DEEPAVERAGERS: OFFLINE REINFORCEMENT LEARNING BY SOLVING DERIVED NON-PARAMETRIC MDPS

Aayam Shrestha, Stefan Lee, Prasad Tadepalli, Alan Fern
Oregon State University
Corvallis, OR 97330, USA
{shrestaa, leestef, tadepall, alan.fern}@oregonstate.edu

ABSTRACT

We study an approach to offline reinforcement learning (RL) based on optimally solving finitely-represented MDPs derived from a static dataset of experience. This approach can be applied on top of any learned representation and has the potential to easily support multiple solution objectives as well as zero-shot adjustment to changing environments and goals. Our main contribution is to introduce the Deep Averagers with Costs MDP (DAC-MDP) and to investigate its solutions for offline RL. DAC-MDPs are a non-parametric model that can leverage deep representations and account for limited data by introducing costs for exploiting under-represented parts of the model. In theory, we show conditions that allow for lower-bounding the performance of DAC-MDP solutions. We also investigate the empirical behavior in a number of environments, including those with image-based observations. Overall, the experiments demonstrate that the framework can work in practice and scale to large complex offline RL problems.

1 INTRODUCTION

Research in automated planning and control has produced powerful algorithms to solve for optimal, or near-optimal, decisions given accurate environment models. Examples include the classic value- and policy-iteration algorithms for tabular representations or more sophisticated symbolic variants for graphical model representations (e.g. Boutilier et al. (2000); Raghavan et al. (2012)). In concept, these planners address many of the traditional challenges in reinforcement learning (RL). They can perform "zero-shot transfer" to new goals and changes to the environment model, accurately account for sparse reward or low-probability events, and solve for different optimization objectives (e.g. robustness). Effectively leveraging these planners, however, requires an accurate model grounded in observations and expressed in the planner's representation. On the other hand, model-based reinforcement learning (MBRL) aims to learn grounded models to improve RL's data efficiency. Despite developing grounded environment models, the vast majority of current MBRL approaches do not leverage near-optimal planners to help address the above challenges. Rather, the models are used as black-box simulators for experience augmentation and/or Monte-Carlo search. Alternatively, model learning is sometimes treated as purely an auxiliary task to support representation learning.

The high-level goal of this paper is to move toward MBRL approaches that can effectively leverage near-optimal planners for improved data efficiency and flexibility in complex environments. However, there are at least two significant challenges. First, there is a mismatch between the deep model representations typically learned in MBRL (e.g. continuous state mappings) and the representations assumed by many planners (e.g. discrete tables or graphical models). Second, near-optimal planners are well-known for exploiting model inaccuracies in ways that hurt performance in the real environment, e.g. (Atkeson, 1998). This second challenge is particularly significant for offline RL, where the training experience for model learning is fixed and limited.

We address the first challenge above by focusing on tabular representations, which are perhaps the simplest, but most universal representation for optimal planning. Our main contribution is an offline MBRL approach based on optimally solving a new model called the Deep Averagers with Costs MDP (DAC-MDP). A DAC-MDP is a non-parametric model derived from an experience dataset and a corresponding (possibly learned) latent state representation. While the DAC-MDP is defined over the entire continuous latent state space, its full optimal policy can be computed by solving a standard (finite) tabular MDP derived from the dataset. This supports optimal planning via any

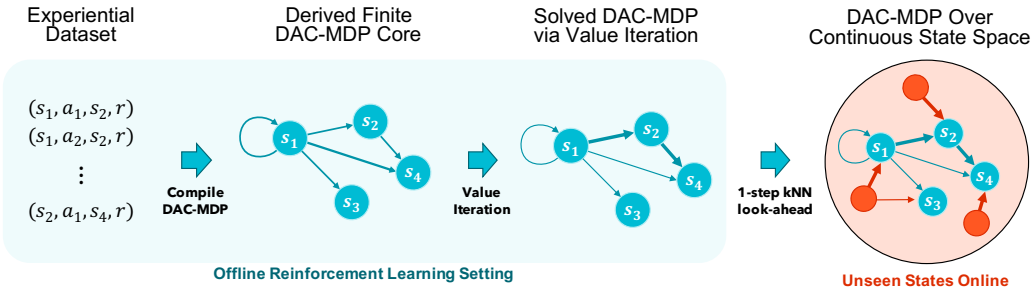

Figure 1: Overview of Offline RL via DAC-MDPs. Given a static experience dataset, we first compile it into a finite tabular MDP which is at most the size of the dataset. This MDP contains the "core" states of the full continuous DAC-MDP. The finite core-state MDP is then solved via value iteration, resulting in a policy and Q-value function for the core states. This finite Q-function is used to define a non-parametric Q-function for the continuous DAC-MDP, which allows for Q-values and hence a policy to be computed for previously unseen states.

tabular MDP solver, e.g. value iteration. To scale this approach to typical offline RL problems, we develop a simple GPU implementation of value iteration that scales to millions of states. As an additional engineering contribution, this implementation will be made public.

