Impact of State Visitation Mismatch Methods on the Performance of On-Policy Reinforcement Learning

Hongwoo Cho[1]
Supervisor(s): Frans Oliehoek[1]**, Stephan Bongers**[1]
[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Hongwoo Cho
Final project course: CSE3000 Research Project
Thesis committee: Frans Oliehoek, Stephan Bongers, Catholijn Jonker

An electronic version of this thesis is available at http://repository.tudelft.nl/.

Abstract

In the field of reinforcement learning (RL), effectively leveraging behavior-agnostic data to train and evaluate policies without explicit knowledge of the behavior policies that generated the data is a significant challenge. This research investigates the impact of state visitation mismatch methods on the performance of on-policy RL methods, an area crucial for improving policy performance in real-world applications where behavior policies are often unknown. Specifically, we compare the convergence speed and performance of Q-learning when initialized with Q-values learned through the Distribution Correction Estimation (DICE) method versus traditional random initialization. By generating datasets representing behavior and target policies, we employ the DICE estimator to initialize Q-values, and subsequently run Q-learning for both DICE-initialized and randomly-initialized scenarios. Our results demonstrate that initializing Q-learning with DICE Q-values enhances convergence speed, leading to faster attainment of near-optimal policies. This study provides valuable insights into the effectiveness of state visitation mismatch methods in improving the efficiency and performance of on-policy RL algorithms, contributing to the development of more robust RL applications in behavior-agnostic settings.

1 Introduction

Behavior-agnostic reinforcement learning (RL) addresses the challenge of learning effective policies from data generated by unknown behavior policies. This scenario is increasingly relevant as companies accumulate vast amounts of data without explicit knowledge of the underlying mechanisms. For instance, an electric car company might have extensive records of human driving behavior without understanding the specific decision-making processes of the drivers. The primary question is whether this data can be leveraged to train an AI agent capable of driving autonomously, while avoiding the errors made by human drivers. This research investigates the evaluation and optimization of policies using behavior-agnostic data.

RL has demonstrated remarkable success in a variety of applications, including recommendation systems, autonomous driving [13], games [6], robotics [8], conversational systems [2, 3] and in other fields. However, many of these successes rely on large amounts of data generated by simulators that mimic the environment. In real-world applications, such as autonomous driving, access to such simulators is often not feasible. Instead, we typically have access to off-policy data collected by potentially multiple, possibly unknown behavior policies [7]. This creates a challenge for deploying new target policies, as performance evaluation in the real environment can be expensive and risky. For example, allowing an AI agent to drive autonomously in a city poses significant safety concerns. This scenario exemplifies behavior-agnostic RL, where off-policy data is available, but the explicit behavior policies that generated the data are unknown. The concept of behavior-agnostic RL is thus critical for advancing AI in these scenarios [4].

Several state-of-the-art RL methods focus on policy evaluation through direct interaction with the environment. However, in many real-world applications, we often rely on offline data generated from unknown behavior policies due to the impracticality of continuous interaction with the environment. This behavior-agnostic setting introduces unique challenges, particularly the "curse of horizon," where importance weights used to correct for distribution mismatches grow exponentially with trajectory length [5]. Off-policy evaluation methods, like those used in behavior-agnostic RL, provide tools to leverage this offline data effectively. By correcting state-action visitation distributions instead of trajectory distributions as recent research has proposed, these methods can mitigate the exponential dependence on trajectory length. Although this dual formulation is computationally intensive, it shows promise in improving policy performance. By using off-policy evaluation methods such as the Distribution Correction Estimation (DICE) to initialize Q-values, we can provide a more accurate starting point for on-policy RL methods like Q-learning. This approach aims to enhance convergence speed, effectively bridging the gap between off-policy and on-policy paradigms and making the most of available offline data to inform online learning processes.

