Policy-Aware Learning of Transition Models Using a Causal Approach

Anonymous submission

Abstract

Predicting what will happen when a reinforcement learning (RL) agent is deployed to the real world is important to provide safety guarantees about its behaviour. In some cases, the agent's training experience can be significantly different from the deployment experience. Thereby, learning a transition model with the unadjusted training data can lead to poor performance when predicting the agent's behaviour under the optimal policy. To mitigate this issue, we propose a policy matching (PM) algorithm based on the causal Bayesian network factorisation. It adjusts the transition model learning by taking into account the difference between agent's interventions at training and under the optimal policy. Experiments in popular RL environments demonstrate that the PM method improves the transition model performance at deployment when the model misgeneralisation problem is otherwise severe.

1 Introduction

Reinforcement learning (RL) (Sutton and Barto 2018) is a field of artificial intelligence aimed at solving sequential decision-making problems. In off-policy RL (Watkins and Dayan 1992; Mnih et al. 2013; Van Hasselt, Guez, and Silver 2016), agents are trained by taking many suboptimal actions to find the best policy. When deployed, they make only optimal decisions, thereby making their deployment experience different from the training one.

In order to provide guarantees about the agent's safety, one needs to know how the agent will interact with the environment when executing its optimal policy (Thomas and Brunskill 2016). This requires learning a transition model, which predicts the next state given the current action and state. When an agent's deployment data is not available, a standard approach is to fit the model using its training data. However, this method is naive since it emphasises all transitions equally and does not account for a change in the structure of policy intervention (Pearl 2009). Instead, one can make the transition dynamics learning focus only on the experience which is consistent with the optimal policy.

To illustrate, consider an example when an agent aims to reach a goal from a start position as soon as possible while avoiding the cliff (Figure 1). At training, an off-policy RL agent follows a lot of suboptimal paths, such as the red one. When fitting a perfect world model is difficult, the dynamics model learning will mostly focus on minimising prediction error for suboptimal transitions as they are encountered frequently (Ma et al. 2023). This can happen at the expense of having high accuracy for optimal path transitions (the blue line), which leads to unreliable predictions at the agent's deployment.

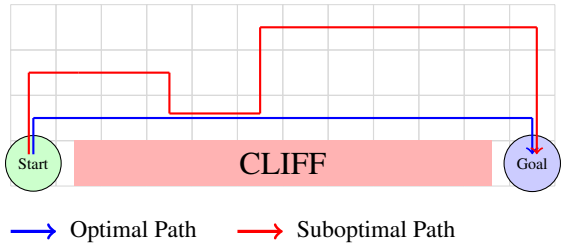

Figure 1: RL Cliff-Walking environment with optimal (blue) and suboptimal (red) paths.

The key idea of our proposed method called "policy matching" (**PM**) is to make the best use of the agent's training data to optimise the model's performance under the optimal policy. Our approach shares similarities with the previous works (Ma et al. 2023) focused on learning transition models for more efficient policy improvement in model-based RL. In contrast to Ma et al. (2023), we do not assume access to the data that comes from the optimal policy interventions. This reflects real-life scenarios in healthcare (Jiang and Li 2015; Thomas and Brunskill 2016; Rebello et al. 2023), where the off-policy evaluation of a treatment assignment can be too risky or costly. We resolve the issue of unavailable optimal policy data by adjusting the training data based on the causal Bayesian network factorisation (Pearl 2009; Koller and Friedman 2009). Our method adapts the loss function at the transition model learning so that, in expectation, this loss becomes more similar to the loss at deployment.

First, we show evidence of misgeneralisation in standard RL environments when off-policy methods are used to train agents. This is demonstrated by the reduced accuracy of the transition model when used at deployment. Secondly, we provide theoretical foundations for why policy matching helps mitigate the distribution mismatch between train-

ing and deployment data. Finally, we showcase that PM improves the transition prediction at deployment in the Lunar Lander environment, in which the misgeneralisation of a naive model is especially high. The method performs similarly to the naive approach in the Cart Pole environment due to a less severe issue of model misgeneralisation.

2 Preliminaries

In this section, we introduce concepts from reinforcement learning and causality which are important for our work.

Reinforcement learning

Fundamentals. Consider a Markov Decision Process (MDP) defined as a tuple ($\mathcal{S}$, $\mathcal{A}$, $\mathcal{R}$, $\mathbf{P}$, $\mathbf{R}$, ρ, γ), where S_t is the state at time t, $\mathcal{S}$ is the state space, A_t is the action at time t, $\mathcal{A}$ is the action space, R_t is the reward signal at time t, $\mathcal{R}$ is the reward space, $\mathbf{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition probability function, $\mathbf{R} : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$ is the expected reward function, $\rho : \mathcal{S} \to [0, 1]$ is the initial state distribution and $\gamma \in (0, 1]$ is the discount rate. By formalising a sequential decision-making problem as an MDP, one can apply reinforcement learning algorithms to find an optimal policy $\pi(A_t|S_t)$, where $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is the probability of taking an action in a given state (Sutton and Barto 2018). RL methods are aimed at finding a policy π that maximises the action-state value function:

$$Q_\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} \gamma^t R_{t+1} | S_t = s, A_t = a \right], \quad (1)$$

which is an expected sum of discounted rewards when an agent starts in state s, takes an actions a and follows the policy π thereafter. This optimisation often requires two steps: policy evaluation and policy improvement. At policy evaluation, a policy π is evaluated by estimating $Q_\pi(s, a)$. Then, if there are any states in which a policy can be changed to increase $Q_\pi(s, a)$, those changes are applied to produce a new improved policy $\tilde{\pi}$. The two steps are repeated iteratively until convergence and the optimal policy π^* is returned. The optimal policy is defined as:

$$\pi^*(a_t \mid s_t) = \begin{cases} 1, & \text{if } a_t = \arg\max_{a'} Q_{\pi^*}(s_t, a'), \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

