Hindsight Experience Replay

Bongseok kim

서울과학기술대학교 데이터사이언스학과

Abstract

Abstract

- Dealing with sparse rewards is one of the biggest challenges in Reinforcement Learning (RL)
- present a novel technique called *Hindsight Experience Replay* which allows **sample-efficient learning** from rewards which are **sparse and binary** and therefore avoid the need for complicated **reward engineering**.
- can be combined with any off-policy RL algorithm

Introduction

Introduction

Reinforcement learning (RL) combined with neural networks has recently led to a wide range of successes in learning policies for sequential decision-making problems (simulated environments).

- playing Atari games (Mnih et al., 2015)
- game of Go (Silver et al., 2016)
- helicopter control (Ng et al.,2006)

Introduction

- However, a common challenge, especially for robotics, is the need to **engineer a reward function**
- The necessity of cost engineering limits the applicability of RL in the real world
- it requires both RL expertise and domain-specific knowledge
- Moreover, it is not applicable in situations where we **do not know what admissible behaviour** may look like

Motivation

The goal of the game is to find the key.

- avoiding monsters and obstacles
- needs specific action sequences
  - Get down and up ladder or rope $\longrightarrow$ jump over the skull ..

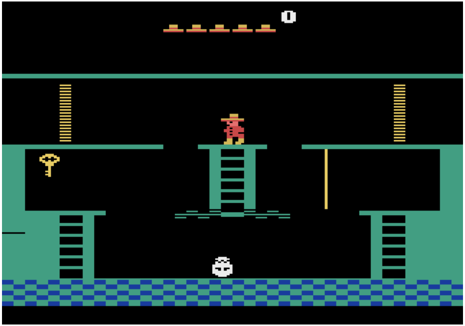


그림 1: Montezuma's Revenge

Motivation

In Montezuma's Revenge game

- **do not know what admissible or proper behaviour** may look like as mentioned
- sparse reward : agent succeed in getting the key, you get a 1 or 0 reward.
- needs enough exploration
- *Hindsight Experience Replay* deals with **sparse reward problem**

Hindsight Experience Replay

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

Background

Universal Value Function Approximators (UVFA)

Remind value approximation

- $V(s)$: represents the utility of any state s in achieving the agent's over all goal or reward function.
- $V(s;\theta)$: that estimate the long-term reward from any state s, using parameter θ
- we hope that value approximation work well on **unseen states (generalization over states)**

Universal Value Function Approximators (UVFA)

Suppose we want to make converter‘s input Voltage ,V_t^{in}
as close as possible to output Voltage, $V_{t+1}^{out} = 70$

- Aussme that system dynamics
  - $V_{t+1}^{out} = V_t^{in} \cdot a_t + \epsilon$
  - $a_t \in \{0, 1\}$
- we can simply train agent with reward function
  - $R(s, a) = (V_{t+1}^{out} - 70)^2$
- However in Real world, Agent should action with respect to all possible V_{t+1}^{out}
  - should we train the agent thousands of diffrent setting ?
  - $R(s, a) = (V_{t+1}^{out} - 50)^2$, goal 1
  - $R(s, a) = (V_{t+1}^{out} - 30)^2$, goal 2
  - ..

Universal Value Function Approximators (UVFA)

Again, consider combining a value function with a goal.

- $V_g(s)$: represent the utility of any state s in achieving a given specific goal g
- This value function is only valid for a specific goal g.
- $V(s, g; \theta)$: value func tion approximation to both states s and goals g,*UVFA*
- we hope that *UVFA* can work well on **unseen states and goal g (generalization over states and goal)**

Universal Value Function Approximators (UVFA)

We can concatenate state and g for policy, value function input ($\simeq$ artificial state) $[V_{in}^{t}, V_{out}]$

- $V(s||g, \theta)$
- $\pi(s_t||g) \rightarrow a_t$
- $R(s||g, a) = (V_{t+1}^{out} - g)^2$

- In practice
  - we just set $g \in \{30, 75\}$ and train agents
  - we can generalize $g \in \mathbb{R}$, unseen voltage reference

Universal Value Function Approximators (UVFA)

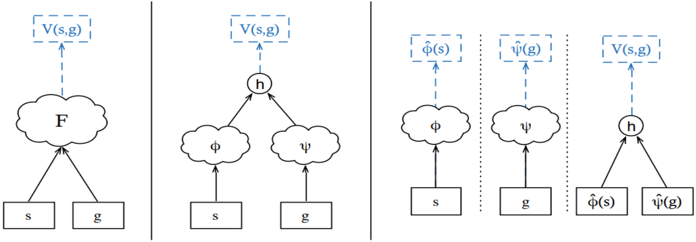


Figure 1. Diagram of the presented function approximation architectures and training setups. In blue dashed lines, we show the learning targets for the output of each network (cloud). **Left**: concatenated architecture. **Center**: two-stream architecture with two separate sub-networks ϕ and ψ combined at h. **Right**: Decomposed view of two-stream architecture when trained in two stages, where target embedding vectors are formed by matrix factorization (right sub-diagram) and two embedding networks are trained with those as multi-variate regression targets (left and center sub-diagrams).

Hindsight Experience Replay

Motivating example

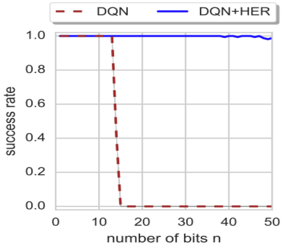
Consider a bit-flipping environment with

- the state space: S = $\{0, 1\}^n$ ex) $[0, 0, 0, 0, 1, 1, 1]$
- action space : $A = \{0, 1, ..n - 1\}$ for some integer n
  executing $i - th$ action filps the $i - th$ bit of state
- goal state : $[1, 1, 1, 1, 0, 0, 0]$
- $r_g(s, a)$: −1 as long as it is not in the goal state

Motivating example

- Standard RL algorithms are bound to fail in this environment for n > 40
  - such as DDPG, DQN ..
- because they will never experience any reward other than −1
- reward shaping such as, $r_g(s, a) = -||s - g||^2$ may work this environment but difficult to apply more complicated problems

Figure 1: Bit-flipping experiment.



