1707.01495 - Hindsight Experience Replay

Hindsight Experience Replay

Marcin Andrychowicz[*], **Filip Wolski**, **Alex Ray**, **Jonas Schneider**, **Rachel Fong**,
Peter Welinder, **Bob McGrew**, **Josh Tobin**, **Pieter Abbeel**[†], **Wojciech Zaremba**[†]
OpenAI

- **Yunqiu Xu**
- Focus on sparse reward:

  - HER: allows sample-efficient learning from rewards
  - Reward can be sparse and binary
  - Do not need complicated reward engineering
  - Experiment on robot arm manipulating
- Some references, implementations, and further readings

  - DanielTakeshi's notes
  - Learning from mistakes with Hindsight Experience Replay
  - Two minute papers
  - OpenAI's implementation
  - minsangkim142's implementation
  - OpenAI posted a further blog "Ingredients for Robots Research", which contains some possible improvements of HER
  - 1709.10089 - Overcoming Exploration in Reinforcement Learning with Demonstrations
  - 1712.00948 - Hierarchical Actor-Critic

---

1. Introduction

- Challenges: reward engineering
  - Sparse reward is hard to deal
  - Designing reward function is complex

- e-greedy based exploration is inefficient
- Insight from human learning: 人类可以从不好的结果中吸取教训, 而RL只能根据得到的 reward学习
- 从另一份工作(Universal value function approximators, Schaul 2015)中得到的灵感: 每 个episode都设定不同的目标 (**疑问, 是否类似课程学习这样循序渐进的**)
- Hindsight Experience Replay:
  - Suitable with off-policy RL (e.g. DQN)
  - Assumption: 可以将每个state设置成目标

2. Background

- DDPG
  - AC-like DQN for continuous action spaces
  - Actor:
    - $\pi : S \to A$
    - Target policy to choose action
    - Try to maximize action value with respect to policy's parameters
  - Critic:
    - $Q^\pi : S \times A \to R$
    - Action-value function to evaluate Q value
    - Try to minimize Bellman error
  - Learning: update C using Bellman, update A using PG
- UVFA: Universal Value Function Approximators
  - There are more than one goal we may try to achieve
  - Learning: for each episode sample a state-goal pair, so the "goal" stay fixed in this episode

3. HER

3. Problem Setup

- Key idea: replay episodes with a different goal.
- Assumption:

  - Need multiple goals in an environment, for each state, we can get specific reward
  - 注意这里和sparse reward不冲突, 不更改reward function, 只是对于未完成的episode换了下final state, 然后计算reward的时候判断能否到达这个新的final state
- Store an episode $(s_1, s_2, \ldots, s_T)$ in replay buffer twice:

  - One is running with original goal
  - Another is with "final state" in this episode: if the agent still fails at s_T, then set this state s_T as goal for this episode
- Simplest version

  - Store both final state s_T and original goal g per episode
  - Shape a mapping function $m(s_T)$ to represent state-goal pair

    1. Run your policy and store everything you observe (state **and goal**, action, reward, next state **and goal**) tuple into an experience buffer.

    2. Sample **K** goals from the states visited in the episode obtained at step 1, and for each goal store (state **and sampled goal**, action, **reward with respect to the sampled goal**, next state **and sampled goal**) tuple into the buffer.

    3. Repeat step 1–2 N times.

    4. Sample M number of experience tuples (batch) from the buffer and train the network with the said batch experience. (Do this B times)

3.2 Algorithm

Algorithm 1 Hindsight Experience Replay (HER)

Given:
- an off-policy RL algorithm $\mathbb{A}$,   ▷ e.g. DQN, DDPG, NAF, SDQN
- a strategy $\mathbb{S}$ for sampling goals for replay,   ▷ e.g. $\mathbb{S}(s_0, \ldots, s_T) = m(s_T)$
- a reward function $r : \mathcal{S} \times \mathcal{A} \times \mathcal{G} \to \mathbb{R}$.   ▷ e.g. $r(s, a, g) = -[f_g(s) = 0]$

Initialize $\mathbb{A}$ ▷ e.g. initialize neural networks
Initialize replay buffer R
for episode $= 1, M$ **do**
    Sample a goal g and an initial state s_0.
    for $t = 0, T - 1$ **do**
        Sample an action a_t using the behavioral policy from $\mathbb{A}$:
             $a_t \leftarrow \pi_b(s_t || g)$ ▷ $||$ denotes concatenation
        Execute the action a_t and observe a new state s_{t+1}
    end for
    for $t = 0, T - 1$ **do**
         $r_t := r(s_t, a_t, g)$
        Store the transition $(s_t || g, a_t, r_t, s_{t+1} || g)$ in R ▷ standard experience replay
        Sample a set of additional goals for replay $G := \mathbb{S}(\textbf{current episode})$
        for $g' \in G$ **do**
             $r' := r(s_t, a_t, g')$
            Store the transition $(s_t || g', a_t, r', s_{t+1} || g')$ in R ▷ HER
        end for
    end for
    for $t = 1, N$ **do**
        Sample a minibatch B from the replay buffer R
        Perform one step of optimization using $\mathbb{A}$ and minibatch B
    end for
end for

