D4RL: Datasets for Deep Data-Driven Reinforcement Learning

Justin Fu  
UC Berkeley  
justinjfu@eecs.berkeley.edu

Aviral Kumar  
UC Berkeley  
aviralk@berkeley.edu

Ofir Nachum  
Google Brain  
ofirnachum@google.com

George Tucker  
Google Brain  
gjt@google.com

Sergey Levine  
UC Berkeley, Google Brain  
svlevine@eecs.berkeley.edu

Abstract

The offline reinforcement learning (RL) problem, also referred to as batch RL, refers to the setting where a policy must be learned from a dataset of previously collected data, without additional online data collection. In supervised learning, large datasets and complex deep neural networks have fueled impressive progress, but in contrast, conventional RL algorithms must collect large amounts of on-policy data and have had little success leveraging previously collected datasets. As a result, existing RL benchmarks are not well-suited for the offline setting, making progress in this area difficult to measure. To design a benchmark tailored to offline RL, we start by outlining key properties of datasets relevant to applications of offline RL. Based on these properties, we design a set of benchmark tasks and datasets that evaluate offline RL algorithms under these conditions. Examples of such properties include: datasets generated via hand-designed controllers and human demonstrators, multi-objective datasets, where an agent can perform different tasks in the same environment, and datasets consisting of a heterogeneous mix of high-quality and low-quality trajectories. By designing the benchmark tasks and datasets to reflect properties of real-world offline RL problems, our benchmark will focus research effort on methods that drive substantial improvements not just on simulated benchmarks, but ultimately on the kinds of real-world problems where offline RL will have the largest impact.

1 Introduction

The last decade has seen impressive progress across a range of machine learning application domains, driven in large part by high-capacity deep neural network models in hand with large and diverse training datasets [Goodfellow et al., 2016]. While reinforcement learning (RL) algorithms have also benefitted from expressive deep neural network representations [Mnih et al., 2015], RL is inherently an active learning framework, which limits the extent to which RL can leverage large, previously collected datasets. The offline RL setting [Lange et al., 2012], also referred to as full batch RL, where agents must learn from large, fixed datasets of logged interaction, provides a bridge between the flexible control formalism provided by RL and the data-driven success of supervised learning. The promise of offline RL in this context is potentially enormous: in the same way that deep neural networks can enable powerful pattern recognition when provided with large datasets, offline RL methods equipped with high-capacity models can enable training of powerful decision making engines entirely from existing previously collected datasets. This could have profound implications for a range of application domains, from robotics and autonomous driving to operations research, healthcare, and other decision-making support domains.

Website with tasks and datasets is available at [https://sites.google.com/view/d4rl/home](https://sites.google.com/view/d4rl/home)

Figure 1: A small selection of tasks contained within the benchmark.
The offline RL setting also addresses several major limitations of the standard online RL formulation. First, by maximally leveraging offline data, RL algorithms can likely get away with substantially lower sample complexity: while online RL methods might require millions or even billions of time steps of experience to learn each task, many settings, such as autonomous driving, natural language interfaces, and recommender systems already offer abundant sources of previously collected logged data. Although this data may not correspond to any specific task that a practitioner may be interested in solving, utilizing it in an offline RL framework may enable these tasks to be solved with minimal or no additional data collection. Second, the offline RL setting alleviates many of the safety concerns associated with online RL. In many domains, from robotics to medical diagnosis, the cost of failure is unacceptable. Even if an online RL algorithm might converge on an effective policy that fails rarely, failures during the training process may make the algorithm impractical to use. The offline RL setting allows policies to be pre-trained on large datasets, such that they may achieve an acceptable baseline level of performance the first time they are deployed.

Finally, several authors [Lange et al., 2012, Agarwal et al., 2019b] argue for the offline RL setting simply as a platform for research that cleanly decouples the dynamic programming and optimization aspects from the exploration aspect. Because the optimization process no longer affects the data collected, offline RL provides a clean framework for testing improvements in RL optimization algorithms absent of other confounding factors.

Unfortunately, current offline RL methods have not yet lived up to the full promise that offline RL has in enabling reinforcement learning from large datasets. While recent work has investigated a number of possible technical reasons for this [Wu et al., 2019], one of the major challenges in addressing these issues has been the lack of realistic evaluation protocols. Most recent works in this area have used data collected from full or partial training runs of standard online RL methods. However, such datasets are not necessarily representative of the kinds of scenarios in which offline RL might be used in practice. In fact, it is not even clear whether this setting permits much improvement over the best (or even the average) policy that collected the training data. In this work, our aim is to design a set of benchmark tasks and datasets that better reflect the kinds of scenarios where offline RL might be utilized in practice, as well as tasks where we can expect large potential improvement from offline RL over the behavior policy.

