

1707.08817 - Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards

Leveraging Demonstrations for Deep Reinforcement Learning on Robotics Problems with Sparse Rewards

Matej Večerík, Todd Hester, Jonathan Scholz, Fumin Wang
Olivier Pietquin, Bilal Piot, Nicolas Heess
Thomas Rothörl, Thomas Lampe, Martin Riedmiller
Deepmind

matejvecerik, toddhester, jscholz, awaw
pietquin, piot, heess
tcr, thomaslampe, riedmiller@google.com

- **Yunqiu Xu**
 - DDPGfD:
 - Combine DQfD with DDPG → continuous action space
 - Typical RL: carefully engineered shaping rewards → sample inefficiency
 - Use demonstration: reduce exploration
 - Similar to DQfD, store both demonstration and actual interactions in replay buffer, and use prioritized mechanism
 - Experiment on both simulated and real robot manipulating tasks
-

1. Introduction

- Challenges: robot task with sparse reward
 - The goal of hand-coded reward function is natural to be sparse
 - Use demonstration can reduce exploration and carefully reward shaping
- This work can be seen as an extension of DQfD
- Why use DDPG:

- Compared with on-policy methods, it can learn from arbitrary transition data
- Experience replay → maintain these transitions to propagate sparse rewards
- Compared with vanilla DQN, DDPG can handle continuous action space

2. Related Work

- Our work: combine imitation learning with learning from task rewards, to handle continuous action space
- DAGGER:
 - Simple imitation learning, can not learn to improve
 - Need human expert's participation
- Inverse RL
 - Learn cost/reward function under which the demonstration is optimal
 - Requires knowledge of the dynamics
 - Can not handle continuous tasks
- Guided Cost Learning (GCL) and GAIL:
 - Do not need knowledge of dynamics and hand-crafted features
 - Learn reward and policy to imitate expert demonstrations
 - At each episode, sample demonstrations of current policy, then use these sampled demonstrations and expert demonstration to generate reward function
 - GCL and GAIL are different from how they generate reward function
 - Weakness:
 - Data generation can be unstable (e.g. GAN)
 - Can not handle continuous tasks
- Guided Policy Search
 - Find an optimal policy by decomposing the task into 3 steps
 - Step 1: learn dynamics from expert demonstrations
 - Step 2: Find locally optimal policies via optimal control methods (e.g. iLQG or DDP)
 - Step 3: Fit the demonstrations via supervised learning

- Assumption: reward must be shaped carefully, which may be impractical
- DQfD → discrete version of this work

3. DDPG from Demonstrations

- Modification 0: Before training, store expert demonstrations into replay buffer, and keep them forever
- Modification 1: use prioritized replay to handle sparse rewards

$$p_i = \delta_i^2 + \lambda_3 |\nabla_a Q(s_i, a_i | \theta^Q)|^2 + \epsilon + \epsilon_D$$

- δ_i : TD error for this transition
- $|\nabla_a Q(s_i, a_i | \theta^Q)|^2$: loss applied to the actor
- ϵ and ϵ_D : probabilities for all / expert demonstrations to be sampled
- Modification 2: mix 1-step and n-step returns → help to propagate the Q-values along the trajectories
- Modification 3: do multiple learning updates per env step
 - With same minibatch, multiple updates will be performed to improve data efficiency
 - May lead to unstable → 20-40 in practice → balance efficiency and stability
- Modification 4: L2 regularization to stabilize final performance
- DDPG:

- Actor: policy to maximize action value with respect to parameters, update by policy gradient
- Critic: action-value function to evaluate Q value, update by Bellman function

$$\nabla_{\theta^\pi} L_{Actor}(\theta^\pi) = -\nabla_{\theta^\pi} J(\theta^{\pi_i}) + \lambda_2 \nabla_{\theta^\pi} L_{reg}^A(\theta^\pi)$$

