Offline Preference-Based Apprenticeship Learning

Daniel Shin [1] **Daniel S. Brown** [1]

Abstract

We study how an offline dataset of prior (possibly random) experience can be used to address two challenges that autonomous systems face when they endeavor to learn from, adapt to, and collaborate with humans : (1) identifying the human's intent and (2) safely optimizing the autonomous system's behavior to achieve this inferred intent. First, we use the offline dataset to efficiently infer the human's reward function via pool-based active preference learning. Second, given this learned reward function, we perform offline reinforcement learning to optimize a policy based on the inferred human intent. Crucially, our proposed approach does not require actual physical rollouts or an accurate simulator for either the reward learning or policy optimization steps, enabling both safe and efficient apprenticeship learning. We identify and evaluate our approach on a subset of existing offline RL benchmarks that are well suited for offline reward learning and also evaluate extensions of these benchmarks which allow more open-ended behaviors. Our experiments show that offline preference-based reward learning followed by offline reinforcement learning enables efficient and high-performing policies, while only requiring small numbers of preference queries. Videos available at https://sites.google.com/view/offline-prefs.

1. Introduction

For automated sequential decision making systems to effectively interact with humans in the real world, we want these systems to be able to safely and efficiently adapt to and learn from different users. Apprenticeship learning (Abbeel & Ng, 2004)—also called learning from demonstrations (Argall et al., 2009) or imitation learning (Osa et al., 2018)—seeks to allow robots and other autonomous systems to learn how to perform a task through human feedback. While most apprenticeship learning methods require expert demonstrations (Arora & Doshi, 2021), demonstrations are often difficult and learning from preferences (Wirth et al., 2017) can enable users to adapt and teach learning agents without needing to be able to skillfully perform the task themselves. Learning customized policies via preferences has many applications. For example, an assistive robot in a hospital may need to adapt its policy to a variety of patient preferences, physical abilities, and rehabilitation goals. Similarly, a house cleaning robot may need to adapt based on the house layout, occupants, and cleaning tasks. In these scenarios, there is a need for safe and efficient customization of the robot policy to user preferences. In particular, a standard online reinforcement learning or inverse reinforcement learning approach involving trial and error in the environment is likely unacceptable in these situations due to safety and efficiency concerns.

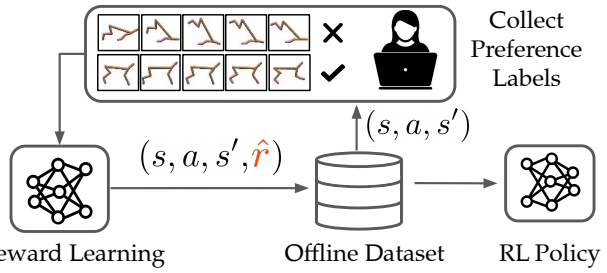


Figure 1. **Offline Preference-Based Apprenticeship Learning.** Given a database consisting of trajectories without reward labels, we query an expert for preference labels over trajectory segments from the database, learn a reward function from preferences, and then perform offline RL using rewards provided by the learned reward function.

Prior work on preference-based apprenticeship learning typically focuses on online methods that require actual rollouts in the environment or access to an accurate simulator or model (Wirth et al., 2017; Christiano et al., 2017; Sadigh et al., 2017; Biyik & Sadigh, 2018; Brown et al., 2019; Lee et al., 2021). While simulations and model-based planning may be possible in some situations, there is the added burden of sim-to-real transfer. Some work has tried to make imitation learning methods safer by enabling the learner

[1]University of California, Berkeley, USA. Correspondence to: Daniel Shin <danielshin@berkeley.edu>, Daniel S. Brown <dsbrown@berkeley.edu>.

Workshop on Human-AI Collaboration in Sequential Decision-Making at the 36^{th} International Conference on Machine Learning (ICML) 2021.

to approximate risky actions (Zhang & Cho, 2016; Hoque et al., 2021) and request human assistance, or by providing performance bounds (Brown & Niekum, 2018; Brown et al., 2020) on the performance of the imitation learner's policy; however, these methods still rely on direct interactions with the environment, which can be costly in terms of data collection time and may lead to unsafe actions.

Ideally, we would like to both identify a human's reward function and also optimize the corresponding optimal policy without requiring expensive and possibly unsafe environment interactions. While there has been some work on offline apprenticeship learning, prior work has focused mainly on simple environments with discrete actions and hand-crafted reward features (Klein et al., 2011; Bica et al., 2021) and require datasets that consist of expert demonstrations (Lee et al., 2019). Other work has considered higher-dimensional continuous tasks, but assumes access to expert demos or requires experts to label trajectories with reward values (Cabi et al., 2019; Zolna et al., 2020). By contrast, we focus on fully offline apprenticeship learning via small numbers of preference queries which are often much easier to provide than fine-grained reward labels or near-optimal demonstrations (Wirth et al., 2017).

In this paper, we propose to leverage offline datasets for safe and efficient apprenticeship learning. To enable long-term interactions and the ability to adapt to a wide variety of users, we propose that robots and other autonomous agents should, when possible, make use of large repositories of prior experience. In particular, we propose Offline Preference-based Apprenticeship Learning (OPAL), a novel, yet intuitive, approach that consists of *offline preference-based reward learning combined with offline reinforcement learning*. This combination has two appealing and desirable properties: safety and efficiency. No interactions with the environment are needed for either reward learning or policy learning, removing the sample complexity and safety concerns that come from trial and error learning in the environment. Using preference over trajectories in the offline dataset also eliminates the need to have humans directly provide demonstrations in the environment and enables highly efficient reward inference. Additionally, having access to an offline dataset enables robots and other autonomous agents to reuse prior data that may come from a variety of sources. Not only does this makes reward and policy learning more efficient, but this diversity can also enable better adaptation to different user preferences. Our approach is summarized in Figure 1.

