Review for HW4

10-403 RECITATION (4/10/20)

* 2D PUSHER ENVIRONMENT SLIDES ARE INSPIRED BY NICHOLAY TOPIN, PREVIOUS TA

Why DDPG?

Challenge

- DQN can only handle discrete and low-dimensional action spaces while it solves problems with high-dimensional observation space

Goal

- Combine the ideas of DQN with Deterministic Policy Gradient (DPG) to extend to the continuous action domain

$$\begin{aligned}\nabla_{\theta^\mu} J &\approx \mathbb{E}_{s_t\sim\rho^\beta}\left[\nabla_{\theta^\mu} Q(s,a|\theta^Q)|_{s=s_t,a=\mu(s_t|\theta^\mu)}\right]\\ &= \mathbb{E}_{s_t\sim\rho^\beta}\left[\nabla_a Q(s,a|\theta^Q)|_{s=s_t,a=\mu(s_t)}\nabla_{\theta_\mu}\mu(s|\theta^\mu)|_{s=s_t}\right]\end{aligned}$$

DDPG

Key Ideas

- Replay memory (from DQN)
- Target network (from DQN)
- **Soft target updates**

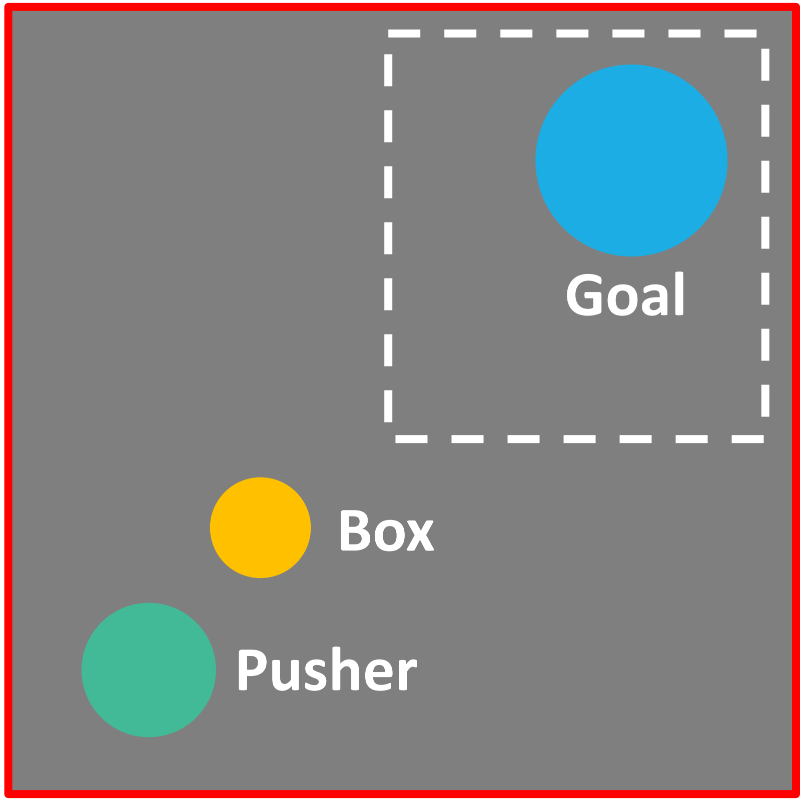
$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}$$

$$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}$$

- **Batch normalization (applied to input of every layer)**
- **Exploration policy with a noise**

