Learning Policy-Aware Models for Model-Based Reinforcement Learning via Transition Occupancy Matching

Yecheng Jason Ma* JASONYMA@SEAS.UPENN.EDU
Kausik Sivakumar* KAUSIK@SEAS.UPENN.EDU
Jason Yan JASYAN@SEAS.UPENN.EDU
Osbert Bastani OBASTANI@SEAS.UPENN.EDU
Dinesh Jayaraman DINESHJ@SEAS.UPENN.EDU
University of Pennsylvania

Editors: R. Firoozi, N. Mehr, E. Yel, R. Antonova, J. Bohg, M. Schwager, M. Kochenderfer

Abstract

Standard model-based reinforcement learning (MBRL) approaches fit a transition model of the environment to all past experience, but this wastes model capacity on data that is irrelevant for policy improvement. We instead propose a new "transition occupancy matching" (TOM) objective for MBRL model learning: a good environment model has the property that the current policy experiences the same distribution of transitions, whether deployed in the real environment or inside the model. We derive TOM directly from a novel lower bound on the standard reinforcement learning objective. To optimize TOM, we show how to reduce it to a form of importance weighted maximum-likelihood estimation, where the automatically computed importance weights identify policy-relevant past experiences from a replay buffer, enabling stable optimization. TOM thus offers a plug-and-play model learning sub-routine that is compatible with any backbone MBRL algorithm. On various Mujoco continuous robotic control tasks, we show that TOM successfully focuses model learning on policy-relevant experience and drives policies faster to higher task rewards than alternative model learning approaches. Our full technical report with Appendix is linked: https://www.seas.upenn.edu/ jasonyma/materials/L4DC-2023.pdf

1. Introduction

Model-based reinforcement learning (MBRL) (Sutton, 1991) is an effective paradigm for sample-efficient policy learning. In MBRL, an agent learns a dynamics model of its environment from its own experience. This learned dynamics model acts as a simulator, generating fictitious experience for policy optimization. The improved policy then generates new environment experiences, which are used to improve the dynamics model, and the cycle continues. MBRL's sample efficiency, coupled with breakthroughs in deep neural networks, have enabled impressive applications such as mastering Atari games and simulated robot control from pixels (Hafner et al., 2019a,b, 2020; Kaiser et al., 2019), in-hand dexterous manipulation (Nagabandi et al., 2020), and real-world robotics control (Finn and Levine, 2017; Ebert et al., 2018; Wu et al., 2022).

The de facto standard approach to model learning trains the parameters of a neural network dynamics model by maximizing the likelihood of *all* observed environment transitions (MLE). To see that this is inefficient, consider a car on a road passing through rocky terrain. For the task of driving well, the agent need not know the complex dynamics of driving over the rocks; a model of the simple dynamics of the road surface would be sufficient. However, the MLE approach aims

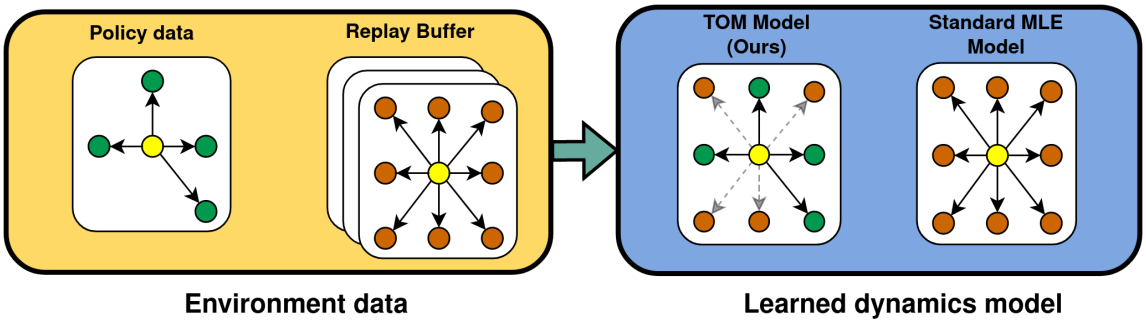

Figure 1: Transition Occupancy Matching (TOM) enables learning a dynamics model that fits the policy's visitation distribution accurately to enable rapid policy improvement.

instead to learn a more comprehensive model, emphasizing all past experience equally, even when most of it is irrelevant to improving the current policy. For example, reinforcement learning (RL) policies acting largely randomly in early stages of training would mostly generate experiences of driving over rocks, and MLE models would continue to fit this data even after the policy has learnt to stay on the road. This wastefulness is largely due to the objective mismatch (Lambert et al., 2020) between MLE model learning and optimal policy learning.

To enable efficient training of task-relevant models and accelerate policy learning, we propose "transition occupancy matching" **(TOM)**. At a high level, TOM changes the model learning objective to focus more on environment transitions that the current policy can experience, generating "policy-aware" models well-suited for policy improvement. For example, in the car setting above, as the policy starts to spend more time on the road, the dynamics model adapts by fitting the "footprint" of this policy which now includes more on-road experience, driving faster policy improvement.

While the above procedure is intuitive, we derive TOM from first principles. Our **first** contribution is a novel lower bound to the standard RL objective, containing two parts: (1) standard policy search through reward maximization with respect to a learned model, as in many prior MBRL approaches, and (2) a novel f-divergence model learning objective that aims to match the footprints of the current policy in the real environment and in the learned model. **Second,** we overcome optimization challenges associated with the TOM model learning objective by showing that it is mathematically analogous to the offline imitation learning objective. With this insight, we adapt a recently proposed offline imitation approach (Ma et al., 2022a) to reduce the TOM model-learning objective to *importance-weighted* maximum likelihood model learning, where higher weights are assigned to past transitions that lie in the footprint of the current policy. This permits a stable, easy-to-implement optimization procedure. **Third,** we demonstrate that, when plugged into a standard MBRL system, TOM's model learning procedure induces better policies for various simulated robotics tasks faster than alternative approaches, by continuously focusing the model on the most relevant past experiences.

2. Related Work

Our work is broadly related to the objective mismatch problem (Lambert et al., 2020) in MBRL. One prominent approach to address the objective mismatch problem is to make dynamics learning *value-aware* (Farahmand et al., 2017; Farahmand, 2018; Grimm et al., 2020; Farquhar et al., 2021;

Voelcker et al., 2022). In particular, this line of work attempts to learn dynamics models that capture aspects of environment dynamics that impact accurate estimation of the value functions. However, this paradigm entangles the policy's footprint with the task it is trying to solve, and often require well-shaped dense reward in the environment so that the value functions are not degenerate in the early iterations of policy optimization.