However, while there are many standard reinforcement learning benchmarks, there are many fewer benchmarks for apprenticeship learning or imitation learning. Indeed, recent work has shown that simply using standard reinforcement learning benchmarks and masking the rewards is not sufficiently challenging since often learning a +1 or -1 reward everywhere is sufficient for imitating RL policies (Freire et al., 2020). While there has been progress in developing imitation learning benchmarks, existing domains assume full access to the environment which is only realistic in simulations due to efficiency and safety concerns. To effectively study offline apprenticeship learning, we also require a set of benchmark domains to evaluate different approaches.

One of the contributions of our paper is to evaluate existing offline RL benchmarks from D4RL (Fu et al., 2020) in the offline apprenticeship learning setting, where there is no access to the true reward function. We find that some standard offline RL benchmarks are ill-suited for reward learning—simply replacing all actual rewards in the offline dataset with zeros or a constant results in performance similar to the performance obtained using the true rewards. However, we also identify environments where reward learning is necessary for good performance. We identify high variability and multi-task data as two settings where reward learning is necessary for good policy learning. In our experiments, we focus on this subset of tasks where reward learning is beneficial.

We evaluate OPAL when using a static set of offline preferences as well as when using active offline preference queries. To generate active queries we investigate both Bayesian dropout and ensembling to approximate the posterior distribution over the true reward function and test both disagreement (Christiano et al., 2017) and information gain (Houlsby et al., 2011) acquisition functions for selecting informative queries. Given the learned rewards we then evaluate the performance of offline reinforcement learning on the learned rewards. Crucially, both our reward learning and policy optimization do not require any policy rollouts in the actual environment used to generate the offline data. We evaluate our methods in 2-D continuous control maze navigation tasks and high-dimensional continuous locomotion and manipulation tasks (Fu et al., 2020). We also propose several new tasks designed for open-ended reward learning in offline settings. Our results show that fully offline apprenticeship learning is possible. Given only small numbers of preference queries to learn reward functions, we are able to optimize policies that perform on par with or close to policies trained with ground truth rewards. Videos are available at `https://sites.google.com/view/offline-prefs`.

2. Problem Definition

We model our problem as a Markov decision process (MDP), defined by the tuple $(S, A, r, P, \rho_0, \gamma)$, where S denotes the state space, A denotes the action space, $r : S \times A \to \mathbb{R}$ denotes the reward, $P(s'|s, a)$ denotes the transition dynamics, $\rho_0(s)$ denotes the initial state distribution, and $\gamma \in (0, 1)$

denotes the discount factor.

In contrast to standard reinforcement learning, we do not assume access to the reward function r. Furthermore, we do not assume access to the MDP during training. Instead, we are provided a static dataset, $\mathcal{D}$, consisting of N state, action, next-state transitions tuples, $\mathcal{D} = \{(s_0, a_0, s'_0), ..., (s_N, a_N, s'_N)\}$. Unlike imitation learning, we do not assume that this dataset comes from a single expert attempting to optimize a specific reward function $R(s, a)$. The dataset $\mathcal{D}$ can contain data collected randomly, data collected from a variety of hand-crafted policies, or even from multiple demonstrators each optimizing different reward functions.

Instead of having access to the reward function, r, we assume access to a small number of pairwise preference over trajectories from the offline dataset $\mathcal{D}$. Given these preferences, the goal is to find a policy $\pi(a|s)$ that maximizes the expected cumulative discounted rewards (also known as the discounted returns), $J(\pi) = \mathbb{E}_{\pi,P,\rho_0}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)\right]$, under the unknown true reward function r in a fully offline fashion—no access to the MDP other than offline samples contained in $\mathcal{D}$

3. Offline Preference-Based Apprenticeship Learning

We are interested in offline preference-based apprenticeship learning. As such we consider learning from both static datasets of preferences obtained before reward learning as well as active preference-based learning approaches which sequentially select new queries to be labeled. As a baseline for our active offline reward learning, we compare to T-REX, an offline preference-based reward learning algorithm proposed by Brown et. al. (Brown et al., 2019) that uses a fixed set of ranked suboptimal demonstrations to extrapolate the human's intent beyond the best demonstration. Given a sequence of m demonstrations ranked from worst to best, $\tau_i, ..., \tau_m$, T-REX performs reward inference via the Bradley-Terry pairwise preference ranking loss (Bradley & Terry, 1952; Brown et al., 2019). Thus, given a preference over trajectories, $\tau_i \prec \tau_j$, T-REX seeks to maximize the probability of the preference label:

$$P\big(\tau_i \prec \tau_j \mid \theta\big) = \frac{\exp \sum_{s \in \tau_j} \hat{r}_\theta(s)}{\exp \sum_{s \in \tau_i} \hat{r}_\theta(s) + \exp \sum_{s \in \tau_j} \hat{r}_\theta(s)}, \quad (1)$$

by approximating the reward at state s using a neural network, $\hat{r}_\theta(s)$, such that $\sum_{s \in \tau_i} \hat{r}_\theta(s) < \sum_{s \in \tau_j} \hat{r}_\theta(s)$ when $\tau_i \prec \tau_j$. Given the learned reward function $\hat{r}_\theta(s)$, T-REX then seeks to optimize a policy $\hat{\pi}$ with better-than-demonstrator performance through *online* reinforcement learning using $\hat{\pi}$. Note that T-REX does not actively query the human for more data and thus cannot select new queries to minimize uncertainty over the true reward function. By constrast, we experiment with two different forms of active queries: ensemble disagreement queries and information gain queries. We briefly describe each of these in the following sections.

3.1. Representing Reward Uncertainnty

We consider two of the most popular methods for obtaining uncertainty estimates when using deep learning: ensembles (Lakshminarayanan et al., 2016) and Bayesian dropout (Gal & Ghahramani, 2016).

Ensemble Queries Taking inspiration from online active preference learning work by Christiano et al. (2017), we test the effectiveness of training an ensemble of reward models to approximate the reward function posterior using the Bradley-Terry preference model described above. Similar to prior work, we found that simply initializing the ensemble member networks with different random seeds was sufficient to produce a diverse set of reward functions (Reddy et al., 2020).

Bayesian Dropout We also also test the effectiveness of using dropout to approximate the reward function posterior. As proposed by Gal & Ghahramani (2016), we train a reward network using pairwise preferences and apply dropout to the last layer of the network during training. Then to predict a return distribution over candidate pairs of trajectories we run each trajectory through the network using dropout to obtain an approximate distribution over the returns.

