The Effect of State-visitation Mismatch on Off-policy Performance in Behaviour-agnostic Reinforcement Learning

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

June 23, 2024

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

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

Abstract

Off-policy evaluation has some key problems with one of them being the "curse of horizon". With recent breakthroughs [1] [2], new estimators have emerged that utilise importance sampling of the individual state-action pairs and reward rather than over the whole trajectory. With the difference between behaviour and target policy, the state-visitation mismatch occurs. This paper is interested in answering the question how the degree of state-visitation mismatch affects the overall target policy performance. The approach is to calculate the state-visitation mismatch with the KL divergence, which consists of the state-visitation distribution of the behaviour policy and the distribution correction ratio of the DICE estimator. The state-visitation mismatch can be quantified in way. Furthermore, the effect on the target policy performance is quantified by the MSE between the estimated, empirical cumulative reward and the estimated reward by the DICE estimator. By analysing the KL divergence and MSE value, one may argue that the state-visitation mismatch does impact the performance of the target policy but further research needs to be conducted.

1 Introduction

Reinforcement learning is a section of AI that has demonstrated notable successes in various domains such as robotics, games and recommendation systems [3]. A key problem within reinforcement learning is policy evaluation, which aims to estimate the agent's reward based on the actions it took given the target policy in an environment. While the data the agent is trained on could be generated by interacting directly with the environment, this is not always possible due to high costs/risks. Behaviour-agnostic reinforcement learning provides a solution by having agents learn based on off-policy data where they do not know the behaviour policies that generated said data [4].

Behaviour-agnostic reinforcement learning presents numerous challenges with one significant challenge known as the "curse of horizon" [1]. This refers to the problem of using importance sampling to accurately estimating the value of a target policy over a long time period using off-policy evaluation methods. The value of a policy is estimated by summing the rewards obtained over a sequence of visiting states and taking actions in the environment. As the time horizon increases, the variance of the estimated value also increases. This variance poses challenges, particularly in behaviour-agnostic reinforcement learning scenarios, where the data collected by the behaviour policy may not accurately represent the long-term rewards of following the target policy.

Existing research has made significant advancements in off-policy evaluation and behaviour-agnostic reinforcement learning. Firstly, researchers have found a way to break the "curse of horizon" by using importance sampling on the average visitation distribution of state-action pairs, reducing the estimator's variance [1]. Furthermore, researchers have been able to develop estimators that model the ratio between the state-visitation distributions of the behaviour and target policy. These estimators are called and categorised as the DIstribution Correction Estimation (DICE) family with examples being DualDICE [4], AlgaeDICE [5], GenDICE [6] and GradientDICE [7]. In addition, researchers have created a unified framework that reduces the estimators' variance and bias by formulating the off-policy evaluation problem as a linear program [8]. This approach combines the strengths of various DICE estimators by optimising a linear objective function, resulting in more accurate estimations. However, there remains a knowledge gap regarding how state-visitation mismatches, arising

from differences between the behaviour and target policies, affect the performance of the target policy.

The objective of this research is to investigate how state-visitation mismatches affect target policy performance in behaviour-agnostic reinforcement learning scenarios. First, the degree of state-visitation mismatch and target policy performance is identified and quantified using various metrics. Afterwards, an empirical study is carried out to examine how the degree of state-visitation mismatch may influence the performance of the target policy in multiple experimental configurations. With a focus on high-risk and/or high-cost applications, this research attempts to enhance the development of greater reinforcement learning techniques and methods.

2 Understanding Off-policy Evaluation

This section provides the background necessary to understand the concept of off-policy evaluation in reinforcement learning. First of all, the Markov Decision Process (MDP) is introduced, which describes the environment and how an agent interacts with it. Subsequently, the performance of a policy is quantified by the normalised expected cumulative discounted reward, providing a measure of the agent's performance. Thereafter, the concept of the state-visitation mismatch, importance sampling and curse of horizon are explained, emphasising the challenges in off-policy evaluation. Last of all, the application of DICE estimators in off-policy evaluation helps to address the state-visitation mismatch and provide better evaluations.

2.1 Markov Decision Process (MDP)

A Markov Decision Process (MDP) is an abstraction of the environment in reinforcement learning and describes how an agent interacts with the environment to generate the training data under a policy π. An MDP is defined by a tuple $(S, A, R, T, \mu_0, \gamma)$, where [9]:

- S represents the state space;
- A represents the action space;
- R is the reward function $R(s, a)$, which gives the reward after transitioning from state s by taking action a;
- T is the transition probability function $T(s'|s, a)$, which defines the probability of transitioning to state s' from state s after taking action a;
- μ_0 is the initial state distribution;
- γ is the discount factor where $\gamma \in (0, 1]$.