3.3 Some Code Details

- 第一个内循环用于构建 goals replay G
  - 正常跑算法 A, 并把得到的transition $(s_t, a_t, r_t, s_{t+1}, g)$ 存入 G
  - 注意这里 transition 和一般的 transition 有所不同, 多了一个 g
  - g 即为任务预设的目标状态, 在这个循环里是不变的 `g = np.copy(env.target)`
- 第二个内循环用于构建 experience replay R
  - 首先使用map function, 获取t时刻的reward r_t, 在具体实现过程中, 就直接从G里面**按顺序取出**一个transition进行改造, 即为初始 g 下的transition

```
s, a, r, s_n, g = episode_experience[t]
inputs = np.concatenate([s,g],axis = -1)
new_inputs = np.concatenate([s_n,g],axis = -1)
buff.add(np.reshape(np.array([inputs,a,r,new_inputs]),[1,4]))
```

  - 注意这里将 $s_t || g$ 表示为这两个状态的连接, 后同

- 接下来在这里加入HER循环, 从 G 中**随机选取**一个transition, 将其 s_{t+1} 作为该 transition的goal, 存入 R. **注意这里因为goal state更换了, 计算 r_n 时为计算是否到达了这个新目标, 到达了为0, 反之为-1**

```python
for k in range(K):
    # Sample a transition from G, set its final state as g'
    future = np.random.randint(t, size)
    _, _, _, g_n, _ = episode_experience[future]
    # s||g', s_n||g'
    inputs = np.concatenate([s,g_n],axis = -1)
    new_inputs = np.concatenate([s_n, g_n],axis = -1)
    # Compute r' using reward function
    final = np.sum(np.array(s_n) == np.array(g_n)) == size
    if shaped_reward:
        r_n = 0 if final else -np.sum(np.square(np.array(s_n) == np.array(g_n)))
    else:
        r_n = 0 if final else -1
    # Store hindsight transition
    buff.add(np.reshape(np.array([inputs,a,r_n,new_inputs]),[1,4]))
```

- 第三个内循环就是正常的 DQN, 从 R 中选取minibatch进行学习, 注意需要对原有网络进行改造, 因为输入维度发生了变化
- 可以看出HER并未对reward function进行改造, 只是用一些可能对学习有利的数据扩增了experience replay, 从而缓解sparse reward问题

4. Experiment

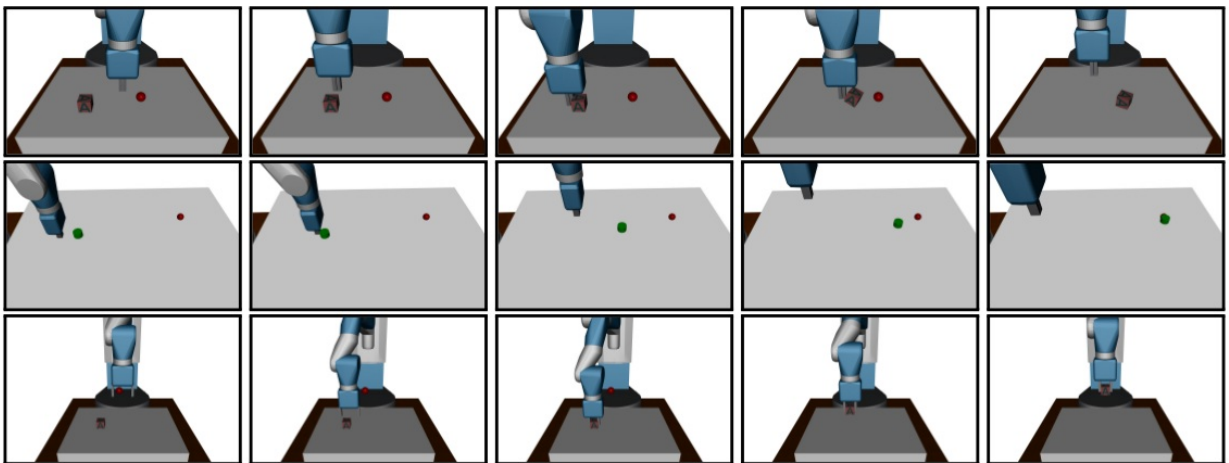- Robot arm manipulating tasks:

