Policy Optimization with Demostration

Author Shao-Xiong,Zheng
Department of Computer Science
National Chiao Tung University
a84959947mp45@gmail.com

1 Introduction

Please provide a clear overview of the selected paper. You may want to discuss the following aspects:

- The main research challenges tackled by the paper

  Reinforcement Learning (RL) solves sequential decision making problems based on the experiences collected by interacting with environments.However, exploration problems, that how to gain more experience with novel strategies to improve the performance in the long run, are still challenging in deep RL methods.

  When the reward signals are sparse and rare,existing methods still struggle to explore effectively to learn meaningful policies. This is because most of them rely on heuristic exploration strategies,e.g.ϵ-greedy for value based methods ,noise-based exploration for policy gradient methods,which are undirected and incapable of finding interesting states explicitly.

  Some recent works (Nair et al., 2017; Houthooft et al., 2016; Plappert et al., 2017; Pathak et al., 2017) have been devoted to tackling the exploration problems in RL. They are basically rooted in the following two ideas.

  1) Reshape the original reward function by encouraging the agent to visit states never seen before, driven by intrinsic curiosity (Pathak et al., 2017) or information gain (Houthooft et al., 2016).

  2) Use demonstration trajectories sampled from an expert policy to guide the learning procedure, by either putting the demonstrations into a replay memory (Nair et al., 2017; Hester et al., 2017; Vecer ık et al., 2017) or using them to pretrain the policy in a supervised manner (Silver et al., 2016).

  Learning from demonstrations has shown promising performance in overcoming exploration difficulties in sparsereward environments. However, existing schemes cannot fully leverage the power of the demonstration data, limited by only treating them in the same way as self-generated data, and usually require a tremendous number of high-quality demonstrations which are difficult to collect at scale.

- The high-level technical insights into the problem of interest

  To address above problem, we propose to combine above two ideas by developing a principled method that rewards demonstration-like actions more during interaction with environments, which thus encourages exploration for meaningful states when feedback is sparse. The intuition is that, when the reward signal is not available, the agent should mimic the demonstrated behavior in early learning stages for exploration.

After acquiring sufficient skills, the agent can explore new states on its own. This is actually a dynamic intrinsic reward mechanism that can be introduced for reshaping native rewards in RL.

- The main contributions of the paper (compared to the prior works)

  we propose a novel Policy Optimization from Demonstration (POfD) method, which can acquire knowledge from demonstration data to boost exploration, even though the data are scarce and imperfect. We realize our idea with three technical novelties.

  1) We reformulate the policy optimization objective by adding a demonstration-guided exploration term, which measures the divergence between the current policy and the expert one, forcing expert-alike exploration. We theoretically analyze the benefits brought by POfD to vanilla policy gradient ones, in terms of improvement over the expected return.

  2) We convert the proposed objective to a new one defined on occupancy measure (Ho Ermon, 2016) for better exploiting demonstrations and establish an optimization-friendly lower bound.

  3) We eventually draw a connection between optimizing the derived lower bound and generative adversarial training (Goodfellow et al., 2014), and show that the optimization can thus easily proceed in a manner of alternating between two sub-procedures. One is fitting a discriminator measuring the similarity between expert data and self-generated data to reshape the reward; the other one is updating the policy with the gradient from the reshaped reward function.

  POfD is general and compatible with most policy gradient methods. We also show that existing replay memory based learning from demonstration methods (Hester et al., 2017; Vecer ık et al., 2017) can be interpreted as degenerated cases of our method int terms of how to leverage the demonstration data.

  We evaluate our POfD on physical locomotion tasks based on Mujoco (Todorov et al., 2012) in sparse-reward environments. We compare POfD against 5 state-of-the-art baselines. The experiments clearly demonstrate that POfD surpasses all the well-established baselines and performs very well in these environments. Its performance is even comparable with that achieved by policy gradient methods in oracle dense-reward environments.

- Your personal perspective on the proposed method

  I think the article is interesting. My research area is GOMOKU and connect-6. In GOMOKU, we use the alpha go-zero method to train, but in order to save time, I will first get on with self-play (MTCS from a small model (such as 32x5 ResNet)),and get the complete Demostration, and expand to a large model (such as 64x10 ResNet). the method training method will increase the speed by 10 times than before, but I also found some disadventage, that is, the effectiveness of the model generated by demostration It can't be better than the original, and it must continue self-play (it saves a lot of time as a whole).I encountered the problem mentioned in the above paper,following:

  (1) Not sure that the trajectory used is good enough

  (2) our new model only update according to the trajectory of demostraion, there is no way to explore it itself.

  I already understand the overall mathematical derivation of this paper. This paper may solve my problem. Half of the alpha go-zero update method is policy optimization. The point of this article is that it can be used as long as it is policy-based! So I look forward to using it in the future.