Instead of focusing on the value function, our approach is more direct and *policy-aware* (Eysenbach et al., 2021; Wang et al., 2022), cognizant of the current policy's footprint without entangling it with the task it is solving. The closest work to ours is PMAC (Wang et al., 2022), which proposes to up-weight the most recent transitions in the replay buffer according to a hand-crafted weight schedule in regressing the dynamics model; however, this approach is heuristic in nature, and sensitive to hyperparameters. Furthermore, PMAC suffers from *recency bias*; due to the inherent variance in policy optimization, the most recent transitions may be of low quality and not most relevant to improving the current policy, but PMAC would assign them high weights regardless. In contrast, TOM first establishes a lower bound to the true policy return objective in the transition occupancy space, then leverages techniques in dual reinforcement learning to derive a principled and optimal approach for assigning transition weights that is empirically effective.

3. Background: Model-Based RL, State-Action Occupancies, and Bellman Flows

In this section, we will first go over the preliminaries for model-based reinforcement learning and then discuss the concept of state-action occupancy.

Model-based reinforcement learning. We consider an infinite horizon discounted Markov decision process (MDP) (Puterman, 2014) $\mathcal{M} = (S, A, R, T, \mu_0, \gamma)$ where S denotes its state space, A its action space, R the reward, $T(s, a)$ the transition function, $\mu_0(s)$ its initial state distribution, and $\gamma \in (0, 1]$ the discount factor. A policy $\pi : S \to \Delta(A)$ is a state-conditioned action distribution. The objective of RL is to find the policy π that maximizes the discounted return:

$$J(\pi) \coloneqq \mathbb{E}_{s_0 \sim \mu_0, a_t \sim \pi(\cdot|s_t), s_{t+1} \sim T(\cdot|s_t, a_t)} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t.a_t) \right]. \tag{1}$$

We consider the online reinforcement learning setting, in which the agent directly interacts with the environment, collects new transitions (s, a, r, s') and stores them in its replay buffer D. The agent's policy is updated using samples from D. We define the replay buffer empirical policy as $\pi_D(a \mid s) := \frac{n(s,a)}{n(s)}$, where $n(\cdot)$ is the raw count of a state(-action) in D.

Since the true dynamics T is not known, MBRL builds an approximate dynamics model $\hat{T}$ which is learned from data. That is, a function approximator is built by designing $\hat{T}(s, a)$ as a probability distribution and by maximizing the likelihood of observing next state s' given current state-action pair (s, a) over transitions present in the collected replay buffer $(s, a, s') \sim D$. This can also be presented as minimizing the reverse KL divergence between transitions conditioned on samples in the replay buffer:

$$\arg\min_{\hat{T}} \mathbb{E}_{D(s,a,s')} \mathrm{KL}\left(T(\cdot \mid s, a) \| \hat{T}(\cdot \mid s, a) \right) = \arg\min_{\hat{T}} \mathbb{E}_{D(s,a,s')} \left[\log \hat{T}(\cdot \mid s, a) \right]. \tag{2}$$

State-Action Occupancy. The state-action occupancies (also known as stationary distribution) $d_T^\pi(s,a) : \mathcal{S} \times \mathcal{A} \to [0,1]$ of policy π is

$$d_T^\pi(s,a) \coloneqq (1-\gamma)\sum_{t=0}^{\infty}\gamma^t \text{Pr}(s_t = s, a_t = a \mid s_0 \sim \mu_0, a_t \sim \pi(s_t), s_{t+1} \sim T(s_t, a_t)). \quad (3)$$

This captures the relative state-action visitation frequencies of policy π under dynamics T.[1] The policy's state occupancies can be obtained by marginalizing over actions: $d^\pi(s) = \sum_a d^\pi(s,a)$. With this definition, we notice the following relationship between the policy and its occupancy distributions:

$$\pi(a \mid s) = \frac{d^\pi(s,a)}{d^\pi(s)}. \quad (4)$$

By construction, every policy's visitation distribution $d^\pi(s,a)$, must satisfy the single step transpose Bellman equation:

$$d^\pi(s,a) = (1-\gamma)\mu_0(s)\pi(a \mid s) + \gamma\pi(a \mid s)\sum_{\tilde{s},\tilde{a}} T(s \mid \tilde{s},\tilde{a})d(\tilde{s},\tilde{a}). \quad (5)$$

This is known as the Bellman *flow* constraint, which intuitively restricts the "flow" of a policy's state-action distribution where each $d^\pi(s,a)$ must be expressed as a weighted sum. Conversely, a state-action occupancy distribution $d(s,a)$ needs to satisfy the Bellman flow constraint in order for it to be a valid $d^\pi(s,a)$ for some policy π:

$$\sum_a d(s,a) = (1-\gamma)\mu_0(s) + \gamma\sum_{\tilde{s},\tilde{a}} T(s \mid \tilde{s},\tilde{a})d(\tilde{s},\tilde{a}), \forall s \in S, a \in A. \quad (6)$$

Given d^π, one can express the RL objective (1) as:

$$J(\pi) = \frac{1}{1-\gamma}\mathbb{E}_{(s,a)\sim d^\pi(s,a)}[R(s,a)]. \quad (7)$$

Finally, we can reframe (7) as a constrained optimization problem directly in the space of state-action occupancy distributions d by incorporating the Bellman flow constraint (6):

$$\begin{aligned} &\max_d \frac{1}{1-\gamma}\mathbb{E}_{(s,a)\sim d(s,a)}[R(s,a)] \\ &\text{s.t.} \quad \sum_a d(s,a) = (1-\gamma)\mu_0(s) + \gamma\sum_{\tilde{s},\tilde{a}} T(s \mid \tilde{s},\tilde{a})d(\tilde{s},\tilde{a}), \forall s \in S, a \in A \end{aligned} \quad (8)$$

4. Transition Occupancy Matching

We now develop our approach, transition occupancy matching (TOM). In Section 4.1, we extend state-action occupancy to a new concept: transition occupancy, which formalizes the intuitive notion of the "policy footprint", motivated in the introduction. Next, in Section 4.2 we derive a novel lower bound to the RL objective that naturally suggests learning a policy-aware dynamics model as the key ingredient for model-based policy learning. Finally, Section 4.3 develops the full TOM algorithm by casting policy-aware model learning as an offline imitation problem.

1. Unless otherwise specified, we assume $d^\pi(s,a)$ is computed under the true dynamics T.

4.1. Extending State-Action Occupancy to "Transition Occupancy"

Given a policy's state-action occupancy distribution $d^\pi_T(s,a)$ under a transition function T (with generality, T in this section refers to any transition function and not necessarily the true environment dynamics), we define its *transition occupancy distribution* (TOD) as:

$$d^\pi_T((s,a),s') := T(s' \mid s,a) d^\pi_T(s,a). \tag{9}$$

Intuitively, $d^\pi_T((s,a),s')$ captures the relative frequency of any transition tuple (s,a,s') that a policy visits under T. One immediate property of this definition is that we can back out the transition function as follows:

$$T(s' \mid s,a) := \frac{d^\pi_T((s,a),s')}{\sum_{s'} d^\pi_T((s,a),s')}, \forall \pi \tag{10}$$