Figure 2: Different tasks: *pushing* (top row), *sliding* (middle row) and *pick-and-place* (bottom row). The red ball denotes the goal position.
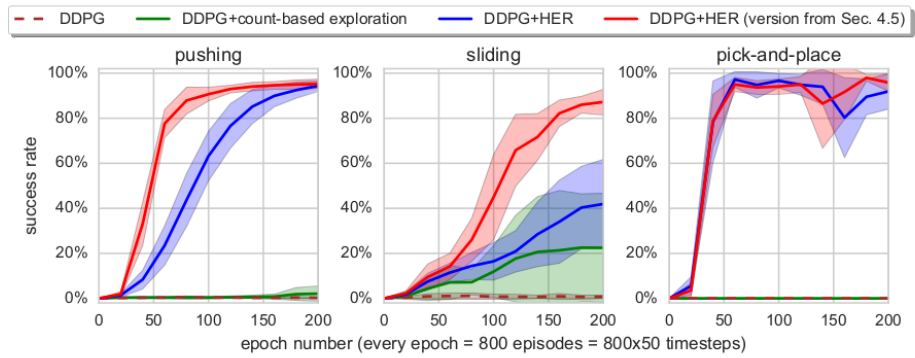
- Does HER improve performance
  - Multiple goals

Figure 3: Learning curves for multi-goal setup. An episode is considered successful if the distance between the object and the goal at the end of the episode is less than 7cm for pushing and pick-and-place and less than 20cm for sliding. The results are averaged across 5 random seeds and shaded areas represent one standard deviation. The red curves correspond to the `future` strategy with $k = 4$ from Sec. 4.5 while the blue one corresponds to the `final` strategy.
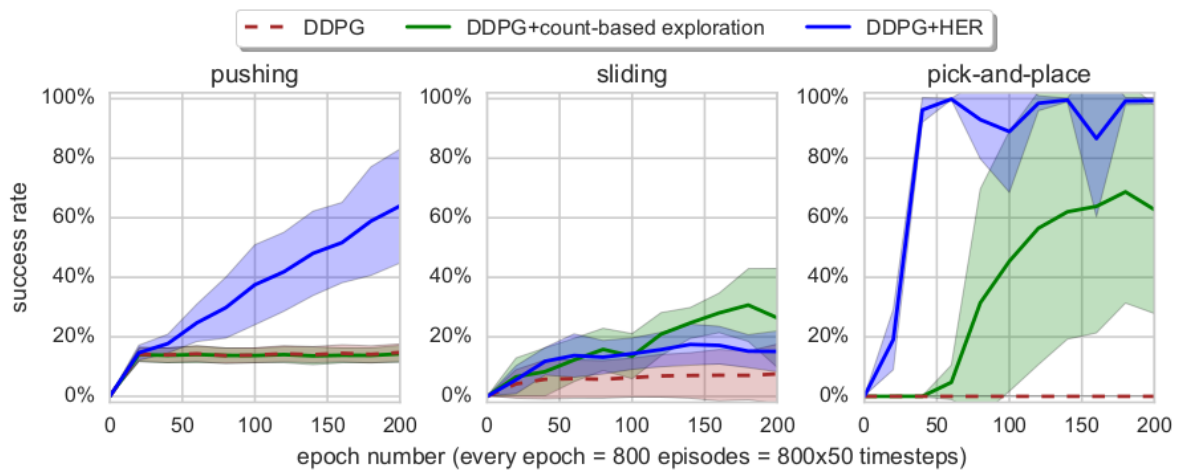
○ Only one goal

Figure 4: Learning curves for the single-goal case.

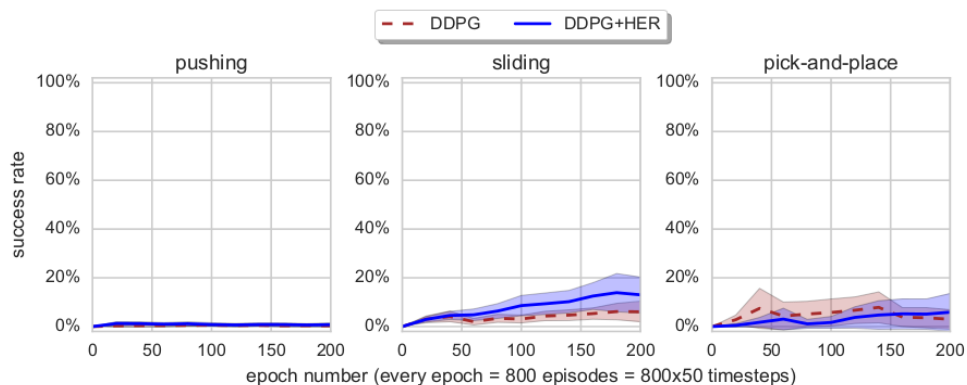• How does HER interact with reward shaping (not only binary)

Figure 5: Learning curves for the shaped reward $r(s, a, g) = -|g - s'_{\mathbf{object}}|^2$ (it performed best among the shaped rewards we have tried). Both algorithms fail on all tasks.

