Supplemental Materials for: Robust Inverse Reinforcement Learning Through Bayesian Theory of Mind

A Additional Related Work

Bayesian IRL Ramachandran and Amir [57] first proposed a Bayesian formulation of IRL to solve the reward ambiguity problem. A MAP inference approach was proposed in [58] and a variational inference approach was proposed in [59]. Their formulations considers non-entropy-regularized policies and the dynamics model is fixed during reward inference. In contrast, simultaneous estimation of reward and dynamics can potentially infer the demonstrator's biased beliefs about the environment, which is desirable for psychology and human-robot interaction studies [15, 19, 17]. Despite the attractiveness, simultaneous estimation is challenging because of the need to invert the agent's planning process, especially in continuous domains. Reddy et al. [17] avoids this by representing agent discrete action policies using neural network-parameterized Q functions and regularizing the Bellman error to be small over the entire state-action space. This method however cannot be straightforwardly adapted to the continuous action case. Kwon et al. [60] avoids this by first training a task-conditioned policy on a distribution of environments with known parameters using meta reinforcement learning and then use the meta-trained policy to guide inference. This precludes the method from being used in general settings with unknown task distributions. To our knowledge, our proposed algorithms are the first to address simultaneous estimation in general environments.

Decision-aware model learning Decision-aware model learning aims to solve the objective mismatch problem in model-based RL [61]. Many proposed methods in this class use value-targeted regression similar to our model loss in (16) [62, 63]. Our analysis and that of Vemula et al. [52] suggest that value-targeted model objectives may be related to robust objectives. Furthermore, since the set of value-equivalent models only shrink for increasingly larger set of policy and values [62], using value-aware model objective alone may not be optimal and additional prediction-based regularizations may be needed.

B Missing Proofs

B.1 Proofs For Section 3.1

Derivation of BTOM Gradients (section 3.1). Recall the definition of the optimal entropy-regularized policy and value functions:

$$\begin{aligned}\hat{\pi}(a|s;\theta) &= \frac{\exp(Q_\theta(s,a))}{\sum_{\tilde{a}}\exp(Q_\theta(s,\tilde{a}))}\\ Q_\theta(s,a) &= R_{\theta_1}(s,a) + \gamma \mathbb{E}_{\hat{P}_{\theta_2}(\cdot|s,a)}[V_\theta(s')]\\ V_\theta(s) &= \log\sum_{\tilde{a}}\exp(Q_\theta(s,\tilde{a}))\end{aligned} \tag{20}$$

The gradient of the policy log likelihood in terms of the Q function gradient is obtained as follow:

$$\begin{aligned}\nabla_\theta \log\hat{\pi}(a|s;\theta) &= \nabla_\theta Q_\theta(s,a) - \nabla_\theta V_\theta(s)\\ &= \nabla_\theta Q_\theta(s,a) - \frac{1}{Z_\theta}\nabla_\theta\sum_{\tilde{a}}\exp(Q_\theta(s,\tilde{a}))\\ &= \nabla_\theta Q_\theta(s,a) - \frac{1}{Z_\theta}\sum_{\tilde{a}}\exp(\nabla_\theta Q_\theta(s,\tilde{a}))\\ &= \nabla_\theta Q_\theta(s,a) - \mathbb{E}_{\tilde{a}\sim\hat{\pi}(\cdot|s;\theta)}[\nabla_\theta Q_\theta(s,\tilde{a})]\end{aligned} \tag{21}$$

where $Z_\theta = \sum_{a'}\exp(Q_\theta(s,a'))$ is the normalizer.

Recall $\rho^{\hat{\pi}}_{\hat{P}}(\tilde{s}, \tilde{a}|s, a)$ is the discounted state-action occupancy measure starting from pair (s, a). We define for any function $f(s, a)$:

$$\mathbb{E}_{\rho^{\hat{\pi}}_{\hat{P}}(\tilde{s},\tilde{a}|s,a)}[f(s,a)] = \mathbb{E}_{\tau\sim(\hat{P},\hat{\pi})}\left[\sum_{t=0}^{\infty}\gamma^t f(s,a)\middle| s_0 = s, a_0 = a\right] \tag{22}$$

We now derive Q function gradients with respect to the reward parameters θ_1 and dynamics parameters θ_2, respectively.

