Discriminator-Weighted Offline Imitation Learning from Suboptimal Demonstrations

Haoran Xu  
JD Intelligent Cities Research  
JD Technology  
ryanxhr@gmail.com

Xianyun Zhan  
Institute for AI Industry Research  
Tsinghua University  
zhanxianyuan@gmail.com

Honglei Yin  
JD Intelligent Cities Research  
JD Technology  
yinhonglei93@gmail.com

Huiling Qin  
JD Intelligent Cities Research  
JD Technology  
orekinana@gmail.com

Abstract

We study the problem of offline Imitation Learning (IL) where an agent aims to learn an optimal expert behavior policy without additional online environment interactions. Instead, the agent is provided with a static offline dataset of state-action-next state transition triples from both optimal and non-optimal behaviors. This strictly offline imitation learning problem arises in many real-world problems, where environment interactions and expert annotations are costly. Prior works that address the problem either require that expert data occupies the majority proportion of the offline dataset, or need to learn a reward function and perform offline reinforcement learning (RL) based on the learned reward function. In this paper, we propose an imitation learning algorithm to address the problem without additional steps of reward learning and offline RL training for the case when demonstrations containing large proportion of suboptimal data. Built upon behavioral cloning (BC), we introduce an additional discriminator to distinguish expert and non-expert data, we propose a cooperation strategy to boost the performance of both tasks, this will result in a new policy learning objective and surprisingly, we find its equivalence to a generalized BC objective, where the outputs of discriminator serve as the weights of the BC loss function. Experimental results show that our proposed algorithm achieves higher returns and faster training speed compared to baseline algorithms.

1 Introduction

The recent success of reinforcement learning (RL) in many domains showcases the great potential of applying this family of learning methods to real-world applications. A key prerequisite for RL is to design a reward function that specifies what kind of agent behavior is preferred. However, in many real-world applications, designing a reward function is prohibitively difficult [Ng et al. 1999, Irpan 2018]. By contrast, imitation learning (IL) provides a much easier way to leverage the reward function implicitly from the collected demonstrations and has achieved great success in many sequential decision making problems [Pomerleau 1989, Ng et al. 2000, Ho and Ermon 2016].

However, popular IL methods such as behavioral cloning (BC) and generative adversarial imitation learning (GAIL) [Ho and Ermon 2016], assume the expert demonstration is optimal. Unfortunately, it is often difficult to obtain optimal demonstrations for many real-world tasks, because human experts often make mistakes due to various reasons, such as the difficulty of the task, partial observability of the environment, or the presence of distraction. Given such noisy expert demonstrations, which
We consider the standard fully observed Markov Decision Process (MDP) setting \cite{Sutton1998}. Current methods that deal with suboptimal demonstrations either require additional labels, which can be done explicitly by annotating each demonstration with confidence scores by human experts \cite{Wu2019}, or implicitly by ranking noisy demonstrations according to their relative performance through interacting with the environment \cite{Brown2019, Zolna2020}. However, human annotation and environment interaction are laborious and expensive in real-world settings, such as in medicine, healthcare, and industrial processes.

In this work, we investigate a pure offline learning setting where the agent has access to neither the expert nor the environment for additional information. The agent, instead, has only access to a small pre-collected dataset of state-action-next state transition triples sampled from the expert and a large batch offline dataset sampled from one or multiple behavior policies that could be highly sub-optimal. Prior works that address the problem are based on variants of BC or inverse RL. \cite{Sasaki2021} reuse another policy learned by BC as the weight of original BC objective, however, this requires that expert data occupy the majority proportion of the offline dataset, otherwise the policy will be misguided to imitate the suboptimal data. \cite{Zolna2020} first learns a reward function that prioritizes expert data over others and then performs offline RL based on this reward function. This algorithm is extremely expensive to run, requiring solving offline RL in an inner loop, which itself is a challenging problem and prone to training instability \cite{Kumar2019} and hyperparameter sensitivity \cite{Wu2019b}.

In this paper, we propose an offline imitation learning algorithm to learn from demonstrations that contain records of both optimal and non-optimal behaviors, BC and GAIL all fail to imitate the optimal policy \cite{Wu2019, Ma2020}. Current methods that deal with suboptimal demonstrations require additional steps of reward learning and offline RL training. Built upon the task of behavioral cloning (BC), we introduce an additional task to learn a discriminator to distinguish expert and non-expert data, we propose a cooperation strategy to boost the performance of both tasks. This results in a new policy learning objective and surprisingly, we find its equivalence to a generalized BC objective, where the outputs of the discriminator serve as (potentially stochastic and not necessarily optimal). Concretely, we have an expert dataset in the form \( \{s_i, a_i, s'_i\}_{i=1}^{n_e} \) where \((s, a)\) is sampled from distribution \(d^\pi\) and \(s'\) is sampled from \(\rho(s, a)\).

In our problem setting, we also have an offline static dataset consisting of i.i.d tuples \( \{s_i, a_i, s'_i\}_{i=1}^{n_a} \) s.t. \((s, a)\) \( \sim \rho(s, a), s' \sim P(s, a)\), where \(\rho \in \Delta(S \times A)\) is an offline state-action distribution resulting from some other behavior policies. Note that these behavior policies could be much worse than the expert \(\pi_e\). Our goal is to only leverage the offline batch data \(D_b = D_e \cup D_a\) to learn an optimal policy \(\pi\) with regard to optimizing the ground truth reward \(r\), without any interaction with the environment or expert.

### 2.2 A Generalized Behavioral Cloning Objective

In order to discard low-quality demonstrations and only clone the best behavior available, previous works \cite{Sasaki2021, Zolna2020} consider a generalized behavioral cloning
objective to imitate demonstrations unequally, that is,
\[
\min_{\pi} \mathbb{E}_{(s,a) \sim D_o} [-\log \pi(a|s) \cdot f(s,a)]
\]
(1)
where \( f : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1] \) denotes an arbitrary weight function.

- If \( f(s,a) = 1 \) for \( \forall (s,a) \in \mathcal{S} \times \mathcal{A} \), the objective \((1)\) corresponds to the vanilla BC objective.
- If \( f(s,a) = \pi'(a|s) \), where \( \pi' \) is an old policy which was previously optimized with \( D_o \), the objective \((1)\) corresponds to the objective of Behavioral Cloning from Noisy Demonstrations [Sasaki and Yamashina 2021]. Since \( \sum_a \pi'(a|s) = 1 \) for \( \forall s \in \mathcal{S} \) is satisfied, \( \pi'(a|s) \) can be interpreted as the weights for the weighted action sampling.
- If \( f(s,a) = \mathbb{I}[A^e(s,a)] \), where \( \mathbb{I} \) is the indicator function which creates a boolean mask that eliminates samples which are thought to be worse than the current policy, the objective \((1)\) corresponds to the objective of Offline Reinforced Imitation Learning [Zolna et al. 2020a].

