Deep Model-Based Reinforcement Learning via Estimated Uncertainty and Conservative Policy Optimization

Qi Zhou, HouQiang Li, Jie Wang*
University of Science and Technology of China
zhouqida@mail.ustc.edu.cn
{lihq, jiewangx}@ustc.edu.cn

Abstract

Model-based reinforcement learning algorithms tend to achieve higher sample efficiency than model-free methods. However, due to the inevitable errors of learned models, model-based methods struggle to achieve the same asymptotic performance as model-free methods. In this paper, We propose a **P**olicy **O**ptimization method with **M**odel-**B**ased **U**ncertainty (POMBU)—a novel model-based approach—that can effectively improve the asymptotic performance using the uncertainty in Q-values. We derive an upper bound of the uncertainty, based on which we can approximate the uncertainty accurately and efficiently for model-based methods. We further propose an uncertainty-aware policy optimization algorithm that optimizes the policy conservatively to encourage performance improvement with high probability. This can significantly alleviate the overfitting of policy to inaccurate models. Experiments show POMBU can outperform existing state-of-the-art policy optimization algorithms in terms of sample efficiency and asymptotic performance. Moreover, the experiments demonstrate the excellent robustness of POMBU compared to previous model-based approaches.

1 Introduction

Model-free reinforcement learning has achieved remarkable success in sequential decision tasks, such as playing Atari games (Mnih et al. 2015; Hessel et al. 2018) and controlling robots in simulation environments (Lillicrap et al. 2016; Haarnoja et al. 2018). However, model-free approaches require large amounts of samples, especially when using powerful function approximators, like neural networks. Therefore, the high sample complexity hinders the application of model-free methods in real-world tasks, not to mention data gathering is often costly. In contrast, model-based reinforcement learning is more sample efficient, as it can learn from the interactions with models and then find a near-optimal policy via models (Kocijan et al. ; Deisenroth and Rasmussen 2011; Levine and Abbeel 2014; Nagabandi et al. 2018). However, these methods suffer from errors of learned models, which hurt the asymptotic performance (Schneider 1997; Abbeel, Quigley, and Ng 2006). Thus, compared to model-free methods, model-based algorithms can learn more quickly but tend to learn suboptimal policies after plenty of trials.

Early model-based methods achieve impressing results using simple models, like linear models (Bagnell and Schneider 2001; Levine et al. 2016) and Gaussian processes (Kuss and Rasmussen 2004; Deisenroth and Rasmussen 2011). However, these methods have difficulties in high-dimensional and non-linear environments due to the limited expressiveness of models. Recent methods use neural network models for better performance, especially for complicate tasks (Punjani and Abbeel 2015; Nagabandi et al. 2018). Some methods further characterize the uncertainty in models via neural network ensembles (Rajeswaran et al. 2017; Kurutach et al. 2018), or Bayesian neural networks (Depeweg et al. 2017). Although the uncertainty in models improves the performance of model-based methods, recent research shows that these methods still struggle to achieve the comparable asymptotic performance to state-of-the-art model-free methods robustly (Wang et al. 2019).

Inspired by previous work that improves model-free algorithms via uncertainty-aware exploration (O'Donoghue et al. 2018), we propose a theoretically-motivated algorithm to estimate the uncertainty in Q-values and apply it to the exploration of model-based reinforcement learning. Moreover, we propose to optimize the policy conservatively by encouraging a large probability of performance improvement, which is also informed by the estimated uncertainty. Thus, we use the uncertainty in Q-values to enhance both exploration and policy optimization in our model-based algorithm.

Our contributions consist of three parts.

First, we derive an upper bound of the uncertainty in Q-values and present an algorithm to estimate it. Our bound is tighter than previous work (O'Donoghue et al. 2018), and our algorithm is feasible for deep model-based reinforcement learning, while many previous methods only focus on model-free cases (Osband et al. 2016; 2017), or assume simple models (Dearden, Friedman, and Andre 1999).

Second, we propose to optimize the policy conservatively based on an estimated probability of performance improvement, which is estimated via the uncertainty in Q-values and is useful to prevent the overfitting to the biased models.

*Corresponding author.
Copyright © 2020, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Third, we propose a **P**olicy **O**ptimization method with **M**odel-**B**ased **U**ncertainty (POMBU), which combines our uncertainty estimation algorithm with the conservative policy optimization algorithm. Experiments show that POMBU achieves excellent robustness and can outperform state-of-the-art policy optimization algorithms.