2 Problem Formulation

Please present the formulation in this section. You may want to cover the following aspects:

- Your notations (e.g. MDPs, value functions, function approximators,...etc)

  MDP is defined by a tuple (S, A, P, r, γ), where S and A are the state space and the action space respectively,$P(s^{'}|s,a)$is the transition distribution of taking action a at state s,$r(s,a)$is the reward function, and $\gamma \in (0,1)$ is the discount factor.

  Given a stochastic policy $\pi(a|s) = p(a|s;\pi)$ (**Agent policy distribution**)mapping from states to action probabilities, the performance of π is usually evaluated by its **expected discounted reward** $\eta(\pi)$:

  $$\eta(\pi) = \mathbb{E}_\pi[r(s,a)] = \mathbb{E}_{s0,a0,s1,...}\left[\ \sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)\ \right], (1)$$

  where $(s_0, a_0, s_1, ...)$ is a trajectory generated by executing policy π,i.e., $s_0{\sim}p_0, a_t{\sim}\pi(\cdot|s_t, a_t)$ and $s_{t+1} \sim P(\cdot|s_t, a_t)$. Similarly, following the standard definitions,the value function$V_\pi$and action value function Q_π can be written as $V_\pi = \mathbb{E}_\pi[r(\cdot,\cdot)|s_0 = s]$ and $Q_\pi(s,a) = \mathbb{E}_\pi[r(\cdot,\cdot)|s_0 = s, a_0 = a]$ Subtracting $Q_\pi(s,a)$ by $V_\pi(s)$ gives the advantage function $A_\pi(s,a) = Q_\pi(s,a) - Q_\pi(s)$ that reflects the expected additional reward that the agent will get after taking action a in state s.

  Reinforcement Learning (RL) is a set of algorithms trying to infer a policy achieving maximal reward $\eta(\pi)$ with regard to some form of reward signals $r(s,a)$ from trajectories $\mathcal{D} = \{\tau_i\}$ (**a set of trajectory**) generated by executing a current policy (on-policy methods) or some other policy (off-policy methods). Each trajectory consists of a sequence of state transitions $\tau = \{(s_0, a_0), (s_1, a_1), \ldots, (s_T, a_T)\}$ The occupancy measure, defined as follows, characterizes the distribution of action-state pairs when executing policy π.

  Definition 1. (Occupancy measure) Let $\rho_\pi(s) : \mathcal{S} \rightarrow \mathbb{R}$ denote the unnormalized distribution of state visitation by following policy π in the environment:

  $$\rho_\pi(s) = \sum_{t=0}^{\infty}\gamma^t P\left(s_t = s|\pi\right)$$

  Then the unnormalized distribution of state-action pairs $\rho_\pi(s,a) = \rho_\pi(s)\pi(a|s)$ is called occupancy measure of policy π Substituting $\rho_\pi(s,a)$ into Eqn. (1), we have the following equivalent formulation for the expectation over policy π :

  $$\begin{aligned}\mathbb{E}_\pi[r(s,a)] &= \sum_{t=0}^{\infty}\sum_s P\left(s_t = s|\pi\right)\sum_a \pi(a|s)\gamma^t r(s,a) \\ &= \sum_s \rho_\pi(s)\sum_a \pi(a|s)r(s,a) \qquad\qquad , (2) \\ &= \sum_{s,a}\rho_\pi(s,a)r(s,a)\end{aligned}$$

  which provides an alternative way to compute the expected discounted return. An important property of the occupancy measure is that it uniquely specifies a policy, as described in the following lemma.

  Lemma 1.Suppose ρ is the occupancy measure for $\pi_\rho(a|s) \triangleq \frac{\rho(s,a)}{\sum_{a'}\rho(s,a')}$. Then π_ρ is the only policy whose occupancy measure is ρ .

Proof .

$$\rho_\pi(s,a) = \rho_\pi(s)\pi_\rho(a|s)$$
$$=> \pi_\rho(a|s) = \rho_\pi(s,a)/\rho_\pi(s)$$
$$=> \pi_\rho(a|s) = \rho_\pi(s,a)/\sum_{a'}\rho_\pi(s,a')$$

In particular, in addition to sparse rewards from environments as in traditional RL settings, the agent is also provided with a few (and possibly imperfect) demonstrations $\mathcal{D}^E = \{\tau_1, \tau_2, \ldots, \tau_N\}$, where the i -th trajectory $\tau_i = \left\{\left(s_0^i, a_0^i\right), \left(s_1^i, a_1^i\right), \ldots, \left(s_T^i, a_T^i\right)\right\}$ is generated from executing an unknown expert policy π_E in the environment. We aim to develop a method that can boost exploration through effectively leveraging $\mathcal{D}^E$ in such settings and $\max$ imize $\eta(\pi)$ in Eqn. (1)

we use π_E to denote the expert policy that gives the relatively good $\eta(\pi)$, and use $\mathbb{E}_{\mathcal{D}}$ to denote empirical expectation estimated from the demonstrated trajectories $\mathcal{D}^E$. We have the following reasonable and necessary assumption on the quality of the expert policy π_E.

