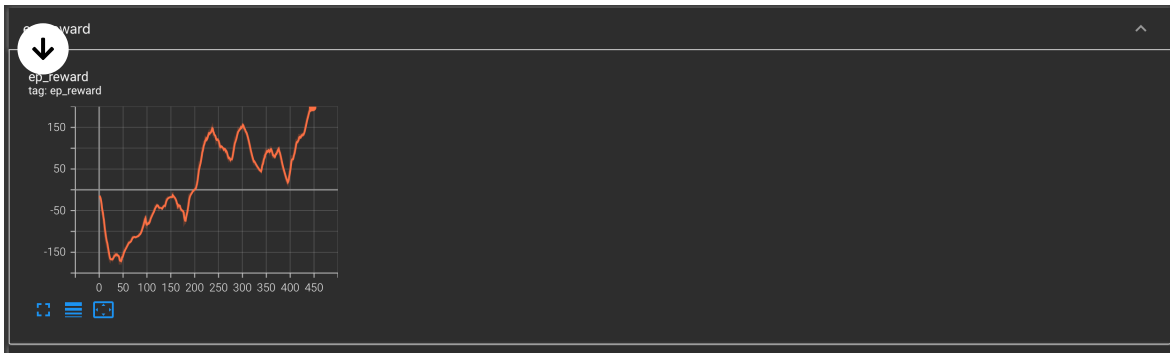
$\delta_{K-1} = r_K + \gamma V(s_K) - V(s_{K-1})$

```
# refer to slide Lec11 P.6, use defition of GAE
advantages = []
advantage = 0
next_value = 0

stepCount = 0
for r, v in zip(reversed(rewards), reversed(values)):
    stepCount += 1
    td_error = r + next_value * self.gamma - v
    advantage = td_error + advantage * self.gamma * self.lambda_
    next_value = v
    advantages.insert(0, advantage)
    if self.num_steps is not None and stepCount > self.num_steps:
        break
advantages = torch.Tensor(advantages)
advantages = (advantages - advantages.mean()) / (advantages.std())
return advantages
```

Reward

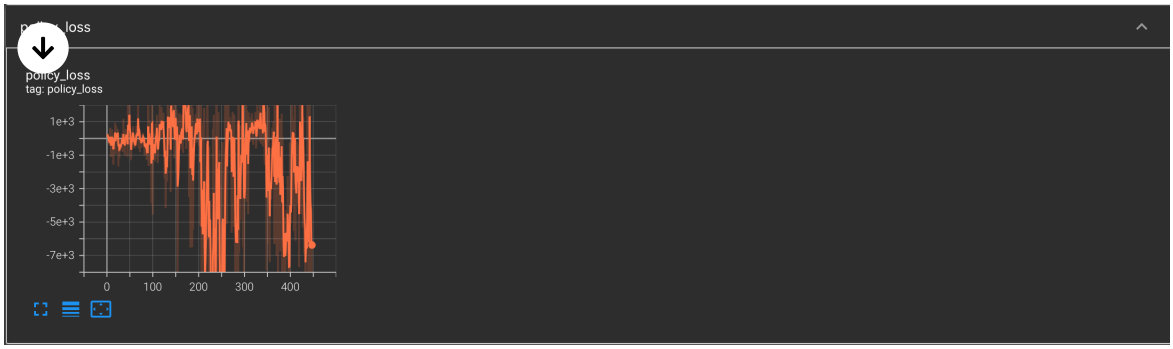


EWMA_Reward



Episode_Length

Policy_Loss



Training

```
Episode 20      length: 119     reward: -306.08468716706574      ewma reward: -158.67930241665272
Episode 40      length: 188     reward: -180.30257182147074      ewma reward: -158.26239144468232
Episode 60      length: 89      reward: -78.80511240438142       ewma reward: -129.34892286326019
Episode 80      length: 79      reward: -69.40044386237153       ewma reward: -110.30496278907867
Episode 100     length: 999     reward: 4.291255190501703        ewma reward: -86.11791006917521
Episode 120     length: 123     reward: 36.892970659997104       ewma reward: -34.95159069287097
Episode 140     length: 999     reward: 117.63315393843953       ewma reward: -22.275185306241937
Episode 160     length: 109     reward: -109.41942622016546      ewma reward: -26.0440866106612
Episode 180     length: 630     reward: -90.99088491636368       ewma reward: -79.66463514301721
Episode 200     length: 150     reward: 33.703125767635925       ewma reward: 1.9953777851598913
Episode 220     length: 220     reward: 16.879081604054846       ewma reward: 111.86629041416319
Episode 240     length: 390     reward: 242.02772113380448       ewma reward: 134.95395684242428
Episode 260     length: 133     reward: 55.224814187120444       ewma reward: 96.396850408255
Episode 280     length: 417     reward: 259.204169971016         ewma reward: 92.12937752124816
Episode 300     length: 283     reward: 226.3311918915353        ewma reward: 156.6171296453797
Episode 320     length: 79      reward: -47.42472689646631       ewma reward: 75.71128335227651
Episode 340     length: 212     reward: 259.9323656759929        ewma reward: 53.459291019332596
Episode 360     length: 175     reward: -10.251174622791126      ewma reward: 98.37663080842466
Episode 380     length: 279     reward: -89.16486454551442       ewma reward: 73.1449249886289
Episode 400     length: 410     reward: 257.9009331691898        ewma reward: 54.9980567810101
Episode 420     length: 166     reward: 36.805004003878395       ewma reward: 129.19434503519935
Episode 440     length: 398     reward: 250.49130871299369       ewma reward: 193.25982079258833
Solved! Running reward is now 201.27898424727456 and the last episode runs to 299 time steps!
Episode 1       Reward: 191.96337246926416
Episode 2       Reward: 208.83916730249626
Episode 3       Reward: 285.0393582553071
Episode 4       Reward: 46.45398832435623
Episode 5       Reward: 185.97919906280947
Episode 6       Reward: 236.04690554735154
Episode 7       Reward: 257.02887779900055
Episode 8       Reward: 231.50008373562733
Episode 9       Reward: 233.25313947264533
Episode 10      Reward: 17.090291902061267
```