Finally, we need to specify an analogous Bellman flow constraint for valid transition occupancies. Note that the original Bellman flow constraint (6) already contains a TOD term on the right: $T(s \mid \tilde{s},\tilde{a}) d^\pi(\tilde{s},\tilde{a}) = d((\tilde{s},\tilde{a}),s')$. Therefore, by multiplying both sides of (6) by $T(s' \mid s,a)$, we obtain the *Bellman transition flow* constraint:

$$d^\pi_T((s,a),s') = (1-\gamma)\mu_0(s)T(s' \mid s,a)\pi(a \mid s) + \gamma T(s' \mid s,a)\pi(a \mid s)\textstyle\sum_{\tilde{s},\tilde{a}} d^\pi_T((\tilde{s},\tilde{a}),s), \forall (s,a,s') \in S \times A \times S \tag{11}$$

4.2. Policy-Aware Lower Bound via Transition Occupancy f-Divergence

With this new notion of "transition occupancy distribution" (TOD), we can rewrite the RL objective as:

$$J(\pi) = \frac{1}{1-\gamma}\mathbb{E}_{d^\pi_T(s,a)}[R(s,a)] = \frac{1}{1-\gamma}\mathbb{E}_{d^\pi_T((s,a),s')}[R(s,a)]. \tag{12}$$

Now, $\log J(\pi)$ permits a simple lower bound in terms of TODs:

$$\begin{aligned}
\log J(\pi) &= \log \mathbb{E}_{d^\pi_T((s,a),s')}[R(s,a)] + C \\
&= \log \mathbb{E}_{d^\pi_{\hat{T}}((s,a),s')}\left[\frac{d^\pi_T((s,a),s')}{d^\pi_{\hat{T}}((s,a),s')} R(s,a)\right] + C \\
&\geq \mathbb{E}_{d^\pi_{\hat{T}}((s,a),s')}\left[\log \frac{d^\pi_T((s,a),s')}{d^\pi_{\hat{T}}((s,a),s')} + \log R(s,a)\right] + C \qquad \text{(by Jensen's inequality)} \\
&\geq \underbrace{-\mathrm{D}_f(d^\pi_T((s,a),s') \| d^\pi_T((s,a),s'))}_{\text{TOM model learning objective}} + \underbrace{\mathbb{E}_{d^\pi_{\hat{T}}((s,a),s')}[\log R(s,a)]}_{\text{standard RL within a learned model}} + C,
\end{aligned} \tag{13}$$

where C is a constant that does not impact optimization and f is any f-divergence that upper bounds the KL divergence.

This lower bound serves as a surrogate objective, and consequently (iteratively) maximizing it will lead to maximizing the true RL objective. More importantly, it consists of two expressions that directly suggests a recipe for MBRL: (1) the first expression suggests trainining the dynamics model by minimizing the f-divergence between the distributions of real and fake policy rollouts, and (2) the second expression suggests optimizing the policy w.r.t. the learned model. The second expression is just the RL objective (12) with $\hat{T}$ instead of T, and can be optimized by any existing

RL algorithm. As such, TOM's technical contribution is a model learning sub-routine aimed at minimizing this f-divergence:

$$\min_{\hat{T}} \mathrm{D}_f(d^{\pi}_{\hat{T}}((s,a),s') \| d^{\pi}_{T}((s,a),s') \tag{14}$$

4.3. Optimizing the TOM objective

We observe that optimizing (14) requires estimating $d^{\pi}_{\hat{T}}((s,a),s')$; however, doing so requires rolling out π in $\hat{T}$, which suffers from the compounding error of multi-step trajectory rollout in one-step dynamics models (Lambert et al., 2022) that makes the estimation inaccurate. Furthermore, this direct optimization approach fails to leverage the replay buffer D that the agent has already collected. To circumvent these issues, we propose a practical algorithm that can learn a policy-aware model tailored to the visitation distribution of π without any additional samples from the real environment. The algorithm is derived by treating transition occupancy matching as an *offline* imitation learning problem. We begin by illustrating the intuition of this reduction; then, we provide the technical derivation.

Transition Occupancy Matching as Offline Imitation: An Analogy. Below, we compare the TOM problem (left) and the well-known *state-action* occupancy matching problem (right) (Nachum and Dai, 2020; Ma et al., 2022a; Kim et al., 2022):

$$\min_{\hat{T}} \mathrm{D}_f(d^{\pi}_{\hat{T}}((s,a),s') \| d^{\pi}_{T}((s,a),s') \qquad \text{vs.} \qquad \min_{\pi} \mathrm{D}_f(d^{\pi}_{T}((s,a)) \| d^{\pi^*}_{T}((s,a)) \tag{15}$$

For state-action occupancy matching, the environment dynamics T is fixed, and the goal is to learn a policy π that matches the distribution of a target (optimal) policy π^*. And the TOM problem precisely *reverses* the role of the policy π and the dynamics model T. In this analogy, a "transition function" should map from "state" (s,a), under "action" s', to a distribution over new "states" $p(s',a'|s,a,s') = p(a'|s,a,s')$, which further reduces to $p(a'|s')$ under Markov assumptions. Note that this last distribution is precisely the policy π. In other words, the policy takes the place of the transition function and vice versa.

Regularized TOM For Offline Model Learning. Given this analogy, we wish to derive a policy-aware model learning algorithm by adapting a suitable state-occupancy based imitation learning problem. In particular, we extend SMODICE (Ma et al., 2022a) to allow TOM to learn the policy-aware dynamics model using just the replay buffer D, circumventing the issues laid out at the beginning of the section. More specifically, we first regularize $d^{\pi}_{\hat{T}}((s,a),s')$ to the transition occupancy distribution $d^{\pi_D}_{T}((s,a),s')$ of the empirical policy π_D via a choice of f-divergence, which is a crucial step in enabling learning $\hat{T}$ using solely the replay buffer D without any additional simulated samples from $\hat{T}$ itself:

$$\begin{aligned} \max_{d^{\pi}_{\hat{T}}} \quad & -\mathrm{D}_f(d^{\pi}_{\hat{T}}((s,a),s') \| d^{\pi}_{T}((s,a),s')) - \mathrm{D}_f(d^{\pi}_{\hat{T}}((s,a),s') \| d^{\pi_D}_{T}((s,a),s')) \\ \text{s.t.} \quad & \sum_{s'} d^{\pi}_{\hat{T}}((s,a),s') = (1-\gamma)\mu_0(s)\pi(a \mid s) + \gamma\pi(a \mid s) \sum_{\tilde{s},\tilde{a}} d^{\pi}_{\hat{T}}((\tilde{s},\tilde{a}),s), \forall (s,a) \in S \times A \end{aligned} \tag{16}$$

Here, we have incorporated the Bellman transition flow constraint (11).