Assumption 1. In early learning stages, we assume acting according to expert policy π_E will provide higher advantage value with a margin as least δ over current policy π, i.e.

$$\mathbb{E}_{a_E \sim \pi_E, a \sim \pi}\left[A_\pi\left(s, a_E\right) - A_\pi(s,a)\right] \geq \delta$$

We do not need to assume the expert policy π_E to be advantageous over all the policies, as our proposed POfD will learn from both rewards and demonstrations and can possibly learn a policy better than π_E through exploration on its own in later learning stages.

- The technical assumptions

Suppose π_θ is a θ -parameterized policy and is differentiable. Policy gradient methods optimize the expected return $\eta\left(\pi_\theta\right)$ by updating θ with the gradient of $\eta\left(\pi_\theta\right)$ w.r.t. θ . They usually start optimizing the policy π_θ by random exploration, which may cause slow convergence when the state and action spaces have high dimensionality, and may even lead to failure when the environment feedback is sparse. We propose to address this problem by forcing the policy to explore in the nearby region of the expert policy π_E (as shown in Fig.1), which is specified by several demonstrated trajectories $\mathcal{D}^E$

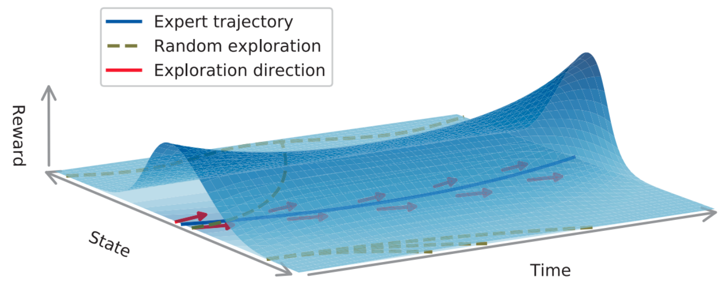


Figure 1: Our proposed POfD explores in the high-reward regions (red arrows), with the aid of demonstrations (the blue curve). It thus performs better than random explorations (olive green dashed curves) in sparse-reward environments.

Besides maximizing the expected return $\eta\left(\pi_\theta\right)$ through learning from sparse feedback during interaction with environments, we also encourage the policy π to explore by "following" the demonstrations $\mathcal{D}^E$. We therefore introduce demonstration-guided exploration term $\mathcal{L}_M\left(\pi_\theta, \pi_E\right) = D_{JS}\left(\pi_\theta, \pi_E\right)$, which is defined over Jensen-Shannon divergence between current policy π_θ and the expert one π_E, to the vanilla objective $\eta\left(\pi_\theta\right)$. This gives a new learning objective:

$$\mathcal{L}\left(\pi_\theta\right)=-\eta\left(\pi_\theta\right)+\lambda_1 D_{JS}\left(\pi_\theta, \pi_E\right)$$

where λ_1 is a trading-off parameter. However, the above policy divergence measure between π_θ and π_E is infeasible as π_E is unknown. Fortunately, leveraging the one-to-one correspondence between the policy and occupancy measure as given by Lemma 1, we can instead define a divergence over the occupancy measures $\rho_\pi(s,a)$ and $\rho_{\pi_E}(s,a)$, which is easier to optimize through adversarial training on demonstrations, as we will show later. Based on their occupancy measure $\mathcal{L}_M \triangleq D_{JS}\left(\rho_\theta, \rho_E\right)$, where ρ_θ and ρ_E are short for ρ_{π_θ} and ρ_{π_E}, our proposed demonstration guided learning objective is

$$\mathcal{L}\left(\pi_\theta\right)=-\eta\left(\pi_\theta\right)+\lambda_1 D_{JS}\left(\rho_\theta, \rho_E\right), \textbf{(3)}$$
$$P.S. D_{JS}\left(\rho_\theta, \rho_E\right)=\left(D_{KL}\left(\rho_\theta, \rho_E\right)+D_{KL}\left(\rho_E, \rho_\theta\right)\right)/2$$

We here slightly abuse D_{JS} to apply it to the unnormalized distribution ρ. Later, we will establish a lower bound to it without needing to directly optimize it.

Benefits of Exploration with Demonstrations

To better understand the new objective (3), we here prove that introducing the guiding term $D_{JS}\left(\rho_\theta, \rho_E\right)$ boosts the advantage value for the learned policy, and brings nontrivial benefits in terms of policy improvement for policy gradient methods. To see this, we introduce following the useful expression at first that is commonly used in policy gradient methods,

$$\eta(\pi)=\eta\left(\pi_{\text{old}}\right)+\mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t A_{\pi_{\text{old}}}(s,a)\right], \textbf{(4)}$$

In the above, the expected return $\eta(\pi)$ of policy π is expressed in terms of the advantage over the policy π_{old} in the previous iteration. Eqn. (4) can be rewritten as

$$\eta(\pi)=\eta\left(\pi_{\text{old}}\right)+\sum_s \rho_\pi(s)\sum_a \pi(a|s)A_{\pi_{\text{old}}}(s,a), \textbf{(5)}$$