Off-policy reinforcement learning. Off-policy RL methods rely on the idea of learning an optimal (target) policy $\pi^*(A_t|S_t)$ based on interactions with the environment using another, behaviour policy $b_t(A_t|S_t)$, which is defined as

$$b_t(a_t \mid s_t) = \begin{cases} 1 - \varepsilon_t + \dfrac{\varepsilon_t}{|\mathcal{A}|}, & \text{if } a_t = \arg\max_{a'} Q_t(s_t, a'), \\ \dfrac{\varepsilon_t}{|\mathcal{A}|}, & \text{otherwise,} \end{cases} \quad (3)$$

where $Q_t(s, a')$ is the agent's estimate of the action-state value function at time t, $\varepsilon_t \in [0, 1]$ (exploration rate) is the probability of taking a random action at time t and $|\mathcal{A}|$ is the number of possible actions. In word s, the behaviour policy allows us to test with probability ε_t whether seemingly suboptimal actions are actually optimal. In practice, ε_t is often set to a high value in the beginning of training and decays towards some value ε_{min} as the agent converges to an optimal policy.

The fundamental off-policy method is Q-learning (Watkins and Dayan 1992), which is often used for MDPs with small and discrete action and state spaces. When those spaces are large or the state space is continuous, one needs to use approximation methods such as deep learning (LeCun, Bengio, and Hinton 2015). For example, a Deep Q-network (DQN) method represents the action-state value function as a neural network $M : \mathcal{S} \to \mathbb{R}^{|\mathcal{A}|}$ (Mnih et al. 2013). A Double Deep Q-network (DDQN) algorithm enhances DQN by improving learning stability and increasing accuracy of $Q_\pi(s, a)$ estimate (Van Hasselt, Guez, and Silver 2016) Finally, a Deep Deterministic Policy Gradient (DDPG) method allows to perform off-policy learning when both state and action spaces are continuous (Lillicrap et al. 2015).

In contrast to on-policy RL, where the behaviour and target policy are the same, off-policy methods allow $b_t(A_t|S_t)$ to perform suboptimal actions with non-zero probability at all times (Watkins and Dayan 1992). Thereby, the agent's training data D_{train} can contain many suboptimal actions and may not be a good representation of the experience during optimal policy deployment D_{deploy}. The distance between the training distribution $\mathcal{D}_{\text{train}}$ and the deployment distribution $\mathcal{D}_{\text{deploy}}$ can be too large, making the naive use of D_{train} suboptimal when training an MDP transition model.

Causality

The framework of causality provides concepts such as a causal graph and Bayesian factorisation which can be used to analyse the effects of interventions on the data-generating process (Pearl 2009). A causal graph (also called causal Bayesian network) is a directed acyclic graph (DAG) $G = (\mathcal{V}, \mathcal{E})$, where each node $V_i \in \mathcal{V}$ represents a random variable, and each directed edge $(V_i \to V_j) \in \mathcal{E}$ indicates a direct causal effect from variable V_i to variable V_j. For each random variable V_j, one can identify a set of its causal parents $\mathbf{PA}_j$, which is defined as the set of random variables such that $(V_i \to V_j) \in \mathcal{E}\ \forall V_i \in \mathbf{PA}_j$. Figure 2 shows how an MDP transition model can be represented as a causal graph, in which the current state S_t and action A_t are considered exogenous and they are causal parents of the next state S_{t+1}.

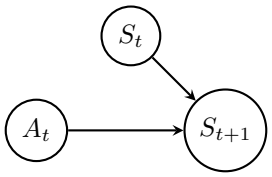

Figure 2: Causal graph of the MDP transition model, where S_t and A_t are assumed to be exogenous.

Knowing the causal parents of each random variable allows us to represent a joint distribution of $V_1, ..., V_n$ using

the causal Bayesian network factorisation:

$$\mathbb{P}(V_1, ..., V_n) = \prod_{i=1}^{n} \mathbb{P}(V_i \mid \mathbf{PA}_i), \tag{4}$$

which typically provides a more compact representation. When a reinforcement learning agent interacts with the environment using a policy $\pi^{(i)}$ (Figure 3), the transition tuple (S_t, A_t, S_{t+1}) follows a distribution $\mathcal{D}_{(i)}$. In general, interventions induced by two distinct policies $\pi^{(i)}$ and $\pi^{(j)}$ imply a mismatch between $\mathcal{D}_{(i)}$ and $\mathcal{D}_{(j)}$. When a transition model is learned using the data generated by a behaviour policy b_t, the model's predictions of deployment experience can be very inaccurate in off-policy RL because b_t is different from the deployed policy π^*.

