SimuDICE: Offline Policy Optimization Through World Model Updates and DICE Estimation

Catalin Brița[1], Stephan Bongers[1], and Frans A. Oliehoek[1]

EEMCS Faculty, Delft University of Technology, Delft, Netherlands
{c.e.brita,s.r.bongers,f.a.oliehoek}@tudelft.nl

Abstract. In offline reinforcement learning, deriving an effective policy from a pre-collected set of experiences is challenging due to the distribution mismatch between the *target policy* and the *behavioral policy* used to collect the data, as well as the limited sample size. Model-based reinforcement learning improves sample efficiency by generating simulated experiences using a learned dynamic model of the environment. However, these synthetic experiences often suffer from the same distribution mismatch. To address these challenges, we introduce SimuDICE, a framework that iteratively refines the initial policy derived from offline data using synthetically generated experiences from the world model. SimuDICE enhances the quality of these simulated experiences by adjusting the sampling probabilities of state-action pairs based on stationary DIstribution Correction Estimation (DICE) and the estimated confidence in the model's predictions. This approach guides policy improvement by balancing experiences similar to those frequently encountered with ones that have a distribution mismatch. Our experiments show that SimuDICE achieves performance comparable to existing algorithms while requiring fewer pre-collected experiences and planning steps, and it remains robust across varying data collection policies.

Keywords: Offline Reinforcement Learning · Model-Based Reinforcement Learning · DIstribution Correction Estimations (DICE).

1 Introduction

Reinforcement Learning (RL) [34] has demonstrated numerous successes in domains such as games [10] and robotics [36], largely due to simulation-based trial and error [26,31]. While feasible in game environments (where the game can be considered a simulator) or in simple real-world scenarios that can be accurately simulated, such direct or easy access to the environment is often not possible. Furthermore, in some areas such as medicine [27] the deployment of a new policy, even just for the sake of performance evaluation, may be risky or costly.

Offline RL [23], also known as batch RL [22], addresses the challenge of training agents from static, pre-collected datasets, which are typically composed of off-policy data [34]. A key issue in this context is the mismatch between the state visitation distribution of the *target (candidate) policy* and the *behavioral* *(logging) policy*. The problem is exacerbated throughout the learning process as this mismatch grows, potentially leading to issues such as divergence in off-policy learning [2,38], making direct online-to-offline transitions challenging [8,19].

The agent has to perform *offline policy evaluation* during the learning process, a task that is particularly challenging due to the policy-induced state-action distribution mismatch. Early work on this problem tackled the challenge using products of importance sampling [29]. Subsequent approaches have focused on improving the variance by directly learning density ratios [12,25]. The DIstribution Correction Estimation (DICE) family of algorithms [28,40,43,44] achieves impressive results by leveraging optimization techniques to directly estimate stationary distribution corrections, significantly reducing variance.

Most prior work in offline RL consists of model-free methods. These studies show that directly using off-policy RL algorithms yields poor results due to distribution mismatch and function approximation errors. To address this, various modifications were proposed, such as Q-network ensembles [8,39], regularization towards the behavioral policy [15,18,39], and implicit Q-learning [17].

Model-based RL (MBRL) involves learning a dynamics model of the environment that can be used for policy improvement. MBRL has shown significant success in online learning, demonstrating excellent sample efficiency [9,10,11]. However, applying MBRL algorithms directly to offline datasets presents challenges due to the same distribution mismatch issue. Specifically, it is difficult to obtain a globally accurate model because the dataset may not cover the entire state-action space. Consequently, planning with a learned model without safeguards against model inaccuracy can lead to ‘hallucinated‘ states [13] and ‘model exploitation‘ [4,14,20], leading to the possibility of poor policy performance.

Most of the prior work in offline MBRL [3,35] pre-train a one-step forward model via maximum likelihood estimation to be a simple mimic of the world and then uses it to improve the policy, without any change to the model. This results in an *objective mismatch*, namely the objective function used for model training (accurately predicting the environment) is unrelated to its utilization (policy optimization). Recent works have identified *objective mismatch* in the model training and utilization as problematic [7,21].

We introduce *SimuDICE*[1], an algorithm that iteratively improves the target policy by adjusting the sampling probabilities within a world model. Unlike prior methods that focus solely on generating safe experiences, we extend that approach by integrating DICE estimations, usually used for offline policy evaluation. These estimations reveal how reward distributions shift from the behavioral dataset to the target policy, which leads us to choose to explore those transitions further. Our experiments show that incorporating the DICE estimations with a model prediction confidence safeguard achieves superior results with less data and fewer planning steps compared to uniform sampling experiences.

[1] https://github.com/Catalin-2002/SimuDICE

2 Related Work

Offline Reinforcement Learning (RL) In offline RL, agents are trained only from pre-collected datasets, avoiding the risks associated with real-time data collection. One of the main issues in offline RL is the distribution mismatch between the *behavioral (logging) policy* and the *target policy* that is being optimized.

Fujimoto et al. [8] tackled this by stabilizing Q-learning to reduce bootstrapping errors caused by this mismatch, specifically through the introduction of a method to limit the overestimation of Q-values. Building on this, Wu et al. [39] proposed a conservative Q-learning approach known as Behavior-Regularized Actor-Critic (BRAC), which further mitigates Q-value overestimation by incorporating a penalty term that keeps the learned policy close to the behavior policy. Levine et al. [24] emphasize the broader challenges in offline RL, highlighting the necessity of techniques that effectively manage limited and biased datasets. Most prior work in this area focuses only on model-free approaches, exploring algorithms such as the Q-network ensembles [8,39], behavioral policy regularization [15,18,39], and implicit Q-learning [17].