Starting at an initial state $s_0 \sim \mu_0$, a behaviour policy π interacts with the environment to produce a probability distribution $\pi(\cdot|s_t)$ across the action space A. An action a_t is sampled from this distribution and applied to the environment at each step $t \geq 0$. Afterwards, the environment generates a reward r_t based on the reward function $R(s_t, a_t)$ and transitions to a next state s_{t+1} based on the transition probability function $T(s_{t+1}|s_t, a_t)$.

2.2 Policy Evaluation

The value of a policy π is defined as the normalised expected cumulative discounted reward [8]:

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

- π is the strategy the agent follows in the environment, mapping states s_t to actions a_t;
- $\rho(\pi)$ represents the long-term expected reward (value) of following policy π, normalised by the factor 1 - γ;
- γ is the discount factor that determines the present value of future rewards. A smaller γ values future rewards less, while a larger γ values them more;
- E represents the expected value or average over all possible sequences of state and actions (trajectories) the agent could take;
- $\sum_{t=0}^{\infty}\gamma^t R(s_t,a_t)$ is the sum of rewards after taking action a_t in state s_t where time step t goes from 0 to infinity;
- $s_0 \sim \mu_0$ indicates that the initial state s_0 comes from the initial state distribution μ_0;
- $\forall t$, $a_t \sim \pi(s_t)$ indicates that for each time step t, the action a_t is chosen according to the policy π based on the current state s_t;
- $s_{t+1} \sim T(s_t,a_t)$ indicates that the next state s_{t+1} is determined by the transition probability function $T(s_{t+1}|s_t,a_t)$.

To summarise, the equation describes the cumulative reward that an agent could expect by following a specific policy π from any initial state while taking the entire future trajectory and discounted rewards into account. A trajectory is a sequence of states and actions an agent experiences while interacting with an environment. In reinforcement learning the policy being evaluated is referred to as the target policy.

2.3 State-visitation Mismatch

State-visitation mismatch occurs when the state distribution visited by the behaviour policy differs from the state distribution visited by the target policy [9]. A larger mismatch could lead to inaccurate estimates in off-policy evaluation since the data collected under the behaviour policy may not accurately reflect the performance of the target policy. The mismatch is quantified by comparing the state-visitation distributions of the two policies and is defined by:

$$\Delta_{SV} = \sum_{s\in S, a\in A} |d^{\pi}(s,a) - d^{\mu}(s,a)|, \tag{2}$$

- Δ_{SV} represents the state-visitation mismatch;
- $d^{\pi}(s,a)$ is the state-action pair probability under the target policy π;

- $d^{\mu}(s, a)$ is the state-action pair probability under the behaviour policy μ;
- S represents the finite state space;
- A represents the finite action space.

2.4 Importance Sampling

Importance sampling is a method to adjust the collected data to account for the differences between the behaviour and target policy by re-weighting the data using importance weights [1]. An importance weight is the density ratio between the probability of selecting an action under the target policy π and the probability of selecting the same action under the behaviour policy μ. The importance weight/density ratio is defined by:

$$\beta_t = \frac{\pi(a_t|s_t)}{\mu(a_t|s_t)}, \tag{3}$$

- β_t represents the importance weight/density ratio at time step t;
- $\pi(a_t|s_t)$ is the probability of selecting action a_t in state s_t under the target policy π;
- $\mu(a_t|s_t)$ is the probability of selecting action a_t in state s_t under the behaviour policy μ.

2.5 Curse of Horizon

The curse of horizon refers to the difficulty of applying importance sampling to estimate the value of a target policy over long time horizons [1]. The density ratio of a trajectory is the product of density ratios of all the state-action pairs in the trajectory and is defined by:

$$w_{0:T}(\tau) = \prod_{t=0}^{T} \beta_t, \tag{4}$$

- $w_{0:T}(\tau)$ represents the density ratio of trajectory τ up to time step T;
- $\prod_{t=0}^{T} \beta_t$ is the product T density ratios.

Since the trajectory density ratio is the product of T density ratios, the variance may grow exponentially with T. However, researchers have developed an approach that significantly lowers the variance of the density ratio and makes it independent of the time horizon [1]. This is achieved by importance sampling on the average state-visitation distribution of single steps of state-action pairs instead of the distribution of whole trajectories.

