AMBER: Adaptive Multi-Batch Experience Replay for Continuous Action Control

Seungyul Han and Youngchul Sung

Dept. of Electrical Engineering

KAIST

SURL Workshop, IJCAI 2019, Macao

Aug. 12, 2019

Proximal Policy Optimization (PPO)

- Proximal policy optimization [Schulman et al., 2017] : A stable RL algorithm.
- PPO updates the policy parameter θ with the following objective function :

$$\hat{J}_{PPO}(\theta) = \frac{1}{M}\sum_{m=0}^{M-1} \min\{\rho_m \hat{A}_m, \mathrm{clip}_\epsilon(\rho_m)\hat{A}_m\}$$

  – where $\rho_m = \frac{\pi_\theta(a_m|s_m)}{\pi_{\theta_i}(a_m|s_m)}$ is importance sampling (IS) weight,
  – $\hat{A}_m$ is estimated by generalized advantage estimation (GAE) [Schulman et al., 2015],
  – $\mathrm{clip}_\epsilon(\cdot) = \mathrm{clip}(\cdot, 1-\epsilon, 1+\epsilon)$.

- θ is updated to maximize the objective function.
- Clipped IS weight enables stable policy update.

On-Policy Learning

- **On-policy learning** : PPO only uses the current sample batch B_i at i-th policy update.

$$B_i = \{(s_{i,0}, a_{i,0}, r_{i,0}), \cdots, (s_{i,N-1}, a_{i,N-1}, r_{i,N-1})\} \tag{1}$$

- Previous batches generated by old policies are not used for the update.
- On-policy learning is sample-inefficient since we can use information from old samples for the policy update.
- Recent RL algorithms (ACER, Q-prop, IPG, etc.) reuse old samples to enhance sample efficiency.

Off-Policy Learning

- In off-policy learning, we store old samples in experience replay buffer $\mathbf{R}$.
- For example, DQN stores independent time samples in the buffer.
- ACER stores episodic samples in episodic replay buffer.
- For the policy update, off-policy RL algorithm randomly choose minibatch or episodic samples in the buffer.
- Off-policy learning enhances sample efficiency and usually achieves higher performance.

Contributions

- PPO has low sample-efficiency.
- We aim to reuse old sample batches for the policy update.
- However, older batches have larger IS weight and most samples in the batches are clipped.
- To overcome these drawbacks, we propose a new replay scheme :
  Adaptive Multi-Batch Experience Replay (AMBER)
- It adaptively selects the number of batches to avoid large batch average IS weight.

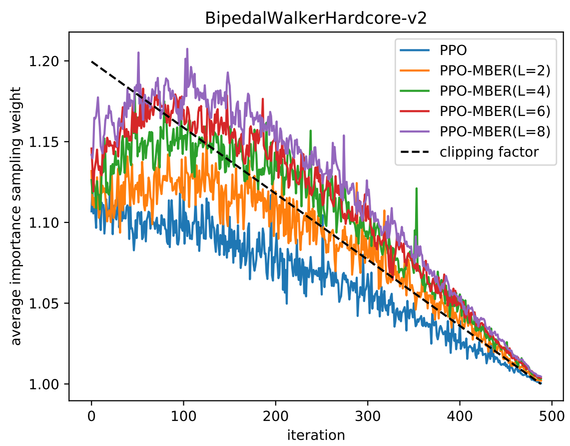

Figure 1: Average IS weight of BipedalWalkerHardcore.