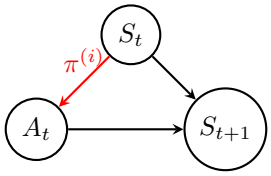

Figure 3: Causal graph of the MDP transition model under the intervention induced by the policy π^i. Action A_t is now a function of the current state S_t.

3 Policy Matching

In this section, we describe the policy matching (PM) algorithm used to estimate the transition model of a Markov Decision Process. This method optimises the model's performance in the data under the intervention of an optimal policy π^*. To justify our proposed algorithm, we use the theory of change of measure and causal Bayesian network factorisation.

Algorithm Outline

Assume we have access to the transition training data of an off-policy reinforcement learning agent $D_{\text{train}} = \langle s_0, a_0, s_1\rangle, \langle s_1, a_1, s_2\rangle, ..., \langle s_{T-1}, a_{T-1}, s_T\rangle$. This data was generated by the agent's interventions in the MDP using a behaviour policy b_t with the exploration rate $\varepsilon_t \in [\varepsilon_{\text{min}}, 1]$. We assume $\varepsilon_{\text{min}} > 0$, so the agent never takes an optimal action with probability 1 during training. While we do not have access to b_t, we know the optimal policy π^* and its corresponding action-state function $Q_{\pi^*}(s, a)$. The goal is to predict the agent's transitions when it executes π^*. We are not allowed to deploy the optimal policy to generate additional data, e.g. for safety or financial reasons.

In order to predict transitions, one needs to estimate a transition model $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ using a supervised learning algorithm by minimising an empirical loss function $\mathcal{L} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$. We propose to use the policy matching (PM) procedure to achieve this, which implies calculating the empirical loss function $\mathcal{L}$ as:

$$\mathcal{L}(f, D_{\text{train}}) = \frac{1}{T} \sum_{t=0}^{T-1} w_t L(\hat{s}_{t+1}, s_{t+1}), \tag{5}$$

where $w_t = \mathbb{1}\{\arg\max_{a'} \pi^*(a'|s_t) = a_t\}$ is the policy match indicator, $\hat{s}_{t+1} = f(s_t, a_t)$ is the next state prediction and $L(\hat{s}_{t+1}, s_{t+1}) \geq 0$ is the sample loss. The key idea is to ignore the sample loss for the state-action pair $\langle s_t, a_t\rangle$ in $\mathcal{L}$ if the action a_t is not consistent with the optimal policy π^*. Finally, the transition model f is learned by minimising $\mathcal{L}$ in the training data D_{train}:

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \mathcal{L}(f, D_{\text{train}}), \tag{6}$$

where $\mathcal{F}$ is the space of transition models and $\hat{f}$ is the learned transition model. The proposed method is summarised in Algorithm 1.

Algorithm 1: Transition Model Learning via Policy Matching (PM)

Input: Agent's training data $D_{\text{train}} = \langle s_0, a_0, s_1\rangle, \langle s_1, a_1, s_2\rangle, ..., \langle s_{T-1}, a_{T-1}, s_T\rangle$ and optimal policy π^*

1: Select a transition model $f : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ which outputs the next state prediction $\hat{s}_{t+1}$
2: Choose a sample prediction loss $L(\hat{s}_{t+1}, s_{t+1}) \geq 0$
3: **for** $t = 0, ..., T-1$ **do**
4: $\quad w_t = \mathbb{1}\{\arg\max_{a'} \pi^*(a'|s_t) = a_t\}$ **(PM)**
5: **end for**
6: Define the empirical loss function as: $\mathcal{L}(f, D_{\text{train}}) = \frac{1}{T} \sum_{t=0}^{T-1} w_t L(\hat{s}_{t+1}, s_{t+1})$
7: Learn the transition model f by minimising $\mathcal{L}$ in D_{train}:

$$\hat{f} = \arg\min_{f \in \mathcal{F}} \mathcal{L}(f, D_{\text{train}})$$

8: **return** $\hat{f}$

Theoretical Foundations

When learning a transition model, we have the following optimisation problem:

$$\min_f \mathbb{E}_{\mathcal{D}_{\text{deploy}}} \left[L(f(S_t, A_t), S_{t+1}) \right], \tag{7}$$

where $L : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ is the loss function and $\mathcal{D}_{\text{deploy}}$ is the data distribution at optimal policy deployment. In practice, the expected loss function is approximated by the empirical loss function $\mathcal{L}$ as shown in the policy matching algorithm. When there is no access to data sampled from $\mathcal{D}_{\text{deploy}}$, we can only use the data from the training distribution $\mathcal{D}_{\text{train}}$ for the model learning. Using the Radon-Nykodym theorem (Tao 2011), we can change the measure of the expectation in Equation 7 and represent the optimisation problem as follows:

$$\min_f \mathbb{E}_{\mathcal{D}_{\text{train}}} \left[\frac{\mathbb{P}_{\text{deploy}}(S_t, A_t, S_{t+1})}{\mathbb{P}_{\text{train}}(S_t, A_t, S_{t+1})} L(f(S_t, A_t), S_{t+1}) \right] \tag{8}$$

The causal Bayesian network factorisation implies that the joint distribution can be decomposed as $\mathbb{P}(S_{t+1}, A_t, S_t) = \mathbb{P}(S_t)\pi(A_t|S_t)\mathbb{P}(S_{t+1}|A_t, S_t)$. This can be applied to the joint distributions inside the expectation

since the MDP transition data can be represented as a causal graph (Figure 4). Because the underlying transition dynamics at training and deployment $\mathbb{P}_{\text{train}}(S_{t+1} \mid A_t, S_t)$ and $\mathbb{P}_{\text{deploy}}(S_{t+1} \mid A_t, S_t)$ are the same, we get a simplified problem in Equation 9.

