Policy Aware Model Learning via Transition Occupancy Matching

Jason Yecheng Ma*, Kausik Sivakumar*, Osbert Bastani, Dinesh Jayaraman
University of Pennsylvania

Abstract

Model-based reinforcement learning (MBRL) is an effective paradigm for sample-efficient policy learning. The pre-dominant MBRL strategy iteratively learns the dynamics model by performing maximum likelihood (MLE) on the entire replay buffer and trains the policy using fictitious transitions from the learned model. Given that not all transitions in the replay buffer are equally informative about the task or the policy's current progress, this MLE strategy cannot be optimal and bears no clear relation to the standard RL objective. In this work, we propose Transition Occupancy Matching (TOM), a policy-aware model learning algorithm that maximizes a lower bound on the standard RL objective. TOM learns a *policy-aware* dynamics model by minimizing an f-divergence between the distribution of transitions that the current policy visits in the real environment and in the learned model; then, the policy can be updated using any pre-existing RL algorithm with log-transformed reward. TOM's practical implementation builds on tools from dual reinforcement learning and learns the optimal transition occupancy ratio between the current policy and the replay buffer; leveraging this ratio as importance weights, TOM amounts to performing MLE model learning on the correct, policy aware transition distribution. Crucially, TOM is a model learning *sub-routine* and is compatible with any backbone MBRL algorithm that implements MLE-based model learning. On the standard set of Mujoco locomotion tasks, TOM is more sample efficient and achieves higher asymptotic performance.

1 Introduction

Model-based reinforcement learning (Sutton, 1991) is an effective paradigm for sample-efficient policy learning. By learning a dynamics model using the agent's collected experiences, the dynamics model can provide fictitious transitions for policy learning, reducing the required number of transitions in the true environment for learning an effective policy. This advantage of MBRL, coupled with breakthroughs in deep neural networks, has 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 & Levine, 2017; Ebert et al., 2018).

Despite these impressive empirical results, the fundamental question of how to best learn a dynamics model remains open. The predominant approach in model learning is to perform supervised learning, such as maximum likelihood (MLE) estimation, on the entire replay buffer the agent has collected (Chua et al., 2018; Janner et al., 2019). Given that not all transitions in the replay buffer are equally informative about the task or the policy[1]'s current progress, this MLE strategy cannot be optimal and bears no clear relation to the standard RL objective. As a simple thought experiment, let us consider when the replay buffer is pre-populated with a very large amount of transitions collected via random actions. MLE model learning would focus on fitting the dynamics on these random transitions well due to their sheer collective size and practically ignore any agent-specific experience collected later on. As such, though the MLE loss may be low, the learned dynamics model may not be useful for the purpose of *policy learning*, which is the ultimate objective we care about in any reinforcement learning approach. This apparent lack of synergy between model learning and policy learning is broadly known as model *objective mismatch* (Lambert et al., 2020) in the literature.

[1]We use policy and agent interchangably.

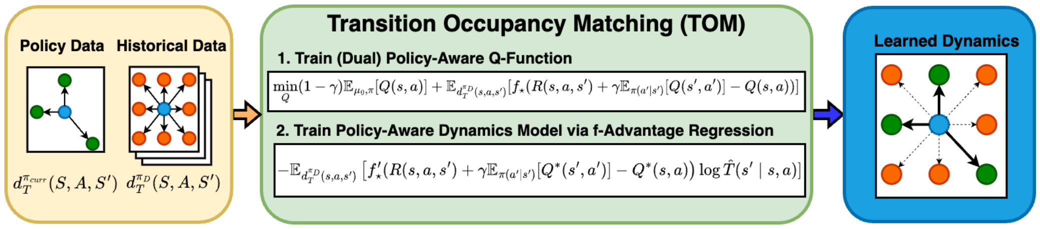
Policy Data
Historical Data
$d_T^{\pi_{curr}}(S, A, S')$
$d_T^{\pi_D}(S, A, S')$
Transition Occupancy Matching (TOM)
1. Train (Dual) Policy-Aware Q-Function
$\min_Q (1-\gamma)\mathbb{E}_{\mu_0,\pi}[Q(s,a)] + \mathbb{E}_{d_T^{\pi_D}(s,a,s')}[f_\star(R(s,a,s') + \gamma\mathbb{E}_{\pi(a'|s')}[Q(s',a')] - Q(s,a))]$
2. Train Policy-Aware Dynamics Model via f-Advantage Regression
$-\mathbb{E}_{d_T^{\pi_D}(s,a,s')}\left[f'_\star(R(s,a,s') + \gamma\mathbb{E}_{\pi(a'|s')}[Q^*(s',a')] - Q^*(s,a)) \log \hat{T}(s' \mid s,a)\right]$
Learned Dynamics