To address the second challenge of model inaccuracy due to limited data, DAC-MDPs follow the *pessimism in the face of uncertainty* principle, which has been shown effective in a number of prior contexts (e.g. (Fonteneau et al., 2013)). In particular, DAC-MDPs extend Gordon's Averagers framework (Gordon, 1995) with additional costs for exploiting transitions that are under-represented in the data. Our second contribution is to give a theoretical analysis of this model, which provides conditions under which a DAC-MDP solution will perform near optimally in the real environment.

Our final contribution is to empirically investigate the DAC-MDP approach using simple latent representations derived from random projections and those learned by Q-iteration algorithms. Among other results, we demonstrate the ability to scale to Atari-scale problems, which is the first demonstration of optimal planning being effectively applied across multiple Atari games. In addition, we provide case studies in 3D first-person navigation that demonstrate the flexibility and adaptability afforded by integrating optimal planning into offline MBRL. These results show the promise of our approach for marrying advances in representation learning with optimal planning.

2 Formal Preliminaries

A Markov Decision Process (MDP) is a tuple $\langle \mathcal{S}, \mathcal{A}, T, R \rangle$ (Puterman, 1994), with state set $\mathcal{S}$, action set $\mathcal{A}$, transition function $T(s, a, s')$, and reward function $R(s, a)$. A policy π maps states to actions and has Q-function $Q^\pi(s, a)$ giving the expected infinite-horizon β-discounted reward of following π after taking action a in s. The optimal policy π^* maximizes the Q-function over all policies and state-action pairs. Q^* corresponds to the optimal Q-function that satisfies $\pi^*(s) = \arg\max_a Q^*(s, a)$. Q^* can be computed given the MDP by repeated application of the *Bellman Backup Operator* B, which for any Q-function Q, returns a new Q-function given by,

$$B[Q](s, a) = R(s, a) + \gamma \mathbb{E}_{s' \sim T(s,a,\cdot)} \left[\max_a Q(s', a) \right]. \quad (1)$$

The objective of RL is to find a near-optimal policy without prior knowledge of the MDP. In the online RL setting, this is done by actively exploring actions in the environment. Rather, in the offline RL, which is the focus of this paper, learning is based on a static dataset $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}$, where each tuple gives the reward r_i and next state s'_i observed after taking action a_i in state s_i.

In strict offline RL setting, the final policy selection must be done using only the dataset, without direct access to the environment. This includes all hyperparameter tuning and the choice of when to stop learning. Evaluations of offline RL, however, often blur this distinction, for example, reporting performance of the best policy obtained across various hyperparameter settings as evaluated via new online experiences (Gulcehre et al., 2020). Here we consider an evaluation protocol that makes the amount of online access to the environment explicit. In particular, the offline RL algorithm is

allowed to use the environment to evaluate N_e policies (e.g. an average over repeated trials for each policy), which, for example, may derive from different hyperparameter choices. The best of the evaluated policies can then be selected. Note that $N_e = 1$ corresponds to pure offline RL.

3 DEEP AVERAGERS WITH COSTS MDPS (DAC-MDPS)

From a practical perspective our approach carries out the following steps as illustrated in Figure 1. (1) We start with a static experience dataset, where the states are assumed to come from a continuous latent state space. For example, states encoded via random or learned deep representations. (2) Next we compile the dataset into a tabular MDP over the "core states" of the DAC-MDP (those in the dataset). This compilation uses k-nearest neighbor (kNN) queries to define the reward and transition functions (Equation 2) functions of the core states. (3) Next we use a GPU implementation of value iteration to solve for the tabular MDP's optimal Q-function. (4) Finally, this tabular Q-function is used to define the Q-function over the entire DAC-MDP (Equation 3). Previously unseen states at test time are assigned Q-values and in turn a policy action via kNN queries over the core states.

Conceptually, our DAC-MDP model is inspired by Gordon's (1995) early work that showed the convergence of (offline) approximate value iteration for a class of function approximators; *averagers*, which includes methods such as k nearest neighbor (kNN) regression, among others. It was also observed that approximate value iteration using an averager was equivalent to solving an MDP derived from the offline data. That observation, however, was not investigated experimentally and has yet to be integrated with deep representation learning. Here we develop and evaluate such an integration.

The quality of averagers MDP, and model-learning in general, depends on the size and distribution of the dataset. In particular, an optimal planner can exploit inaccuracies in the underrepresented parts of the state-action space, which can lead to poor performance in the real environment. The DAC-MDPs aim to avoid this by augmenting the derived MDPs with costs/penalties on under-represented transitions. This turns out to be essential to achieving good performance on challenging benchmarks.

3.1 DAC-MDP DEFINITION

A DAC-MDP is defined in terms of an experience dataset $\mathcal{D} = \{(s_i, a_i, r_i, s'_i)\}$ from the true MDP $\mathcal{M}$ with continuous latent state space $\mathcal{S}$ and finite action space $\mathcal{A}$. The DAC-MDP $\tilde{M} = (\mathcal{S}, \mathcal{A}, \tilde{R}, \tilde{T})$ shares the same state and action spaces as $\mathcal{M}$, but defines the reward and transition functions in terms of empirical averages over the k nearest neighbors of (s, a) in $\mathcal{D}$.