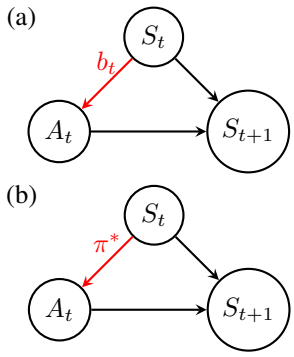

Figure 4: MDP transition models represented as causal graphs: (a) MDP transition model under the intervention of the behaviour policy $b_t(A_t|S_t)$ and (b) transition model under the intervention of the optimal policy $\pi^*(A_t|S_t)$.

$$\min_f \mathbb{E}_{\mathcal{D}_{\text{train}}} \left[\frac{\pi^*(A_t|S_t)}{b_t(A_t|S_t)} \frac{\mathbb{P}_{\text{deploy}}(S_t)}{\mathbb{P}_{\text{train}}(S_t)} L(f(S_t, A_t), S_{t+1}) \right] \tag{9}$$

This equation provides us with the method to reweight each transition $\langle S_t, A_t, S_{t+1} \rangle$ in the training loss function. However, we need to estimate the unknown marginal distributions of the state under training and deployment distribution ($\mathbb{P}_{\text{train}}(S_t)$ and $\mathbb{P}_{\text{deploy}}(S_t)$), which is typically hard. However, if the term inside the expectation of Equation 7 is close to being independent of S_t, we can replace the objective function with the expected loss conditional on S_t. As a result, we can remove the ratio of marginal distributions $\frac{\mathbb{P}_{\text{deploy}}(S_t)}{\mathbb{P}_{\text{train}}(S_t)}$ and get the following problem:

$$\min_f \mathbb{E}_{\mathcal{D}_{\text{train}}} \left[\frac{\pi^*(A_t|S_t)}{b_t(A_t|S_t)} L(f(S_t, A_t), S_{t+1}) \mid S_t \right] \tag{10}$$

Since we assume access to $\pi^*(A_t|S_t)$, the only unknown in Equation 10 is the behaviour policy $b_t(A_t|S_t)$, which can be estimated from observed training data D_{train}. Under specific conditions, we can avoid estimating $b_t(A_t|S_t)$ to solve the optimisation problem. The justification for that is provided in Theorem 3.1 and Corollary 3.1.

Theorem 3.1 (Upper Bound of the Expected Loss)**.** *Denote the total number of discrete actions available in the environment as $|\mathcal{A}|$. Also, define a minimum exploration rate at training as $\varepsilon_{min} > 0$, such that $b_t(A_t|S_t) \geq \frac{\varepsilon_{min}}{|\mathcal{A}|}$. Then:*

$$\mathbb{E}_{\mathcal{D}_{train}} \left[\frac{\pi^*(A_t|S_t)}{b_t(A_t|S_t)} L(f(S_t, A_t), S_{t+1}) \mid S_t \right] \leq \frac{|\mathcal{A}|}{\varepsilon_{min}} \mathbb{E}_{\mathcal{D}_{train}} \left[\pi^*(A_t|S_t) L(f(S_t, A_t), S_{t+1}) \mid S_t \right] \tag{11}$$

In off-policy reinforcement learning, it is common that $\varepsilon_{\text{min}} > 0$ since the behaviour policy does not need to converge to an optimal one.

Corollary 3.1 (Problem Equivalence)**.** *Given the conditions of Theorem 3.1, solving the following optimisation problem:*

$$\min_f \mathbb{E}_{\mathcal{D}_{train}} \left[\pi^*(A_t|S_t) L(f(S_t, A_t), S_{t+1}) \mid S_t \right] \tag{12}$$

is equivalent to minimising the upper bound of the objective in Equation 10.

Ultimately, the proposed policy matching procedure (Algorithm 1) solves the following optimisation problem:

$$\min_f \mathbb{E}_{\mathcal{D}_{\text{train}}} \left[\pi^*(A_t|S_t) L(f(S_t, A_t), S_{t+1}) \right], \tag{13}$$

for which solution can be close to the solution in Equation 12. Since moving from Equation 9 to Equation 13 requires adding and removing the conditioning on S_t, the effect of this on the policy matching accuracy remains to be studied. Importantly, our approximations lead to a PM algorithm which does not require estimating $b_t(A_t|S_t)$ and the marginal distributions of S_t.

4 Experimental Evaluation

In order to evaluate the policy matching algorithm, we perform transition model learning in popular RL environments.

RL environments and algorithms. We use two RL environments with continuous state space and discrete actions from the Gymnasium package (Towers et al. 2024): Lunar Lander and Cart Pole. In both environments, we use default settings. We use a DDQN method (Van Hasselt, Guez, and Silver 2016) over 600 episodes. We train DDQN agents using a Q-network $Q : \mathcal{S} \rightarrow \mathcal{A}$ with two 128-node hidden layers. At the beginning of training, ε_t is 1 and it is decayed exponentially. We perform experiments with different random seeds and we iterate over 6 different ϵ_{min} (0.01, 0.11, 0.21, 0.31, 0.41 and 0.51). The summary of RL settings can be found in Table 1. We store the RL training and deployment transition data only if its learned policy is optimal. We consider a policy optimal if it generates the average episodic reward of at least 200 and 195 over 100 episodes in Lunar Lander and Cart Pole respectively.