Distribution mismatch correction techniques Various approaches have been developed to mitigate the policy-induced state-action distribution mismatch. Precup et al. [29] tackle the problem using products of importance sampling ratios, though this approach suffers from large variance. To correct the distribution mismatch without incurring a large variance, Hallak et al. [12] and Liu et al. [25] propose learning the density ratio between the state distribution of the target policy and sampling distribution directly.

DualDICE [28] is a relaxation of previous methods and enables learning from multiple unknown behavior policies. They achieved impressive results by doing a change of variable technique on the density ratios calculation. GenDICE [43] is a generalization of DualDICE, stabilizing estimations in the average reward setting. GradientDICE [44] outlines that GenDICE is not a convex-concave saddle-point problem in all settings and proposes a provably convergent method under linear function approximation. Despite their differences, all these algorithms use minimax optimizations, allowing them to be combined under the regularized Lagrangians of the same linear problem [40].

Model-based offline RL Model-based RL (MBRL) uses a learned model of the environment to generate additional experiences, thereby enhancing sample efficiency. The Dyna-Q algorithm [33] was among the first to integrate model-based techniques with reinforcement learning, combining planning and learning into a unified framework. Over time, MBRL has become a key approach in online reinforcement learning, with several algorithms demonstrating exceptional sample efficiency by incorporating sophisticated dynamics models that closely simulate the environment [10,11].

However, in offline settings where interactions with the environment are restricted, inaccuracies in the learned models can lead to the generation of unrealistic or 'hallucinated' states within the synthetic experiences. To mitigate this issue, recent methods such as MOPO [42] and MOReL [16] incorporate uncertainty estimates into the model's predictions. This approach helps guide the policy towards safer and more reliable regions of the state space, reducing the risk of overfitting to unreliable synthetic experiences.

In this work, we extend pessimistic offline MBRL approaches by incorporating DICE estimations with a prediction confidence estimation. This combination aims to allow the algorithm to safely explore regions with estimated divergence between the offline dataset and the candidate policy, improving robustness. Although we explore a simple setting with a Tabular World Model in this work, we believe this research direction has the potential to mitigate overfitting to unreliable synthetic data when applied with more complex world models.

3 Preliminaries

In this section, we introduce the theoretical framework and describe the problem setting. We also present the DIstribution Correction Estimation (DICE) algorithm, which is the basis for updating sampling probabilities in SimuDICE.

3.1 Theoretical framework

We consider a Markov Decision Process (MDP) [30], in which the environment is defined by a tuple $\mathcal{M} = \langle S, A, R, T, \mu_0, \gamma \rangle$ where S represents the state space, A is the action space, R is a reward function, T is the transition probability function, μ_0 is the initial state distribution, and $\gamma \in [0, 1)$ is the discount factor.

A policy π in an MDP decides what action the agent should take given some state s. Formally, it is a mapping $\pi : S \to \Delta(A)$, where $\pi(s)$ represents the probability distribution over actions A in state s. The goal of the agent is to maximize the cumulative expected reward (its return), given by Eq. (1).

$$\rho(\pi) = \mathbb{E}_{s_0 \sim \mu_0} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid a_t \sim \pi(\cdot \mid s_t) \right] \tag{1}$$

To evaluate the performance of a policy, we define two functions: the Value function $V^\pi(s)$, which is the expected return of policy π from state s (Eq. 2), and the Q-value function $Q^\pi(s, a)$, which represents the expected return following a policy π starting from state s with action a (Eq. 3).

$$V^\pi(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s \right] \tag{2}$$

$$Q^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \mid s_0 = s, a_0 = a \right] \tag{3}$$

Bellman equation The Bellman equation provides a recursive definition of the Q-value by decomposing it into immediate reward and the discounted value of the next state-action pair. Eq. (4) shows how the Bellman equation is applied to the Q-value policy function $Q^{\pi}(s,a)$.

$$Q^{\pi}(s,a) = R(s,a) + \gamma \mathbb{E}_{s' \sim T(s,a), a' \sim \pi(\cdot|s')} \left[Q^{\pi}(s',a') \right] \tag{4}$$

The Bellman operator $\mathcal{B}^{\pi}$ iteratively applies the Bellman equation to update the Q-values until convergence, leading to a formulation as in Eq. (5).

$$\mathcal{B}^{\pi} Q(s,a) = R(s,a) + \gamma \mathbb{E}_{s' \sim T(s,a), a' \sim \pi(\cdot|s')} [Q(s',a')] \tag{5}$$

3.2 Problem setting

Offline Reinforcement Learning The focus of this work is offline RL. Unlike online RL where the agent actively interacts with the environment to gather data and update its policy, offline RL aims to derive the optimal policy π from a pre-collected dataset of experiences. Specifically, we assume access to a finite dataset $\mathcal{D} = \left\{ \left(s_0^{(i)}, s^{(i)}, a^{(i)}, r^{(i)}, s'^{(i)} \right) \right\}_{i=1}^{N}$, where $s_0 \sim \mu_0$, $(s^{(i)}, a^{(i)}) \sim d^{\mathcal{D}}$ are samples from an unknown distribution $d^{\mathcal{D}}$, $r^{(i)} \sim R(s^{(i)}, a^{(i)})$, and $s'^{(i)} \sim T(s^{(i)}, a^{(i)})$.

Model-Based RL MBRL involves learning an MDP $\hat{\mathcal{M}} = \langle S, A, \hat{R}, \hat{T}, \hat{\mu_0}, \gamma \rangle$ which uses the learned transitions $\hat{T}$ instead of the true transitions T, and the learned reward function $\hat{R}$ instead of the true reward function R. In this work, we assume the initial distribution μ_0 is unknown and learned from the data.