The distance metric $d(s, a, s', a')$ considers state action pairs with different actions to be infinitely distant. Otherwise, the distance between pairs with the same action is the euclidean distance between their states. In particular, the distance between (s, a) and a data tuple (s_i, a_i, r_i, s'_i) is given by $d(s, a, s_i, a_i)$. Also, we let $kNN(s, a)$ denote the set of indices of the k nearest neighbors to (s, a) in $\mathcal{D}$, noting that the dependence on $\mathcal{D}$ and d is left implicit. Given hyperparameters k (smoothing factor) and C (cost factor) we can now specify the DAC-MDP reward and transition function.

$$\tilde{R}(s, a) = \frac{1}{k} \sum_{i \in kNN(s,a)} r_i - C \cdot d(s, a, s_i, a_i), \quad \tilde{T}(s, a, s') = \frac{1}{k} \sum_{i \in kNN(s,a)} I[s' = s'_i] \quad (2)$$

The reward for (s, a) is simply the average reward of the nearest neighbors with a penalty for each neighbor that grows linearly with the distance to a neighbor. Thus, the farther (s, a) is to its nearest neighbor set, the less desirable its immediate reward will be. The transition function is simply the empirical distribution over destination states of the nearest neighbor set.

Importantly, even though a DAC-MDP has an infinite continuous state space, it has a special finite structure. Since the transition function $\tilde{T}$ only allows transitions to states appearing as destination states in $\mathcal{D}$. We can view $\tilde{M}$ as having a finite core set of states $\mathcal{S}_D = \{s'_i \mid (s_i, a_i, r_i, s'_i) \in \mathcal{D}\}$. States in this core do not transition to non-core states and each non-core state immediately transitions to the core for any action. Hence, the value of core states is not influenced by non-core states. Further, once the core values are known, we can compute the values of any non-core state via one-step look ahead using $\tilde{T}$. *Thus, we can optimally solve a DAC-MDP by solving just its finite core.*

Specifically, let $\tilde{Q}$ be the optimal Q-function of $\tilde{M}$. We can compute $\tilde{Q}$ for the core states by solving the finite MDP $\tilde{M}_D = (\mathcal{S}_D, \mathcal{A}, \tilde{R}, \tilde{T})$. We can then compute $\tilde{Q}$ for any non-core state on demand via the following one-step look-ahead expression. This allows us to compute the optimal policy of

$\tilde{M}$, denoted $\tilde{\pi}$, using any solver of finite MDPs.

$$\tilde{Q}(s,a) = \frac{1}{k} \sum_{i \in kNN(s,a)} r_i + \gamma \max_a \tilde{Q}(s_i', a) - C \cdot d(s, a, s_i, a_i) \tag{3}$$

3.2 DAC-MDP Performance Lower Bound

We are ultimately interested in how well the optimal DAC-MDP $\tilde{M}$ policy $\tilde{\pi}$ performs in the true MDP $\mathcal{M}$. Without further assumptions, $\tilde{\pi}$ can be arbitrarily sub-optimal in $\mathcal{M}$, due to potential "non-smoothness" of values, limited data, and limited neighborhood sizes. We now provide a lower-bound on the performance of $\tilde{\pi}$ in $\mathcal{M}$ that quantifies the dependence on these quantities. Smoothness is characterized via a Lipschitz smoothness assumptions on $B[\tilde{Q}]$, where B is the Bellman operator for the true MDP and $\tilde{Q}$ is the optimal Q-function for $\tilde{M}$ using hyperparameters k and C. In particular, we assume that there is a constant $L(k, C)$, such that for any state-action pairs (s, a) and (s', a')

Lipschitz Smoothness Assumption:

$$\left| B[\tilde{Q}](s,a) - B[\tilde{Q}](s',a') \right| \leq L(k,C) \cdot d(s,a,s',a').$$

This quantifies the smoothness of the Q-function obtained via one-step look-ahead using the true dynamics The coverage of the dataset is quantified in terms of the worst case average distance to a kNN set defined by $\bar{d}_{max}$=$\max_{(s,a)} \frac{1}{k} \sum_{i \in kNN(s,a)} d(s,a,s_i,a_i)$. The bound also depends on Q_{max}=$\max_{(s,a)} \tilde{Q}(s,a)$ and $M_{N,k}$, which is the maximum number of distinct kNN sets over a dataset of size N. For $L2$ distance over a d-dimensional space,$M_{N,k}$ is bounded by $O\left((kN)^{\frac{d}{2}+1}\right)$.