Transition model. In order to estimate a transition model f, we use one of the tree-based methods as they are the most suitable for tabular transition data in our RL experiments (Grinsztajn, Oyallon, and Varoquaux 2022; Shwartz-Ziv and Armon 2022). We fit a random forest (RF) model (Ho 2002) since it is less prone to overfitting than gradient-boosting methods (Chen and Guestrin 2016; Ke et al. 2017) due to its lower variance. We use simple hyperparameter settings (100 trees) since this proves to be enough to fit the training data well.

Metrics. The coefficient of determination (Wright 1921) is a standard regression metric and is used to measure the accuracy of a transition model:

$$R^2 = 1 - \frac{\sum_{t=0}^{T-1} (y_t - \hat{y}_t)^2}{\sum_{t=0}^{T-1} (y_t - \bar{y})^2} \tag{14}$$

Its value is 1 when the model fits data perfectly and can be arbitrarily low, The R^2 is zero if the prediction performance is the same as if we used the average of the target ($\bar{y}$). Maximising R^2 is equivalent to minimising the mean squared error (MSE) $\frac{1}{T}\sum_{t=0}^{T-1}(y_t - \hat{y}_t)^2$, but we prefer to report R^2 because it scales MSE by the variance of the target. We also measure the transition model misgeneralisation score (MS) as:

$$\text{MS} = R^2(\hat{f}, D_{\text{train}}) - R^2(\hat{f}, D_{\text{deploy}}), \qquad (15)$$

where $\hat{f}$ is trained using only D_{train}. This metric quantifies how difficult it is for the model $\hat{f}$ to generalise to deployment data. Further, we denote $R^2(\hat{f}, D_{\text{train}})$ as R^2_{train} and $R^2(\hat{f}, D_{\text{deploy}})$ as R^2_{deploy} to be concise.

Results

After training DDQN agents, we use their training data to fit transition models using a naive approach (no data adjustment) and the proposed policy matching method. We remove experiments in which R^2 in deployment data is negative, which implies extremely high misgeneralisation. This happens in less than 1% and 3% experiments in Lunar Lander and Cart Pole respectively. After this filtering, we have 26 experiments per ε_{min} in Lunar Lander and 58 experiments per ε_{min} in Cart Pole. For both RL environments, we estimate the severity of misgeneralisation by calculating the MS based on the naive transition model learning. The higher it is, the worse the model generalises to the deployment data. Then, we assess the quality of the policy matching method by comparing R^2_{deploy} of the naive and PM transition models.

The high percentage of large misgeneralisation scores (top chart in Figure 5) of a naive model shows that the misgeneralisation problem is significant in the Lunar Lander. This motivates the need to adapt training data to match the deployment data using our policy matching algorithm. Indeed, the bottom chart in Figure 5 demonstrates a consistent outperformance of the proposed PM method when compared to the naive approach. The distribution of R^2_{deploy} is more concentrated in the region closer to 1 when using model training via policy matching. Hence, we conclude that our approach improves deployment experience prediction accuracy in this environment. The results for Cart Pole suggest less clear evidence of misgeneralisation (Figure 6) as the majority of MC scores are below 0.1. As a result, our PM approach does not provide improvements and performs similarly to the naive method.

Sensitivity to ε_{min}. We also study the sensitivity of our results to a minimum exploration rate. Interestingly, we notice that higher ε_{min} tends to imply a higher misgeneralisation score of a naive model (Figure 7). Since the average exploration rate is linearly dependent on the minimum exploration rate (see Figure 10), this may imply that the more the agent takes random actions at training, the more different the deployment experience is. However, Figure 9 contradicts this interpretation as there is no clear relationship between the misgeneralisation score and the percentage of cases when an optimal action was taken. Finally, we observe that higher

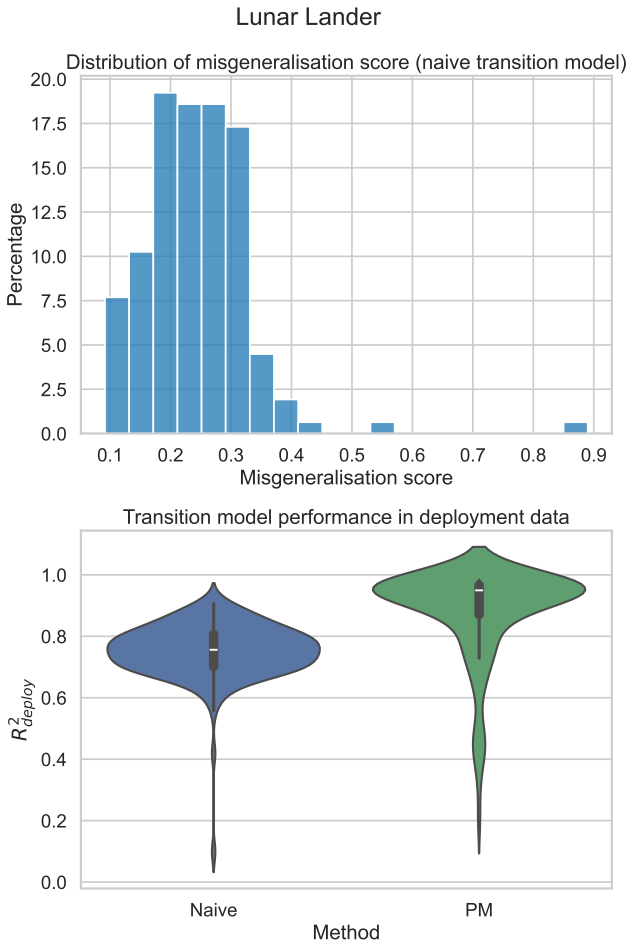

Figure 5: Lunar Lander experiment results (26 experiments with different random seeds per ε_{min}).