$$\mu'(s_t) = \mu(s_t|\theta_t^{\mu}) + \mathcal{N}$$

2D Pusher Environment



State

- $(X_g, Y_g, X_b, Y_b, X_p, Y_p)$

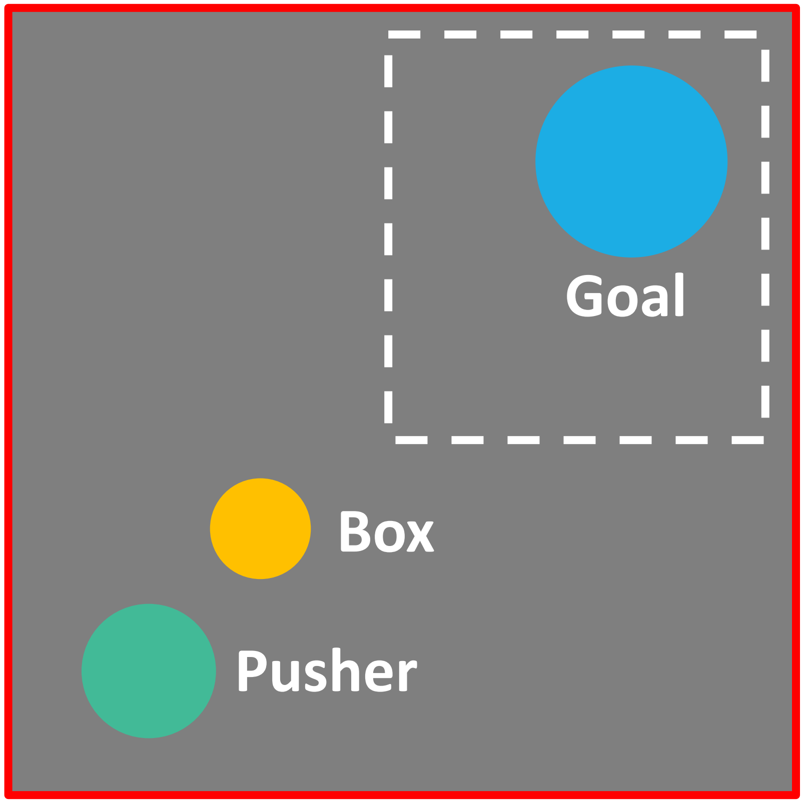
Initial State

- The location of box and pusher are fixed
- The goal location varies

Action

- (X_{move}, Y_{move})

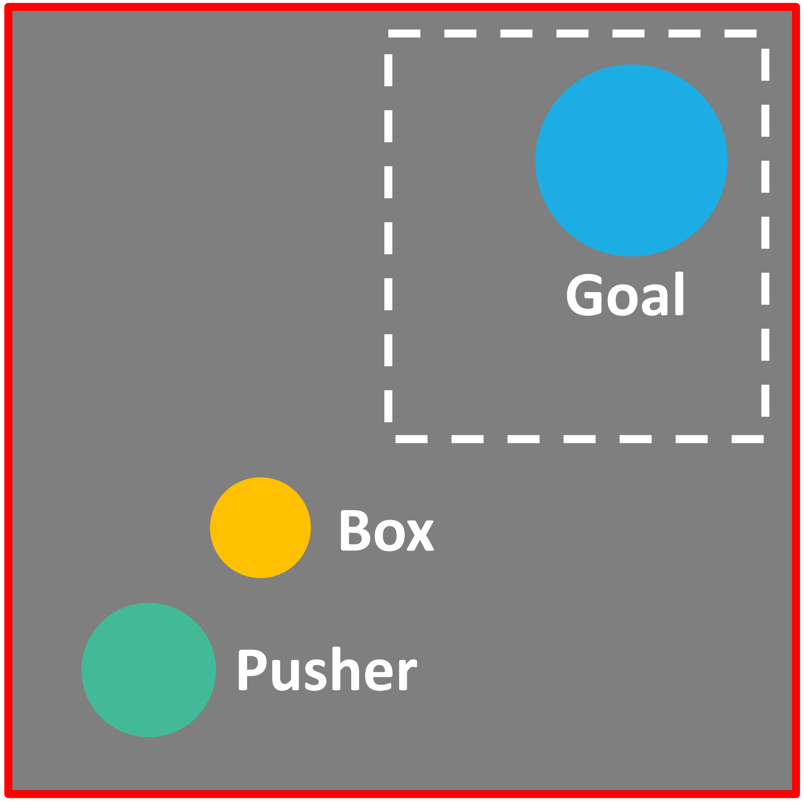
2D Pusher Environment



Reward

- -1 for non-terminal step
- $-T$ and terminates if out of bounds
- 0 and terminates if box touches a goal
- -1 and terminates after T steps

