

Hindsight Goal Prioritization for Sparse Reward Environments

Final Project

Reinforcement Learning

Flavio Maiorana (2051396)

08/09/2023



SAPIENZA
UNIVERSITÀ DI ROMA



Table of Contents

1 Introduction

► Introduction

► DDPG

► Replay buffer

► Results

► Conclusions



Problem statement

1 Introduction

Robotics environments

- Complex and different goals
- Sparse rewards
- Continuous action space



Problems with exploration and reward shaping

- Goal may be too complex and observation space is big: we may never get reward 1
- Classical off-policy algorithms don't valorize much the failed episodes



Solution: Enhancing the Replay Buffer



Fetch

1 Introduction

- Based on the 7-DoF Fetch Manipulator arm, with a two-fingered parallel gripper
- Tasks: Reach, Push, Slide and Pick-and-Place
- Action: $\text{Box}(-1.0, 1.0, (4,), \text{float32}) \rightsquigarrow$ Displacement in meters of the EE
- Observation: dictionary with info about the robot's end effector state and goal
 - Observation: ndarray of shape (25,) \rightsquigarrow kinematic info of the block object and EE
 - Desired goal: ndarray of shape (3,) \rightsquigarrow desired position of the EE or the block
 - Achieved goal: ndarray of shape (3,) \rightsquigarrow current position of the EE or the block
- Reward: if we use sparse rewards -1 for every timestep and 0 for reaching the goal
- Termination: episodes have no termination since they have infinite horizon. Thus, they are truncated after T steps (by default 50)



Table of Contents

2 DDPG

- ▶ Introduction
- ▶ DDPG
- ▶ Replay buffer
- ▶ Results
- ▶ Conclusions



Architecture - 1

2 DDPG

- Two neural networks in performing actor-critic policy gradient
 - **Actor** $\mu_\theta(s_i)$: observed state → action maximising the action-value function
 - **Critic** $Q_\phi(s_i, \mu_\theta(s_i))$: state and action → value of the action-value function
- Experience is collected according to the behavior policy $\pi_b = \mu_\theta(s_i) + \mathcal{N}$
- Transitions are stored in a **replay buffer**
- Every learning step, a batch $B = \{(s_i, a_i, d_i, r_i, s'_i)\}$ is sampled



Architecture - 2

2 DDPG

- Critic optimization through Bellman equation by minimizing

$$L(\phi) \approx \frac{1}{|B|} \sum_{i=1}^{|B|} \underbrace{\left[r_i + \gamma Q_{\phi^-}(s'_i, \mu_{\theta^-}(s'_i)) - Q_\phi(s_i, a_i) \right]^2}_{\text{Temporal Difference Error}} \quad (1)$$

- Actor optimization through policy gradient theorem by maximising

$$J(\theta) \approx \frac{1}{|B|} \sum_{i=1}^{|B|} Q_\phi(s_i, \mu_\theta(s_i)) \quad (2)$$

- **Target networks** μ_{θ^-} and Q_{ϕ^-} for stability, updated with polyak averaging
 - $Q_{\phi^-} \leftarrow \rho Q_{\phi^-} + (1 - \rho) Q_\phi$
 - $\mu_{\theta^-} \leftarrow \rho \mu_{\theta^-} + (1 - \rho) \mu_\theta$



Table of Contents

3 Replay buffer

- ▶ Introduction
- ▶ DDPG
- ▶ Replay buffer
- ▶ Results
- ▶ Conclusions



Structure

3 Replay buffer

- Matrix with M columns and T rows, with T the max number of steps in one episode
- The element (i, j) contains the j_{th} transition of the i_{th} episode
- A transition is defined as $(s_i, a_i, \text{achieved}_i, \text{desired}_i, r_i, s'_i)$
 - achieved_i is the current state as if the goal was reached
 - desired_i is the actual final goal of the episode
 - The reward is computed with desired_i

Training steps in Fetch

- For each step in the episode, select an action, clip it to the action space and observe the result of applying it
- Store the episode in a row of the buffer
- Sample $|B|$ transitions, each from a sampled episode
- Do a gradient step