Theorem 3.1. *For any data set $\mathcal{D}$ of size N, let $\tilde{Q}$ and $\tilde{\pi}$ be the optimal Q-function and policy for the corresponding DAC-MDP with parameters k and C. If $B[\tilde{Q}]$ is Lipschitz continuous with constant $L(k, C)$, then with probability at least $1 - \delta$,*

$$V^{\tilde{\pi}} \geq V^* - \frac{2\left(L(k,C) \cdot \bar{d}_{max} + Q_{max}\epsilon(k,N,\delta)\right)}{1-\gamma}, \quad \epsilon(k,N,\delta) = \sqrt{\frac{1}{2k} \ln \frac{2M_{N,k}}{\delta}},$$

which for L2 distance over a d-dimensional space yields $\epsilon(k,N,\delta) = O\left(\sqrt{\frac{1}{k}\left(d \ln kN + \ln \frac{1}{\delta}\right)}\right)$.

The full proof is in the Appendix **??**. The first term in the bound characterizes how well the dataset represents the MDP from a nearest neighbor perspective. In general, $\bar{d}_{max}$ decreases as the dataset becomes larger. The second term characterizes the variance of stochastic state transitions in the dataset and decreases with the smoothness parameter k. Counter-intuitively we see that for Euclidean distance, the second term can grow logarithmically in the dataset size.This worst-case behavior is due to not making assumptions about the distribution of source states-actions in the dataset. Thus, we can't rule out adversarial choices that negatively influence the kNN sets of critical states.

Practical Issues: There are some key practical issues of the framework regarding the scalability of VI and selection of hyperparameters. We exploit the fixed sparse structure of DAC-MDPs along with parallel compute afforded by GPUs to gracefully scale our VI to millions of states. This can provide anywhere between 20-1000x wall clock speedup over its serial counterpart. To reduce the sensitivity to the smoothness parameter k, we use weighted averaging based on distances to nearest neighbors instead of uniform averaging in Equation 2. Furthermore, we use different smoothness parameter k and k_π to build the DAC-MDP and compute the policy respectively. We find that using larger values of k_π is generally beneficial to reduce the variance. Finally we use a simple rule-of-thumb of setting cost factor C to be in the order of magnitude of observed rewards.

3.3 Representation Learning

While DAC-MDPs can be defined over raw observations, for complicated observation spaces, such as images, performance will likely be poor in practice for simple distance functions. Thus, combining the DAC-MDP framework with deep representation learning is critical for observation-rich environments. Since the primary focus of this paper is on the introduction of DAC-MDPs, rather than representation learning, below we describe three basic representation-learning approaches used in our experiments. An important direction for future work is to investigate the effectiveness of other alternatives and to develop representation-learning specifically for the DAC-MDP framework.

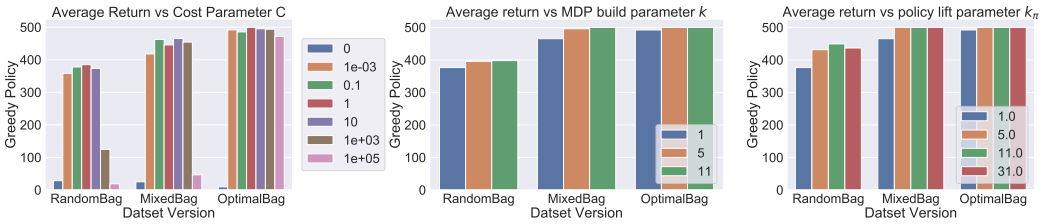

Figure 2: (a) Greedy Policy performance for CartPole with varying (a) cost parameter C. (b) smoothness parameter k. (c) policy smoothing parameter k_π

Random Projection (RND): This baseline simply produces a representation vector as the output of a chosen network architecture with randomly initialized weights (e.g. a CNN with random weights for images). **Offline DQN (DQN):** From the first term of Theorem 3.1 we see that decreasing the Lipschitz constant of $B[\tilde{Q}]$ can improve the performance bound. This suggests that a representation that is smooth with respect to the Q-function can be beneficial. For this purpose, our second representation directly uses the penultimate layer of DQN (Mnih et al., 2013) after it has been trained on our offline dataset $\mathcal{D}$. Since the penultimate layer is just a linear mapping away from the DQN Q-values, it is reasonable to believe that this representation will be smooth with respect to meaningful Q-functions for the environment. **BCQ - Imitation (BCQ):** The offline RL algorithm BCQ (Fujimoto et al., 2019) trains an imitation network to emulate the action distribution of the behavioral policy. This policy is used to help avoid backing up action values for under-represented actions. We use the penultimate layer of the imitation network trained on our dataset as the third type of representation. This representation is not as well-motivated by our performance bounds, but is intuitively appealing and provides a policy-centric contrast to Q-value modeling.

4 Experiments

The experiments are divided into three sections. First, we present exploratory experiments in Cartpole to illustrate the impact of DAC-MDP parameters using an idealized representation. Second, we demonstrate the scalability of the approach on Atari games with image-based observations. Finally, we demonstrate use-cases enabled by our planning-centric approach in a 3D first-person domain.