3.3 DualDICE estimation

In this section, we will elaborate on the DICE estimation algorithm used in SimuDICE, namely DualDICE [28], which is often used for *off-policy evaluation*. They obtained impressive results by reducing the *off-policy evaluation* problem to density ratio estimation and doing a change of variable optimization trick. The policy value can be rewritten using the importance weighing trick (Eq. 6).

$$\rho(\pi) = \mathbb{E}_{(s,a) \sim d^{\pi}} [r(s,a)] = \mathbb{E}_{(s,a) \sim d^{\mathcal{D}}} \left[\frac{d^{\pi}(s,a)}{d^{\mathcal{D}}(s,a)} r(s,a) \right], \tag{6}$$

where d^{π} is the discounted state visitation distribution (Eq. 7).

$$d^{\pi}(s,a) := (1-\gamma) \sum_{t=0}^{\infty} \gamma^t \cdot \Pr[s_t = s, a_t = a \mid s_0 \sim \mu_0, \pi] \tag{7}$$

Eq.(6) can be rewritten in the offline setting as a weighted average (Eq. 8), reducing the problem to estimating the density ratios (Eq. 9) for policy correction.

$$\mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[\frac{d^{\pi}(s,a)}{d^{\mathcal{D}}(s,a)}r(s,a)\right] = \frac{1}{N}\sum_{i=1}^{N}\frac{d^{\pi}(s^{(i)},a^{(i)})}{d^{\mathcal{D}}(s^{(i)},a^{(i)})}r^{(i)} \tag{8}$$

$$w_{\pi/\mathcal{D}}(s,a) := \frac{d^{\pi}(s,a)}{d^{\mathcal{D}}(s,a)} \tag{9}$$

DualDICE [28] optimizes a (bounded) function [2]: $\nu : S \times A \to \mathbb{R}$ as in Eq.(10).

$$\min_{\nu:S\times A\to\mathbb{R}} J(\nu) := \frac{1}{2}\mathbb{E}_{(s,a)\sim d^{\mathcal{D}}}\left[(\nu - \mathcal{B}_0^{\pi}\nu)(s,a)^2\right] - (1-\gamma)\mathbb{E}_{s_0\sim\mu,a_0\sim\pi(s_0)}\left[\nu(s_0,a_0)\right] \tag{10}$$

$\mathcal{B}_0^{\pi}$ is used to denote the expected Bellman operator with respect to policy π and zero-reward: $\mathcal{B}_0^{\pi}\nu(s,a) := \gamma\mathbb{E}_{s'\sim T(s,a),a'\sim\pi(s')}[\nu(s',a')]$. The authors of DualDICE [28] state that the first term alone leads to a trivial solution $\nu \equiv 0$, which is avoided by the second term that ensures $\nu^* > 0$. They prove that the Bellman residuals of ν^* are exactly the desired distribution corrections (Eq. 11).

$$w_{\pi/\mathcal{D}}(s,a) = (\nu^* - \mathcal{B}_0^{\pi}\nu^*)(s,a). \tag{11}$$

4 SimuDICE

For clarity and ease of understanding, we start by presenting an idealized version of SimuDICE, discussing its theoretical foundations. We then proceed to describe the practical implementation used in our experiments. Algorithm 1 presents the broad framework, while Figure 1 shows an illustration of the algorithm.

Algorithm 1 SimuDICE: Offline Policy Optimization

```
Require: Dataset $\mathcal{D}$
1: Learn approximate dynamics model $\hat{T} : S \times A \to S$ using $\mathcal{D}$.
2: Initialize estimated model confidence $\mathcal{C}$ using $\mathcal{D}$.
3: Learn an initial target policy $\pi_{\text{target}}$ using $\mathcal{D}$.
4: for iteration = 1 to number_iterations do
5:     $w_{\pi/\mathcal{D}} \leftarrow$ DualDICE($\mathcal{D}, \pi_{\text{target}}$)
6:     $\mathcal{P} \leftarrow$ updateProbabilities($\mathcal{C}, w_{\pi/\mathcal{D}}$)
7:     $\pi_{\text{target}} \leftarrow$ planner($\hat{T}, \mathcal{P}, \pi_{\text{target}}$)
8: end for
9: return $\pi_{\text{target}}$
```

[2] Note that ν is a state-action value function, analogous to Q-values, although Q-values were not utilized in this context for completeness and to maintain generality.

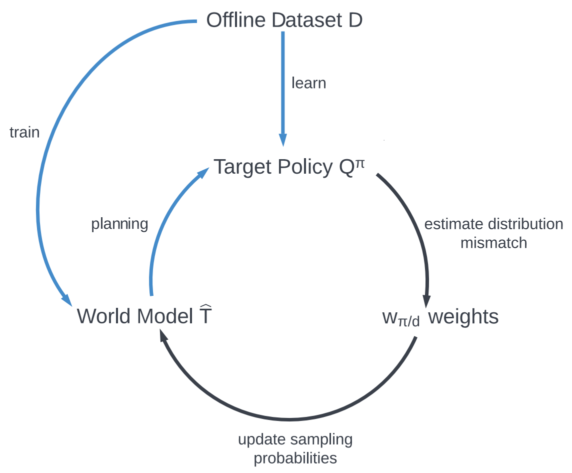

Fig. 1. The components of SimuDICE and their interactions. Transitions adapted from Dyna-Q [33] are in blue, while those unique to SimuDICE are depicted in black.

Learning the world model The first step involves using the offline dataset to learn an approximate dynamics model $\hat{T}(\cdot \mid s, a)$. Usually in literature, this is achieved through maximum likelihood estimator [5] or other techniques like dynamics modeling [10,11] or diffusion [1,6].