The key contribution of this work is the introduction of D4RL: a suite of tasks and datasets for benchmarking progress in offline RL. We focus our design around two primary principles: the tasks should be conducive to experimentation but also realistic, and the set of tasks and datasets should exercise dimensions of the offline RL problem that cover the sorts of challenging scenarios where current offline RL algorithms may struggle. We focus on dataset compositions that might reflect realistic settings: data from human demonstrations, passively collected logs of multiple different tasks distinct from the task being learned, and data from non-learned “scripted” controllers. To exercise algorithmic challenges, we provide tasks with different types of data distributions, such as data from behavior policies that cannot be represented precisely by Markovian policies (e.g., demonstrations or stateful hand-designed controllers), and tasks with strict safety considerations. Finally, we evaluate several state-of-the-art algorithms [Haarnoja et al., 2018, Kumar et al., 2019, Wu et al., 2019] to measure the gap in existing algorithms, demonstrating that while the algorithms perform well in the specific tasks they were designed for, they perform poorly on tasks such as data collected from hand-designed controllers and multi-task behavior. We hope that our evaluation provides insight into existing shortcomings in offline RL methods, and that our benchmark provides a meaningful metric for progress in this emerging area of RL.

2 Background

The offline reinforcement learning problem statement is formalized within a Markov decision process (MDP), defined by a tuple \((S, A, P, R, \rho_0, \gamma)\), where \(S\) denotes the state space, \(A\) denotes the action space, \(P(s'|s, a)\) denotes the transition distribution, \(\rho_0(s)\) denotes the initial state distribution, \(R(s, a)\) denotes the reward function, and \(\gamma \in (0, 1)\) denotes the discount factor. The goal in RL is to find a policy \(\pi(a|s)\) that maximizes the expected cumulative discounted rewards \(J(\pi) = E_{\pi, P, \rho_0} \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right]\), also known as the discounted returns.

In episodic RL, the algorithm is given access to the MDP via trajectory samples for arbitrary \(\pi\) of the algorithm’s choosing. So-called “off-policy” algorithms in this paradigm may use experience replay [Lin, 1984].
store these trajectories in a replay buffer $\mathcal{D}$ of transitions $(s_t, a_t, s_{t+1}, r_t)$ and use an off-policy algorithm such as Q-learning [Watkins and Dayan, 1992] in order to optimize $\pi$. However, these methods still iteratively collected additional data to append to the buffer using the latest policy. Omitting this collection step can result in poor results. For example, running state-of-the-art off-policy RL algorithms on trajectories collected from an expert policy can result in diverging Q-values [Kumar et al., 2019].

In contrast, in offline RL, the algorithm no longer has access to a simulator, and instead is presented with a fixed dataset of transitions $\mathcal{D}$, akin to supervised learning. However, in contrast to supervised learning, the optimal supervision at a state may not be provided, and the algorithm needs to implicitly perform complex decision making in order to act optimally. In the case that the dataset is generated by sampling trajectories from a single policy, the sampling policy is referred to as a behavior policy $\pi_B$. While off-policy RL algorithms can in principle be used in the offline setting, the lack of ability to collect additional data can break assumptions required for convergence, such as in online Q-learning [Sutton and Barto, 2018].

3 Task Design Factors

Offline RL presents unique challenges for existing RL algorithms. Offline RL algorithms must be able to handle extensive distribution shift, as well as data collected via unconventional means, such as through human demonstration or hand-designed controllers, which may not be representable by the chosen policy class. Moreover, the strongest convergence results in RL require conditions such as on-policy data collection [Bertsekas and Tsitsiklis, 1996], function class restrictions [Gordon, 1995] or restrictions on the MDP family [Agarwal et al., 2019a; Du et al., 2020].

In order to design a benchmark that provides a meaningful measure of progress towards realistic applications of offline RL, our benchmark suite is designed to cover a range of dataset and task properties. The main design factors relate to how the data distribution is generated. In practice, one may not have control over the type of data available, so we outline several properties to explore which we believe may be problematic for existing RL algorithms.

Undirected and multitask data (U) naturally arises when data is passively logged, such as recording user interactions on the internet or recording videos of a car for autonomous driving. This data may not necessarily be directed towards the specific task one is trying to accomplish. However, parts of trajectories in such a dataset can still provide useful information for the policy we are trying to learn. In navigation, one may be able to piece together parts of different sub-optimal trajectories to form a shortest path to a target goal. For example, in the figure to the upper-right, if an agent is given trajectories A and B in a dataset, it can form trajectory C by combining the corresponding halves of A and B in order to make the shortest path along the bottom half of the diagram. Undirected data may also be collected by purely exploratory agents that may be optimizing a different objective from evaluation or may not be optimizing any objective at all, such as agents driven via intrinsic motivation.

Suboptimal Agents (S). For tasks with a clear objective, as opposed to the undirected and multitask settings, the given datasets may not always represent behaviors from optimal agents. This in some sense represents a tradeoff in applications with imitation learning: in imitation learning we generally require expert datasets but do not require rewards to be specified. In contrast, with offline RL, we can still potentially learn optimal policies from suboptimal data but require rewards to be specified in order to provide a learning signal for the algorithm. We also note that this type of data is currently the predominant method for benchmarking offline RL algorithms within the field of deep RL [Fujimoto et al., 2018; Kumar et al., 2019; Wu et al., 2019].

Data generated from a non-RL policy (P). Behavior in realistic scenarios may not originate from learned neural network or linear controllers. For example, human demonstrations may utilize external cues for dataset generation that are not visible to the policy during training. This may make the dataset generation process non-Markovian, making it impossible to replicate the observed behavior with Markovian policies. Hand-designed controllers may not be representable by the learner’s policy class, introducing bias into the learning process [Lu et al., 2018]. However, offline RL algorithms must be able to learn effectively even when the underlying data is not representable by the algorithm class.