Multi-Batch Experience Replay

- We consider multi-batch experience replay (MBER) that stores multiple previous batches in the replay buffer.
- At i-th iteration, $\mathbf{R}$ has L sample batches : $B_i, \cdots, B_{i-L+1}$.
- To compute PPO objective function from old samples, sample batch has estimated advantage $\hat{A}_t$, target value $\hat{V}_t$, statistics of policy distribution (μ_t, σ_t).
- $B_i = \{(s_{i,n}, a_{i,n}, \hat{A}_{i,n}, \hat{V}_{i,n}, \mu_{i,n}, \sigma_{i,n})\}, n = 0, \cdots, N-1.$

Multi-Batch Experience Replay

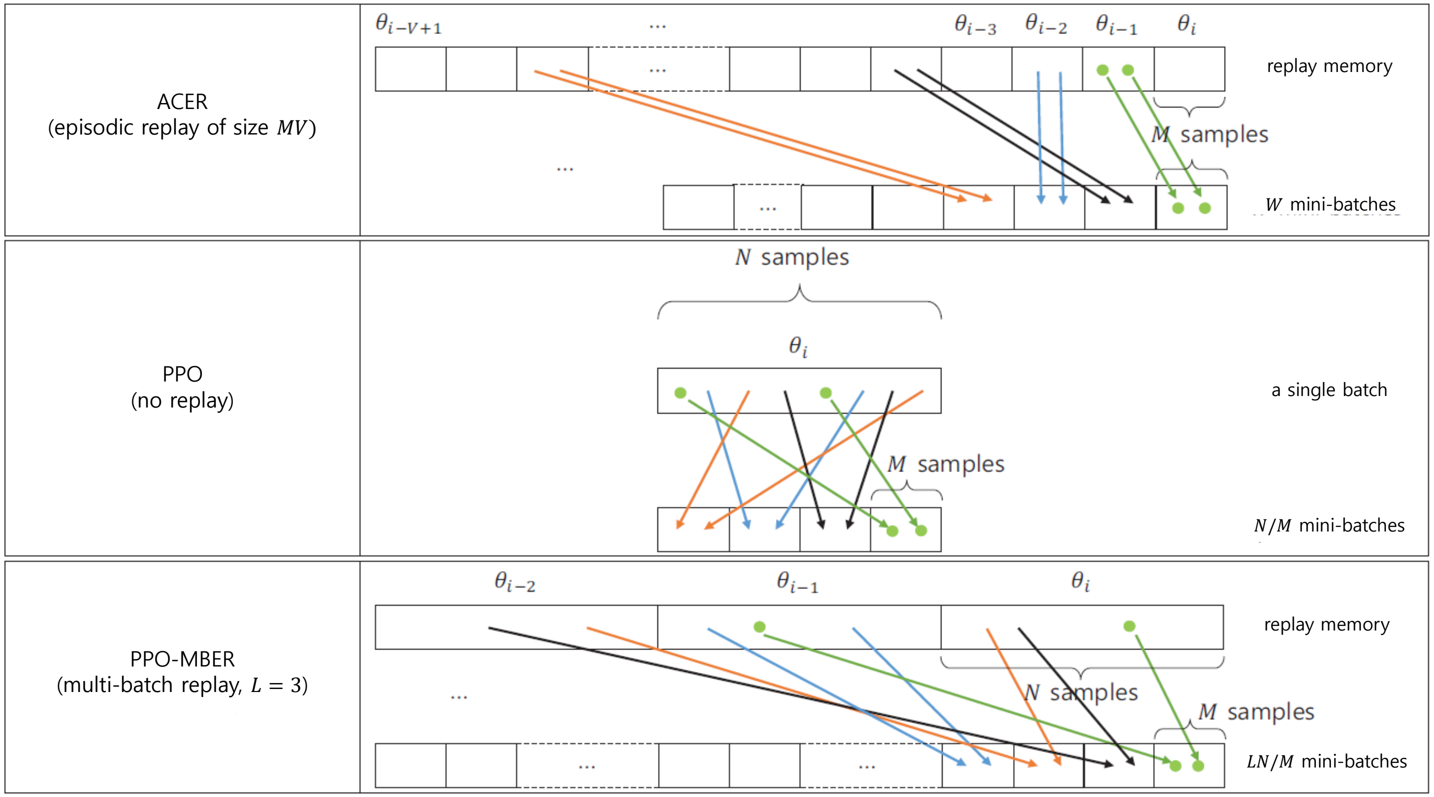

Figure 2: Batch construction of ACER, PPO, and PPO with the proposed MBER.