The objective \((1)\) can also be deemed as the objective of Soft Q Imitation Learning [Reddy et al. 2019] with \( f(s,a) = 1 \) for \( (s,a) \in D_e \) and \( f(s,a) = 0 \) for \( (s,a) \in D_o \) in online IL literature; or the objective of off-policy actor-critic (Off-PAC) algorithm [Degris et al. 2012] with \( f(s,a) = Q^e(s,a) \cdot \pi(a|s)/\pi_o(a|s) \) in online RL literature.

3 Discriminator-Weighted Behavioral Cloning

We now continue to describe our approach for offline imitation learning from demonstrations that (perhaps) contain large-proportional suboptimal data without additional steps of reward learning and offline RL training. Built upon the task of BC, we introduce an additional task to learn a discriminator to distinguish expert and non-expert data, we propose a cooperation strategy to boost the performance of both tasks, this will result in a new generalized BC objective. We then provide the interpretation of weights in our generalized BC objective, this gives the intuition about why our method can work.

3.1 Learn the policy and discriminator separately

It is obvious that we can avoid the negative impact of suboptimal demonstrations presented in \( D_o \) by only imitating \( D_e \), which can be written as
\[
\min_{\pi} \mathbb{E}_{(s,a) \sim D_e} [-\log \pi(a|s)]
\]
(2)
We call the task of learning a policy using objective \((2)\) as the BC task. The drawback of BC task is that the learned policy may not be able to generalize due to the potential limited size and state coverage of \( D_e \). It does not fully utilize the information from \( D_o \). If we can select those high-reward transitions from \( D_o \) and combine them with \( D_e \), we are expected to get a better policy.

Now let’s consider another different task, which aims to learn a discriminator by contrasting expert and non-expert transitions, given by
\[
\min_{d} \mathbb{E}_{(s,a) \sim D_e} [-\log d(s,a)] + \mathbb{E}_{(s,a) \sim D_o} [-\log(1 - d(s,a))]
\]
(3)
We call this the discriminating task, which is similar to how the discriminator is trained in GAIL [Ho and Ermon 2016] and GAN [Goodfellow et al. 2014a], except that the second term is sampled from an offline dataset instead of online samples induced by the policy.

However, optimizing objective \((3)\) will make the discriminator learned to assign 1 to all transitions from \( D_e \) and 0 to all transitions from \( D_o \). This limiting behavior is unsatisfactory because \( D_o \) can contain some successful (high-reward) transitions. This bears similarity to the positive-unlabeled (PU) classification problem [Elkan and Noto, 2008], where both positive and negative samples exist in the unlabeled data.

To solve this problem, previous works [Du Plessis et al., 2015; Xu and Denil, 2019; Zolna et al., 2020a] re-weight different losses for positive and unlabeled data, in order to obtain an estimate of
model loss on negative samples that is not directly available. Applying PU learning to objective (3) yields the following objective:

$$\min_\eta \mathbb{E}_{(s,a) \sim D_e} [-\log d(s,a)] + \mathbb{E}_{(s,a) \sim D_o} [-\log (1 - d(s,a))] - \eta \mathbb{E}_{(s,a) \sim D_e} [-\log (1 - d(s,a))]$$ (4)

where \(\eta\) is a hyperparameter, corresponds to the proportion of positive samples to unlabeled samples. Intuitively, the second term in (4) could make \(d(s,a)\) of state-action pairs from \(D_e\) become 0 if similar state-action pairs are included in \(D_o\), and the third term in (4) balances the impact of the second term, i.e., avoids \(d(s,a)\) of state-action pairs from \(D_e\) becoming 0.

To summarize, the BC task aims to imitate the expert behavior from \(D_e\), but ignores the valuable information in \(D_o\); the discriminating task aims to contrast expert and non-expert transitions from \(D_e\) and \(D_o\), by only using state-action information as input. Both tasks lack enough information to improve their own performance, which, however, can be obtained from the other task. It then seems natural to find a scheme to incorporate the policy into the training of the discriminator and effectively use the discriminator to help the training of the policy.

### 3.2 Learn the policy and discriminator cooperatively

We propose a cooperation strategy to boost the performance of both tasks. To boost the performance of the discriminating task, we add the imitation information from \(\pi\) to the input of the discriminator, yielding the following discriminator learning objective \(L_d\) to be minimized as:

$$\min_\pi L_d = \min_\pi \mathbb{E}_{(s,a) \sim D_e} [-\log d(s,a, \log \pi(a|s))] +$$

$$\mathbb{E}_{(s,a) \sim D_o} [-\log (1 - d(s,a, \log \pi(a|s)))] - \eta \mathbb{E}_{(s,a) \sim D_e} [-\log (1 - d(s,a, \log \pi(a|s)))]$$ (5)

Supposed \(\pi\) is learned to be optimal, i.e., assigns large probabilities to expert actions in expert states, the discriminator will receive additional learning signal. It will be easier for the discriminator to contrast expert and non-expert transitions from \(D_e\) and \(D_o\), by only using state-action information as input. Without this information from \(\pi\), the discriminator is much harder to learn by only using information from \((s,a)\).

Now let’s dive deeper into objective (5). As \(\pi\) now appears in the input of the discriminator \(d\), hence both \(d\) and its loss \(L_d\) become functionals of \(\pi\). We are interested to see how does the imitation information from \(\log \pi\) affect \(L_d\), and further impact \(d\). In other words, given the current \(d\), we want to explore how to change the behavior of \(\pi\) such that \(d\) can be better learned. To achieve this goal, we define a functional \(J(\pi)\) for \(L_d\) with the function \(\pi\) as the variable, and fix the parameter \(\theta_d\) of \(d\) to exclude its own influence. Note that in the functional \(J(\pi)\), \(d\) is now fixed and its parameter \(\theta_d\) will no longer be considered as a variable. \(J(\pi)\) can be formally define as the following integral form to further eliminate the effect of changes in \(s, a\) on \(d\):

$$J(\pi) = \int \int \frac{\partial L_d(s,a,d, \log \pi)}{\partial d(s,a, \log \pi)} ds da = \int \int F(s,a,\pi,\pi') ds da,$$ (6)

where we denote \(F = \partial L_d / \partial d\) and \(F\) is assumed to be continuously differentiable with respect to \(s, a, \pi\) and \(\pi'\) (derivative of \(\pi\)). To more robustly learn the discriminator \(d\), inspired by the idea of adversarial learning [Lowd and Meek, 2005], we make the policy \(\pi\) challenge the discriminator \(d\) by doing the opposite to the task of \(d\) (i.e., minimize \(L_d\)), in other words, we want \(\pi\) to maximize \(L_d\) under current \(d\). By doing so, the policy will provide as little information in \(\log \pi\) as possible such that minimizing \(L_d\) becomes harder for the discriminator, this can be seen as minimizing the worst-case error [Carlini et al., 2019] [Fawzi et al., 2016] [Goodfellow et al., 2014b], which makes the robustness of the discriminator significantly improved.