To alleviate difficulties brought by complex dependency of $\rho_\pi(s)$ over π, policy gradient methods usually optimize the following surrogate objective, which is a local approximation to $\eta(\pi)$ up to first order:

$$J_{\pi_{\text{old}}}(\pi)=\eta\left(\pi_{\text{old}}\right)+\sum_s \rho_{\pi_{\text{old}}}(s)\sum_a \pi(a|s)A_{\pi_{\text{old}}}(s,a)$$

where ρ_π is replaced by $\rho_{\pi_{\text{old}}}$ and we ignore the change in state distribution due to policy update. Policy gradient methods are guaranteed to improve $\eta(\pi)$ monotonically by optimizing the above surrogate $J_{\pi_{\text{old}}}(\pi)$ with a sufficiently small update step $\pi_{\text{old}} \rightarrow \pi$ such that $D_{KL}^{\max}\left(\pi, \pi_{\text{old}}\right)$ is bounded. In particular, TRPO imposes hard constraints with fixed penalty on $D_{KL}^{\max}\left(\pi, \pi_{\text{old}}\right)$ while PPO and natural policy gradient use $D_{KL}^{\max}\left(\pi, \pi_{\text{old}}\right)$ for regularization with fixed or adaptive weights.

POfD additionally imposes a regularization $D_{JS}\left(\pi_\theta, \pi_E\right)$ between π_θ and π_E in order to encourage explorations around regions demonstrated by the expert π_E. We formally show the benefits from leveraging the expert demonstration in this way by giving the following theorem.

Theorem 1

Let

$$\alpha=D_{KL}^{\max}\left(\pi_{old}, \pi\right)=\max_s D_{KL}\left(\pi(\cdot|s), \pi_{old}(\cdot|s)\right),$$

$$\beta=D_{JS}^{\max}(\pi_E, \pi)=\max_s D_{JS}\left(\pi(\cdot|s), \pi_E(\cdot|s)\right),$$

and π_E is an expert policy satisfying Assumption 1. Then we have

$$\eta(\pi) \geq J_{\pi_{old}}(\pi) - \frac{2\gamma(4\beta\epsilon_E + \alpha\epsilon_\pi)}{(1-\gamma)^2} + \frac{\delta}{1-\gamma}$$

We provide proof in the supplement. The above theorem implies the benefits of adding matching regularization. Let

$$\begin{aligned}
&M_i(\pi) = J_{\pi_i}(\pi) - C_{\pi_E} D_{JS}^{\max}(\pi, \pi_E) - C_\pi D_{KL}^{\max}(\pi, \pi_i) + \hat{\delta} \\
&\text{ where } C_{\pi_E} = \tfrac{8\gamma\epsilon_E}{(1-\gamma)^2}, C_\pi = \tfrac{2\gamma\epsilon_\pi}{(1-\gamma)^2}, \hat{\delta} = \tfrac{\delta}{1-\gamma}. \\
&\text{ Then} \\
&\eta(\pi_{i+1}) \geq M_i(\pi_{i+1}) \\
&\eta(\pi_i) = M_i(\pi_i) + C_{\pi_E} D_{JS}^{\max}(\pi_i, \pi_E) - \hat{\delta} \\
&\eta(\pi_{i+1}) - \eta(\pi_i) \\
&\quad \geq M_i(\pi_{i+1}) - M_i(\pi_i) - C_{\pi_E} D_{JS}^{\max}(\pi_i, \pi_E) + \hat{\delta}
\end{aligned}$$

The above result is reminiscent of classic monotonic improvement guarantees for policy gradient methods. However, POfD brings another factor to the improvement $-C_{\pi_E} D_{JS}^{\max}(\pi_i, \pi_E) + \hat{\delta}$. This implies that following the demonstrations (i.e., having small $D_{JS}^{\max}(\pi_i, \pi_E)$) will fully utilize the advantage $\hat{\delta}$ and bring improvement with a margin over pure policy gradient methods.

In this subsection, we introduce a practical optimization algorithm for Eqn. (3), which is compatible with any policy gradient methods. In particular, instead of performing optimization on the difficult Jensen-Shannon divergence directly, we optimize its lower bound given as follows.

we introduce a practical optimization algorithm for Eqn. (3), which is compatible with any policy gradient methods. In particular, instead of performing optimization on the difficult Jensen-Shannon divergence directly, we optimize its lower bound given as follows.

Theorem 2.