3.2. Active Learning Query Selection

Given a distribution over likely returns for each trajectory in a candidate preference query, we then calculate an estimate of the value of obtaining a label for each candidate query. We consider two query acquisition functions: disagreement and information gain. In contrast to prior work on active preference learning, we do not require on-policy rollouts in the environment (Christiano et al., 2017; Lee et al., 2021) nor require synthesizing queries using a model (Sadigh et al., 2017). Instead, we generate candidate queries by randomly choosing pairs of sub-trajectories obtained from the offline dataset. We then score each candidate pair of trajectories from our pool of potential queries. We take the trajectory pair with the highest score and ask for a pairwise preference label.

Disagreement When using disagreement to select active queries, we select pairs with the highest ensemble disagreement amongst the different return estimates obtained from the ensemble or Bayesian dropout. The disagreement is cal-

culated as the variance in the binary comparison predictions: if fraction p of the posteior samples predict $\tau_i \succ \tau_j$ while the other $1-p$ ensemble models predict $\tau_i \preceq \tau_j$, then the variance of the query pair (τ_i, τ_j) is $p(1-p)$.

Information Gain Queries As an alternative to disagreement, we also consider using information gain (Cover, 1999) between the reward function parameters θ and the outcome Y of querying the human for a preference. Our goal is to learn the reward function parameters θ. As noted above, we model our uncertainty using an approximate posterior $p(\theta \mid \mathcal{D})$, given by training an ensemble of dropout network on our offline dataset $\mathcal{D}$. Houlsby et al. (2011) show that the information gain of a potential query can be formulated as:

$$I(\theta; Y \mid \mathcal{D}) = H(Y \mid \mathcal{D}) - \mathbb{E}_{\theta \sim p(\theta|\mathcal{D})}[H(Y \mid \theta, \mathcal{D})]. \quad (2)$$

Intuitively, the information gain will be maximized when the first term is high, meaning that the overall model has high entropy, but the second term is low, meaning that each individual sample from the posterior has low entropy. This will happen when the individual hypotheses strongly disagree with each other and there is no clear majority. We approximate both terms in the information gain equation with samples from the approximate posterior obtained via ensembbling or dropout. See Appendix A for further details.

3.3. Policy Optimization

Given a learned reward function obtained via preference-learning, we can then use any existing offline or batch reinforcement learning algorithm to learn a policy without requiring knowledge of the transition dynamics or rollouts in the actual environment (Levine et al., 2020). In our experiments we evaluate several common offline RL algorithms.

4. Experiments and Results

We first evaluate a set of offline RL benchmarks from D4RL (Fu et al., 2020) to determine which domains are most suited for apprenticeship learning. In particular, we seek domains where simply imputing an all zero or constant reward does not result in good performance. After isolating several domains where learning a shaped reward matters, we evaluate OPAL on these selected benchmark domains and investigate the performance of different active query strategies. We finally, propose and evaluate tasks specifically designed for offline apprenticeship learning: a maze navigation task where there is a preference over paths in the maze, a task where there is no explicit goal location, but rather there is a preference for the agent to travel in a counter clockwise direction around the environment, and an adapted CartPole task where we learn balancing, clockwise windmilling, and counterclockwise windmilling behaviors all from the same offline dataset of random transitions.

4.1. Evaluating Offline RL Benchmarks

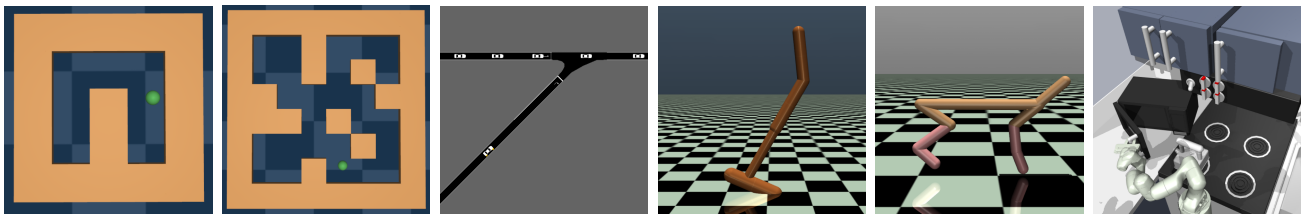
An important assumption made in our problem statement is that we do not have access to the reward function. Before describing our approach for reward learning, we first investigate in what cases we actually need to learn a reward function. We study a set of popular benchmarks used for offline RL (Fu et al., 2020). To test the sensitivity of these methods to the reward function, we evaluate the performance of offline RL algorithms on these benchmarks when we set the reward equal to zero for each transition in the offline dataset and when we set all the rewards equal to a constant (the average reward over the entire dataset).

We evaluated four of the most commonly used offline RL algorithms: Advantage Weighted Regression (AWR) (Peng et al., 2019), Batch-Constrained deep Q-learning (BCQ) (Fujimoto et al., 2019), Bootstrapping Error Accumulation Reduction (BEAR) (Kumar et al., 2019), and Conservative Q-Learning (CQL) (Kumar et al., 2020).

Table 1 shows the resulting performance across a sampling of the tasks in the D4RL benchmarks (Fu et al., 2020). We find that for tasks with only expert data, there is no need to learn a reward function since using an all zero reward function can often lead to similar performance. We attribute this to the fact that many offline RL algorithms are similar to behavioral cloning (Levine et al., 2020). Thus, when an offline RL benchmarks consists of only expert data, offline RL algorithms do not need reward information to perform well. We discuss this in more detail in Appendix B.

However, when the offline dataset contains a wide variety of data that is either random or multi-task (Maze2D), then we find that running offline RL algorithms on the dataset with an all zero reward function significantly hurts performance. Sometimes even resulting in performance worse than that of a uniformly random policy.