In this work, we use for simplicity a Tabular World Model [32]. This memory-based model replicates previously observed experiences by averaging the rewards for each transition and selecting the most frequently observed next state (Algorithm 2). This approach allows the model to handle stochastic environments.

Algorithm 2 Learning the Tabular World Model

```
1: Initialize an empty tabular model.
2: for all state-action pairs $(s, a)$ in the dataset do
3:     Update the model with the observed next state $s'$ and reward $r$.
4: end for
5: for all state-action pairs $(s, a)$ in the model do
6:     Compute the average reward for $(s, a)$.
7:     Determine the most frequently observed next state $s'$.
8: end for
9: Return the learned tabular model.
```

Learning the initial policy In this stage, we use the offline pre-collected dataset to derive an initial policy, using Experience Replay [26], which makes the policy more stable. We apply the Q-learning formula as in Eq. (12).

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right], \tag{12}$$

Updating the Sampling Probabilities The core of SimuDICE consists of updating the sampling probabilities, a process that involves two steps. Firstly, we calculate the stationary distribution correction, $w_{\pi/\mathcal{D}}$, using DualDICE [28].

The second step involves adding a safeguard to guide the model toward 'safe experiences.' This safeguard can take various forms, such as incorporating pessimism or other risk-averse strategies. In this work, because our world model cannot predict new experiences, for completness, we consider the confidence of a prediction, $\mathcal{C}(s, a)$, as the normalized frequency of occurrences in the dataset.

Let $\mathcal{P}(s, a)$ be the probability that the world model will sample state s and action a. We consider $\mathcal{P}(s, a)$ as the normalized sum of the confidence $\mathcal{C}(s, a)$ and the regularized softmax of the $w_{\pi/\mathcal{D}}$ weights. The regularization term λ is introduced to align the scale of confidence predictions with DICE estimations, preventing mode collapse. Additionally, in environments with large state-action spaces, the values of $w_{\pi/\mathcal{D}}$ become too small, leading to floating-point errors. Below we show the derivation of the likelihood function $\mathcal{P}(s, a)$. First, we define the likelihood $\mathcal{L}(s, a)$ (Eq. 13) of sampling the state-action pair (s, a).

$$\mathcal{L}(s,a) = \mathcal{C}(s,a) + \frac{e^{w_{\pi/\mathcal{D}}(s,a)\cdot\lambda}}{\sum_{(s',a')} e^{w_{\pi/\mathcal{D}}(s',a')\cdot\lambda}} / \lambda \tag{13}$$

The final probability function is obtained by normalizing $\mathcal{L}(s, a)$ (Eq. 14).

$$\mathcal{P}(s,a) = \frac{\mathcal{L}(s,a)}{\sum_{(s',a')} \mathcal{L}(s',a')} \tag{14}$$

Planning The planning phase of the algorithm improves the policy using synthetically generated data, enabling the agent to learn from a more diverse range of experiences. The policy is updated using the same method as experiences from the offline dataset. In SimuDICE, experiences (s, a) are sampled using the World model $\hat{T}$ with different probabilities, calculated based on how likely is the experience to be encountered by running the policy and how confident is the world model on the prediction. The world model predicts the subsequent reward and next state (r, s') and updates the Q-values using Eq. 12. A pseudocode of how planning works can be seen in Algorithm 3.

Algorithm 3 Planner

```
Require: Dynamics model $\hat{T}$, Probability distribution $\mathcal{P}$, Current policy $\pi$
1: for iteration = 1 to planning_iterations do
2:     Sample state-action pair $(s, a)$ using $\mathcal{P}$
3:     $(s', r) \leftarrow \hat{\mathcal{T}}(s, a)$
4:     Update policy $\pi$ using the observed transition $(s, a, s', r)$
5: end for
6: return Updated policy $\pi$
```

5 Experiments

Our experiments aim to answer the following questions: 1) how do the quality (value) and size (number of experiences collected) of the behavior policy impact the policy learned by SimuDICE; 2) how does the policy derived by SimuDICE compare with other algorithms across different off-policy datasets; and 3) what effect do various components and parameters have on SimuDICE's performance?

To answer the above questions, we consider commonly studied benchmark tasks from the Gymnasium Library [37]. Our experimental setup closely follows previous work [8,19,39]. We focus on *Toy Text* environments: *Taxi*, *FrozenLake*, and *CliffWalking*, as illustrated in Figure 2. For each environment, we consider three distinct logged datasets. These datasets are collected following the approach of Wu et al. [39], each dataset containing the equivalent of 500 timesteps of environment interaction. First, we partially train a policy (π_p) to achieve values of approximately -2.36, -2.55, and 0.04 for the three environments, respectively. We then generate three noisy variants of π_p using epsilon-greedy strategies (with ϵ values of 0.1, 0.4, and 0.7), introducing different magnitudes of noise to π_p. We also added a penalty of 10 when the agent fails to complete the task.

We evaluate the learned policies by performing rollouts in the (real) environment, strictly for evaluation purposes. These rollouts are not available to the algorithm and are not used for any learning. This evaluation protocol is consistent with prior work [8,19,39]. We report results as average per-step reward over 500 plays and 5 seeds, using identical hyperparameters across all environments.

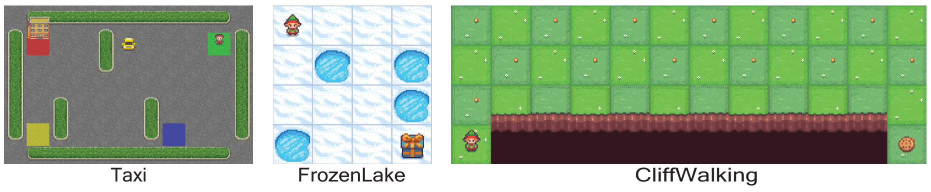

Fig. 2. Illustration of the suite of tasks considered in this work. These tasks require the RL agent to learn to navigate grid environments to accomplish certain tasks.