Algorithm 1 Transition Occupancy Matching

1: **Require**: current policy π and its environment rollout(s) τ, replay buffer D
2: `// Discriminator Learning`
3: Train discriminator $c^*(s, a, s')$ to separate policy-relevant transitions from others in the replay buffer (27) and derive $R(s, a, s')$.
4: `// Q-Function Learning`
5: Train policy-relevance Q-function $Q^*(s, a)$ using (17)
6: `// Model Learning`
7: Train policy-aware dynamics model $\hat{T}(s' \mid s, a)$ using (19)

Offline Optimization In The Dual Form. Next, Eq (16) admits a simple dual problem that can be optimized using solely the replay buffer and whose optimal solution can be directly used to compute the primal optimal $d^{\pi}_{\hat{T}^*}$ and thereby $\hat{T}^*$:

Proposition 1 *The dual problem to* (16) *is:*

$$\min_{Q \geq 0} (1-\gamma)\mathbb{E}_{\mu_0,\pi}[Q(s,a)] + \mathbb{E}_{d^{\pi_D}_T(s,a,s')}\left[f_\star \left(\underbrace{\log \frac{d^{\pi}_T(s,a,s')}{d^{\pi_D}_T(s,a,s')}}_{:=r(s,a,s')} + \gamma \mathbb{E}_{\pi(a'|s')}[Q(s',a')] - Q(s,a) \right) \right] \tag{17}$$

where $f_\star$ denotes the convex conjugate function of f. Given the optimal Q^*, $\forall (s, a, s') \in S \times A \times S$, the primal optimal solution satisfies

$$d^{\pi}_{\hat{T}^*}((s,a),s') = d^{\pi_D}_T((s,a),s') f'_\star \left(r(s,a,s') + \gamma \mathbb{E}_{\pi(a'|s')}[Q^*(s',a')] - Q^*(s,a) \right) \tag{18}$$

See Appendix A for a proof. Conceptually, this dual problem learns a Q-function that informs the relevance of state-action pairs for learning $d^{\pi}_{\hat{T}^*}((s,a),s')$ according to reward $r(s,a,s')$ (this is not the same as the task reward R), which is the log-ratio of the transition occupancy distributions of the current policy π and the replay buffer empirical policy π_D; in Appendix A.2, we detail how to estimate $r(s,a,s')$ in practice by training a transition discriminator. Crucially, this dual problem depends on only the replay buffer D, which includes samples from μ_0, and the current policy π, without any requirement on further data collection. Therefore, in practice, we can approximate Q^* by optimizing (17) using stochastic gradient descent (SGD) on D.

Then, we can use (18) to construct the optimal importance weights $\frac{d^{\pi}_{\hat{T}^*}((s,a),s')}{d^{\pi_D}_T((s,a),s')}$ and perform weighted regression (Ma et al., 2022b):

$$\begin{aligned} &\min_{\hat{T}} - \mathbb{E}_{d^{\pi}_{\hat{T}^*}((s,a),s')}[\log \hat{T}(s' \mid s,a)] \\ &= - \mathbb{E}_{d^{\pi_D}_T(s,a,s')}\left[f'_\star \left(r(s,a,s') + \gamma \mathbb{E}_{\pi(a'|s')}[Q^*(s',a')] - Q^*(s,a) \right) \log \hat{T}(s' \mid s,a) \right] \end{aligned} \tag{19}$$

Here, the expectation depends only on the replay buffer distribution π_D, permitting supervised learning directly on D. As such, we see that the dynamics model is still trained with MLE, but just on a different distribution that is *policy-aware*; equivalently, the model is learned via *behavior cloning* (BC) on the current policy's transition occupancy distribution. As such, TOM retains the training stability of supervised learning, while simultaneously ensuring policy-awareness that enables optimizing a well-defined lower bound to the true RL objective and improved sample efficiency. The TOM algorithm is summarized in Alg. 1, and a full version is in Alg. 2, Appendix B.

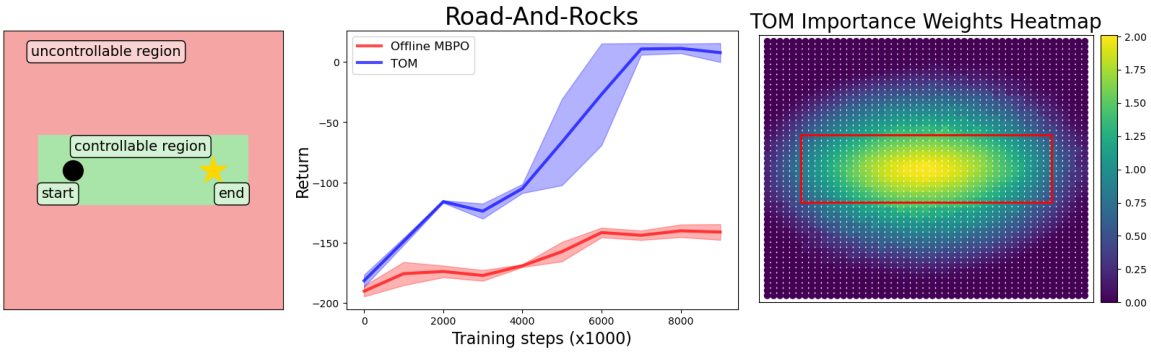

Figure 2: Road-And-Rocks toy environment (left). TOM substantially outperforms (offline) MBPO (middle), and is doing so because it is able to correctly assign higher importance weights to the controllable region (right).

5. Experiments

To evaluate TOM, we replace MLE model learning in a standard MBRL algorithm, MBPO (Janner et al., 2019) with TOM-based model learning. We measure policy rewards and sample efficiency of policy learning, and also investigate whether TOM successfully focuses the model on policy-relevant samples.

5.1. Offline Linear MBRL on Road-And-Rocks

We first evaluate TOM on a "road-and-rocks" task, based on the introduction example, shown in Figure 2 (left). A "car" agent can steer in any direction. In the green "road" region, these actions uniformly produce the desired motions. In the red "rocks" region, the effects are different at each location, simulating the complex dynamics of driving on rocky terrain. The task is to drive on the road from a start to a goal position. For this toy experiment, we train a linear dynamics model. Off-road dynamics are highly non-linear, however the linear model will suffice if focused on the correct task-relevant on-road data. To isolate the effect of model learning from the quality of data collected by online exploration (Lambert et al., 2022), we consider an offline model-based reinforcement learning (Yu et al., 2020; Kidambi et al., 2020) setting; policy optimization takes place using only pre-recorded "offline" environment transitions without any further data collection. The offline data includes random exploratory dataset over the entire map, plus a small amount of "expert" trajectories that successfully complete the task; this resembles practical use cases where the dataset provides high coverage of the state space but is mostly sub-optimal and task-irrelevant. For TOM, one such expert trajectory is treated as the current policy footprint.

As expected, Figure 2 (middle) shows vastly better environment reward curves when replacing MBPO model-learning with TOM. The inferred importance weights for transitions in various parts of the heatmap, shown in Figure 2 (right), confirm that TOM successfully focuses the model on the task-relevant road regions, de-emphasizing the highly non-linear rocky regions.