$$\begin{aligned}
\nabla_{\theta_1} Q_\theta(s,a) &= \nabla_{\theta_1} R_{\theta_1}(s,a) + \gamma \mathbb{E}_{s'\sim\hat{P}_{\theta_2}(\cdot|s,a)}[\nabla_{\theta_1} V_\theta(s')] \\
&= \nabla_{\theta_1} R_{\theta_1}(s,a) + \gamma \mathbb{E}_{s'\sim\hat{P}_{\theta_2}(\cdot|s,a),a'\sim\hat{\pi}(\cdot|s';\theta)}[\nabla_{\theta_1} Q_\theta(s',a')] \\
&= \nabla_{\theta_1} R_{\theta_1}(s,a) + \gamma \mathbb{E}_{s'\sim\hat{P}_{\theta_2}(\cdot|s,a),a'\sim\hat{\pi}(\cdot|s';\theta)}\Big[\\
&\quad \nabla_{\theta_1} R_{\theta_1}(s',a') + \gamma \mathbb{E}_{s''\sim\hat{P}_{\theta_2}(\cdot|s',a'),a''\sim\hat{\pi}(\cdot|s'';\theta)}[\nabla_{\theta_1} Q_\theta(s'',a'')]\Big] \\
&= \nabla_{\theta_1} R_{\theta_1}(s,a) + \mathbb{E}_{\tau\sim(\hat{P},\hat{\pi})}\left[\sum_{h=1}^{\infty}\gamma^h \nabla_{\theta_1} R_{\theta_1}(s_h,a_h)\middle| s_0 = s, a_0 = a\right] \\
&= \mathbb{E}_{\rho^{\hat{\pi}}_{\hat{P}}(\tilde{s},\tilde{a}|s,a)}\left[\nabla_{\theta_1} R_{\theta_1}(\tilde{s},\tilde{a})\right]
\end{aligned} \tag{23}$$

In line two we used the result that $\nabla_\phi V_\phi(s)$ for both $\phi = \theta_1$ and $\phi = \theta_2$ corresponds to the second term in (21) .

$$\begin{aligned}
\nabla_{\theta_2} Q_\theta(s,a) &= \nabla_{\theta_2} R_{\theta_1}(s,a) + \nabla_{\theta_2}\gamma \mathbb{E}_{s'\sim\hat{P}_{\theta_2}(\cdot|s,a)}[V_\theta(s')] \\
&= \gamma \sum_{\tilde{s}} V_\theta(\tilde{s}) \nabla_{\theta_2} \hat{P}_{\theta_2}(\tilde{s}|s,a) + \gamma \mathbb{E}_{s'\sim\hat{P}_{\theta_2}(\cdot|s,a),a'\sim\hat{\pi}(\cdot|s';\theta)}[\nabla_{\theta_2} Q_\theta(s',a')] \\
&= \gamma \sum_{\tilde{s}} V_\theta(\tilde{s}) \nabla_{\theta_2} \hat{P}_{\theta_2}(\tilde{s}|s,a) + \gamma \mathbb{E}_{s'\sim\hat{P}_{\theta_2}(\cdot|s,a),a'\sim\hat{\pi}(\cdot|s';\theta)}\Big[\\
&\quad \gamma \sum_{\tilde{s}} V_\theta(\tilde{s}) \nabla_{\theta_2} \hat{P}_{\theta_2}(\tilde{s}|s',a') + \gamma \mathbb{E}_{s''\sim\hat{P}_{\theta_2}(\cdot|s',a'),a''\sim\hat{\pi}(\cdot|s'';\theta)}[\nabla_{\theta_2} Q_\theta(s'',a'')]\Big] \\
&= \gamma \sum_{\tilde{s}} V_\theta(\tilde{s}) \nabla_{\theta_2} \hat{P}_{\theta_2}(\tilde{s}|s,a) + \mathbb{E}_{\tau\sim(\hat{P},\hat{\pi})}\left[\sum_{h=1}^{\infty}\gamma^{h+1} \sum_{\tilde{s}} V_\theta(\tilde{s}) \nabla_{\theta_2} \hat{P}_{\theta_2}(\tilde{s}|s_h,a_h)\middle| s_0 = s, a_0 = a\right] \\
&= \mathbb{E}_{\rho^{\hat{\pi}}_{\hat{P}}(\tilde{s},\tilde{a}|s,a)}\left[\gamma \sum_{s'} V_\theta(s') \nabla_{\theta_2} \hat{P}_{\theta_2}(s'|\tilde{s},\tilde{a})\right]
\end{aligned} \tag{24}$$

We make a quick remark on the identifiability of simultaneous estimation.