To maximize \(L_d\) for \(\pi\) under current \(d\), we let the functional \(J(\pi)\) attains its maxima with respect to \(\pi\). According to the calculus of variations [Gelfand et al., 2000], the extrema (maxima or minima) of functional \(J(\pi)\) can be obtained by finding a function \(\pi\) such that the functional derivative of \(J(\pi)\) is equal to zero. We can show with theoretical derivation (see Appendix A), that ensuring the functional derivative of \(J(\pi)\) equal to zero requires \(\partial F / \partial \theta_\pi = 0\), where \(\theta_\pi\) is the parameters of \(\pi\).
A keen reader may find the similarity of our approach to the adversarial learning in GAN \cite{Goodfellow2014}. We thus term our algorithm Discriminator-Weighted Behavioral Cloning (DWBC).

Above behavioral cloning weights induce different behaviors on the imitation of transitions from \( \mathcal{D}_e \) and \( \mathcal{D}_o \). Supposed \( d \) is learned to be optimal, i.e., assigns large values (close to 1) to expert transitions and small values to non-expert transitions (close to 0). The weight of those expert transitions in \( \mathcal{D}_e \) will become large while the weight of those non-expert transitions will become small. For transitions in \( \mathcal{D}_e \), their weights can be adjusted by tuning the parameter \( \alpha \). Note that even if the discriminator is learned to be totally wrong (i.e., assign small values to expert transitions and large values to non-expert transitions), which may occur at the very beginning of training, the behavior cloning weights \( \alpha - \frac{\eta}{d(1-d)} (\alpha \geq 1, \eta < 1) \) will not be drastically changed under value clipping. This means that the policy can still learn from the expert dataset \( \mathcal{D}_e \). Even though the weight for \( \mathcal{D}_e \) is temporarily incorrect, it will be corrected as the discriminator becomes better and better.

Eq. (8) implies that our approach is also a variant of generalized behavioral cloning objective, as discussed in section 2.2, but uses a different form of weights. Unlike Offline Reinforced Imitation Learning \cite{Zolna2020}, which uses the discriminator as the reward and learns a value function as the weight, our approach uses the discriminator outputs directly as the weight. This can greatly reduce the training time and avoid the overestimation issue in estimating the value function offline \cite{Kumar2019}. We thus term our algorithm Discriminator-Weighted Behavioral Cloning (DWBC).

A keen reader may find the similarity of our approach to the adversarial learning in GAN \cite{Goodfellow2014}, which makes the generator (in our case is the policy \( \pi \)) and the discriminator \( d \) learn
against each other, thus improves the performance of both tasks. However, the learning strategy of DWBC has several differences compared with GAN and is also much easier to learn. In contrast to the fully adversarial setting in GAN, DWBC adopts a semi-cooperative strategy that shares information between the policy \( \pi \) and the discriminator \( d \) to improve the performance of both parties. Moreover, GAN needs to solve a min-max optimization problem and is known to suffer from training instability and issues such as mode collapse [Arjovsky et al., 2017]. In DWBC, we solve the min-max optimization problem implicitly by transforming the max optimization problem to a new learning loss imposed on the policy. We present an illustration of the learning mechanism of DWBC in Figure 1.

4 Related Work

4.1 Offline Imitation Learning

Offline IL, which has not received considerable attention, is a promising area because it makes IL more practical to satisfy critical safety desiderata. Offline IL methods can be typically folded into two paradigms: Behavioral Cloning (BC) and Offline Inverse Reinforcement Learning (Offline IRL). BC [Pomerleau, 1989] is the simplest IL method that can be used in the offline setting, it considers the policy as a conditional distribution \( \pi(\cdot|s) \) over actions, recent work [Florence et al., 2021] enhances BC by using energy-based models [LeCun et al., 2006]. BC has shown to have no inferior performance compared to popular IL algorithms such as GAIL [Ho and Ermon, 2016] when clean expert demonstrations are available [Ma, 2020]. Unlike BC, offline IRL considers matching the state-action distribution induced by the expert policy, this can be achieved implicitly by adversarial training or explicitly by learning a reward function. Offline IRL algorithms based on adversarial training [Kostrikov et al., 2019, Sun et al., 2021, Swamy et al., 2021, Jarbou and Perchet, 2021] use Integral Probability Metrics (IPMs) [Sriperumbudur et al., 2009] as a distance measure to solve the dual problem. They introduce a discriminator and aim to find the saddle point of a min-max optimization problem, like GAN [Goodfellow et al., 2014a]. Jarrett et al. [2020] avoids the need of min-max problem by fixing the policy to be energy-based models, in such case the KL divergence from the demonstrator’s state-action distribution to that of the policy can be computed in closed form. However, recent work finds several fundamental mathematical misconceptions in their proposed approach and we refer the reader to Swamy et al. [2021] for more details.

The common problem of these works is that they imitate equally to all demonstrations, this will hinder the performance if the demonstrations contain suboptimal data. To solve this, Sasaki and Yamashina [2021] reuses another policy learned by BC as the weight of original BC objective, however, this requires that expert data occupies the majority proportion of the offline dataset, otherwise the policy will be misguided to imitate the suboptimal data. Zolna et al. [2020a] and Konyushkova et al. [2020] first construct a reward function that discriminates expert and exploratory trajectories, then use it to solve an offline RL problem. Instead of the adversarial learning scheme, the reward function can also be learned by cascading to two supervised learning steps [Klein et al., 2013]. However, offline IRL based on reward learning is extremely expensive to run, requiring solving offline RL in an inner loop, which itself is a challenging problem and prone to training instability [Kumar et al., 2019] and hyperparameter sensitivity [Wu et al., 2019b]. Our algorithm can be seen as a combination of these two algorithms in that we use a generalized BC objective to imitate demonstrations selectively and we train a discriminator to distinguish expert and non-expert data and use the output of the discriminator as the weight of the generalized BC objective. There is also one recent work [Chang et al., 2021] that tries to solve the offline IL problem by adopting techniques from pessimistic model-based offline policy learning [Yu et al., 2020, 2021].

4.2 Offline Reinforcement Learning

One research area highly related to offline IL is offline RL [Lange et al., 2012, Levine et al., 2020], which considers performing effective RL by utilizing arbitrary given, static offline datasets, without any further environment interactions. Note that in offline RL, the training dataset is allowed to have non-optimal trajectories and the reward for each state-action-next state transition triple is known.

Our algorithm draws connection to a branch of methods in offline RL literature that performs "filtered" behavioral cloning explicitly or implicitly. More specifically, Peng et al. [2019], Nair et al. [2020] and Wang et al. [2020] estimate an advantage function, which represents the change in expected return...
when taking action $a$ instead of following the current policy, and perform weighted regression based on the advantage function, defined as $L_\pi = \mathbb{E}_{(s,a)\sim D_e}[-\log \pi(a|s) \cdot f(A_\pi(s,a))]$. The advantage $A_\pi$ can be estimated by Monte-Carlo methods [Schulman et al., 2017; Peng et al., 2019] or Q-value based methods [Schulman et al., 2015; Nair et al., 2020]. The filter function $f$ can be a binary filter [Wang et al., 2020] or an exponential filter [Peng et al., 2019; Nair et al., 2020].

