Variational Model-based Policy Optimization

Yinlam Chow[1] , **Brandon Cui**[2] , **Moonkyung Cui**[1] and **Mohammad Ghavamzadeh**[1]
[1]Google AI
[2]Facebook AI Research
{yinlamchow, mkryu, ghavamza}@google.com, bcui@fb.com

Abstract

Model-based reinforcement learning (RL) algorithms allow us to combine model-generated data with those collected from interaction with the real system in order to alleviate the data efficiency problem in RL. However, designing such algorithms is often challenging because the bias in simulated data may overshadow the ease of data generation. A potential solution to this challenge is to jointly learn and improve model and policy using a universal objective function. In this paper, we leverage the connection between RL and probabilistic inference, and formulate such an objective function as a variational lower-bound of a log-likelihood. This allows us to use expectation maximization (EM) and iteratively fix a baseline policy and learn a variational distribution, consisting of a model and a policy (E-step), followed by improving the baseline policy given the learned variational distribution (M-step). We propose model-based and model-free policy iteration (actor-critic) style algorithms for the E-step and show how the variational distribution learned by them can be used to optimize the M-step in a fully model-based fashion. Our experiments on a number of continuous control tasks show that our model-based (E-step) algorithm, which we refer to as *variational model-based policy optimization* (VMBPO), is more sample-efficient and robust to hyper-parameter tuning than its model-free (E-step) counterpart. Using the same control tasks, we also compare VMBPO with several state-of-the-art model-based and model-free RL algorithms and show its sample efficiency and performance.

1 Introduction

Model-free reinforcement learning (RL) algorithms that learn a good policy without constructing an explicit model of the system's dynamics have shown promising results in complex simulated problems [Mnih *et al.*, 2013, 2015; Schulman *et al.*, 2015; Haarnoja *et al.*, 2018]. However, these methods are not sample efficient, and thus, not suitable for problems in which data collection is burdensome. Model-based RL algorithms address the data efficiency issue of the model-free methods by learning a model, and combining model-generated data with those collected from interaction with the real system [Sutton, 1990; Janner *et al.*, 2019]. However, designing model-based RL algorithms is often challenging because the bias in model may affect the process of learning policies and result in worse asymptotic performance than the model-free counterparts. A potential solution to this challenge is to incorporate the policy/value optimization method in the process of learning the model. An ideal case here would be to have a universal objective function that is used to learn and improve model and policy jointly.

Casting RL as a probabilistic inference has a long history (e.g., Todorov 2008; Toussaint 2009; Kappen *et al.* 2012; Rawlik *et al.* 2013). This formulation has the advantage that allows powerful tools for approximate inference to be employed in RL. One such class of tools are variational techniques that have been successfully used in RL (e.g., Neumann 2011; Levine and Koltun 2013; Abdolmaleki *et al.* 2018). Another formulation of RL with strong connection to probabilistic inference is the formulation of policy search as an expectation maximization (EM) style algorithm (e.g., Dayan and Hinton 1997; Peters and Schaal 2007; Peters *et al.* 2010; Chebotar *et al.* 2017; Abdolmaleki *et al.* 2018). The main idea here is to write the expected return of a policy as a (pseudo)-likelihood function, and then conditioning on the success in maximizing the return, find the policy that most likely would have been taken. Another class of RL algorithms that are related to the inference formulation are entropy-regularized algorithms that add an entropy term to the reward and find the soft-max optimal policy (e.g., Levine and Abbeel 2014; Nachum *et al.* 2017; Haarnoja *et al.* 2018; Fellows *et al.* 2019). For a comprehensive tutorial on RL as probabilistic inference, we refer readers to Levine [2018].

In this paper, we leverage the connection between RL and probabilistic inference, and formulate an objective function for jointly learning and improving model and polciy as a variational lower-bound of a log-likelihood. This allows us to use EM: iteratively fix a baseline policy and learn a variational distribution, consisting of a model and a policy (E-step), followed by improving the baseline policy given the learned variational distribution (M-step). We propose model-based and model-free policy iteration (PI) style algorithms for the E-step and show how the variational distribution that they learn can be used to optimize the M-step only from model-generated samples. It is important to note that both algorithms are model-based and only differ in using model-based and model-free algorithms for the E-step. We call our algorithm that uses model-based PI for the E-step, *variational model-based policy optimization* (VMBPO). Our experiments on a number of continuous control tasks show that VMBPO is more sample-efficient and robust to hyper-parameter tuning than its model-free counterpart. Using the same control tasks, we also compare VMBPO with several state-of-the-art model-based and model-free RL algorithms, including model-based policy optimization (MBPO) [Janner *et al.*, 2019] and maximum a posteriori policy optimization (MPO) [Abdolmaleki *et al.*, 2018], and show its sample efficiency and performance.

2 Preliminaries

We study the RL problem in which the agent's interaction with the environment is modeled as a Markov decision process (MDP) $\mathcal{M} = \langle \mathcal{X}, \mathcal{A}, r, p, p_0 \rangle$, where $\mathcal{X}$ and $\mathcal{A}$ are state and action spaces; $r : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ is the reward function; $p : \mathcal{X} \times \mathcal{A} \to \Delta_{\mathcal{X}}$ is the transition kernel ($\Delta_{\mathcal{X}}$ is the set of probability distributions over $\mathcal{X}$); and $p_0 : \mathcal{X} \to \Delta_{\mathcal{X}}$ is the initial state distribution. A stationary Markovian policy $\pi : \mathcal{X} \to \Delta_{\mathcal{A}}$ is a probabilistic mapping from states to actions. Each policy π is evaluated by its *expected return*, i.e., $J(\pi) = \mathbb{E}[\sum_{t=0}^{T-1} r(x_t, a_t) \mid p_0, p, \pi]$, where T is the (random) time of hitting a *terminal state*.[1] We denote by $\mathcal{X}^0$ the set of all terminal states. The agent's goal is to find a policy with maximum expected return, i.e, $\pi^* \in \arg\max_{\pi \in \Delta_{\mathcal{A}}} J(\pi)$. We denote by $\xi = (x_0, a_0, \ldots, x_{T-1}, a_{T-1}, x_T)$, a system trajectory of length T, whose probability under a policy π is defined as $p_\pi(\xi) = p_0(x_0) \prod_{t=0}^{T-1} \pi(a_t|x_t) p(x_{t+1}|x_t, a_t)$. Finally, we define $[T] := \{0, \ldots, T-1\}$.

3 Policy Optimization as Inference

Policy search in RL can be formulated as a probabilistic inference problem (e.g., Todorov 2008; Toussaint 2009; Kappen *et al.* 2012; Levine 2018). The goal in the conventional RL formulation is to find a policy whose generated trajectories maximize the expected return. In contrast, in the inference formulation, we start with a prior over trajectories and then estimate the posterior conditioned on a desired outcome, such as reaching a goal state. In this formulation, the notion of a desired (optimal) outcome is introduced via *independent* binary random variables $\mathcal{O}_t,\ t \in [T]$, where $\mathcal{O}_t = 1$ denotes that we acted optimally at time t. The likelihood of $\mathcal{O}_t$, given the state x_t and action a_t, is modeled as

$$p(\mathcal{O}_t = 1 \mid x_t, a_t) = \exp(\eta \cdot r(x_t, a_t)),$$

where $\eta > 0$ is a temperature parameter. This allows us to define the log-likelihood of π being optimal as

$$\begin{aligned} \log p_\pi(\mathcal{O}_{0:T-1} = 1) &= \log \int_\xi p_\pi(\mathcal{O}_{0:T-1} = 1, \xi) \\ &= \log \mathbb{E}_{\xi \sim p_\pi}\big[p(\mathcal{O}_{0:T-1} = 1 \mid \xi)\big], \end{aligned} \tag{1}$$

where $p(\mathcal{O}_{0:T-1} = 1 \mid \xi)$ is the likelihood of trajectory ξ being optimal and is defined as

$$\begin{aligned} p(\mathcal{O}_{0:T-1} = 1 \mid \xi) &= \prod_{t=0}^{T-1} p(\mathcal{O}_t = 1 \mid x_t, a_t) \\ &= \exp\big(\eta \cdot \sum_{t=0}^{T-1} r(x_t, a_t)\big). \end{aligned} \tag{2}$$

As a result, finding an optimal policy in this setting would be equivalent to maximizing the log-likelihood in (1), i.e., $\pi^*_{\text{soft}} \in \arg\max_\pi \log p_\pi(\mathcal{O}_{0:T-1} = 1)$. A potential advantage of formulating RL as an inference problem is the possibility of using a wide range of approximate inference algorithms, including variational methods. In variational inference, we approximate a distribution $p(\cdot)$ with a potentially simpler (e.g., tractable factored) distribution $q(\cdot)$ in order to make the whole inference process more tractable. If we approximate $p_\pi(\xi)$ with a variational distribution $q(\xi)$, we will obtain the following variational lower-bound for the log-likelihood in (1):

$$\begin{aligned} \log p_\pi(\mathcal{O}_{0:T-1} = 1) &= \log \mathbb{E}_{\xi \sim p_\pi}\big[\exp\big(\eta \cdot \sum_{t=0}^{T-1} r(x_t, a_t)\big)\big] \\ &= \log \mathbb{E}_{\xi \sim q(\xi)}\big[\frac{p_\pi(\xi)}{q(\xi)} \cdot \exp\big(\eta \cdot \sum_{t=0}^{T-1} r(x_t, a_t)\big)\big] \\ &\overset{\text{(a)}}{\geq} \mathbb{E}_{\xi \sim q(\xi)}\big[\log \frac{p_\pi(\xi)}{q(\xi)} + \eta \cdot \sum_{t=0}^{T-1} r(x_t, a_t)\big] \\ &= \eta \cdot \mathbb{E}_q\big[\sum_{t=0}^{T-1} r(x_t, a_t)\big] - \text{KL}(q||p_\pi) := \mathcal{J}(q; \pi), \end{aligned} \tag{3}$$

(a) is from Jensen's inequality and $\mathcal{J}(q;\pi)$ is the evidence lower-bound (ELBO) of the log-likelihood function. A variety of algorithms have been proposed (e.g., Peters and Schaal 2007; Hachiya *et al.* 2009; Neumann 2011; Levine and Koltun 2013; Abdolmaleki *et al.* 2018; Fellows *et al.* 2019), whose main idea is to approximate π^*_{soft} by maximizing $\mathcal{J}(q;\pi)$ w.r.t. both q and π. This often results in an EM-style algorithm in which we first fix π and maximize $\mathcal{J}(\cdot;\pi)$ for q (E-step), and then given the q obtained in the E-step, we maximize $\mathcal{J}(q;\cdot)$ for π (M-step).