To study offline apprenticeship learning we focus our attention on the D4RL benchmark domains shown in Figure 2. We selected both of the Maze environments, the Flow Merge traffic simulation, the Hopper and Halfcheetah random environments, and the Franka Kitchen-Complete domains. Maze2D represents an environment with low dimensional observation space and the task is goal oriented, whereas MuJoCo represents an environment with high dimensional observation space and task is not goal oriented. We also consider the Franka Kitchen-Complete task from D4RL and show that our approach is able to learn policies close to expert performance.



(a) U-Maze (b) Medium Maze (c) Flow Merge (d) Hopper (e) Halfcheetah (f) Franka Kitchen

Figure 2. Experimental domains chosen from D4RL (Fu et al., 2020) for use in offline preference-based apprenticeship learning.

Table 1. Offline RL performance on D4RL benchmark tasks comparing the performance with the true reward to performance using a constant reward equal to the average reward over the dataset (Avg) and zero rewards everywhere (Zero). Even when all the rewards are set to the average or to zero, many offline RL algorithms still perform surprisingly well. Results are averages over three random seeds. Results are normalized such that the best ground truth performance across all offline reinforcement learning algorithms and all settings (true reward, Avg, and Zero) is 100.0 and the random policy performance is 0.0. Results are averaged over three random seeds. Bolded environments are ones where we find significant degradation in performance and are used for later experiments.

TASK	AWR		BCQ		BEAR		CQL	
	AVG	ZERO	AVG	ZERO	AVG	ZERO	AVG	ZERO
FLOW-RING-RANDOM-V1	68.3	67.5	68.9	54.7	102.2	98.5	64.7	79.2
FLOW-MERGE-RANDOM-V1	-14.7	-14.5	6.1	2	-25.3	-22.4	-8.7	-30.6
MAZE2D-UMAZE	12.8	10.0	-23.7	-14.0	-8.6	18.6	-6.4	-20.6
MAZE2D-MEDIUM	-15.9	-11.5	-23.9	-18.2	-16.7	-24.8	-11.7	-28.1
HALFCHEETAH-RANDOM	6.9	6.9	8.3	8.3	8.3	8.3	0.1	3.7
HALFCHEETAH-MEDIUM-REPLAY	71.7	69.9	71.4	77.6	58.7	61.5	24.3	-3.8
HALFCHEETAH-MEDIUM	73.2	71.7	89.1	89.6	89.9	87.0	92.9	91.3
HALFCHEETAH-MEDIUM-EXPERT	6.4	6.1	102.2	101.1	47.5	44.6	41.6	51.7
HALFCHEETAH-EXPERT	6.9	6.7	99.5	94.0	47.2	85.1	2.9	33.0
HOPPER-RANDOM	11.7	1.1	17.6	19.0	20.7	7.1	51.9	-0.8
HOPPER-MEDIUM-REPLAY	45.0	38.0	43.2	36.1	26.0	52.4	79.2	22.1
HOPPER-MEDIUM	69.4	55.9	102.3	93.6	36.1	112.9	129.3	123.0
HOPPER-MEDIUM-EXPERT	26.5	25.3	132.2	60.8	23.3	23.8	49.7	53.0
HOPPER-EXPERT	28.6	19.6	139.0	134.2	72.5	95.4	42.3	99.1
KITCHEN-COMPLETE	10.8	8.4	88.0	12.0	0.0	37.7	80.0	66.7
KITCHEN-MIXED	20.0	32.9	85.7	28.6	117.7	128.9	63.5	111.1
KITCHEN-PARTIAL	138.2	94.1	78.4	39.2	367.6	379.4	196.1	130.7

4.2. Apprenticeship Learning on a Subset of D4RL

4.2.1. Oracle Preference Labels

To facilitate a better comparison across different methods for active learning, we use oracle preference labels, where one trajectory sequence is preferred over another if it has higher ground-truth return. We ran a preliminary pilot study and which gives evidence that our method has the potential to work well with noisy human preference labels, but this is left as an area for future work.

4.2.2. Maze Navigation

For our first set of experiments, we are using the dense version of the Maze2d domain (Fu et al., 2020), which involves moving a force-actuated ball (along the X and Y axis) to a fixed target location. The observation is 4 dimensional, which consists of the (x, y) location and velocities. The offline dataset consists of one continuous trajectory of the agent navigation to random intermediate goal locations. The ground truth reward is defined to be the negative exponentiated distance. The Maze2D-Umaze environment contains an associating offline dataset with one million data points and the Maze2D-Medium environment contains an associating offline dataset with two million data points.

For our experimental setup, we first randomly select 5 pairs of trajectory and train 5 epochs with our models. After this initial training process, for each round, one additional pair of trajectories is queried to be added to the training set and we train one more epoch on this augmented dataset. The trained reward model is then used to predict all reward labels in the offline dataset, which will be used to train an offline

RL algorithm (e.g. AWR (Peng et al., 2019)).

Results are provided in Table 2. We find ensemble disagreement to be the best performing approach for both Maze2D-Umaze and Maze2D-Medium. The offline RL performance for ensemble disagreement are shown in Figure 3. However, none of the approaches were able to recover a policy on par with the policy trained with ground truth rewards. This is most likely due the fact this domain is goal oriented and learning a reward with trajectory comparisons might be difficult since the destination matter more than the path. Ranking accuracy of each approach is provided in the Appendix. Notably, random query barely improves after round 5 in terms of ranking accuracy whereas all of our approaches continue to improve after round 5.

4.2.3. MuJoCo Tasks

We additionally ran our experiments on two MuJoCo environments, Hopper-v2 and Halfcheetah-v2. For the tasks, the agent is rewarded for making the Hopper or Halfcheetah move forward as fast as possible. The MuJoCo tasks presents an additional challenge in comparison to the Maze2d domain since they have higher dimensional observations spaces, 11 for Hopper-v2 and 17 for Halfcheetah-v2. Both the Hopper-v2 and Halfcheetah-v2 environment has an associating offline dataset with one million data points.

Our experimental setup is similar to one for Maze2d except we start with 50 pairs of trajectory instead of 5 and we add 10 trajectories per active query instead of one. We decided to have 50 pairs of initial trajectory and add 10 additional pairs per round of querying due to the fact that MuJoCo tasks have higher dimensional observational spaces, which requires more data points to learn accurately.