Let $h(u) = \log\left(\frac{1}{1+e^{-u}}\right), \bar{h}(u) = \log\left(\frac{e^{-u}}{1+e^{-u}}\right)$ and $U(s,a) : \mathcal{S} \times A \to \mathbb{R}$ be an arbitrary function. Then we have

$$\begin{aligned}
&D_{JS}(\rho_\pi, \rho_E) \geq \\
&\quad \sup_U \left(\mathbb{E}_{\rho_\pi}[h(U(s,a))] + \mathbb{E}_{\rho_E}[\bar{h}(U(s,a))]\right) + \log 4
\end{aligned}$$

We defer the proof to supplement due to space limit. With the above theorem, the occupancy measure matching objective $\mathcal{L}_M$ can be written as

$$\mathcal{L}_M \triangleq \sup_{D \in (0,1)^{S \times \mathcal{A}}} \mathbb{E}_{\pi_\theta}[\log(D(s,a))] + \mathbb{E}_{\pi_E}[\log(1 - D(s,a))]$$

where $D(s,a) = \frac{1}{1+e^{-U(s,a)}} : \mathcal{S} \times \mathcal{A} \to (0,1)$ is an arbitrary mapping function followed by a sigmoid activation function for scaling. The supremum ranging over $D(s,a)$ thus represents the optimal binary classification loss of distinguishing the current policy π_θ and the expert policy π_E w.r.t. the state-action pairs sampled from ρ_θ and ρ_E To avoid potential overfitting risks, we introduce causal entropy $-H(\pi_\theta)$ as another regularization term, similar to (Ziebart et al., 2008; Ziebart, 2010). Therefore, the overall objective of our proposed POfD is formulated as

$$\min_\theta \mathcal{L} = -\eta(\pi_\theta) - \lambda_2 H(\pi_\theta) + \lambda_1 \sup \mathbb{E}_{\pi_\theta}[\log(D(s,a))] + \mathbb{E}_{\pi_E}[\log(1 - D(s,a))]$$
$$D \in (0,1)^{S \times \mathcal{A}}$$

It is actually a minimax problem closely related to the learning target of Generative Adversarial Networks (GANs) . GANs aim to train a generative model G to produce samples indistinguishable from the ones from real distributions for a well-trained discriminative model D. In our case, the true distribution is the expert policy $\pi_E(a|s)$, or equivalently the expert occupancy measure $\rho_E(s,a)$. The generator to learn is the policy model $\pi_\theta(a|s)$. Suppose D is parameterized by w. By labeling expert state-action pairs as true ("1") and policy state-action pairs as false ("0"), we get the following objective,

$$\begin{array}{c}\min_\theta \max_w \mathcal{L} = -\eta\left(\pi_\theta\right) - \lambda_2 H\left(\pi_\theta\right) + \\ \lambda_1\left(\mathbb{E}_{\pi_\theta}\left[\log\left(D_w(s,a)\right)\right] + \mathbb{E}_{\pi_E}\left[\log\left(1-D_w(s,a)\right)\right]\right)\end{array}, \quad \textbf{(6)}$$

Moreover, the minimax objective (6) can be simplified by substituting Eqn. (1) and Eqn. (2) into it, resulting in dynamic reward reshaping mechanism over the original reward signal:

$$\begin{array}{c}\min_\theta \max_w -\mathbb{E}_{\pi_\theta}\left[r'(s,a)\right] - \lambda_2 H\left(\pi_\theta\right) \\ + \lambda_1 \mathbb{E}_{\pi_E}\left[\log\left(1-D_w(s,a)\right)\right]\end{array}, \quad \textbf{(7)}$$

where $r'(s,a) = r(a,b) - \lambda_1 \log\left(D_w(s,a)\right)$ is the reshaped reward function. This function augments the environment reward with demonstration information. When the environment feedback is sparse or exploration is insufficient, the augmented reward can force the policy to generate similar trajectories as the expert policy π_E. In other words, the divergence of π and π_E is minimized. Therefore, our algorithm is able to explore the environment more efficiently. The above objective can be optimized efficiently by alternately updating policy parameters θ and discriminator parameters w. First, trajectories are sampled by executing the current policy π_θ and mixed with demonstration data to train the discriminator by SGD. The gradient is given by

$$\mathbb{E}_\pi\left[\nabla_w \log\left(D_w(s,a)\right)\right] + \mathbb{E}_{\pi_E}\left[\nabla_w \log\left(1-D_w(s,a)\right)\right]$$

Then, fixing the discriminator D_w, i.e., a fixed reward function $r'(s,a)$, the policy π_θ is optimized with a chosen policy gradient method. In particular, by policy gradient theorem (Sutton et al., 2000), the reshaped policy gradient is:

$$\begin{array}{c}\nabla_\theta \mathbb{E}_{\pi_\theta}\left[r'(s,a)\right] = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s) Q'(s,a)\right] \\ \text{where} \quad Q'(\bar{s},\bar{a}) = \mathbb{E}_{\pi_\theta}\left[r'(s,a)|s_0=\bar{s}, a_0=\bar{a}\right]\end{array}$$

The gradient for causal entropy regularization is given by

$$\nabla_\theta \mathbb{E}_{\pi_\theta}\left[-\log \pi_\theta(a|s)\right] = \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(a|s) Q^H(s,a)\right]$$

$$\text{where } Q^H(\bar{s},\bar{a}) = \mathbb{E}_{\pi_\theta}\left[-\log \pi_\theta(a|s)|s_0=\bar{s}, a_0=\bar{a}\right]$$