4 Variational Model-based Policy Optimization

In this section, we describe the ELBO objective function used by our algorithms, study the properties of the resulted optimization problem, and propose algorithms to solve it. We propose to use the variational distribution $q(\xi) = p_0(x_0) \prod_{t=0}^{T-1} q_c(a_t|x_t) q_d(x_{t+1}|x_t, a_t)$ to approximate $p_\pi(\xi)$. Note that q has the same initial state distribution as p_π (defined in Section 2), but has different control strategy (policy), q_c, and dynamics, q_d. Using this variational distribution, we may write the ELBO objective (3) as

$$\begin{aligned} \mathcal{J}(q;\pi) = \mathbb{E}_q\big[\sum_{t=0}^{T-1} \eta \cdot r(x_t, a_t) - \log \frac{q_c(a_t|x_t)}{\pi(a_t|x_t)} \\ - \log \frac{q_d(x_{t+1}|x_t, a_t)}{p(x_{t+1}|x_t, a_t)}\big], \quad \text{where } \mathbb{E}_q[\cdot] := \mathbb{E}[\cdot \mid p_0, q_d, q_c]. \end{aligned} \tag{4}$$

To maximize $\mathcal{J}(q;\pi)$ w.r.t. q and π, we first fix π and compute the variational distribution **(E-step)**:

[1] Similar to Levine [2018], our setting can be easily extended to infinite-horizon γ-discounted MDPs. This can be done by modifying the transition kernels, such that any action transitions the system to a terminal state with probability $1 - \gamma$, and all standard transition probabilities are multiplied by γ.

$$
\begin{aligned}
q^* = (q_c^*, q_d^*) \in \operatorname*{arg\,max}_{q_c \in \Delta_{\mathcal{A}}, q_d \in \Delta_{\mathcal{X}}} \;& \mathbb{E}\big[\sum_{t=0}^{T-1} \eta \cdot r(x_t, a_t) \\
& - \log \frac{q_c(a_t|x_t)}{\pi(a_t|x_t)} - \log \frac{q_d(x_{t+1}|x_t, a_t)}{p(x_{t+1}|x_t, a_t)} \mid p_0, q_d, q_c\big],
\end{aligned} \tag{5}
$$

and then optimize π given q^*, i.e., $\arg\max_\pi \mathcal{J}(q^*; \pi)$ **(M-step)**. Note that in (5), q_c^* and q_d^* are both functions of π, but we remove π from the notation to keep it lighter.

Remark 1. *In our formulation (our choice of the variational distribution q), the M-step is independent of the true dynamics, p, and thus, can be implemented offline (using samples generated by the model q_d). Moreover, as we will see in Section 5, we also use the model, q_d, in the E-step. As discussed throughout the paper, using simulated samples (from q_d) and reducing the need for real samples (from p) is an important feature of our proposed model-based formulation and algorithms.*

Remark 2 (relationship with MPO)**.** *There are similarities between our variational formulation and the one used in the maximum a posteriori policy optimization (MPO) algorithm [Abdolmaleki* et al.*, 2018]. However, MPO sets its variational dynamics, q_d, to be the dynamics of the real system, p, which results in a model-free algorithm, while our approach is model-based, since we learn q_d and use it to generate samples in both E-step and M-step of our algorithms.*

In the rest of this section, we study the E-step optimization (5) and propose algorithms to solve it.

4.1 Properties of the E-step Optimization

We start by defining two Bellman-like operators related to the E-step optimization (5). For any variational policy $q_c : \mathcal{X} \to \Delta_{\mathcal{A}}$ and any value function $V : \mathcal{X} \to \mathbb{R}$, such that $V(x) = 0, \; \forall x \in \mathcal{X}^0$, we define the *$q_c$-induced operator*, $\mathcal{T}_{q_c}$, and the *optimal operator*, $\mathcal{T}$, as

$$
\begin{aligned}
\mathcal{T}_{q_c}[V](x) := \mathbb{E}_{a \sim q_c(\cdot|x)} \Big[& \eta \cdot r(x,a) - \log \frac{q_c(a|x)}{\pi(a|x)} \\
& + \max_{q_d \in \Delta_{\mathcal{X}}} \mathbb{E}_{x' \sim q_d(\cdot|x,a)} \big[V(x') - \log \frac{q_d(x'|x,a)}{p(x'|x,a)} \big] \Big],
\end{aligned} \tag{6}
$$

$$
\mathcal{T}[V](x) := \max_{q_c \in \Delta_{\mathcal{A}}} \mathcal{T}_{q_c}[V](x). \tag{7}
$$

We define the *optimal value function* of the E-step, V_π, as

$$
\begin{aligned}
V_\pi(x) := \mathbb{E}\big[\sum_{t=0}^{T-1} \eta \cdot r(x_t, a_t) &- \log \frac{q_c^*(a_t|x_t)}{\pi(a_t|x_t)} \\
&- \log \frac{q_d^*(x_{t+1}|x_t, a_t)}{p(x_{t+1}|x_t, a_t)} \mid p_0, q_d^*, q_c^* \big].
\end{aligned} \tag{8}
$$

For any value function V, we define its associated action-value function $Q : \mathcal{X} \times \mathcal{A} \to \mathbb{R}$ as

$$
Q(x,a) := \eta \cdot r(x,a) + \log \mathbb{E}_{x' \sim p(\cdot|x,a)} \big[\exp\big(V(x')\big) \big]. \tag{9}
$$

The following lemmas, whose proofs are reported in Appendices A.1 and A.2, show properties of operators $\mathcal{T}_{q_c}$ and $\mathcal{T}$, and their relation with the (E-step) optimal value function, V_π.

Lemma 1. *The q_c-induced and optimal operators, defined by* (6) *and* (7)*, can be rewritten as*

$$
\mathcal{T}_{q_c}[V](x) = \mathbb{E}_{a \sim q_c(\cdot|x)} \big[Q(x,a) - \log \frac{q_c(a|x)}{\pi(a|x)} \big], \tag{10}
$$

$$
\mathcal{T}[V](x) = \log \mathbb{E}_{a \sim \pi(\cdot|x), x' \sim p(\cdot|x,a)} \big[\exp\big(\eta \cdot r(x,a) + V(x') \big) \big]. \tag{11}
$$

Lemma 2. *The q_c-induced and optimal operators are monotonic and contractive. Moreover, the optimal value function V_π is the unique fixed-point of $\mathcal{T}$ ($\mathcal{T}[V_\pi](x) = V_\pi(x), \; \forall x \in \mathcal{X}$).*

From the definition of Q-function in (9) and Lemma 2, we prove (in Appendix A.3) the following proposition for the action-value function associated with the E-step optimal value function V_π.

Proposition 1. *The E-step optimal value function V_π and its associated action-value function Q_π, defined by* (9)*, are related as $V_\pi(x) = \log \mathbb{E}_{a \sim \pi(\cdot|x)} \big[\exp\big(Q_\pi(x,a) \big) \big], \forall x \in \mathcal{X}$.*

In the rest of this section, we show how to derive a closed-form expression for the variational distribution $q^* = (q_c^*, q_d^*)$. For any value function V, we define its corresponding variational dynamics, q_d^V, as the solution to the maximization problem in the definition of $\mathcal{T}_{q_c}$ (see Eq. 6), i.e.,

$$
q_d^V(\cdot|x,a) \in \operatorname*{arg\,max}_{q_d \in \Delta_{\mathcal{X}}} \mathbb{E}_{x' \sim q_d} \big[V(x') - \log \frac{q_d(x'|x,a)}{p(x'|x,a)} \big], \tag{12}
$$

and its corresponding variational policy q_c^Q (Q is the action-value function associated with V, defined by Eq. 9), as the solution to the maximization problem in the definition of $\mathcal{T}$ (see Eqs. 7 and 10), i.e.,

$$
q_c^Q(\cdot|x) \in \operatorname*{arg\,max}_{q_c \in \Delta_{\mathcal{A}}} \; \mathbb{E}_{a \sim q_c(\cdot|x)} \big[Q(x,a) - \log \frac{q_c(a|x)}{\pi(a|x)} \big]. \tag{13}
$$

We now derive (proof in Appendix A.4) closed-form expressions for the variational distributions q_d^V and q_c^Q.

Lemma 3. *The variational dynamics and policy corresponding to a value function V and its associated action-value function Q can be written in closed-form as*

$$
\begin{aligned}
q_d^V(x'|x,a) &= \frac{p(x'|x,a) \cdot \exp\big(V(x')\big)}{\mathbb{E}_{x' \sim p(\cdot|x,a)} \big[\exp\big(V(x')\big) \big]} \\
&= \frac{p(x'|x,a) \cdot \exp\big(V(x')\big)}{\exp\big(Q(x,a) - \eta \cdot r(x,a)\big)}, \quad \forall x, x' \in \mathcal{X}, \; \forall a \in \mathcal{A},
\end{aligned} \tag{14}
$$

$$
q_c^Q(a|x) = \frac{\pi(a|x) \cdot \exp\big(Q(x,a)\big)}{\mathbb{E}_{a \sim \pi(\cdot|x)} \big[\exp\big(Q(x,a)\big) \big]}, \quad \forall x \in \mathcal{X}, \; \forall a \in \mathcal{A}. \tag{15}
$$

Equations 14 and 15 show that the variational dynamics, q_d^V, and policy, q_c^Q, can be seen as an *exponential twisting* of the dynamics p and policy π with weights V and Q, respectively. For the special case $V = V_\pi$ (the E-step optimal value function), these distributions can be written in closed-form as

$$
\begin{aligned}
q_d^*(x'|x,a) &= \frac{p(x'|x,a) \cdot \exp\big(V_\pi(x')\big)}{\exp\big(Q_\pi(x,a) - \eta \cdot r(x,a)\big)}, \\
q_c^*(a|x) &= \frac{\pi(a|x) \cdot \exp\big(Q_\pi(x,a)\big)}{\exp\big(V_\pi(x)\big)},
\end{aligned} \tag{16}
$$

where the denominator of q_c^* is obtained by applying Proposition 1 to replace Q_π with V_π.

4.2 Policy and Value Iteration for the E-step

Using the results of Section 4.1, we now propose *model-based* and *model-free* dynamic programming (DP) style algorithms, i.e., policy iteration (PI) and value iteration (VI), for solving the E-step problem (5). The model-based algorithms compute the variational dynamics, q_d, at each iteration, while the model-free counterparts compute q_d only at the end (upon convergence). Having access to q_d at each iteration has the advantage that we may generate samples from the model, q_d, when we implement the sample-based version (RL version) of these DP algorithms in Section 5.

In the **model-based PI** algorithm, at each iteration k, given the current variational policy $q_c^{(k)}$, we

Policy Evaluation: Compute the $q_c^{(k)}$-induced value function $V_{q_c^{(k)}}$ (the fixed-point of the operator $\mathcal{T}_{q_c^{(k)}}$) by iteratively applying $\mathcal{T}_{q_c^{(k)}}$ from (6), i.e., $V_{q_c^{(k)}}(x) = \lim_{n\to\infty} \mathcal{T}^n_{q_c^{(k)}}[V](x), \ \forall x \in \mathcal{X}$, where the variational model q_d in (6) is computed using (14) with $V = V^{(n)}$. We then compute the corresponding Q-function $Q_{q_c^{(k)}}$ using (9).

Policy Improvement: Update the variational distribution $q_c^{(k+1)}$ using (15) with $Q = Q_{q_c^{(k)}}$.[2]

Upon convergence, i.e., $q_c^{(\infty)} = q_c^*$, we compute q_d^* from (14).

The **model-free PI** algorithm is exactly the same, except in its policy evaluation step, the $q_c^{(k)}$-induced operator $\mathcal{T}_{q_c^{(k)}}$ is applied using (10) (without the variational dynamics q_d). In this case, the variational dynamics q_d is computed only upon convergence, q_d^*, using (14).

Lemma 4. *The model-based and model-free PI algorithms converge to their optimal values, q_c^* and q_d^*, defined by (5), i.e., $q_c^{(\infty)} = q_c^*$ and $q_d^{(\infty)} = q_d^*$.* *(proof in Appendix A.5)*

We can similarly derive **model-based** and **model-free (VI)** algorithms for the E-step. These algorithms start from an arbitrary value function V and iteratively apply the optimal operator $\mathcal{T}$ from (6) and (7) (model-based) and (11) (model-free) until convergence, i.e., $V_\pi(x) = \lim_{n\to\infty} \mathcal{T}^n[V](x), \ \forall x \in \mathcal{X}$. Given V_π, these algorithms first compute Q_π from Proposition 1, and then compute (q_c^*, q_d^*) using (16). From the properties of the optimal operator $\mathcal{T}$ in Lemma 2, both model-based and model-free VI algorithms converge to q_c^* and q_d^*.