hyperparameter_1	value
random_seed	10
learning_rate	0.01
learning_rate_decay	0.9
discount_factor	0.99
trace_decay	0.99
hidden_size	128

c. reinforce gae lambda=0.8

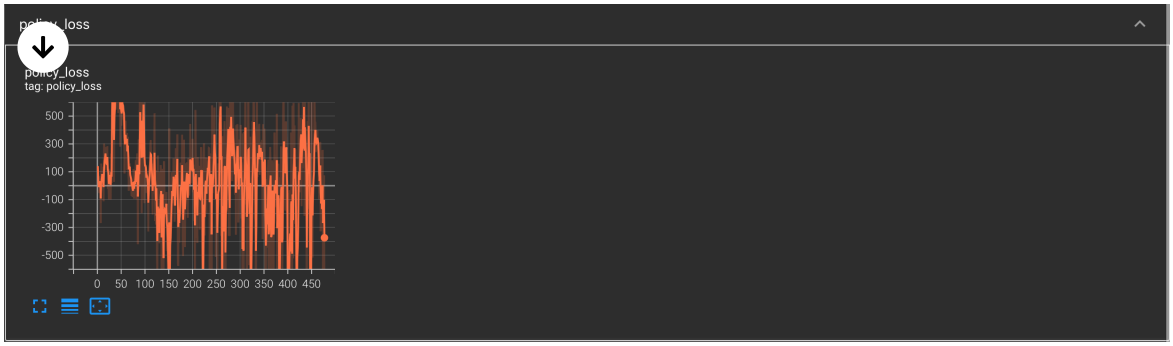
- 這邊有加上判斷 episode 次數超過 500 就卡掉
- 所以不是真的跑到要求

Reward

EWMA_Reward

Episode_Length

Policy_Loss



Training

```
Episode 20      length: 138     reward: -196.68395535916        ewma reward: -127.8989481220854
Episode 40      length: 86      reward: -657.1144955690355      ewma reward: -283.97530656088145
Episode 60      length: 72      reward: -461.5938068995624      ewma reward: -441.9484943512645
Episode 80      length: 79      reward: -88.63682919181348      ewma reward: -355.68690695737
Episode 100     length: 245     reward: -71.44587811276388      ewma reward: -275.739623168452
Episode 120     length: 217     reward: -215.35888885784885     ewma reward: -204.6927809383527
Episode 140     length: 190     reward: -208.00886419172872     ewma reward: -177.7223399011013
Episode 160     length: 106     reward: -144.75595338116477     ewma reward: -101.27799046382759
Episode 180     length: 230     reward: -135.44186633383555     ewma reward: -99.2623104246206
Episode 200     length: 287     reward: 9.448457892925077       ewma reward: -80.78846716914282
Episode 220     length: 799     reward: -160.69720054000632     ewma reward: -139.9168063739193
Episode 240     length: 589     reward: 144.6672608203757       ewma reward: -48.96311581287912
Episode 260     length: 168     reward: -45.196670576694615     ewma reward: -114.27868072956166
Episode 280     length: 140     reward: -135.76438656280752     ewma reward: -109.70297320244065
Episode 300     length: 168     reward: 28.27833743404048       ewma reward: -58.962672553619726
Episode 320     length: 380     reward: -57.154336258243035     ewma reward: -56.150496674930785
Episode 340     length: 213     reward: -210.4366690754444      ewma reward: -107.38334366327985
Episode 360     length: 526     reward: -94.10642278638817      ewma reward: -93.19439482565035
Episode 380     length: 157     reward: -11.774687476388522     ewma reward: -63.27483784859901
Episode 400     length: 999     reward: -3.687993391008379      ewma reward: -64.56363392883219
Episode 420     length: 273     reward: -90.2734550177167       ewma reward: -49.88012707228572
Episode 440     length: 999     reward: 85.81342541431884       ewma reward: -86.94100605479544
Episode 460     length: 131     reward: -173.09286987965518     ewma reward: -97.71752326295282
Episode 480     length: 166     reward: -137.97600115194558     ewma reward: -50.26533224300745
Episode 500     length: 167     reward: -23.57962281439822      ewma reward: -39.15757480175209
Solved! Running reward is now -40.73127567194943 and the last episode runs to 245 time steps!
Episode 1       Reward: -31.148852063941476
Episode 2       Reward: -14.561764138150991
Episode 3       Reward: -205.53713590814056
Episode 4       Reward: -249.6750200572577
Episode 5       Reward: -133.810794843626
Episode 6       Reward: -195.03977668006928
Episode 7       Reward: -216.64175400254206
Episode 8       Reward: -196.3886072499917
Episode 9       Reward: -50.79953493323336
Episode 10      Reward: -81.4995434350619
```

