A Supplementary for Preliminaries

A.1 Supplementary for Revisiting Existing Experience Replay Methods

CER (Competitive Experience Replay). The limitations of exploration are addressed by introducing an approach called Competitive Experience Replay (CER). This technique attempts to introduce competition between two agents that try to learn the same task to emphasize exploration. Intuitively, Agent A (the Agent ultimately used for evaluation) is penalized for accessing states that are also accessed by rival Agent B, while B is rewarded for accessing states discovered by A. This approach maintains the rewards for the original task, allowing exploration to favor the behavior that is most appropriate for accomplishing the task goal. Two agents A, B learn the same task, and require A to be punished for being similar to B, and B to be rewarded for being similar to A, so as to restrict A and B to have different strategies. Sample collection of A, B was completed independently, and each carried out its own strategy (HER) and formed a new experience pool.

In the specific implementation, the reward control of the agent is adjusted. In any mini-batch, if the two states satisfy $|sAi - sBi| < \delta$, then the reward of A is reduced by 1 and the reward of B is increased by 1. Given a game with N agents with policies parameterized by $\theta = \{\theta_1, ..., \theta_N\}$, and let $\pi = \{\pi_1, ..., \pi_N\}$ be the set of all agent policies. $g = [g_1, ..., g_N]$ represents the concatenation of each agent s goal, $s = [s_1, ..., s_N]$ the concatenated states, and $a = [a_1, ..., a_N]$ the concatenated actions. With this notation, we can write the gradient of the expected return for agent i, $J(\theta_i) = \mathbb{E}[R_i]$ as:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{\pi}[\nabla_{\theta_i} \log \pi_i(a_i|s_i, g_i) Q_i^{\pi}(s, a, g)] \tag{1}$$

With deterministic policies $\mu = \{\mu_1, ..., \mu_N\}$, the gradient becomes:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{\mu}[\nabla_{\theta_i} \mu_i(s_i, g_i) \nabla_{a_i} Q_i^{\mu}(s, a, g)|_{a_i = \mu_i(s_i, g_i)}] \tag{2}$$

The centralized action-value function Q_i^{μ}, which estimates the expected return for agent i, is updated as: $\mathcal{L}(\theta_i) = \mathbb{E}_{s,a,r,s'}[(Q_i^{\mu}(s, a, g) - y)^2]$, $y = r_i + \gamma Q_i^{\mu'}(s', a_1', ..., a_N' g)|_{a_j' = \mu_j'(s_i)'}$, where $\mu' = \{\mu_{\theta_1'}, \cdots, \mu_{\theta_N'}$ is the set of target policies with delayed parameters θ_i'. In practice people usually soft update it as $\theta_i' \leftarrow \tau\theta_i + (1 - \tau)\theta_i'$, where τ is a Polyak coefficient.

HER(Hindsight Experience Replay) The experience replay technique against sparse reward is aimed at a special scenario: there are only sparse binary rewards in the environment. The rewards are used to indicate whether the task is completed or not, that is, the intelligence receives a reward only when the task is completed. The HER algorithm proposed in the paper (reference) can learn good strategies when the rewards are binary and sparse. The idea is that after an episode $s_0, s_1, ..., s_T$, each transition is stored, and each experience contains not only the initial set of goals, but also some reset goals. The reason for this is that a object can affect only the actions of an agent, but does not affect the dynamic changes of the environment. Therefore, if you use the off-policy algorithm (where the learned policies are inconsistent with the policies being executed), you can change a target in the trajectory to any other target. HER gives several methods for target selection:

Based on final state: Select the target $m(s_T)$ corresponding to the last state s_T reached by each episode as the replay target.

Based on future transition: Select the target implemented for each of the k states from transitions occurring after the current transition as the replay target.

Based on episode: Randomly select the target corresponding to k states in the current episode as the replay target.

Random: Select the target corresponding to k states from the states encountered in the current whole training process as the replay target.

A.2 Deep Q-Networks(DQN)

Deep Q-Network (DQN) [3] is a model-free RL algorithm for discrete action spaces. In this paper, we only summarize it informally, for more details see [3]. A neural network Q approximates Q^* is maintained in DQN. $\pi_Q(s) = arrg\max_{a \in A} Q(s, a)$ denotes a greedy policy w.r.t. Q. An $\varepsilon - greedy$ strategy w.r.t. Q is a policy taking a random action with probability ε (uniform selecting from $\mathcal{A}$) and taking the action $\pi_Q(s)$ with probability $1 - \varepsilon$.