2.6 Off-policy Evaluation Using the DICE Estimators

The goal of off-policy evaluation is to estimate the performance of a target policy using training data generated by a different behaviour policy. The behaviour policy could be either partially or fully unknown. This separation of policies is important in many real-life situations, such as autonomous driving and robotics, where deploying the target policy directly could raise safety concerns [10].

The Distribution Correction Estimation (DICE) family is a group of estimators, including DualDICE [4], AlgaeDICE [5], GenDICE [6] and GradientDICE [7]. These estimators estimate $\rho(\pi)$ by utilising a fixed, finite dataset $\mathcal{D} = \left\{\left(s_0^{(i)}, s^{(i)}, a^{(i)}, r^{(i)}, s'^{(i)}\right)\right\}_{i=1}^{N}$, where [8]:

- $s_0^{(i)} \sim \mu_0$, indicating that the initial state $s_0^{(i)}$ comes from the initial state distribution $\mu_0^{(i)}$;
- $(s^{(i)},a^{(i)}) \sim d^{\mathcal{D}}$, indicating that the state-action pair is sampled from some unknown distribution $d^{\mathcal{D}}$;
- $r^{(i)}$ is the reward obtained by the reward function $R(s^{(i)}, a^{(i)})$;
- $s'^{(i)} \sim T(s^{(i)}, a^{(i)})$, indicating that the next state is determined by the transition probability function $T(s^{(i)}, a^{(i)})$.

The DICE estimators estimate $\rho(\pi)$ as follows [8]:

$$\rho(\pi) = E_{(s,a,r)\sim d^{\mathcal{D}}}\left[\zeta^*(s,a) \cdot r\right], \tag{5}$$

- $(s, a, r) \sim d^{\mathcal{D}}$ is shorthand notation for $(s, a) \sim d^{\mathcal{D}}$, $r^{(i)} = R(s^{(i)}, a^{(i)})$, $s'^{(i)} \sim T(s^{(i)}, a^{(i)})$;
- $\zeta^*(s, a) := \frac{d^{\pi}(s,a)}{d^{\mathcal{D}}(s,a)}$ is the distribution correction ratio, also known as the per-step density ratio, where:
  - $d^{\pi}(s, a)$ is the probability of selecting action a in state s under the target policy π;
  - $d^{\mathcal{D}}(s, a)$ is the probability of selecting action a in state s in the dataset $\mathcal{D}$.
- $E_{(s,a,r)\sim d^{\mathcal{D}}}$ is the expected state-action-reward triplets from the dataset$\mathcal{D}$.

Overall, $\rho(\pi)$ is calculated by summing the distribution correction ratios multiplied by the corresponding rewards and then averaging the sum over all samples in the dataset $\mathcal{D}$. The correction ratio $\zeta^*(s, a)$ represents the differences between the state-visitation probability distributions of the behaviour and target policy (state-visitation mismatches), and helps with reducing the estimator's variance in order to provide more reliable policy estimations. The DICE estimators approximate this ratio without knowledge of d^{π} or $d^{\mathcal{D}}$.

3 Related Work

Although off-policy evaluation has been extensively researched in contextual bandits [11] [12], recent discoveries within off-policy reinforcement learning indicate extended research to improve state-of-the-art off-policy evaluation methods and estimators will not stop any time soon. The recent breakthrough of tackling the curse of horizon by calculating the importance weights based on the state-visitation distributions rather than the cumulative product of the whole trajectory [1], has been the foundation for new off-policy estimators and methods. First of all, a new estimator based on double reinforcement learning [13] was created that utilises both importance sampling and q-functions [2]. Q-functions are directly estimated

in the direct method approach to estimate the target policy performance. Moreover, the estimators of the DICE family, such as [4], [5], [6] and [7], all model the ratio between the state-visitation distributions of the behaviour and target policy to estimate the value of the target policy. In addition, a Lagrangian-type estimator similar to the DICE estimators was developed that is doubly robust and reduces bias [14]. However, this estimator is not suitable for the behaviour-agnostic setting. Lastly, an off-policy policy optimisation method was developed that corrects for the difference in state-visitation distributions between the behaviour and target policy by minimising the KL divergence between the two probability distributions [15]. However, this paper focusses more on the effect of the state-visitation mismatch on the target policy performance (off-policy evaluation) rather than using the state-visitation mismatch for off-policy policy optimisation. Furthermore, the action space is discrete as opposed to the continuous action space of the policy optimisation paper.