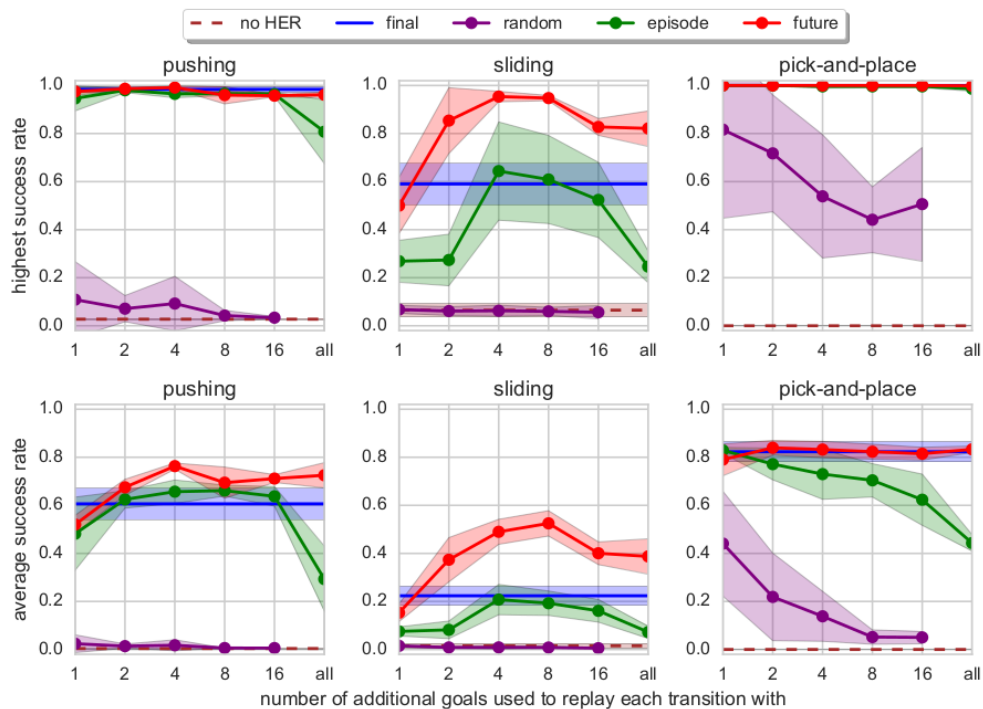• How many goals should we replay each trajectory with and how to choose them

Figure 6: Ablation study of different strategies for choosing additional goals for replay. The top row shows the highest (across the training epochs) test performance and the bottom row shows the average test performance across all training epochs. On the right top plot the curves for `final`, `episode` and `future` coincide as all these strategies achieve perfect performance on this task.

5. Summary

- HER:
  - Try to handle sparse reward
  - If the original goal can not be achieved in this episode, set final state as goal
  - Suitable for off-policy method: e.g. DQN / DDPG / Rainbow
- Future work from OpenAI's further blog:

Future work	Description
Automatic hindsight goal creation	现在的工作是比较简单但低效的做法, 即先跑一轮内循环构建goal replay, 然后再从中选取, 未来工作可尝试可学习的自动生成方法

Future work	Description
HER + HRL	目前已有基于HER进行HRL的工作 (1712.00948 - Hierarchical Actor-Critic), 本工作设置了有层次的goal并应用HER, 未来工作可尝试应用HER于higher-level policy选择的动作. 一个例子, 我们之前的假定是在每个state够可以得到reward, 这里可以将假定修改为: 若高层指定低层实现目标A, 但底层终止于状态B, 这时可以假定高层指定底层实现目标B
Faster information propagation	经典的构建target net的方法效率不高, 未来工作可以尝试其他稳定学习的方法
On-policy HER	现在HER基于off-policy的算法, 可尝试on-policy, 如结合PPO. 相关方向工作 1711.06006 - Hindsight policy gradients
HER与其他工作结合	Rainbow, reverse curriculum learning, DQfD
Unbiased HER	-
Richer value functions	-
HER + multi-step returns	-
RL with very frequent actions	-