Figure 1: Transition Occupancy Matching (TOM). Given the current policy's rollout(s) and the replay buffer, TOM reduces policy-aware model learning to an offline imitation learning problem, and enables learning a dynamics model that fits the policy's visitation distribution accurately to enable rapid policy improvement.

In this work, we propose Transition Occupancy Matching (TOM), a principled solution to model objective mismatch by focusing the model capacity on learning the transition distribution of the *current policy*. More specifically, TOM proposes a novel lower bound to the reinforcement learning objective that decomposes into (1) a f-divergence between the distribution of transitions that the current policy visits in the real environment and in the learned model, and (2) the RL objective with respect to the learned model. Since (2) can be implemented by any MBRL algorithm, TOM's technical contribution is a practical algorithm for minimizing (1) based on the idea of occupancy measure matching (Ghasemipour et al., 2019; Ma et al., 2022b). At a technical level, TOM introduces the concept of transition occupancy, which bears structural similarity to the notion of state occupancy commonly seen in the imitation learning literature (Ho & Ermon, 2016; Ghasemipour et al., 2019; Kim et al., 2022; Ma et al., 2022b). We derive several properties of transition occupancy that find equivalents in state occupancy. This equivalence makes the intuition that policy-aware model learning can be cast as an imitation learning apparent. In particular, the expert can be thought of the distribution of transitions that the current policy visits, and the dynamics model is the imitating agent, whose "state" is a concatenation of the current state and action, and the "action" is the next state.

Given this intuition, learning a dynamics model tailored to the current policy can be achieved without any additional data collection by performing *offline* imitation learning (Kim et al., 2022; Ma et al., 2022b), for which the replay buffer effectively plays the role of the offline dataset. In particular, TOM uses f-advantage regression (Ma et al., 2022b;d;c) to derive a stable routine for learning the optimal importance weights for weighing the replay buffer data to correctly simulate sampling from transitions from the current policy. Using these importance weights, TOM trains a policy-aware model by performing weighted regression on the replay buffer data according to the importance weights. As such, TOM is by-design policy-aware and focuses the dynamics' model's capacity on the portions of the replay buffer that is most relevant to the *current* policy and its improvement in performance using the model. See Figure 1 for an overview.

Since TOM modifies only the transition weights for MLE-based model learning, it is modular in nature and is compatible with any underlying MBRL algorithm. In practice, we implement TOM on top of a standard deep MBRL algorithm, MBPO (Janner et al., 2019). By changing only the regression weights for model learning, TOM realizes substantial gain in Mujoco locomotion tasks with improved sample efficiency and asymptotic performance. These results suggest that MLE-based model learning is not inherently the culprit of model objective mismatch, but it is rather the correct transition distribution that we need to capture in order to make MLE-based model learning both sample-efficient and asymptotically optimal.

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 idea is conservative in nature as the value-weighted model loss is taken over the supremum/average over all, or a large hypothesis class of, value functions (Farahmand et al., 2017), which is intractable to compute in practice. Various approximations , such as using the current policy's value function, are implemented (Farahmand, 2018; Voelcker et al., 2022) in practice. The key shortcoming of VAML is the inherent gap between its theory and practice and the assumption that the environment comes with dense reward so 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 the 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

In this section, we will first go over the preliminaries for model-based reinforcement learning and then discuss the concept of state-action occupancy. With these two concepts as preface, we introduce *transition occupancy*, which leverages the notion of state-action occupancy to build a dynamics model that is policy aware.

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's 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:

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

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.