Exploring DAC-MDP Parameters: We explore the impact of the DAC-MDP parameters in Cartpole. To separate the impact of state-representation and DAC-MDP parameters, we use the ideal state representation consisting of the true 4-dimensional system state. We generate three datasets of size 100k each: (1) $RandomBag$: Trajectories collected by random policies. (2) $OptimalBag$: Trajectories collected by a well-trained DQN policy . (3) $MixedBag$: Trajectories collected by a mixed bag of ϵ-greedy DQN policies using $\epsilon \in [0, 0.1, 0.2, 0.4, 0.6, 1]$.

We first illustrate the impact of the cost parameter C by fixing $k = k_\pi = 1$ and varying C from 0 to 1e6. Figure 2a shows that for all datasets when there is no penalty for under-represented transitions, i.e. $C = 0$, the policy is extremely poor. This shows that even for this relatively simple problem with a relatively large dataset, the MDPs from the original cost-free averagers framework, can yield low-quality policies. At the other extreme, when C is very large, the optimal policy tries to stay as close as possible to transitions in the dataset. This results in good performance for the *OptimalBag* dataset, since all actions are near optimal. However, the policies resulting from the *MixedBag* and *RandomBag* datasets fail. This is due to those datasets containing a large number of sub-optimal actions, which the policy should actually avoid for purposes of maximizing reward. Between these extremes the performance is relatively insensitive to the choice of C.

Figure 2b explores the impact of varying k using $k_\pi = 1$ and $C = 1$. The main observation is that there is a slight disadvantage to using $k = 1$, which defines a deterministic finite MDP, compared to $k > 1$, especially for the non-optimal datasets. This indicates that the optimal planner is able to benefit by reasoning about the stochastic transitions of the DAC-MDP for $k > 1$. Otherwise the results are relatively insensitive to k. We set $k = 5$ for remaining experiments unless otherwise stated. Finally, Figure 2c varies the policy smoothing parameter k_π from 1 to 31. Similar to the results for varying k, there is a benefit to using a value of $k_\pi > 1$, but otherwise the sensitivity is low. Thus, even for this relatively benign problem setting, some amount of averaging at test time appears beneficial. We find that DAC-MDPs are slightly more sensitive to these parameters in low data regime. Similar experiments for low data regime are presented in Appendix **??**.

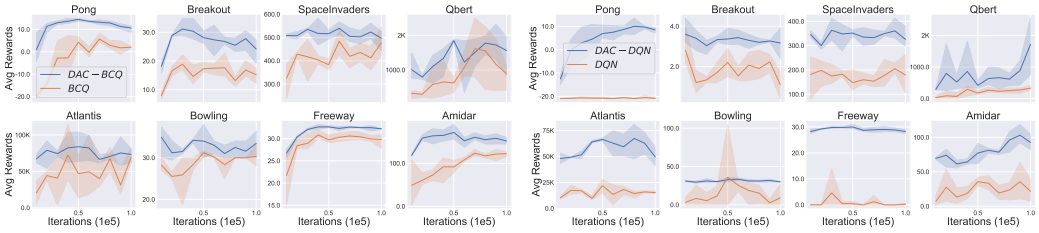

Figure 3: Results on Atari 100K (left)BCQ (right) DQN. Each agent is trained for 100K iterations, and evaluated on 10 episodes every 10K steps. At each of these evaluation checkpoints, we use the internal representation to compile DAC-MDPs. We then evaluate the DAC-MDPs for $N_e = 6$.

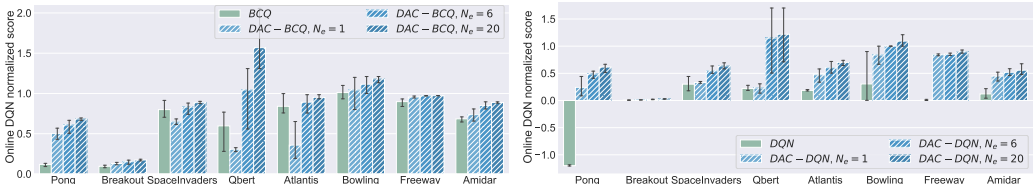

Figure 4: Results for different choices of candidate policies N_e on 100K dataset. Here we plot the final performance of (left) BCQ representation and (right) DQN representation along with the corresponding performance of DAC-MDPs for different values of N_e.

Experiments on Atari: We choose 8 Atari games ranging from small to medium action spaces: Pong, Breakout, Space Invaders, Q Bert, Atlantis, Bowling, Freeway, and Amidar. Following recent work (Fujimoto et al., 2019), we generated datasets by first training a DQN agent for each game. Next, to illustrate performance on small datasets and scalability to larger datasets, we generated two datasets of sizes 100K and 2.5 million for each game using an ϵ-greedy version of the DQN policy. In particular, each episode had a 0.8 probability of using $\epsilon = 0.2$ and $\epsilon = 0.001$ otherwise. This ensures a mixture of both near optimal trajectories and more explorative trajectories.