2D Pusher Environment



Characteristic
- Sparse reward
- Random actions rarely push a box into a goal

Key Question
- Can we learn something even when we don't touch a goal? (e.g. moved box)

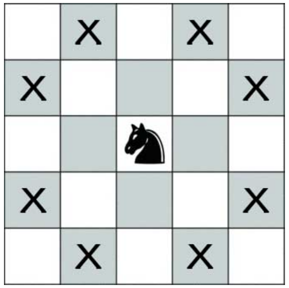
Preliminary: Task vs. Goal

Task

- Def: A piece of work to be done
- e.g. Beating an opponent in the chess (or Starcraft)

Goal

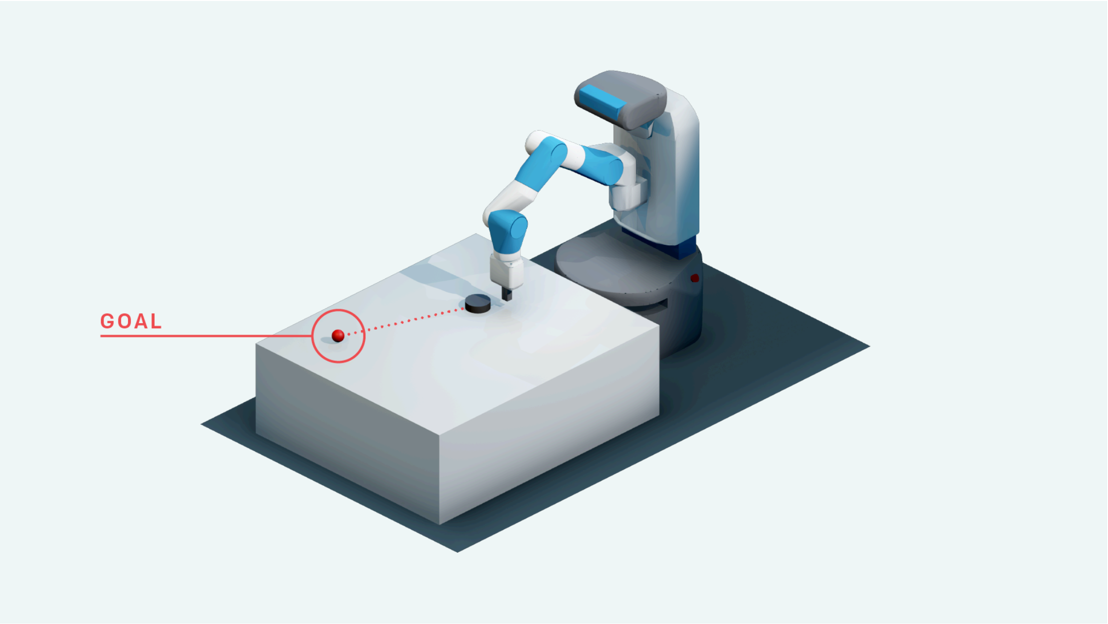
- Def: The end toward which effort is directed
- e.g. Move the knight to one of good positions



Motivation

Dealing with sparse rewards

- How can we evade the need for complicated reward engineering?



Problem Statement

In the many-goals setting, how can we achieve sample-efficient learning even if the reward signal is binary and sparse?

Assumptions

- We use an off-policy RL algorithm
- The goals influence the agent's actions, but not the environment dynamics

Proposed Method

Hindsight **E**xperience **R**eplay (a.k.a. **HER**)

Key idea

- Replay each episode with a different goal than the one the agent was trying to achieve, e.g, one of the goals already achieved in the episode (Hindsight)