ε_{min} usually implies worse model performance at deployment (Figure 9). While the patterns in Figures 7 and 9 are evident, we leave their more in-depth analysis and interpretation for future work.

5 Related Work

Several works consider the problem of policy-aware transition dynamics training to improve the efficiency of model-based RL. Ma et al. (2023) propose a way to estimate occupancy rates of transition experiences to emphasise those which are more in line with a policy of interest. Janner et al. (2019) propose Model-Based Policy Optimization (MBPO), which integrates policy-awareness into model learning. The model is trained to prioritise accuracy in regions visited by the current policy, thus improving sample efficiency and performance by focusing on policy-relevant dynamics. Feinberg et al. (2018) present Model-Based Value Expansion (MVE), a method that enhances model-free RL by incorporating short model-based rollouts into value estimation. The transition model is trained in a policy-aware manner, focusing on accurately predicting transitions along the trajecto-

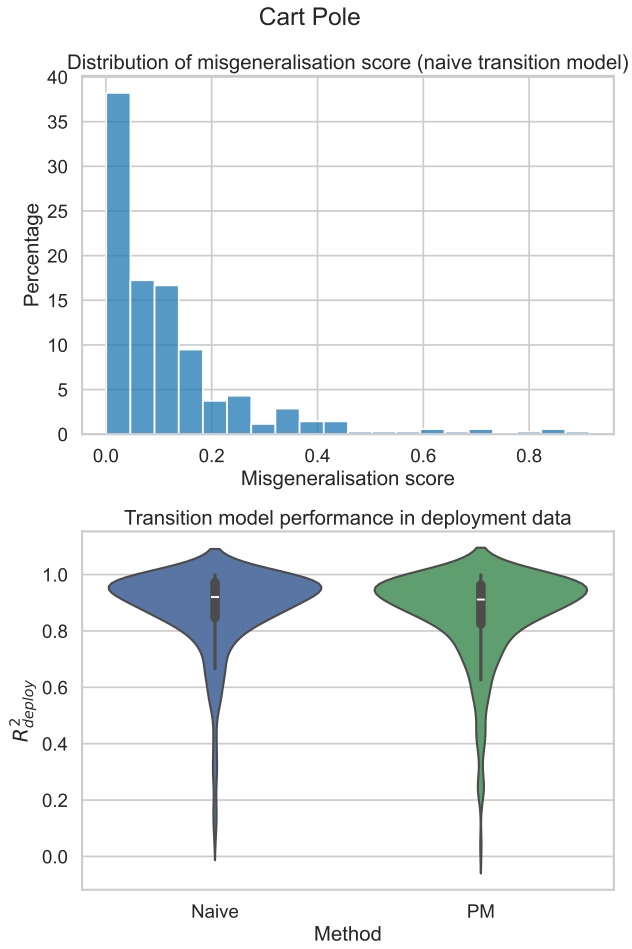

Figure 6: Cart Pole experiment results (58 experiments with different random seeds per ε_{min}).

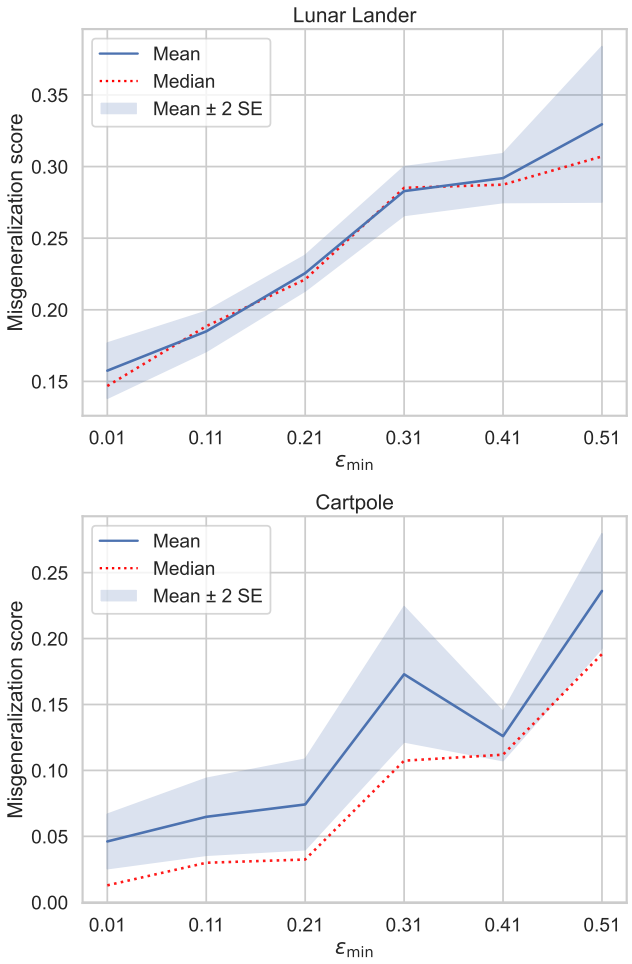

Figure 7: Misgeneralisation score of a naive transition model in Lunar Lander and Cart Pole RL environments. For each ε_{min}, we calculate mean, median and standard error across multiple random seeds.