Remark B.1. *Simultaneous reward-dynamics estimation of the form (5) without specific assumptions on the prior $P(\theta)$ is in general unidentifiable.*

Proof. Let $\mathbf{R} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ and $\mathbf{P} \in \mathbb{R}_+^{|\mathcal{S}||\mathcal{A}|\times|\mathcal{S}|}$, $\sum_{s'} \mathbf{P}^a_{ss'} = 1$, $\mathbf{Q} \in \mathbb{R}^{|\mathcal{S}||\mathcal{A}|}$ and $\mathbf{V} \in \mathbb{R}^{|\mathcal{S}|}$ be a set of Bellman-consistent reward, dynamics, and value functions in matrix form. Let $\mathbf{P'} \neq \mathbf{P}$ be an alternative dynamics model. We can always find an alternative reward $\mathbf{R'} = \mathbf{R} + \Delta\mathbf{R}$, where:

$$\begin{aligned}
\Delta\mathbf{R} &= (\mathbf{Q} - \mathbf{Q}) - \gamma(\mathbf{P'V} - \mathbf{PV}) \\
&= -\gamma\Delta\mathbf{PV}
\end{aligned} \tag{25}$$

without changing the value functions and optimal entropy-regularized policy. □

Remark B.1 implies that existing simultaneous estimation approaches which do not use explicit or implicit regularizations, such as the SERD algorithm by [18], cannot in general accurately estimate expert reward. Paired with theorem 3.1, it shows that these algorithms cannot in general achieve good performance.

B.2 Proofs For Section 3.2

Derivation of discounted likelihood (14).

$$
\begin{aligned}
&\mathbb{E}_{P(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t \log\hat{\pi}(a_t|s_t;\theta)\right] \\
&= \mathbb{E}_{P(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t \left(Q_\theta(s_t,a_t) - V_\theta(s_t)\right)\right] \\
&= \mathbb{E}_{P(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t \left(R_{\theta_1}(s_t,a_t) + \gamma\mathbb{E}_{s'\sim\hat{P}_{\theta_2}(\cdot|s_t,a_t)}[V_\theta(s')]\right)\right] - \mathbb{E}_{P(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t V_\theta(s_t)\right] \\
&= \mathbb{E}_{P(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t R_{\theta_1}(s_t,a_t)\right] - \mathbb{E}_{\mu}\Big[V_\theta(s_0)\Big] \\
&\quad + \mathbb{E}_{P(\tau)}\left[\sum_{t=0}^{\infty}\gamma^{t+1}\mathbb{E}_{s'\sim\hat{P}_\theta(\cdot|s_t,a_t)}[V_\theta(s')]\right] - \mathbb{E}_{P(\tau)}\left[\sum_{t=1}^{\infty}\gamma^t V_\theta(s_t)\right] \\
&= \mathbb{E}_{P(\tau)}\left[\sum_{t=0}^{\infty}\gamma^t R_{\theta_1}(s_t,a_t)\right] - \mathbb{E}_{\mu}\Big[V_\theta(s_0)\Big] \\
&\quad + \mathbb{E}_{P(\tau)}\left[\sum_{t=0}^{\infty}\gamma^{t+1}\mathbb{E}_{s'\sim\hat{P}_\theta(\cdot|s_t,a_t)}[V_\theta(s')]\right] - \mathbb{E}_{P(\tau)}\left[\sum_{t=0}^{\infty}\gamma^{t+1}\mathbb{E}_{s'\sim P(\cdot|s_t,a_t)}[V_\theta(s')]\right] \\
&= \underbrace{\mathbb{E}_{\rho^\pi_P}\Big[R_{\theta_1}(s_t,a_t)\Big] - \mathbb{E}_{\mu}\Big[V_\theta(s_0)\Big]}_{\ell(\theta)} + \underbrace{\gamma\mathbb{E}_{\rho^\pi_P}\Big[\mathbb{E}_{s'\sim\hat{P}_\theta(\cdot|s_t,a_t)}V_\theta(s') - \mathbb{E}_{s''\sim P(\cdot|s_t,a_t)}V_\theta(s'')\Big]}_{\textbf{T1}}
\end{aligned}
\tag{26}
$$

The following lemma shows that **T1** is negligible if the estimated dynamics is accurate under the *expert* distribution, which is available from the offline dataset.