5.1 Algorithm comparisons

We compare SimuDICE with two other methods: offline Q-learning and a variant of SimuDICE that uses uniform sampling probabilities, which we refer to as offline Dyna-Q. To evaluate their effectiveness across different planning scenarios, we assess both SimuDICE and offline Dyna-Q using 10 and 20 planning steps.

Figure 3 shows that SimuDICE consistently outperforms or matches the performance of the other algorithms across various settings. In particular, in the

Taxi environment, which is considered more challenging due to its larger state-action space and the diversity of actions and penalties, SimuDICE significantly outperforms the other algorithms. Even with only 10 planning steps, SimuDICE's performance often surpasses the performance of the 20-step offline Dyna-Q variant, with this difference being more pronounced in environments where data was collected with lower epsilon-greedy values.

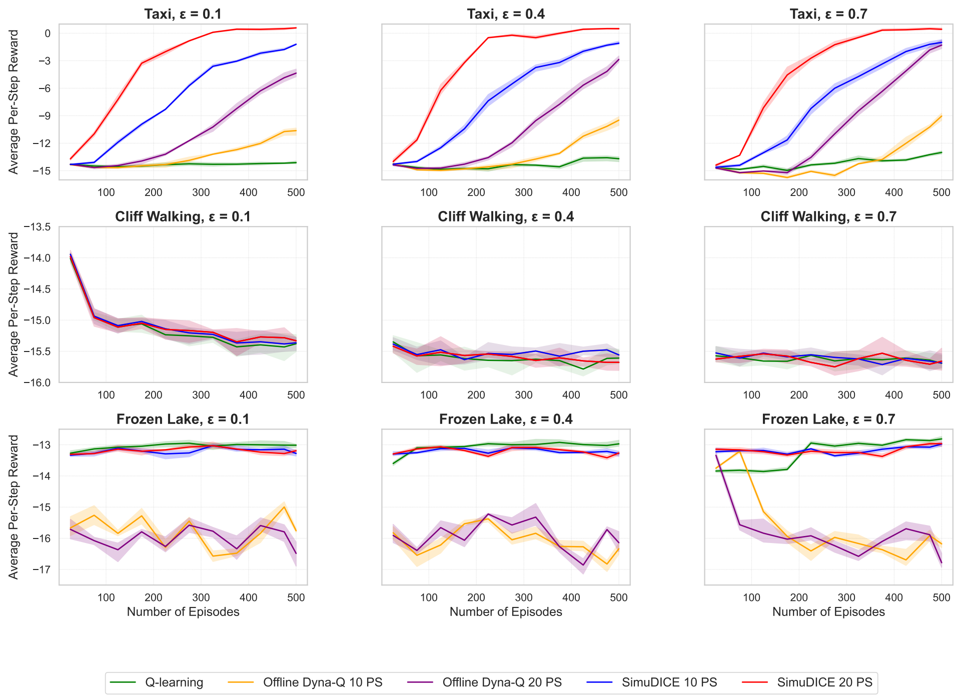

Fig. 3. Comparison of algorithm performance in discrete tabular environments: *Taxi*, *CliffWalking*, and *FrozenLake*, under varying epsilon-greedy data collection policies ($\epsilon = 0.1, 0.4, 0.7$). Each plot shows the average per-step reward as a function of the number of trajectories in the offline data. The results are averaged over 500 steps and 5 different random seeds. The shaded regions represent the variance across the different seeds. PS represents the number of planning steps.

In the CliffWalking environment, the performance of algorithms shows no significant difference, with all being within the same variance range.

In the FrozenLake environment, both offline Q-learning and SimuDICE perform similarly, achieving comparable average per-step rewards with no noticeable differences. The key observation is that the offline Dyna-Q method consistently underperforms compared to the others, regardless of the offline data quality.

5.2 Ablation study

In this section, we conduct an ablation study to evaluate the effect of various parameters on the performance of SimuDICE. Our analysis focuses only on the *Taxi* environment, as it represents the most complex scenario out of the three.

Planning Steps: *How does the number of planning steps affect the model's performance?*

We carry out an experimental evaluation to determine how different numbers of planning steps affect the agent's performance. Figure 4 shows that while the number of planning steps improves the performance, the relationship is not linear. The improvement is particularly evident in low-data cases, with the difference starting to decrease with the number of experiences in the offline dataset.

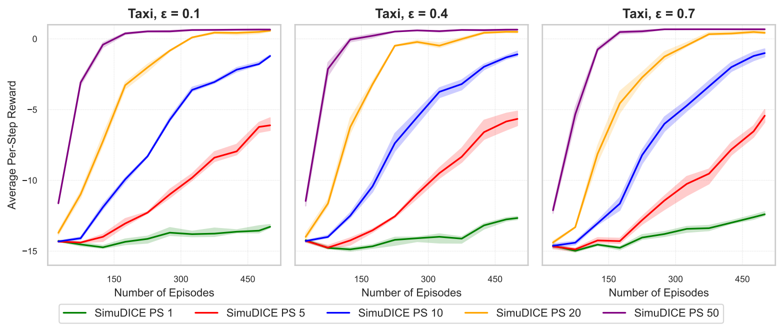

Fig. 4. Impact of the number of planning steps on the average per-step reward under different epsilon-greedy data collection policies, with varying epsilon values.