4 Methodology

This section describes the methodology used to answer the research question: **How does the degree of state-visitation mismatch impact the performance of target policies in behaviour-agnostic off-policy evaluation?** Firstly, three metrics are discussed to quantify the state-visitation mismatch between the behaviour and target policy, the evaluated performance of the target policy and the effect of the state-visitation mismatch on the target policy performance respectively. Afterwards, a novel approach is introduced to conduct the empirical analysis of the effect of the state-visitation mismatch on the target policy performance by creating datasets based on the behaviour policy, running the DICE estimator on the datasets and visualising the results. More details about the algorithms and choice of DICE estimator are provided later on in Section 4.2.2.

4.1 Quantifying Metrics

Before running the algorithms, the metrics necessary to quantify the effect of state-visitation mismatch on the target policy performance are provided. First of all, the KL divergence is considered to measure the state-visitation mismatch between the behaviour and target policy. Secondly, the target policy performance evaluated by the cumulative reward is explained. To conclude, the mean squared error (MSE) is discussed to quantify the difference between the empirical performance of a policy and the estimator's approximation.

4.1.1 Kullback-Leibler Divergence

The Kullback-Leibler (KL) divergence is a metric to quantify the difference between two probability distributions [16]. It is used to measure the degree of state-visitation mismatch between the behaviour and target policy. It quantifies the difference between two distributions P and Q, where P represents the empirical probability distribution of the dataset while Q represents an approximate probability distribution of the dataset. The KL divergence is defined as:

$$D_{KL}(P|Q) = \sum_{x} P(x) \log \frac{P(x)}{Q(x)} \tag{6}$$

- x is an instance of a state-action pair (s, a) where s comes from a finite state space S and a comes from a finite action space A;

- P represents the empirical probability distribution, which is calculated by iterating over the dataset created from the environment, counting the occurrences of each state-action pair (s, a) and dividing each state-action pair count by the total number of state-action pairs;
- Q represents the approximate probability distribution, which is retrieved by running the DICE estimators. The DICE estimators estimate the distribution correction ratio $\zeta(s, a)$ between the state distributions of the behaviour and target policy in order to estimate the value of the target policy $\rho(\pi)$ as mentioned in equation 5.

The distribution correction ratio $\zeta^*(s, a)$ is calculated by $\frac{d^\pi(s,a)}{d^\mathcal{D}(s,a)}$, with the probability distribution under the target policy as the nominator and the probability distribution under the behaviour policy as the denominator. However, looking at equation 6, the inverse of $\zeta^*(s, a)$ is needed to calculate the KL divergence. Therefore, the equation to calculate the state-visitation distribution is defined by:

$$D_{KL}(P|Q) = \sum_{(s,a)} P(s, a) \log \frac{1}{\zeta(s, a)} \tag{7}$$

4.1.2 Cumulative Reward

The cumulative reward is the normalised expected reward discounted over time. However, instead of the time approaching infinity, the summation is bound by the length of the trajectory, which is sampled from the dataset. The equation is given by equation 5 and is estimated by the DICE estimators when the datasets are run. Combined with the fact that the DICE estimators reduce their variance by using the distribution correction ratios to correct for differences between the behaviour and target policy, the cumulative reward is chosen as the parameter of interest to reflect the target policy performance.

4.1.3 Mean Squared Error (MSE)

The mean squared error (MSE) measures the difference between the actual and estimated values and is selected to quantify the difference between the empirical performance and the estimated performance of the target policy. The MSE is defined as:

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2 \tag{8}$$

- n is the number of values used to calculate the MSE;
- Y_i represents the empirical performance of the target policy, which is calculated by iterating over the episodes of the dataset and calculating the reward obtained in each step while taking the discount factor into account;
- $\hat{Y}_i$ represents the estimated performance of the target policy, which is obtained by running the DICE estimator on the dataset.

It should be mentioned that the empirical performance is only approximated and used as a baseline. By calculating the MSE between the empirical and estimated performance, the effect of the state-visitation mismatch becomes apparent. A lower MSE means the difference

between the empirical and estimated performance is small, suggesting the state-visitation mismatch has a minimal or no effect on the target policy performance while a higher MSE indicates that the state-visitation has some effect on the target policy performance.

4.2 Calculating the Effect of State-visitation Mismatch on the Target Policy Performance

The proposed approach is separated into three algorithms. The first algorithm creates the datasets based on the behaviour policy and calculates the empirical probability distribution needed to calculate the KL divergence. The following algorithm runs a DICE estimator on the datasets and obtains the distribution correction ratios and cumulative reward. The last algorithm combines the values from the previous two algorithms to visualise the results in various plots. The algorithms to create the datasets and run the DICE estimator are enhancements of existing algorithms from the DICE estimator repository[1]. The pseudocode of all the algorithms are located at the end of the subsections and the lines marked in red are the adjustments made to the original algorithm.

