Hindsight Experience Replay

NIPS 2017
OpenAI

Dealing with sparse rewards

강동구
ehdrndd@gmail.com

What is problem?

Sparse reward

standard solution : reward shaping.   but need domain knowledge

가령, 하키를 배운다고 했을때,

puck을 쳐서, 골대의 바깥쪽 오른쪽 그물을 맞혔다고 상상해보자.

RL이라면, 방금 하키를 쳤던 모든 과정(action)을 안좋게 평가하므로 이것으로부터 배우는바가 적다.(or Nothing)

하지만 만약 골대가 조금만 오른쪽에 있었다면, 방금 action sequence는 매우 좋은 샷이었다.

(Main idea)

re-examine fail trajectory with different goal - while this trajectory may not help us learn how to achieve the state g, but how to achieve the state s_T

"the real problem is not in lack of diversity of states being visited, rather it is simply impractical to explore such a large state space"

Why HER is good?

1. Learning possible if the reward signal is sparse and binary(-1 or 0)
2. No need reward 엔지니어링.
3. Sample efficiency
4. Any off-policy RL 알고리즘과 같이 쓸 수 있음
5. Multi goal

가정 :

1. goal이 agent의 (특정 or terminal) state
2. fully observable environment

Background

1. DQN : model-free, discrete action

$$\mathcal{L} = \mathbb{E}\left(Q(s_t, a_t) - y_t\right)^2 \quad y_t = r_t + \gamma \max_{a' \in \mathcal{A}} Q(s_{t+1}, a')$$

2. DDPG : model-free, continuous action

$$\mathcal{L}_a = -\mathbb{E}_s Q(s, \pi(s)) \quad y_t = r_t + \gamma Q(s_{t+1}, \pi(s_{t+1}))$$

3. UVFA : 여러개의 goal 해결 가능한 DQN의 확장판.DDPG에도 적용가능

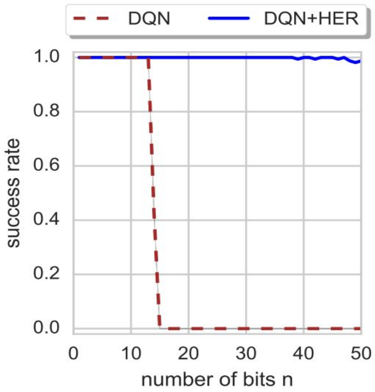
$$Q^\pi(s_t, a_t, g) = \mathbb{E}[R_t | s_t, a_t, g]$$

$\pi : \mathcal{S} \times \mathcal{G} \to \mathcal{A}$ and gets the reward $r_t = r_g(s_t, a_t)$

Environment

1. BitFilp

```
Initial State:  [0 0 0 0 0 1 1 0 0 1 1 1 1 1 1]
Goal:   [1 0 0 0 0 0 0 0 1 1 1 1 0 0 0]
State at step 0: [1 0 0 0 0 1 1 0 0 1 1 1 1 1 1]
State at step 1: [1 0 0 0 0 1 1 0 1 1 1 1 1 1 1]
State at step 2: [1 0 0 0 0 1 1 0 1 1 1 1 0 1 1]
State at step 3: [1 0 0 0 0 0 1 0 1 1 1 1 0 1 1]
State at step 4: [1 0 0 0 0 0 1 0 1 1 1 1 0 0 1]
State at step 5: [1 0 0 0 0 0 1 0 1 1 1 1 0 0 0]
State at step 6: [1 0 0 0 0 0 0 0 1 1 1 1 0 0 0]
Success!
Press enter...
```



with shaped reward

$$r_g(s,a) = -||s-g||^2$$

without shaped reward
: goal이 에피소드상의
state에 없을시 reward는
계속 -1.

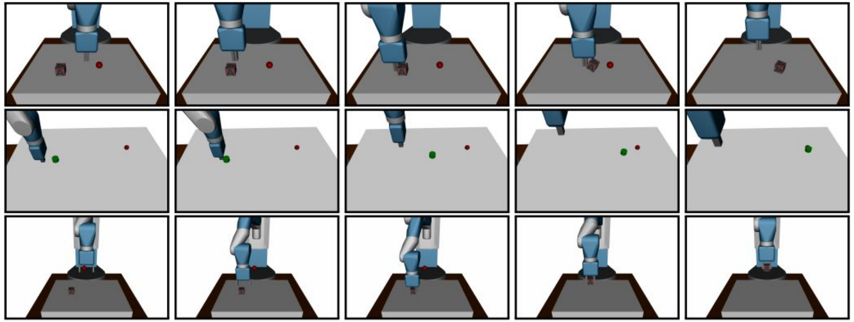
re-examine this trajectory with a different goal — while this trajectory may not help us learn how to achieve the state g, it definitely tells us something about how to achieve the state s_T. This information