Hypothesis

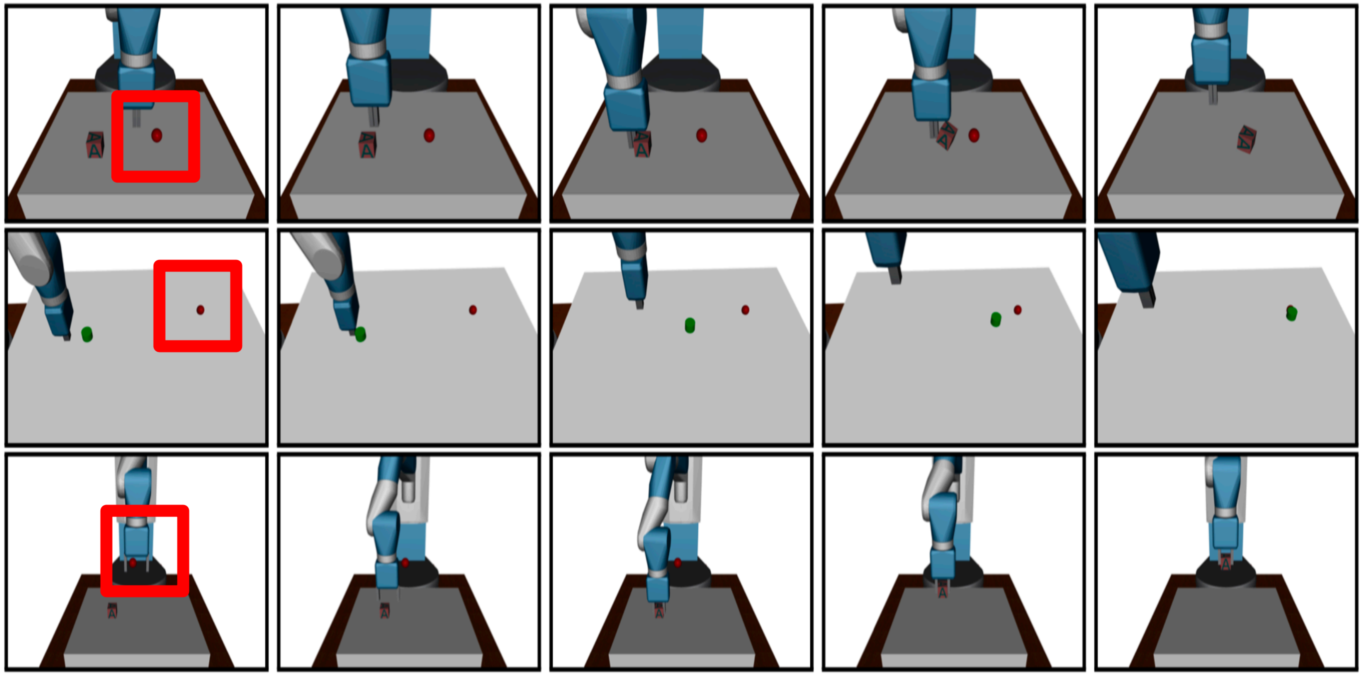
- Replaying with a different goal plays the role of implicit curriculum in a way to facilitate learning

Proposed Method

Algorithm 1 Hindsight Experience Replay (HER)

```
Given:
  • an off-policy RL algorithm 𝔸,                          ▷ e.g. DQN, DDPG, NAF, SDQN
  • a strategy 𝕊 for sampling goals for replay,             ▷ e.g. 𝕊(s_0, ..., s_T) = m(s_T)
  • a reward function r : 𝒮 × 𝒜 × 𝒢 → ℝ.                    ▷ e.g. r(s, a, g) = −[f_g(s) = 0]
Initialize 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 𝔸:
            a_t ← π_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 := 𝕊(current episode)
        for g' ∈ 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 𝔸 and minibatch B
    end for
end for
```