2 Background

A finite-horizon Markov decision process (MDP) $\mathcal{M}$ is defined by the tuple $(\mathcal{S}, \mathcal{A}, P, R, \rho, H)$. Here, $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a finite set of actions, P is a third-order tensor that denotes the transition probabilities, R is a matrix that denotes the rewards, ρ denotes the distribution of initial states and H is the horizon length. More specifically, at the state s and selecting the action a, $P_{sas'} \in [0,1]$ is the probability of transitioning to the state s', and $R_{sa} \in [-R_{\max}, R_{\max}]$ is the obtained reward. We represent a posterior of MDPs as $(\Omega, \mathcal{F}, Pr)$, where Ω is the sample space containing all possible MDPs, $\mathcal{F}$ is a σ-field consisting of subsets of Ω, and $Pr : \mathcal{F} \to [0,1]$ measures the posterior probability of MDPs. We assume that each MDP in Ω is different from others only in terms of P and R. In this case, P is a random tensor and R is a random matrix. For any random variable, matrix or tensor X, $\mathbb{E}_{\mathcal{M}}[X]$ and $\mathbb{D}_{\mathcal{M}}[X]$ denotes its expectation and variance respectively. When without ambiguity, we write $\mathbb{E}_{\mathcal{M}}[X]$ as $\bar{X}$ for short, e.g., $\bar{P}$ denotes $\mathbb{E}_{\mathcal{M}}[P]$ and $\bar{P}_{sas'}$ denotes $\mathbb{E}_{\mathcal{M}}[P_{sas'}]$.

Let π denotes a policy. π_{sa} denotes the probability of taking the action a at the state s. Considering the posterior of MDPs, the expected return J_π is a random variable, which is defined by

$$J_\pi = \mathbb{E}_{\tau\sim(\mathcal{M},\pi)}\left[\sum_{h=1}^{H} R_{s^h a^h}\right].$$

Here $\tau = (s^1, a^1, \ldots, s^H, a^H)$ is a trajectory. $\tau \sim (\mathcal{M}, \pi)$ means that the trajectory is sampled from the MDP $\mathcal{M}$ under policy π. That is, s^1 is sampled from the initial state distribution ρ of $\mathcal{M}$, a^h is sampled with the probability $\pi_{s^h a^h}$ and s^{h+1} is sampled with the probability $P_{s^h a^h s^{h+1}}$ in $\mathcal{M}$. Our goal is to find a policy maximizing J^π in real environment.

Given an MDP $\mathcal{M}$, we define the corresponding state-action value function Q, the state value function V and the advantage function A as follow:

$$V_\pi^h(s) = \mathbb{E}_{\tau\sim(\mathcal{M},\pi)}\left[\sum_{l=h}^{H} R_{s^l a^l} \,\middle|\, s^h = s\right],$$

$$Q_\pi^h(s,a) = \mathbb{E}_{\tau\sim(\mathcal{M},\pi)}\left[\sum_{l=h}^{H} R_{s^l a^l} \,\middle|\, s^h = s, a^h = a\right],$$

$$A_\pi^h(s,a) = Q_\pi^h(s,a) - V_\pi^h(s).$$

When the policy π is fixed, we write $V_\pi^h(s)$, $Q_\pi^h(s,a)$ and $A_\pi^h(s,a)$ as V_s^h, Q_{sa}^h and A_{sa}^h respectively for short. In this case, for any time-step h, V_s^h, Q_{sa}^h and A_{sa}^h are random variables mapping Ω to $\mathbb{R}$. Hence, V^h is a random vector. Q^h and A^h are random matrices.

3 Uncertainty Estimation

In this section, we consider a fixed policy π. Similarly to the uncertainty Bellman equation (UBE) (O'Donoghue et al. 2018), we regard the standard deviations of Q-values as the uncertainty. In this section, we derive an upper bound of $\mathbb{D}_{\mathcal{M}}\left[Q_{sa}^h\right]$ for each s, a, h, and prove that our upper bound is tighter than that of UBE. Moreover, we propose an uncertainty estimation algorithm for deep model-based reinforcement learning and discuss its advantages. We provide related proofs in Appendix A.1-A.4.

Upper Bound of Uncertainty in Q-values

To analyze the uncertainty, we first make two assumptions.

Assumption 1 *Each MDP in Ω is a directed acyclic graph.*

This assumption is common (Osband, Van Roy, and Wen ; O'Donoghue et al. 2018). It means that the agent cannot visit a state more than twice within the same episode. This assumption is weak because each finite horizon MDP violating the assumption can be converted into a similar MDP that satisfying the assumption (O'Donoghue et al. 2018).