4.2.1 Generating the Datasets

Executing the algorithm below creates the datasets needed to run the DICE estimator. The core component of the algorithm is the creation of the on-policy and off-policy datasets based on the behaviour policy in addition to the state-visitation distribution of the behaviour policy.

The on-policy dataset μ is created using the environment env, environment size s, boolean indicating whether to use the default environment map def and α. The environment env, size s and default value def clearly play a role in the creation of the on-policy dataset. The size s and default value def decide the dimensions of the environment. The "Frozen Lake" environment is considered and further details can be found in Section 5.1. The α also plays a important role in dataset creation. In the DICE estimator repository, the ϵ-greedy policy is utilised for the Frozen Lake environment. The ϵ determines the probability of the agent taking an arbitrary action instead of the greedy action, which is the best action considered by the agent in a given state. A higher ϵ means the agent explores more by taking arbitrary actions while a lower ϵ means the agent exploits more by taking the action the agent considers the best. The ϵ is calculated by $0.2 * (1.0 - \alpha)$. Thus, a higher α results in a lower ϵ and vice versa. However, the choice was made to modify the equation to allow the datasets to contain a wider/narrower range of state-action pairs, resulting in different values for the KL divergence and MSE. The modified equation is defined by:

$$\epsilon = 1.0 * (1.0 - \alpha). \tag{9}$$

The off-policy dataset π starts as an empty dataset, which is iteratively updated by sampling trajectories from μ and adding them to π. Due to the nature of how the off-policy dataset is created, the choice was made to consider multiple datasets per α since each dataset may differ in their trajectories due to sampling.

The state-visitation distribution of the behaviour policy is calculated by creating a dictionary d and adding all the state-action pairs of the μ as keys to d. Afterwards, the occurrences

[1]DICE: The DIstribution Correction Estimation Library, https://github.com/google-research/dice_rl.

of each state-action pair p is counted by iterating through the episodes as well as the total number of pairs k. Lastly, each state-action pair occurrence count is divided by k to obtain the state-visitation distribution. This distribution is saved to the repository for later usage.

Algorithm 1 Creating the Datasets

Require: Environment name env, environment size s, boolean indicating to use the default environment map def, list of α indicating how close the behaviour policy is to the target policy A, number of datasets to create for each α value m, number of trajectories to collect n and maximum length of a trajectory l,

```
for each s in S do
    for each α in A do
        for i = 1 to m do
            Create an on-policy dataset μ based on env, s, def and α
            Create an off-policy dataset π based on n and l
            Create an empty dictionary d                ▷ The state-visitation distribution
            k = 0                                       ▷ The total number of state-action pairs
            for i = 1 to n do
                Sample a batch of trajectories b from μ
                for each trajectory t in b do
                    for each step j in t do
                        Create a state-action pair p_j
                        if p_j not in d then
                            Add p_j as a key to d
                        end if
                        Increment the value of key p_j in d and k
                    end for
                end for
                for each p_j in d do
                    Divide the value of key p_j by k
                end for
                Add the sampled trajectories to π
            end for
        end for
    end for
end for
```

4.2.2 Running the DICE Estimator

After creating all the datasets, the DICE estimator runs over the datasets to estimate the target policy performance $\rho(\pi)$. The DICE estimator of choice is Neural DICE. The choice was made due to its usage of neural networks to iteratively approximate the cumulative reward during training and it is the default estimator of the DICE estimator repository.

A higher number of training steps n results in more accurate and reliable estimations of $\rho(\pi)$. In each training step, the neural network calculates the training step and losses in order to improve the accuracy the off-policy evaluation. Besides the training losses, it also calculates and stores the distribution correction ratio $\zeta(s, a)$. Due to the iterative and

optimising nature of the neural network, the $\zeta(s, a)$ of the last training step is stored in a file to calculate the KL divergence later. Furthermore, the cumulative reward cr is estimated and stored alongside the empirical performance (ground truth). Lastly, the cumulative reward over the training steps is logged and stored in order to observe the convergence of the cumulative.