In the rest of the paper, we focus on the PI approach, in particular the model-based one, and only report the details of the VI algorithms in Appendix B. In the next section, we show how the PI algorithms can be implemented and combined with a routine for solving the M-step, when the true MDP model p is unknown (the RL setting) and the state and action spaces are large that require using function approximation.

5 Variational Model-based RL Algorithm

In this section, we propose a RL algorithm, called *variational model-based policy optimization* (VMBPO). It is an EM-style algorithm based on the variational formulation proposed in Section 4. The E-step of VMBPO is the sample-based implementation of the model-based PI algorithm, described in Section 4.2. We describe the E-step and M-step of VMBPO in details in Sections 5.1 and 5.2, and report its pseudo-code in Algorithm 1 in Appendix C. VMBPO uses 8 neural networks to represent: policy π, variational dynamics q_d, variational policy q_c, log-likelihood ratio $\nu = \log(q_d/p)$, value function V, action-value function Q, target value function V', and target action-value function Q', with parameters θ_π, θ_d, θ_c, θ_ν, θ_v, θ_q, θ'_v, and θ'_q, respectively.

5.1 The E-step of VMBPO

At the beginning of the E-step, we generate a number of samples (x, a, r, x') from the current baseline policy π, i.e., $a \sim \pi(\cdot|x)$ and $r = r(x,a)$, and add them to the buffer $\mathcal{D}$. The E-step consists of four updates: **1)** computing the variational dynamics q_d, **2)** estimating the log-likelihood ratio $\log(q_d/p)$, **3)** computing the q_c-induced value and action-value functions V_{q_c} and Q_{q_c} (critic update), and finally **4)** computing the new variational policy q_c (actor update). We describe the details of each step below.

Step 1. (Computing q_d) We find q_d as a solution to the optimization problem (12) for V equal to the target value network V'. Since the q_d^V in (14) is the solution of (12), we compute q_d by minimizing $\mathrm{KL}(q_d^{V'}||q_d)$, which results in the following *forward* KL loss (for all $x \in \mathcal{X}$ and $a \in \mathcal{A}$):

$$\theta_d = \arg\min_\theta \ \mathrm{KL}\big(p(\cdot|x,a) \cdot \exp(\eta \cdot r(x,a) + V'(\cdot;\theta'_v) \\ - Q'(x,a;\theta'_q)) \;||\; q_d(\cdot|x,a;\theta)\big) \qquad (17)$$

$$\overset{\text{(a)}}{=} \arg\max_\theta \ \mathbb{E}_{x'\sim p(\cdot|x,a)}\big[\exp(\eta \cdot r(x,a) + V'(x';\theta'_v) \\ - Q'(x,a;\theta'_q)) \cdot \log(q_d(\cdot|x,a;\theta))\big], \qquad (18)$$

where **(a)** is by removing the θ-independent terms from (17). We update θ_d by taking several steps in the direction of the gradient of a sample average of the loss function (18), i.e.,

$$\theta_d = \arg\max_\theta \sum_{(x,a,r,x')\sim\mathcal{D}} \exp\big(\eta \cdot r + V'(x';\theta'_v) \\ - Q'(x,a;\theta'_q)\big) \cdot \log\big(q_d(x'|x,a;\theta)\big), \qquad (19)$$

where (x, a, r, x') are randomly sampled from $\mathcal{D}$. The intuition here is to focus on learning the dynamics model in the regions of the state-action space that have higher temporal difference (regions with higher anticipated future return).

Step 2. (Computing $\log(q_d/p)$) Using the duality of f-divergence [Nguyen *et al.*, 2008] w.r.t. the *reverse* KL-divergence, the log-likelihood ratio $\log(q_d(\cdot|x,a;\theta_d)/p(\cdot|x,a))$ is a solution to

$$\log\big(\frac{q_d(x'|x,a;\theta_d)}{p(x'|x,a)}\big) = \arg\max_{\nu:\mathcal{X}\times\mathcal{A}\times\mathcal{X}\to\mathbb{R}} \mathbb{E}_{x'\sim q_d(\cdot|x,a;\theta_d)}[\nu(x'|x,a)] \\ - \mathbb{E}_{x'\sim p(\cdot|x,a)}\big[\exp\big(\nu(x'|x,a)\big)\big], \qquad (20)$$

for all $x, x' \in \mathcal{X}$ and $a \in \mathcal{A}$. Note that the optimizer of (20) is unique almost surely (at (x,a,x') with $\mathbb{P}(x'|x,a) > 0$), because q_d is absolutely continuous w.r.t. p (see the definition of q_d in Eq. 14) and the objective function of (20) is strictly concave. The optimization problem (20) allows us to compute $\nu(\cdot|\cdot;\theta_\nu)$ as an approximation to the log-likelihood ratio

[2] When the number of actions is large, the denominator of (15) cannot be computed efficiently. In this case, we replace (15) in the policy improvement step of our PI algorithms with $q_c^{(k+1)} = \arg\min_{q_c} \mathrm{KL}(q_c||q_c^Q)$, where $Q = Q_{q_c^{(k)}}$. We also prove the convergence of our PI algorithms with this update in Appendix A.5.

$\log(q_d(\cdot;\theta_d)/p)$. We update θ_ν by taking several steps in the direction of the gradient of a sample average of (20), i.e.,

$$\theta_\nu = \arg\max_\theta \sum_{(x,a,x')\sim\mathcal{E}} \nu(x'|x,a;\theta) - \sum_{(x,a,x')\sim\mathcal{D}} \exp(\nu(x'|x,a;\theta)), \quad (21)$$

where $\mathcal{E}$ is the set of samples for which x' is drawn from the variational dynamics, i.e., $x' \sim q_d(\cdot|x,a)$. Here we first sample (x,a,x') randomly from $\mathcal{D}$ and use them in the second sum. Then, for all (x,a) that have been sampled, we generate x' from q_d and use the resulting samples in the first sum.

Step 3. (critic update) To compute V_{q_c} and its action-value Q_{q_c}, we rewrite (6) with the maximizer q_d from Step 1 and the log-likelihood ratio $\log(q_d/p)$ from Step 2:

$$\mathcal{T}_{q_c}[V](x) = \mathbb{E}_{a\sim q_c(\cdot|x)}\Big[\eta\cdot r(x,a) - \log\frac{q_c(a|x)}{\pi(a|x)} \\ + \mathbb{E}_{x'\sim q_d(\cdot|x,a;\theta_d)}\big[V'(x';\theta'_v) - \nu(x'|x,a;\theta_\nu)\big]\Big].$$

Since $\mathcal{T}_{q_c}$ can be written as both (10) and (22), we compute the q_c-induced Q-function by setting the RHS of these equations equal to each other, i.e., for all $x\in\mathcal{X}$ and $a\sim q_c(\cdot|x;\theta_c)$,

$$Q(x,a;\theta_q) = \eta\cdot r(x,a) + \\ \mathbb{E}_{x'\sim q_d(\cdot|x,a;\theta_d)}\big[V'(x';\theta'_v) - \nu(x'|x,a;\theta_\nu)\big]. \quad (22)$$

Since the expectation in (22) is w.r.t. the variational dynamics (model) q_d, we can estimate Q_{q_c} only with samples generated from the model. We do this by taking several steps in the direction of the gradient of a sample average of the square-loss obtained by setting both sides of (22) equal, i.e.,

$$\theta_q = \arg\min_\theta \sum_{(x,a,r,x')\sim\mathcal{E}} \big(Q(x,a;\theta) - \eta\cdot r - V'(x';\theta'_v) + \nu(x'|x,a;\theta_\nu)\big)^2. \quad (23)$$

Note that in (22), the actions are generated by q_c. Thus, in (23), we first randomly sample x, then sample a from $q_c(\cdot|x;\theta_c)$, and finally draw x' from $q_d(\cdot|x,a;\theta_d)$. If the reward function is known (chosen by the designer of the system), then it is used to generate the reward signals $r = r(x,a)$ in (23), otherwise, a reward model has to be learned.

After estimating Q_{q_c}, we approximate V_{q_c}, the fixed-point of $\mathcal{T}_{q_c}$, using $\mathcal{T}_{q_c}$ definition in (10) as $\mathcal{T}_{q_c}[V](x) \approx V(x) \approx \mathbb{E}_{a\sim q_c(\cdot|x)}\big[Q(x,a;\theta_q) - \log\frac{q_c(a|x;\theta_c)}{\pi(a|x;\theta_\pi)}\big]$. This results in updating V_{q_c} by taking several steps in the direction of the gradient of a sample average of the square-loss obtained by setting the two sides of the above equation equal, i.e.,

$$\theta_v = \arg\min_\theta \sum_{(x,a)\sim\mathcal{E}} \big(V(x;\theta) - Q(x,a;\theta_q) + \log\frac{q_c(a|x;\theta_c)}{\pi(a|x;\theta_\pi)}\big)^2, \quad (24)$$

where x is randomly sampled and $a\sim q_c(\cdot|x;\theta_c)$ (without sampling from the true environment).

Step 4. (actor update) We update the variational policy q_c (policy improvement) by solving the optimization problem (13) for the Q estimated by the critic in Step 3. Since the q_c that optimizes (13) can be written as (15), we update it by minimizing $\mathrm{KL}(q_c||q_c^Q)$. This results in the following *reverse* KL loss (for all $x\in\mathcal{X}$):

$$\theta_c = \arg\min_\theta \mathrm{KL}\big(q_c(\cdot|x;\theta)||\frac{\pi(\cdot|x;\theta_\pi)\cdot\exp(Q(x,\cdot,;\theta_q))}{Z(x)}\big) \\ = \arg\min_\theta \mathbb{E}_{a\sim q_c}\big[\log(\frac{q_c(a|x;\theta)}{\pi(a|x;\theta_\pi)}) - Q(x,a,;\theta_q)\big].$$

If we reparameterize q_c using a transformation $a = f(x,\epsilon;\theta_c)$, where ϵ is a Gaussian noise, we can update θ_c by taking several steps in the direction of the gradient of a sample average of the above loss, i.e.,

$$\theta_c = \arg\min_\theta \sum_{(x,\epsilon)} \log\big(q_c(f(x,\epsilon;\theta)|x)\big) - Q(x,a,;\theta_q) \\ - \log\big(\pi(a|x;\theta_\pi)\big). \quad (25)$$

We can also compute q_c as the closed-form solution to (15), as described in Abdolmaleki *et al.* [2018]. They refer to this as non-parametric representation of the variational distribution.

5.2 The M-step of VMBPO

As described in Section 4, the goal of the M-step is to improve the baseline policy π, given the variational model $q^* = (q_c^*, q_d^*)$ learned in the E-step, by solving the following optimization problem:

$$\pi \leftarrow \arg\max_{\pi\in\Pi} \mathcal{J}(q^*;\pi) := \mathbb{E}_{q^*}\big[\sum_{t=0}^{T-1}\eta\cdot r(x_t,a_t) \\ - \log\frac{q_c^*(a_t|x_t)}{\pi(a_t|x_t)} - \log\frac{q_d^*(x_{t+1}|x_t,a_t)}{p(x_{t+1}|x_t,a_t)}\big]. \quad (26)$$

A nice feature of (26) is that it can be solved using only the variational model q^*, without the need for samples from the true environment p. However, it is easy to see that if the policy space considered in the M-step, Π, contains the policy space used for q_c in the E-step, then we can trivially solve the M-step by setting $\pi = q_c^*$. Although this is an option, it is more efficient in practice to solve a regularized version of (26). A practical way to regularize (26) is to make sure that the new baseline policy π remains close to the old one, which results in the following optimization problem:

$$\theta_\pi \leftarrow \arg\max_\theta \ \mathbb{E}_{q^*}\big[\sum_{t=0}^{T-1}\log(\pi(a_t|x_t;\theta)) \\ - \lambda\cdot\mathrm{KL}\big(\pi(\cdot|x_t;\theta_\pi)||\pi(\cdot|x_t;\theta)\big)\big]. \quad (27)$$

This is equivalent to the weighted MAP formulation used in the M-step of MPO [Abdolmaleki *et al.*, 2018]. In MPO, they define a prior over the parameter θ and add it as $\log P(\theta)$ to the objective function of (26). Then, they set the prior $P(\theta)$ to a specific Gaussian and obtain an optimization problem similar to (27) (see Section 3.3 in Abdolmaleki *et al.* 2018). However, in the absence of a variational dynamics model (i.e., $q_d = p$), they need real samples to solve their optimization problem, while our model-based approach uses simulated samples.