Assumption 2 *The random vector R_{s_1} and the random matrix P_{s_1} are independent on R_{s_2} and P_{s_2} if $s_1 \neq s_2$.*

This assumption is used in the derivation of UBE (O'Donoghue et al. 2018). It is consistent with the trajectory sampling strategies used in recent model-based algorithms (Chua et al. 2018; Kurutach et al. 2018), which sample a model from the ensemble of models independently per time step to predict the next state and reward.

First, we derive an inequation from these assumptions.

Lemma 1 *Under Assumption 1 and 2, for any $s \in \mathcal{S}$ and $a \in \mathcal{A}$, we have*

$$\mathbb{D}_{\mathcal{M}}\left[Q_{sa}^h\right] \leq u_{sa}^h + \sum_{s',a'} \pi_{s'a'} \bar{P}_{sas'} \mathbb{D}_{\mathcal{M}}\left[Q_{s'a'}^{h+1}\right],$$

$$\textit{where } u_{sa}^h = \mathbb{D}_{\mathcal{M}}\left[R_{sa} + \sum_{s'} P_{sas'} \bar{V}_{s'}^{h+1}\right].$$

We consider u_{sa}^h as a local uncertainty, because we can compute it locally with $\bar{V}$.

Then, we can derive our main theorem from this lemma.

Theorem 1 *Under Assumption 1 and 2, for any policy π, there exists a unique solution U satisfying the following equation:*

$$U_{sa}^h = u_{sa}^h + \sum_{s',a'} \pi_{s'a'} \bar{P}_{sas'} U_{s'a'}^{h+1} \tag{1}$$

for any (s,a) and $h = 1, 2, \ldots, H$, where $U^{H+1} = \mathbf{0}$, and furthermore $U \geq \mathbb{D}_{\mathcal{M}}[Q]$ pointwise.

Theorem 1 means that we can compute an upper bound of $\mathbb{D}_{\mathcal{M}}\left[Q_{sa}^h\right]$ by solving the Bellman-style equation (1).

Moreover, we provide the following theorem to show the convergence when computing U iteratively.

Theorem 2 *For arbitrary* $(U)_1$*, if*

$$\left(U_{sa}^h\right)_{i+1} = \left(u_{sa}^h\right)_i + \sum_{s',a'} \pi_{s'a'} \bar{P}_{sas'} \left(U_{s'a'}^{h+1}\right)_i,$$

for any (s,a)*,* $h = 1,2,\ldots,H$ *and* $i \geq 1$*, where* $\left(U^{H+1}\right)_i = \mathbf{0}$ *and* $(u)_i$ *converges to* u *pointwise, we have* $(U)_i$ *converges to* U *pointwise.*

Theorem 2 shows that we can solve the equation (1) iteratively if the estimated local uncertainty is not accurate per update but converges to the correct value, whis is significant when we use an estimated $\bar{V}$ to compute the uncertainty.

As U_{sa}^h is an upper bound of $\mathbb{D}_{\mathcal{M}}\left[Q_{sa}^h\right]$, $\sqrt{U_{sa}^h}$ is an upper bound of the uncertainty in Q_{sa}^h. We use the upper bound to approximate the uncertainty in our algorithm similarly to UBE. We need to analyze the accuracy of our estimates.

Here, we compare our upper bound U with that of UBE under the same assumptions, and hence we need to make an extra assumption used in UBE.

Assumption 3 R_s *is independent on* P_s *for any* $s \in \mathcal{S}$*.*

This assumption is not used to derive our upper bound of the uncertainty but is used in UBE. Under the assumption 2 and 3, we have R is independent on P.

The upper bound B derived in UBE satisfies

$$B_{sa}^h = \nu_{sa}^h + \sum_{s',a'} \pi_{s'a'} \bar{P}_{sas'} B_{s'a'}^{h+1},$$

where $\nu_{sa}^h = (Q_{\max})^2 \sum_{s'} \mathbb{D}_{\mathcal{M}}\left[P_{sas'}\right] / \bar{P}_{sas'} + \mathbb{D}_{\mathcal{M}}\left[\mu_{sa}\right]$.

Here, $Q_{\max}$ is an upper bound of all $|Q_{sa}^h|$ for any s, a, h and MDP. For example, we can regard $H\bar{R}_{\max}$ as $Q_{\max}$.