Different Sampling Probabilities Formulas: *How does the algorithm's performance change when we alter the method for estimating sampling probabilities?*

We compare the sampling probability formula from SimuDICE with two alternatives, each using a different method to transform model confidence and $w_{\pi/\mathcal{D}}$ into sampling probabilities. This comparison evaluates how effectively these variants guide the world model in sampling 'valuable' synthetic experiences. To ensure normalization, each likelihood is converted into a probability by dividing by the sum of all likelihoods, making them sum to 1. Formula 1 is the default function in SimuDICE (Eq. 15). Formula 2 (Eq. 16) depends solely on model confidence, while Formula 3 applies lambda-regularized DICE estimations $w_{\pi/\mathcal{D}}$ over a uniform sampling model, disregarding model confidence (Eq. 17).

$$\mathcal{L}_1(s,a) = \mathcal{C}(s,a) + \frac{e^{w_{\pi/\mathcal{D}}(s,a)\cdot\lambda}}{\lambda\sum_{(s',a')} e^{w_{\pi/\mathcal{D}}(s',a')\cdot\lambda}} \tag{15}$$

$$\mathcal{L}_2(s,a) = \mathcal{C}(s,a) \tag{16}$$

$$\mathcal{L}_3(s,a) = \frac{1}{S\cdot A} + \frac{e^{w_{\pi/\mathcal{D}}(s,a)\cdot\lambda}}{\lambda\sum_{(s',a')} e^{w_{\pi/\mathcal{D}}(s',a')\cdot\lambda}} \tag{17}$$

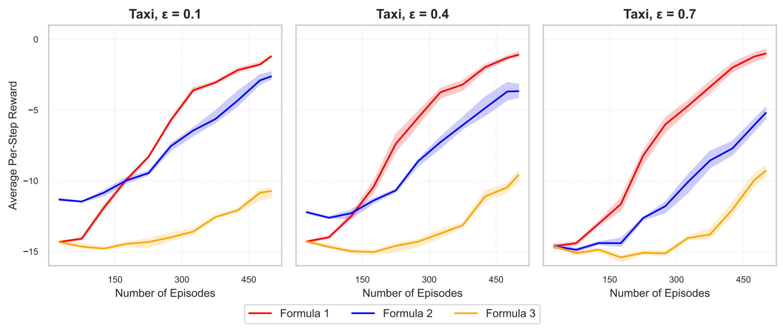

Fig. 5. Comparison of average per-step rewards achieved by SimuDICE using different sampling formulas, across different epsilon-greedy offline dataset collection policies.

Figure 5 shows that the formula used in SimuDICE outperforms others under varying data qualities. However, when the *target policy* is close to the *behavioral policy* used for data collection, alternative sampling methods may outperform it. Specifically, the SimuDICE formula excels in scenarios with diverse data but yields inferior results when the data lacks diversity and is already close to the desired distribution).

Number of iterations: *How does the number of iterations (the number of updates to the sampling probabilities) change the performance of the algorithm?*
To verify the effectiveness of the number of iterations (i.e., the frequency of updating the sampling probabilities) on the performance of the agent, we conducted a comparative analysis using the SimuDICE with 10 planning steps.

Figure 6 shows that varying the number of iterations has a negligible effect on the performance of SimuDICE in the Taxi environment.

6 Discussion

This work introduced SimuDICE, a framework for policy optimization in offline reinforcement learning, and assessed its effectiveness across various scenarios. SimuDICE addresses the *data needs* and the *state-action distribution mismatch*. In this section, we discuss the findings, their implications, and their limitations.

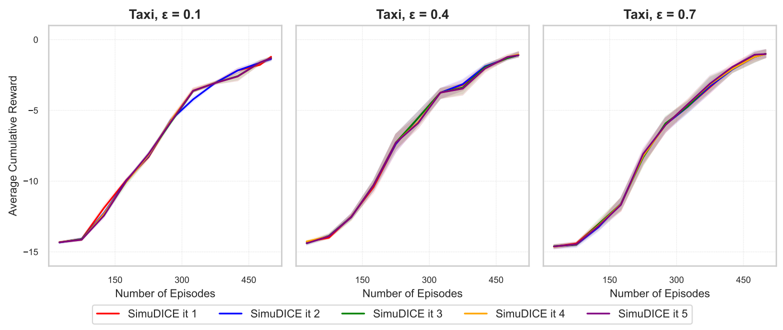

Fig. 6. Effect of iteration number on the average per-step reward achieved by SimuDICE across different epsilon-greedy offline dataset collection policies.

Key Findings and Implications Based on the experimental results, we derive the following key findings and implications of the proposed method:

– **Improved Sample Efficiency:** Our experiments show that SimuDICE achieves greater sample efficiency compared to uniform sampling experiences during the planning stage and vanilla Q-learning. This advantage is particularly evident in data-rich environments where the collected data diverges significantly from the optimal policy. We attribute this improvement to the adjustment of sampling probabilities using DICE estimations. In environments where the behavioral policy closely aligns with the target policy, most experiences can be considered ‘optimal,‘ which can cause problems with this method. However, in environments with more diverse data, where some experiences are more useful than others, SimuDICE outperforms the aforementioned methods, achieving a better policy in most cases.

– **Distribution Mismatch Correction:** Our results show that in addition to SimuDICE being more sample-efficient than the methods we compared it with, it also requires fewer planning steps to achieve comparable or superior policies. This suggests that we can achieve equal or better policies while generating fewer synthetic experiences using the world model, which implicitly reduces the risk of ‘hallucination‘. While the DICE estimations play an important role in this improvement, it is worth mentioning that the confidence prediction estimation was added for completeness as a safeguard to balance exploitation-exploration for these experiences.