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:

$$\mathbb{E}_{D(s,a,s')} D_{\mathrm{KL}} \left(T(\cdot \mid s, a) \| \hat{T}(\cdot \mid s, a)\right) \tag{2}$$

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

$$d^\pi_T(s, a) := (1 - \gamma) \sum_{t=0}^{\infty} \gamma^t \Pr(s_t = s, a_t = a \mid s_0 \sim \mu_0, a_t \sim \pi(s_t), s_{t+1} \sim T(s_t, a_t)) \tag{3}$$

This captures the relative state-action visitation frequencies of policy π under dynamics T. 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:

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

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

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

Intuitively, this constraint restricts the "flow" of a policy's state-action distribution where each $d^\pi(s,a)$ must be expressed as a weighted sum. This is known in the literature as the *Bellman flow* constraint. 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:

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

Given d^π, one can express the RL objective as

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

Extending this, we incorporate the Bellman flow constraint in order to ensure that the optimized occupancy distribution corresponds to that of some policy π. Thus, we arrive at the optimization problem:

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

4 Transition Occupancy Matching

In this section, we first develop the machinery of transition occupancy. Then, we derive a lower bound to the standard RL objective by recasting the problem to the transition occupancy space, which that suggests a novel approach to model learning via matching the distribution of policy transitions in the true and learned dynamics. Then, we provide the intuition for casting policy-aware model learning as an offline imitation learning problem. Finally, we derive TOM in full.

4.1 Transition Occupancy

Given a policy's state-action occupancy distribution $d^\pi_T(s,a)$ under a transition function T, we can define its *transition occupancy distribution* (TOD):

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

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

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

These definitions are entirely analogous to equation 3-4, suggesting that if our goal were matching transition occupancy distributions, then we may be able to borrow algorithms from state occupancy matching. The one missing piece so far is the equivalent of a Bellman flow constraint (equation 6), which allows us to pose an optimization problem in the TOD space. Our key observation is that the original Bellman flow constraint 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 equation 6 by $T(s' \mid s,a)$, we obtain the *Bellman transition flow* constraint:

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

Given these definitions, we can begin deriving a TOD-based lower bound to the RL objective that enables policy-aware model learning via transition occupancy matching.

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

Consider the RL objective $J(\pi)$, we can write it as

$$J(\pi) = \mathbb{E}_{d_T^\pi(s,a)}[R(s,a)] = \mathbb{E}_{d_T^\pi((s,a),s')}[R(s,a)] \tag{12}$$

Then, since $\log(\cdot)$ is a monotonically increasing function, the optimal policy to $J(\pi)$ and $\log J(\pi)$ is identical. The latter formulation enables us to derive a simple lower bound with respect to transition occupancy distributions:

$$\begin{aligned} \log J(\pi) &= \log \mathbb{E}_{d_T^\pi((s,a),s')}[R(s,a)] \\ &= \log \mathbb{E}_{d_{\hat{T}}^\pi((s,a),s')}\left[\frac{d_T^\pi((s,a),s')}{d_{\hat{T}}^\pi((s,a),s')} R(s,a)\right] \\ &\geq \mathbb{E}_{d_{\hat{T}}^\pi((s,a),s')}\left[\log \frac{d_T^\pi((s,a),s')}{d_{\hat{T}}^\pi((s,a),s')} + \log R(s,a)\right], \quad \text{Jensen's inequality} \\ &\geq -\mathrm{D}_f(d_{\hat{T}}^\pi((s,a),s') \| d_T^\pi((s,a),s')) + \mathbb{E}_{d_{\hat{T}}^\pi((s,a),s')}[\log R(s,a)], \end{aligned} \tag{13}$$

which hold for any f-divergence that upper bounds the KL divergence. This lower bound directly suggests a recipe for MBRL: (1) trains the dynamics model by minimizing the f-divergence between the distributions of real and fake policy rollouts, and (2) optimize the policy w.r.t. the learned model. The second step is standard and we can utilize any existing MBRL algorithm, and TOM's technical contribution is a model learning sub-routine aimed at minimizing this f-divergrence:

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

4.3 Algorithm

At a high level, TOM uses the current policy π and its rollouts τ to train a dynamics 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. A pseudocode is provided in Alg. 1, and we begin by illustrating the intuition of this approach. Then, we delve into the technical derivations.