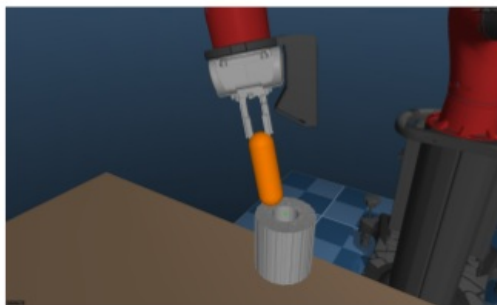
$$L_{Critic}(\theta^Q) = L_1(\theta^Q) + \lambda_1 L_n(\theta^Q) + \lambda_2 L_{reg}^C(\theta^Q)$$

- Summary of modifications compared with DDPG
 - Add human demonstrations into replay buffer
 - Prioritized replay
 - Mix 1-step ($L_1(\theta^Q)$) and n-step return ($L_n(\theta^Q)$)

- Learning multiple times per env step → **DQfD 未提及**
- L2 regularization

4. Experiment

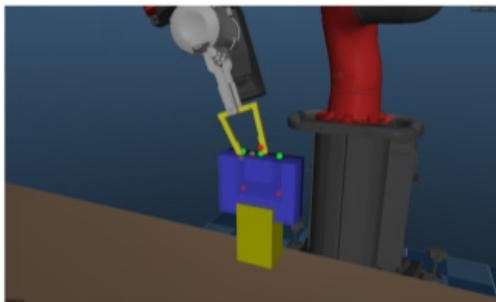
- Tasks: easy to specify goal, but difficult to specify distance function for reward shaping → easy to stuck in local optimal
 - A set of insertion tasks with a range of exploration difficulties



(a) Peg Insertion Task.



(b) Hard-drive Task.



(c) Clip Insertion Task



(d) Cable Insertion Task.

Figure 1: This figure shows the four different insertion tasks.

- 2 kinds of reward functions
- Device: 7-DOF robot arm
- Demonstration data is collected by human demonstrator
- Result 1: performance