6 Experiments

To illustrate the effectiveness of VMBPO, we (i) compare it with several state-of-the-art RL methods, and (ii) evaluate sample efficiency of MBRL via ablation analysis.

Comparison with Baseline RL Algorithms. We compare VMBPO with five baselines, two popular model-free algorithms: MPO [Abdolmaleki *et al.*, 2018] and SAC [Haarnoja *et al.*, 2018], and three recent model-based algorithms: MBPO [Janner *et al.*, 2019], PETS [Chua *et al.*, 2019], and STEVE [Buckman *et al.*, 2018]. We also compare VMBPO with its variant that uses a model-free PI algorithm to solve the E-step (see Section 4.2). We refer to this variant as VMBPO-MFE and describe it in details in Appendix D. We evaluate all

Environment	VMBPO	MBPO	STEVE	PETS	VMBPO-MFE	SAC	MPO
Pendulum	-125.8 ± 73.7	-126.0 ± 78.4	-6385.3 ± 799.7	-183.5 ± 1773.9	-125.7 ± 130.1	**-124.7** ± 199.0	-131.9 ± 315.9
Hopper	**2897.4** ± 630.	2403.1 ± 556.	279.0 ± 237.1	94.5 ± 114.2	1368.7 ± 184.1	2020.8 ± 954.1	1509.7 ± 756.0
Walker2D	**4226.1** ± 843.0	3883.3 ± 753.5	336.3 ± 196.3	93.5 ± 134.1	3334.5 ± 122.8	3026.4 ± 888.9	2889.4 ± 712.7
HalfCheetah	13120 ± 933.1	11877 ± 997.1	482.9 ± 596.9	**13272.6** ± 4926.4	4647.3 ± 505.8	9080.3 ± 1625.1	4969.2 ± 623.7
Reacher	**-11.4** ± 27.0	-12.6 ± 25.9	-141.8 ± 355.7	—	-55.5 ± 39.0	-23.9 ± 23.8	-75.9 ± 336.7
Reacher7DoF	**-13.8** ± 20.5	-15.1 ± 98.8	—	-45.6 ± 36.1	-33.5 ± 49.6	-27.4 ± 112.0	-38.4 ± 53.8

Table 1: The mean ± standard deviation of the final returns with the best hyper-parameter configuration for each algorithm. VMBPO outperforms other baselines. VMBPO-MFE performs better than MPO but is sometimes unstable.

Environment	VMBPO	MBPO	VMBPO-MFE	SAC	MPO
Pendulum	-147.4 ± 94.1	**-146.8** ± 272.6	-511.9 ± 384.4	**-146.8** ± 450.6	-605.2 ± 389.6
Hopper	**2292.5** ± 1256.0	1638.7 ± 881.5	485.4 ± 389.3	1262.2 ± 803.3	780.8 ± 629.6
Walker2D	**3326.6** ± 1276.1	2977.8 ± 997.3	1447.1 ± 767.1	1341.6 ± 1092.6	1590.3 ± 860.7
HalfCheetah	**10366.7** ± 3477.2	7586.1 ± 4814.1	2834.6 ± 1062.9	6312.0 ± 2299.7	3258.2 ± 970.1
Reacher	**-13.5** ± 38.7	-17.5 ± 44.8	-122.2 ± 507.0	-77.2 ± 50.6	-168.2 ± 477.1
Reacher7DoF	**-15.2** ± 66.4	-17.2 ± 101.6	-78.9 ± 439.1	-114.2 ± 196.9	-93.8 ± 426.9

Table 2: The mean ± standard deviation of the average of the final returns over all hyper-parameter configurations. VMBPO is much more robust to change in hyper-parameters than the other baselines. We do not include PETS and STEVE because their hyper-parameters are directly adopted from their papers.

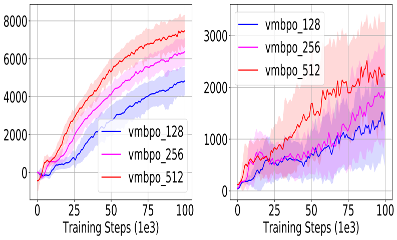

Figure 1: Performance of VMBPO with different number of synthetic samples.

the algorithms on a classical control task: *Pendulum*, and five MuJoCo tasks: *Hopper*, *Walker2D*, *HalfCheetah*, *Reacher*, and *Reacher7DoF*. We use similar neural network architectures (for the dynamics model, value functions, and policies) for VMBPO and MBPO. The detailed description of the network architectures and hyper-parameters is reported in Appendix E. Since we use a parametric representation for q_c in the E-step of VMBPO, as discussed in Section 5.2, we simply set $\pi = q_c^*$ in its M-step. We set the number of training steps to $400,000$ for the difficult environments (Walker2D, HalfCheetah), to $150,000$ for the medium one (Hopper), and to $50,000$ for the simpler ones (Pendulum, Reacher, Reacher7DOF). We evaluate policy performance every $1,000$ training steps. Each measurement is an average return over 5 episodes, generated with a separate random seed.

To illustrate the relative performance of the algorithms, we report the average return of VMBPO, VMBPO-MFE, and the baselines, with their best hyper-parameters, in Table 1 and Figure 2 (see Appendix E.1). The results show that VMBPO performs better than the baselines in most of the tasks, and usually converges faster even when the final performances are similar. The data-efficiency of VMBPO is mainly the result of using synthetic data generated by the learned model, and its extra performance can be attributed to jointly learning model and policy using a universal objective function.

The results also show that VMBPO-MFE outperforms MPO in 4 out of the 6 domains. However, in some cases its learning curve degrades and results in poor final performance. This is partly due to the instability in critic learning caused by sample variance amplification in exponential-TD minimization (see Eq. 39 in Sec. D.1). A way to alleviate this issue is to add a temperature term τ to the exponential-TD update [Borkar, 2002], although tuning this hyper-parameter is often non-trivial.[3]

To study the sensitivity of the algorithms w.r.t. the hyper-parameters, we report their performance averaged over all hyper-parameter/random-seed configurations in Table 2 and Figure 3 (see Appendix E.1). These results show that VMBPO is much more robust to change in hyper-parameters than the other algorithms, with the best average performance over all the tasks.

Ablation Study. We study the dependence of the VMBPO performance on the number of samples generated from the dynamics model q_d. Here we only experiment with two tasks, Hopper and HalfCheetah, and with fewer training steps $100,000$. At each step, we update the actor and critic using $\{128, 256, 512\}$ synthetic samples. Figure 1 shows the learning performance averaged over all hyper-parameter/random-seed configurations and illustrates how synthetic data can help with policy learning. The results show that increasing the amount of synthetic data generally improves the policy convergence rate. In the early phases, when the model is inaccurate, sampling from it may slow down learning, while in the later phases, with an improved model, adding more synthetic data can lead to a more significant performance boost.

7 Conclusion

We formulated the problem of jointly learning and improving model and policy in RL as a variational lower-bound of a log-likelihood and proposed EM-style algorithms to solve it. Our algorithm, called variational model-based policy optimization (VMBPO), uses model-based policy iteration for solving the E-step. We compared our (E-step) model-based and model-free algorithms with each other, and with a number of state-of-the-art model-based (e.g., MBPO) and model-free (e.g., MPO) RL algorithms, and showed its sample efficiency and performance.

We briefly discussed VMBPO algorithms in which the E-step is solved by value iteration methods. However, full implementation of these algorithms and studying their relationship with the existing methods requires more work that we leave for future. Another future directions are: **1)** finding more efficient implementation for VMBPO, and **2)** using VMBPO style algorithms in solving control problems from high-dimensional observations, by learning a low-dimensional latent space and a latent dynamics, and perform control there. This class of algorithms is referred to as learning controllable embedding [Watter *et al.*, 2015; Levine *et al.*, 2020; Cui *et al.*, 2021].

[3] The variance is further amplified with a large τ, but the critic learning is hampered by a small τ.

References

A. Abdolmaleki, J. Springenberg, Y. Tassa, R. Munos, N. Heess, and M. Riedmiller. Maximum a posteriori policy optimisation. In *Proceedings of the 6th International Conference on Learning Representations*, 2018.

D. Bertsekas. *Dynamic programming and optimal control*. Athena scientific Belmont, MA, 1995.

V. Borkar. Q-learning for risk-sensitive control. *Mathematics of operations research*, 27(2):294–311, 2002.

J. Buckman, D. Hafner, G. Tucker, E. Brevdo, and H. Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *Advances in Neural Information Processing Systems*, pages 8224–8234, 2018.

Y. Chebotar, M. Kalakrishnan, A. Yahya, A. Li, S. Schaal, and S. Levine. Path integral guided policy search. In *IEEE International Conference on Robotics and Automation*, 2017.

K. Chua, R. McAllister, R. Calandra, and S. Levine. Unsupervised exploration with deep model-based reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2019.

B. Cui, Y. Chow, and M. Ghavamzadeh. Control-aware representations for model-based reinforcement learning. In *Proceedings of the Ninth International Conference on Learning Representations*, 2021.

P. Dayan and G. Hinton. Using expectation-maximization for reinforcement learning. *Neural Computation*, 9(2):271–278, 1997.

M. Fellows, A. Mahajan, T. Rudner, and S. Whiteson. VIREL: A variational inference framework for reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 7120–7134, 2019.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1861–1870, 2018.

H. Hachiya, J. Peters, and M. Sugiyama. Efficient sample reuse in EM-based policy search. In *Proceedings of the European Conference on Machine Learning*, 2009.

M. Janner, J. Fu, M. Zhang, and S. Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems 32*, pages 12519–12530, 2019.

H. Kappen, V. Gomez, and M. Opper. Optimal control as a graphical model inference problem. *Machine Learning*, 87(2):159–182, 2012.

S. Levine and P. Abbeel. Learning neural network policies with guided policy search under unknown dynamics. In *Advances in Neural Information Processing Systems*, 2014.

S. Levine and V. Koltun. Variational policy search via trajectory optimization. In *Advances in Neural Information Processing Systems*, 2013.

N. Levine, Y. Chow, R. Shu, A. Li, M. Ghavamzadeh, and H. Bui. Prediction, consistency, curvature: Representation learning for locally-linear control. In *Proceedings of the 8th International Conference on Learning Representations*, 2020.

S. Levine. Reinforcement learning and control as probabilistic inference: Tutorial and review. *arXiv:1805.00909*, 2018.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing Atari with deep reinforcement learning. *preprint arXiv:1312.5602*, 2013.

V. Mnih, K. Kavukcuoglu, D. Silver, A. Rusu, J. Veness, M. Bellemare, A. Graves, M. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

O. Nachum, M. Norouzi, K. Xu, and D. Schuurmans. Bridging the gap between value and policy based reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2775–2785, 2017.

G. Neumann. Variational inference for policy search in changing situations. In *Proceedings of the 28th international conference on machine learning*, pages 817–824, 2011.

X. Nguyen, M. Wainwright, and M. Jordan. Estimating divergence functionals and the likelihood ratio by penalized convex risk minimization. In *Advances in neural information processing systems*, pages 1089–1096, 2008.

J. Peters and S. Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on machine learning*, 2007.

J. Peters, K. Mulling, and Y. Altun. Relative entropy policy search. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, 2010.