Algorithm 1 Transition Occupancy Matching

```
1: **Require**: current policy $\pi$ and its environment rollout(s) $\tau$, replay buffer $D$
2: // Discriminator Learning
3: Train discriminator $c^*(s,a,s')$ using $\tau, D$ (equation 21) and derive $R(s,a,s')$.
4: // Q-Function Learning
5: Train derived value function $Q(s,a)$ using equation 22
6: // Model Learning
7: Derive optimal ratios $\xi^*(s,a,s')$ through equation 23
8: Train policy-aware dynamics model $\hat{T}$ using equation 25
```

To begin, we put the transition occupancy matching problem and the well-known state-action occupancy matching problem (Nachum & Dai, 2020; Ma et al., 2022b; Kim et al., 2022) side-by-side:

$$\min_{\hat{T}} \mathrm{D}_f(d_{\hat{T}}^\pi((s,a),s') \| d_T^\pi((s,a),s'), \qquad \min_{\pi} \mathrm{D}_f(d_T^\pi((s,a)) \| d_T^{\pi^*}((s,a)) \tag{15}$$

In the latter case, the dynamics 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 *reverse* the role of the policy π and the dynamics model T. In particular, we can think of T as a *policy*, which takes in "state" (s,a) and outputs an "action" s', and the *fixed* policy π as a *partial* transition function that takes s and outputs a. Given this conceptual similarity, it is perceivable that we can derive a policy-aware model learning algorithm by adapting a suitable state-occupancy based imitation learning problem.

To this end, we first observe that optimizing equation 14 requires sampling from $d_{\hat{T}}^\pi((s,a),s')$ and optimizing $\hat{T}$ via RL, which can be unstable and fails to leverage the replay buffer D that the agent has already collected. To get around this issue, we introduce an additional f-divergence regularization term as in Ma et al. (2022b) to enable learning $\hat{T}$ without collecting any additional simulated samples from it:

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

We can write down the Lagrangian-dual problem:

$$\begin{aligned}\max_{d^\pi_{\hat{T}}} \min_Q & -\mathrm{D}_f(d^\pi_{\hat{T}}((s,a),s') \| d^\pi_T((s,a),s')) - \mathrm{D}_f(d^\pi_{\hat{T}}((s,a),s') \| d^{\pi_D}_T((s,a),s')) \\ & + \sum_{s,a} Q(s,a)\left((1-\gamma)\mu_0(s)\pi(a \mid s) + \gamma\pi(a \mid s)\sum_{\tilde{s},\tilde{a}} d^\pi_{\hat{T}}((\tilde{s},\tilde{a}),s) - \sum_{s'} d^\pi_{\hat{T}}((s,a),s')\right)\end{aligned} \tag{17}$$

We note two identities:

$$\sum_{s,a} Q(s,a)\left(\sum_{s'} d^\pi_{\hat{T}}((s,a),s')\right) = \mathbb{E}_{d^\pi_T((s,a),s')}[Q(s,a)] \tag{18}$$

and

$$\sum_{s,a} Q(s,a)\left(\gamma\pi(a \mid s)\sum_{\tilde{s},\tilde{a}} d^\pi_{\hat{T}}((\tilde{s},\tilde{a}),s)\right) = \gamma\mathbb{E}_{d^\pi_{\hat{T}}((s,a),s')}\mathbb{E}_{\pi(a'|s')}[Q(s',a')] \tag{19}$$

Using these identities, some algebra, and strong duality (assuming Slater's condition holds), we get

$$\begin{aligned}\min_Q \max_{d^\pi_{\hat{T}} \geq 0} (1-\gamma)\mathbb{E}_{\mu_0,\pi}[Q(s,a)] + \mathbb{E}_{d^\pi_{\hat{T}}(s,a,s')}&\left[\underbrace{\log\frac{d^\pi_T(s,a,s')}{d^{\pi_D}_T(s,a,s')}}_{:=R(s,a,s')} + \gamma\mathbb{E}_{\pi(a'|s')}[Q(s',a')] - Q(s,a)\right] \\ - \mathrm{D}_f(d^\pi_{\hat{T}}((s,a),s') \| d^{\pi_D}_T((s,a),s'))&\end{aligned} \tag{20}$$

$R(s,a,s')$ can be estimated by training a discriminator c that distinguishes transitions from π and π_D: In the continuous case, we can train a discriminator $c : \mathcal{S} \to (0,1)$:

$$\min_c \mathbb{E}_{(s,a,s') \sim d^\pi_T(s,a,s')}\left[\log c(s,a,s')\right] + \mathbb{E}_{d^{\pi_D}_T(s,a,s')}\left[\log 1 - c(s,a,s')\right] \tag{21}$$

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 $R(s,a,s') = -\log\left(\frac{1}{c^\star(s,a,s')} - 1\right)$.

