Experience Replay Optimization

Daochen Zha, Kwei-Herng Lai, Kaixiong Zhou and Xia Hu

Department of Computer Science and Engineering, Texas A&M University

Emails: {daochen.zha, khla037, zkxiong, xiahu}@tamu.edu
What is Experience Replay

- **Definition**: A memory buffer that stores past transitions (experiences) which are replayed for later use.
- A key technique behind contemporary off-policy RL algorithms. It greatly stabilizes the training and improves the sample efficiency.
Replay Strategy Matters

- **Uniform Sampling**: Sample transitions in the memory with equal probabilities. Applied in most off-policy algorithms.

- **Prioritized Experience Replay [1]**: Prioritize the transitions with higher temporal differences (TD) errors. Improved performance for DQN on Atari environments.

- **Various Memory Size [2][3]**: Manage the size of the replay buffer.

- **Remember/Forget Experience [4]**: Selectively remember/forget experience can improve the performance.

Humans tend to replay the memories that will lead to the most rewarding future decisions.

We are motivated to use the feedback from the environment as a rewarding signal to adjust the replay strategy.
Sampling with Replay Policy

• A priority score function \( \phi(f_{B_i} | \theta^\phi) \in (0,1) \) where \( f_{B_i} \) are some features for a transition.

• Given the scores for all the transitions

\[
\lambda = \{ \phi(f_{B_i} | \theta^\phi) | B_i \in B \}.
\]

• Before feeding into standard RL training, we narrow down all the transitions to a subset of transitions \( B^S \):

\[
I \sim \text{Bernoulli}(\lambda),
\]

\[
B^S = \{ B_i | B_i \in B \land I_i = 1 \}.
\]

• Then \( B^S \) is used to update the agent with standard procedures.
Training with Policy Gradient

- The replay-reward is defined as the improvement of the cumulative reward:

\[ r^r = r^c_\pi - r^c_{\pi^r}. \]

where \( r^c_\pi \) and \( r^c_{\pi^r} \) are the recent cumulative rewards.

- By using the REINFORCE trick, we can calculate the gradient of the improvement \( J \) w.r.t \( \theta \phi \):

\[
\nabla_{\theta \phi} J = \nabla_{\theta \phi} \mathbb{E}_I[r^r] \\
= \mathbb{E}_I[r^r \nabla_{\theta \phi} \log P(I|\phi)].
\]

- Where \( \phi \) is the abbreviation for \( \phi(f_{B_i}|\theta \Phi) \).
Applying to Off-Policy Algorithms

**Algorithm 1** ERO enhanced DDPG

1. Initialize policy $\pi$, replay policy $\phi$, buffer $B$
2. for each iteration do
   3. for each timestep $t$ do
      4. Select action $a_t$ according to $\pi$ and state $s_t$
      5. Execute action $a_t$ and observe $s_{t+1}$ and $r_t$
      6. Store transition $(s_t, a_t, r_t, s_{t+1})$ into $B$
   7. if episode is done then
      8. Calculate the cumulative reward $r_c^\pi$
      9. if $r_c^\pi \neq null$ then
         10. $B^s = \text{UpdateReplayPolicy}(r_c^\pi, r_c^{\pi'}, B)$
   11. end if
   12. Set $r_c^\pi \leftarrow r_c^\pi$
   13. end if
3. end for
4. for each training step do
   5. Uniformly sample a batch $\{B^s_i\}$ from $B^s$
   6. Update the critic of $\pi$ with Eq. (9) and (10)
   7. Update the actor of $\pi$ with Eq. (11)
   8. Update target networks with Eq. (12) and (13)
   9. Update $\lambda$ for each transition in $\{B_i\}$
4. end for

**Algorithm 2** UpdateReplayPolicy

Input:
- Cumulative reward of current policy $r_c^\pi$
- Cumulative reward of previous policy $r_c^{\pi'}$
- Buffer $B$

Output:
- Sampled subset $B^s$
1. Calculate replay-reward based on Eq. (3)
2. for each replay updating step do
   3. Randomly sample a batch $\{B_i\}$ from $B$
   4. Update replay policy based on Eq. (8)
5. end for
6. Sample a subset $B^s$ from $B$ using Eq. (2)

Update Replay Policy
Sample New Transitions
Experimental Settings

• **Baselines:** Vanilla-DDPG, Proportional Prioritized Experience Replay (PER-pro), Rank-based PER (PER-rank).

• **Environments:** Continuous control tasks from OpenAI gym.

• **Implementation Details:**

  ➢ DDPG implementation in OpenAI baselines.

  ➢ For ERO, three features are used: the reward of the transition, the temporal difference (TD) error, and the current timestep.

  ➢ The replay policy is MLP with two hidden layers (64-64).
Effectiveness Evaluation

- ERO outperforms Vanilla-DDPG and rule-based replay strategy (PER-prop and PER-rank).
Efficiency Evaluation

- **ERO only requires slightly more computation than Vanilla-DDPG for replay policy update.**
Replay Policy Analysis

• The learned replay policy of ERO samples more transitions with low TD errors in HalfCheetah (More studies are needed to understand this aspect in the future work).
• ERO samples more recent transitions than Vanilla-DDPG.
Conclusions

• Formulate experience replay as a learning problem.

• Propose ERO, a general framework for effective and efficient use of replay memory.

• Conducted experiments on 8 continuous control tasks from OpenAI Gym demonstrate the effectiveness of ERO.
Acknowledgement

• DATA Lab and collaborators

Data Analytics at Texas A&M (DATA Lab)

• National Science Foundation (NSF)
• Everyone attending the talk