Experimental Setting



Pushing

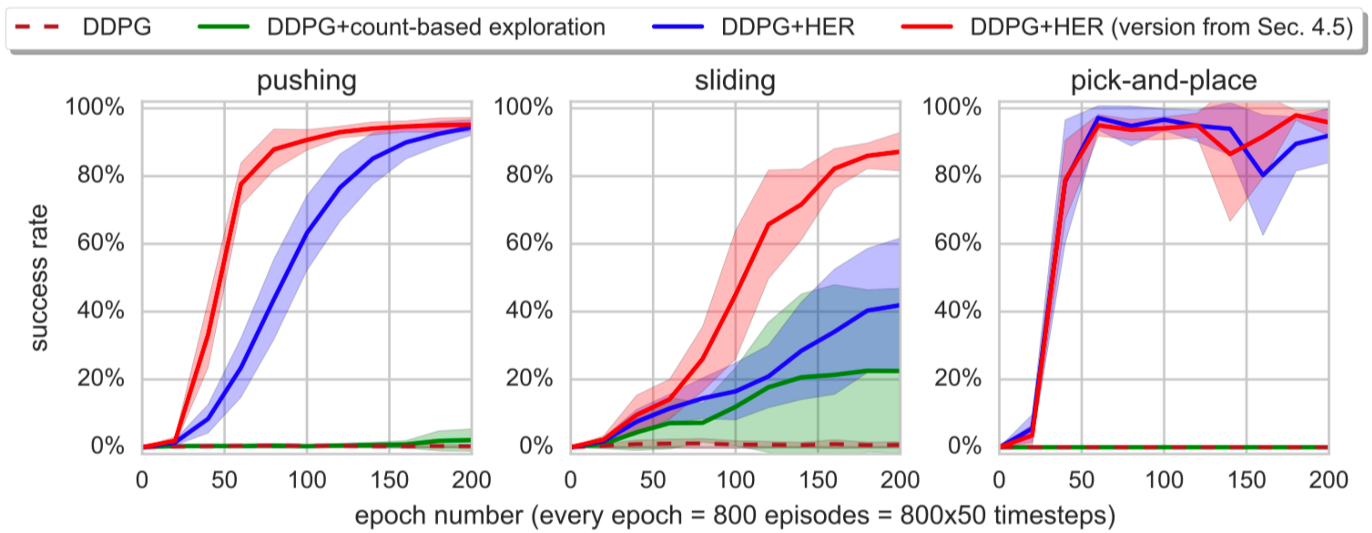
Sliding

Pick-and-place

State	Goal	Reward
Physical info	Desired position of the object	$f_g(s) = [\lvert g - s_{obj}\rvert \leq \epsilon]$

Result

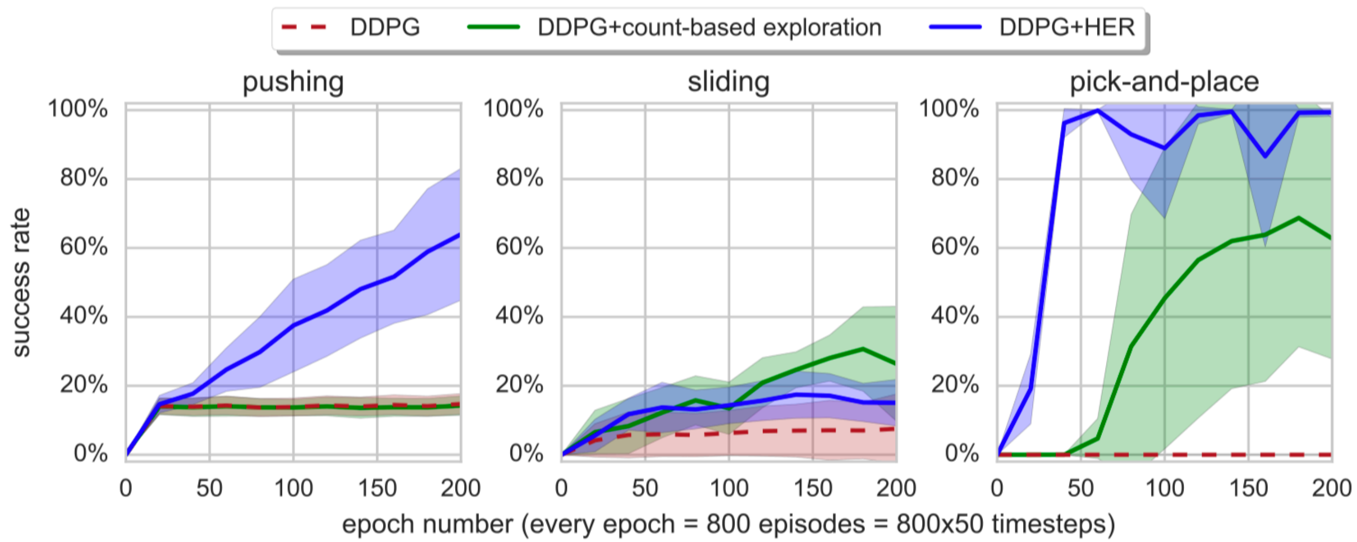
Does HER improve performance?