Then, we recognize that the inner maximization is precisely the Fenchel conjugate of $\mathrm{D}_f(d^\pi_{\hat{T}}((s,a),s') \| d^{\pi_D}_T((s,a),s'))$ at $R(s,a,s') + \gamma\mathbb{E}_{\pi(a'|s')}[Q^*(s',a')] - Q^*(s,a)$, which allows us to reduce equation 20 to the Fenchel *dual* problem of equation 16:

$$\min_Q (1-\gamma)\mathbb{E}_{\mu_0,\pi}[Q(s,a)] + \mathbb{E}_{d^{\pi_D}_T(s,a,s')}\left[f_\star(R(s,a,s') + \gamma\mathbb{E}_{\pi(a'|s')}[Q(s',a')] - Q(s,a)\right] \tag{22}$$

This Q-function intuitively captures how easy it is to distinguish π from π_D by rolling out respective policies in T. More importantly, the optimal Q^* can be approximated from equation 22 using stochastic gradient descent (SGD) on the replay buffer data D, as all expectations can be approximated using samples from D and the current policy π. Then, we can leverage Fenchel duality to obtain the optimal *importance weight ratio* incorporating the optimal solution $d^\pi_{\hat{T}^*}((s,a),s')$ for the primal problem (equation 16):

$$\xi^*(s,a,s') := \frac{d^\pi_{\hat{T}^*}((s,a),s')}{d^{\pi_D}_T((s,a),s')} = f'_\star\left(R(s,a,s') + \gamma\mathbb{E}_{\pi(a'|s')}[Q^*(s',a')] - Q^*(s,a)\right) \tag{23}$$

Using this, we can also back out the optimal policy-aware dynamics model:

$$\hat{T}^*(s' \mid s,a) = \frac{d_T^{\pi_D}((s,a),s')f'_\star\left(R(s,a,s') + \gamma\mathbb{E}_{\pi(a'|s')}[Q^*(s',a')] - Q^*(s,a)\right)}{\sum_{s'} d_T^{\pi_D}((s,a),s')f'_\star\left(R(s,a,s') + \gamma\mathbb{E}_{\pi(a'|s')}[Q^*(s',a')] - Q^*(s,a)\right)} \tag{24}$$

In the continuous control regime where we cannot sum over all possible s', we can resort to using equation 23 as importance weights and perform f-advantage regression (Ma et al., 2022d):

$$\begin{aligned}&\min_{\hat{T}} -\mathbb{E}_{d^{\pi}_{\hat{T}^*}((s,a),s')}[\log \hat{T}(s' \mid s,a)] \\ &= -\mathbb{E}_{d_T^{\pi_D}(s,a,s')}\left[f'_\star\left(R(s,a,s') + \gamma\mathbb{E}_{\pi(a'|s')}[Q^*(s',a')] - Q^*(s,a)\right)\log\hat{T}(s' \mid s,a)\right]\end{aligned} \tag{25}$$

Here, we see that the dynamics model is still trained with MLE, but just on a different distribution that is current *policy-aware*; equivalently, the model learned via *behavior cloning* (BC) on the current policy's transition occupancy distribution, which ensures policy-awareness and enables optimizing a well-defined lower bound to the true RL objective. The TOM algorithm is summarized in Alg. 1, and a full version is Alg. 2.

5 Experiments

We experimentally demonstrate that TOM is an effective way of model learning by providing conclusive evidence for two questions

- Does TOM enable more sample-efficient MBRL?
- Does TOM's principled lower bound permits higher asymptotic performance?
- Does TOM assign higher weights to samples from the current policy's transition distribution?

To investigate the first question, we build TOM on top of a standard deep MBRL algorithm which implements MLE model learning. To answer the second question, we design an *offline* model learning experiment. Here, through tight control of transitions in the offline dataset, we probe whether TOM can accurately assign higher weights to transitions more relevant to a hand-designed policy.