We first trained offline DQN and BCQ agents on the 100K dataset for 100K iterations and evaluated on 10 test episodes every 10K iterations. At each evaluation iteration, we construct DAC-MDPs using the latent state representation at that point. We consider an offline RL setting where N_e=6, i.e. we evaluate 6 policies (on 10 episodes each) corresponding to 6 different DAC-MDPs, set to a parameter combination of k=5, $C \in \{1, 100, 1M\}$, and $k_\pi \in \{11, 51\}$. For each representation and evaluation point, the best performing DAC-MDP is recorded. The entire 100K iteration protocol was repeated 3 times. Figure 3 shows the averaged curves with 90% confidence intervals.

The main observation is that for both BCQ and DQN representations the corresponding DAC-MDP performance (labeled DAC-BCQ and DAC-DQN) is usually better than BCQ or DQN policy at each point. Further, the DAC-MDP performance is usually better than the maximum performance of pure BCQ/DQN across the iterations. These results point to an interesting possibility of using DAC-MDPs to select the policy at the end of training at any fixed number of training iterations, rather than using online evaluations along the curve. Interestingly, the first point on each curve corresponds to a random network representation which is often non-trivial in many domains. However, we do see that in many domains the DAC-MDP performance further improves as the representation is learned, showing that the DAC-MDP framework is able to leverage improved representations.

Figures 4 investigates the performance at the final iteration for different values of N_e. For $N_e = 1$ we use $k = 5, k_\pi = 11, C = 1$ and for $N_e = 20$ we expand on the parameter set used for $N_e = 6$. First we see that in the majority of cases, even $N_e = 1$ is able to perform as well or better than BCQ or DQN and that increasing N_e often significantly improves performance. We do see that for three games, SpaceInvaders, QBert, and Atlantis, DAC-BCQ ($N_e = 1$) is significantly worse than the BCQ policy. This illustrates the potential challenge of hyperparameter selection in the offline framework without the benefit of additional online evaluations.

Finally we show results for all representation on the 2.5M dataset in Figure 5. In all but one case (Breakout DAC-DQN), we see that DAC-MDP improves performance or is no worse than the corresponding BCQ and DQN policies. Further in most cases, the DAC-MDP performance improves

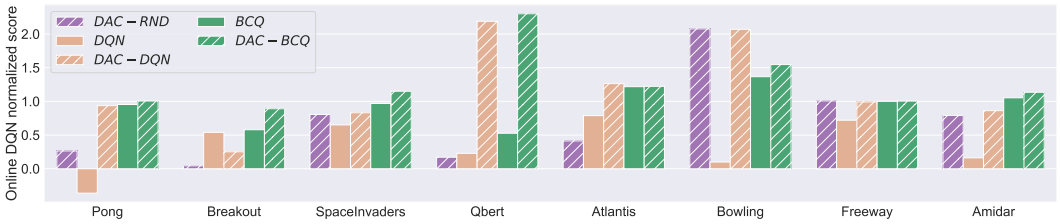

Figure 5: Atari results for 2.5M dataset. We show the final performance of BCQ and DQN trained for 2.5M iterations. We also use the same representation for the DAC-MDPs named as DAC-BCQ and DAC-DQN respectively. All DAC-MDPs are evaluated with $N_e = 6$.

over the Online DQN agent used to generate the dataset which was trained on 10M online transitions. While random representations perform reasonably in some cases, they are significantly outperformed by learned representations. These results show, for the first time, that optimal planning can result in non-trivial performance improvement across a diverse set of Atari games. Given the simplicity of the representation learning and the difference between random and learned representations, these results suggest significant promise for further benefits from improved representations.

Illustrative Use Cases in 3D Navigation: We now show the flexibility of an optimal planning approach using three use cases in Gym Mini-World (Chevalier-Boisvert, 2018), a first-person continuous 3D navigation environment. In all scenarios the agent has three actions: Move Forward, Turn Right, and Turn Left. Episodes lasted for a maximum of 100 time steps. The raw observations are 42x42 RGB images. We use random CNN representations in all experiments, which were effective for this environment. Our offline dataset was collected by following a random policy for 25K time steps. In these experiments we use $N_e = 2$ with $k = 5, k_\pi \in \{11, 51\}$, and $C = 0.01$. All the rooms used for the experiments and their optimal policies are visualized in the Appendix **??**.

Case 1: Adapting to Modified Action Space. We designed a simple room with a blue box in a fixed position and an agent that is spawned in a random initial position. The agent gets a -1 reward for bumping into the walls and a terminal reward of +1 for reaching the blue box. The DAC-MDP achieves an average total reward of 0.98 using the offline dataset. Next, we simulate the event that an actuator is damaged, so that the agent is unable to turn left. For most offline RL approaches this would require retraining the agent on a modified dataset from scratch. Rather our DAC-MDP approach can account for such a scenario by simply attaching a huge penalty to all left actions. The solution to this modified DAC-MDP achieves an average score of 0.96 and is substantially different than the original since it must find revolving paths with just the turn right action.