Motivating idea

Consider episode with a states sequences $s_1, \dots, s_T$ and goal $g \neq s_1, \dots, s_T$ which implies

- for example, $s_T : [0, 0, 0, 0, 1]$ $g : [0, 0, 0, 1, 1]$
- agent fail to acheive goal
- agent received a reward of -1 at every time step

The pivotal idea behind *HER* approach is to re-examine this trajectory with a different goal. Consider again

- How about think of $s_T : [0, 0, 0, 0, 1]$ $g : [0, 0, 0, 0, 1]$?
- it may not help us learn how to achieve the state g,
- definitely tells us something about how to achieve the state s_T
- With this modification **at least half of the replayed trajectories contain rewards different from -1**

Motivating idea

Suppose that we have (s_{T-1}, a, s_T, r)

Example of Experience Replay Buffer with no HER

- $[1, 1, 1, 0, 1, 0]$, 1 , $[1, 0, 1, 0, 1, 0]$ -1
- $[1, 1, 1, 0, 1, 0]$, 4 , $[1, 0, 1, 0, 0, 0]$ -1

Example of Experience Replay Buffer with HER (k=2)

- $[1, 1, 1, 0, 1, 0]$, 1 , $[1, 0, 1, 0, 1, 0]$ -1
- $[1, 1, 1, 0, 1, 0]$, 1 , $[1, 0, 1, 0, 1, 0]$ 1, pseduo $g = [1, 0, 1, 0, 1, 0]$
- $[1, 1, 1, 0, 1, 0]$, 4 , $[1, 0, 1, 0, 0, 0]$ -1
- $[1, 1, 1, 0, 1, 0]$, 4 , $[1, 0, 1, 0, 0, 0]$ -1, pseduo $g = [1, 0, 1, 0, 0, 0]$

Mutil-goal RL

- to achieve multiple different goals follow the approach from *Universal Value Function Approximators* in HER
- Train polices and value functions which takes as input not only state $s \in S$ but also a goal $g \in G$
  - $\pi(s_t||g) \to a_t$
  - $Q^\pi(s_t, a_t, g) = \mathbb{E}[R_t|s_t, a_t, g]$
- HER show that training an agent to perform multiple tasks can be easier than training it to perform only one task

HER Strategy

after experiencing some episode $s_0, s_1, ..., s_T$, store in the replay buffer every transition $s_t \to s_{t+1}$ not only with the original goal used for this episode but also with a subset of other goals

Question : how to choose goal state ?

- **Final** : replay with k final state s_T are desired goal (coin flip example)
- **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

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

Experiments

Does HER improve performance?

- States : consists of angles and velocities of all robot joints ... from mujoco physics engin
- Goal : Goals describe the desired position of the object (a box or a puck depending on the task) with some fixed tolerance
- Rewards : use binary and sparse rewards

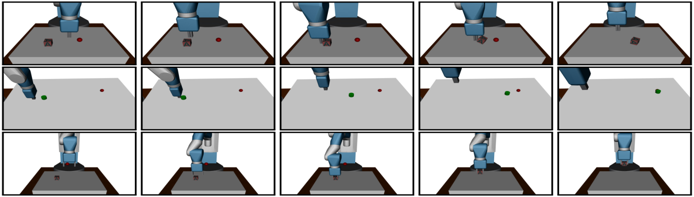
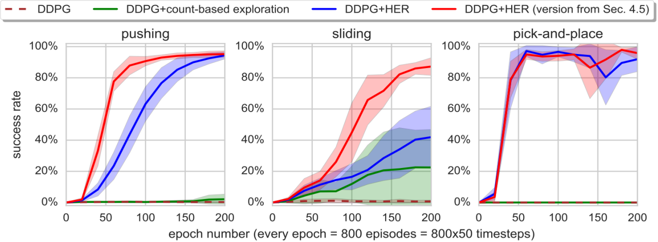


Figure 2: Different tasks: *pushing* (top row), *sliding* (middle row) and *pick-and-place* (bottom row). The red ball denotes the goal position.

Does HER improve performance?

- DDPG without HER is unable to solve any of the tasks
- DDPG with count-based exploration is only able to make some progress on the sliding task.
- DDPG with HER solves all tasks almost perfectly. .



How does HER interact with reward shaping?

- So far, we only considered binary reward
  - $r(s,a,g) = -|g - s_{object}| > \epsilon$
- Check out in case of reward shaping
  - $r(s,a,g) = \lambda|g - s_{object}|^p - |g - s'_{object}|^p$
- Surprisingly neither DDPG, nor DDPG+HER was able to successfully solve any of the tasks with any of these reward functions

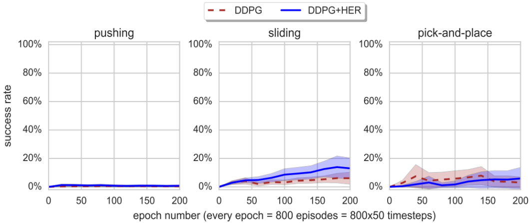


Figure 5: Learning curves for the shaped reward $r(s,a,g) = -|g - s'_{\mathbf{object}}|^2$ (it performed best among the shaped rewards we have tried). Both algorithms fail on all tasks.

```
"Homines, dum docent, discunt. (By teaching, we learn)."
```