Theorem 3 *Under the assumption 1, 2 and 3,* U_{sa}^h *is a tighter upper bound of* $\mathbb{D}_{\mathcal{M}}\left[Q_{sa}^h\right]$ *than* B_{sa}^h*.*

This theorem means that approximating the uncertainty using our upper bound U is more accurate than using the upper bound B derived in UBE.

Uncertainty Estimation Algorithm

First, we characterizes the posterior of MDPs approximatively using a deterministic model ensemble (please refer to the Section 5 for the details of training models). A deterministic ensemble is denoted by $(f_{w_1}, f_{w_2}, \ldots, f_{w_K})$. Here, for any $i = 1,2,\ldots,K$, $f_{w_i} : \mathcal{S} \times \mathcal{A} \to \mathcal{S} \times \mathbb{R}$ is a single model that predicts the next state and reward given a state and an action, and w_i is its parameters. We define the posterior probability of MDPs by

$$Pr\{P_{sas'} = 1, R_{sa} = x\} = \frac{1}{K} \sum_{i=1}^{K} \mathbf{eq}\left((s',x), f_{w_i}(s,a)\right),$$

where **equal** is defined by

$$\mathbf{eq}\left((s_1,x_1),(s_2,x_2)\right) = \begin{cases} 1, & \text{if } s_1 = s_2 \text{ and } x_1 = x_2, \\ 0, & \text{otherwise.} \end{cases}$$

Then, we can construct an MDP $\hat{\mathcal{M}}$ defined according to the posterior of MDPs, such that its transition tensor $\hat{P}$ is equal to $\bar{P}$ and its reward matrix $\hat{R}$ is equal to $\bar{R}$. Hence, the state value matrix $\hat{V}$ of the MDP $\hat{\mathcal{M}}$ is equal to $\bar{V}$.

Algorithm 1: Uncertainty Estimation for Q-values

Input : A approximate value function $\tilde{V}_\phi$; An ensemble model $\{f_{w_1}, f_{w_2}, \ldots, f_{w_K}\}$; A trajectory $\tau = (s_1, a_1, \ldots, s_H, a_H)$;

Output: Estimates of $(\sqrt{U_{s_1a_1}^1}, \ldots, \sqrt{U_{s_Ha_H}^H})$;

```
D^{H+1} = 0;
for i = H, H-1, ..., 1 do
    for j = 1, 2, ..., K do
        (s_j, r_j) = f_{w_j}(s^i, a^i);
        q_j = r_j + Ṽ_φ(s_j);
    end
    q = (1/K) Σ_{j=1}^K q_j;
    d^i = (1/K) Σ_{j=1}^K (q_j - q)^2;
    D^i = d^i + D^{i+1};
end
return (√D^1, √D^2, ..., √D^H);
```

Moreover, we use a neural network $\tilde{V}_\phi : \mathcal{S} \to \mathbb{R}$ to predict $\hat{V}_s^h$ for any state s and time step h, which is equivalent to predicting $\bar{V}$. We train $\tilde{V}_\phi$ by minimizing ℓ_2 loss function

$$L_v(\phi) = \mathbb{E}_{\tau\sim(\hat{\mathcal{M}},\pi)}\left[\frac{1}{H}\sum_{h=1}^{H}\left\|\tilde{V}_\phi(s^h) - \sum_{l=h}^{H}\hat{R}_{s^l a^l}\right\|_2^2\right]. \tag{2}$$

Finally, given an imagined trajectory τ sampled from $\hat{\mathcal{M}}$ under π, we can estimate the uncertainty in Q-values via the algorithm 1. Note that for long-horizon tasks, we can introduce a discount factor γ similarly to previous work (O'Donoghue et al. 2018). The modified uncertainty estimation method can be found in Appendix B.

Discussion

In this part, we discuss some advantages of our algorithm to estimate the uncertainty in Q-values.

Accuracy Based on the Theorem 3, our upper bound of the uncertainty is tighter than UBE, which means a more accurate estimation. Intuitively, our local uncertainty depends on $\bar{V}^h$ while that of UBE depends on $Q_{\max}$. Therefore, our local uncertainty has a weaker dependence on H and can provide a relatively accurate estimation for long-horizon tasks (see an example in Appendix C). Moreover, considering an infinite set of states, our method ensures the boundedness of the local uncertainty because $\bar{V}^h$ and R are bounded. Therefore, our method has the potential to apply to tasks with continuous action spaces.