The optimization details are summarized in Alg. 1. It is compatible with any policy gradient methods, $e.g.$, TRPO (Schulman et al., 2015) and PPO (Schulman et al., 2017).

$$\begin{array}{c}J_{\mathrm{DQN}} = \mathbb{E}\left[\left(R_t(n) - Q_w\left(s_t, a_t\right)\right)^2\right], \text{ where} \\ R_t(n) = \sum_{i=t}^{t+n-1} \gamma^{i-t} r_i + \max_a \gamma^n Q_w\left(s_{t+n}, a\right)\end{array}$$

Algorithm 1 Policy optimization with demonstrations

```
Input: Expert demonstrations $\mathcal{D}_E = \left\{\tau_1^E, \ldots, \tau_N^E\right\}$ , initial policy and discriminator parameters $\theta_0$ and $w_0$
, regularization weights $\lambda_1, \lambda_2$ , maximal iterations $I$ .
for $i = 1$ to $I$ do
Sample trajectories $\mathcal{D}_i = \{\tau\}, \tau \sim \pi_{\theta_i}$ .
Sample expert trajectories $\mathcal{D}_i^E \subset \mathcal{D}^E$ .
Update discriminator parameters from $w_i$ to $w_{i+1}$ with the gradient
$\hat{\mathbb{E}}_{\mathcal{D}_i}\left[\nabla_w \log\left(D_w(s,a)\right)\right] + \hat{\mathbb{E}}_{\mathcal{D}_i^E}\left[\nabla_w \log\left(1-D_w(s,a)\right)\right]$
Update the rewards in $\mathcal{D}_i$ with
$r'(s,a) = r(a,b) - \lambda_1 \log\left(D_{w_i}(s,a)\right), \forall(s,a,r) \in \mathcal{D}_i$
Update the policy with policy gradient method (e . g ., TRPO, PPO) using the following gradient
    $\hat{\mathbb{E}}_{\mathcal{D}_i}\left[\nabla_\theta \log \pi_\theta(a|s) Q'(s,a)\right] - \lambda_2 \nabla_\theta H\left(\pi_{\theta_i}\right)$
end for
```

- The optimization problem of interest

DQfD and DDPGfD are two latest LfD methods. They both leverage demonstrations to aid exploration in RL, aiming to improve RL in terms of either convergence speed or performance. Here we provide a new perspective that interprets DQfD and DDPGfD through occupancy measure matching, which thus connects them with our POfD.

DQfD The DQfD method is built upon Deep Q-Networks (DQN) . As a Q-learning algorithm, DQN models the Q value with a w -parameterized neural network $Q_w(s,a)$. The objective for optimizing Q_w is

$$\begin{array}{c} J_{\mathrm{DQN}} = \mathbb{E}\left[\left(R_t(n) - Q_w\left(s_t, a_t\right)\right)^2\right], \text{ where} \\ R_t(n) = \sum_{i=t}^{t+n-1} \gamma^{i-t} r_i + \max_a \gamma^n Q_w\left(s_{t+n}, a\right) \end{array}, \mathbf{(8)}$$

DQfD takes advantage of demonstration data by putting them into a replay memory $\mathcal{D}$ and keeping them throughout the Q-learning process. Here we show minimizing Eqn. with expert demonstration and self-generated off-policy data is actually equivalent to imposing an occupancy measure matching regularization to the original DQN objective [1]. Let $\mathcal{D}^E$ denote the replay memory containing expert data only. Then the objective (8) for DQfD can be separated as

$$J_{\mathrm{DQfD}} = \hat{\mathbb{E}}_{\mathcal{D}}\left[\left(R_t(n) - Q_w\left(s_t, a_t\right)\right)^2\right] + \alpha \hat{\mathbb{E}}_{\mathcal{D}^E}\left[\left(R_t(n) - Q_w\left(s_t, a_t\right)\right)^2\right] \quad, \mathbf{(9)}$$

where α is specified by the ratio of samples from $\mathcal{D}$ and $\mathcal{D}^E$ Based on Eqn.(2),

$$\begin{aligned} Q_w(s,a) = \mathbb{E}\left[r(\cdot,\cdot)|s_0 = s, a_0 = a\right] = \rho_\pi(s,a) r(s,a), \\ \text{and } Q^E(s,a) = \rho_E(s,a) r(s,a). \end{aligned}$$

$$\text{Thus,} \hat{\mathbb{E}}_{\mathcal{D}^E}\left[\left(R_t(n) - Q_w\left(s_t, a_t\right)\right)^2\right] \quad = \hat{\mathbb{E}}_{\mathcal{D}^E}\left[\left(\hat{\rho}_E(s,a) - \rho_\pi(s,a)\right)^2 r^2(s,a)\right], \mathbf{(10)}$$