While Chen et al. [2021] and Janner et al. [2021] perform filtered behavioral cloning more implicitly. They abstract offline RL as a sequence modeling problem and use Transformer architecture [Vaswani et al., 2017] to perform credit assignments directly via self-attention. Owing to the memorization power of Transformer in capturing long-term dependencies across timesteps, these methods are able to discard low-quality transitions, do behavior cloning only on high-reward transitions across different trajectories and stitch them together to obtain an optimal trajectory.

5 Experiments

We present empirical evaluations of DWBC in a variety of settings. We start with describing our experimental setup, considered datasets and baseline algorithms. Then we evaluate DWBC against other baseline algorithms on a range of robotic locomotion tasks with different types of datasets. Finally, we analyze the property of the discriminator. We show that a well-trained discriminator can be used to perform offline policy selection [Fu et al., 2020a], which is of independent interest.

5.1 Settings

We construct experiments on both widely-used D4RL MuJoCo datasets and more complex Adroit hand manipulation environment [Fu et al., 2020b]. To verify the effectiveness of our methods, we use three setting to generate $D_e$ and $D_o$. Note that we use ground truth rewards only to perform this data split step and we discard the rewards afterward.

- In setting 1, we use mixed datasets in Mujoco environments. We sort from high to low of all trajectories based on the total reward summed over the entire trajectory. We define a trajectory as well-performing if it is among the top 20% of all trajectories. We then sample every $X^{th}$ trajectory from the well-performing trajectories to constitute $D_e$ and use the remaining trajectories in the dataset to constitute $D_o$. Note that with $X$ becomes larger, $D_o$ will contain more proportion of well-performing data.
- In setting 2, we use expert and random datasets in Mujoco environments. We first sample 10 trajectories from expert datasets and 1000 trajectories from random datasets. We then random sample $X$ trajectories from those 10 expert trajectories and combine them with those 1000 random trajectories to constitute $D_o$, we use the remaining $10 - X$ trajectories to constitute $D_e$.
- In setting 3, we use human datasets in Adroit environments. We use the same procedure to constitute $D_e$ and $D_o$ as in setting 1.

We list all datasets used in this paper and the number of trajectories and transitions in $D_e$ and $D_o$ in Appendix C, different $X$ is labeled after the dataset name.

5.2 Baseline and ablated algorithms

We compare DWBC with the following baseline algorithms:

- **BC-exp & BC-all:** Behavioral cloning on expert data or on all data. BC-exp is trained only on $D_e$, hence it is not exposed to low-performing transitions in $D_o$. On the other hand, it may not be able to generalize due to the limited size of $D_e$. BC-all may generalize better than BC-pos due to access to a much larger dataset, but its performance may be negatively impacted by the low-quality data in $D_o$.
- **BCND:** BCND is trained on all data, it reuses another policy learned by BC as the weight of BC, its performance will be worse if the suboptimal data occupies the major part of the offline dataset.
- **ORIL:** ORIL learns a reward function and uses it to solve an offline RL problem. It suffers from large computational costs and the difficulty of performing offline RL under distributional shift.
- **DWBC-noads:** We include one ablation of DWBC that trains the discriminator without the adversarial learning machinery (i.e., no $\log \pi$ as input), with all others remaining the same as DWBC.
Table 1: Results for Mujoco and Adroit datasets. Scores are undiscounted average returns of the policy at the last iteration of training, averaged over 5 random seeds. We bold the highest values.

<table>
<thead>
<tr>
<th>Task name</th>
<th>BC-exp</th>
<th>BC-all</th>
<th>BCND</th>
<th>ORIL</th>
<th>DWBC-noads</th>
<th>DWBC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hopper</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mixed-2</td>
<td>1547</td>
<td>811</td>
<td>437</td>
<td>1345</td>
<td>2450</td>
<td>2531</td>
</tr>
<tr>
<td>mixed-5</td>
<td>1263</td>
<td>811</td>
<td>437</td>
<td>998</td>
<td>2271</td>
<td>2451</td>
</tr>
<tr>
<td>mixed-10</td>
<td>1458</td>
<td>811</td>
<td>437</td>
<td>1489</td>
<td>1798</td>
<td>2231</td>
</tr>
<tr>
<td>exp-rand-3</td>
<td>1200</td>
<td>314</td>
<td>52</td>
<td>49</td>
<td>1531</td>
<td>2231</td>
</tr>
<tr>
<td>exp-rand-6</td>
<td>1070</td>
<td>314</td>
<td>52</td>
<td>51</td>
<td>1604</td>
<td>1610</td>
</tr>
<tr>
<td>HalfCheetah</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mixed-2</td>
<td>4451</td>
<td>4210</td>
<td>4456</td>
<td>/</td>
<td>4980</td>
<td>5011</td>
</tr>
<tr>
<td>mixed-5</td>
<td>4553</td>
<td>4210</td>
<td>4456</td>
<td>44</td>
<td>5011</td>
<td>5018</td>
</tr>
<tr>
<td>mixed-10</td>
<td>4358</td>
<td>4210</td>
<td>4456</td>
<td>989</td>
<td>5022</td>
<td>5017</td>
</tr>
<tr>
<td>exp-rand-3</td>
<td>6072</td>
<td>5753</td>
<td>6007</td>
<td>6013</td>
<td>6021</td>
<td>6107</td>
</tr>
<tr>
<td>exp-rand-6</td>
<td>5875</td>
<td>5753</td>
<td>6007</td>
<td>6107</td>
<td>5803</td>
<td>5955</td>
</tr>
<tr>
<td>Walker2d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mixed-2</td>
<td>2031</td>
<td>784</td>
<td>760</td>
<td>2208</td>
<td>2355</td>
<td>2436</td>
</tr>
<tr>
<td>mixed-5</td>
<td>2014</td>
<td>784</td>
<td>760</td>
<td>2481</td>
<td>3112</td>
<td>3111</td>
</tr>
<tr>
<td>mixed-10</td>
<td>1611</td>
<td>784</td>
<td>760</td>
<td>2384</td>
<td>3219</td>
<td>3258</td>
</tr>
<tr>
<td>exp-rand-3</td>
<td>3078</td>
<td>211</td>
<td>6</td>
<td>955</td>
<td>4547</td>
<td>4666</td>
</tr>
<tr>
<td>exp-rand-6</td>
<td>2871</td>
<td>211</td>
<td>6</td>
<td>5</td>
<td>3673</td>
<td>4250</td>
</tr>
<tr>
<td>Ant</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mixed-2</td>
<td>2682</td>
<td>2255</td>
<td>917</td>
<td>/</td>
<td>1050</td>
<td>2000</td>
</tr>
<tr>
<td>mixed-5</td>
<td>2381</td>
<td>2255</td>
<td>917</td>
<td>/</td>
<td>1982</td>
<td>3111</td>
</tr>
<tr>
<td>mixed-10</td>
<td>2285</td>
<td>2255</td>
<td>917</td>
<td>/</td>
<td>2310</td>
<td>3417</td>
</tr>
<tr>
<td>exp-rand-3</td>
<td>1071</td>
<td>151</td>
<td>1045</td>
<td>710</td>
<td>875</td>
<td>1230</td>
</tr>
<tr>
<td>exp-rand-6</td>
<td>870</td>
<td>151</td>
<td>1045</td>
<td>639</td>
<td>626</td>
<td>1127</td>
</tr>
<tr>
<td>Pen</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>human-2</td>
<td>1888</td>
<td>806</td>
<td>1684</td>
<td>2262</td>
<td>2571</td>
<td>2486</td>
</tr>
<tr>
<td>human-3</td>
<td>1780</td>
<td>806</td>
<td>1684</td>
<td>2487</td>
<td>2362</td>
<td>2617</td>
</tr>
<tr>
<td>human-5</td>
<td>1531</td>
<td>806</td>
<td>1684</td>
<td>2111</td>
<td>2271</td>
<td>2487</td>
</tr>
<tr>
<td>Door</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>human-2</td>
<td>45</td>
<td>31</td>
<td>-4</td>
<td>-12</td>
<td>51</td>
<td>53</td>
</tr>
<tr>
<td>human-3</td>
<td>40</td>
<td>31</td>
<td>-4</td>
<td>-50</td>
<td>-3</td>
<td>10</td>
</tr>
<tr>
<td>human-5</td>
<td>38</td>
<td>31</td>
<td>-4</td>
<td>-3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Hammer</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>human-2</td>
<td>-187</td>
<td>-230</td>
<td>-163</td>
<td>-222</td>
<td>-87</td>
<td>-88</td>
</tr>
<tr>
<td>human-3</td>
<td>-191</td>
<td>-230</td>
<td>-163</td>
<td>-237</td>
<td>-97</td>
<td>-96</td>
</tr>
<tr>
<td>human-5</td>
<td>-213</td>
<td>-230</td>
<td>-163</td>
<td>-159</td>
<td>-60</td>
<td>24</td>
</tr>
<tr>
<td>Relocate</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>human-2</td>
<td>4</td>
<td>2</td>
<td>7</td>
<td>-4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>human-3</td>
<td>3</td>
<td>2</td>
<td>7</td>
<td>-8</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>human-5</td>
<td>3</td>
<td>2</td>
<td>7</td>
<td>9</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