- Yes. DDPG+HER improves over baselines while baselines fail to solve (Pushing, Pick-and-Place) or makes slower progress (Sliding)

Result

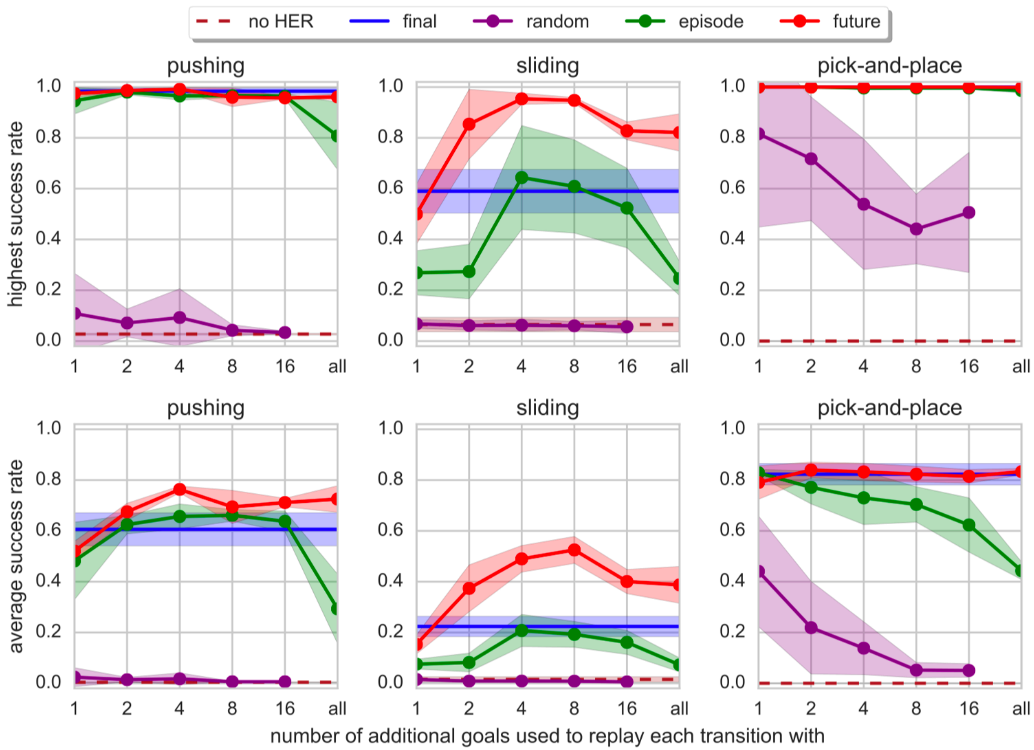
Does HER improve performance **even if there is a single goal we care about?**



- Yes. DDPG+HER is better than DDPG while DDPG fails to solve (Pushing, Pick-and-Place) or makes slower progress (Sliding)

Result

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



Result

Trained policy in the simulator solves the pick-and-place task well on the physical robot without any finetuning

Conclusion

HER enables sample-efficient learning using only sparse and binary rewards

We can combine HER with arbitrary off-policy RL algorithms

As a result, HER successfully learns complicated behavior three challenging tasks that vanilla RL algorithm fails to solve

- Note that it is the first work to solve these tasks only with binary and sparse reward

Question or Discussion