Lemma B.2. *Let* $\epsilon = \mathbb{E}_{(s,a)\sim P(\tau)} D_{KL}(P(\cdot|s,a)||\hat{P}(\cdot|s,a))$ *and* $R_{max} = \max_{s,a}|R_\theta(s,a)| + \log|\mathcal{A}|$*, it holds that*

$$
|\textbf{T1}| \le \frac{\gamma R_{max}}{(1-\gamma)^2}\sqrt{2\epsilon} \tag{27}
$$

Proof.

$$
\begin{aligned}
|\textbf{T1}| &= \left|\sum_{t=0}^{\infty}\gamma^{t+1}\mathbb{E}_{(s_t,a_t)\sim P(\tau)}\left[\sum_{s'}V_\theta(s')\left(\hat{P}(s'|s_t,a_t) - P(s'|s_t,a_t)\right)\right]\right| \\
&\overset{(1)}{\le} \sum_{t=0}^{\infty}\gamma^{t+1}\mathbb{E}_{(s_t,a_t)\sim P(\tau)}\left[\sum_{s'}|V_\theta(s')|\left|\hat{P}(s'|s_t,a_t) - P(s'|s_t,a_t)\right|\right] \\
&\overset{(2)}{\le} \sum_{t=0}^{\infty}\gamma^{t+1}\|V_\theta(\cdot)\|_\infty\mathbb{E}_{(s_t,a_t)\sim P(\tau)}\left[\left\|\hat{P}(\cdot|s_t,a_t) - P(\cdot|s_t,a_t)\right\|_1\right] \\
&\overset{(3)}{\le} \sum_{t=0}^{\infty}\gamma^{t+1}\|V_\theta(\cdot)\|_\infty\sqrt{2\mathbb{E}_{(s_t,a_t)\sim P(\tau)}D_{KL}(P||\hat{P})} \\
&= \frac{\gamma}{1-\gamma}\|V_\theta(\cdot)\|_\infty\sqrt{2\epsilon}
\end{aligned}
$$

where (1) follows from Jensen's inequality, (2) follows from Holder's inequality, and (3) follows from Pinsker's inequality.

Finally, given $\mathcal{H}(\pi(a|s)) = -\sum_a \pi(a|s)\log\pi(a|s) \leq -\sum_a \pi(a|s)\log\frac{1}{|\mathcal{A}|} = \log|\mathcal{A}|$, we have $\|V_\theta(\cdot)\|_\infty \leq \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t\left(\max_{s,a}|R_\theta(s,a)| + \log|\mathcal{A}|\right)\right] = \frac{R_{\max}}{1-\gamma}$.

□

B.3 Proofs For Section 3.4

We first restate a slight modification of the result from [52], which decomposes the real environment performance gap between the expert and the learner into their policy and model advantages in the estimated dynamics:

Lemma B.3. *(Performance difference via advantage in model; Lemma 4.1 in [52]) Let d^π_P denote the marginal state-action distribution following policy π in environment P. The following relationship holds:*

$$\mathbb{E}_{(s,a)\sim d^\pi_P}\left[\log\hat{\pi}_{\hat{P}}(a|s)\right] = \mathbb{E}_{s\sim d^\pi_P}\left[\mathbb{E}_{a\sim\pi}Q^{\hat{\pi}}_{\hat{P}}(s,a) - V^{\hat{\pi}}_{\hat{P}}(s)\right] \tag{28}$$

$$= \underbrace{(1-\gamma)\mathbb{E}_{s\sim\mu}\left[V^\pi_P(s) - V^{\hat{\pi}}_P(s)\right]}_{\textit{Performance difference in real environment}} \tag{29}$$

