1 Why did we not improve the performance of PPO?

We hypothesize that our algorithm works better on tasks whose initial and undiscounted stationary state distributions differ significantly. The difference between the initial and undiscounted stationary state distributions implies a large distance between the discounted and undiscounted state distributions such that the task requires the state distribution correction to emphasize states appropriately.

Our hypothesis is supported by the dominant performances of our algorithm on MountainCar continuous, Pendulum and two dm_control Reacher tasks [Tassa et al., 2020]. Our results are shown in Figure 1.

The list of hyperparameters is the same as the one described in Table 5 in the appendix. We still use random search [Bergstra and Bengio, 2012], and each algorithm gets tested on 200 combinations of hyperparameters.

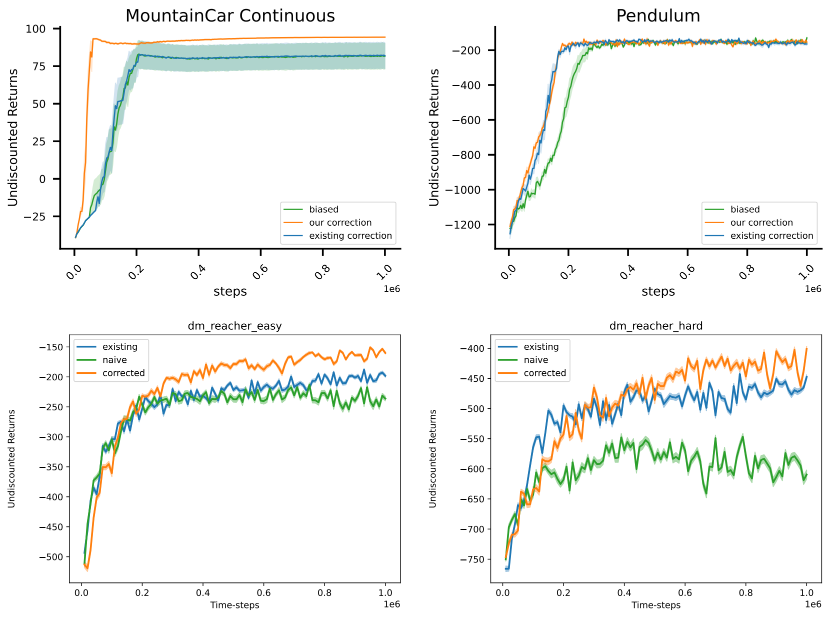

Figure 1: Our algorithm gains the best performance in all four tasks.

2 How much bias do we add to the algorithm?

Let us consider a data buffer $\mathcal{D}$ consisting of k trajectories under the policy π, each with length T. Let us recall the proposed state emphasis trace $c_{\mathcal{D}}(s)$, defined in Section 5, which equals

$$c_{\mathcal{D}}(s) = \frac{\sum_{i=1}^{|\mathcal{D}|} \gamma^{t_i} \mathbb{1}[S_i = s]}{\sum_{i=1}^{|\mathcal{D}|} \mathbb{1}[S_i = s]}.$$

Also, this trace is the optimal solution that minimizes the neural network's mean square error. This section provides an error bound using the trace $c_{\mathcal{D}}$.

Corollary 2.1. *Consider an irreducible and positive recurrent MDP under policy π with a finite state space $\mathcal{S}$. Given a data buffer $\mathcal{D}$ consisting of k trajectories under a policy π, each with length T, if*

$$k \geq \frac{2}{\epsilon^2} \log \frac{|\mathcal{S}|}{\delta}$$

and

$$T \geq \frac{\log \frac{\epsilon}{2}}{\log \gamma},$$

then with probability at least $1 - \delta$, the error is smaller than ϵ, that is

$$\max_{s \in \mathcal{S}} |\hat{d}_{\mathcal{D}}(s) c_{\mathcal{D}}(s)(1-\gamma)\frac{|\mathcal{D}|}{k} - d_{\pi,\gamma}(s)| \leq \epsilon.$$

Here $\hat{d}_{\mathcal{D}}(s)$ is the sampling distribution from the given data buffer, equaling to

$$\hat{d}_{\mathcal{D}}(s) = \frac{1}{|\mathcal{D}|} \sum_{i=1}^{|\mathcal{D}|} \mathbb{1}[S_i = s].$$

The true approximated correction equals to $(1-\gamma)\frac{|\mathcal{D}|}{k} c_{\mathcal{D}}(s)$. As discussed in the paper, our neural network only aims to learn $c_{\mathcal{D}}(s)$ and let the learning rate absorbs other constants. That's why our bound multiplies several constants to $c_{\mathcal{D}}$.

This corollary claims the following. Weighting each data sample in the buffer by our approximated correction changes the state sampling distribution. This modified state distribution converges to the discounted stationary state distribution at the speed of $O(\max\{\sqrt{\frac{T}{|\mathcal{D}|}}, \gamma^T\})$.