Narrow data distributions (N), such as those from deterministic expert policies, are particularly problematic for offline RL algorithms and may cause divergence both empirically [Fujimoto et al., 2018; Kumar et al., 2019].
and theoretically [Munos, 2003, Farahmand et al., 2010, Kumar et al., 2019, Agarwal et al., 2019a, Du et al., 2020]. Narrow datasets may commonly arise in situations such as when utilizing human demonstrations for autonomous driving, or when using hand-crafted policies for robotic control. An important challenge in offline RL is to be able to gracefully handle diverse data distributions without algorithms diverging or producing performance worse than the provided behavior.

In addition to the specific distributional properties discussed above, there are several additional benchmark-wide design considerations. We wish to include a variety of qualitatively different tasks, in order to provide diversity in the domains tested. Therefore, we include both locomotion and manipulation robotics tasks. We also provide tasks with a wide range of difficulty, from simple baseline tasks and tasks current algorithms can already solve to harder problems that are currently out of reach. Finally, for the purpose of comparability with prior works, we also include the OpenAI Gym robotic locomotion tasks used in Fujimoto et al. [2018], Kumar et al. [2019], Wu et al. [2019].

4 Tasks and Datasets

Given the properties outlined in Section 3 and taking into account ease of experimentation, we designed the following tasks and datasets. A tabular organization of each task, and what properties they correspond to, is provided in Table 1.

4.1 Diagnostic Tasks

We first introduce a collection of diagnostic tasks, that can be easily visualized for the purposes of debugging and prototyping algorithms.

Maze2D. The Maze2D domain is a navigation task requiring a 2D agent to reach a fixed goal location. The tasks are designed to test the ability of offline RL algorithms to be able to piece together parts of different trajectories in order to find the shortest path to a new goal. Three maze layouts are provided—a small U-shaped maze, referred to as maze2d-umaze, and two larger complex mazes referred to as maze2d-medium and maze2d-large. The “umaze” and “medium” mazes are shown to the right, and the “large” maze is shown below.

The data is generated by selecting goal location at random and then using a planner that generates sequences of waypoints that are followed using a PD controller. An example of this process is shown in the figure on the right where the agent begins at location (1). The waypoints, represented by green circles, are planned along the path to a goal (2), represented by a yellow star. Upon reaching a threshold distance to a waypoint, the controller updates its internal state to track the next waypoint along the path to the goal. Once a goal is reached, a new goal is selected (3) and the process continues. This data generation process provides both multitask data, due to random goals being chosen, and data generated via a non-markovian policy, since the controller maintains memory on the most recently achieved waypoint. An agent can theoretically outperform the best trajectory in the dataset by assembling different parts of trajectories. For example, an agent navigating from (1) to (3) can combine the first half of the green trajectory with the latter half of the yellow trajectory to form a successful path, even though no such path may have been realized in the dataset.

MiniGrid. The MiniGrid toolkit [Chevalier-Boisvert et al., 2018] involves navigating an agent within a small discrete Gridworld environment. It provides a discrete counterpart to the pointmass maze domain, in order to test behavior with multitask data. This benchmark consists of two tasks: minigrid-fourrooms and minigrid-fourrooms-random. Both of these tasks use the “Four Rooms” environment within the MiniGrid package, which is a gridworld navigation task inside a maze consisting of four large, 8x8 rooms connected via “doors”, or small openings, in the walls.
The minigrid-fourrooms task uses data collected via a planner tasked with navigating to randomly selected goal locations, and the minigrid-fourrooms-random task consists of data collected via uniform random actions.

4.2 Benchmark Tasks

Next, we introduce a collection of tasks designed to provide challenging benchmarks for evaluating offline RL algorithms.

Gym-MuJoCo. The gym benchmark tasks (Hopper, HalfCheetah, Walker2d) have been popularly used for evaluation in a lot of prior work in RL including prior work in offline RL [Fujimoto et al., 2018, Kumar et al., 2019, Wu et al., 2019]. In our first set of benchmark tasks, we provide standardized datasets with various properties, which we will describe next, for three standard Gym MuJoCo benchmarks – HalfCheetah, Hopper, and Walker2d considered in Kumar et al. [2019].

The first dataset consists of data generated by unrolling suboptimal, “medium-quality” policies for these three domains. This dataset was generated by first training a policy, online, using a standard RL algorithm (SAC) for a few steps, then stopping the training and collecting 1M samples from this partially trained policy. We name these datasets as hopper-medium-v0, halfcheetah-medium-v0 and walker2d-medium-v0. The second type of dataset is generated by unrolling a randomly initialized policy on these three domains. We refer to these datasets as:

hopper-random-v0, halfcheetah-random-v0 and walker2d-random-v0.

The next dataset on these environments was generated by mixing varying amounts of expert demonstrations and suboptimal data, generated via a partially trained policy or by unrolling a uniform-at-random policy. We combine $10^6$ samples of expert data with $10^6$ samples of a medium-quality data to generate these datasets. These datasets are referred to as:

halfcheetah-medium-expert-v0, walker2d-medium-expert-v0, and hopper-medium-expert-v0.

Our final type of dataset is generated by collecting all samples observed during a training run of an online SAC algorithm until the resulting policy achieves a desired level of return. This is equivalent to using multiple suboptimal policies contained in a first fraction of the replay buffer. We refer to these datasets as:

hopper-mixed-v0, halfcheetah-mixed-v0 and walker2d-mixed-v0.