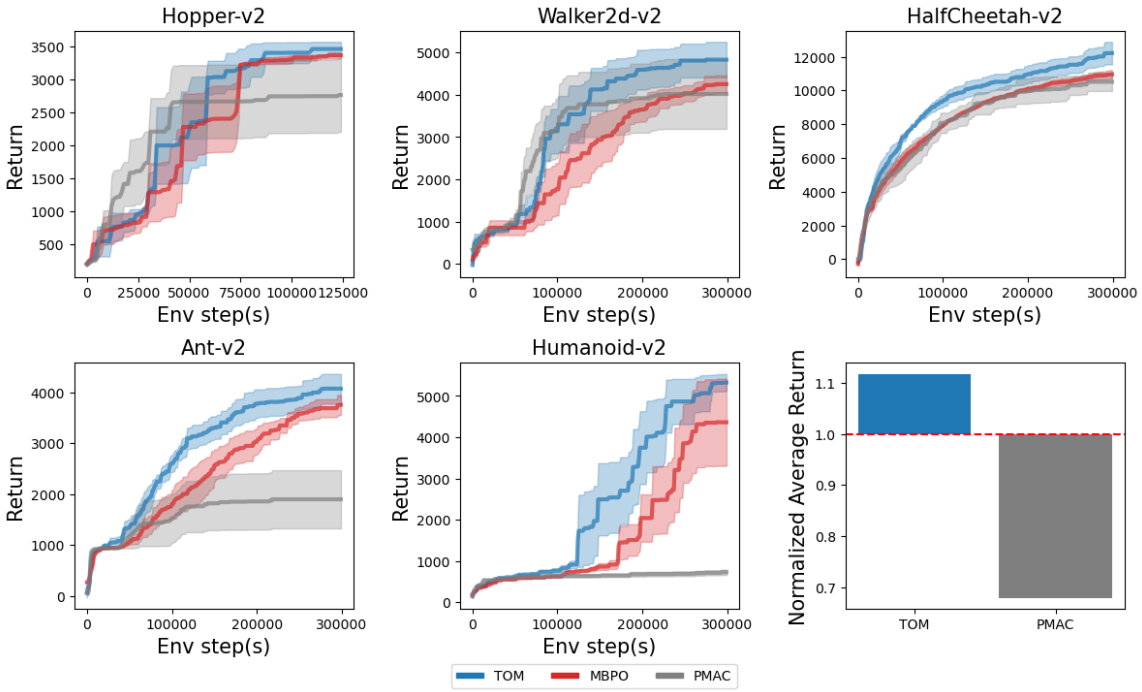

Figure 3: Return plots on Mujoco environments.

5.2. Online Deep MBRL on Simulated Robotics Tasks

Having demonstrated that TOM performs as expected in the toy road-and-rocks environment above, we move to standard MuJoCo continuous control benchmarks (Todorov et al., 2012).

Baselines. To evaluate TOM model learning, we stay with the widely used MBPO framework (Janner et al., 2019) for MBRL, but now compare TOM against not only standard MLE model learning, but also a recently proposed alternative model learning approach, Policy-adaptation Model-based Actor-Critic (PMAC) (Wang et al., 2022), which heuristically upweights recent transitions in the replay buffer according to an exponentially decaying schedule. See Appendix C for more implementation details. We will release all code upon publication.

Environments and Training Details. On 5 standard Mujoco environments: Hopper, Walker, HalfCheetah, Ant, and Humanoid, we train policies for up to 300k environment steps. We inherit standard MBPO hyperparameters for TOM and all our model-learning baselines; see Appendix C for details. We evaluate all algorithms for 4 seeds and report the cumulative maximum average return over 10 test rollouts achieved during training.

Reward Curves. The reward curves in Figure 3 show that TOM performs the best in aggregate across all environments, in terms of both sample efficiency and final policy performance. TOM's gains are larger in environments with more complex dynamics, such as Walker, HalfCheetah, and Humanoid, where judicious use of model capacity is more critical. Notably on Humanoid, TOM learns a policy that achieves at least 60% more reward than plain MBPO. PMAC is able to initially outpace TOM on the simpler environments such as Hopper and Walker; however, this advantage is quickly erased as training continues, and PMAC converges to a worse return than TOM. Furthermore, on the more difficult environments such as Ant and Humanoid, PMAC is highly sub-optimal

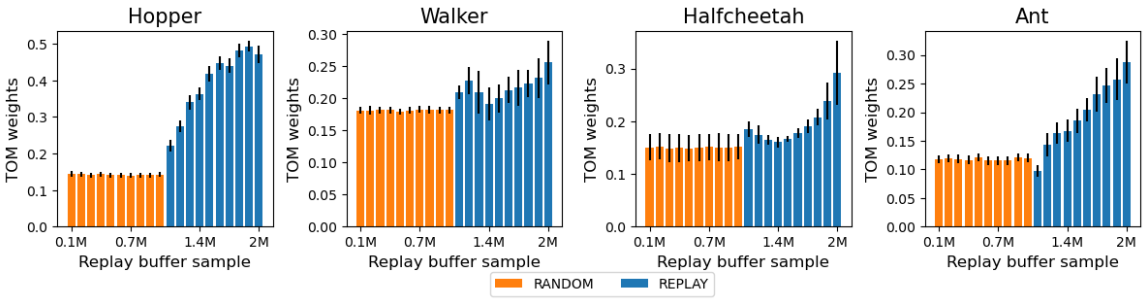

Figure 4: TOM transition importance weights.

and reaches much lower asymptotic performance. These results suggest that heuristically weighting more recent transitions in the replay buffer is not a stable method and does not scale, demonstrating that a more principled approach like TOM is needed in order to be correctly policy-aware.

Analysis of TOM Importance Weights. It is clear above that TOM performs well relative to baselines in these tasks, but does it do so for the right reasons? In other words, does it actually focus the model on the right data? To investigate this systematically, we manually curate a carefully ordered replay buffer, ordering it such that the first 50% of the data contains random transitions, and the next 50% contains data from various ordered stages in the training progress of a soft-actor critic (SAC) (Haarnoja et al., 2018) policy agent, as provided by the D4RL dataset (Fu et al., 2021). We fix the fully trained "expert" SAC agent as the policy of interest, and run TOM model learning fully offline. We should expect good importance weights to be uniformly low over the first half of the replay buffer and then increase as we progress through the second half. And indeed, Figure 4 clearly shows these trends for the average TOM importance weights in various chunks of the replay buffer.

It is also pertinent that TOM importance weights do not only rise at the very end of the replay buffer, when data is near-optimal; instead, they start to rise earlier, sometimes even non-monotonically, suggesting that the recency heuristic by PMAC, is not the sole determinant of relevance for model learning. In Appendix D, we show that this also commonly occurs during the deep online MBRL experiments: TOM-assigned weights are quite frequently higher for older data than for newer ones.

6. Conclusion

We have introduced Transition Occupancy Matching (TOM), a principled policy-aware model learning approach to address the objective mismatch challenge in model-based reinforcement learning. TOM introduces the notion of transition occupancy and derives a simple lower bound to the reinforcement learning objective, which permits casting learning a policy-aware dynamics model as learning the optimal importance weights for weighted regression model updates. The importance weights are derived from the theory of dual reinforcement learning, and TOM's practical implementation is modular and compatible with any MBRL algorithm that implements MLE regression-based model learning. On the standard suite of Mujoco tasks, TOM improves the learning speed of a standard MBRL algorithm while achieving significantly higher asymptotic performance compared to non-policy aware methods.