5.3 Comparative Evaluations

We show the comparative results in Table 1 and include the learning curves in Appendix C. It can be shown from Table 1 that DWBC outperforms baseline algorithms on most tasks (25 out of 32 tasks), especially in Mujoco datasets (18 out of 20 tasks), showing that DWBC is well suited to make effective use of the expert dataset $D_e$ and the mixed quality dataset $D_o$.

As expected, the performance of BC-exp declines as $X$ becomes larger. This is because that a larger $X$ means the number of well-performing transitions is smaller. In some datasets (e.g., Halfcheetah_exp-rand-6 and Ant_exp-rand-6), there is no clear winner between BC-exp and BC-all, which suggests that the quality of $D_o$ for the considered tasks varies. BCND performs poorly compared to other methods due to the majority of low-quality data in the mixed datasets. It usually scores below BC-all. ORIL struggles to learn in some tasks (especially in the Ant datasets), which suggests their learned reward function does not accurately contrast expert and non-expert data. We also find that the performance of ORIL tends to decrease in some tasks (e.g., Halfcheetah_mixed-5 and Ant_exp-rand-6), this "overfitting" phenomenon also occurs in the experiments of offline RL papers [Kumar et al., 2019, Wu et al., 2019b]. This is perhaps due to the limited data size and model generalization bottleneck [Neyshabur, 2017].
We also find that DWBC-noads performs worse than DWBC, especially when $D_o$ contains more expert data, under which circumstance it is hard for the discriminator to distinguish between expert and non-expert data without the help of our proposed adversarial learning machinery.

### 5.4 Additional Experiments

**Offline policy selection by the discriminator.** Offline policy selection (OPS) [Paine et al., 2020; Yang et al., 2020; Dereventsov et al., 2021] considers the problem of choosing the best policy from a set of policies given only offline data. This problem is critical in the offline settings (i.e., offline RL and offline IL) because the online execution is often very costly and safety-aware, deploying a problematic policy may damage the real-world systems [Tang and Wiens, 2021]. Note that existing offline RL/IL methods break the offline assumption by evaluating different policies corresponding to their rewards in online environment interactions. However, this online evaluation is often infeasible and hence undermines the initial assumption of offline RL/IL.

In this paper, we propose a novel offline policy selection method, simply by using our learned discriminator $d$. More specifically, as the discriminator gets state $s$, action $a$ and the log probability $\log \pi(a|s)$ as input, we can use expert state-action pairs from $D_e$ and different policy $\pi$ as input. The discriminator learned by DWBC will assign large values (close to 1) when the evaluated policy is close to the policy learned by DWBC, this means that the policy is more optimal.

To validate our proposed idea, we conduct experiments in Hopper, Walker2d and Pen environment. In Hopper and Walker2d, we use mixed-2, mixed-5 and mixed-10 datasets, in Pen, we use mixed-2, 3 and mixed-5 datasets. We compare three algorithms (BC-exp, ORIL and DWBC) trained in these datasets, total of 9 policies in each environment. We first train DWBC, then we use the learned discriminator along with $D_e$ to compute the value $d(s,a,\log \pi_i(a|s))$ of each policy $\pi_i$. We plot average $d(s,a,\log \pi_i(a|s))$ versus the policy’s true return in Figure 2. As shown, the discriminator’s values can well reflect the rank between almost every two policies. This means that we can first train a DWBC policy and then use the trained discriminator $d$ to select the best policy among our candidate policy sets, without executing them in the environment to get the actual returns.