AntMaze. In order to test the efficacy of offline RL algorithms with hand-designed or hardcoded planners, we devised a set of tasks where the goal is to control a MuJoCo ant in a navigation scenario. The data is generated via first separately training a goal reaching policy on the ant domain and then using it in conjunction with a higher level waypoint generator, which provides appropriate sub-goals to the lower-level goal conditioned policy in order to guide it fastest to the target location. We further designed three mazes of varying size and complexity – a small, easy U-maze, a medium-sized maze and a big maze which is hard to navigate. These mazes were previously used in the Maze2d domain with the task of controlling a point-mass. These tasks can be considered as high-dimensional and complex to control versions of the Maze2d tasks, with the emphasis on controlling an ant robot.

The datasets were generated via three procedures – a narrow dataset commanding the ant to go to a specific goal from a specific start location (antmaze-umaze-v0), a diverse dataset commanding the ant to go to a randomly sampled goal from a randomly sampled start location (antmaze-umaze-diverse-v0, antmaze-medium-diverse-v0, and antmaze-large-diverse-v0), and a “play” dataset [Lynch et al., 2019] where the ant is commanded specific hand-picked locations in the maze, starting from a different set of hand-picked start locations (antmaze-large-play-v0 and antmaze-medium-play-v0), thereby consisting of a set of undirected but structured, and narrow behaviors that cannot be represented in the space of Markovian
policies, and resembles human demo behavior that is structured and narrow, but not aimed at solving the evaluation task.

**Adroit.** The Adroit domain [Rajeswaran et al., 2018] (pictured right) involves controlling a 24-DoF simulated hand tasked with hammering a nail, opening a door, twirling a pen, or picking up and moving a ball. Three types of datasets are provided for each task - a small amount of demonstration data from a human (“human”), a large amount of expert data from a fine-tuned RL policy (“expert”). Because the original demonstration dataset only contains 25 trajectories per task, we also include a dataset generated by behavioral cloning a policy from the human demonstrations, and mixing data at a 50-50 ratio with the demonstrations, referred to as “cloned”. The mixing is performed because the cloned policy themselves do not successfully complete the task, making the dataset otherwise impossible to learn from.

The Adroit domain is designed to measure the effect of a narrow expert data distribution as well as performance when using human demonstrations on a high-dimensional robotic manipulation task. This corresponds to the narrow data distributions and non-RL policy dataset properties. Additionally, the Adroit domain has several unique properties that make it qualitatively different from the Gym MuJoCo tasks. First, the demonstration data is realistic and collected through a motion capture system. Second, each task is difficult to solve with standard online RL due to sparse rewards and exploration challenges, which make cloning and online RL alone insufficient in solving the task.

### 4.3 Interface and Evaluation

All tasks consist of a large offline dataset of transition samples for training, and a simulator for evaluation. The mapping is not one-to-one – several tasks use the same simulator with different datasets. All code is be available online at [https://github.com/rail-berkeley/d4rl](https://github.com/rail-berkeley/d4rl).

The simulator provides an OpenAI Gym [Brockman et al., 2016] interface, which consists of a `reset()` function that samples from the initial state distribution, and a `step(action)` function which simulates the successor state and reward given an action.

The dataset consists of a dictionary containing observations, actions and rewards, which are lists containing states, actions, and rewards sampled by executing a behavior policy inside the simulator. Additionally, each dataset may have multiple infos keys containing environment- and dataset-specific values that may be useful for debugging.

We additionally provide reference values to normalize scores for each environment roughly to the range between 0 and 100, by computing normalized score = \( \frac{\text{score} - \text{random score}}{\text{expert score} - \text{random score}} \). A normalized score of 0 corresponds to the average returns (over 100 episodes) of an agent taking actions uniformly at random across the action space. A score of 100 corresponds to the average returns of a domain-specific expert. For Maze2D and AntMaze domains, this corresponds to the performance of the hand-designed controller used to collect data. For Adroit, this corresponds to a policy trained with behavioral cloning on human-demonstrations and fine-tuned with RL. For Gym-MuJoCo, this corresponds to a soft-actor critic [Haarnoja et al., 2018] agent.

### 5 Benchmarking Prior Methods

We now present results for a number of recently proposed offline RL algorithms, as well as several baselines, on our proposed offline RL benchmarks. The purpose of this evaluation is to a) provide a useful baseline to gauge the difficulty of each task, and b) identify areas of shortcomings in existing offline RL algorithms in order to guide future research. As recent research in offline RL has been primarily done in the continuous action domain, for benchmarking purposes we leave out the Minigrid domain, as its action space is discrete.