References

Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv preprint arXiv:1812.00568*, 2018.

Benjamin Eysenbach, Alexander Khazatsky, Sergey Levine, and Ruslan Salakhutdinov. Mismatched no more: Joint model-policy optimization for model-based rl. *arXiv preprint arXiv:2110.02758*, 2021.

Amir-massoud Farahmand. Iterative value-aware model learning. *Advances in Neural Information Processing Systems*, 31, 2018.

Amir-massoud Farahmand, Andre Barreto, and Daniel Nikovski. Value-aware loss function for model-based reinforcement learning. In *Artificial Intelligence and Statistics*, pages 1486–1494. PMLR, 2017.

Greg Farquhar, Kate Baumli, Zita Marinho, Angelos Filos, Matteo Hessel, Hado P van Hasselt, and David Silver. Self-consistent models and values. *Advances in Neural Information Processing Systems*, 34:1111–1125, 2021.

Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE, 2017.

Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2021.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

Christopher Grimm, André Barreto, Satinder Singh, and David Silver. The value equivalence principle for model-based reinforcement learning. *Advances in Neural Information Processing Systems*, 33:5541–5552, 2020.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019a.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019b.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. *Advances in Neural Information Processing Systems*, 32, 2019.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. *arXiv preprint arXiv:2005.05951*, 2020.

Geon-Hyeong Kim, Seokin Seo, Jongmin Lee, Wonseok Jeon, HyeongJoo Hwang, Hongseok Yang, and Kee-Eung Kim. DemoDICE: Offline imitation learning with supplementary imperfect demonstrations. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=BrPdX1bDZkQ.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Nathan Lambert, Brandon Amos, Omry Yadan, and Roberto Calandra. Objective mismatch in model-based reinforcement learning. *arXiv preprint arXiv:2002.04523*, 2020.

Nathan Lambert, Kristofer Pister, and Roberto Calandra. Investigating compounding prediction errors in learned dynamics models. *arXiv preprint arXiv:2203.09637*, 2022.

Yecheng Jason Ma, Andrew Shen, Dinesh Jayaraman, and Osbert Bastani. Smodice: Versatile offline imitation learning via state occupancy matching. *arXiv preprint arXiv:2202.02433*, 2022a.

Yecheng Jason Ma, Jason Yan, Dinesh Jayaraman, and Osbert Bastani. How far i'll go: Offline goal-conditioned reinforcement learning via f-advantage regression. *arXiv preprint arXiv:2206.03023*, 2022b.

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

Anusha Nagabandi, Kurt Konolige, Sergey Levine, and Vikash Kumar. Deep dynamics models for learning dexterous manipulation. In *Conference on Robot Learning*, pages 1101–1112. PMLR, 2020.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991.

Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

Claas Voelcker, Victor Liao, Animesh Garg, and Amir-massoud Farahmand. Value gradient weighted model-based reinforcement learning. *arXiv preprint arXiv:2204.01464*, 2022.

Xiyao Wang, Wichayaporn Wongkamjan, and Furong Huang. Live in the moment: Learning dynamics model adapted to evolving policy. In *Decision Awareness in Reinforcement Learning Workshop at ICML 2022*, 2022. URL https://openreview.net/forum?id=2rlTW4fXFdf.

Philipp Wu, Alejandro Escontrela, Danijar Hafner, Ken Goldberg, and Pieter Abbeel. Daydreamer: World models for physical robot learning. *arXiv preprint arXiv:2206.14176*, 2022.

Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. *arXiv preprint arXiv:2005.13239*, 2020.

Appendix A. Technical Derivations

In this section, we provide the omitted technical derivations in the main text.

A.1. Proof of Proposition 1

We begin by stating an assumption on the strict feasibility of the Bellman transition flow constraint in (16).

Assumption 1 There exists at least one $d(s, a, s')$ such the Bellman transition flow constraint is satisfied and $\forall s \in \mathcal{S}, d(s) > 0$.

This assumption is mild and typically satisfied for any practical MDP in which every state is reachable from the initial state distribution.

Now, we first write down the Lagrangian of (16):

$$\max_{d^\pi_{\hat{T}}} \min_{Q} - \mathrm{D}_f(d^\pi_{\hat{T}}((s,a),s') \| d^\pi_T((s,a),s')) - \mathrm{D}_f(d^\pi_{\hat{T}}((s,a),s') \| d^{\pi_D}_T((s,a),s')) + \sum_{s,a} Q(s,a) \left((1-\gamma)\mu_0(s)\pi(a \mid s) + \gamma\pi(a \mid s) \sum_{\tilde{s},\tilde{a}} d^\pi_{\hat{T}}((\tilde{s},\tilde{a}),s) - \sum_{s'} d^\pi_{\hat{T}}((s,a),s') \right) \tag{20}$$

We use the following two identities to simplify the objective:

$$\sum_{s,a} Q(s,a) \left(\sum_{s'} d^\pi_{\hat{T}}((s,a),s') \right) = \mathbb{E}_{d^\pi_T((s,a),s')}[Q(s,a)] \tag{21}$$

and

$$\sum_{s,a} Q(s,a) \left(\gamma\pi(a \mid s) \sum_{\tilde{s},\tilde{a}} d^\pi_{\hat{T}}((\tilde{s},\tilde{a}),s) \right) = \gamma \mathbb{E}_{d^\pi_{\hat{T}}((s,a),s')} \mathbb{E}_{\pi(a'|s')}[Q(s',a')], \tag{22}$$

which follow from standard algebraic manipulations. Using these identities and the strict feasibility assumption, strong duality enables switching the order of optimization and simplifies the objective to

$$\min_{Q} \max_{d^\pi_{\hat{T}} \geq 0} (1-\gamma)\mathbb{E}_{\mu_0,\pi}[Q(s,a)] + \mathbb{E}_{d^\pi_{\hat{T}}(s,a,s')} \left[\underbrace{\log \frac{d^\pi_T(s,a,s')}{d^{\pi_D}_T(s,a,s')}}_{:=R(s,a,s')} + \gamma \mathbb{E}_{\pi(a'|s')}[Q(s',a')] - Q(s,a) \right] - \mathrm{D}_f(d^\pi_{\hat{T}}((s,a),s') \| d^{\pi_D}_T((s,a),s')) \tag{23}$$

Then, we recognize that the inner maximization is precisely the Fenchel conjugate of

$$\mathrm{D}_f(d^\pi_{\hat{T}}((s,a),s') \| d^{\pi_D}_T((s,a),s')) \tag{24}$$

at $R(s,a,s') + \gamma \mathbb{E}_{\pi(a'|s')}[Q^*(s',a')] - Q^*(s,a)$, which allows us to reduce (23) to the Fenchel *dual* problem of (16):

$$\min_Q (1-\gamma)\mathbb{E}_{\mu_0,\pi}[Q(s,a)] + \mathbb{E}_{d_T^{\pi_D}(s,a,s')}\left[f_\star(R(s,a,s') + \gamma \mathbb{E}_{\pi(a'|s')}[Q(s',a')] - Q(s,a)\right] \quad (25)$$