Algorithm 2 Running the DICE Estimator

Require: Off-policy dataset π, state-visitation distribution d, environment name env, environment size s, boolean indicating to use the default environment map def, list of α indicating how close the behaviour policy is to the target policy A, number of datasets to create for each α value m, number of trajectories to collect n, maximum length of a trajectory l and number of training steps k

```
for each s in S do
    for each α in A do
        for i = 1 to m do
            C = [ ]                ▷ List of reward estimations over time for the convergence
            Retrieve the off-policy dataset π based on env, s, def, α, n and l
            Create an empty copy d' of d          ▷ For the distribution correction ratios
            for step i in k do
                Sample a batch of transitions T from π
                Calculate the training loss and update the neural network based on T
                Calculate the distribution correction ratios ζ
                if i == k − 1 then
                    Update d' with ζ
                end if
                if i == 500 or i == k − 1 then
                    Estimate the cumulative reward c
                    Add c to C
                end if
            end for
        end for
    end for
end for
```

4.2.3 Visualising All the Results

Due to the size of the algorithm, it is located in Appendix A. The algorithm collects all the files that were stored earlier, meaning the state-visitation distribution d, distribution correction ratio ζ, cumulative reward cr and cumulative reward convergence crc.

First, the KL divergence is calculated with d and ζ. For each α, multiple datasets were created, which may lead to different KL divergences. Therefore, the KL divergences per α are averaged and appended to a new list $kl\_divergence\_values$ to use in the plot later on.

Afterwards, cr is used to calculate the MSE. For each dataset per α, cr is split up into the ground truth g and last estimated reward ler, which are appended to their respective list, $ground\_truths$ and $last\_estimated\_rewards$. When all the g's and ler's are appended, the average ground truth tg over all the datasets is calculated. This value is used as the

empirical performance of the target policy Y and the elements of $last\_estimated\_rewards$ are the estimated performance of the target policy $\hat{Y}$. The MSE values mse_values are calculated per dataset per α and stored in a nested list. A list is created for each α and contains the MSE values for each individual dataset with the same α.

Thereafter, the average cumulative reward convergence is calculated. For each dataset per α, the crc is retrieved and appended to a list. Each crc is a list of the cumulative reward at time step t where t goes from 0 to the maximum length of a trajectory l. To calculate the average cumulative reward convergence, all the crc's are averaged per t.

Lastly, the $kl\_divergence\_values$ and mse_values mentioned earlier are extracted and used to plot the KL divergence against the MSE in a double box plot. Each box plot corresponds to a specific α's MSE values calculated from the datasets.

5 Evaluation of the Datasets

5.1 Experimental Configuration

This section explains the set-up used to conduct the experiments. First of all, a modified version[2] of the DICE estimator repository is used. Also, as mentioned earlier, the choice of environment is 'Frozen Lake'. Frozen Lake is a discrete environment from OpenAI's Gym API[3] that involves crossing a frozen lake from start to goal without falling into any holes by walking over the frozen lake.

Figure 1: OpenAI's Default Frozen Lake Environment

In addition, the estimator of choice to approximate the target policy's value is the Neural DICE estimator. Besides, the number of training steps for the DICE estimator is set to twenty-five thousand due to the observed convergence towards the average ground truth in Figure 2.
Additionally, the ϵ is determined by the alpha α with the interval [0.0, 1.0]. A higher α means the behaviour and target policy are very similar while the opposite is true for a lower α. Therefore, several values for α are used to create the datasets, specifically the values $(0.0, 0.2, 0.4, 0.6, 0.8, 1.0)$. Furthermore, the values [4, 10, 50] are considered for the environment size. Also, the number of datasets per α is set to five due to the scope of the research and time constraints, resulting in a total of thirty datasets per environment size. Lastly, the number of trajectories sampled per dataset is two hundred and maximum length of a trajectory is set to fifty steps.

[2]Modified dice_rl repository: https://github.com/kevin966/dice_rl
[3]Frozen Lake environment, https://www.gymlibrary.dev/environments/toy_text/frozen_lake/

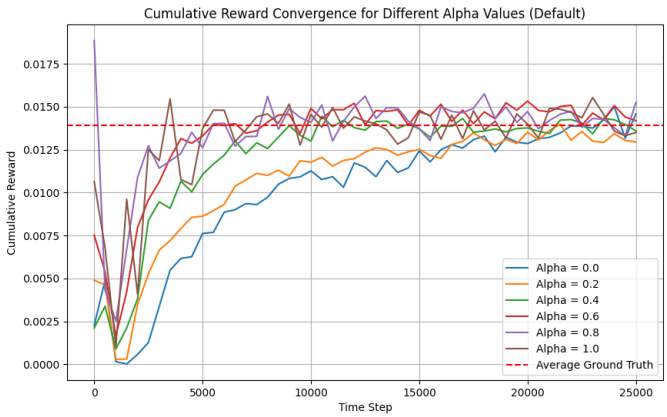

Figure 2: The cumulative reward convergence plotted for different alpha values in the default Frozen Lake environment

5.2 Results

In all four of the plots, it can be observed that the MSE is the highest for the $\alpha = 0.0$. For the default environment size, the box plots for the other α values are very flattened to the point same data points overlap. For the environment of size 4, the MSE values are quite low for the other α values as well. It is the same story for the environment sizes of 10 and 50 but there is a small box plot in the environment size of 10 for $\alpha = 0.4$.