ries induced by the policy to improve value estimates and policy performance. Eysenbach et al. (2024) address the objective mismatch between model accuracy and policy performance by jointly training the world model and policy. Wang, Wongkamjan, and Huang (2022) introduce Policy-adapted Dynamics Model Learning (PDML) for model-based reinforcement learning. This method dynamically adjusts the transition model to the changes in the state-action distribution of the current policy. The results show significant improvements in terms of sample efficiency and performance in continuous control tasks.

There is also a relation between our work and off-policy evaluation (OPE). This field is concerned with predicting the off-policy agent's value function under the optimal policy (Jiang and Li 2015; Thomas and Brunskill 2016; Rebello et al. 2023). The OPE goal is to produce an estimator $\hat{v}(D)$ of the value function under the evaluation policy $v(\pi_e)$ that has a low expected squared error: $\mathbb{E}\left[(\hat{v}(D) - v(\pi_e))^2\right]$. Although the OPE methods may include transition model learning, providing highly accurate transition predictions is not the main concern in off-policy evaluation.

6 Conclusion

We propose a policy matching (PM) method to adapt a transition model training to changes in the agent's policy. In our experiments, we demonstrate the presence of transition model misgeneralisation in the Lunar Lander RL environment, while this problem is less evident in the Cart Pole. Then, we show that our proposed PM method helps improve the prediction performance of a transition model at deployment when the misgeneralisation problem is severe (Lunar Lander). Since the issue is less pronounced in Cart Pole, our method is on par with the naive approach in that environment. We also find that model performance at deployment and the degree of misgeneralisation are correlated with the minimum exploration rate during the agent's training.

Our work has several limitations that open avenues for future research. Firstly, our theoretical results are limited to the conditional loss, while the underlying optimisation implies using the unconditional one. Secondly, we leave the relationship between ε_{min} and the misgeneralisation score un-

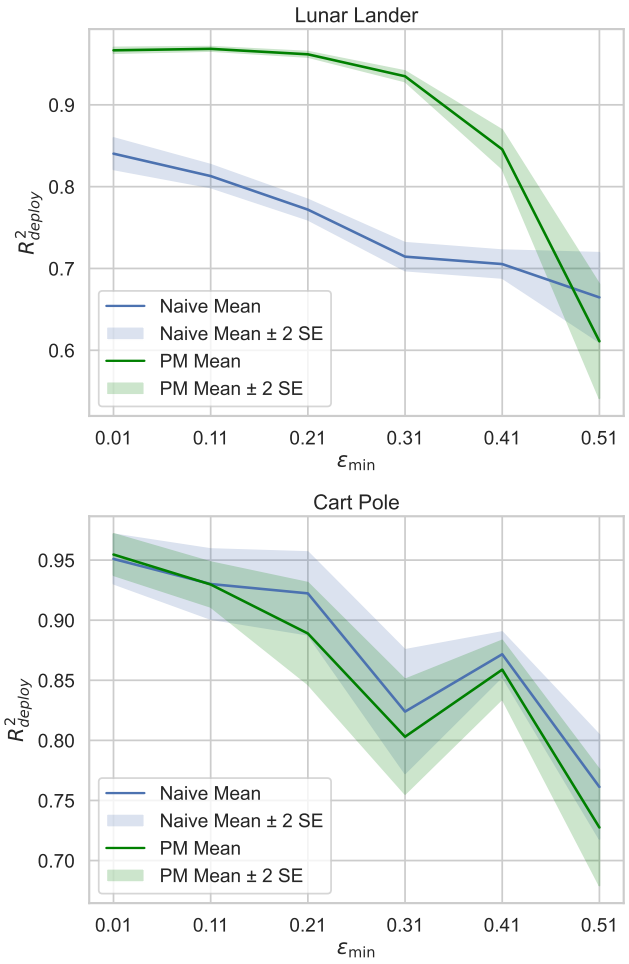

Figure 8: Transition model performance at deployment in relationship to the minimum exploration rate ε_{min}.

explained. Furthermore, our method assumes access to the optimal policy π^*, which may not always be available in practice. Future papers could explore alternative approaches that utilise the changing structure of the exploration rate over time, such as placing greater emphasis on later training transitions due to exploration rate decay. Finally, it should be possible to improve the method's performance if one can estimate the behaviour policy and marginal distribution of the state accurately. We leave exploring this direction for future work.

A RL Settings

Notations:

- ε_{max}: maximum exploration rate ϵ_t
- γ: discount rate
- α: learning rate
- $|B|$: batch size
- τ: the update rate of the target network

	Lunar Lander	**Cart Pole**
Algorithm	DDQN	DDQN
ε_{max}	1	1
γ	0.99	0.99
α	1e-4	1e-4
$\lvert B \rvert$	128	128
τ	0.005	0.005

Table 1: Settings of RL agents.