– **Objective Mismatch:** Our ablation study reveals no significant performance gain from updating the sampling probabilities multiple times during a run. We believe this may be due to the simplicity of the environments used in our experiments, as the approach was originally theorized for more complex settings [7,21]. Additionally, the basic nature of the components, such as the world model, may have influenced these results. We recommend further research to determine if multiple iterations of updating sampling probabilities improve outcomes and to investigate whether similar results to those achieved by the AMPL algorithm [41] can be obtained.

Limitations In this part, we outline the limitations of this study and indicate some possible future work directions.

– **Simple environments:** While SimuDICE showed promising results in deterministic grid-world environments, their simplicity limits the generalizability of the findings. Therefore, future research should focus on evaluating SimuDICE in more complex settings to assess its effectiveness better.
– **Sensitivity to sample probabilities formula:** In the ablation study, we show that the performance of SimuDICE is affected by different variations of the sampling probability formula. Given the limited number of scenarios the algorithm was tested on, a more extensive evaluation across a wider range of environments is necessary to assess its effectiveness.
– **Comparison with other algorithms:** This study conducted a comparison between SimuDICE, vanilla offline Q-learning, and offline Dyna-Q [33], where the latter is effectively SimuDICE with uniform sampling probabilities. While this comparison gives an intuition of its improvements, a comparison with other algorithms might provide more insights into its performance.

7 Conclusion

We have presented SimuDICE, a framework for optimizing policies in model-based offline reinforcement learning by adjusting the world model's sampling probabilities using DualDICE estimation and the estimated prediction confidence. The main innovation of SimuDICE lies in its ability to correct the state-action distribution mismatch between the *behavior policy* and *target policy* through a bi-objective optimization that balances experience realism and diversity, thereby preventing mode collapse and the generation of hallucinated states.

Our experiments show that modifying sampling probabilities offer advantages over uniform sampling, achieving similar average per-step rewards with fewer pre-collected experiences and planning steps. SimuDICE also demonstrates greater robustness to variations in data collection policies.

Future work includes (1) incorporating a world model that can generate novel experiences, using a more stable and robust DICE estimator, and implementing a policy that can handle continuous state-action spaces; (2) further exploring the impact of altered sampling probabilities on the stability and robustness of SimuDICE across various data collection policies; and (3) evaluating the algorithm's performance in more complex, continuous environments where greater divergence between the behavioral and target policies is present.

Acknowledgments. S. Bongers and F. Oliehoek are supported by the Mercury Machine Learning Lab, a collaboration between TU Delft, UvA, and booking.com.

References

1. Alonso, E., Jelley, A., Micheli, V., Kanervisto, A., Storkey, A., Pearce, T., Fleuret, F.: Diffusion for world modeling: Visual details matter in atari (2024), https://arxiv.org/abs/2405.12399
2. Baird, L.: Residual algorithms: Reinforcement learning with function approximation. Machine Learning **20**(1-2), 65–81 (1995)
3. Cang, C., Rajeswaran, A., Abbeel, P., Laskin, M.: Behavioral priors and dynamics models: Improving performance and domain transfer in offline rl. arXiv preprint arXiv:2106.09119 (2021), abs/2106.09119
4. Clavera, I., Rothfuss, J., Schulman, J., Fujita, Y., Asfour, T., Abbeel, P.: Model-based reinforcement learning via meta-policy optimization. arXiv preprint arXiv:1809.05214 (2018), https://arxiv.org/abs/1809.05214
5. Deisenroth, M.P., Rasmussen, C.E.: Pilco: A model-based and data-efficient approach to policy search. In: Proceedings of the 28th International Conference on Machine Learning (ICML-11). pp. 465–472 (2011)
6. Ding, Z., Zhang, A., Tian, Y., Zheng, Q.: Diffusion world model (2024), https://arxiv.org/abs/2402.03570
7. Eysenbach, B., Khazatsky, A., Levine, S., Salakhutdinov, R.: Mismatched no more: Joint model-policy optimization for model-based rl. arXiv preprint arXiv:2110.02758 (2021)
8. Fujimoto, S., Meger, D., Precup, D.: Off-policy deep reinforcement learning without exploration. In: Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 2052–2062. PMLR (2019)
9. Hafner, D., Lillicrap, T., Norouzi, M., Ba, J.: Dreamer: Reinforcement learning with latent world models. In: International Conference on Learning Representations (2020)
10. Hafner, D., Pasukonis, J., Ba, J., Lillicrap, T.: Mastering diverse domains through world models. arXiv preprint arXiv:2301.04104 (2023)
11. Hafner, D., Schrittwieser, J., Mankowitz, D., Barreto, A., Lillicrap, T.: Mastering atari with discrete world models. arXiv preprint arXiv:2010.02193 (2020)
12. Hallak, A., Mannor, S.: Consistent on-line off-policy evaluation. In: Proceedings of the 34th International Conference on Machine Learning (ICML 2017). pp. 1372–1383. PMLR (2017)
13. Jafferjee, T., Imani, E., Talvitie, E.J., White, M., Bowling, M.: Hallucinating value: A pitfall of dyna-style planning with imperfect environment models. arXiv preprint arXiv:2006.04363 (2020)
14. Janner, M., Fu, J., Zhang, M., Levine, S.: When to trust your model: Model-based policy optimization. arXiv preprint arXiv:1906.08253 (2019), https://arxiv.org/abs/1906.08253
15. Jaques, N., Ghandeharioun, A., Shen, J.H., Ferguson, C., Lapedriza, À., Jones, N., Gu, S., Picard, R.W.: Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. arXiv preprint arXiv:1907.00456 (2019), https://arxiv.org/abs/1907.00456
16. Kidambi, R., Rajeswaran, A., Netrapalli, P., Joachims, T.: Morel: Model-based offline reinforcement learning. In: Advances in Neural Information Processing Systems (2020)
17. Kostrikov, I., Nair, A., Levine, S.: Offline reinforcement learning with implicit q-learning. arXiv preprint arXiv:2110.06169 (2021)