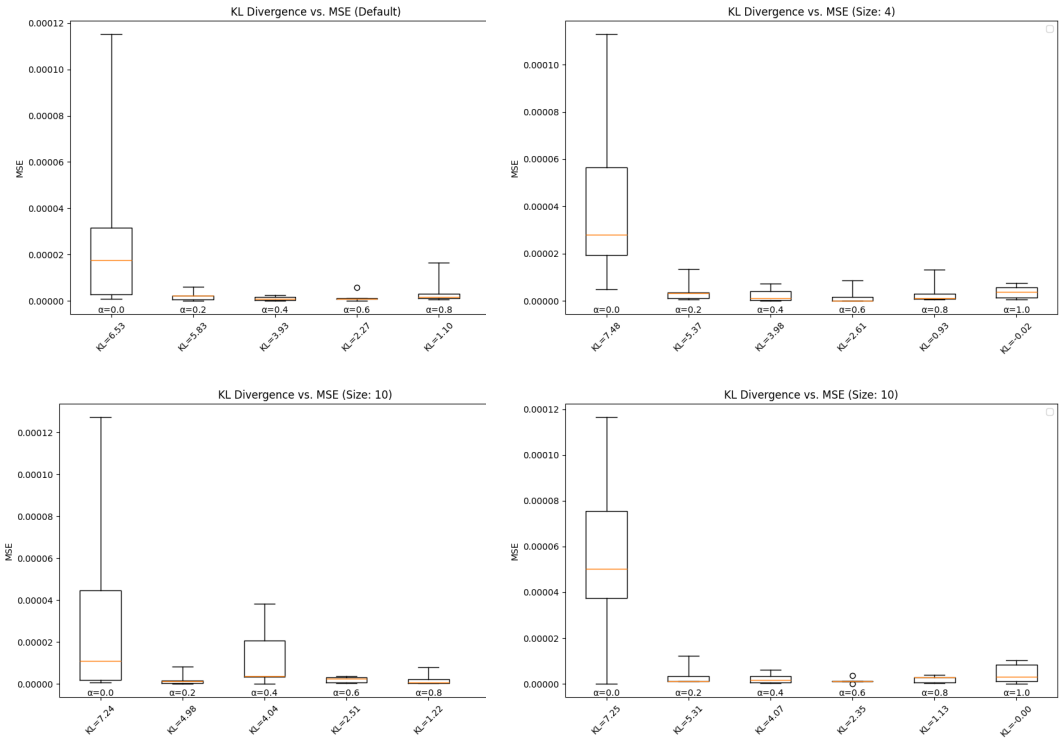

Figure 3: The KL divergence plotted against the MSE for different alpha values and environment sizes

6 Responsible Research

Transparency is ensured by reporting the experimental set-up and elaborating on the methodology. The environments used in this research are simple toy examples free of social biases and discriminations. The repository for the environments is open-source and is continuously developed by OpenAI. All the datasets were created using these environments and are readily available as well. The repository contains detailed documentation including the parameter settings and approach.

7 Discussion

The lack in trends in the plots could be due to the limitations of this experiment since only one estimator and environment were considered. While there were perturbations within the datasets of the Frozen lake environment due to the variations in environment size and α values, these were not enough to show any significant trends in the plots. Furthermore, the simplicity of the Frozen Lake environment might have played a role as well. The size of the environment was taken into account by customising the dimensions of the environment, the task itself remained unchanged. In addition, the lack in variation of the estimator might have had some impact as well since the estimator chosen for this experiment is more suitable

for continuous policies rather than discrete ones.

8 Conclusions & Future Work

8.1 Limitations

There are certain limitations to this experiment. First of all, the KL divergence metric used for the state-visitation mismatch only works on probability distributions of discrete policies. Furthermore, only one environment was used in this experiment, which could influence the results. Besides, only one estimator was used, which specialises in continuous policies.

8.2 Conclusion

Based on the plots, one could see a trend in the KL divergence lowering the more the α approaches 1.0. However, the plots are not significantly enough to draw any conclusions on the affect of the state-visitation mismatch on the target policy performance. There are no clear trends to be observed that could argue that there is an effect visible.