Results are shown in Figure 4. We found dropout information gain to be the best performing approach for the Hopper task and ensemble disagreement to be the best performing approach for the Halfcheetah task. Notably, the policy learned after 10 rounds of querying is on par with the policy learned with the the ground truth reward. This is most likely because trajectories in MuJoCo tasks matter more than the destination. So learning a reward with trajectory comparisons is natural and better suited for these tasks.

4.2.4. Flow Tasks

The Flow environment involves directing traffic such that the flow of traffic is maximized. The particular road configuration we used in our experiment is the merge configuration. We found the original TREX to be the best performing query method but all query methods were able to recover near ground truth performance.

4.2.5. Robotic Manipulation

The FrankaKitchen domain involves interacting with various objects in the kitchen to reach a certain state configuration. The objects you can interact with includes a water kettle, a light switch, a microwave, and cabinets. Zero masking the reward causes the performance to degrade significantly. The performance recovered with Ensemble Disagreement not only recovers performance similar to the Ground Truth performance but also converges to a better performance faster.

4.3. New Offline Preference-Based Apprenticeship Learning Tasks

To explore the full benefits of learning a reward function from preferences, we created two additional tasks to highlight the need for a shaped reward function, rather than simply a goal indicator. To create these environments we adapted common gym environments including several that are in the D4RL set of benchmarks.

4.3.1. Maze Navigation with Constraint Region

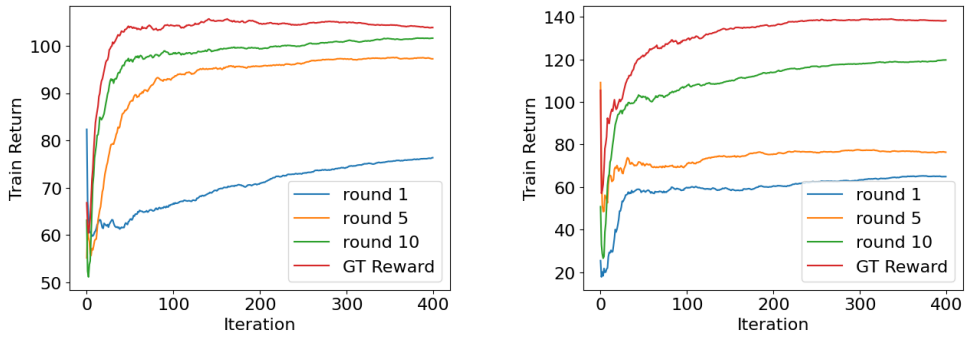
We created a new variant of the Maze2d-Medium task but where there is a constraint region in the middle of the maze that the supervisor does not want the robot to enter. Figure 7 shows this scenario where the highlighted yellow region is traversable by the agent, but this behavior is undesirable. After only 25 active queries using ensemble disagreement we obtain the behavior shown in Figure 7 where the offline RL policy has learned to reach the goal while avoiding the constraint region.

4.3.2. Open Maze Orbiting

Next, we took the D4RL Open Maze environment and dataset and used it to teach an agent to patrol the domain in counter clockwise orbits. The dataset only contains the agent moving to randomly chosen goal locations, thus this domain highlights the benefits of stiching together data from an offline database as well as the benefits of learning a shaped reward function rather than simply using a goal classifer to use as the reward function which has been used in prior work on offline RL work (Eysenbach et al., 2021).

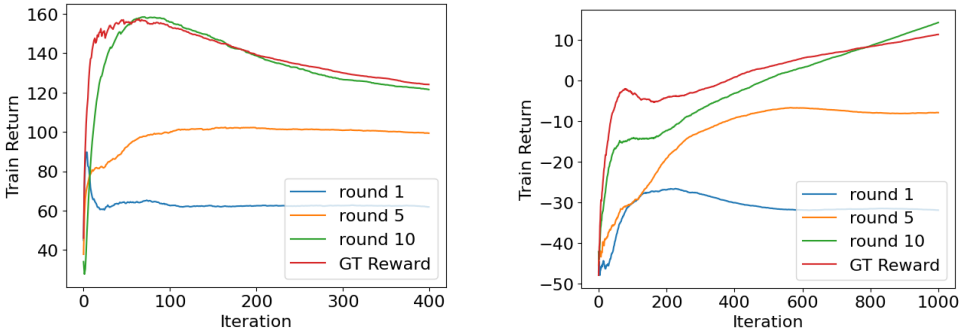
4.3.3. Open Ended CartPole Behaviors

We created an offline dataset of 1,000 random trajectories of length 200 in a modified CartPole domain where the trajectories only terminate at the end of 200 timesteps—this allows the pole to swing below the track and also allows the cart to move off the visible track to the left or right. Given this dataset of behavior agnositic data, we wanted to see whether we could optimize different policies, corresponding to different human preferences. We optimized the



(a) Maze2d-Umaze (b) Maze2d-Medium

Figure 3. (left) Ensemble disagreement after 10 rounds has similar performance to ground truth reward in Maze2d-Umaze. (right) Ensemble disagreement after 10 rounds has slightly worse performance compared to ground truth reward in Maze2d-Medium.



(a) Hopper-Random-v2 (b) Half-Cheetah-v2

Figure 4. (left) Dropout disagreement after 10 rounds has similar performance to ground truth reward in Hopper. (right) Ensemble disagreement after 10 rounds has similar performance compared to ground truth reward in Halfcheetah.