For baseline algorithms, we use offline soft actor-critic (SAC) [Haarnoja et al., 2018], an off-policy actor-critic algorithm adapted to train exclusively from offline data, bootstrapping error accumulation reduction (BEAR) [Kumar et al., 2019], an actor-critic algorithm that uses a support constraint for the policy, and BRAC [Wu et al., 2019], an actor-critic algorithm with policy regularization or value function penalties. Additionally, we report results for behavioral cloning and SAC evaluated in the standard online setting to
| Domain          | Task Name                        | Controller Type | # Samples | Properties |
|-----------------|----------------------------------|-----------------|-----------|------------|
| Maze2D          | maze2d-umaze                     | Planner         | $10^6$    | U, P       |
|                 | maze2d-medium                    | Planner         | $10^6$    | U, P       |
|                 | maze2d-large                     | Planner         | $10^6$    | U, P       |
| MiniGrid        | minigrid-fourrooms               | Planner         | $10^6$    | U          |
|                 | minigrid-fourrooms-random        | Policy          | $10^5$    | U          |
| AntMaze         | antmaze-umaze                    | Planner         | $10^6$    | U, N       |
|                 | antmaze-umaze-diverse           | Planner         | $10^6$    | U, P       |
|                 | antmaze-medium-play              | Planner         | $10^6$    | U, P, N    |
|                 | antmaze-medium-diverse           | Planner         | $10^6$    | U, P       |
|                 | antmaze-large-play               | Planner         | $10^6$    | U, P, N    |
|                 | antmaze-large-diverse            | Planner         | $10^6$    | U, P       |
| Adroit          | pen-human                        | Human           | 5000      | P, N       |
|                 | pen-cloned                       | Policy          | $5 \times 10^5$ | N       |
|                 | pen-expert                       | Policy          | $5 \times 10^5$ | N       |
|                 | hammer-human                     | Human           | 11310     | P, N       |
|                 | hammer-cloned                    | Policy          | $10^6$    | N          |
|                 | hammer-expert                    | Policy          | $10^6$    | N          |
|                 | door-human                       | Human           | 6729      | P, N       |
|                 | door-cloned                      | Policy          | $10^6$    | N          |
|                 | door-expert                      | Policy          | $10^6$    | N          |
|                 | relocate-human                   | Human           | 9942      | P, N       |
|                 | relocate-cloned                  | Policy          | $10^6$    | N          |
|                 | relocate-expert                  | Policy          | $10^6$    | N          |
| Gym-MuJoCo      | hopper-random                    | Policy          | $10^6$    | S          |
|                 | hopper-medium                    | Policy          | $10^6$    | S, N       |
|                 | hopper-mixed                     | Policy          | 200920    | S          |
|                 | hopper-medium-expert             | Policy          | $2 \times 10^6$ | S       |
|                 | halfcheetah-random               | Policy          | $10^6$    | S          |
|                 | halfcheetah-medium               | Policy          | $10^6$    | S, N       |
|                 | halfcheetah-mixed                | Policy          | 101000    | S          |
|                 | halfcheetah-medium-expert        | Policy          | $2 \times 10^6$ | S       |
|                 | walker2d-random                  | Policy          | $10^6$    | S          |
|                 | walker2d-medium                  | Policy          | $10^6$    | S, N       |
|                 | walker2d-mixed                   | Policy          | 100930    | S          |
|                 | walker2d-medium-expert           | Policy          | $2 \times 10^6$ | S       |

Table 1: Statistics and properties for each task in the benchmark. For the controller type, “planner” refers to a hand-designed navigation planner, “human” refers to human demonstrations, and “policy” refers to random or neural network policies. The number of samples refers to the number of environment transitions recorded in the dataset.
provide additional points of comparison. In most domains, we expect online SAC to outperform offline algorithms when given the same amount of data, since the algorithm is able to collect on-policy data. There are a few exceptions, such as for environments where exploration challenges (such as the Adroit domain) make it difficult for RL algorithms to find reward signal, where offline RL may have advantages. Aggregated results for all algorithms, normalized to lie approximately between 0 and 100, are reported in Table 2. The raw, unnormalized scores can be found in Appendix A.

We find the most success for the evaluated offline methods in scenarios where data is generated via some previously trained policy, such as in the Adroit and Gym-MuJoCo domains. In these domains, offline RL algorithms are able to match expert performance when given a sizable amount of export data, and outperform the behavior policy when given suboptimal data. In these settings, existing algorithms are able to gracefully handle varying qualities of single-policy datasets, and match or exceed behavioral cloning. This positive result is somewhat expected, however, as this is the predominant setting where algorithms have been benchmarked on. However, datasets generated by a mixture of policies of varying qualities, such as the medium-expert datasets or the mixed datasets often tend to be challenging for current algorithms, with mixed success and failure results, indicating that data generated from a non-RL policy can be challenging even on these previously studied domains.

In general, we find that tasks with the undirected data property, such as the Maze2D and AntMaze domains, are challenging for existing methods. Even in the simple Maze2D domain, all evaluated offline methods lag significantly behind online performance, even when noise is injected into data collecting policy, which typically makes offline learning easier. We hypothesize that in continuous spaces, the evaluated algorithms still find it difficult to combine different pieces of trajectories found in the dataset in order to assemble a successful shortest path trajectory to a target location.

We also note that the human demonstration setting (Adroit) is somewhat inconclusive. The evaluated methods performed poorly, but the datasets contain very few samples (in the order of \(10^4\) samples) as human demonstrations for a particular task are somewhat expensive to collect. The original work using this dataset [Rajeswaran et al., 2018] was able to successfully learn policies by combining imitation learning on the demonstrations with online RL, but found the demonstrations themselves insufficient for reasonable success rates.