K. Rawlik, M. Toussaint, and S. Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. In *Proceedings of Robotics: Science and Systems*, 2013.

J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1889–1897, 2015.

R. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of the 7th International Conference on Machine Learning*, 1990.

E. Todorov. General duality between optimal control and estimation. In *Proceedings of the 47th IEEE Conference on Decision and Control*, pages 4286–4292, 2008.

M. Toussaint. Robot trajectory optimization using approximate inference. In *Proceedings of the 26th International Conference on Machine Learning*, pages 1049–1056, 2009.

M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in Neural Information Processing Systems 28*, pages 2746–2754. 2015.

A Proofs of Section 4

A.1 Proof of Lemma 1

Before proving Lemma 1, we first state and prove the following results.

Lemma 5. *For any state $x \in \mathcal{X}$, action-value function Q, and policy π, we have*

$$\max_{q_c \in \Delta_{\mathcal{A}}} \mathbb{E}_{a \sim q_c(\cdot|x)} \Big[Q(x,a) - \log \frac{q_c(a|x)}{\pi(a|x)} \Big] = \log \mathbb{E}_{a \sim \pi(\cdot|x)} \big[\exp \big(Q(x,a) \big) \big]. \tag{28}$$

Analogously, for any state-action pair $(x,a) \in \mathcal{X} \times \mathcal{A}$, value function V, and transition kernel $p(\cdot|x,a)$, we have

$$\max_{q_d \in \Delta_{\mathcal{X}}} \mathbb{E}_{x' \sim q_d(\cdot|x,a)} \Big[V(x') - \log \frac{q_d(x'|x,a)}{p(x'|x,a)} \Big] = \log \mathbb{E}_{x' \sim p(\cdot|x,a)} \big[\exp \big(V(x') \big) \big]. \tag{29}$$

Proof. We only prove (28) here, since the proof of (29) follows similar arguments. The proof of (28) comes from the following sequence of equalities:

$$\begin{aligned}
\max_{q_c \in \Delta_{\mathcal{A}}} \mathbb{E}_{a \sim q_c(\cdot|x)} \Big[Q(x,a) - \log \frac{q_c(a|x)}{\pi(a|x)} \Big] &= \max_{q_c \in \Delta_{\mathcal{A}}} \mathbb{E}_{a \sim q_c(\cdot|x)} [(Q(x,a) + \log \pi(a|x)) - \log q_c(a|x)] \\
&\overset{\text{(a)}}{=} \log \int_a \exp(Q(x,a) + \log \pi(a|x)) = \log \mathbb{E}_{a \sim \pi(\cdot|x)} \big[\exp \big(Q(x,a) \big) \big].
\end{aligned}$$

(a) This follows from Lemma 4 in Nachum *et al.* [2017]. □

We now turn to the proof of our main lemma.

Proof of Lemma 1. From (6), for any state $x \in \mathcal{X}$, we may write the q_c-induced operator as

$$\begin{aligned}
\mathcal{T}_{q_c}[V](x) &= \mathbb{E}_{a \sim q_c(\cdot|x)} \Big[\eta \cdot r(x,a) - \log \frac{q_c(a|x)}{\pi(a|x)} + \max_{q_d \in \Delta_{\mathcal{X}}} \mathbb{E}_{x' \sim q_d(\cdot|x,a)} \big[V(x') - \log \frac{q_d(x'|x,a)}{p(x'|x,a)} \big] \Big] \\
&\overset{\text{(a)}}{=} \mathbb{E}_{a \sim q_c(\cdot|x)} \Big[\eta \cdot r(x,a) - \log \frac{q_c(a|x)}{\pi(a|x)} + \log \mathbb{E}_{x' \sim p(\cdot|x,a)} \big[\exp(V(x')) \big] \Big] \\
&\overset{\text{(b)}}{=} \mathbb{E}_{a \sim q_c(\cdot|x)} \Big[Q(x,a) - \log \frac{q_c(a|x)}{\pi(a|x)} \Big].
\end{aligned}$$

(a) From Lemma 5.
(b) From the definition of the Q-function in (9).
This concludes the proof of (10), the first statement of Lemma 1. Now to prove the second statement (Eq. 11), we may write

$$\begin{aligned}
\mathcal{T}[V](x) &= \max_{q_c \in \Delta_{\mathcal{A}}} \mathbb{E}_{a \sim q_c(\cdot|x)} \Big[\eta \cdot r(x,a) - \log \frac{q_c(a|x)}{\pi(a|x)} + \max_{q_d \in \Delta_{\mathcal{X}}} \mathbb{E}_{x' \sim q_d(\cdot|x,a)} \big[V(x') - \log \frac{q_d(x'|x,a)}{p(x'|x,a)} \big] \Big] \\
&\overset{\text{(a)}}{=} \max_{q_c \in \Delta_{\mathcal{A}}} \mathbb{E}_{a \sim q_c(\cdot|x)} \Big[\eta \cdot r(x,a) - \log \frac{q_c(a|x)}{\pi(a|x)} + \log \mathbb{E}_{x' \sim p(\cdot|x,a)} \big[\exp(V(x')) \big] \Big] \\
&\overset{\text{(b)}}{=} \log \mathbb{E}_{a \sim \pi(\cdot|x)} \Big[\exp \Big(\eta \cdot r(x,a) + \log \mathbb{E}_{x' \sim p(\cdot|x,a)} \big[\exp \big(V(x') \big) \big] \Big) \Big] \\
&= \log \mathbb{E}_{a \sim \pi(\cdot|x)} \Big[\exp \big(\eta \cdot r(x,a) \big) \mathbb{E}_{x' \sim p(\cdot|x,a)} \big[\exp \big(V(x') \big) \big] \Big] \\
&= \log \mathbb{E}_{a \sim \pi(\cdot|x), x' \sim p(\cdot|x,a)} \big[\exp \big(\eta \cdot r(x,a) + V(x') \big) \big].
\end{aligned}$$

(a) and **(b)** both come from Lemma 5.
This concludes the proof of (11), the second statement of Lemma 1. □

A.2 Proof of Lemma 2

We only prove the properties of the optimal operator, $\mathcal{T}$, here. The proof for the q_c-induced operator, $\mathcal{T}_{q_c}$, follows similar arguments.

1. Monotonicity: For any functions $V, W : \mathcal{X} \to \mathbb{R}$, such that $V(x) \leq W(x)$, $\forall x \in \mathcal{X}$, we have $\mathcal{T}[V](x) \leq \mathcal{T}[W](x)$, $\forall x \in \mathcal{X}$.

Proof. For the case of $x \in \mathcal{X}^0$ (the set of terminal states), the property trivially holds. For the case of $x \in \mathcal{X}$, from the definition of the optimal operator in (11), it is easy to see that for all $x \in \mathcal{X}$, we have

$$\eta \cdot r(x,a) + \log \int_{x'} p(x'|x,a) \exp\big(V(x')\big) \leq \eta \cdot r(x,a) + \log \int_{x'} p(x'|x,a) \exp\big(W(x')\big),$$

which means $\mathcal{T}[V](x) = \log \mathbb{E}_{a\sim\pi(\cdot|x),x'\sim p(\cdot|x,a)}\big[\exp\big(\eta \cdot r(x,a) + V(x')\big)\big] \leq \mathcal{T}[W](x) = \log \mathbb{E}_{a\sim\pi(\cdot|x),x'\sim p(\cdot|x,a)}\big[\exp\big(\eta \cdot r(x,a) + W(x')\big)\big]$, $\forall x \in \mathcal{X}$. This completes the proof of monotonicity. □

2. Contraction: There exists a vector with positive components, i.e., $\rho : \mathcal{X} \to \mathbb{R}_{\geq 0}$, and a discounting factor $0 < \gamma < 1$ such that

$$\|\mathcal{T}[V] - \mathcal{T}[W]\|_\rho \leq \gamma \|V - W\|_\rho,$$

where the weighted norm is defined as $\|V\|_\rho = \max_{x\in\mathcal{X}} \frac{V(x)}{\rho(x)}$.

Proof. For the case of $x \in \mathcal{X}^0$ (the set of terminal states), the property trivially holds because the contraction maps to zero. For the case of $x \in \mathcal{X}$, following the construction in Proposition 3.3.1 in Bertsekas [1995], consider a risk-sensitive entropy-regularized stochastic shortest path problem (via dynamic exponential risk formulation from Borkar [2002]), where the reward are all equal to $1/\eta$. Based on similar arguments as in Proposition 3.3.1, there exists a fixed point value function $\hat{V}$, such that

$$\hat{V}(x) = 1 + \max_{q_c \in \Delta_\mathcal{A}} \int_a q_c(a|x) \left(\log \mathbb{E}_{x'\sim p(\cdot|x,a)}\big[\exp\big(\hat{V}(x')\big)\big] - \log \frac{q_c(a|x)}{\pi(a|x)} \right).$$

Using the results from Lemma 1, the above statement further implies that:

$$\hat{V}(x) = 1 + \log \mathbb{E}_{a\sim\pi(\cdot|x),x'\sim p(\cdot|x,a)}\big[\exp\big(\hat{V}(x')\big)\big].$$

Notice that $\hat{V}(x) \geq 1$ for all $x \in \mathcal{X}$. By defining $\rho(x) = \hat{V}(x)$, and by constructing $\gamma = \max_{x\in\mathcal{X}} (\rho(x) - 1)/\rho(x)$, one immediately has $0 < \gamma < 1$, and

$$\begin{aligned}
&\max_{q_c \in \Delta_\mathcal{A}} \mathbb{E}_{a\sim q_c(\cdot|x)}\Big[\max_{q_d \in \Delta_\mathcal{X}} \int_{x'} q_d(x'|x,a)\big(V(x') - \log \frac{q_d(x'|x,a)}{p(x'|x,a)}\big) - \log \frac{q_c(a|x)}{\pi(a|x)}\Big] \\
&= \log \mathbb{E}_{a\sim\pi(\cdot|x),x'\sim p(\cdot|x,a)}\big[\exp\big(\hat{V}(x')\big)\big] = \rho(x) - 1 \leq \gamma\rho(x).
\end{aligned}$$

Then by following the same lines of analysis as in Proposition 1.5.2 of Bertsekas [1995], one can show that $\mathcal{T}$ is a contraction operator. □

3. Unique Fixed-point Solution The optimal value function V_π is its unique fixed-point, i.e., $\mathcal{T}[V_\pi](x) = V_\pi(x)$, $\forall x \in \mathcal{X}$.

Proof. Let $V_\pi(x)$ be the optimal value function of the E-step problem in Eq. 5, and let V^* be a fixed point solution: $V(x) = \mathcal{T}[V](x)$, for any $x \in \mathcal{X}$. For the case when $x \in \mathcal{X}^0$, the following result trivially holds: $V_\pi(x) = \mathcal{T}[V_\pi](x) = V^*(x) = 0$. Below, we show the equality for the case of $x_0 \in \mathcal{X}$.

First, we want to show that $V_\pi(x_0) \leq V^*(x_0)$. Consider the greedy policy $\overline{q}_c^*$ constructed from the Bellman operator $\arg\max_{q_c \in \Delta} \mathcal{T}_{q_c}[V^*](x)$. Recall that $V^*(x)$ is a fixed point solution to $V(x) = \mathcal{T}[V](x)$, for any $x \in \mathcal{X}$. Then for any bounded initial value function V_0, the contraction property of Bellman operator $\mathcal{T}_{\overline{q}_c^*}$ implies that

$$\begin{aligned}
V^*(x) &= \lim_{n\to\infty} \mathcal{T}_{\overline{q}_c^*}^n [V_0](x) \\
&= \lim_{n\to\infty} \max_{q_d \in \Delta_\mathcal{X}} \mathbb{E}\big[\sum_{t=0}^{n-1} \eta \cdot r(x_t, a_t) - \mathrm{KL}(\overline{q}_c^* || \pi)(x_t) - \mathrm{KL}(q_d || p)(x_t, a_t) \mid q_d, \overline{q}_c^*, P_0\big],
\end{aligned}$$

for which the transient assumption of stopping MDPs further implies that

$$V^*(x) = \max_{q_d \in \Delta_\mathcal{X}} \mathbb{E}\big[\sum_{t=0}^{\mathrm{T}^*-1} \eta \cdot r(x_t, a_t) - \mathrm{KL}(\overline{q}_c^* || \pi)(x_t) - \mathrm{KL}(q_d || p)(x_t, a_t) \mid q_d, \overline{q}_c^*, P_0\big].$$

Since $\overline{q}_c^*$ is a feasible solution to the E-step problem, this further implies that $V_\pi(x_0) \leq V^*(x_0)$.