$$+ \underbrace{\gamma\mathbb{E}_{(s,a)\sim d^{\hat{\pi}}_P}\left[\mathbb{E}_{s'\sim P}V^{\hat{\pi}}_{\hat{P}}(s') - \mathbb{E}_{s''\sim\hat{P}}V^{\hat{\pi}}_{\hat{P}}(s'')\right]}_{\textit{Model (dis)advantage under learner distribution}} \tag{30}$$

$$+ \underbrace{\gamma\mathbb{E}_{(s,a)\sim d^\pi_P}\left[\mathbb{E}_{s'\sim\hat{P}}V^{\hat{\pi}}_{\hat{P}}(s') - \mathbb{E}_{s''\sim P}V^{\hat{\pi}}_{\hat{P}}(s'')\right]}_{\textit{Model advantage under expert distribution}} \tag{31}$$

The performance bound in theorem 3.1 can be obtained from lemma B.3 as follow:

Theorem B.4. *(Restate of theorem 3.1) Let $\epsilon_{\hat{\pi}} = -\mathbb{E}_{(s,a)\sim d^\pi_P}[\log\hat{\pi}_{\hat{P}}(a|s)]$ be the policy estimation error and $\epsilon_{\hat{P}} = \mathbb{E}_{(s,a)\sim d^\pi_P}D_{KL}[P(\cdot|s,a)||\hat{P}(\cdot|s,a)]$ be the dynamics estimation error. Let $R_{max} = \max_{s,a}|R_\theta(s,a)| + \log|\mathcal{A}|$. Assuming bounded expert-learner marginal state-action density ratio $\left\|\frac{d^{\hat{\pi}}_P(s,a)}{d^\pi_P(s,a)}\right\|_\infty \leq C$, we have the following (absolute) performance bound for the IRL agent:*

$$|J_P(\hat{\pi}) - J_P(\pi)| \leq \frac{1}{1-\gamma}\epsilon_{\hat{\pi}} + \frac{\gamma(C+1)R_{max}}{(1-\gamma)^2}\sqrt{2\epsilon_{\hat{P}}} \tag{32}$$

Proof.

$$\begin{aligned}
&|J_P(\hat{\pi}) - J_P(\pi)| \\
&\leq \frac{1}{1-\gamma}\epsilon_{\hat{\pi}} \\
&\quad + \frac{\gamma}{1-\gamma}\mathbb{E}_{(s,a)\sim d^\pi_P}\left[\left|\frac{d^{\hat{\pi}}_P(s,a)}{d^\pi_P(s,a)}\left(\mathbb{E}_{s'\sim P}V^{\hat{\pi}}_{\hat{P}}(s') - \mathbb{E}_{s''\sim\hat{P}}V^{\hat{\pi}}_{\hat{P}}(s'')\right)\right|\right] \\
&\quad + \frac{\gamma}{1-\gamma}\mathbb{E}_{(s,a)\sim d^\pi_P}\left[\left|\mathbb{E}_{s'\sim\hat{P}}V^{\hat{\pi}}_{\hat{P}}(s') - \mathbb{E}_{s''\sim P}V^{\hat{\pi}}_{\hat{P}}(s'')\right|\right] \\
&\leq \frac{1}{1-\gamma}\epsilon_{\hat{\pi}} \\
&\quad + \frac{\gamma}{1-\gamma}\left\|\frac{d^{\hat{\pi}}_P(\cdot,\cdot)}{d^\pi_P(\cdot,\cdot)}\right\|_\infty\left\|V^{\hat{\pi}}_{\hat{P}}(\cdot)\right\|_\infty\mathbb{E}_{(s,a)\sim d^\pi_P}\left[\left\|\hat{P}(\cdot|s,a) - P(\cdot|s,a)\right\|_1\right] \\
&\quad + \frac{\gamma}{1-\gamma}\left\|V^{\hat{\pi}}_{\hat{P}}(\cdot)\right\|_\infty\mathbb{E}_{(s,a)\sim d^{\hat{\pi}}_P}\left[\left\|\hat{P}(\cdot|s,a) - P(\cdot|s,a)\right\|_1\right] \\
&= \frac{1}{1-\gamma}\epsilon_{\hat{\pi}} + \frac{\gamma(C+1)R_{\max}}{(1-\gamma)^2}\sqrt{2\epsilon_{\hat{P}}}
\end{aligned} \tag{33}$$