8.3 Future Work

Future work can be done by selecting different or multiple, more complex environments and estimators to minimise the bias and influence it can have on the results. Furthermore, different metrics for the state-visitation mismatch and the effect of the state-visitation mismatch on the target policy performance could be considered and looked into.

A Algorithm

Algorithm 3 Visualising the Results

Require: State-visitation distribution d, distribution correction ratio ζ, cumulative reward cr, cumulative reward convergence crc environment name env, environment size s, boolean indicating to use the default environment map def, list of α indicating how close the behaviour policy is to the target policy A, number of datasets to create for each α value m, number of trajectories to collect n, maximum length of a trajectory l and number of training steps k

for each s in S **do**
    $kl\_divergence\_values = [\,]$
    for each α in A **do**
        $kl\_divergence\_values\_per\_alpha = [\,]$
        for $i = 1$ to m **do**
            Retrieve the d and ζ based on env, s, def, α, n and l
            Calculate the KL divergence kl based on d and ζ
            Append kl to $kl\_divergence\_values\_per\_alpha$
        end for
        Append the alpha and average KL divergence per alpha $kl\_divergence\_values$
    end for
end for
for each s in S **do**
    $mse\_values = [\,]$
    $ground\_truths = [\,]$
    $last\_estimated\_rewards = [\,]$
    for each α in A **do**
        for $i = 1$ to m **do**
            Retrieve the cr based on env, s, def, α, n and l
            Append the ground truth g to $ground\_truths$
            Append the last estimated reward ler to $last\_estimated\_rewards$
        end for
    end for
    Calculate the average ground truth tg based on $ground\_truths$
    for each α in A **do**
        for $i = 1$ to m **do**
            Calculate the MSE mse based on the tg and $last\_estimated\_rewards$
            Append the α and mse to mse_values
        end for
    end for
end for
for each s in S **do**
    $reward\_convergences = [\,]$
    for each α in A **do**
        $reward\_over\_time\_peralpha = [\,]$
        for $i = 1$ to m **do**
            Retrieve the crc based on env, s, def, α, n and l
            Append crc to $reward\_over\_time\_peralpha$
        end for
        Append the average cumulative reward over time per alpha to $reward\_convergences$
    end for
    Plot the cumulative reward convergences
end for
for each s in S **do**
    Plot the KL divergences against the MSE
end for

References

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

[2] Nathan Kallus and Masatoshi Uehara. Efficiently breaking the curse of horizon in off-policy evaluation with double reinforcement learning, 2023.

[3] MUHAMMAD KHAN, Somia Mehak, Hafiz Arslan Ramzan, W Yasir, SHAGUFTA ANWAR, and MUHAMMAD MAJEED. Quantitative studies of deep reinforcement learning in gaming, robotics, and real-world control systems. *Bulletin of Business and Economics (BBE)*, 12:389–395, 08 2023.

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

[5] Ofir Nachum, Bo Dai, Ilya Kostrikov, Yinlam Chow, Lihong Li, and Dale Schuurmans. Algaedice: Policy gradient from arbitrary experience, 2019.

[6] Ruiyi Zhang, Bo Dai, Lihong Li, and Dale Schuurmans. Gendice: Generalized offline estimation of stationary values, 2020.

[7] Shangtong Zhang, Bo Liu, and Shimon Whiteson. Gradientdice: Rethinking generalized offline estimation of stationary values, 2020.

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

[9] Ofir Nachum and Bo Dai. Reinforcement learning via fenchel-rockafellar duality, 2020.

[10] Shangding Gu, Long Yang, Yali Du, Guang Chen, Florian Walter, Jun Wang, and Alois Knoll. A review of safe reinforcement learning: Methods, theory and applications, 2024.

[11] Miroslav DudÃk, Dumitru Erhan, John Langford, and Lihong Li. Doubly robust policy evaluation and optimization. *Statistical Science*, 29(4), November 2014.

[12] Yu-Xiang Wang, Alekh Agarwal, and Miroslav Dudik. Optimal and adaptive off-policy evaluation in contextual bandits, 2017.

[13] Nathan Kallus and Masatoshi Uehara. Double reinforcement learning for efficient off-policy evaluation in markov decision processes, 2020.

[14] Ziyang Tang, Yihao Feng, Lihong Li, Dengyong Zhou, and Qiang Liu. Doubly robust bias reduction in infinite horizon off-policy estimation, 2019.

[15] Riashat Islam, Komal K. Teru, Deepak Sharma, and Joelle Pineau. Off-policy policy gradient algorithms by constraining the state distribution shift, 2019.

[16] S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951.