This leads to the following research question: What is the impact of state visitation mismatch methods on the performance of on-policy RL methods? Specifically, we aim to determine whether initializing Q-learning with Q-values estimated from behavior-agnostic data using the DICE method can improve performance compared to traditional random initialization. To explore these questions, we designed experiments in the Frozen Lake environment to compare the convergence speed and performance of Q-learning under different initialization strategies. By leveraging off-policy evaluation methods to inform on-policy learning, we seek to provide insights into the potential benefits of combining these two paradigms, ultimately contributing to more robust and efficient RL algorithms.

The rest of the paper is organized as follows. Section 2 explains the background information about reinfocement learning. Section 3 describes experiment set-up and is followed by an evaluation of relevant results in Section 4. The ethical implications and reproducibility of our work are discussed in Section 5. Finally, Section 6 suggests possible extensions to our work and concludes the paper.

2 Background

This section provides background information on the key concepts and methodologies used in this research. Section 2.1 explains about Markov Decision Processes and state visitation mismatches. Section 2.2 explains about policy evaluation and Distribution Correction Estimation (DICE) is explained in Section 2.3. Lastly, details on Q-Learning method is given in Section 2.4.

2.1 Markov Decision Processes and State Visitation Mismatches

This research considers an infinite-horizon Markov Decision Process (MDP) setting [11], in which the environment is described by a tuple $M = \langle S, A, R, T, \mu_0, \gamma \rangle$. In this tuple, S gives the state space, A gives the action space, R gives the reward function such that $R = S \times A \rightarrow R$, T gives the transition probability function, μ_0 gives the initial state distribution, and lastly γ gives the discount factor $\gamma \in (0, 1)$. This discount factor determines the importance of future rewards in the agent's decision-making process by dictating how much the agent values immediate rewards compared to future rewards. A higher γ places more importance on future rewards, promoting long-term strategies, while a lower γ emphasizes immediate rewards. The MDP diagram can be shown by figure 1.

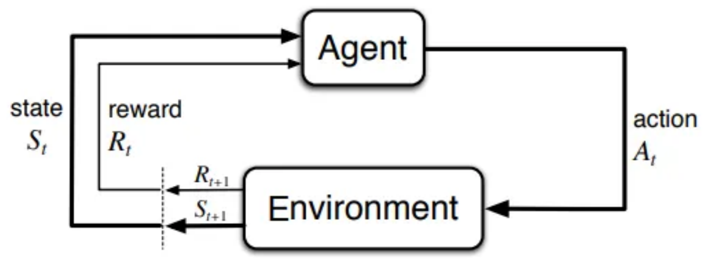

Figure 1: A diagram illustrating the interaction between the agent and the environment in a MDP. At each time step t, the agent observes the current state S_t and selects an action A_t. The environment then transitions to a new state S_{t+1} and provides a reward R_{t+1} to the agent based on the action taken. This process repeats as the agent interacts with the environment over time, aiming to maximize the cumulative reward. Reproduced from [12].

In the context of reinforcement learning, a trajectory τ is a sequence of state-action-reward transitions that the agent experiences as it interacts with the environment. Formally, a trajectory can be represented as: $\tau = (s_0, a_0, r_1, s_1, a_1, r_2, ...)$, where s_t is the state at time step t, a_t is the action taken at time step t and r_{t+1} is the reward received after taking action a_t and transitioning to state s_{t+1}. This can be shown as in figure 2.