HER

3 Replay buffer

- The intention is to valorize also failed episodes (majority in robotics environments)
- Done by storing episodes multiple times, substituting the desired goal with a **Hindsight Goal**, treating the episode as if it was successful

Additional steps for HER in DDPG

- Store every transition $(s_t || g, a_t, r_t, s_{t+1} || g)$ of the episode
- Sample a set of additional achieved goals G from the current episode
- Store $(s_t || g', a_t, r_t, s_{t+1} || g')$ for every $g' \in G$
- Perform gradient descent, feeding also the goals into the neural networks

Different strategies can be adopted to sample goals: final, future, random



HER

Pseudocode of how transitions are sampled with future strategy

- 1: $p_{future} \leftarrow 1 - \frac{1}{1+k}$ $\triangleright k$ the number of additionally stored episodes: $p_{future} \propto k$
- 2: $i_{ep} \leftarrow$ episodes indices uniformly sampled among the stored ones
- 3: $i_t \leftarrow$ transitions indices uniformly sampled from 0 to T
- 4: $i_{HER} \leftarrow$ indices of "resampled" future transitions, according to p_{future}
- 5: $i_{future} \leftarrow$ future transitions indices uniformly sampled from t to T
- 6: Substitute $goals[i_{HER}]$ with a future goal indexed by i_{future}
- 7: Recompute the reward of changed transitions



HGR

Prioritizing future goals

- For the j_{th} transition, a transition to be the hindsight goal is sampled according to

$$P'(j, i) = \frac{|\delta_{ji}|^{\alpha'}}{\sum_{j=0}^{T-2} \sum_{i=j+1}^{T-1} |\delta_{ji}|^{\alpha'}}, \text{ with } j + 1 \leq i \leq T - 1 \quad (3)$$

- Priorities: matrix of shape $\left(M, \frac{(T-1) \cdot T}{2}\right)$, with M the buffer size in episodes
- New transitions stored with maximal priority, to let them be replayed at least once



HGR

Prioritizing episodes

- Episodes are sampled according to the probability

$$P(n) = \frac{|\delta^{(n)}|^\alpha}{\sum_n |\delta^{(n)}|^\alpha} \quad (4)$$

where $|\delta^{(n)}|$ is the average TD Error of an episode

- Sumtree for efficiently sampling from this probability
 - Priorities in the leaves
 - Every node contains the sum of its children



HGR

Annealing the bias

- Importance sampling weights helps to scale down the sample's gradient when it is updated frequently and remains the same when it is rarely updated

$$w_{ji}^{(n)} = \left(\frac{1}{N_e} \frac{1}{P(n)} \right)^\beta \cdot \left(\frac{2}{(T-1) \cdot T} \cdot \frac{1}{P'(j, i)} \right)^{\beta'} \quad (5)$$

- In order to always scale the gradients down we set $w_{ji}^{(n)} = \frac{w_{ji}^{(n)}}{\max(w_{ji}^{(n)})}$
- Mintree for storing minimum weight, for efficiently retrieving inverse of max weight