6 Related Work

Recently proposed evaluations for offline or batch RL algorithms have primarily been instantiated as learning from a fixed dataset of behaviors generated by a previously trained behavior policy. The quality of this agent may range from the random behavior of an initial policy to near-expert behavior from a fully trained policy. This evaluation protocol has been used in domains such as continuous control for robotics [Fujimoto et al., 2018] [Kumar et al., 2019] [Wu et al., 2019], navigation [Laroche et al., 2019], industrial control [Hein et al., 2017], and Atari video games [Agarwal et al., 2019b]. While this method may be adequate for demonstrating progress towards the more traditional, policy improvement-centric goal of offline RL, our focus in this work is primarily to use offline RL as a means to scale RL to large datasets. Thus, these benchmarks lack properties that might be seen in large, cheaply collected datasets, such as behavior from multiple tasks and human demonstrations, which can adversely affect algorithm performance.

Offline reinforcement learning using large datasets has also been used in real-world systems where evaluation using a simulator is not possible, such as in robotics [Cabi et al., 2019] and dialogue systems [Henderson et al., 2008] [Pietquin et al., 2011] [Jaques et al., 2019]. Moreover, significant efforts have been made to incorporate large-scale datasets into off-policy RL [Kalashnikov et al., 2018] [Mo et al., 2018] [Gu et al., 2017], but these works generally use large numbers of robots to collect online interaction during training. We believe these to be promising directions for future research, but the primary goal of this work is to provide an effective platform for developing algorithms, and simulated environments to enable cheap and reliable evaluation and comparative benchmarking.
| Domain   | Task Name                  | SAC  | BC    | SAC-off | BEAR   | BRAC-p | BRAC-v |
|----------|----------------------------|------|-------|---------|--------|--------|--------|
| Maze2D   | maze2d-umaze               | 26.9 | -0.4  | -0.0    | -1.2   | 0.3    | 1.8    |
|          | maze2d-medium              | 0.5  | -5.4  | -4.7    | -3.2   | -6.1   | -5.9   |
|          | maze2d-large               | 0.6  | -0.4  | -1.4    | -1.7   | -2.0   | -0.6   |
|          | antmaze-umaze              | 0.0  | 65.0  | 0.0     | 73.0   | 50.0   | 70.0   |
|          | antmaze-umaze-diverse      | 0.0  | 55.0  | 0.0     | 61.0   | 40.0   | 70.0   |
|          | antmaze-medium-play        | 0.0  | 0.0   | 0.0     | 0.0    | 0.0    | 0.0    |
|          | antmaze-medium-diverse     | 0.0  | 0.0   | 0.0     | 0.0    | 0.0    | 0.0    |
|          | antmaze-large-play         | 0.0  | 0.0   | 0.0     | 0.0    | 0.0    | 0.0    |
|          | antmaze-large-diverse      | 0.0  | 0.0   | 0.0     | 0.0    | 0.0    | 0.0    |
| AntMaze  | pen-human                  | 21.6 | 34.4  | 6.3     | -1.0   | 8.1    | 0.6    |
|          | pen-cloned                 | 21.6 | 56.9  | 23.5    | 26.5   | 1.6    | -2.5   |
|          | pen-expert                 | 21.6 | 85.1  | 6.1     | 105.9  | -3.5   | -3.0   |
|          | hammer-human               | 0.2  | 1.5   | 0.5     | 0.3    | 0.3    | 0.2    |
|          | hammer-cloned              | 0.2  | 0.8   | 0.2     | 0.3    | 0.3    | 0.3    |
|          | hammer-expert              | 0.2  | 125.6 | 25.2    | 127.3  | 0.3    | 0.3    |
|          | door-human                 | -0.2 | 0.5   | 3.9     | -0.3   | -0.3   | -0.3   |
|          | door-cloned                | -0.2 | -0.1  | 0.0     | -0.1   | -0.1   | -0.1   |
|          | door-expert                | -0.2 | 34.9  | 7.5     | 103.4  | -0.3   | -0.3   |
|          | relocate-human             | -0.2 | 0.0   | 0.0     | -0.3   | -0.3   | -0.3   |
|          | relocate-cloned            | -0.2 | -0.1  | -0.2    | -0.3   | -0.3   | -0.3   |
|          | relocate-expert            | -0.2 | 101.3 | -0.3    | 98.6   | -0.3   | -0.4   |
| Adroit   | halfcheetah-random         | 100.0| 2.1   | 30.5    | 25.5   | 23.5   | 28.1   |
|          | halfcheetah-medium         | 100.0| 36.1  | -4.3    | 38.6   | 44.0   | 45.5   |
|          | halfcheetah-mixed          | 100.0| 38.4  | -2.4    | 36.2   | 45.6   | 45.9   |
|          | halfcheetah-medium-expert  | 100.0| 35.8  | 1.8     | 51.7   | 43.8   | 45.3   |
|          | walker2d-random            | 100.0| 1.6   | 4.1     | 6.7    | 0.8    | 0.5    |
|          | walker2d-medium            | 100.0| 6.6   | 0.9     | 33.2   | 72.7   | 81.3   |
|          | walker2d-mixed             | 100.0| 11.3  | 1.9     | 10.8   | -0.3   | 0.9    |
|          | walker2d-medium-expert     | 100.0| 6.4   | -0.1    | 26.0   | 3.1    | 66.6   |
|          | hopper-random              | 100.0| 9.8   | 11.3    | 9.5    | 11.1   | 12.0   |
|          | hopper-medium              | 100.0| 29.0  | 0.8     | 47.6   | 31.2   | 32.3   |
|          | hopper-mixed               | 100.0| 11.8  | 3.5     | 25.3   | 0.7    | 0.8    |
|          | hopper-medium-expert       | 100.0| 111.9 | 1.6     | 4.0    | 1.1    | 0.8    |