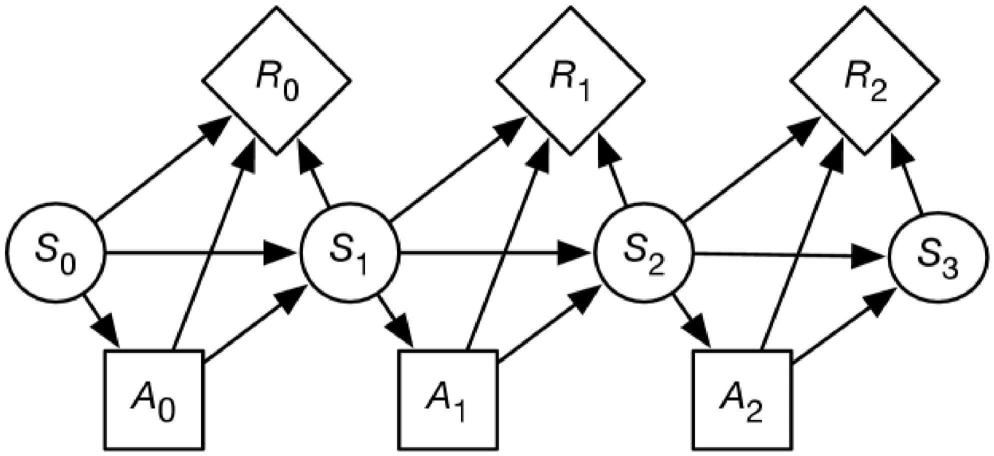

Figure 2: An example of a trajectory in a MDP. It shows the sequence of states (S_0, S_1, S_2, S_3), actions $(A_0.A_1, A_2)$, and rewards (R_0, R_1, R_2) over time. Reproduced from [10].

State visitation mismatches occur when the distribution of states and actions encountered by the agent under the behavior policy (used to collect offline data) differs from the distribution under the target policy (the policy we aim to optimize during online RL). This discrepancy can lead to significant challenges in policy evaluation and optimization. In the context of RL, the behavior policy π_b is the policy used to collect the data, while the target policy π is the policy being evaluated or optimized. The distribution mismatch arises because the target policy may induce a different distribution over state-action pairs compared to the behavior policy. This leads to biased estimates if the mismatch is not properly corrected. For example, if the behavior policy tends to visit certain states more frequently than the target policy, the value estimates for those states may be biased, affecting the overall performance of the target policy.

2.2 Policy Evaluation

Policy evaluation refers to the process of estimating the value function of a policy, which is defined as the normalized expected cumulative discounted reward it obtains [15]. In reinforcement learning, the value of a policy π is a crucial metric that reflects the long-term expected return when following the policy starting from an initial state distribution μ_0. According to the paper "Off-Policy Evaluation via the Regularized Lagrangian" by Yang, the value of a policy π is given by:

$$\rho(\pi) := (1-\gamma)\mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t) \mid s_0 \sim \mu_0, \forall t, a_t \sim \pi(s_t), s_{t+1} \sim T(s_t,a_t)\right] \tag{1}$$

This definition of the value function as the normalized expected cumulative discounted reward is equivalent to the normalized expected per-step reward when considering an infinite time horizon and the discount factor.

Off-policy evaluation (OPE) is particularly valuable in scenarios where the data is generated by a behavior policy different from the target policy. OPE methods estimate the value of a target policy using data collected from the behavior policy. This is crucial in behavior-agnostic settings where the behavior policies generating the data are unknown. One major challenge in off-policy evaluation is the "curse of horizon," where the importance weights used for distribution correction grow exponentially with trajectory length [5]. To mitigate this, DICE (Distribution Correction Estimation) methods provide a way to effectively leverage offline data for more accurate policy evaluation, which is explained in Section 2.4.

2.3 Q-Learning

Q-Learning is a model-free reinforcement learning algorithm that enables agents to learn optimal policies through interaction with an environment [14]. It is a form of temporal difference learning, which combines ideas from dynamic programming and Monte Carlo methods. Q-Learning aims to estimate the optimal action-value function, Q^*, which represents the maximum expected cumulative reward for taking a given action in a given state and following the optimal policy thereafter.