Table of Contents

4 Results

- ▶ Introduction
- ▶ DDPG
- ▶ Replay buffer
- ▶ Results
- ▶ Conclusions



Training setup

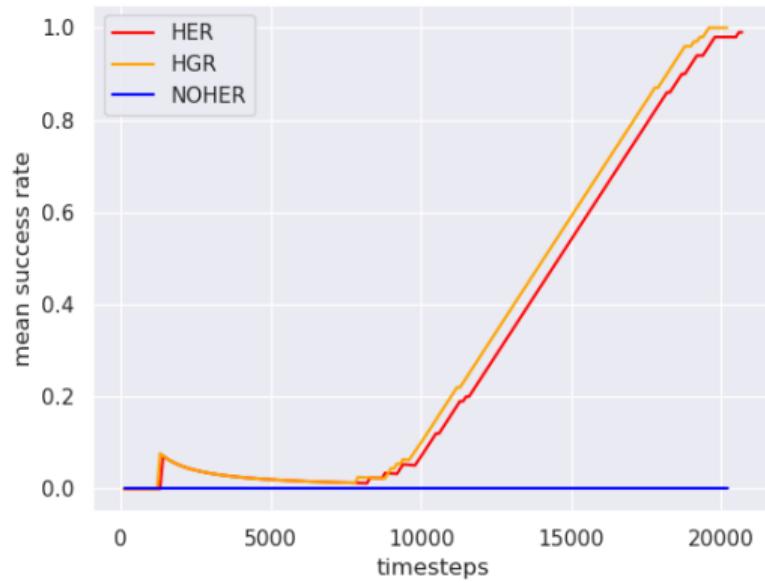
4 Results

- Normal noise with standard deviation 0.2
- $\gamma = 0.98$
- Learning rates 0.001, polyak 0.95
- buffer capacity $1 \cdot 10^6$, batch size 256
- gradient steps 50
- $\alpha = \alpha' = 0.6$
- $\beta_0 = \beta' = 0.4$
- β organized with linear schedule from β_0 to 1



Reach

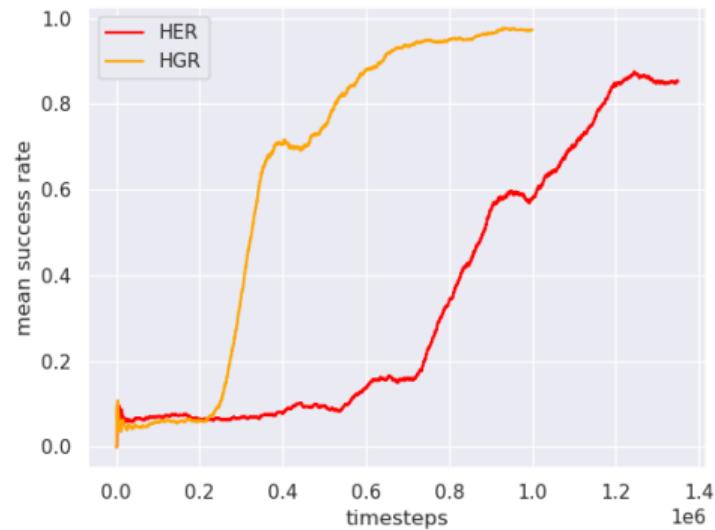
4 Results





Push

4 Results





Pick-and-Place

4 Results

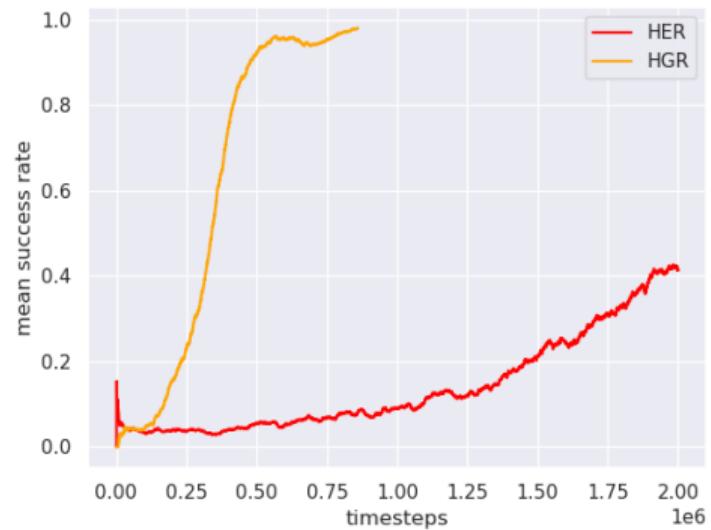




Table of Contents

5 Conclusions

- ▶ Introduction
- ▶ DDPG
- ▶ Replay buffer
- ▶ Results
- ▶ Conclusions



Final considerations

5 Conclusions

- Sampling additional goals solves the exploration problem in multi-goal sparse reward environments
- Prioritizing episodes and goals makes sampling even more effective and thus significantly reduces training time and the amount of collected experience



Resources

5 Conclusions

- T. M. Luu, C. D. Yoo: HGR on Replay Buffer for Sparse Reward Environment
- Marcin Andrychowicz et al. : Hindsight Experience Replay
- Tom Schaul et al. : Prioritized Experience Replay
- OpenAI Spinning Up



Hindsight Goal Prioritization for Sparse Reward Environments

Thank you for listening!

Any questions?