Table 2: Normalized evaluation results comparing online SAC (SAC), offline SAC (SAC-off), bootstrapping error reduction (BEAR), behavior-regularized actor critic using policy (BRAC-p) or value (BRAC-v) regularization, and behavioral cloning (BC). Results are reported as an average over 3 seeds on the final iteration of training, and normalized to roughly lie between 0 and 100 (higher is better). A score of 0 corresponds to a random policy, and 100 corresponds to a domain-specific expert.
7 Discussion

We have proposed an open-source benchmark for offline reinforcement learning. The selection of the benchmark tasks were motivated by properties that we believe realistic data is likely to have, such as narrow data distributions, and undirected or multitask behavior. Existing benchmarks have largely been concentrated on robotic control using data generated by previously optimized RL algorithms [Fujimoto et al., 2018, Kumar et al., 2019, Wu et al., 2019], which in our view, can give a misleading sense of progress as this is a particularly narrow application of offline RL. Indeed, our evaluations reveal a lack of ability for existing offline RL algorithms to handle properties such as undirected data, which may be crucial for the success of offline RL in many real-world domains.

Ultimately, we would like to see offline RL applications move from simulated domains to real-world domains where significant amounts of offline data is easily obtainable. This includes exciting areas such as recommender systems and natural language interfaces, where user behavior can be easily logged, and medical diagnoses, where doctors must record symptoms, diagnoses, and treatments in complete medical records for each patient. A key challenge for developing algorithms on these domains is that reliable evaluation must be done in a real system, which significantly slows down the pace at which one can iteratively improve an algorithm.

Offline RL holds great promise as a potential paradigm to leverage vast amounts of existing sequential data within the flexible decision making framework of reinforcement learning. We hope that providing a benchmark that is representative of potential problems in offline RL, but that still can be cheaply evaluated in simulation, will greatly accelerate progress in this field and create new opportunities to apply RL in many real-world application areas.

References

Alekh Agarwal, Sham M Kakade, Jason D Lee, and Gaurav Mahajan. Optimality and approximation with policy gradient methods in markov decision processes. ArXiv preprint ArXiv:1908.00261, 2019a.

Rishabh Agarwal, Dale Schuurmans, and Mohammad Norouzi. Striving for simplicity in off-policy deep reinforcement learning. CoRR, abs/1907.04543, 2019b. URL http://arxiv.org/abs/1907.04543.

Dimitri P Bertsekas and John N Tsitsiklis. Neuro-dynamic programming. Athena Scientific, 1996.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. ArXiv preprint ArXiv:1606.01540, 2016.

Serkan Cabi, Sergio Gómez Colmenarejo, Alexander Novikov, Ksenia Konyushkova, Scott Reed, Rae Jeong, Konrad ˙Zołna, Yusuf Aytar, David Budden, Mel Vecerik, et al. A framework for data-driven robotics. ArXiv preprint ArXiv:1909.12200, 2019.

Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. https://github.com/maximecb/gym-minigrid 2018.

Simon S. Du, Sham M. Kakade, Ruosong Wang, andLin F. Yang. Is a good representation sufficient for sample efficient reinforcement learning? In International Conference on Learning Representations, 2020. URL https://openreview.net/forum?id=rigenAVKPE.

Amir-massoud Farahmand, Csaba Szepesvári, and Rémi Munos. Error propagation for approximate policy and value iteration. In Advances in Neural Information Processing Systems, pages 568–576, 2010.

Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. ArXiv preprint ArXiv:1812.02900, 2018.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.

Geoffrey J Gordon. Stable function approximation in dynamic programming. In Machine Learning Proceedings 1995, pages 261–268. Elsevier, 1995.
Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In 2017 IEEE international conference on robotics and automation (ICRA), pages 3389–3396. IEEE, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. arXiv preprint arXiv:1801.01290, 2018.

Daniel Hein, Steffen Udluft, Michel Tokic, Alexander Hentschel, Thomas A Runkler, and Volkmar Sterzing. Batch reinforcement learning on the industrial benchmark: First experiences. In 2017 International Joint Conference on Neural Networks (IJCNN), pages 4214–4221. IEEE, 2017.

James Henderson, Oliver Lemon, and Kalliirroi Georgila. Hybrid reinforcement/supervised learning of dialogue policies from fixed data sets. Computational Linguistics, 34(4):487–511, 2008.

Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Àgata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind W. Picard. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. CoRR, abs/1907.00456, 2019. URL http://arxiv.org/abs/1907.00456.

Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In Conference on Robot Learning, pages 651–673, 2018.

Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In Advances in Neural Information Processing Systems, 2019. URL http://arxiv.org/abs/1906.00949.

Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In Reinforcement learning, pages 45–73. Springer, 2012.

Romain Laroche, Paul Trichelair, and Remi Tachet Des Combes. Safe policy improvement with baseline bootstrapping. In International Conference on Machine Learning (ICML), 2019.

Long-Ji Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. Machine learning, 8(3-4):293–321, 1992.

Tyler Lu, Dale Schuurmans, and Craig Boutilier. Non-delusional q-learning and value-iteration. In Advances in neural information processing systems, pages 9949–9959, 2018.