Q-Learning operates by estimating the value of taking a specific action in a specific state and using these estimates to inform decision-making. The value of a state-action pair, known as the Q-value, represents the expected cumulative reward the agent can achieve starting from that state and action, and then following an optimal policy. The Q-value $Q(s,a)$ for a state

s and action a is updated incrementally using the formula:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[R(s,a) + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{2}$$

where:

- $Q(s,a)$ is the current Q-value of state s and action a.
- α is the learning rate, which determines how much new information overrides the old information.
- $R(s,a)$ is the reward received after taking action a in state s.
- γ is the discount factor, which determines the importance of future rewards.
- $\max_{a'} Q(s',a')$ is the maximum Q-value of the next state s' over all possible actions a'.

2.4 Distribution Correction Estimation (DICE)

The Distribution Correction Estimation (DICE) method addresses the challenge of distribution mismatch by correcting state-action visitation distributions instead of trajectory distributions. This approach reduces the exponential growth of importance weights with trajectory length, thereby alleviating the "curse of horizon." The DICE estimator helps correct the distribution mismatch between the behavior policy and the target policy. Specifically, it computes the Q-values for the state and action pairs, ensuring that these estimates account for the differences in state-action visitation distributions.

The DICE methods utilize the following expression for the policy value, as stated by Yang [15]:

$$\rho(\pi) = \mathbb{E}_{(s,a,r)\sim d_D}[\zeta^*(s,a) \cdot r], \tag{3}$$

where $\zeta^*(s,a) := d^\pi(s,a)/d^D(s,a)$ is the distribution correction ratio. Existing DICE estimators aim to approximate this ratio without directly knowing d_π or d_D, and then use the equation above to estimate $\rho(\pi)$.

To put this concept into practice, the DICE estimator uses the following optimization framework:

$$\begin{aligned} &\max_{\zeta \geq 0} \min_{Q,\lambda} \mathcal{L}_D(\zeta, Q, \lambda) := (1-\gamma)\mathbb{E}_{s_0\sim\mu_0, a_0\sim\pi}[Q(s_0,a_0)] + \lambda \\ &+ \mathbb{E}_{(s,a,r,s')\sim d_D, a'\sim\pi}[\zeta(s,a)\cdot(\alpha_R \cdot R(s,a) + \gamma Q(s',a') - Q(s,a) - \lambda)] \\ &+ \alpha_Q \mathbb{E}_{(s,a)\sim d_D}[f_1(Q(s,a))] - \alpha_\zeta \mathbb{E}_{(s,a)\sim d_D}[f_2(\zeta(s,a))] \end{aligned} \tag{4}$$

.

In simpler terms, this equation seeks to find the best correction factor ζ that minimizes the discrepancy between the expected rewards under the behavior policy and the target policy. By adjusting the Q-values $Q(s,a)$ to reflect this correction, the DICE method provides a more accurate estimate of the value of following the target policy. This helps in better policy evaluation and optimization by ensuring that the initial Q-values are closer to the true values, thereby reducing the exploration needed to discover optimal actions.

3 Experimental Setup

This section details the experimental setup used to evaluate the performance of Q-Learning when initialized with Q-values obtained through the Distribution Correction Estimation (DICE) method versus traditional random initialization [15]. The goal is to measure the convergence speed and effectiveness of each approach in achieving a near-optimal policy. The setup comprises three main steps: dataset generation, running the DICE estimator, and lastly the Q-Learning.

3.1 Dataset Generation

To evaluate the impact of state visitation mismatch methods, two datasets were generated using a simulation environment based on an infinite-horizon MDP. The datasets contain sequences of state-action-reward transitions, with each transition representing the agent's interaction with the environment. The first dataset is the behavior policy dataset. This setting ensures that the behavior policy is different from the target policy. The other dataset is the target policy dataset, in which this setting ensures that the target policy is followed exactly. Each dataset comprises 400 trajectories with a maximum trajectory length of 250 steps. The environment used for this purpose is "FrozenLake". These two types of dataset were created by setting the alpha value in the command when creating the dataset, which is a parameter that controls the policy used for generating the trajectories. Alpha of 0 is used to create the behavior policy dataset, representing the behavior policy, which is sub-optimal and explores the environment more randomly. This setting ensures a diverse set of state-action-reward transitions, capturing various behaviors that the agent might encounter. Then, alpha of 1 is used to create the target policy dataset, representing the target policy, which is closer to the optimal policy. The target policy dataset follows a more deterministic approach, focusing on the intended optimal behavior in the FrozenLake environment. The exact command can be found in this repository[1].