**Comparision of run time.** We also evaluate the run time of training DWBC and other baseline algorithms for 250 thousand training steps (does not include evaluation run time cost). All run time experiments were executed on NVIDIA V100 GPUs. For a fair comparison, we use the same policy network size in BC, BCND, ORIL and DWBC. The discriminator network size is also kept the same in ORIL and DWBC. The results are reported in Figure 3. Unsurprisingly, we find the run time of our approach is only slightly more than BC, while other baselines (ORIL, BCND) are over 7 times more costly than BC. The reason that ORIL is costly to run is due to the additional effort to solve an offline RL problem. The high computation cost of BCND is due to its inner iterations of training $K$ policy ensembles ($K = 5$ in our experiment), which is also mentioned in their paper [Sasaki and Yamashina, 2021]. This demonstrates the effectiveness of DWBC by only adds a limited cost to the original BC algorithm while providing substantially improved performance.
6 Conclusion and Limitation

In this paper, we propose a new offline imitation learning algorithm that can learn from suboptimal demonstrations without environment interactions or expert annotations. Experimental results show that our algorithm achieves higher returns and faster training speed compared to baseline algorithms, under different scenarios. One limitation of our work is that since our algorithm is based on weighted BC, the covariate shift problem of BC [Ross et al., 2011] will be inherited. That is, there is no way for the policy to learn how to recover if it deviates from the behavior policy to a new state not seen in the demonstrations. In future work, we will consider modifying the main task from action matching to state-action distribution matching, which is known to be more robust to distributional shift [Kostrikov et al., 2019].

References


A Derivation Details

In this section, we provide the detailed design intuition and theoretical derivation of DWBC.

A.1 Decomposition and Reformulation of Learning Tasks

As discussed in Section 3.1, learning from both the expert dataset $D_e$ and suboptimal dataset $D_o$ implies the need of jointly solving two tasks: a BC task and a discriminator task. A straightforward solution is to learn the two tasks separately, which solves the BC task by imitating the expert demonstrations in $D_e$ and learn the discriminator via PU learning using data from both $D_e$ and $D_o$:

**BC task:** $\pi(a|s) \leftarrow \arg\min_\pi \mathcal{L}_{BC}$

**Discriminating task:** $d(s,a) \leftarrow \arg\min_d \mathcal{L}_d$

where $\mathcal{L}_{BC}$ and $\mathcal{L}_d$ are discussed and given in objectives (2) and (4) in the main article as follows:

$$\mathcal{L}_{BC} = \mathbb{E}_{(s,a) \sim D_e} [-\log \pi(a|s)]$$

$$\mathcal{L}_d = \eta \mathbb{E}_{(s,a) \sim D_e} [-\log d(s,a)] + \mathbb{E}_{(s,a) \sim D_e} [-\log(1 - d(s,a))]$$

Naïvely solving above two tasks separately is insufficient. First, the BC task only learn from the expert dataset $D_e$, fails to utilize the potential valuable information in the suboptimal dataset $D_o$. Second, both tasks lack sufficient information to improve their own performance. For example, a good discriminator could provide important information to distinguish the potential expert samples in the suboptimal dataset $D_o$, which are valuable for the learning of policy $\pi$; a well-performed policy $\pi(a|s)$ will assign large probabilities to expert actions under expert states, which could provide additional learning signal for the discriminator $d$ to more easily contrast expert and non-expert transitions in $D_o$.

There are two existing approaches can be used to jointly solve above two tasks, however, both of them have some drawbacks. One approach is to cast the problem into a multi-task style multi-objective optimization problem, by optimizing an augmented loss $\beta \mathcal{L}_{BC} + (1 - \beta) \mathcal{L}_d$, $\beta \in (0, 1)$ for both $\pi$ and $d$. The problem is that the BC task and the discriminating task are different tasks, the potential contradiction of the two tasks in certain settings may impede both tasks from achieving the best performance. Moreover, properly selecting the hyperparameter $\beta$ is very tricky. Another approach is to adopt a GAN-style model [Goodfellow et al., 2014] which treats the policy as the generator and optimize it implicitly through solving a min-max optimization problem with the discriminator loss $\mathcal{L}_d$. However, this is very costly and is known to suffer from training instability and issues such as mode collapse [Arjovsky et al., 2017]. Moreover, although we have an explicit loss function $\mathcal{L}_{BC}$ for $\pi$, it is not used in such a GAN-style model, which results in potential loss of information.

In this paper, we design an new cooperative learning mechanism to address above issues, which also results in a computationally efficient practical algorithm. It includes three key ingredients: 1) sharing information between the BC task and the discriminating task to achieve cooperative learning; 2) enabling the BC task to learn on both expert and suboptimal data by introduce an additional corrective loss $\mathcal{L}_w$ impacted by the discriminator outputs; 3) solving both tasks in fully supervised learning manner to maintain computational efficiency. In our approach, we consider following alternative formulation to establish information sharing across the two tasks and enable cooperative learning:

**BC task:** $\pi(a|s) \leftarrow \arg\min_\pi \alpha \mathcal{L}_{BC} + \mathcal{L}_w, \quad \alpha > 1$

**Discriminating task:** $d(s,a,\log \pi(a|s)) \leftarrow \arg\min_d \mathcal{L}_d$

with the new $\mathcal{L}_d$ given in objective (5) as follows:

$$\mathcal{L}_d = \eta \mathbb{E}_{(s,a) \sim D_e} [-\log d(s,a,\log \pi(a|s))] + \mathbb{E}_{(s,a) \sim D_o} [-\log(1 - d(s,a,\log \pi(a|s)))] - \eta \mathbb{E}_{(s,a) \sim D_o} [-\log(1 - d(s,a,\log \pi(a|s)))]$$

In above reformulation, we design the information provided by the policy to discriminator as the element-wise imitation loss value $\log \pi(a|s)$, and the information provided to the policy as the
addional corrective loss term $L_w$ computed using output values of the discriminator $d$ on samples from both $D_o$ and $D_o$. With the involvement of $L_w$, we can allow the policy $\pi$ learning from sub-optimal data under the guidance of the discriminator. Moreover, to more robustly learn the discriminator $d$, we borrow the idea of adversarial learning [Lowd and Meek, 2005] and make the policy $\pi$ challenge $d$ by providing as little useful information in $\log \pi(a|s)$ as possible.

We will show in the next section that with the choice of element-wise imitation loss $\log \pi(a|s)$ as the form of information and the adversarial behavior of policy $\pi$, an exact form of $L_w$ can be derived, and eventually, transforms the original BC task into a cost sensitive learning problem.

### A.2 Drivation of the Corrective Loss Term $L_w$

In this section, we resort to functional analysis and calculus of variation to derive the exact form of $L_w$. Under the reformulated problem (10), both the discriminator $d$ and its loss $L_d$ are impacted by the information provided by policy $\pi (\log \pi(a|s))$. Hence they are now functional of $\pi$. We are interested to see how the variation of $\pi$ impacts $L_d$, and further influence $d$. Moreover, we can use a specific form of $L_w$ to alter the behavior of the learned $\pi$ to achieve the desired adversarial behavior.

Denote $\theta_\pi$ and $\theta_d$ as the parameters of the policy $\pi$ and discriminator $d$ respectively. As discussed in Section 3.2, to analyze the impact of $\pi$ on $d$, we fix the parameters $\theta_d$ of $d$ to exclude its own influence.