which can be interpreted as a regularization forcing current policy's occupancy measure to match the expert's empirical occupancy measure, weighted by the potential reward we will get from that state-action pair. For high reward stateaction pairs, there is a greater penalty on the discrepancy between $\hat{\rho}_E$ and ρ_π DDPGfD Similar to DQfD, DDPGfD leverages the demonstration data by putting them into the replay memory. But DDPGfD is based on an actor-critic framework (Lillicrap et al., 2015). Aside from a Q-network Q_w, DDPGfD introduces another policy network parameterized by θ to model a deterministic policy $\pi_\theta(s)$. Its Q-network is optimized off-policy with Eqn. (8), while its policy network is optimized directly with the following gradient of Q -value $Q_w(s,a)$ w.r.t. the action $a = \pi_\theta(s)$:

$$\nabla_\theta J_{\mathrm{DDPGFD}} \approx \mathbb{E}_{s,a}\left[\nabla_a Q_w(s,a) \nabla_\theta \pi_\theta(s)\right], a = \pi_\theta(s)$$

The above equation shows that the update of policy π_θ is not directly dependent on the demonstration data $\mathcal{D}_E$ but depends on the learned Q-network Q_w solely. since DDPGfD shares the same objective function for Q_w as DQfD, as well as the same way of leveraging demonstrations as shown in Eqn. (9) and Eqn. (10). Thus we can draw a similar conclusion, i.e. demonstrations in DDPGfD induce an occupancy measure matching regularization.

Although the above replay memory based LfD methods can benefit RL algorithms to some extent in sparse-reward environments, they can not sufficiently exploit the demonstration data due to following limitations.First, such a paradigm utilizes expert trajectories only by treating them as learning reference, whose effect may be significantly underexploited when demonstrations are few, as verified by our experiments. Second, to be compatible with collected data during training, the demonstrated trajectories are required to be associated with rewards for each state transition. However, the rewards in demonstrations may differ from the ones used for learning the policy in the current environment (Ziebart et al., 2008), or they may be unavailable.

By reformulating the original objective (9) into (10), we find that, instead of mixing expert data with self-generated data, putting an occupancy measure matching regularization can avoid the requirement of rewards in demonstrations, as what POfD does. In this way, LfD will no longer be restricted to the off-policy RL setting.

3 Theoretical Analysis

Please present the theoretical analysis in this section. Moreover, please formally state the major theoretical results using theorem/proposition/corollary/lemma environments. Also, please clearly highlight your new proofs or extensions (if any).

we aim at investigating

1) whether POfD can aid exploration when provided with a few demonstrations, even though the demonstrations are imperfect, and

2) whether POfD can succeed and achieve high empirical return, especially when feedback of the environment is extremely sparse.

To comprehensively assess our method, we conduct extensive experiments on eight widely used physical control tasks, ranging from low-dimensional ones such as cartpole and mountain car to high-dimensional and naturally sparse environments based on OpenAI Gym and Mujoco. The specifications of environments we used are given in Table 1

Settings

In view of the property of each individual environment, we apply four ways for sparsifying their builtin dense rewards for evaluating performance of different methods in sparse-reward environments, detailed as follows.

TYPEl: a reward of +1 is given when the agent reaches the terminal state, and otherwisely 0. This is also employed by for MountainCar.

TYPE2: a reward of +1 is given when the agent survives for a while (e.g., 200 steps for CartPole).

TYPE3: a reward of +1 is given for every time the agent moves forward over a specific number of units in Mujoco environments.

TYPE4: specially designed for InvertedDoublePendulum, a reward +1 is given when the second pole stays above a specific height of 0.89. Instead of providing dense rewards to state-action pairs at every step,the environment directly tells the agent the target state, $e.g.$ reaching a goal and moving forward, through providing sparse rewards.

This is more practical and hence we do not consider extensive reward engineering. For clearer distinction, we call the original simulator dense environments, while the ones with altered reward are called sparse environments.

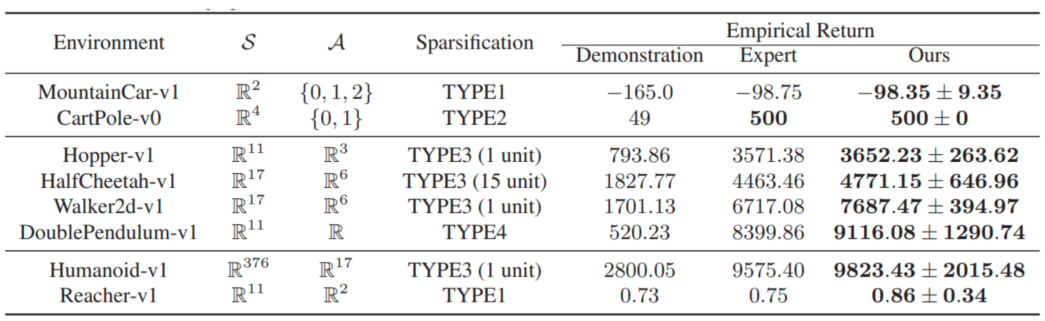
Without specification, we use only one single imperfect trajectory as demonstrations. To collect demonstrations, we train an agent insufficiently by running TRPO in the corresponding dense environment. Then, an imperfect trajectory is randomly sampled by executing the agent. Our POfD is evaluated using two metrics. First, for each sparse environment, the method is run for 5 times with different random initialization and we investigate training curves to understand how POfD facilitates exploration. Second, empirical returns in numerical values are calculated by averaging over 500 cumulated rewards for the learned policy. We compare POfD against five strong baselines, including