- We sample mini-batches from the replay and update the policy by the same epoch with PPO.

Main Problem

- Reusing old sample enhances sample efficiency, but the performance of MBER largely depends on the replay length L and action dimension d of task.
- To find the reason of performance fluctuation, we first define batch average IS weight as

$$R_{i,l} = \frac{1}{N}\sum_{n=0}^{N-1}\left(1 + abs\left(1 - \frac{\pi_{\theta_i}(a_{i-l,n}|s_{i-l,n})}{\pi_{\theta_{i-l}}(a_{i-l,n}|s_{i-l,n})}\right)\right) \quad (2)$$

- It represents the statistic difference between the current sample batch B_i and l-th previous old sample batch B_{i-l}.
- If $R_{i,l}$ is far from 1, they have large statistic difference and otherwise, they have similar statistics.

Main Problem

- Fig. 3. shows $R_{i,l}$ of several tasks (Pendulum, BipedalWalkerHardcore, Humanoid).
- Action dimension - Pendulum : 1, BipedalWalkerHardcore : 4, Humanoid : 17.
- Older sample batch has larger batch average IS weight.
- Batch average IS weight becomes larger as action dimension increases.
- It is natural because the policy independently products distribution of each action dimension.

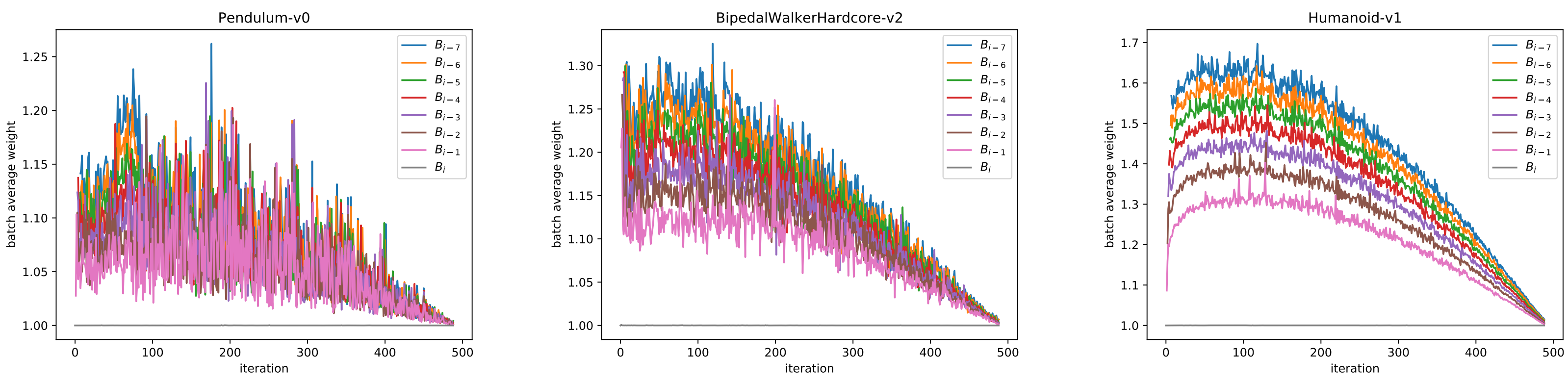

Figure 3: Batch average IS weight $R'_{i,l}$ $(l = 0, \cdots, 7)$ of Pendulum, BipedalWalkerHardcore, and Humanoid

Main Problem

- Large batch average IS weight enlarges the amount of clipped sample in PPO loss.

$$\hat{J}_{PPO}(\theta) = \frac{1}{M}\sum_{m=0}^{M-1} \min\{\rho_m \hat{A}_m, \mathrm{clip}_\epsilon(\rho_m)\hat{A}_m\}$$

- Clipped sample causes zero-gradient, so it is not used for the update.
- Then, most samples of high action-dimension tasks and old sample batches are not used for the update.
- It makes performance degradation when the replay length L or action-dimension d is too large.