Second, we want to show that $V_\pi(x_0) \geq V^*(x_0)$. Consider the optimal policy q_c^* of the E-step problem. Note that V^* is a fixed point solution to equation: $V^*(x) = \mathcal{T}[V^*](x)$, for any $x \in \mathcal{X}$. Immediately the above result yields the following inequality:

$$V^*(x) = \mathcal{T}_{\overline{q}_c^*}[V^*](x) \leq \mathcal{T}_{q_c^*}[V^*](x), \ \forall x \in \mathcal{X},$$

the first equality holds because $\overline{q}_c^*(\cdot|x)$ is the minimizer of the optimization problem in $\mathcal{T}[V^*](x)$, $x \in \mathcal{X}$. By recursively applying Bellman operator $\mathcal{T}_{q_c^*}$, one has the following result:

$$\begin{aligned} V^*(x) &\leq \lim_{n\to\infty} \mathcal{T}_{q_c^*}^n [V^*](x) \\ &= \max_{q_d \in \Delta_{\mathcal{X}}} \mathbb{E}\big[\sum_{t=0}^{\mathrm{T}^*-1} \eta \cdot r(x_t, a_t) - \mathrm{KL}(q_c^* || \pi)(x_t) - \mathrm{KL}(q_d||p)(x_t, a_t) \mid q_d, q_c^*, x_0 = x \big] = V_\pi(x), \ \forall x \in \mathcal{X}. \end{aligned}$$

Combining the above analysis, we prove the claim of $V_\pi(x_0) = V^*(x_0)$, and the greedy policy of the fixed-point equation, i.e., $\overline{q}_c^*$, is an optimal policy to the E-step problem. □

A.3 Proof of Proposition 1

Proof. The proof follows by combining the definition of the Q-function (9) and Lemma 2 that indicates V_π is the unique fixed-point of the optimal operator $\mathcal{T}$. Therefore, for any $x \in \mathcal{X}$, we can write

$$\begin{aligned} V_\pi(x) = \mathcal{T}[V_\pi](x) &\overset{\text{(a)}}{=} \log \mathbb{E}_{a\sim\pi(\cdot|x)} \Big[\exp(\eta \cdot r(x,a)) \cdot \mathbb{E}_{x'\sim p(\cdot|x,a)} \big[\exp(V(x')) \big] \Big] \\ &\overset{\text{(b)}}{=} \log \mathbb{E}_{a\sim\pi(\cdot|x)} \big[\exp(Q(x,a)) \big]. \end{aligned}$$

(a) This is from (11), the second statement of Lemma 1.
(b) If we apply exponential to both sides of (9), we see that what is inside the bracket is equal to $\exp(Q(x,a))$.
This concludes the proof. □

A.4 Proof of Lemma 3

Proof. Since the variational policy, q_c^Q is the solution to the optimization problem (13), following Corollary 6 in Nachum *et al.* [2017], we may write that

$$q_c^Q(a|x) = \frac{\exp\big(Q(x,a) + \log \pi(a|x)\big)}{\int_a \exp\big(Q(x,a) + \log \pi(a|x)\big)} = \frac{\pi(a|x) \cdot \exp\big(Q(x,a)\big)}{\mathbb{E}_{a\sim\pi(\cdot|x)}\big[\exp\big(Q(x,a)\big)\big]} \ .$$

This proves (15), the second statement of Lemma 3. To prove (14), the first statement of Lemma 3, we use the fact that the variational dynamics, q_d^V is the solution to the optimization problem (12), and thus, following Corollary 6 in Nachum *et al.* [2017], we may write that

$$q_d^V(x'|x,a) = \frac{\exp\big(V(x') + \log p(x'|x,a)\big)}{\int_a \exp\big(V(x') + \log p(x'|x,a)\big)} = \frac{p(x'|x,a) \cdot \exp\big(V(x')\big)}{\mathbb{E}_{x'\sim p(\cdot|x,a)}\big[\exp\big(V(x')\big)\big]} \ .$$

This completes the proof. The second equality in (14) is straightforward, because by taking exponential from both sides of (9), we have

$$\mathbb{E}_{x'\sim p(\cdot|x,a)}\big[\exp\big(V(x')\big)\big] = \exp\big(Q(x,a) - \eta \cdot r(x,a)\big).$$

□

A.5 Proof of Lemma 4

In Lemma 2, we proved that the q_c-induced operator, $\mathcal{T}_{q_c}$, is monotonic and contraction. Therefore, it is clear that for any q_c, starting from an arbitrary value function V and iteratively applying $\mathcal{T}_{q_c}$, we will converge to the fixed-point of this operator, i.e., $V_{q_c} = \mathcal{T}_{q_c} V_{q_c}$. This proves that the policy evaluation step at each iteration k takes $q_c^{(k)}$, as input and returns the $q_c^{(k)}$-induced value function $V_{q_c^{(k)}}$. What needs to be proved is the policy improvement step to show that $V_{q_c^{(k+1)}}(x) \geq V_{q_c^{(k)}}(x), \ \forall x \in \mathcal{X}$, when for all $x \in \mathcal{X}$ and $a \in \mathcal{A}$, we have

$$q_c^{(k+1)}(a|x) = \frac{\pi(a|x) \cdot \exp\big(Q_{q_c^{(k)}}(x,a)\big)}{\mathbb{E}_{a\sim\pi(\cdot|x)}\big[\exp\big(Q_{q_c^{(k)}}(x,a)\big)\big]},$$

and

$$Q_{q_c^{(k)}}(x,a) = \eta \cdot r(x,a) + \log \mathbb{E}_{x' \sim p(\cdot|x,a)}\big[\exp\big(V_{q_c^{(k)}}(x')\big)\big].$$

Proof. Since from (13), for all $x \in \mathcal{X}$ and $a \in \mathcal{A}$, we have

$$q_c^{(k+1)}(a|x) = \arg\max_{q_c} \mathbb{E}_{a\sim q_c(\cdot|x)}\big[Q_{q_c^{(k)}}(x,a) - \log\frac{q_c(a|x)}{\pi(a|x)}\big],$$

for all $x \in \mathcal{X}$, we may write

$$\begin{aligned}\mathbb{E}_{a\sim q_c^{(k+1)}(\cdot|x)}\big[Q_{q_c^{(k)}}(x,a) - \log\frac{q_c^{(k+1)}(a|x)}{\pi(a|x)}\big] &\geq \mathbb{E}_{a\sim q_c^{(k)}(\cdot|x)}\big[Q_{q_c^{(k)}}(x,a) - \log\frac{q_c^{(k)}(a|x)}{\pi(a|x)}\big] \\ &\overset{(a)}{=} \mathcal{T}_{q_c^{(k)}}[V_{q_c^{(k)}}](x) = V_{q_c^{(k)}}(x). \end{aligned} \tag{30}$$

(a) This is from (10).

We know that if we start from any arbitrary value function V and iteratively apply $\mathcal{T}_{q_c^{(k+1)}}$, we will convergence to $V_{q_c^{(k+1)}}$. If we start from $V = V_{q_c^{(k)}}$, for all $x \in \mathcal{X}$, we have

$$\begin{aligned} V_{q_c^{(k+1)}}(x) &= \lim_{n\to\infty} \mathcal{T}^n_{q_c^{(k+1)}}[V_{q_c^{(k)}}](x) = \lim_{n\to\infty} \mathcal{T}^{n-1}_{q_c^{(k+1)}}\Big[\mathcal{T}_{q_c^{(k+1)}}[V_{q_c^{(k)}}]\Big](x) \\ &\overset{(a)}{=} \lim_{n\to\infty} \mathcal{T}^{n-1}_{q_c^{(k+1)}}\Big[\mathbb{E}_{a\sim q_c^{(k+1)}(\cdot|x)}\big[Q_{q_c^{(k)}}(x,a) - \log\frac{q_c^{(k+1)}(a|x)}{\pi(a|x)}\big]\Big] \\ &\overset{(b)}{\geq} \lim_{n\to\infty} \mathcal{T}^{n-1}_{q_c^{(k+1)}}[V_{q_c^{(k)}}](x) \geq \ldots \geq \mathcal{T}_{q_c^{(k+1)}}[V_{q_c^{(k)}}](x) \geq V_{q_c^{(k)}}(x). \end{aligned}$$

(a) This is by replacing $\mathcal{T}_{q_c^{(k+1)}}[V_{q_c^{(k)}}](x)$ with $\mathbb{E}_{a\sim q_c^{(k+1)}(\cdot|x)}\big[Q_{q_c^{(k)}}(x,a) - \log\frac{q_c^{(k+1)}(a|x)}{\pi(a|x)}\big]$ from (10).
(b) This is from (30) and the monotonicity of the operator $\mathcal{T}_{q_c^{(k+1)}}$ from Lemma 2.
This concludes the proof. □

The policy improvement step of the model-based and model-free PI algorithms discussed in Section 4.2 perform the update of Eq. 15. Calculating the denominator of this update when the number of actions is large or infinite (continuous action space) could not be done efficiently. In this case, similar to a number of algorithms in the literature (e.g., soft actor-critic), we replace the update of Eq. 15 with the following KL minimization:

$$q_c^{(k+1)}(\cdot|x) = \arg\min_{q_c\in\Delta_{\mathcal{A}}} \text{KL}\left(q_c(\cdot|x) \;||\; \frac{\pi(\cdot|x)\cdot\exp\big(Q_{q_c^{(k)}}(x,\cdot)\big)}{\mathbb{E}_{a\sim\pi(\cdot|x)}\big[\exp\big(Q_{q_c^{(k)}}(x,a)\big)\big]}\right), \quad \forall x \in \mathcal{X}. \tag{31}$$

Now in the following corollary, we prove that even in this case we will see policy improvement, and thus, the algorithms will eventually converge to the optimal variational distributions $q^* = (q_c^*, q_d^*)$.