5.1 Online Model Based Learning

5.1.1 Algorithms and Environments

We begin by describing baselines and environments.While many decision-aware MBRL algorithms have been introduced in recent years, we emphasize that a thorough comparison to them is not the aim of our paper; many of them require specific network architectures and introduce fundamentally new ways for training the model and the policy, thus making a direct comparison difficult. Instead, our goal is to compare our policy-aware model learning approach over a standard deep MBRL algorithm that is not policy-aware To this end, we consider Model Based Policy Optimization (MBPO) (Janner et al., 2019) as the backbone MBRL algorithm. MBPO learns the dynamics model by maximum likelihood and optimizes the policy via Soft Actor-Critic (SAC) using simulated samples from the learned dynamics model. To validate the effective of our occupancy matching approach to policy-aware model learning, we compare against an alternative policy-aware model learning algorithm, Policy-adaptation Model-based Actor-Critic (PMAC) (Wang et al., 2022), which heuristically upweights recent transitions in the replay buffer because they are more likely to be relevant to the current policy. We implement both TOM and PMAC on top of the same MBPO implementation, changing only how the weights are assigned to replay buffer data for model learning. See Appendix **??** for more implementation details. In Appendix A.1, we also provide TOM pseudocode.

We compare TOM, MBPO and PMAC on 5 different Mujoco (Todorov et al., 2012) environments (Hopper-v2, Walker2d-v2, HalfCheetah-v2, Ant-v2 and Humanoid-v2), and the results are shown in Figure 2. The x-axis refers to the total number of environment step taken by the agent, and the y-axis is the cumulative maximum average return on 10 test rollouts.

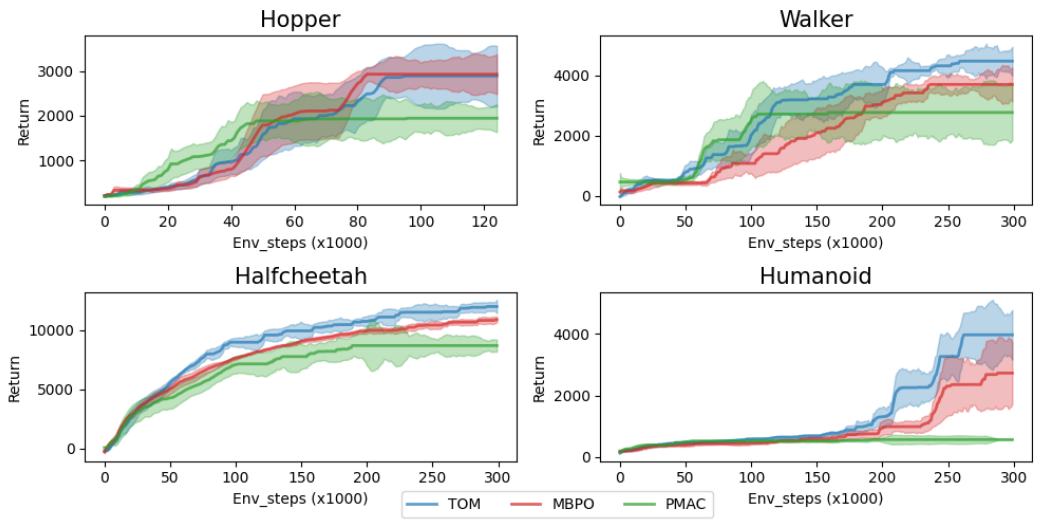

Figure 2: **Return plots on standard Mujoco GYM environments**

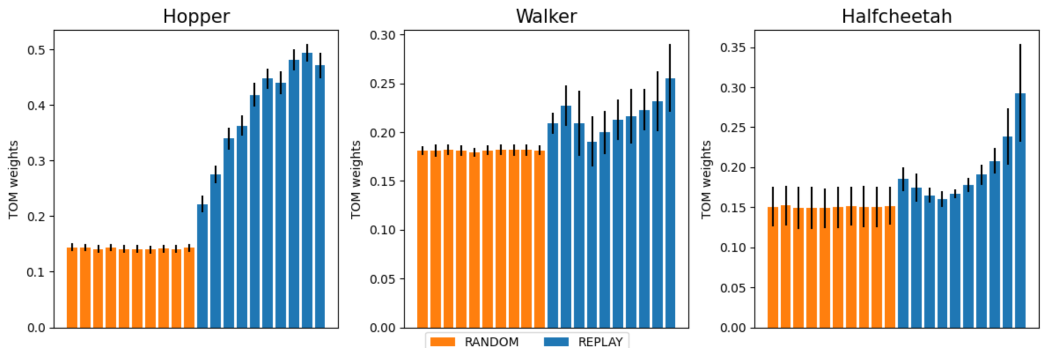

Figure 3: **TOM transition weights** - TOM progressively assigns higher weights to expert policy's transitions