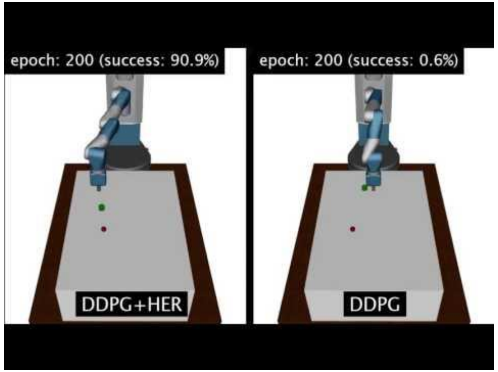
2. Robot Arm Manipulation(3 task)

pushing

sliding

pick-and-place





Environment

7-DOF Fetch Robotics arm which has a two-fingered parallel gripper

State: angles,velocities of all robot joints as well as positions, rotations and velocities (linear and angular) of all objects

pick-and-place의 경우 박스가 잡혔는지에 대한 상태(0 or 1)가 추가되고, 학습 에피소드의 절반을 잡은상태에서 시작

Goals: desired position of the object with some fixed tolerance of ε. i.e. $f_g(s) = [|g - s_{\mathbf{object}}| \le \epsilon]$

Rewards: binary and sparse reward $r(s, a, g) = -[f_g(s') = 0]$

State-goal distribution: For all tasks the initial position of the gripper is fixed, while the initial position of the object and the target are randomized

Observations: …

Actions: …

Strategy: ...

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 $G := \mathbb{S}($**current episode**$)$
    for $g' \in G$ **do**
      $r' := r(s_t, a_t, g')$ ← don't use env.step()
      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

$$f_g(s) = [|g - s_{\textbf{object}}| \leq \epsilon]$$

대괄호 안이 True이면 0

after experiencing some episode s0, s1, . . . , sT we store in the replay buffer every transition st → st+1 not only with the original goal used for this episode but also with a subset of other goals

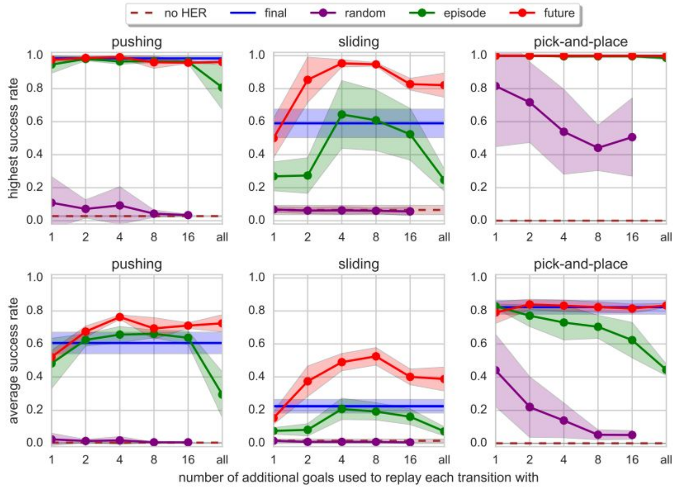
Strategy $\mathbb{S}(s_0, \ldots, s_T) = m(s_T)$

final : the final state of the environment

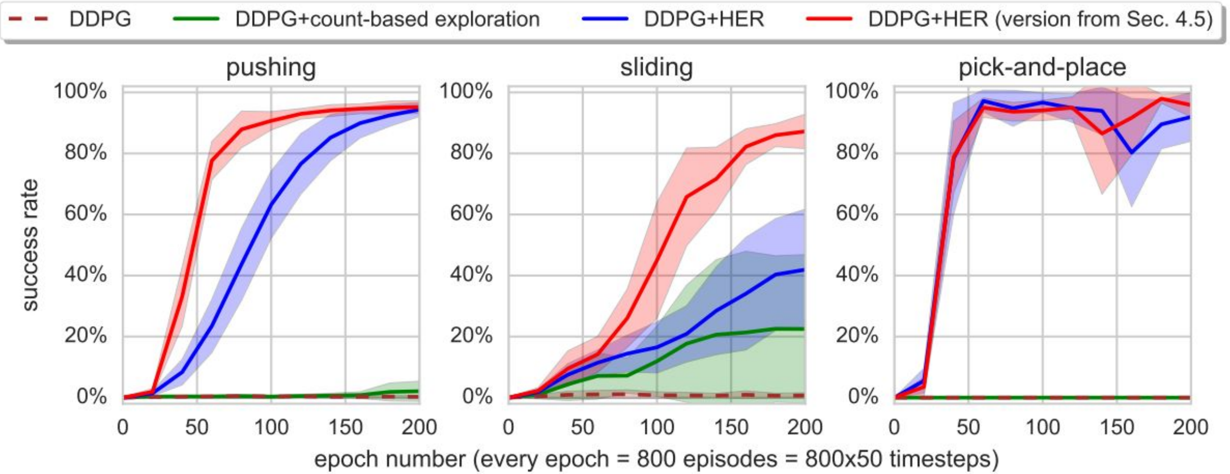
future : replay with k random states which come from the same episode as the transition being replayed and were observed after it

episode : replay with k random states coming from the same episode as the transition being replayed

random : replay with k random states encountered so far in the whole training procedure



Does HER improve performance?



red : “future” with k = 4
blue : “final”

Does HER improve performance even if there is only one goal we care about?