The objective of the "FrozenLake" environment is to navigate the agent from the starting point at the top-left corner to the goal at the bottom-right corner of the grid while avoiding holes in the ice. The ice is slippery, making the agent's movements stochastic; hence, the agent may not always move in the intended direction. The agent receives a reward of $+1$ upon reaching the goal. If the agent falls into a hole, the episode ends with a reward of 0. No rewards are given for moving to safe tiles. The challenge is to learn a policy that maximizes the cumulative reward by reaching the goal efficiently while avoiding holes [9]. An example of the frozen lake environment is shown by figure 3.

[1]Code and instructions: https://github.com/hwcho12357/ResearchProject

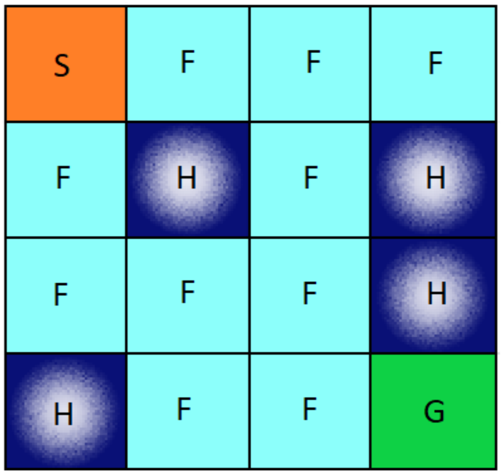

Figure 3: An example of the FrozenLake environment from OpenAI Gym [9]. The agent starts at the top-left corner and must navigate to the bottom-right corner while avoiding holes. S represents the start state, F represents the FrozenLake state in which the agent can move onto, H represents the hole and G represents the goal state.

3.2 Running the DICE Estimator

Once the datasets are generated, the DICE estimator is used to estimate the average reward of the target policy using data collected from the behavior policy. The DICE estimator helps correct the distribution mismatch between the behavior policy and the target policy. While running the DICE estimator, the following consideration has been made regarding the target policy selection. The choice of the target policy can significantly influence the estimated Q-values. Different target policies might yield different Q-values through the DICE estimator. For this study, we select the target policy that aligns with the intended optimal behavior in the FrozenLake environment.

The DICE estimator works by initializing two neural networks, the nu and zeta networks, which are trained to minimize the Bellman residuals and correct for distribution mismatches. During the training process, these networks iteratively update their weights to better approximate the state-action values. The weights, derived from these trained networks, effectively represent the Q-values for the state-action pairs. By applying the DICE estimator, we generate a more accurate set of Q-values that reflect the corrected visitation distributions. These Q-values are then saved and used to initialize the Q-values in the Q-Learning algorithm.

3.3 Q-Learning

In this paper, we use Q-Learning to evaluate the convergence speed and performance of policies initialized with Q-values estimated by the DICE method versus those with randomly initialized Q-values. The process involves two sets of Q-values, which will be described as DICE Q-values: Q-values are initialized using estimates from the DICE estimator. Then another set of Q-values will be described in this report as the Random Q-values: Q-values are initialized to zero, representing traditional random initialization.

The training process involves the following steps:

1. Environment Interaction: The agent interacts with the FrozenLake environment, taking actions and receiving rewards based on the current policy derived from the Q-values.

2. Q-Value Update: After each action, the Q-values are updated using the Q-Learning update rule as shown in Equation 2.

3. Performance Monitoring: The performance is monitored by tracking the average reward per step. This involves calculating the average reward received at each step of the learning process.