Then, leveraging Lemma 3 from Ma et al. (2022a), it follows that

$$d^\pi_{\hat{T}_*}((s,a),s') = d^{\pi_D}_T((s,a),s') f'_\star\left(r(s,a,s') + \gamma \mathbb{E}_{\pi(a'|s')}[Q^*(s',a')] - Q^*(s,a)\right) \quad (26)$$

A.2. Discriminator Training

In practice, $r(s,a,s')$ can be estimated by training a discriminator $c : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to (0,1)$ that distinguishes transitions from π and π_D:

$$\min_c \mathbb{E}_{(s,a,s') \sim d^\pi_T(s,a,s')}\left[\log c(s,a,s')\right] + \mathbb{E}_{d_T^{\pi_D}(s,a,s')}\left[\log 1 - c(s,a,s')\right] \quad (27)$$

The optimal discriminator is $c^\star(s,a,s') = \frac{d^\pi_T(s,a,s')}{d^\pi_T(s,a,s') + d^{\pi_D}_T(s,a,s')}$ (Goodfellow et al., 2014), so we can use $r(s,a,s') = -\log\left(\frac{1}{c^\star(s,a,s')} - 1\right)$.

Appendix B. Pseudocode

Algorithm 2 Transition Occupancy Matching for continuous control

Initialize policy π_ϕ, predictive model p_ω,discriminator c_ψ, Q function Q_θ, environment dataset $\mathcal{D}_{\text{env}}$, model dataset $\mathcal{D}_{\text{model}}$, Current policy pool dataset $\mathcal{D}_{\text{pol}}$,choice of f-divergence f
for N epochs **do**
  `// Train Expert Discriminator`
  Train Discriminator c_ψ using $\mathcal{D}_{\text{pol}}$ and $\mathcal{D}_{\text{env}}$
  `// Train Lagrangian Q Function`
  for U iterations **do**
    Sample minibatch data from environment pool $\{s_t^i, a_t^i, s_{t+1}^i\}_{i=1}^N \sim \mathcal{D}_{\text{env}}$ and $\{s_0^i\}_{i=1}^M \sim \mathcal{D}_{\text{env}}(\mu_0)$
    Obtain reward: $R_i = c_\psi(s_t^i, a_t^i, s_{t+1}^i), i = 1, ..., N$
    Compute value objective
    $\mathcal{L}(\theta) \coloneqq (1-\gamma)\frac{1}{M}\sum_{i=1}^M V_\theta(s_0^i) + \frac{1}{N} f_\star\left(R_i + \gamma V(s_{t+1}^i) - Q(s_t^i, a_t^i)\right)$
    where $V(s_{t+1}) = \frac{1}{P}\Sigma_p Q(s_{t+1}, a_{t+1}^p)$ where $a_{t+1}^p \sim \pi_\phi(s_{t+1})$
    Update Q_θ using SGD: $Q_\theta \leftarrow Q_\theta - \eta_Q \nabla \mathcal{L}(\theta)$
  `// Model Learning`
  for H iterations **do**
    `// Compute Optimal Importance Weights for all env pool samples`
    Compute $\xi^*(s_t^i, a_t^i, s_{t+1}^i) = f'_\star\left(R(s_t^i, a_t^i, s_{t+1}^i) + \gamma V(s_{t+1}^i) - Q(s_t^i, a_t^i)\right), i = 1, ..., N$
    Sample minibatch data from environment pool accoriding to the weights $\{s_t^i, a_t^i, s_{t+1}^i\}_{i=1}^N \sim \mathcal{D}_{\text{env}}$ according to $\xi^*(s_t^i, a_t^i, s_{t+1}^i)$
    Obtain reward: $R = c_\psi(s_t^i, a_t^i, s_{t+1}^i), i = 1, ..., N$
    `// Weighted Regression`
    Train dynamics model p_ω on sampled minibatch
  for E steps **do**
    Take action in environment according to π_ϕ; add to $\mathcal{D}_{\text{env}}$
    for M model rollouts **do**
      Sample s_t according to weights $\xi^*(s_t^i, a_t^i, s_{t+1}^i)$ from $\mathcal{D}_{\text{env}}$
      Perform k-step model rollout starting from s_t using policy π_ϕ; add to $\mathcal{D}_{\text{model}}$
    for G gradient updates **do**
      Update policy parameters on model data: $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi, \mathcal{D}_{\text{model}})$
  Update $\mathcal{D}_{\text{pol}}$ for the current trained policy

Appendix C. Implementation Details

C.1. Hyperparameters and Architecture

We standardize hyperparameters across all experiments and environments; they are listed in Table C.1.

In terms of network architectures, the dynamics value function Q_θ is implemented as a simple 2-layered ReLU network each with 256 neurons in the hidden dimension. The discriminator c_ψ has the same architecture as the value function but with Tanh as its activation. The dynamics model p_ω

is a sigmoid activated 4 layer neural network with 200 hidden neurons in each layer. In Humanoid, we use 400 hidden neurons in each layer for the dynamics model.

We employ Soft-Actor-Critic (SAC) (Haarnoja et al., 2018) for policy optimization and use default architecture in a publicly released implementation.

Hyperparameter	Value
Optimizer	Adam Kingma and Ba (2014)
Learning Rate	3e-4
Divergence	χ^2-divergence
Discriminator update steps per iteration	100
Discriminator batch size	256
Value network update steps per iteration	1000
Value network batch size	256
Dynamics model update steps per iteration	30
Dynamics model batch size	256
Policy network batch size	256
Current policy buffer capacity	1000
Replay buffer capacity	1000000
Rollout batch size	100000
Model rollout step(s)	1
PMAC decay rate	0.996

C.2. PMAC

In this section we describe the PMAC (Wang et al., 2022) baseline implementation details. Like (Wang et al., 2022), we employ a sampling based regression approach, where the importance weights for the replay buffer are interpreted as the probability mass for each sample. These importance weights are assigned via an exponential up-weighting heuristic that pays more attention to recently collected transitions. Specifically, in this implementation the transitions collected by historical data are decayed by 0.996 while the transitions collected by the current policy rollouts share weight of 0.004 (1 - 0.996). We implement PMAC on the same base MBPO implementation that TOM is built on, providing fair comparison of the different model learning approaches.

C.3. Current policy buffer

We approximate the current policy buffer with the last 1000 transitions collected by the agent in the environment. This circumvents the need of collecting extra data by employing the current policy.