The function of the discriminator $d(\theta_d, s, a, \log \pi(a|s))$ can thus be replaced by $d(s, a, \log \pi(a|s))$. Similarly, with $\theta_d$ fixed, we can define a functional $J(\pi)$ for $L_d$ which depends on function $\pi$ and further eliminate the effect of changes in $s, a$ on $d$ by integrating over $s$ and $a$ as in Eq. (6):

$$J(\pi) = \int \int \frac{\partial L_d(s, a, d, \log \pi(a|s))}{\partial d(s, a, \log \pi(a|s))} \, ds \, da = \int \int F(s, a, \pi, \pi') \, ds \, da,$$

where $F = \partial L_d/\partial d$ is assumed to be a continuously differentiable function with respect to $s, a, \pi$ and $\pi'$ (derivative of $\pi$). Note that as $\theta_d$ is fixed and not considered as a variable in our analysis, $\theta_d$ no longer contributes gradients in the partial derivative $\partial L_d/\partial d$, and functional $F$ is given as follows:

$$F(s, a, \pi, \pi') = - \mathbb{E}_{(s, a) \sim D_o} \left[ \frac{\eta}{d(s, a, \log \pi(a|s))} \right] + \mathbb{E}_{(s, a) \sim D_o} \left[ \frac{1}{1 - d(s, a, \log \pi(a|s))} \right] - \mathbb{E}_{(s, a) \sim D_o} \left[ \frac{\eta}{1 - d(s, a, \log \pi(a|s))} \right].$$  \hfill (11)

To enforce the adversarial behavior of $\pi$, we want to make $\pi$ challenge the discriminator $d$ by maximizing $L_d$ under current $d$ (doing the opposite to $d$). By doing so, the policy will provide as little information in $\log \pi$ as possible such that minimizing $L_d$ becomes harder for the discriminator. This can be seen as minimizing the worst-case error, which makes the robustness of the discriminator significantly improved.

Maximizing $L_d$ for $\pi$ under current $d$ is equivalent to finding the maxima of functional $J(\pi)$ ($L_d$ with $\theta_d$ and $d$ fixed). According to the calculus of variations [Gelfand et al., 2000], the extrema (maxima or minima) of functional $J(\pi)$ can be obtained by finding a function $\pi$ where the functional derivative is equal to zero. We can show with following proposition that ensuring the functional derivative of $J(\pi)$ equal to zero requires $\partial F/\partial \theta_\pi = 0$.

**Proposition 1.** With the parameters $\theta_d$ of the discriminator $d$ fixed, the functional $J(\pi)$ and $F(s, a, \pi, \pi')$ defined as in Eq. (6) and (11), if uniform continuity of both $F$ and its derivative is satisfied, the necessary condition for $J(\pi)$ attaining its extrema is $\partial F(s, a, \pi, \pi')/\partial \theta_\pi = 0$.

**Proof.** We derive the finite difference approximation of the functional derivative of $J(\pi)$ by Taylor expansion,

$$J(\pi) = J(\pi_{\theta_\pi}) + J'(\pi_{\theta_\pi}) \delta \pi + o(\delta \pi),$$

where $\delta \pi$ is the variation of $\pi$. As $o(\delta \pi)$ is a higher order infinitesimal term when $\theta_\pi \rightarrow 0$, which can be ignored. Plug Eq. (12) into the form of $J(\pi)$, we obtain the finite difference approximation of $J(\pi)$ as follows:

$$\Delta J(\pi) = J(\pi) - J(\pi_{\theta_\pi}) = J'(\pi_{\theta_\pi}) \delta \pi = \frac{d}{d\theta_\pi} F(s, a, \pi, \pi') \, ds \, da \, \delta \pi.$$  \hfill (13)
If the uniform continuity of both $F$ and its derivative is satisfied, we can swap the order of integration and differentiation as well as $ds$ $da$ and $\delta\pi$. The above equation can be simplified as:

$$
\Delta J(\pi) = \int \int \frac{\partial F(s, a, \pi, \pi')}{\partial \theta_e} ds \, da \, \delta\pi = \int \int \frac{\partial F(s, a, \pi, \pi')}{\partial \theta_e} \delta\pi \, ds \, da.
$$

(14)

As $\pi$ can be any function in some function space $\mathcal{F}$. Ensuring the functional derivative of $J(\pi)$ equal to 0 also requires $\partial F(s, a, \pi, \pi')/\partial \theta_e = 0$.

Given the form of functional $F$ in Eq. (11), we can derive the detailed form of $\partial F/\partial \theta_e$ as presented in Eq. (7) of the main article:

$$
\frac{\partial F(s, a, \pi, \pi')}{\partial \theta_e} = \frac{\partial F(s, a, \pi, \pi')}{\partial \log \pi} \cdot \frac{\partial \log \pi}{\partial \theta_e} = \frac{\partial F(s, a, \pi, \pi')}{\partial \log \pi} \cdot \nabla_{\theta_e} \log \pi
$$

$$
= - \mathbb{E}_{(s,a) \sim D_e} \left[ \eta \cdot \nabla_{\theta_e} \log \pi(a | s) \right] + \mathbb{E}_{(s,a) \sim D_o} \left[ \frac{1}{1 - d} \cdot \nabla_{\theta_e} \log \pi(a | s) \right]
$$

$$
- \mathbb{E}_{(s,a) \sim D_e} \left[ \eta \cdot \nabla_{\theta_e} \log \pi(a | s) \right],
$$

where in the last equation, we slightly abuse the notations and write the output value of $d(s, a, \log \pi(a | s))$ as $d$ for simplicity.

As discussed in Section 3.2, above derivative can be equivalently perceived as the gradient of a new loss term for $\pi$, which is exactly the corrective loss term $L_w$ that we are looking for. If we set $\partial L_w / \partial \theta_e = - \partial F / \partial \theta_e$, we recover $L_w$ in following form:

$$
L_w = \mathbb{E}_{(s,a) \sim D_e} \left[ \log \pi(a | s) \cdot \frac{\eta}{d} \right] - \mathbb{E}_{(s,a) \sim D_o} \left[ \log \pi(a | s) \cdot \frac{1}{1 - d} \right] + \mathbb{E}_{(s,a) \sim D_o} \left[ \log \pi(a | s) \cdot \frac{\eta}{1 - d} \right]
$$

The introduction of the minus sign on $\partial F / \partial \theta_e$ is to ensure we find the maxima of $J(\pi)$ rather than its minima. Hence minimizing $L_w$ with respect to $\pi$ (make $\partial L_w / \partial \theta_e = - \partial F / \partial \theta_e = 0$) will drive the functional derivative of $J(\pi)$ to zero, which also lead to the maxima of $J(\pi)$ (update with gradient direction $- \partial F / \partial \theta_e$). Adding the new corrective loss term $L_w$ back to our reformulated problem (10), we obtain the final learning objective of $\pi$ for our BC task (Eq. (8) in the main article):

$$
\min_{\pi} \alpha \mathbb{E}_{(s,a) \sim D_o} \left[ - \log \pi(a | s) \right] - \mathbb{E}_{(s,a) \sim D_e} \left[ - \log \pi(a | s) \cdot \frac{\eta}{d (1 - d)} \right] + \mathbb{E}_{(s,a) \sim D_o} \left[ - \log \pi(a | s) \cdot \frac{1}{1 - d} \right]
$$

Note that the derivation of $L_w$ requires uniform continuity to be satisfied in $F$ and its derivative. The involvement of discriminator output values $1/d(s, a, \log \pi(a | s))$ and $1/(1 - d(s, a, \log \pi(a | s))$ may violate the continuity assumption. We thus clip the discriminator output values to the range of $[0.1, 0.9]$ in our practical algorithm.