Applicability for Model-Based Methods Our method to estimate the uncertainty in Q-values is effective for model-based reinforcement learning. In model-based cases, estimated Q-values are highly dependent on the models. Our

method considers the model when computing the local uncertainty, while most of the existing methods estimate the uncertainty directly via the real-world samples regardless of the models. Ignoring models may lead to bad estimates of uncertainty in model-based cases. For example, the uncertainty estimated by a count-based method (Bellemare et al. 2016; Ostrovski et al. 2017) tends to decrease with the increase of the number of samples, while the true uncertainty keeps high even with a large amount of samples when modeling a complicate MDP using a simple model.

Computational Cost Our method is much more computationally cheap compared with estimating the uncertainty via the empirical standard deviation of Q^h_{sa}. Estimating Q^h_{sa} given an MDP requires plenty of virtual samples, let alone estimating its empirical standard deviation. Previous work reduces the computational cost by learning an ensemble of Q functions (Buckman et al. 2018). However, training an ensemble of Q functions requires higher computational overhead than training a single neural network $\tilde{V}_\phi$ in our method.

Compatibility with Neural Networks Previous methods that estimate uncertainty for model-based methods always assume simple models, like Gaussian processes (Deisenroth and Rasmussen 2011; Dearden, Friedman, and Andre 1999). Estimating uncertainty using Theorem 1 only requires that the models can represent a posterior. This makes our method compatible with neural network ensembles and Bayesian neural networks. For instance, we propose Algorithm 1 with an ensemble of neural networks.

Propagation of Uncertainty As discussed in previous work (Osband, Aslanides, and Cassirer 2018), Bellman equation implies the high dependency between Q-values. Ignoring this dependence will limit the accuracy of the estimates of uncertainty. Our method considers the dependency and propagates the uncertainty via a Bellman-style equation.

4 Conservative Policy Optimization

In this section, we first introduce surrogate objective and then modify it via uncertainty. The modified objective leads to conservative policy optimization because it penalizes the update in the high-uncertainty regions. Let $\pi_\theta : \mathcal{S} \times \mathcal{A} \to [0, 1]$ denotes a parameterized policy, and θ is its parameters. $\pi_\theta(a|s)$ denotes the probability of taking action a at state s.

Surrogate Objective

Recent reinforcement learning algorithms, like Trust Region Policy Optimization (TRPO) (Schulman et al. 2015), Proximal Policy Optimization (PPO) (Schulman et al. 2017), optimize the policy based on surrogate objective. We rewrite the surrogate objective in TRPO and PPO as follow:

$$L_{sr}(\theta) = \mathbb{E}_{\tau\sim(\mathcal{M},\pi_{\theta_{\text{old}}})}\left[\sum_{h=1}^{H} r_\theta(s^h, a^h) A^h_{\text{old}}(s^h, a^h)\right],$$

where θ_{old} are the old policy parameters before the update, A_{old} is the advantage function of $\pi_{\theta_{\text{old}}}$ and

$$r_\theta(s, a) = \frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}.$$

Previous work has proven the surrogate objective is the first order approximation to $J(\pi_\theta) - J(\pi_{\theta_{\text{old}}})$ when θ is around θ_{old} (Schulman et al. 2015; Kakade and Langford 2002). That is, for any θ_{old}, we have the following theorem:

Theorem 4

$$L_{sr}(\theta)|_{\theta=\theta_{old}} = J(\pi_\theta) - J(\pi_{\theta_{old}})|_{\theta=\theta_{old}},$$
$$\nabla_\theta L_{sr}(\theta)|_{\theta=\theta_{old}} = \nabla_\theta \left(J(\pi_\theta) - J(\pi_{\theta_{old}})\right)|_{\theta=\theta_{old}}$$

(see proof in Appendix A.5). Therefore, maximizing $L_{sr}(\theta)$ can maximize $J(\pi_\theta)$ approximately when θ is around θ_{old}.

Uncertainty-Aware Surrogate Objective

To prevent the overfitting of the policy to inaccurate models, we introduce the estimated uncertainty in Q-values into the surrogate objective.