where the last line uses results from lemma B.2. □

C Further Algorithm Details and Pseudo Code

We estimate the dynamics gradient in (16) and (18) using the REINFORCE method with baseline:

$$\begin{aligned}\nabla_{\theta_2} E V_\theta(s,a) &= \sum_{s'} V_\theta(s') \nabla_{\theta_2} \hat{P}_{\theta_2}(s'|s,a) \\ &= \mathbb{E}_{s' \sim \hat{P}(\cdot|s,a)} \left[\left(V_\theta(s') - b(s,a) \right) \nabla_{\theta_2} \log \hat{P}_{\theta_2}(s'|s,a) \right]\end{aligned}$$

Following Rigter et al. [45], we set the baseline to $b(s,a) = Q_\theta(s,a) - R_{\theta_1}(s,a)$ to reduce gradient variance and further normalize $V_\theta(s') - b(s,a)$ across the mini-batch to stabilize training. In the continuous-control setting, the value function can be estimated as $V_\theta(s) = \mathbb{E}_{a \sim \hat{\pi}_\theta}[Q_\theta(s,a) - \log \hat{\pi}(a|s;\theta)]$ with a single sample. We apply this gradient for a fixed number of steps, which is a hyperparameter.

Pseudo code for the proposed algorithms are listed in Algorithm 1 and Algorithm 2.

Algorithm 1 Deep Bayesian Theory of Mind (BTOM)

Require: Dataset $\mathcal{D} = \{\tau\}$, dynamics model $\hat{P}_{\theta_2}(s'|s,a)$, reward model $R_{\theta_1}(s,a)$, hyperparameters λ_1, λ_2

1: **for** $k = 1 : K$ **do**
2: Run MBPO to update learner policy $\hat{\pi}(a|s;\theta)$ and value function $Q_\theta(s,a)$ in dynamics $\hat{P}$
3: Sample real trajectory τ_{real} starting from $(s,a) \sim \mathcal{D}$ and following $\hat{P}$ and $\hat{\pi}$
4: Sample fake trajectory τ_{fake} starting from $s \sim \mathcal{D}$, $a_{\text{fake}} \sim \hat{\pi}(\cdot|s;\theta)$ and following $\hat{P}$ and $\hat{\pi}$
5: Evaluate (15) and take a gradient step
6: Evaluate (16) and take a few gradient steps.
7: **end for**

Algorithm 2 Robust Theory of Mind (RTOM)

Require: Dataset $\mathcal{D} = \{\tau\}$, dynamics model $\hat{P}_{\theta_2}(s'|s,a)$, reward model $R_{\theta_1}(s,a)$, hyperparameters λ_1, λ_2

1: **for** $k = 1 : K$ **do**
2: Run MBPO to update learner policy $\hat{\pi}(a|s;\theta)$ and value function $Q_\theta(s,a)$ in dynamics $\hat{P}$
3: Sample fake trajectory τ_{fake} starting from $s \sim \mathcal{D}$ and following $\hat{P}$ and $\hat{\pi}$
4: Evaluate (17) and take a gradient step
5: Evaluate (18) and take a few gradient steps
6: **end for**

D Implementation Details

Our implementation builds on top of the official RAMBO implementation[1] [45].

D.1 MuJoCo Benchmarks

For the MuJoCo benchmarks described in section 4.2, we follow standard practices in model-based RL.

D.1.1 Dynamics Pre-training

We use an ensemble of $K = 7$ neural networks where each network outputs the mean and covariance parameters of a Gaussian distribution over the difference between the next state and the current state $\delta = s' - s$:

$$\hat{P}^{(k)}_{\theta_2}(\delta|s,a) = \mathcal{N}(\delta|\mu^{(k)}_{\theta_2}(s,a), \Sigma^{(k)}_{\theta_2}(s,a)) \tag{34}$$

[1] https://github.com/marc-rigter/rambo

Each network is a 4-layer feedforward network with 200 hidden units and Sigmoid linear unit (SiLU) activation function. For the initial pre-training step, we maximize the likelihood of dataset transitions using a batch size of 256 and early stop when all models stop improving for more than 1 percent. We then select the 5 best models in terms of mean-squared-error on a 10 % holdout validation set. During model rollouts, we randomly pick one of the 5 best models (elites) to sample the next state.