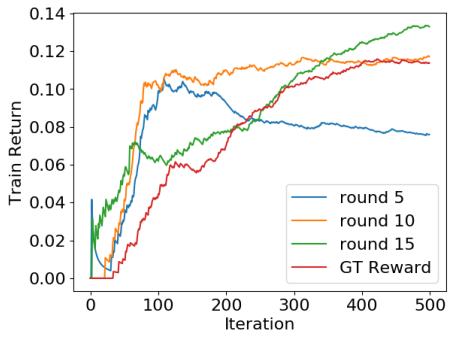


Figure 5. Kitchen-Complete-v1

Figure 6. Dropout disagreement after 15 rounds has similar performance to ground truth reward in Kitchen-complete

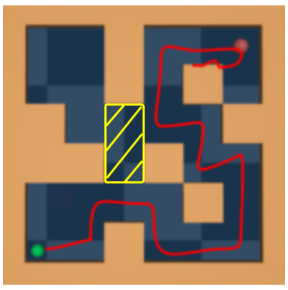


Figure 7. **Constrained Goal Navigation**. The highlighted yellow region represents a constraint region that the supervisor prefers the agent to avoid while also traveling to the goal position shown in red. OPAL produces trajectories that match this preference by taking a more round-about, but more preferred, path to the goal.

following policies: *balance*, where the supervisor prefers the pole to be balanced upright and prefers the cart to stay in the middle of the track; *clockwise windmill*, where the supervisor prefers the cart to stay in the middle of the track but prefers the pole to swing around as fast as possible in the clockwise direction; and *counterclockwise windmill*, which is indentical to clockwise windmill except the preference is for the pole to swing in the counterclockwise direction. OPAL is able to learn policies for all three behaviors. Videos of these behaviors can be found at https://sites.google.com/view/offline-prefs.

Table 2. **OPAL Performance Using Reward Predicted After N Rounds of Querying**: Offline RL performance using rewards predicted with T-REX using a static number of randomly selected pairwise preferences (T-REX), Ensemble Disagreement(EnsemDis), Ensemble Information Gain (EnsemInfo), Dropout Disagreement (DropDis), Dropout Information Gain (DropInfo) and Ground Truth Reward (GT). N is set to 5 for Maze2D-Umaze, 10 for Maze2d-Medium and flow-merge-random, 15 for Hopper and Halfcheetah. N is selected based on the size of the dimension of observation space and the complexity of the environment. Results are averages over three random seeds. Results are normalized such that the ground truth performance is 100.0 and the random policy performance is 0.0. We compare the performance of OPAL using both AWR (Peng et al., 2019) and CQL (Kumar et al., 2020) for policy optimization.

AWR (Peng et al., 2019)	Query Aquisition Method				
	T-REX	EnsemDis	EnsemInfo	DropDis	DropInfo
maze2d-umaze	78.2	**93.5**	89.2	88.0	61.3
maze2d-medium	71.6	**86.4**	71.0	52.6	44.4
hopper	69.0	77.5	72.8	87.6	**90.6**
halfcheetah	96.1	**113.7**	100.6	84.4	91.7
flow-merge-random	**110.1**	89.0	92.1	86.2	84.2
kitchen-complete	79.6	105.0	**158.8**	48.5	65.4

CQL (Kumar et al., 2020)	Query Aquisition Method				
	T-REX	EnsemDis	EnsemInfo	DropDis	DropInfo
maze2d-umaze	87.0	54.8	100.4	**102.3**	95.2
maze2d-medium	41.1	33.1	**115.4**	1.6	34.5
hopper	32.1	28.7	17.5	35.9	**42.0**
halfcheetah	68.4	75.9	76.3	75.0	**79.6**
flow-merge-random	-13.6	44.3	103.2	69.6	**114.0**
kitchen-complete	85.7	100.0	71.4	**114.3**	71.4

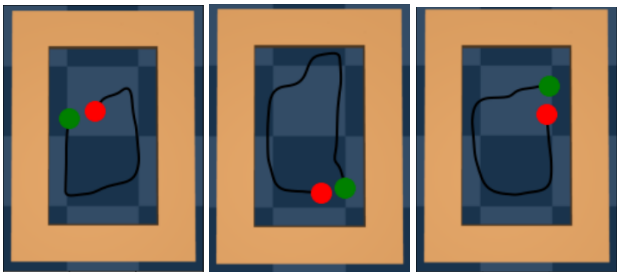


Figure 8. **Learned Orbiting Policy**. The preference is to move in a counter clockwise direction. Green circles represent random initial states and red circles represent the position of the agent at the end of a rollout.

5. Discussion and Future Work

Our goal is to learn a policy from user preferences without requiring access to a model, simulator, or interactions with the environment. Toward this goal we propose OPAL: Offline Preference-based Apprenticeship Learning. We evaluate several different offline RL benchmarks and find that those that contain mainly random data or multi-task data are best suited for reward inference. While most prior work on apprenticeship learning has focused on learning from expert demonstrations, preference-based reward learning enables us to learn a user's intent even from random datasets by selecting informative subsequences of transitions and requesting pairwise preferences over these subsequences. Our results show that using ensemble disagreement as an acquisition function works well on most tasks and with only 10-15 preference queries are able to learn policies that perform similar to policies trained with full access to tens of thousands of ground-truth reward samples.

We are excited about the potential of offline reward and policy learning and believe that our work provides a stepping stone towards safer and more efficient autonomous systems that can better learn from and interact with humans. Preference-learning enables users to teach a system without needing to be able to directly control the system or demonstrate optimal behavior. We believe this makes preferences ideally suited for many tasks in as healthcare, finance, or robotics. Future work includes a user study evaluating OPAL with real human preferences, developing more complex and realistic datasets and benchmarks, developing offline RL algorithms that are well-suited for reward inference, and developing active learning queries that are specifically tailored for offline reinforcement learning.

References

Abbeel, P. and Ng, A. Y. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1, 2004.

Argall, B. D., Chernova, S., Veloso, M., and Browning, B. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

Arora, S. and Doshi, P. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, pp. 103500, 2021.

Bica, I., Jarrett, D., Hüyük, A., and van der Schaar, M. Learning" what-if" explanations for sequential decision-making. In *International Conference on Learning Representations*, 2021.

Biyik, E. and Sadigh, D. Batch active preference-based learning of reward functions. In *Conference on robot learning*, pp. 519–528. PMLR, 2018.

Bradley, R. A. and Terry, M. E. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

Brown, D. and Niekum, S. Efficient probabilistic performance bounds for inverse reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

Brown, D., Goo, W., Nagarajan, P., and Niekum, S. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International Conference on Machine Learning*, pp. 783–792. PMLR, 2019.

Brown, D., Coleman, R., Srinivasan, R., and Niekum, S. Safe imitation learning via fast bayesian reward inference from preferences. In *International Conference on Machine Learning*, pp. 1165–1177. PMLR, 2020.