First, we need to estimate $Pr\{J(\pi_\theta) > J(\pi_{\theta_{\text{old}}})\}$, which means the probability that the new policy outperforms the old one. Because of Theorem 4, $Pr\{L_{sr}(\theta) > 0\}$ can approximate $Pr\{J(\pi_\theta) > J(\pi_{\theta_{\text{old}}})\}$. Then, we assume that a Gaussian can approximate the distribution of $L_{sr}(\theta) > 0$. Thus, $F\left(\mathbb{E}_\mathcal{M}\left[L_{sr}(\theta)\right] / \sqrt{\mathbb{D}_\mathcal{M}\left[L_{sr}(\theta)\right]}\right)$ is approximately equal to $Pr\{L_{sr}(\theta) > 0\}$, where F is the probability distribution function of standard normal distribution.

Then, we need to construct an objective function for optimization. Here, we aims to find a new θ with a large $F\left(\mathbb{E}_\mathcal{M}\left[L_{sr}(\theta)\right] / \sqrt{\mathbb{D}_\mathcal{M}\left[L_{sr}(\theta)\right]}\right)$. As F is monotonically increasing, we can maximize $\mathbb{E}_\mathcal{M}\left[L_{sr}(\theta)\right]$ while minimize $\sqrt{\mathbb{D}_\mathcal{M}\left[L_{sr}(\theta)\right]}$. Therefore, we can maximize

$$\mathbb{E}_\mathcal{M}\left[L_{sr}(\theta)\right] - \alpha\sqrt{\mathbb{D}_\mathcal{M}\left[L_{sr}(\theta)\right]}, \tag{3}$$

where $\alpha \geq 0$ is a hyperparameter.

Moreover, we need to estimate the expectation and the variance of the surrogate objective. Because $L_{sr}(\theta)$ is equal to

$$\mathbb{E}_{\tau\sim(\mathcal{M},\pi_{\theta_{\text{old}}})}\left[\sum_{h=1}^{H}\left(r_\theta(s^h, a^h) - 1\right) Q^h_{\text{old}}(s^h, a^h)\right],$$

we can approximate $\mathbb{E}_\mathcal{M}\left[L_{sr}(\theta)\right]$ and $\sqrt{\mathbb{D}_\mathcal{M}\left[L_{sr}(\theta)\right]}$ as $L_{\text{exp}}(\theta)$ and $L_{\text{std}}(\theta)$ respectively, where

$$L_{\text{exp}}(\theta) = \mathbb{E}_{\tau\sim(\hat{\mathcal{M}},\pi_{\theta_{\text{old}}})}\left[\sum_{h=1}^{H} r_\theta(s^h, a^h)\bar{A}^h_{\text{old}}(s^h, a^h)\right], \tag{4}$$

$$L_{\text{std}}(\theta) = \mathbb{E}_{\tau\sim(\hat{\mathcal{M}},\pi_{\theta_{\text{old}}})}\left[\sum_{h=1}^{H}\left|r_\theta(s^h, a^h) - 1\right|\sqrt{D^h}\right]. \tag{5}$$

Here $\hat{\mathcal{M}}$ is defined in Section 3 using a learned ensemble, $\bar{A}^h_{\text{old}}(s^h, a^h)$ can be approximated by $\sum_{l=h}^{H}\hat{R}_{s^l a^l} - \tilde{V}_\phi(s^h)$, and D^h is computed by Algorithm 1.

However, policy optimization without trust region may lead to unacceptable bad performance (Schulman et al.

Algorithm 2: POMBU

Initialize an ensemble $\{\tilde{f}_{w_1} \dots \tilde{f}_{w_K}\}$ and a policy π_θ;
Initialize a value function $\tilde{V}_\phi$;
Initialize the dataset S as a empty set;
Sample N trajectories using π_θ;
Add the sampled transitions to S;
repeat
  Train the ensemble $\{\tilde{f}_{w_1} \dots \tilde{f}_{w_K}\}$ using S;
  for $i = 1, 2, \dots, M$ **do**
    Sample virtual trajectories from $\hat{\mathcal{M}}$ using π_θ;
    Train $\tilde{V}_\phi$ by minimizeing $L_v(\phi)$;
    Train π_θ by maximizing $L_\pi(\theta)$;
  end
  for $i = 1, 2, \dots, N$ **do**
    Sample virtual trajectories from $\hat{\mathcal{M}}$ using π_θ;
    Train an exploration policy π_{exp} based on π_θ;
    Collect real-world trajectories using π_{exp};
    Add the sampled transitions to S;
  end
until π_θ performs well in the real environment;

2015). Thus, we clip $L_{\text{exp}}(\theta)$ similarly to PPO. That is,

$$L_{\text{clip}}(\theta) = \mathbb{E}_{\tau\sim(\hat{\mathcal{M}},\pi_{\theta_{\text{old}}})}\left[\sum_{h=1}^{H} \hat{r}_\theta(s^h, a^h)\bar{A}^h_{\text{old}}(s^h, a^h)\right]. \tag{6}$$