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. Conference on Robot Learning (CoRL), 2019. URL https://arxiv.org/abs/1903.01973.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. Nature, 518(7540):529–533, February 2015. ISSN 00280836.

Kaichun Mo, Haoxiang Li, Zhe Lin, and Joon-Young Lee. The adobeindoornav dataset: Towards deep reinforcement learning based real-world indoor robot visual navigation. 2018.

Rémi Munos. Error bounds for approximate policy iteration. In Proceedings of the Twentieth International Conference on International Conference on Machine Learning, pages 560–567. AAAI Press, 2003.

Olivier Pietquin, Matthieu Geist, Senthilkumar Chandramohan, and Hervé Frezza-Buet. Sample-efficient batch reinforcement learning for dialogue management optimization. ACM Transactions on Speech and Language Processing (TSLP), 7(3):1–21, 2011.
Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. In *Robotics: Science and Systems*, 2018.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. Second edition, 2018.

Christopher J.C.H. Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.
## Appendices

### A Unnormalized Evaluation Scores

The raw, un-normalized scores for each task and algorithm are reported in the table below. These scores represent the undiscounted return obtained from executing a policy in the simulator, averaged over 3 random seeds.

| Domain  | Task Name          | SAC  | BC   | SAC-off | BEAR  | BRAC-p | BRAC-v |
|---------|--------------------|------|------|---------|-------|--------|--------|
| Maze2D  | maze2d-umaze       | 17.5 | 0.7  | 0.9     | 0.2   | 1.1    | 2.1    |
|         | maze2d-medium      | 6.2  | 1.5  | 2.0     | 3.2   | 0.9    | 1.1    |
|         | maze2d-large       | 6.0  | 4.1  | 2.2     | 1.6   | 1.1    | 3.8    |
| AntMaze | ant-umaze          | 0.0  | 0.7  | 0.0     | 0.7   | 0.5    | 0.7    |
|         | ant-umaze-diverse  | 0.0  | 0.6  | 0.0     | 0.6   | 0.4    | 0.7    |
|         | ant-medium-play    | 0.0  | 0.0  | 0.0     | 0.0   | 0.0    | 0.0    |
|         | ant-medium-diverse | 0.0  | 0.0  | 0.0     | 0.1   | 0.0    | 0.0    |
|         | ant-large-play     | 0.0  | 0.0  | 0.0     | 0.0   | 0.0    | 0.0    |
|         | ant-large-diverse  | 0.0  | 0.0  | 0.0     | 0.0   | 0.0    | 0.0    |
| Adroit  | pen-human          | 739.3| 1121.9| 284.8  | 66.3  | 339.0  | 114.7  |
|         | pen-cloned         | 739.3| 1791.8| 797.6  | 885.4 | 143.4  | 22.2   |
|         | pen-expert         | 739.3| 2633.7| 277.4  | 3254.1| -7.8   | 6.4    |
|         | hammer-human       | -248.7| -82.4| -214.2  | -242.0| -239.7 | -243.8 |
|         | hammer-cloned      | -248.7| -175.1| -244.1  | -241.1| -236.7 | -236.9 |
|         | door-human         | -61.8 | -41.7| 57.2    | -66.4 | -66.5  | -66.4  |
|         | door-cloned        | -61.8 | -60.7| -56.3   | -60.9 | -58.7  | -59.0  |
|         | door-expert        | -61.8 | 969.4| 163.8   | 2980.1| -66.4  | -66.6  |
|         | relocate-human     | -13.7 | -5.6 | -4.5    | -18.9 | -19.7  | -19.7  |
|         | relocate-cloned    | -13.7 | -10.1| -16.1   | -17.6 | -19.8  | -19.4  |
|         | relocate-expert    | -13.7 | 4289.3| -18.2  | 4173.8| -20.6  | -21.4  |
| Gym     | halfcheetah-random | 12135.0| -17.9| 3502.0  | 2885.6| 2641.0 | 3207.3 |
|         | halfcheetah-medium | 12135.0| 4196.4| -808.6  | 4508.7| 5178.2 | 5365.3 |
|         | halfcheetah-mixed  | 12135.0| 4492.1| -581.3  | 4211.3| 5384.7 | 5413.8 |
|         | halfcheetah-medium-expert | 12135.0| 4169.4| -55.7  | 6132.5| 5156.0 | 5342.4 |
|         | walker2d-random    | 4592.3| 73.0  | 192.0   | 307.6 | 38.4   | 23.9   |
|         | walker2d-medium    | 4592.3| 304.8| 44.2    | 1526.7| 3341.1 | 3734.4 |
|         | walker2d-mixed     | 4592.3| 518.6| 87.8    | 495.3 | -11.5  | 44.5   |
|         | walker2d-medium-expert | 4592.3| 297.0| -5.1    | 1193.6| 141.7  | 3058.9 |
|         | hopper-random      | 3234.3| 299.4| 347.7   | 289.5 | 341.0  | 370.5  |
|         | hopper-medium      | 3234.3| 923.5| 5.7     | 1527.9| 994.8  | 1030.0 |
|         | hopper-mixed       | 3234.3| 364.4| 93.3    | 802.7 | 2.0    | 5.3    |
|         | hopper-medium-expert | 3234.3| 3621.2| 32.9  | 109.8 | 16.0   | 5.1    |