The Q-Learning algorithm iteratively updates the Q-values based on the agent's interactions with the environment, using the update formula mentioned above. The goal is to maximize the cumulative reward over time by learning the optimal policy. After running the Q-Learning algorithm with two initial set of Q-values as described above, the number of episodes required for Q-Learning to reach a near-optimal policy will be tracked for both initialization methods.

4 Results and Discussion

In this section, we analyze the performance of Q-learning initialized with Q-values obtained through the Distribution Correction Estimation (DICE) method versus traditional random initialization.

The Q-values extracted from the DICE estimator are shown in Table 1. These values represent the estimated rewards for state-action pairs based on the behavior policy data. State 1 refers to the top-left corner as shown in Figure 3, state 2 is the state right to it and state 16 is the bottom-right corner, which is the end state. For comparison, the Q-values for the random initialization were all set to 0.

Table 1: Q-Values from DICE Estimator

State	Move Left	Move Down	Move Right	Move Up
State 1	0.50368656	0.28417100	0.24744200	0.30284700
State 2	0.00666858	0.00369054	0.00309231	0.00149879
State 3	0.02507504	0.00371495	0.00309155	0.01011084
State 4	0.05221552	0.01277991	0.00067256	0.00430612
State 5	0.26031873	0.00403238	0.00468556	0.00260281
State 6	0.00233800	0.00251356	0.00273770	0.00179974
State 7	0.02266099	0.00031998	0.00077225	0.00000000
State 8	0.00000000	0.00000000	0.00000000	0.01678710
State 9	0.00223199	0.00417178	0.00007266	0.13045901
State 10	0.01144835	0.11829945	0.00535556	0.00362227
State 11	0.00075249	0.01784247	0.00250205	0.00025089
State 12	0.01815625	0.00211173	0.00307075	0.00129442
State 13	0.00279297	0.00271804	0.00299118	0.00082534
State 14	0.00014303	0.00039642	0.02397827	0.00021320
State 15	0.00124678	0.01325846	0.00001431	0.00456260
State 16	0.00000000	0.00000000	0.00000000	0.00000000

By running the Q-learning method on the initial distributions of Q-values (DICE-initialized and zero-initialized), we obtained the cumulative rewards shown in Figure 4.

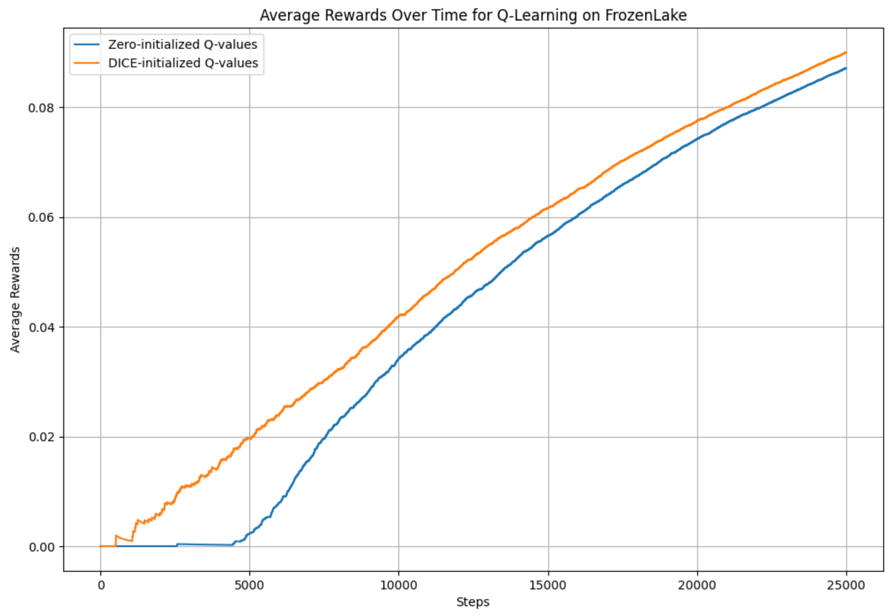

Figure 4: Comparison of Q-Learning convergence with zero-initialized and DICE-initialized Q-values