Adaptive Batch Drop

- To solve the problem, we propose adaptive multi-batch experience replay (AMBER).
- AMBER drops some batches adaptively to avoid too much clipping in PPO loss.
- It only uses old sample batches in the buffer, which satisfy

$$R'_{i-l} < 1 + \epsilon_b, \tag{3}$$

  where ϵ_b is batch drop factor.
- It prevents that the amount of clipped samples becomes too large.
- Note that batch drop does not break sample distribution, which is important to learn the task.

Evaluation

- We evaluate the performance of our method on Mujoco tasks in OpenAI GYM.

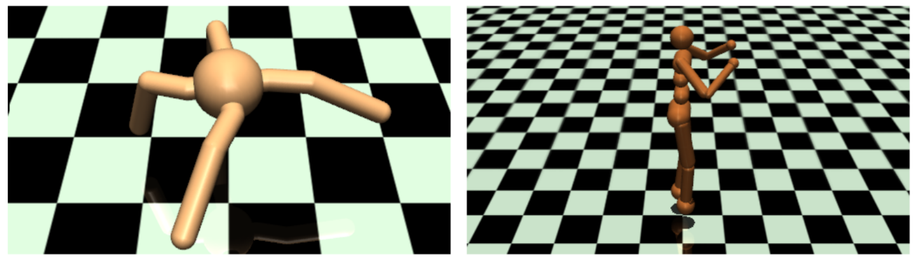

Figure 4: Mujoco tasks

- We compare 3 algorithms:
  - PPO : baseline algorithm
  - PPO-MBER : PPO with simple batch experience replay of various replay length L.
  - PPO-AMBER : PPO with adaptive batch drop.

Evaluation

- Compared with PPO, AMBER enhances the final performance on Mujoco tasks.
- AMBER consistently gets the highest performance for all tasks, whereas the performance of MBER fluctuates as L changes.