Case 2: Varying Horizons. We create a new room by adding a pillar in the simple room described above. Additionally, the agent gets a small non-terminal reward of 0.02 for bumping into the pillar. Depending on the placement of the agent and the expected lifetime of the agent it may be better to go for the single +1 reward or to repeatedly get the 0.02 reward. This expected lifetime can be simulated via the discount factor, which will result in different agent behavior. Typical offline RL approaches would again need to be retrained for each discount factor of interest. Rather, the DAC-MDP approach simply allows for changing the discount factor used by VI and solving for the new policy in seconds. Using a small discount factor (0.95) our short-term agent achieves an average score of 0.91 by immediately attempting to get to the box. However, the long-term agent with larger discount factor (0.995) achieves a score of 1.5 by repeatedly collecting rewards from the pillar. Qualitative observation suggests that both policies perform close to optimally.

Case 3: Robust/Safe Policies. Our third room adds a narrow passage to the box with cones on the side. Tunnel Room simulates the classic dilemma of whether to choose a risky or safe path. The agent can risk getting a -10 reward by bumping into the cones or can optionally follow a longer path around an obstacle to reach the goal. Even with a relatively large dataset, small errors in the model can lead an agent along the risky path to occasionally hit a cone. To build robustness, we find an optimal solution to a modified DAC-MDP where at each step there is a 10% chance of taking a random action. This safe policy avoids the riskier paths where a random "slip" might lead to disaster and always chooses the safer path around the obstacle. This achieved an average score of 0.72. In contrast, the optimal policy for the original DAC-MDP achieved an average score of 0.42 and usually selected the riskier path to achieve the goal sooner (encouraged by the discount factor).

5 Related Work

Averagers Framework. Our work is inspired by the original averagers framework of Gordon (1995) which showed the equivalence of a derived MDP to a type of approximate dynamic programming. A construction similar to the derived MDP was explored in later work (Jong & Stone, 2007) within an online RL setting. A model similar to Gordon's derived MDP was later theoretically studied for efficient exploration in online continuous-state RL (Pazis & Parr, 2013). Due to its exploration goals, that work introduced bonuses rather than penalties for unknown parts of the space. Our work is the first to integrate the averagers derived MDP with pessimistic costs in order to make it applicable to offline RL. Our work is also the first to demonstrate the framework in combination with deep representations for scaling to complex observation spaces such as images.

Pessimism for Offline Model-Based RL. Fonteneau et al. (2013) studied a "trajectory based" simulation model for offline policy evaluation. Similar in concept to our work, pessimistic costs were used based on transition distances to "piece together" disjoint observations, which allowed for theoretical lower-bounds on value estimates. That work, however, did not construct an MDP model that could be used for optimal planning. Most recently, in concurrent work to ours, pessimistic MDPs have been proposed for offline model-based RL (Kidambi et al., 2020; Yu et al., 2020). Both approaches define MDPs that penalizes for model uncertainty based on an assumed "uncertainty oracle" signal and derive performance bounds under the assumption of optimal planning. In practice, however, due to the difficulty of planning with learned deep-network models, the implementations rely on model-free RL, which introduces an extra level of approximation.

Optimal Planning with Deep Representations. Recent work (Corneil et al., 2018) uses variational autoencoders with Gumbel softmax discretization to learn discrete latent representations. These are used to form a tabular MDP followed by optimal planning in an online RL setting. More recently, van der Pol et al. (2020) introduced a contrastive representation-learning and model-learning approach that is used to construct and solve a tabular MDP. Experimental results in both of these works were limited to a small number of domains and small datasets. Unlike these works, which primarily focus on representation learning, our focus here has been on providing a theoretically-grounded model, which can be combined with representation-learning. Works like Kurutach et al. (2018) are mostly focused on learning plannable representations than planning. Instead, they train a GAN to generate a series of waypoints which can then be is leveraged by simple feedback controllers. Also, Yang et al. (2020) and Agarwal et al. (2020) attempt to incorporate long term relationships into their learned latent space and employ search algorithms such as A* and eliptical planners.

Episodic Control and Transition Graphs. Several prior approaches for online RL, sometimes called episodic control, construct different types of explicit transition graphs from data that are used to drive model-free RL (Blundell et al., 2016; Hansen, 2017; Pritzel et al., 2017; Lin et al., 2018; Zhu et al., 2020; Marklund et al., 2020). None of these methods, however, use solutions produced by planning as the final policy, but rather as a way of improving or bootstrapping the value estimates of model-free RL. The memory structures produced by these methods have a rough similarity to deterministic DAC-MDPs with $k = 1$.

6 Summary

This work is an effort to push the integration of deep representation learning with near-optimal planners. To this end, we proposed the Deep Averagers with Costs MDP (DAC-MDP) for offline RL as a principled way to leverage optimal planners for tabular MDPs. The key idea is to define a non-parametric continuous-state MDP based on the data, whose solution can be represented as a solution to a tabular MDP derived from the data. This construction can be integrated with any deep representation learning approach and addresses model inaccuracy by adding costs for exploiting under-represented parts of the model. Using a planner based on a GPU-implementation of value iteration, we demonstrate scalability to complex image-based environments such as Atari with relatively simple representations derived from offline model-free learners. We also illustrate potential use-cases of our planning-based approach for zero-shot adaptation to changes in the environment and optimization objectives. Overall, the DAC-MDP framework has demonstrated the potential for principled integrations of planning and representation learning. There are many interesting directions to explore further, including integrating new representation-learning approaches and exploiting higher-order structure for the next level of scalability.