The convergence speed of Q-learning, when initialized with DICE Q-values, showed a noticeable improvement over random initialization. The results indicate that starting with Q-values estimated by the DICE method leads to faster attainment of near-optimal policies. Specifically, the DICE-initialized Q-values allow the agent to leverage pre-existing knowledge about the state-action values, thus requiring fewer episodes to converge compared to starting from scratch with zero-initialized Q-values.

This improvement can be due to several factors. First, the DICE Q-values provide a more accurate initial estimate of the expected rewards, which reduces the amount of exploration needed for the agent to discover optimal actions. By starting with these better initial estimates, the agent has a head start in identifying promising actions, thus accelerating the learning process. Second, with more accurate initial values, the agent can begin exploiting known good actions sooner. This early exploitation of high-value actions leads to faster convergence towards an optimal policy, as the agent spends less time exploring suboptimal actions and more time refining its strategy based on effective behaviors.

In behavior-agnostic RL settings, where the explicit behavior policies generating the data are unknown, the ability to initialize Q-learning with more accurate Q-values can significantly enhance the efficiency and effectiveness of the learning process. This is particularly valuable in real-world applications, such as autonomous driving [13], where safety and reliability are highly prioritized, and the cost of exploration can be high.

5 Responsible Research

Our research is focused on developing and evaluating reinforcement learning (RL) methods that utilize behavior-agnostic data. This area of study has significant potential for real-world applications, particularly in fields that impact human safety and decision-making, such as autonomous driving [13]. Given the critical nature of these applications, it is essential to address the ethical implications and ensure the reproducibility and repeatability of our research findings. Moreover, we believe that it is important to adhere to the principles documented by the Netherlands Code of Conduct for Research Integrity [1] throughout the entire project, therefore clearly documenting our results, data, and experimental process are also vital.

5.1 Ethical Implications of Behavior-Agnostic RL

The research conducted in this project involves leveraging behavior-agnostic data to train and evaluate RL policies, which raises several ethical implications, especially when applied to critical and safety-sensitive domains such as autonomous driving [13]. Ensuring the safety and reliability of AI systems in these contexts is paramount. The potential for an AI system to inherit biases or mistakes from the behavior data used for training necessitates rigorous testing and validation to confirm that the trained policies can operate safely and reliably without causing harm to humans or property. Moreover, the transparency and accountability of these systems are essential.

5.2 Reproducibility and Repeatability

To ensure the reproducibility and repeatability of our results in Section 4, we have made all our code, datasets, and detailed experimental setups and data publicly accessible. All the code developed for this project, including the implementations of the DICE estimator, which has been cloned from the Google Reserach repository, and Q-Learning algorithms, is available in a public repository. The code and instructions can be found here.[2] Additionally, the datasets generated for the experiments, including the behavior and target policy data, are also accessible. Furthermore, we have also documented the scripts and parameters used in our experiments, which may be used by future researchers to validate our findings.

A critical aspect of ensuring reproducibility is the use of a random seed in our experiments. The random seed used in the experiment was set to 1. The random seed is critical in ensuring reproducibility because it controls the initialization of the random number generator used in various parts of the algorithm, such as state and action selection in RL environments. Specifically, the random seed was used in the following components:

- **Dataset Generation**: The random seed ensured that the state-action-reward transitions generated for both the behavior and target policy datasets were consistent across different runs. This consistency is crucial for comparing the performance of the algorithms under identical conditions.
- **DICE Estimator**: The random seed was used to control the stochastic processes involved in the DICE estimator. This ensured that the Q-values estimated by the DICE method were consistent and reproducible across multiple runs.
- **Q-Learning (Epsilon-Greedy Method)**: Q-Learning (Epsilon-Greedy Method): In the Q-Learning algorithm, the epsilon-greedy method was used to balance exploration and exploitation. The random seed ensured that the sequence of exploratory actions taken by the agent was consistent across different runs, leading to identical learning trajectories and outcomes.