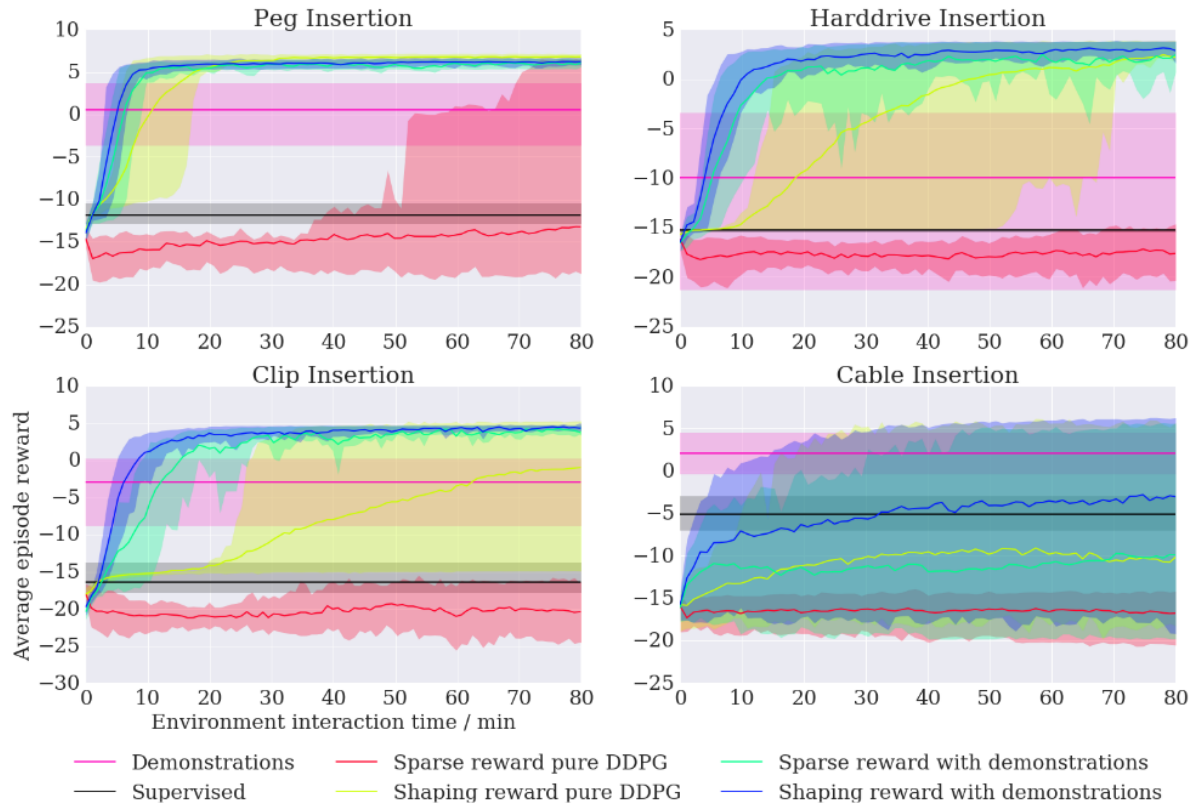


Figure 3: Learning curves show the means and 10th and 90th percentiles of rewards for the four approaches on each of the four tasks, with statistics computed over 64 trials. We measure reward against environment interaction time. Each episode was at most 5s long and the agent control rate was about 6Hz. The plots also show the mean and percentiles for the rewards received in each set of human demonstration and of supervised imitator which predicts demonstration actions trained with an ℓ_2 loss. The results show that DDPGfD out-performs DDPG, even when DDPG is given hand-tuned shaping rewards and DDPGfD exhibits a more robust training behaviour.

- Result 2: the number of demonstration
- Result 3: real robot

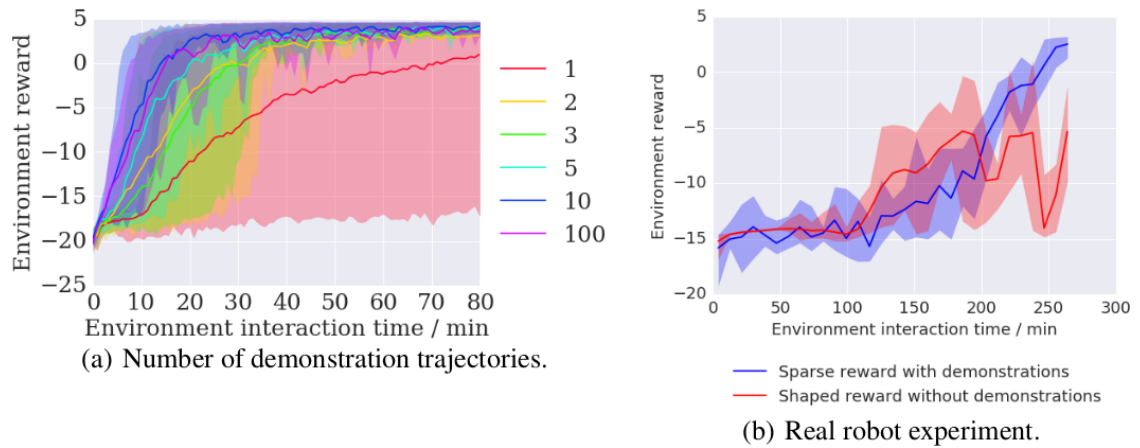


Figure 4: (a) Learning curves for DDPGfD on the clip insertion task with varying amounts of demonstration data. DDPGfD can learn solve the sparse-reward task given only a single trajectory from a human demonstrator. (b) Performance from 2 runs on a real robot. DDPGfD learns faster than DDPG and without the engineered reward function.

5. Summary

- Extend DQfD to DDPGfD → continuous action space
- One-shot imitation can be achieved for some injection tasks
- This work can be compared with OPENAI's work : 1709.10089 - Overcoming Exploration in Reinforcement Learning with Demonstrations