Cabi, S., Colmenarejo, S. G., Novikov, A., Konyushkova, K., Reed, S., Jeong, R., Zolna, K., Aytar, Y., Budden, D., Vecerik, M., et al. Scaling data-driven robotics with reward sketching and batch reinforcement learning. *arXiv preprint arXiv:1909.12200*, 2019.

Christiano, P. F., Leike, J., Brown, T. B., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. In *NIPS*, 2017.

Cover, T. M. *Elements of information theory*. John Wiley & Sons, 1999.

Eysenbach, B., Levine, S., and Salakhutdinov, R. Replacing rewards with examples: Example-based policy search via recursive classification. *arXiv preprint arXiv:2103.12656*, 2021.

Freire, P., Gleave, A., Toyer, S., and Russell, S. Derail: Diagnostic environments for reward and imitation learning. In *Proceedings of the Workshop on Deep Reinforcement Learning at NeurIPS*, 2020.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020.

Fujimoto, S., Meger, D., and Precup, D. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pp. 2052–2062. PMLR, 2019.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pp. 1050–1059. PMLR, 2016.

Hoque, R., Balakrishna, A., Putterman, C., Luo, M., Brown, D. S., Seita, D., Thananjeyan, B., Novoseller, E., and Goldberg, K. Lazydagger: Reducing context switching in interactive imitation learning. *arXiv preprint arXiv:2104.00053*, 2021.

Houlsby, N., Huszár, F., Ghahramani, Z., and Lengyel, M. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

Klein, E., Geist, M., and Pietquin, O. Batch, off-policy and model-free apprenticeship learning. In *European Workshop on Reinforcement Learning*, pp. 285–296. Springer, 2011.

Kumar, A., Fu, J., Tucker, G., and Levine, S. Stabilizing off-policy q-learning via bootstrapping error reduction. *arXiv preprint arXiv:1906.00949*, 2019.

Kumar, A., Zhou, A., Tucker, G., and Levine, S. Conservative q-learning for offline reinforcement learning. *arXiv preprint arXiv:2006.04779*, 2020.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. *arXiv preprint arXiv:1612.01474*, 2016.

Lee, D., Srinivasan, S., and Doshi-Velez, F. Truly batch apprenticeship learning with deep successor features. *arXiv preprint arXiv:1903.10077*, 2019.

Lee, K., Smith, L., and Abbeel, P. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. *arXiv preprint arXiv:2106.05091*, 2021.

Levine, S., Kumar, A., Tucker, G., and Fu, J. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

Osa, T., Pajarinen, J., Neumann, G., Bagnell, J. A., Abbeel, P., and Peters, J. An algorithmic perspective on imitation learning. *arXiv preprint arXiv:1811.06711*, 2018.

Peng, X. B., Kumar, A., Zhang, G., and Levine, S. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Reddy, S., Dragan, A., Levine, S., Legg, S., and Leike, J. Learning human objectives by evaluating hypothetical behavior. In *International Conference on Machine Learning*, pp. 8020–8029. PMLR, 2020.

Sadigh, D., Dragan, A. D., Sastry, S., and Seshia, S. A. Active preference-based learning of reward functions. In *Robotics: Science and Systems*, 2017.

Wirth, C., Akrour, R., Neumann, G., Fürnkranz, J., et al. A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research*, 18(136): 1–46, 2017.

Zhang, J. and Cho, K. Query-efficient imitation learning for end-to-end autonomous driving. *arXiv preprint arXiv:1605.06450*, 2016.

Zolna, K., Novikov, A., Konyushkova, K., Gulcehre, C., Wang, Z., Aytar, Y., Denil, M., de Freitas, N., and Reed, S. Offline learning from demonstrations and unlabeled experience. *arXiv preprint arXiv:2011.13885*, 2020.

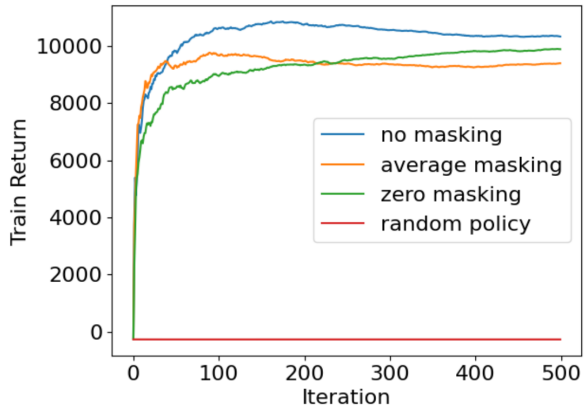
A. Information Gain

Given a pair of trajectories (τ_A, τ_B), let $Y = 0$ denote that the human prefers trajectory A and $Y = 1$ denote that the human prefers trajectory B. Use the Bradley-Terry preference model (Bradley & Terry, 1952) we have that

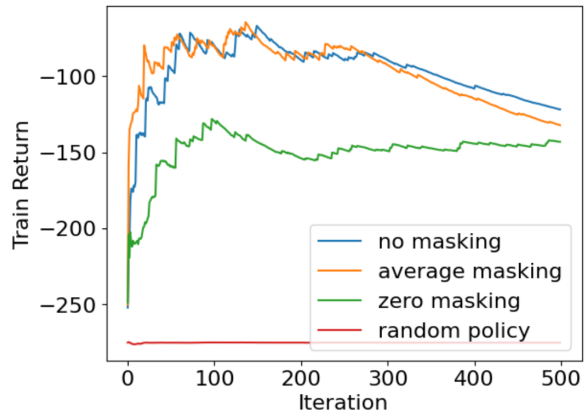
$$P(Y = 0 \mid \theta, \mathcal{D}) = \frac{\exp(\beta R(\xi_A))}{\exp(\beta R(\xi_A)) + \exp(\beta R(\xi_B))} \quad (3)$$

where $R(\xi) = \sum_{(s,a)\in\tau_i} \hat{r}_\theta(s)$ and $P(Y = 1 \mid \theta, \mathcal{D}) = 1 - P(Y = 0 \mid \theta, \mathcal{D})$. To perform queries using Information Gain, we evaluate a pool of potential trajectory pairs, evaluate the information gain for each pair and query the human for the pair that has the highest information gain.