Here, we define $\hat{r}_\theta(s^h, a^h)$ as

$$\begin{cases} \max(1-\epsilon, r_\theta(s^h, a^h)), & \text{if } \bar{A}^h_{\text{old}}(s^h, a^h) < 0, \\ \min(1+\epsilon, r_\theta(s^h, a^h)), & \text{if } \bar{A}^h_{\text{old}}(s^h, a^h) > 0, \end{cases}$$

in which $\epsilon > 0$ is a hyperparameter.

Finally, we obtain the modified surrogate objective

$$L_\pi(\theta) = L_{\text{clip}}(\theta) - \alpha L_{\text{std}}(\theta).$$

Note that, the main difference of our objective from PPO is the uncertainty penalty $L_{\text{std}}(\theta)$. This penalty limits the ratio changes $|r_\theta(s^h, a^h) - 1|$ in high-uncertainty regions. Therefore, this objective is uncertainty-aware and leads to a conservative update.

5 Algorithm

In this section, we propose a **P**olicy **O**ptimization method with **M**odel-**B**ased **U**ncertainty (POMBU) in Algorithm 2. We details each stage of our algorithm as following.

Exploration Policy We train a set of exploration policies by maximizing the $L_{\text{clip}}(\theta)$. Different policies are trained with different virtual trajectories. To explore the unknown, we replace $\bar{A}^h_{\text{old}}(s^h, a^h)$ with $\bar{A}^h_{\text{old}}(s^h, a^h) + \beta\sqrt{D^h}$ in the equation (6). Here, $\beta \geq 0$ controlling the exploration to high-uncertainty regions.

Model Ensemble To predict the next state, a single neural network in the ensemble outputs the change in state and then adds the change to the current state (Kurutach et al. 2018; Nagabandi et al. 2018). To predict the reward, we assume the reward in real environment is computed by a function μ such that $R_{s^h a^h} = \mu(s^h, a^h, s^{h+1})$, which is commonly true in many simulation control tasks. Then, we can predict the reward via the predicted next state. We train the model by minimizing ℓ_2 loss similarly to previous work (Kurutach et al. 2018; Nagabandi et al. 2018) and optimize the parameter using Adam (Kingma and Ba 2014). Different models are trained with different train-validation split.

Policy Optimization We use a Gaussian policy whose mean is computed by a forward neural network and standard deviation is represented by a vector of parameters. We optimizing all parameters by maximizing $L_\pi(\theta)$ via Adam.

6 Experiments

In this section, we fist evaluate our uncertainty estimation method. Second, we compare POMBU to state-of-the-arts. Then, we show how does the estimated uncertainty work by ablation study. Finally, we analyze the robustness of our method empirically. In the following experiments, we report the performance averaged over at least three random seeds. Please refer to Appendix D for the details of experiments. The source code and appendix of this work is available at https://github.com/MIRALab-USTC/RL-POMBU.

Effectiveness of Uncertainty Estimation

We evaluate the effectiveness of our uncertainty estimation method in two simple environments: 2D-point and 3D-point. These environments have continuous state spaces and continuous action spaces. First, we train an ensemble model of the environment and sample state-action pairs from the model using a deterministic policy. Then, we estimate the Q-values of these pairs via the means of virtual returns (computed using the models), and estimate the uncertainty using the algorithm 1. Finally, we compute the real Q-values using the return in real world, compute the ratios of errors to the estimated uncertainties, and count the frequencies of these ratios to draw Figure 1. This figure shows the distribution of ratios is similar to a standard normal distribution after sufficient training of $\tilde{V}_\phi$, which demonstrates the accuracy of the estimated uncertainty.