Appendix D. Online progression

To demonstrate that TOM does not only pay attention to recent transitions, we capture the importance weights calculated by TOM and plot them with respect to the sequential order of collected transitions. The y-axis represents the TOM importance weights and the x-axis is sequential order of replay buffer transitions averaged in 100 bins

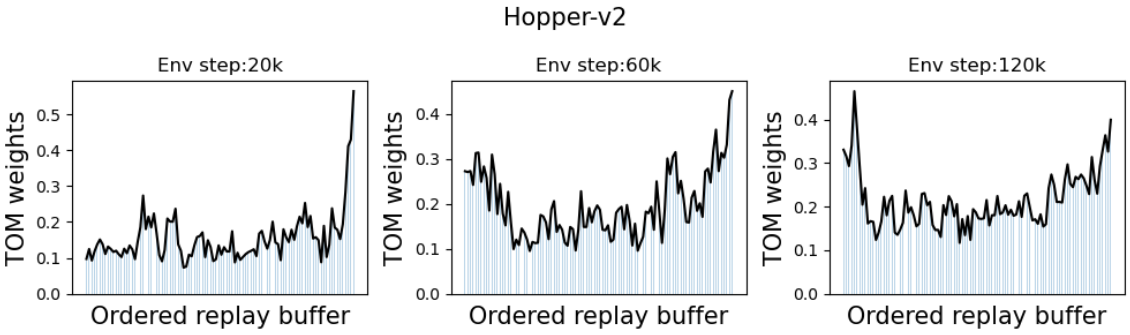

Figure 5: Hopper online buffer weights - total steps (125k)

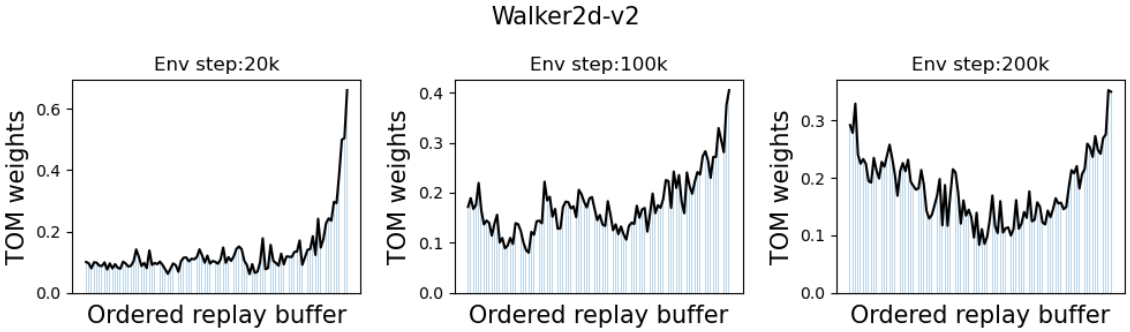

Figure 6: Walker online buffer weights- total steps (300k)

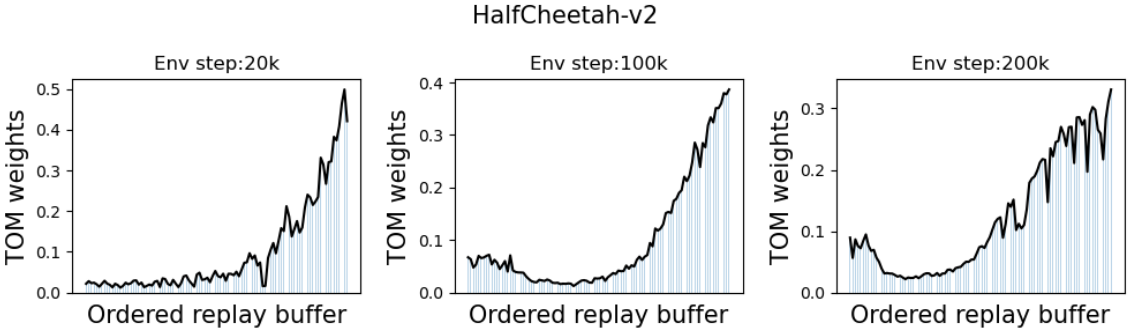

Figure 7: HalfCheetah online buffer weights- total steps (300k)

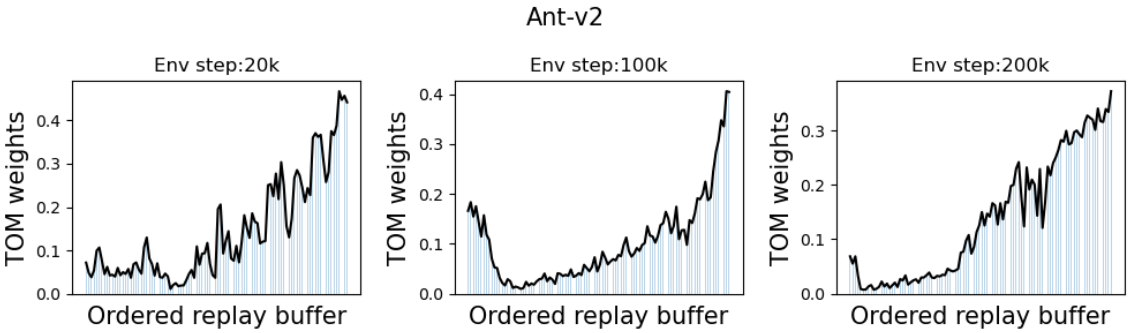

Figure 8: Ant online buffer weights- total steps (300k)

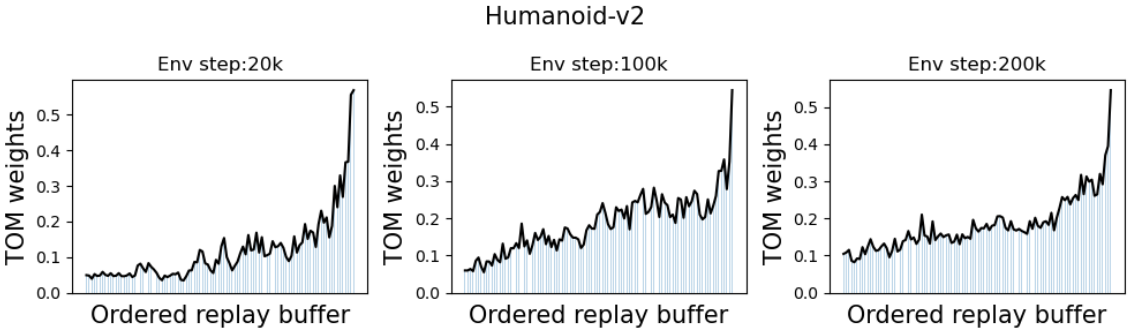

Figure 9: Humanoid online buffer weights- total steps (300k)

These results show that TOM's importance weights are not high only at the very end of the replay buffer where data is near-optimal. It often pays attention to transitions that were collected at earlier stages. This trend is clearly noticeable in Figure 5 (step 120k) and Figure 6 (step 200k), where TOM pays more pays more attention to earlier transitions in the replay buffer, clearly suggesting that paying attention to recent transitions is not the only relevant factor for learning a policy-aware dynamics model.