We use the $\varepsilon - greedy$ policy w.r.t the current approximation of the action-value function Q to generate episodes in the process of training. The transition tuples (s_t, a_t, r_t, s_{t+1}) met during training are stored in the replay buffer. The generation of new episodes is interspersed with neural network training. Mini-batch gradient descent on the loss $\mathcal{L}$ is used to train the network which encourages the approximated Q-function to satisfy the Bellman equation: $\mathcal{L} = \mathbb{E}(Q(s_t, a_t) - y_t)^2$, where $y_t = r_t + \max_{a' \in A} Q(s, a')$ and the tuples (s_t, a_t, r_t, s_{t+1}) are sampled from the replay buffer.

To make the optimization process more stable, the targets y_t is usually computed with a separate target network that changes at a slower rate than the main network. A common practice is to periodically renew the weights of the target network as main network (e.g., [3]) or use the Polyak -averaged version of the main network instead.

A.3 Deep deterministic policy gradient

Deep deterministic policy gradient (DDPG) (Lillicrap et al. 2016) is an off-policy actor-critic algorithm that extends the deterministic policy gradient (DPG) algorithm (Silver et al. 2014) by using deep neural networks as function approximators. DDPG contains two neural networks, i.e., the value-network (a.k.a. critic) that outputs $Q(s, a; \theta)$, and the policy-network (a.k.a. actor) that selects actions based on given states $a = \mu(s, \varnothing)$. DDPG alternately updates these two networks. The value-network is trained using the loss function similar to DQN:

$$L(\theta) = \mathbb{E}_{(s,a,r,s')\sim U(\mathcal{D})}[y - Q(s, a; \theta)^2] \tag{3}$$

where $y = r + \gamma Q(s', \mu(s'; \varnothing^-); \theta^-)$, $Q(s, a; \theta^-)$ and $\mu(s; \varnothing^-)$ are the target networks. The policy-network is trained to output an action a that maximizes the value pre dicted by the value-network:

$$\begin{aligned} \nabla_{\varnothing} J(\mu) &\approx \nabla_{\varnothing} \mathbb{E}_{s\sim U(\mathcal{D})}[Q(s, a|\theta)_{a=\mu(s|\varnothing)}] \\ &= \nabla_{\varnothing} \mathbb{E}_{s\sim U(\mathcal{D})}[\nabla_a Q(s, a|\theta)_{a=\mu(s|\varnothing)} \nabla_{\varnothing} \mu(s|\varnothing)] \end{aligned} \tag{4}$$

B supplementary for related work

However, in the off-policy RL algorithm, Hindsight Experience Replay (HER) enables agents have access to learning from failures, considering the achieved state of the failed experience as a pseudo-goal. As not all failed experiences are equally useful for different stages of learning, repeating all experiences or uniform samples is not necessarily effective, but leads to data duplication and increases the amount of calculation. Curriculum-guided HER is proposed [2], which adaptively selects failed experiences for replay based on the proximity to the real target and the curiosity to explore different pseudo-targets, thus avoiding the reuse of the duplicate data experience data to some extent. Preferential experience replay (PER) is proposed from the perspective of optimal loss function, where uniformly sampled loss functions with the same desired gradient can replace the loss function of non-uniformly sampled data. Aiming at the problem of sparse reward, HER [1] combines incremental learning with experience replay. Based on universal value function approximators, it is applied to multi-objective scenarios. When replaying each episode, a new goal is substituted for the original goal accomplished by the agents, and this new goal have achieved in this episode, which is somewhat similar to the hindsight.

References

[1] Eser Aygün, Ankit Anand, Laurent Orseau, Xavier Glorot, Stephen M Mcaleer, Vlad Firoiu, Lei M Zhang, Doina Precup, and Shibl Mourad. 2022. Proving Theorems using Incremental Learning and Hindsight Experience Replay. In *International Conference on Machine Learning*. PMLR, 1198–1210.

[2] Meng Fang, Tianyi Zhou, Yali Du, Lei Han, and Zhengyou Zhang. 2019. Curriculum-guided hindsight experience replay. *Advances in neural information processing systems* 32 (2019).

[3] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518 (2015), 529–533.