Proof. Define a random variable X_j dependent on the j-th trajectory in the data buffer as

$$X_j = (1-\gamma) \sum_{t=0}^{T-1} \gamma^t \mathbb{1}[S_{t,j} = s].$$

So we can rewrite the approximated correction as

$$\begin{aligned}
c_{\mathcal{D}}(s)(1-\gamma)\frac{|\mathcal{D}|}{k} &= \frac{|\mathcal{D}|}{\sum_{i=1}^{|\mathcal{D}|}\mathbb{1}[S_i = s]}\frac{1}{k}(1-\gamma)\sum_{j=1}^{k}\sum_{t=0}^{T-1}\gamma^t\mathbb{1}[S_{t,j} = s] \\
&= \frac{|\mathcal{D}|}{\sum_{i=1}^{|\mathcal{D}|}\mathbb{1}[S_i = s]}\frac{1}{k}\sum_{j=1}^{k}X_j \\
&= \frac{1}{\hat{d}_{\mathcal{D}}(s)}\frac{1}{k}\sum_{j=1}^{k}X_j.
\end{aligned}$$

Thus for each state s, the error term can be bounded as

$$\begin{aligned}
&|\hat{d}_{\mathcal{D}}(s)c_{\mathcal{D}}(s)(1-\gamma)\frac{|\mathcal{D}|}{k} - d_{\pi,\gamma}(s)| \\
&= |\hat{d}_{\mathcal{D}}(s)\frac{1}{\hat{d}_{\mathcal{D}}(s)}\frac{1}{k}\sum_{j=1}^{k}X_j - d_{\pi,\gamma}(s)| \\
&= |\frac{1}{k}\sum_{j=1}^{k}X_j - d_{\pi,\gamma}(s)| \\
&\le |\frac{1}{k}\sum_{j=1}^{k}X_j - (1-\gamma)\sum_{t=0}^{T-1}\gamma^t P_\pi(S_t = s)| + |(1-\gamma)\sum_{t=0}^{T-1}\gamma^t P_\pi(S_t = s) - d_{\pi,\gamma}(s)| \\
&= |\frac{1}{k}\sum_{j=1}^{k}X_j - (1-\gamma)\sum_{t=0}^{T-1}\gamma^t P_\pi(S_t = s)| + |(1-\gamma)\sum_{t\ge T}\gamma^t P_\pi(S_t = s)| \\
&\le |\frac{1}{k}\sum_{j=1}^{k}X_j - (1-\gamma)\sum_{t=0}^{T-1}\gamma^t P_\pi(S_t = s)| + (1-\gamma)\sum_{t\ge T}\gamma^t \\
&\le |\frac{1}{k}\sum_{j=1}^{k}X_j - (1-\gamma)\sum_{t=0}^{T-1}\gamma^t P_\pi(S_t = s)| + \gamma^T.
\end{aligned}$$

When $T \ge \dfrac{\log\frac{\epsilon}{2}}{\log\gamma}$, we have $\gamma^T \le \frac{\epsilon}{2}$.

Notice that each random variable X_j depends on the corresponding trajectory; thus, $X_j,\ j = 1, \cdots, k$ are independent. Meanwhile, their expectations are the same and for all $j = 1, \cdots, k$, the expectation equals

$$E[X_j] = (1-\gamma)\sum_{t=0}^{T-1}\gamma^t P_\pi(S_t = s).$$

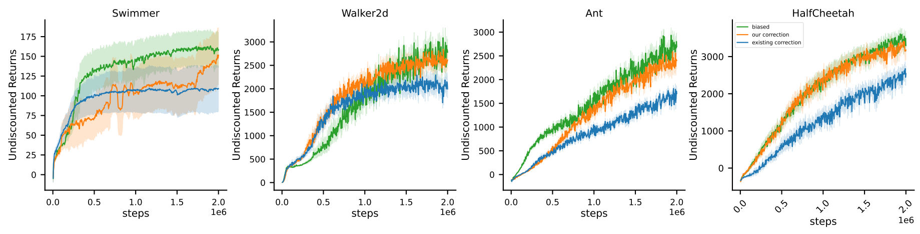

Figure 2: We tune hyperparameters for our algorithm and baselines on each task.

Thus, we can bound the other part with Hoeffding's inequality. For each state s, with probability at most $\frac{\delta}{|\mathcal{S}|}$, when $k \geq \frac{2}{\epsilon^2} \log \frac{|\mathcal{S}|}{\delta}$, we have

$$|\frac{1}{k}\sum_{j=1}^{k} X_j - (1-\gamma)\sum_{t=0}^{T-1} \gamma^t P_\pi(S_t = s)| \leq \frac{\epsilon}{2}.$$

□

3 What are the learning performances of PPO on MuJoCo with hyperparameter tuned for each task?

The result of our algorithm and the existing correction in Figure 4 of the main paper uses hyperparameters tuned on the Hopper task. For the biased PPO, all hyperparameters are copied from the OpenAI implementation. Thus, we tune hyperparameters for our algorithm and baselines on each task separately, and the result is shown in Figure 2. The conclusion is not changed. Our algorithm fills the performance gap between the biased and theoretically correct PPO with the existing correction on MuJoCo tasks.

The list of hyperparameters is the same as the one described in Table 5 in the appendix. We still use random search, and each algorithm gets tested on 900 combinations of hyperparameters.

References

Tassa, Y., Tunyasuvunakool, S., Muldal, A., Doron, Y., Liu, S., Bohez, S., Merel, J., Erez, T., Lillicrap, T.P., Heess, N.M. (2020). dm_control: Software and Tasks for Continuous Control. Softw. Impacts, 6, 100022.

Bergstra, J. and Bengio, Y. Random search for hyper-parameter optimization. Journal of machine learning research, 13(2), 2012.