By setting the random seed, we ensure that the same sequence of random numbers is generated each time the code is run, leading to identical results. This consistency is crucial for other researchers who may want to replicate our experiments and validate our findings.

6 Conclusions and Future Work

In this work, we investigated the impact of state visitation mismatch methods on the performance of on-policy reinforcement learning (RL) methods. Specifically, we aimed to determine whether initializing Q-learning with Q-values obtained through the Distribution Correction Estimation (DICE) method could enhance convergence speed compared to traditional random initialization. Our research question focused on understanding the effects of state visitation mismatch methods, the potential performance improvements from starting with the target policy, and the effectiveness of using an on-policy method from the beginning.

[2]Code and instructions: https://github.com/hwcho12357/ResearchProject

Our study demonstrated that initializing Q-learning with DICE Q-values enhances convergence speed. The DICE-initialized Q-learning achieved near-optimal policies more quickly than when initialized with random Q-values, owing to better initial estimates that reduced the need for exploration. Additionally, the DICE Q-values provided enhanced stability by reducing the variance in rewards, leading to a more stable learning process. This ability to effectively use behavior-agnostic data for initializing Q-learning is particularly valuable in real-world applications where exploration costs are high, and safety is critical.

While our findings demonstrate the effectiveness of initializing Q-learning with DICE Q-values, there are some areas for future improvement. The current study's scalability to larger, more complex environments needs further investigation. For future work, we recommend validating our methods in more complex OpenAI Gym environments such as LunarLander, BipedalWalker, and Humanoid, which offer higher complexity and variability. These environments will provide a more rigorous test of the DICE method's robustness and effectiveness. Additionally, exploring different target policies to assess how they affect the Q-values generated by the DICE estimator could provide insights into optimizing the initialization process further. Selecting target policies that yield better Q-values may enhance the performance and robustness of the Q-learning algorithm.

References

[1] NWO (The Dutch Research Council). Netherlands code of conduct for research integrity.

[2] Jianfeng Gao, Michel Galley, and Lihong Li. Neural approaches to conversational ai, 2019.

[3] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation, 2016.

[4] Yuxi Li. Reinforcement learning applications, 2019.

[5] Qiang Liu, Lihong Li, Ziyang Tang, and Dengyong Zhou. Breaking the curse of horizon: Infinite-horizon off-policy estimation, 2018.

[6] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

[7] Ofir Nachum, Yinlam Chow, Bo Dai, and Lihong Li. Dualdice: Behavior-agnostic estimation of discounted stationary distribution corrections, 2019.

[8] OpenAI, Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, Jonas Schneider, Szymon Sidor, Josh Tobin, Peter Welinder, Lilian Weng, and Wojciech Zaremba. Learning dexterous in-hand manipulation, 2019.

[9] Bethany D. Pena and Daniel T. Banuti. Reinforcement learning for pathfinding with restricted observation space in variable complexity environments, 2021.

[10] David L. Poole and Alan K. Mackworth. *Artificial Intelligence: Foundations of Computational Agents.* 2023.

[11] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* Wiley Series in Probability and Statistics. 1994.

[12] Ayush Singh. Reinforcement learning : Markov-decision process (part 1).

[13] Letian Wang, Jie Liu, Hao Shao, Wenshuo Wang, Ruobing Chen, Yu Liu, and Steven L. Waslander. Efficient reinforcement learning for autonomous driving with parameterized skills and priors, 2023.

[14] Christopher J. C. H. Watkins and Peter Dayan. Q-learning, 1992.

[15] Mengjiao Yang, Ofir Nachum, Bo Dai, Lihong Li, and Dale Schuurmans. Off-policy evaluation via the regularized lagrangian, 2020.