## B Training procedure details

### B.1 Algorithm details

In this section, we present the pseudocode of DWBC in Algorithm 1.

Algorithm 1 Discriminator-Weighted Behavior Cloning (DWBC)

Require: Dataset $D_e$ and $D_o$, hyperparameter $\eta, \alpha$

1: Initialize the imitation policy $\pi$ and the discriminator $d$
2: while training do
3: Sample $(s_o, a_o) \sim D_e$ and $(s_n, a_n) \sim D_o$ to form a training batch $B$
4: Compute $\log \pi(a | s)$ values for samples in $B$ using the learned policy $\pi$
5: Compute discriminator output values $d(s, a, \log \pi(a | s))$ using sampled $(s, a)$ and computed $\log \pi(a | s)$
6: Update $d$ by minimizing the learning objective in Eq. (5)
7: Update $\pi$ by minimizing the learning objective in Eq. (8)
8: end while

16
B.2 Implementation Details

In this paper, all experiments are implemented with Tensorflow and executed on NVIDIA V100 GPUs. For all function approximators, we use fully connected neural networks with RELU activations. For policy networks, we use tanh (Gaussian) on outputs. We use Adam for all optimizers. The batch size is 256 and γ is 0.99. We rescale the reward to [0, 1] as \( r' = \frac{(r - r_{\text{min}})}{(r_{\text{max}} - r_{\text{min}})} \), where \( r_{\text{max}} \) and \( r_{\text{min}} \) is the maximum and the minimum reward in the dataset. Note that any affine transformation of the reward function does not change the optimal policy of the MDP. The policy and discriminator network are all 3-layer MLP with 256 hidden units in each layer. The learning rate for the policy is \( 1e^{-5} \) and the learning rate for the discriminator network is \( 1e^{-4} \). We search \( \alpha \) in \{1, 2, 5, 10\} for best model performance. We clip the output of \( d \) to [0.1, 0.9]. We set \( \eta \) to 0.5 across all tasks, which is the same as the ORIL paper [Zolna et al., 2020a].

C Additional results

C.1 Datasets details

In this section, we list all datasets used in our paper and the number of trajectories and transitions in \( D_e \) and \( D_o \), different \( X \) is labeled after the dataset name.

<table>
<thead>
<tr>
<th>Dataset-X</th>
<th>#( D_e )</th>
<th>#( D_o )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Trajectories</td>
<td>Transitions</td>
</tr>
<tr>
<td>Hopper_mixed-2</td>
<td>204</td>
<td>96,222</td>
</tr>
<tr>
<td>Hopper_mixed-5</td>
<td>82</td>
<td>39,590</td>
</tr>
<tr>
<td>Hopper_mixed-10</td>
<td>41</td>
<td>19,176</td>
</tr>
<tr>
<td>Hopper_exp-rand-3</td>
<td>7</td>
<td>6,993</td>
</tr>
<tr>
<td>Hopper_exp-rand-6</td>
<td>4</td>
<td>3,996</td>
</tr>
<tr>
<td>Halfcheetah_mixed-2</td>
<td>20</td>
<td>19,980</td>
</tr>
<tr>
<td>Halfcheetah_mixed-5</td>
<td>8</td>
<td>7,992</td>
</tr>
<tr>
<td>Halfcheetah_mixed-10</td>
<td>4</td>
<td>3,996</td>
</tr>
<tr>
<td>Halfcheetah_exp-rand-3</td>
<td>7</td>
<td>6,993</td>
</tr>
<tr>
<td>Halfcheetah_exp-rand-6</td>
<td>4</td>
<td>3,996</td>
</tr>
<tr>
<td>Walker2d_mixed-2</td>
<td>109</td>
<td>74,857</td>
</tr>
<tr>
<td>Walker2d_mixed-5</td>
<td>44</td>
<td>31,010</td>
</tr>
<tr>
<td>Walker2d_mixed-10</td>
<td>22</td>
<td>15,569</td>
</tr>
<tr>
<td>Walker2d_exp-rand-3</td>
<td>7</td>
<td>6,993</td>
</tr>
<tr>
<td>Walker2d_exp-rand-6</td>
<td>4</td>
<td>3,996</td>
</tr>
<tr>
<td>Ant_mixed-2</td>
<td>49</td>
<td>46,646</td>
</tr>
<tr>
<td>Ant_mixed-5</td>
<td>20</td>
<td>19,209</td>
</tr>
<tr>
<td>Ant_mixed-10</td>
<td>10</td>
<td>9,866</td>
</tr>
<tr>
<td>Ant_exp-rand-3</td>
<td>7</td>
<td>6,458</td>
</tr>
<tr>
<td>Ant_exp-rand-6</td>
<td>4</td>
<td>3,996</td>
</tr>
<tr>
<td>Pen_human-2</td>
<td>3</td>
<td>597</td>
</tr>
<tr>
<td>Pen_human-3</td>
<td>2</td>
<td>398</td>
</tr>
<tr>
<td>Pen_human-5</td>
<td>1</td>
<td>199</td>
</tr>
<tr>
<td>Door_human-2</td>
<td>3</td>
<td>770</td>
</tr>
<tr>
<td>Door_human-3</td>
<td>2</td>
<td>479</td>
</tr>
<tr>
<td>Door_human-5</td>
<td>1</td>
<td>255</td>
</tr>
<tr>
<td>Hammer_human-2</td>
<td>3</td>
<td>1,485</td>
</tr>
<tr>
<td>Hammer_human-3</td>
<td>2</td>
<td>844</td>
</tr>
<tr>
<td>Hammer_human-5</td>
<td>1</td>
<td>483</td>
</tr>
<tr>
<td>Relocate_human-2</td>
<td>3</td>
<td>1,328</td>
</tr>
<tr>
<td>Relocate_human-3</td>
<td>2</td>
<td>862</td>
</tr>
<tr>
<td>Relocate_human-5</td>
<td>1</td>
<td>511</td>
</tr>
</tbody>
</table>

Table 2: Dataset details.
C.2 Learning curves

In this section, we provide the learning curves of listed algorithms in Section 5. As the learning procedure of BC is quite fast and stable, for more clearly presentation, we plot the results of BC-exp and BC-all as a horizon bar with the shaded area as the standard deviation across different seeds. The average return in $D_e$ is plotted as the red dashed line in all plots.

Figure 4: Learning curves of compared algorithms on different datasets.