Corollary 6. *Let at iteration k, the variational policy, $q_c^{(k+1)}$, is computed as the exact solution to the KL optimization (31). Then, we have $V_{q_c^{(k+1)}}(x) \geq V_{q_c^{(k)}}(x),\ \forall x \in \mathcal{X}$.*

Proof. Since $q_c^{(k+1)}$ is the minimizer of (31), we may write

$$\text{KL}\left(q_c^{(k+1)}(a|x) \;||\; \frac{\pi(a|x)\cdot\exp\big(Q_{q_c^{(k)}}(x,a)\big)}{\mathbb{E}_{a\sim\pi(\cdot|x)}\big[\exp\big(Q_{q_c^{(k)}}(x,a)\big)\big]}\right) \leq \text{KL}\left(q_c^{(k)}(a|x) \;||\; \frac{\pi(a|x)\cdot\exp\big(Q_{q_c^{(k)}}(x,a)\big)}{\mathbb{E}_{a\sim\pi(\cdot|x)}\big[\exp\big(Q_{q_c^{(k)}}(x,a)\big)\big]}\right). \tag{32}$$

Let $Z_{q_c^{(k)}}(x) := E_{a\sim\pi(\cdot|x)}\big[\exp\big(Q_{q_c^{(k)}}(x,a)\big)\big]$, then we may rewrite (32) as

$$\begin{aligned} &\mathbb{E}_{a\sim q_c^{(k+1)}(\cdot|x)}\big[Q_{q_c^{(k)}}(x,a) - \log\frac{q_c^{(k+1)}(a|x)}{\pi(a|x)}\big] - \mathbb{E}_{a\sim q_c^{(k+1)}}\big[Z_{q_c^{(k)}}(x)\big] \geq \\ &\qquad\qquad \mathbb{E}_{a\sim q_c^{(k)}(\cdot|x)}\big[Q_{q_c^{(k)}}(x,a) - \log\frac{q_c^{(k)}(a|x)}{\pi(a|x)}\big] - \mathbb{E}_{a\sim q_c^{(k)}}\big[Z_{q_c^{(k)}}(x)\big]. \end{aligned}$$

Since $Z_{q_c^{(k)}}(\cdot)$ is a function of x, we have

$$\mathbb{E}_{a\sim q_c^{(k+1)}(\cdot|x)}\big[Q_{q_c^{(k)}}(x,a) - \log\frac{q_c^{(k+1)}(a|x)}{\pi(a|x)}\big] - Z_{q_c^{(k)}}(x) \geq \\ \mathbb{E}_{a\sim q_c^{(k)}(\cdot|x)}\big[Q_{q_c^{(k)}}(x,a) - \log\frac{q_c^{(k)}(a|x)}{\pi(a|x)}\big] - Z_{q_c^{(k)}}(x).$$

Thus, the Z terms are removed from both sides of the above inequality and we return to Eq. 30 in the proof of Lemma 4. The rest of the proof is similar to that of Lemma 4. □

B Value Iteration for the E-step

Similar to the E-step of the VMBPO algorithm, at the beginning of the E-step we generate a number of samples (x, a, r, x') from the current baseline policy π and add them to the buffer $\mathcal{D}$. The E-step consists of four updates: **1)** computing the variational dynamics q_d, **2)** estimating the log-likelihood ratio $\log(q_d/p)$, **3)** computing the updated value, V, and action-value, Q, functions (critic update), and finally **4)** computing the variational policy new q_c (actor update). Step **1)**, Step **2)**, and Step **4)** are identical to that in the VMBPO algorithm (with the corresponding value functions estimated by fitted Q iteration). We therefore focus on describing Step **3)** below for both the model-based and model-free cases.

B.1 Step 3: The Model-based E-step Critic Update

To compute V (fixed-point of $\mathcal{T}$) and its action-value Q, we first rewrite (6) with the maximizer q_d from Step 1 and the log-likelihood ratio $\log(q_d/p)$ from Step 2:

$$\begin{aligned}\mathcal{T}[V](x) &= \max_{q_c \in \Delta_{\mathcal{A}}} \mathbb{E}_{a \sim q_c(\cdot|x)} \big[\eta \cdot r(x,a) - \log \frac{q_c(a|x)}{\pi(a|x)} + \mathbb{E}_{x' \sim q_d(\cdot|x,a;\theta_d)}[V'(x';\theta'_v) - \nu(x'|x,a;\theta_\nu)]\big] \\ &= \log \mathbb{E}_{a \sim \pi(\cdot|x)} \big[\exp\big(\eta \cdot r(x,a) + \mathbb{E}_{x' \sim q_d(\cdot|x,a;\theta_d)}[V'(x';\theta'_v) - \nu(x'|x,a;\theta_\nu)]\big)\big]\end{aligned} \tag{33}$$

Since $\mathcal{T}$ can be written as both (11) and (33), we compute the Q-function by setting the RHS of these equations equal to each other, i.e., for all $x \in \mathcal{X}$ and $a \in \mathcal{A}$,

$$Q(x,a;\theta_q) = \eta \cdot r(x,a) + \mathbb{E}_{x' \sim q_d(\cdot|x,a;\theta_d)}[V'(x';\theta'_v) - \nu(x'|x,a;\theta_\nu)]. \tag{34}$$

Similar to that in VMBPO, since the expectation in the above expression is w.r.t. the variational dynamics (model) q_d, we can estimate Q only with samples generated from the model. We learn θ_q by minimizing the following square-loss:

$$\theta_q = \arg\min_\theta \sum_{(x,a,r,x') \sim \mathcal{E}} \big(Q(x,a;\theta) - \eta \cdot r - V'(x';\theta'_v) + \nu(x'|x,a;\theta_\nu)\big)^2, \tag{35}$$

without sampling from the true environment. On the other hand, we approximate V, the fixed-point of $\mathcal{T}$ (i.e., $V(x) = \mathcal{T}[V](x)$, $\forall x \in \mathcal{X}$), using the $\mathcal{T}$ definition in (33). This results in updating V by taking several steps in the direction of the gradient of a sample average of the square-loss obtained by setting the two sides of the fixed-point equation to be equal, i.e.,

$$\theta_v = \arg\min_\theta \sum_{x,a} \big(\exp\big(Q(x,a;\theta_q) - V(x;\theta)\big) - 1\big)^2, \tag{36}$$

where x is randomly sampled and $a \sim \pi(\cdot|x)$.

B.2 Step 3: The Model-free E-step Critic Update

Suppose we have access to the policy π, we now aim to learn the corresponding value functions (critic) V and Q. Recall from Lemma 1, we know that V is a unique solution of fixed-point equation $\mathcal{T}[V](x) = V(x)$, $\forall x \in \mathcal{X}$. Suppose we parameterize $V(x)$ with function approximation $\hat{V}(x;\theta_v)$, and similarly $Q(x,a)$ with $\hat{Q}(x,a;\theta_q)$. Similar to soft DQN, one way to learn $\hat{V}(x;\theta_v)$ and $\hat{Q}(x,a;\theta_q)$ is by minimizing the following objective function respectively, over the data from the replay buffer $\mathcal{D}$ sampled from the environment:

$$\theta^*_q \leftarrow \arg\min_\theta \sum_{(x,a,r,x') \sim \mathcal{D}} \Big(\exp\Big(\hat{Q}(x,a;\theta) - \eta r - \hat{V}(x';\theta'_v)\Big) - 1\Big)^2, \tag{37}$$

where $V(x';\theta'_v)$ is a *target* Q-function, and

$$\theta^*_v \leftarrow \arg\min_\theta \sum_{x,a \sim \pi(\cdot|x)} \big(\exp\big(\hat{Q}(x,a;\theta_q) - V(x;\theta)\big) - 1\big)^2. \tag{38}$$

The above learning is completely *off-policy*—the target is valid no matter how the experience was generated (as long as it is sufficiently exploratory). Under this loss, critic learning can be viewed as ℓ_2-regression of $\exp(\hat{Q}(x,a;\theta_q) - \eta r - V(x;\theta_v))$ w.r.t. the target label 1, such that the value function $V(x;\theta_v)$ is learned to minimize the the *mean squared Bellman error*: $(\mathcal{T}[V](x) - V(x))^2$, i.e., $\exp(V(x;\theta_v)) = \mathbb{E}_{a \sim \pi(\cdot|x), x' \sim p(\cdot|x,a)}\big[\exp\big(\eta \cdot r(x,a) + V(x';\theta_v)\big)\big] = \mathbb{E}_{a \sim \pi(\cdot|x)}\big[\exp(\hat{Q}(x,a;\theta_q))\big]$, $\forall x \in \mathcal{X}$, and to enforce $\hat{Q}(x,a;\theta_q) = \eta \cdot r(x,a) + \log \int_{x' \in \mathcal{X}} P(x'|x,a) \exp \hat{V}(x';\theta_v)$.

C Pseudo-code of VMBPO

This section contains the pseudo-code of our variational model-based policy optimization (VMBPO) algorithm, whose E-step and M-step have been described in details in Sections 5.1 and 5.2.

Algorithm 1 Variational Model-based Policy Optimization (VMBPO)

```
**Inputs**: replay buffer $\mathcal{D}$;   neural networks representing variational dynamics $\theta_d$, variational policy $\theta_c$, log-likelihood ratio $\theta_\nu$, value function $\theta_v$, action-value function $\theta_q$, target value function $\theta_v'$, target action-value function $\theta_q'$, baseline policy $\theta_\pi$;
**for** $t = 1, 2, \dots$ **do**
    **for** a number of interactions with the environment **do**
        Observe state $x$;   Take action $a \sim \pi(\cdot|x;\theta_\pi)$;   Observe $r = r(x,a) \ \wedge \ x' \sim p(\cdot|x,a)$;
        Update the buffer $\mathcal{D} \leftarrow \mathcal{D} \cup (x,a,r,x')$;   Replace $x \leftarrow x'$;
    **end for**
    # E-step   ($K$ is the number of E-step iterations)
    **for** $k = 1, \dots, K$ **do**
        # Step 1   (updating variational dynamics $q_d$)
        Sample a number of $(x,a,r,x') \sim \mathcal{D}$;   Update $q_d$ parameter $\theta_d$ using gradient of (19);
        # Step 2   (updating log-likelihood ratio $\nu = \log(q_d/p)$)
        Sample a number of $(x,a,x') \sim \mathcal{D}$;   Sample $x' \sim q_d$ for the same $(x,a)$;
        Update $\nu$ parameter $\theta_\nu$ using gradient of (21);
        # Step 3   (critic update  –  updating $V_{q_c}$ and $Q_{q_c}$)
        Sample a number of $(x,a,r,x')$ from the model;   # $a \sim q_c(\cdot|x), \ x' \sim q_d(\cdot|x,a)$
        Update $Q$ parameter $\theta_q$ using gradient of (23);
        Sample a number of $(x,a)$ from the model;   # $a \sim q_c(\cdot|x)$
        Update $V$ parameter $\theta_v$ using gradient of (36);
        # Step 4   (actor update  –  updating $q_c$  –  policy improvement)
        Update $q_c$ parameter $\theta_c$ either using gradient of (25) or by solving (15) in closed-form;
        # target networks $\theta_v', \theta_q'$ are set to an exponentially moving average of the value networks $\theta_v, \theta_q$
        $\theta_v' \leftarrow \tau\theta_v + (1-\tau)\theta_v'$;   $\theta_q' \leftarrow \tau\theta_q + (1-\tau)\theta_q'$;
    **end for**
    # M-step   (updating the baseline policy $\pi$)
    Update baseline policy $\pi$ parameter $\theta_\pi$ either by setting it to $\theta_c$ or by solving the MAP problem (27)
**end for**
```

D E-step with Model-free Policy Iteration

In Sec. 4.2, we described a model-free PI algorithm for solving the E-step of our variational formulation. In this algorithm, q_d is computed at the end of the E-step, and thus, only used to generate samples in the M-step. We call the RL-version of this algorithm, whose E-step is model-free and M-step is model-based, VMBPO with model-free E-step (VMBPO-MFE). In VMBPO-MFE, we first estimate Q_{q_c} using (9) and then approximate V_{q_c} by minimizing the (fixed-point) loss $(V - \mathcal{T}_{q_c}V)^2$, where $\mathcal{T}_{q_c}$ is computed from (10) by setting Q equal to the target action-value network Q'. The q_c update (policy improvement) is exactly the same as the actor update (Step 4) of VMBPO, described in Sec. 5.1. After several E-step updates, q_d is computed from (14). This is followed by the M-step, which is identical to that of VMBPO, described in Sec. 5.2. We report the details of VMBPO-MFE and its pseudo-code in Appendix D. Although VMBPO-MFE is less complex than VMBPO, our experiments in Sec. 6 show that it is less sample efficient (i.e., achieves worse performance than VMBPO with the same number of real samples). The main reason for this is that VMBPO uses simulated samples in both E and M steps, while VMBPO-MFE only uses them in the M-step. Moreover, our experiments show that VMBPO-MFE may degenerate in certain cases, due to the instability of the exponential temporal difference (TD) learning in the critic step. The $\log$ expectation term in (9) creates challenges for finding an unbiased empirical loss to estimate Q, and when it is removed by taking exponential from both sides of (9), the resulting exponential terms cause numerical instability in the updates.