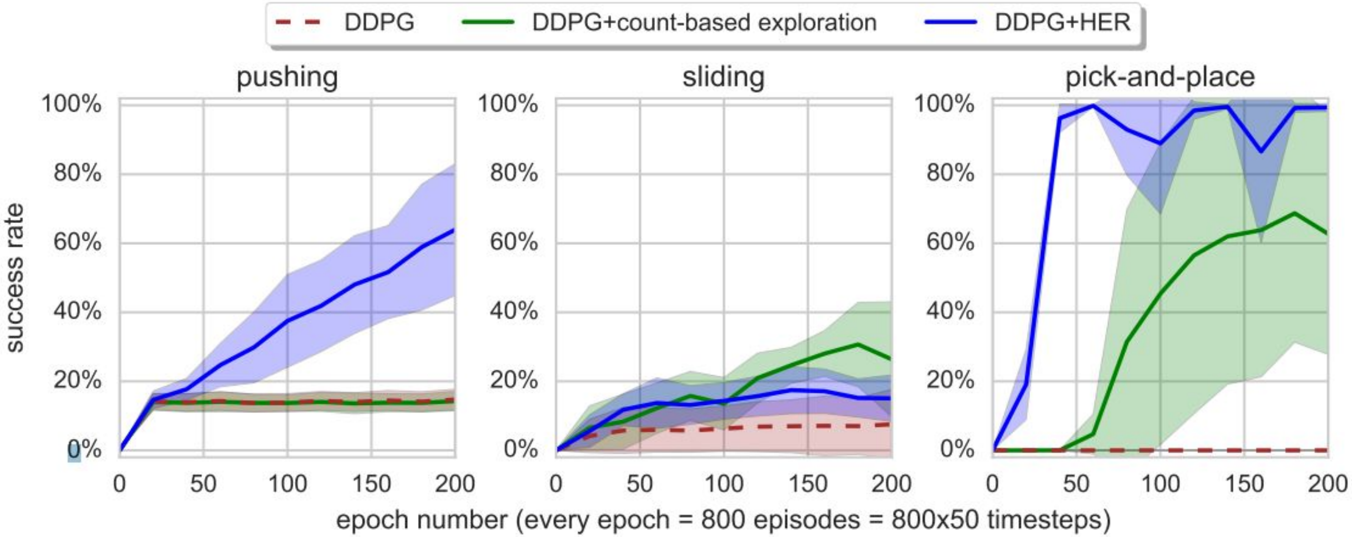


Figure 4: Learning curves for the single-goal case.

goal state is identical in all epsidoes

How does HER interact with reward shaping?

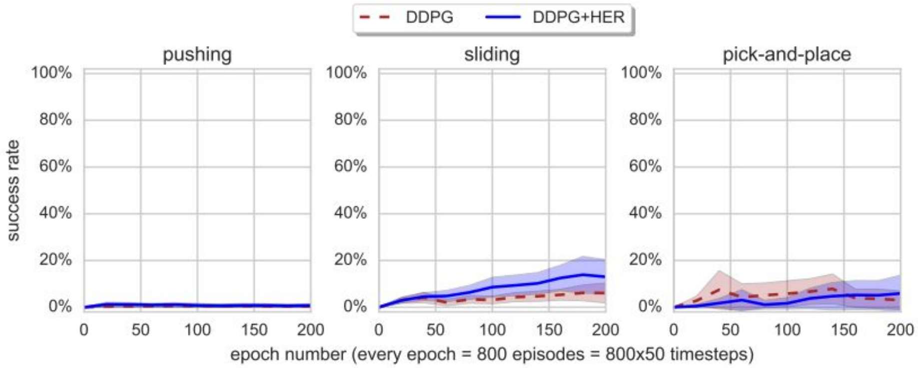


Figure 5: Learning curves for the shaped reward $r(s, a, g) = -|g - s'_{\mathbf{object}}|^2$ (it performed best among the shaped rewards we have tried). Both algorithms fail on all tasks.

reward shaping 안하는게 낫다
Why?

1. reward와 success의 괴리감
2. reward에 적절하지 못한 행동(exploration)을 막음. -> 정확히 하지 못할거면, 하지마라!

주관적 결론

1. HER은 transition의 양과 의미를 늘렸다.
2. 전략중 “final”과 “future”는 마치, 5초뒤의 상황을 지금 어떤 action을 하면 생기는지 알려주는 꼴이다.

의의

: 복잡한 task를 sparse, binary reward에서 푼 first case.(As far as we know)