REFERENCES

A. Agarwal, Sham M. Kakade, A. Krishnamurthy, and W. Sun. Flambe: Structural complexity and representation learning of low rank mdps. *ArXiv*, abs/2006.10814, 2020.

Christopher G Atkeson. Nonparametric model-based reinforcement learning. In *Advances in neural information processing systems*, pp. 1008–1014, 1998.

Charles Blundell, B. Uria, A. Pritzel, Y. Li, Avraham Ruderman, Joel Z. Leibo, Jack W. Rae, Daan Wierstra, and Demis Hassabis. Model-free episodic control. *ArXiv*, abs/1606.04460, 2016.

Craig Boutilier, Richard Dearden, and Moisés Goldszmidt. Stochastic dynamic programming with factored representations. *Artificial intelligence*, 121(1-2):49–107, 2000.

Maxime Chevalier-Boisvert. gym-miniworld environment for openai gym. `https://github.com/maximecb/gym-miniworld`, 2018.

Dane S. Corneil, W. Gerstner, and J. Brea. Efficient model-based deep reinforcement learning with variational state tabulation. *ArXiv*, abs/1802.04325, 2018.

R. Fonteneau, S. Murphy, L. Wehenkel, and D. Ernst. Batch mode reinforcement learning based on the synthesis of artificial trajectories. *Annals of Operations Research*, 208:383–416, 2013.

Scott Fujimoto, Edoardo Conti, Mohammad Ghavamzadeh, and Joelle Pineau. Benchmarking batch deep reinforcement learning algorithms. *ArXiv*, abs/1910.01708, 2019.

Geoorey J Gordon. Stable Function Approximation in Dynamic Programming. Technical report, 1995.

Caglar Gulcehre, Ziyu Wang, A. Novikov, T. L. Paine, Sergio Gomez Colmenarejo, Konrad Zolna, Rishabh Agarwal, Josh Merel, Daniel J. Mankowitz, Cosmin Paduraru, Gabriel Dulac-Arnold, J. Li, Mohammad Norouzi, Matt Hoffman, Ofir Nachum, G. Tucker, Nicolas Heess, and N. D. Freitas. Rl unplugged: Benchmarks for offline reinforcement learning. *ArXiv*, abs/2006.13888, 2020.

Steven Stenberg Hansen. Deep episodic value iteration for model-based meta-reinforcement learning. *ArXiv*, abs/1705.03562, 2017.

Nicholas K Jong and Peter Stone. Model-based function approximation in reinforcement learning. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, pp. 1–8, 2007.

R. Kidambi, A. Rajeswaran, Praneeth Netrapalli, and T. Joachims. Morel : Model-based offline reinforcement learning. *ArXiv*, abs/2005.05951, 2020.

Thanard Kurutach, A. Tamar, Ge Yang, S. Russell, and P. Abbeel. Learning plannable representations with causal infogan. In *NeurIPS*, 2018.

Zichuan Lin, Tianqi Zhao, G. Yang, and L. Zhang. Episodic memory deep q-networks. *ArXiv*, abs/1805.07603, 2018.

Henrik Marklund, Suraj Nair, and Chelsea Finn. Exact (then approximate) dynamic programming for deep reinforcement learning. 2020.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *ArXiv*, abs/1312.5602, 2013.

Jason Pazis and Ronald Parr. Pac optimal exploration in continuous space markov decision processes. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

A. Pritzel, B. Uria, S. Srinivasan, Adrià Puigdomènech Badia, Oriol Vinyals, Demis Hassabis, Daan Wierstra, and Charles Blundell. Neural episodic control. In *ICML*, 2017.

Martin L. Puterman. Markov decision processes: Discrete stochastic dynamic programming. In *Wiley Series in Probability and Statistics*, 1994.

Aswin Raghavan, Saket Joshi, Alan Fern, Prasad Tadepalli, and Roni Khardon. Planning in factored action spaces with symbolic dynamic programming. In *AAAI*. Citeseer, 2012.

Elise van der Pol, Thomas Kipf, Frans A. Oliehoek, and M. Welling. Plannable approximations to mdp homomorphisms: Equivariance under actions. In *AAMAS*, 2020.

G. Yang, A. Zhang, Ari S. Morcos, Joelle Pineau, P. Abbeel, and R. Calandra. Plan2vec: Unsupervised representation learning by latent plans. *ArXiv*, abs/2005.03648, 2020.

Tianhe Yu, G. Thomas, Lantao Yu, S. Ermon, J. Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *ArXiv*, abs/2005.13239, 2020.

Guangxiang Zhu, Zichuan Lin, G. Yang, and C. Zhang. Episodic reinforcement learning with associative memory. In *ICLR*, 2020.