Hyperparameter

hyperparameter_1	value
random_seed	10
learning_rate	0.01
learning_rate_decay	1
discount_factor	0.99
trace_decay	0.8
NN_hidden_size	128

c. reinforce gae lambda=0.98

- 這邊有加上判斷 episode 次數超過 500 就卡掉
- 所以不是真的跑到要求

Reward

EWMA_Reward

Episode_Length

Policy_Loss

Training

```
Episode 20      length: 105     reward: -155.1448111304258      ewma reward: -127.58317675422255
Episode 40      length: 133     reward: -533.9967866406203      ewma reward: -278.5537657179088
Episode 60      length: 123     reward: -525.8615380587273      ewma reward: -278.1300997749183
Episode 80      length: 115     reward: -60.40937074179726      ewma reward: -194.0818796316476
Episode 100     length: 95      reward: -31.72703397201505      ewma reward: -127.43847555289132
Episode 120     length: 308     reward: -104.25330233315677     ewma reward: -145.89463024882204
Episode 140     length: 293     reward: -20.496540048273744     ewma reward: -128.81034980416092
Episode 160     length: 341     reward: -19.61271447940777      ewma reward: -64.86602239931372
Episode 180     length: 141     reward: -141.6994060947078      ewma reward: -70.72337786547656
Episode 200     length: 148     reward: -145.76080612577562     ewma reward: -104.60061430106369
Episode 220     length: 785     reward: 221.36785077738267      ewma reward: -48.84978386688815
Episode 240     length: 235     reward: -14.594209880938294     ewma reward: -59.48205376107391
Episode 260     length: 362     reward: -280.61411555928504     ewma reward: -61.533042388909564
Episode 280     length: 347     reward: 249.6335038786512       ewma reward: -1.2077738305343608
Episode 300     length: 511     reward: -239.35530775028226     ewma reward: -20.382717816683808
Episode 320     length: 193     reward: 48.970291771271775      ewma reward: -50.90527933613881
Episode 340     length: 266     reward: 27.455870141322023      ewma reward: -28.272830482934236
Episode 360     length: 326     reward: 300.6639363704692       ewma reward: -29.032198417046 8
Episode 380     length: 375     reward: 241.77634282376823      ewma reward: -93.47938949678414
Episode 400     length: 179     reward: 28.15463365386836       ewma reward: -13.359156514736656
Episode 420     length: 255     reward: -163.7926895184731      ewma reward: -53.64577291464728
Episode 440     length: 217     reward: -199.949045450901       ewma reward: -94.4456836313767
Episode 460     length: 163     reward: 10.13609102738225       ewma reward: -91.4064119213387
Episode 480     length: 218     reward: -1.9943937429893595     ewma reward: -50.33914077554006
Episode 500     length: 407     reward: -93.77025095285617      ewma reward: -59.337642399641446
Solved! Running reward is now -63.446761668640406 and the last episode runs to 150 time steps!
Episode 1       Reward: -273.7847319281313
Episode 2       Reward: 193.55078191255785
Episode 3       Reward: -18.81383811362494
Episode 4       Reward: 17.878630862263194
Episode 5       Reward: 78.74008079248344
Episode 6       Reward: 245.43378993596656
Episode 7       Reward: 215.97091089248696
Episode 8       Reward: 145.25511775083243
Episode 9       Reward: 269.1973945634143
Episode 10      Reward: -156.66545925565907
```

Hyperparameter

hyperparameter_1	value
random_seed	10
learning_rate	0.01
learning_rate_decay	0.9
discount_factor	0.9999
trace_decay	0.98
NN_hidden_size	128

summarize

- 不同的 λ 可能會影響收斂速度
- 甚至影響收斂結果
- Reward, EWMA_Reward, Policy_Loss, Episode_Length, Policy_Loss 也會不同