1) training the policy with TRPO in dense environments which is referred to as expert;

2) training the policy with TRPO in sparse environments;

3) applying GAIL to learn the policy from demonstrations, under the same setting as our POfD;

4) DQfD and 5) DDPGfD, which are the state-of-the-art LfD algorithms for discrete and continuous actions



Environment	$\mathcal{S}$	$\mathcal{A}$	Sparsification	Empirical Return		
				Demonstration	Expert	Ours
MountainCar-v1	$\mathbb{R}^2$	$\{0, 1, 2\}$	TYPE1	-165.0	-98.75	$\mathbf{-98.35 \pm 9.35}$
CartPole-v0	$\mathbb{R}^4$	$\{0, 1\}$	TYPE2	49	**500**	$\mathbf{500 \pm 0}$
Hopper-v1	$\mathbb{R}^{11}$	$\mathbb{R}^3$	TYPE3 (1 unit)	793.86	3571.38	$\mathbf{3652.23 \pm 263.62}$
HalfCheetah-v1	$\mathbb{R}^{17}$	$\mathbb{R}^6$	TYPE3 (15 unit)	1827.77	4463.46	$\mathbf{4771.15 \pm 646.96}$
Walker2d-v1	$\mathbb{R}^{17}$	$\mathbb{R}^6$	TYPE3 (1 unit)	1701.13	6717.08	$\mathbf{7687.47 \pm 394.97}$
DoublePendulum-v1	$\mathbb{R}^{11}$	$\mathbb{R}$	TYPE4	520.23	8399.86	$\mathbf{9116.08 \pm 1290.74}$
Humanoid-v1	$\mathbb{R}^{376}$	$\mathbb{R}^{17}$	TYPE3 (1 unit)	2800.05	9575.40	$\mathbf{9823.43 \pm 2015.48}$
Reacher-v1	$\mathbb{R}^{11}$	$\mathbb{R}^2$	TYPE1	0.73	0.75	$\mathbf{0.86 \pm 0.34}$

Figure 2: Environment specifications and results

My analyze

I have a supplementary explanation for the image,
column item:

S:State space dimension

A:Action space dimension

Sparsification:use which experimental method

Demonstration:training an agent insufficiently by running TRPO in the corresponding dense environment. Then, an imperfect trajectory is randomly sampled by executing the agent

Expert: use the policy with TRPO in dense environments

Ours: applying POFD to learn the policy from demonstrations

column item:

diffient environment

My analyze

Using this method (Algorithm 1), it is true that in the environment of sparse rewards, the effective of using POFD is very good, even better than the experts trained in the environment of dense rewards. POFD can be seen that in an environment based on incomplete demostration and sparse rewards, POFD can address exploration problem . I think this method is very practical for the vast unknown environment in RL.

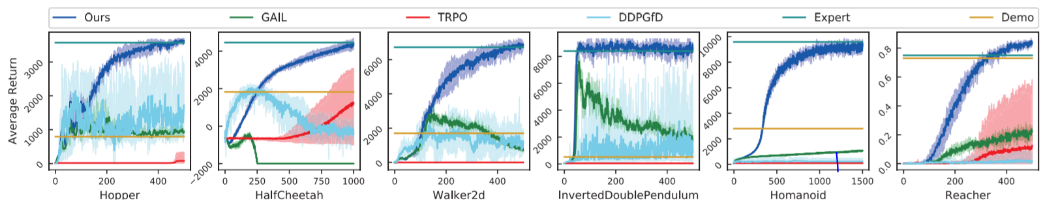


Figure 3: all methods compare

This photo more clearly points out the difference between this method and other methods. Some algorithms, ex: TRPO, cannot be trained on sparse rewards. under the training of POFD algorithms, some games even go beyond expert methods, making me think quite awesome.

4 Conclusion

Please provide succinct concluding remarks for your report. You may discuss the following aspects:

- The potential future research directions

  I think it can be used to compare the time difference between different training methods. I think time is an important factor in considering whether to use an algorithm. I can also have a benchmark to know the proportion of their execution time to verify the correctness of the results.

- Any technical limitations

  I think that this method restrict on the machine efficiency because it is necessary to explore at the same time and compare the demostrastraion to choose better one. If the machine is not good, the time cost will be high. The author's comparison chart only compares episode and reward ,and does not discuss based on time. I think this is a very limited place. If it takes twice as much time as before, I think it would be better to train these times from search(this is based on the situation where the environment can be trained).

- Any latest results on the problem of interest

  One of the disadvantages of this paper is that the sparse rewards environment here is designed by myself. In fact, I think I can find an sparse rewards environment by myself. because this paper is designed by myself,it may make people think the POFD model just fit this environment. it isn't general.