Algorithm 2 Variational Model-based Policy Optimization with Model-free E-step (VMBPO-MFE)

```
Inputs: replay buffer $\mathcal{D}$;   neural networks representing variational policy $\theta_c$, value function $\theta_v$, action-value function $\theta_q$, target value function $\theta'_v$, target action-value function $\theta'_q$, baseline policy $\theta_\pi$;
for $t = 1, 2, \dots$ do
    for a number of interactions with the environment do
        Observe state $x$;   Take action $a \sim \pi(\cdot|x;\theta_\pi)$;   Observe $r = r(x,a) \ \wedge \ x' \sim p(\cdot|x,a)$;
        Update the buffer $\mathcal{D} \leftarrow \mathcal{D} \cup (x,a,r,x')$;   Replace $x \leftarrow x'$;
    end for
    # E-step   ($K$ is the number of E-step iterations)
    for $k = 1, \dots, K$ do
        # Step 1   (critic update – updating $V_{q_c}$ and $Q_{q_c}$)
        Sample a number of $(x,a,r,x')$ from $\mathcal{D}$;            # $a \sim q_c(\cdot|x),\ x' \sim p(\cdot|x,a)$
        Update $Q$ parameter $\theta_q$ using gradient of (39);
        Sample a number of $(x,a)$ from the $\mathcal{D}$;            # $a \sim q_c(\cdot|x)$
        Update $V$ parameter $\theta_v$ using gradient of (40);
        # Step 2   (actor update – updating $q_c$ – policy improvement)
        Update $q_c$ parameter $\theta_c$ either using gradient of (25) or by solving (15) in closed-form;
        # target networks $\theta'_v, \theta'_q$ are set to an exponentially moving average of the value networks $\theta_v, \theta_q$
        $\theta'_v \leftarrow \tau\theta_v + (1-\tau)\theta'_v$;   $\theta'_q \leftarrow \tau\theta_q + (1-\tau)\theta'_q$;
    end for
    # M-step   (updating the baseline policy $\pi$)
    Update baseline policy $\pi$ parameter $\theta_\pi$ either by setting it to $\theta_c$
end for
```

D.1 The Model-free E-step Critic Update

Suppose we have access to the policy π, and posterior policy q_c, we now aim to learn the corresponding value functions (critic) V_{π,q_c} and Q_{π,q_c}. Recall from Lemma 1, we know that V_{π,q_c} is a unique solution of fixed-point equation $\mathcal{T}_{q_c}[V](x) = V(x), \forall x \in \mathcal{X}$. Suppose we parameterize $V_{\pi,q_c}(x)$ with function approximation $\hat{V}_{\pi,q_c}(x;\theta_v)$, and similarly $Q_{\pi,q_c}(x,a)$ with $\hat{Q}_{\pi,q_c}(x,a;\theta_q)$. Similar to soft DQN, one way to learn $\hat{V}_{\pi,q_c}(x;\theta_v)$ and $\hat{Q}_{\pi,q_c}(x,a;\theta_q)$ is by minimizing the following objective function respectively, over the data from the replay buffer $\mathcal{D}$ sampled from the environment:

$$\theta_q^* \leftarrow \arg\min_\theta \sum_{(x,a,r,x')\sim\mathcal{D}} \left(\exp\left(\hat{Q}_{\pi,q_c}(x,a;\theta) - \eta r - \hat{V}_{\pi,q_c}(x';\theta'_v)\right) - 1\right)^2, \tag{39}$$

where $V_{\pi,q_c}(x';\theta'_v)$ is a *target* Q-function, and

$$\theta_v^* \leftarrow \arg\min_\theta \sum_{(x,a,r,x')\sim\mathcal{D}} \left(\hat{V}_{\pi,q_c}(x;\theta) - \int_{a\in\mathcal{A}} q_c(a|x)\left(\hat{Q}_{\pi,q_c}(x,a;\theta_q) - \log\frac{q_c(a|x)}{\pi(a|x)}\right)\right)^2, \tag{40}$$

The above learning is completely *off-policy*—the target is valid no matter how the experience was generated (as long as it is sufficiently exploratory). Under this loss, critic learning can be viewed as ℓ_2-regression of $\exp(\hat{Q}_{\pi,q_c}(x,a;\theta_q) - \eta r - V_{\pi,q_c}(x;\theta_v))$ w.r.t. the target label 1, such that the value function $V_{\pi,q_c}(x;\theta_v)$ is learned to minimize the the *mean squared Bellman error*: $(\mathcal{T}_{q_c}[V](x) - V(x))^2$ and to enforce $\hat{Q}_{\pi,q_c}(x,a;\theta_q) = \eta \cdot r(x,a) + \log\int_{x'\in\mathcal{X}} P(x'|x,a)\exp\hat{V}_{\pi,q_c}(x';\theta_v)$.

E Experimental Details

Environment	State dimension	Action dimension
Pendulum	3	1
Reacher	11	2
Hopper	11	3
Reacher7DoF	14	7
Walker2D	17	6
HalfCheetah	17	6

Table 3: State and Action dimensions of various benchmark environments.

Hyper Parameters for MBPO and VMBPO	Value(s)
Discount Factor	0.99
Number of Model Ensemble Networks	7
Number of Expert Networks	2
Number of Q Ensemble Networks	2
Dynamics Model Network Architecture	MLP with 4 hidden layers of size 200
Critic Network Architecture	MLP with 2 hidden layers of size 200
Actor Network Architecture	MLP with 2 hidden layers of size 200
Exploration policy	$\mathcal{N}(0, \sigma = 1)$
Exploration noise (σ) decay	0.999
Exploration noise (σ) minimum	0.025
Temperature	0.99995
Soft target update rate (τ)	0.005
Replay memory size (Both $\mathcal{D}$, $\mathcal{E}$)	10^6
Mini-batch size (AC)	64
Mini-batch size (Model-learning)	256
Model learning rate	0.0003
Critic learning rates	0.001, 0.0005, 0.0002
Actor learning rates	0.0005, 0.0002, 0.0001
Neural network optimizer	Adam

Table 4: Hyper parameters settings for MBPO and VMBPO. We sweep over the critic learning rates and actor learning rates for tuning.

For baseline algorithms, we either use the code the authors open-sourced or implement on our own and deliberately use the configurations shown in the literature. Among them, Table 4 shows the hyper parameters for MBPO and VMBPO in more detail.

E.1 Additional Experimental Results

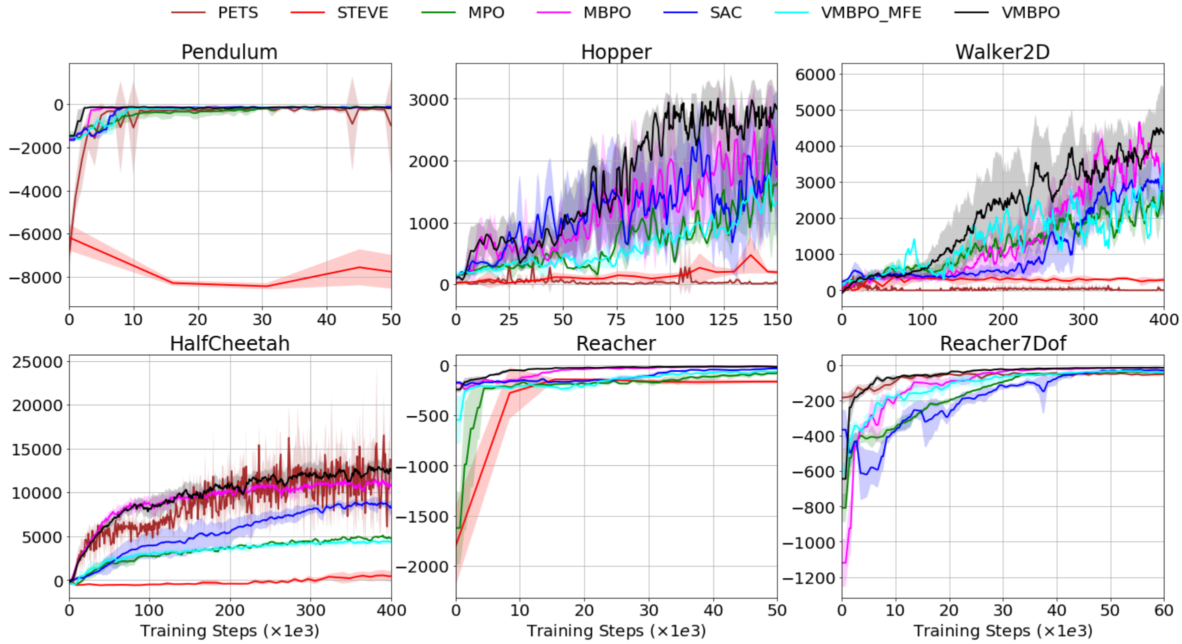

Figure 2: Mean cumulative reward of the best hyper parameter configuration over 5 random seeds.

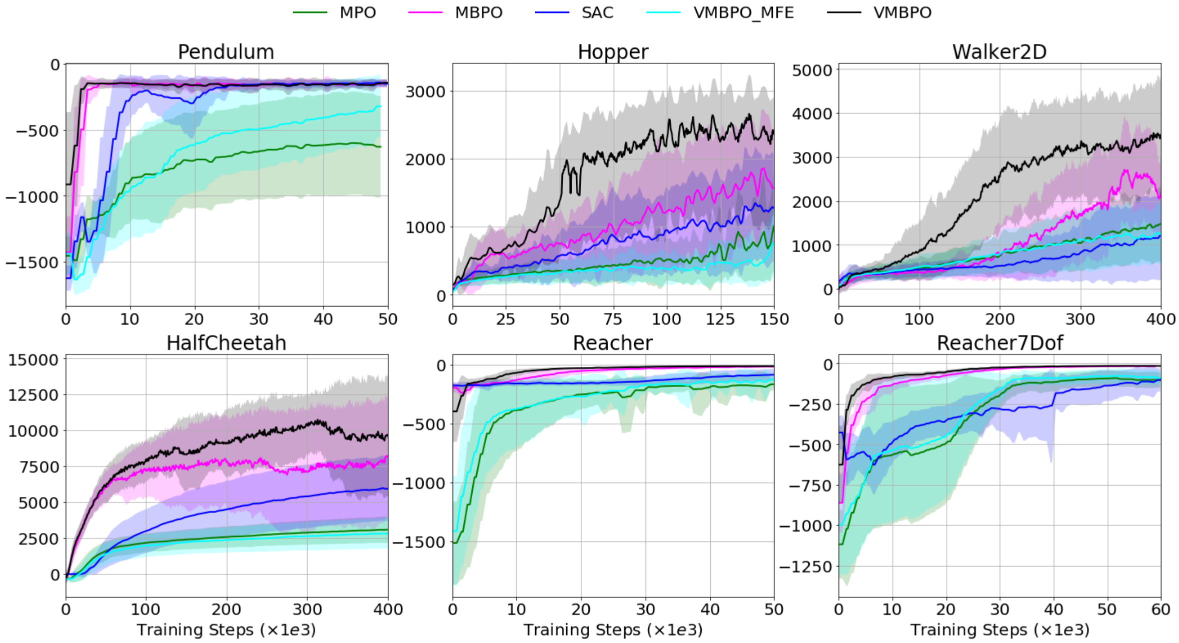

Figure 3: Mean cumulative reward over all hyper-parameter and random-seed configurations. We do not include PETS and STEVE because the hyper-parameters are adopted from their papers.