18. Kumar, A., Fu, J., Soh, M., Tucker, G., Levine, S.: Stabilizing off-policy q-learning via bootstrapping error reduction. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 32, pp. 451–463. Curran Associates, Inc. (2019), https://proceedings.neurips.cc/paper/2019/hash/c2073ffa77b5357a498057413bb09d3a-Abstract.html
19. Kumar, A., Fu, J., Tucker, G., Levine, S.: Stabilizing off-policy q-learning via bootstrapping error reduction. In: Advances in Neural Information Processing Systems. vol. 32. Curran Associates, Inc. (2019)
20. Kurutach, T., Clavera, I., Duan, Y., Tamar, A., Abbeel, P.: Model-ensemble trust-region policy optimization. In: International Conference on Learning Representations (2018), https://arxiv.org/abs/1802.10592
21. Lambert, N., Amos, B., Yadan, O., Calandra, R.: Objective mismatch in model-based reinforcement learning. arXiv preprint arXiv:2002.04523 (2020)
22. Lange, S., Gabel, T., Riedmiller, M.: Batch reinforcement learning. In: Wiering, M., van Otterlo, M. (eds.) Reinforcement Learning: State-of-the-Art, chap. 2, pp. 45–73. Springer (2012). https://doi.org/10.1007/978-3-642-27645-3_2
23. Levine, S., Kumar, A., Tucker, G., Fu, J.: Offline reinforcement learning: Tutorial, review, and perspectives on open problems. arXiv preprint arXiv:2005.01643 (2020)
24. Levine, S., Kumar, A., Tucker, G., Fu, J.: Offline reinforcement learning: Tutorial, review, and perspectives on open problems. arXiv preprint arXiv:2005.01643 (2020)
25. Liu, Q., Li, L., Tang, Z., Zhou, D.: Breaking the curse of horizon: Infinite-horizon off-policy estimation. In: Advances in Neural Information Processing Systems 31 (NeurIPS 2018). pp. 5356–5366. Curran Associates, Inc. (2018)
26. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529–533 (2015)
27. Murphy, S.A., van der Laan, M.J., Robins, J.M., Group, C.P.P.R.: Marginal mean models for dynamic regimes. Journal of the American Statistical Association **96**(456), 1410–1423 (2001)
28. Nachum, O., Chow, Y., Dai, B., Li, L.: Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections. In: Advances in Neural Information Processing Systems. pp. 2315–2325 (2019)
29. Precup, D., Sutton, R.S., Dasgupta, S.: Off-policy temporal-difference learning with function approximation. In: Brodley, C.E., Danyluk, A.P. (eds.) Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001). pp. 417–424. Morgan Kaufmann, San Francisco, CA (2001)
30. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons, New York (1994)
31. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016). https://doi.org/10.1038/nature16961
32. Sutton, R.: Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. Proceedings of the Seventh International Conference on Machine Learning pp. 216–224 (1990)
33. Sutton, R.S.: Dyna, an integrated architecture for learning, planning, and reacting. ACM SIGART Bulletin **2**(4), 160–163 (1991)
34. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, 2 edn. (2018)

35. Swazinna, P., Udluft, S., Runkler, T.A.: Overcoming model bias for robust offline deep reinforcement learning. Engineering Applications of Artificial Intelligence **104**, 104366 (2021). https://doi.org/10.1016/j.engappai.2021.104366
36. Tobin, J., Fong, R., Ray, A., Schneider, J., Zaremba, W., Abbeel, P.: Domain randomization for transferring deep neural networks from simulation to the real world. In: Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 23–30. IEEE (2017)
37. Towers, M., Terry, J.K., Kwiatkowski, A., Balis, J.U., Cola, G.d., Deleu, T., Goulão, M., Kallinteris, A., KG, A., Krimmel, M., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J.J., Shen, A.T.J., Younis, O.G.: Gymnasium (Mar 2023). https://doi.org/10.5281/zenodo.8127026, https://zenodo.org/record/8127025
38. Tsitsiklis, J.N., Van Roy, B.: Analysis of temporal-difference learning with function approximation. In: Advances in Neural Information Processing Systems 9 (NIPS 1996). pp. 1075–1081. MIT Press (1996)
39. Wu, Y., Tucker, G., Nachum, O.: Behavior regularized offline reinforcement learning. CoRR **abs/1911.11361** (2019)
40. Yang, M., Nachum, O., Dai, B., Li, L., Schuurmans, D.: Off-policy evaluation via the regularized lagrangian. In: International Conference on Learning Representations. ICLR (2020)
41. Yang, S., Zhang, S., Feng, Y., Zhou, M.: A unified framework for alternating offline model training and policy learning. In: Advances in Neural Information Processing Systems (2022), https://arxiv.org/abs/2210.05922
42. Yu, T., Kumar, A., Zhang, S., Gao, G., Levine, S.: Mopo: Model-based offline policy optimization. In: Advances in Neural Information Processing Systems (2020)
43. Zhang, R., Dai, B., Li, L., Schuurmans, D.: Gendice: Generalized offline estimation of stationary values. In: International Conference on Learning Representations (2020)
44. Zhang, S., Liu, B., Whiteson, S.: Gradientdice: Rethinking generalized offline estimation of stationary values. arXiv preprint arXiv:2001.11113 (2020)