5.1.2 Results

As shown in Figure 2, TOM is more sample efficient and reaches higher asymptotic performance in all environments. This is better noticed in the harder environments such as Walker, HalfCheetah, and Humanoid. Notably on Humanoid, TOM learns a policy that achieves at least 60% more reward than non-policy aware MBPO. In contrast, PMAC is unstable and reaches much lower asymptotic performance, indicating that heuristically upweighting transitions is sensitive to hyper-parameter choices, and a more principled approach like TOM is needed.

5.2 Offline Progression analysis of model weights

These online RL results in the previous section demonstrates that TOM's principled policy-aware model learning approach can indeed deliver impressive practical improvement. In this section, we qualitatively demonstrate that TOM's policy-aware nature comes from its ability to accurately assign weights to transitions according to their relevance to the current policy's visitation distribution.

5.2.1 Experiment Design

We consider a fully offline model learning setup, in which we carefully control the composition of transitions in the offline dataset for model learning. This enables us to visualize the learned TOM weights accordingly to see whether they meaningfully correlate with relevance to the hand-designed policy. To this end, we consider offline datasets from D4RL (Fu et al., 2021) and construct a combined dataset, in which the first half consists of 1M random transitions and the second half consists of all transitions encountered by a SAC agent (i.e., its replay buffer) trained using 1M environment steps, improving from random to expert performance. We choose the expert policy to be the policy of interest. Given that the first half of the combined dataset is random and the second half progressively become more similar to the expert, if we plot TOM's weights sequentially over the entire dataset, we should expect the weights to be uniformly low the first half and progressively increase in the second half. The results are shown in Figure 3; the bins are averaged over 100000 transition weights sequentially in the order they appear in the dataset.

5.2.2 Result

As shown in Figure 3, the TOM weights indeed progressively increase in the replay buffer portion (i.e., the blue bins) and remains uniformly low in the random portion (i.e., the orange bins). Interestingly, TOM is able to meaningful assign weights of intermediate values for transitions that are of intermediate qualities (compared to the expert); this shows that TOM also does not overfit to the policy of interest and is capable of learning a generalizable notion of policy awareness. These results indeed validate that TOM is able to pay more attention to transitions relevant to the policy of interest, and consequently, learn a policy-aware dynamics model.

6 Conclusion

We have introduced Transition Occupancy Matching (TOM), a principled and 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 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-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. Future directions include scaling TOM to visual MBRL domains (Hafner et al., 2019a;b), real-world robotics tasks (Ebert et al., 2018; Nagabandi et al., 2020), and safe deep model-based RL (Liu et al., 2020; Ma et al., 2022a).

Contribution Statement

JYM conceived the project idea, proposed experiments, and wrote the paper. KS conducted all experiments and assisted on the paper writing. OB and DJ provided feedback and guidance on the project.

References

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

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*, pp. 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)*, pp. 2786–2793. IEEE, 2017.

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

Seyed Kamyar Seyed Ghasemipour, Richard Zemel, and Shixiang Gu. A divergence minimization perspective on imitation learning methods, 2019.

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*, pp. 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.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning, 2016.

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.

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.

Zuxin Liu, Hongyi Zhou, Baiming Chen, Sicheng Zhong, Martial Hebert, and Ding Zhao. Constrained model-based reinforcement learning with robust cross-entropy method. *arXiv preprint arXiv:2010.07968*, 2020.

Yecheng Jason Ma, Andrew Shen, Osbert Bastani, and Jayaraman Dinesh. Conservative and adaptive penalty for model-based safe reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 5404–5412, 2022a.

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

Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2022c.

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*, 2022d.

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*, pp. 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*, pp. 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`.

A Appendix

A.1 Pseudocode

Algorithm 2 Transition Occupancy Matching for continuous control

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

A.2 Hyperparameters and Architecture

We standardize hyperparameters across all experiments and environments; they are listed in Table A.2.

In terms of architecture, 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 & 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	10
Dynamics model batch size	256
Policy network batch size	256
Current policy buffer capacity	1000
Replay buffer capacity	100000
Rollout batch size	100000
Model rollout step(s)	1
PMAC decay rate	0.996