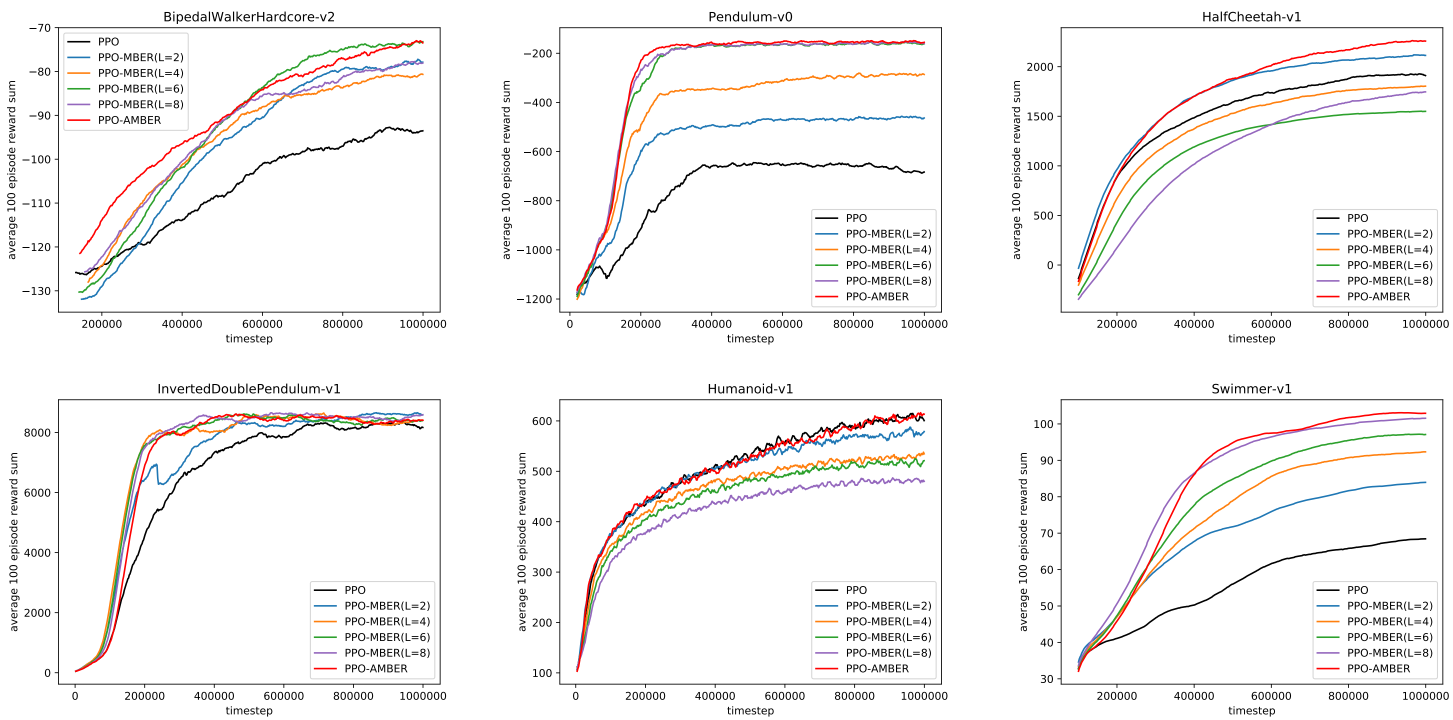

Figure 5: Performance comparison on Mujoco tasks

Ablation Study

- We provide ablation study about the clipping factor of PPO ϵ, and batch drop factor ϵ_b.
- Appropriate ϵ_b enhances sample efficiency without performance degradation by the clipping.
- In summary, $\epsilon = 0.4$ and $\epsilon_b = 0.25$ gets the highest performance.
- We provide other performance comparison with TRPO and ACER, PPO-AMBER has the best performance.

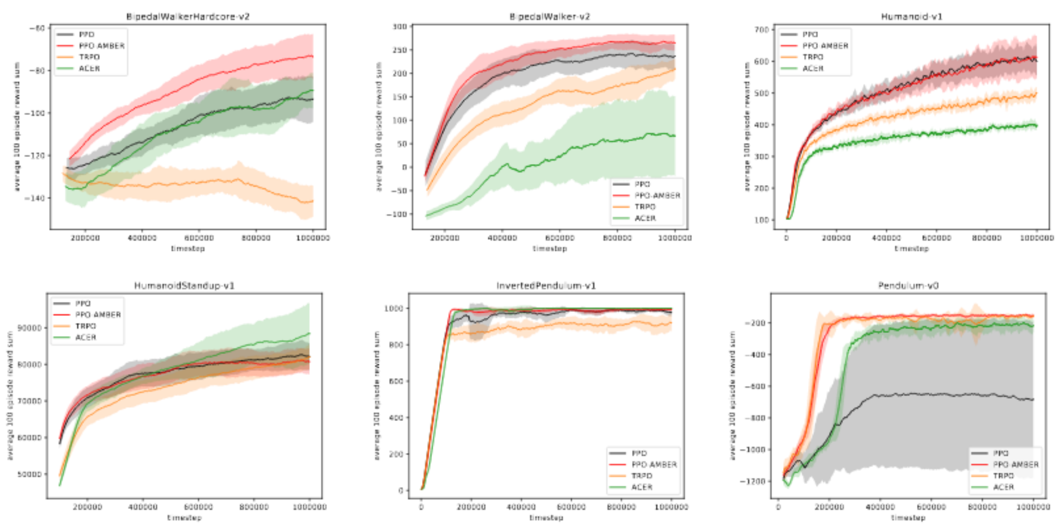

Figure 6: Performance comparison on Mujoco tasks

Further Discussion

- AMBER greatly enhances the performance for lower dimensional tasks, but it does not work for higher dimensional tasks.
- It is because higher dimensional tasks have large batch IS weight even for sample batch of previous iteration.
- Reducing learning rate helps reducing IS weight, but it is not much effective.
- Off-policy generalization in high action dimensional tasks will be future work.

Thank you !