Table 2: Shared hyperparameters across different environments

	Hyparameter	BTOM	RTOM
SAC + MBPO	critic learning rate	3e-4	3e-4
	actor learning rate	3e-4	3e-4
	discount factor (γ)	0.99	0.99
	soft target update parameter (τ)	5e-3	5e-3
	target entropy	-dim(A)	-dim(A)
	minimum temperature (α)	0.1	0.001
	batch size	256	256
	real ratio	0.5	0.5
	model retain epochs	5	5
	training epochs	500	300
	steps per epoch	1000	1000
Dynamics	# model networks	7	7
	# elites	5	5
	adv. rollout batch size	1000	256
	adv. rollout steps	10	10
	adv. update steps	50	50
	adv. loss weighting (λ_1)	0.01	0.01
	supervised. loss weighting (λ_2)	1	1
	learning rate	1e-4	1e-4
	adv. update steps	50	50
Reward	max reward	10	10
	rollout batch size	1000	64
	rollout steps	40	100
	l2 penalty	1e-3	1e-3
	learning rate	1e-4	1e-4
	update steps	1	1

Table 3: Environment-specific hyperparameters

Environment	Hyperparameter	BTOM
HalfCheetah	model rollout batch size	50000
	model rollout steps	5
	model rollout frequency	250
Hopper	model rollout batch size	10000
	model rollout steps	40
	model rollout frequency	250
Walker2d	model rollout batch size	10000
	model rollout steps	40
	model rollout frequency	250

D.1.2 Policy Training

Our policy training process follows MBPO [37] which uses SAC with automatic temperature tuning [64]. Shared hyperparameters across different environments are listed in Table 2 and environment-specific hyperparameters are listed in Table 3. For the actor and critic, we use feedforward neural networks with 2 hidden layers of 256 units and ReLU activation. We train the actor and critic networks using a combination of real and simulated samples. We use a real ratio of 0.5, which is standard practice in model-based RL and IRL. We found that BTOM requires a higher minimum temperature to stablize training, which is set to $\alpha = 0.1$.

We found that different MuJoCo environments require different model rollout hyperparameters, similar to what's reported in [41]. Specifically, Hopper and Walker2d only work with significantly larger rollout steps. We decrease their rollout batch size to reduce computational overhead. HalfCheetah on the other hand works better with smaller rollout steps and larger rollout batch size. In contrast to Lu et al. [41], we did not use different rollout hyperparameters for different datasets.

D.1.3 Reward and Dynamics Training

We use 10 random trajectories from the D4RL MuJoCo expert dataset after removing all expert trajectories that resulted in terminal states.

We use the same network architecture as the actor-critic to parameterize the reward function. We further clip the reward function to a maximum range of ± 10 and apply l2 regularization on all weights with a penalty of 0.001.

As described in the main text, we update the reward function by simulating sample trajectories and taking a single gradient step. For RTOM, we randomly sample expert trajectory segments of length "rollout steps" and use the first step as the start of our simulated sample paths.

We update the dynamics using on-policy rollouts branched from the dataset state-actions. We use the same batch size for reward and dynamics rollouts, which is 1000 for BTOM and 256 for RTOM. Because only the first step in BTOM's real sample paths come from the dataset, it requires a larger batch size to iterate more data samples. We also train BTOM for more epochs than RTOM.

To compute the dynamics log likelihood in the REINFORCE gradient in (34), we treat the ensemble as a uniform mixture and compute the likelihood as:

$$\hat{P}_{\theta_2}(\delta|s,a) = \frac{1}{K}\sum_{k=1}^{K} \hat{P}_{\theta_2}^{(k)}(\delta|s,a) \tag{35}$$

We set the dynamics adversarial loss weighting to $\lambda_1 = 0.01$ for both BTOM and RTOM. We found this to work better than what's in the official RAMBO implementation, which is $\lambda_1 = 0.0768$. Note that the RAMBO author reported $\lambda_1 =$ 3e-4 in their paper but forget to average their REINFORCE loss over the mini-batch of size 256 in their implementation, which is instead treated as a sum by default by TensorFlow. We empirically found that small λ_1 leads to severe model exploitation.