HW6
Reinforcement Learning (Policy-Gradient)

1 Short Background

1.1 A3C

First-order policy-gradient methods learn a good control policy by calculating the gradient of the expected return of policy at the current policy parameters, and updating the parameters in the direction of the gradient:

$$\theta' \leftarrow \theta_{old} + \alpha \nabla_\theta \eta(\pi_\theta)|_{\theta=\theta_{old}} \tag{1}$$

A popular first-order policy-gradient method is Asynchronous Advantage Actor-critic (A3C) [3]. It is highly recommended that you go over the paper and/or refer to popular blogs on A3C (e.g. [2]). A3C calculates the gradient from Eq. 1 as:

$$\nabla_\theta \eta(\pi_\theta) \equiv E_\tau \left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi(a_t|s_t,\theta) A(s_t,a_t) + \beta \sum_{t=0}^{T-1} \nabla_\theta \mathrm{H}(s_t|\theta) \right] \tag{2}$$

where the expectation is over the trajectory roll-outs (τ) using policy π, A is the advantage function, H is the policy entropy, T is the time horizon. β is the weight on the entropy term, and therefor controls the amount of exploration the policy performs.

1.2 Generalized Advantage Estimation

The advantage function $A(s_t, a_t)$ is a measure of the how good action a_t is when in state s_t, compared to the average behavior over all possible actions when in s_t, i.e. $A(s_t, a_t) = Q(s_t, a_t) - V(s_t)$. Different ways of estimating $Q(s_t, a_t)$ lead to different estimates ($\hat{A}$) of advantage and any of these be used in Eq. 2

$$\hat{A}(s_t, a_t) = \sum_{l=0}^{\infty} \gamma^l r_{t+l} - V(s_t) \tag{3}$$

$$\hat{A}(s_t, a_t) = r_t + \gamma V(s_{t+1}) - V(s_t) \tag{4}$$

$$\hat{A}(s_t, a_t) = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l} \tag{5}$$

The last estimator is known as generalized advantage estimator (GAE) [4]. $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ is the TD residual error at time t, γ is the discount factor. λ is a parameter that control the bias-variance trade-off in the advantage estimation. More concretely, $\lambda = 1$ leads to the first estimator; this estimator has high variance and low bias. $\lambda = 0$ leads to the second estimator; this estimator has low variance but high bias. Convince yourself that you can derive the first and second estimators by setting the corresponding λ in Equation 5. Also, the idea of bias and variance for $\hat{A}$ should be clear. You are recommended to skim the GAE paper to get better intuition.

2 Problems

In this assignment, you would understand an existing implementation of A3C that runs efficiently on GPUs. The code, dubbed GA3C, is publicly available [1]. You would explore the effect of entropy regularization parameter (β) and the advantage function (A) in Equation 2. by training policies for a couple of Atari games. Proceed as follows:

- Clone/Fork the code from `https://github.com/tgangwani/IE598_RL`. It is highly recommended that you spend a good amount of the time understanding most of the code. The same framework would be used for later assignments. You'll reap rich dividends if you invest time now. Go over the README.md file on Github to get a feel for the code structure.
- **[PART-1] Sensitivity to** β - Try the following 3 values of β on the game *Seaquest* - $\{0, 1e\text{-}2, 1\}$. Report the learning curve, i.e. plot of average return (RScore from GA3C) vs. time. Explain what you see and try to justify.
- **[PART-2] Bias-Variance Trade-off** - In this part, you'll add GAE to the code. `ProcessAgent.py` has a function `_accumulate_rewards` which calculates the advantage estimate. The current code uses the high variance estimator ($\lambda = 1$). Modify this to calculate advantage using GAE, parameterized by λ. Run the code on the game $Pong$ using the following 3 values of λ - $\{0, 0.9, 1\}$. Report the learning curve. Explain what you see and try to justify.

References

[1] BABAEIZADEH, M., FROSIO, I., TYREE, S., CLEMONS, J., AND KAUTZ, J. Ga3c: Gpu-based a3c for deep reinforcement learning. *arXiv preprint arXiv:1611.06256* (2016).

[2] JULIANI, A. https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2.

[3] MNIH, V., BADIA, A. P., MIRZA, M., GRAVES, A., LILLICRAP, T. P., HARLEY, T., SILVER, D., AND KAVUKCUOGLU, K. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning* (2016).

[4] SCHULMAN, J., MORITZ, P., LEVINE, S., JORDAN, M., AND ABBEEL, P. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438* (2015).