B Exploration Rate Analysis

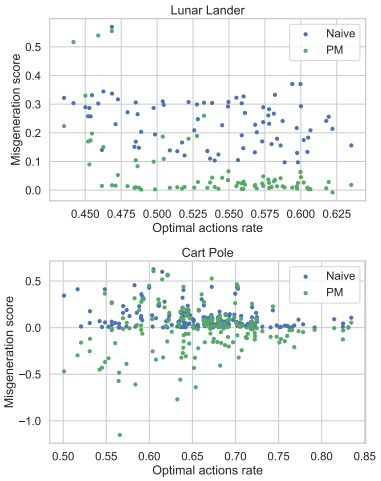

Figure 9: Misgeneralisation score against the optimal action rate in Lunar Lander and Cart Pole.

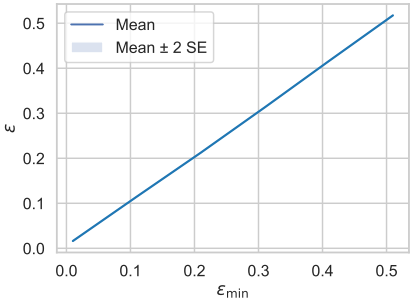

Figure 10: Mean exploration rate per minimum exploration rate in Lunar Lander (averaged across random seeds).

References

Chen, T.; and Guestrin, C. 2016. XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. ACM.

Eysenbach, B.; Khazatsky, A.; Levine, S.; and Salakhutdinov, R. 2024. Mismatched no more: joint model-policy optimization for model-based RL. In *Proceedings of the 36th*

International Conference on Neural Information Processing Systems, NIPS '22. Red Hook, NY, USA: Curran Associates Inc. ISBN 9781713871088.

Feinberg, V.; Wan, A.; Stoica, I.; Jordan, M. I.; Gonzalez, J. E.; and Levine, S. 2018. Model-Based Value Expansion for Efficient Model-Free Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, 3069–3077.

Grinsztajn, L.; Oyallon, E.; and Varoquaux, G. 2022. Why do tree-based models still outperform deep learning on typical tabular data? *Neural Inf Process Syst*.

Ho, T. K. 2002. Random decision forests. In *Proceedings of 3rd International Conference on Document Analysis and Recognition*, volume 1, 278–282 vol.1. IEEE Comput. Soc. Press.

Janner, M.; Fu, J.; Zhang, M.; and Levine, S. 2019. When to Trust Your Model: Model-Based Policy Optimization. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, 12519–12530. Curran Associates, Inc.

Jiang, N.; and Li, L. 2015. Doubly robust off-policy value evaluation for reinforcement learning. *arXiv [cs.LG]*, 652–661.

Ke, G.; Meng, Q.; Finley, T.; Wang, T.; Chen, W.; Ma, W.; Ye, Q.; and Liu, T.-Y. 2017. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Advances in Neural Information Processing Systems*, volume 30, 3146–3154.

Koller, D.; and Friedman, N. 2009. *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press. ISBN 0262013193.

LeCun, Y.; Bengio, Y.; and Hinton, G. 2015. Deep learning. *Nature*, 521(7553): 436–444.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *arXiv [cs.LG]*.

Ma, Y. J.; Sivakumar, K.; Yan, J.; Bastani, O.; and Jayaraman, D. 2023. TOM: Learning policy-aware models for model-based reinforcement learning via transition occupancy matching. *arXiv [cs.LG]*, 259–271.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; and Riedmiller, M. 2013. Playing Atari with deep reinforcement learning. *arXiv [cs.LG]*.

Pearl, J. 2009. *Causality*. Cambridge University Press, 2 edition.

Rebello, A.; Tang, S.; Wiens, J.; and Parbhoo, S. 2023. Leveraging factored action spaces for off-policy evaluation. *arXiv [cs.LG]*.

Shwartz-Ziv, R.; and Armon, A. 2022. Tabular data: Deep learning is not all you need. *Inf. Fusion*, 81: 84–90.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement Learning: An Introduction*. MIT Press, second edition. ISBN 9780262039246.

Tao, T. 2011. *An introduction to measure theory*. Graduate studies in mathematics. Providence, RI: American Mathematical Society.

Thomas, P. S.; and Brunskill, E. 2016. Data-efficient off-policy policy evaluation for reinforcement learning. *arXiv [cs.LG]*.

Towers, M.; Kwiatkowski, A.; Terry, J.; Balis, J. U.; De Cola, G.; Deleu, T.; Goulão, M.; Kallinteris, A.; Krimmel, M.; Kg, A.; Perez-Vicente, R.; Pierré, A.; Schulhoff, S.; Tai, J. J.; Tan, H.; and Younis, O. G. 2024. Gymnasium: A standard interface for reinforcement learning environments. *arXiv [cs.LG]*.

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with Double Q-learning. *Proc. Conf. AAAI Artif. Intell.*, 30(1).

Wang, X.; Wongkamjan, W.; and Huang, F. 2022. Live in the moment: Learning dynamics model adapted to evolving policy. *arXiv [cs.LG]*.

Watkins, C. J. C. H.; and Dayan, P. 1992. Q-learning. *Mach. Learn.*, 8(3): 279–292.

Wright, S. 1921. *Correlation and Causation*.