B. Evaluating Offline RL Benchmarks



(a) Hammer-Expert-v1



(b) Hammer-Human-v1

Figure 9. When the offline dataset consists of expert transitions, naive reward functions like all zeros or the average of all rewards, is able to recover policy comparable to the policy learned with ground truth reward.

Figure 9 shows the results for several different ablations on two offline RL benchmarks. *No Masking* is standard Offline RL with the true dense reward function. *Average Masking* simply replaces all the true rewards with the average reward over the MDP. *Zero Masking* replaces all the rewards with zeros. We also compare to the performance of a random policy. We find that using average and zero masking are very competitive with using the true rewards and that they perform significantly better than a purely random policy.

This is a byproduct of offline reinforcement learning algorithms needing to learn in the presence of out-of-distribution actions. There are broadly two classes of methods towards solving the out-of-distribution problem. Behavioral cloning based methods like Advantage Weighted Regression used in our experiments, train only on actions observed in the dataset, which avoid OOD actions completely. Dynamic programming (DP) methods, like BCQ, constrains the trained policy distribution to lie close to the behavior policy that generated the dataset. As a result of offline reinforcement learning algorithms constraining action to be similar to that of the static dataset, if the static dataset consists of exclusively expert actions, then the offline reinforcement learning algorithm will recover a policy that has expert like action regardless of the reward.

B.1. Advantage-Weighted Regression

Consider Advantage-Weighted Regression with a constant reward function $r(s, a) = c$. The algorithm is simple. We have a replay buffer from which we sample transitions and estimate the value function via regression

$$V* \leftarrow \arg\min_V \mathbb{E}_{s,a\sim D}\left[\|\mathcal{R}^{\mathcal{D}}_{s,a} - V(s)\|\right] \quad (4)$$

then the policy is updated via supervised learning:

$$\pi \leftarrow \arg\max_\pi \mathbb{E}_{s,a\sim D}\left[\log \pi(a|s) \exp\left(\frac{1}{\beta}(\mathcal{R}^{\mathcal{D}}_{s,a} - V(s))\right)\right] \quad (5)$$

If the rewards are all zero, then we have $V(s) = 0, \forall s \in \mathcal{S}$ and

$$\pi \leftarrow \arg\max_\pi \mathbb{E}_{s,a\sim D}\left[\log \pi(a|s) \exp\left(\frac{1}{\beta}0\right)\right] \quad (6)$$

$$= \max_\pi \mathbb{E}_{s,a\sim D}\left[\log \pi(a|s)\right]. \quad (7)$$

This is exactly the behavioral cloning loss which will learn to take the actions in the replay buffer. Thus, AWR is identical to BC when the rewards are all zero.

For non-zero, constant rewards, the advantage term will be zero (at least for deterministic tasks). If all trajectories from the state s have the same length, then we will have zero advantage. However, if there are trajectories that terminate earlier than others, then AWR will put weights on these trajectories proportional to their length (for positive

rewards $c > 0$) and inversely proportional to their length (for negative rewards $c < 0$). Thus, a terminal state will leak information and a good policy can be learned without an informative reward function.

Table 3. Ranking Accuracy After 5 Rounds and 10 Rounds

	RANDOM	ENSEMDIS	ENSEMINFO	DROPDIS	DROPINFO
MAZE2D-UMAZE	0.818/0.867	0.817/0.916	0.77/0.862	0.874/0.893	0.715/0.755
MAZE2D-MEDIUM	0.646/0.686	0.6495/0.7825	0.6375/0.824	0.636/0.795	0.692/0.756
HOPPER	0.929/0.922	0.943/0.966	0.952/0.966	0.946/0.966	0.927/0.96
HALFCHEETAH	0.814/0.873	0.852/0.942	0.866/0.944	0.875/ 0.927	0.833/0.913
FLOW-MERGE-RANDOM	0.808/0.841	0.829/0.881	0.825/0.861	0.846/0.884	0.839/0.893
KITCHEN	0.964/0.978	0.981/0.986	0.967/0.984	0.973/0.990	0.953/0.986

Table 4. Offline RL performance on D4RL benchmark tasks **with no reward information**, i.e., when all rewards in the dataset are set to zero. These represent the raw performance numbers that are unnormalized.

TASK	AWR	BCQ	BEAR	CQL	RANDOM
MAZE2D-UMAZE	55.0	41.8	59.7	38.2	62.0
MAZE2D-MEDIUM	34.5	28.5	22.5	19.6	44.8
HOPPER-RANDOM-V2	29.1	199.4	86.3	11.0	18.4
HALFCHEETAH-RANDOM-V2	-48.3	-1.8	-0.5	158.5	-285.8
FLOW-MERGE-RANDOM-V1	85.6	121.4	68.3	50.5	117
FLOW-RING-RANDOM-V1	-44.3	-67.5	11.6	-23.3	-166.2
HOPPER-MEDIUM-REPLAY-V2	954.6	908.3	1311.2	563.7	18.3
HOPPER-EXPERT-V2	564.2	2666.5	3511.2	2945.1	18.4
HAMMER-EXPERT-V1	9715.1	7433.1	13523.5	1437.8	-274.9

Table 5. Offline RL performance on Adroit tasks using true reward (TR), average reward over the dataset (AR), and zero rewards everywhere (ZR). Even when all the rewards are set to a constant (AR) or to zero (ZR), many offline RL algorithms still perform surprisingly well.

TASK	AWR		BCQ		BEAR		CQL	
	AVG	ZERO	AVG	ZERO	AVG	ZERO	AVG	ZERO
HAMMER-EXPERT	65.9	67.2	65.0	51.8	0.2	92.8	5.8	11.5
PEN-EXPERT	71.3	70.2	97.1	81.3	53.4	91.3	49.5	84.8
DOOR-EXPERT	27.8	33.3	103.0	93.1	-0.3	107.7	74.4	61.9
RELOCATE-EXPERT	5.9	4.8	85.2	41.2	-1.0	286.9	19.8	44.3