Comparison to State-of-the-Arts

We compare POMBU with state-of-the-art policy optimization algorithms on four continuous control tasks in Mujoco (Todorov, Erez, and Tassa 2012): Swimmer, HalfCheetah, Ant, and Walker2d. Our method and our baselines optimize a stochastic policy to complete the tasks. Our baselines include: soft actor critic (SAC) (Haarnoja et al. 2018); proximal policy optimization (PPO); stochastic lower bounds optimization (SLBO) (Luo et al. 2018); model-ensemble trust region policy optimization (METRPO) (Kurutach et al. 2018). To show the benefits of using uncertainty in model-based reinforcement learning, we also compare POMBU to model-ensemble proximal policy optimization (MEPPO),

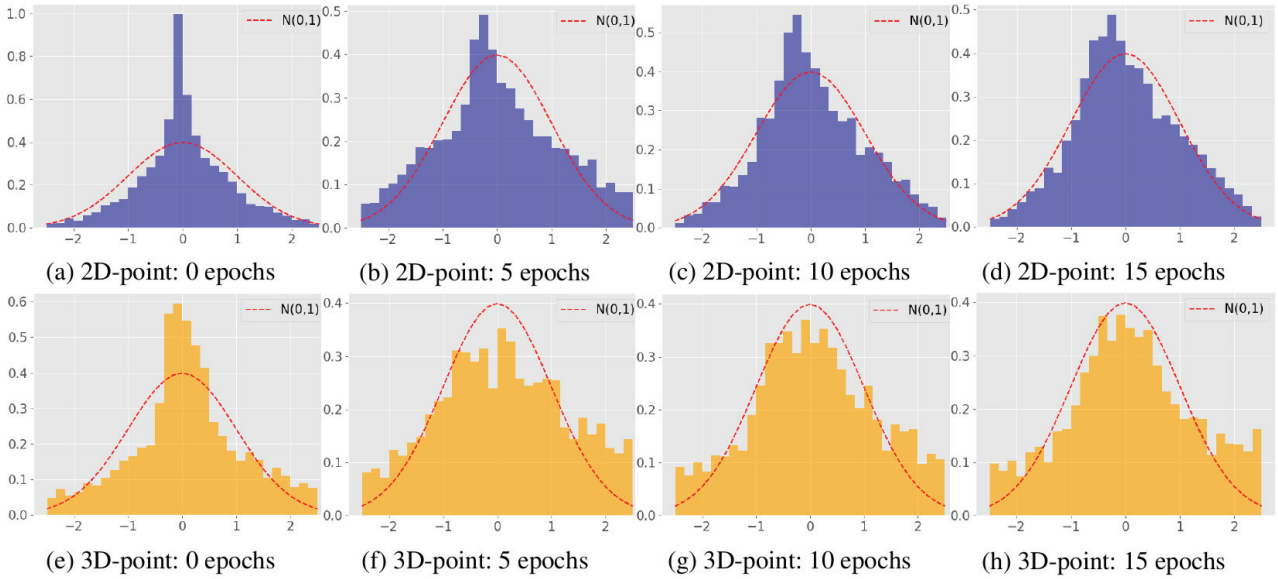


Figure 1: Frequency histograms of the ratios of errors to uncertainties after different numbers of epochs (training $\tilde{V}_\phi$). The red dotted line means the probability density function of the standard normal distribution.

which is equivalent to POMBU when $\alpha = 0$ and $\beta = 0$. We evaluate POMBU with $\alpha = 0.5$ and $\beta = 10$ for all tasks.

The result is shown in Figure 2. The solid curves correspond to the mean and the shaded region corresponds to the empirical standard deviation. It shows that POMBU achieves higher sample efficiency and better final performance than baselines, which highlights the great benefits of using uncertainty. Moreover, POMBU achieves comparable asymptotic performances with PPO and SAC in all tasks.

We also provide Table 1 that summarizes the performance, estimated wall-clock time and the number of used imagined samples and real-world samples in the HalfCheetah task (H=200). Compared to MEPPO, the extra time used in POMBU is small (time: $10.17 \rightarrow 12.05$), while the improvement is significant (mean: $449 \rightarrow 852$; standard deviation: $226 \rightarrow 21$). Compared to SAC, POMBU achieve higher performance with about 5 times less real-world samples. Moreover, in our experiments, the total time to compute the uncertainty (not include the time to train $\tilde{V}_\phi$) is about 1.4 minutes, which is ignorable compared with the overall time.

We further compare POMBU with state-of-the-art model-based algorithms in long-horizon tasks. The compared algorithms include model-based meta policy optimization (MBMPO) (Clavera et al. 2018), probabilistic ensemble with trajectory sampling (PETS) (Chua et al. 2018) and stochastic ensemble value expansion (STEVE) (Buckman et al. 2018) in addition. We directly use some of the results given by Wang et al. (2019), and summarize all results in Table 2. The table shows that POMBU achieves comparable performance with STEVE and PETS, and outperforms other model-based algorithms. It demonstrates that POMBU is also effective in long-horizon tasks.

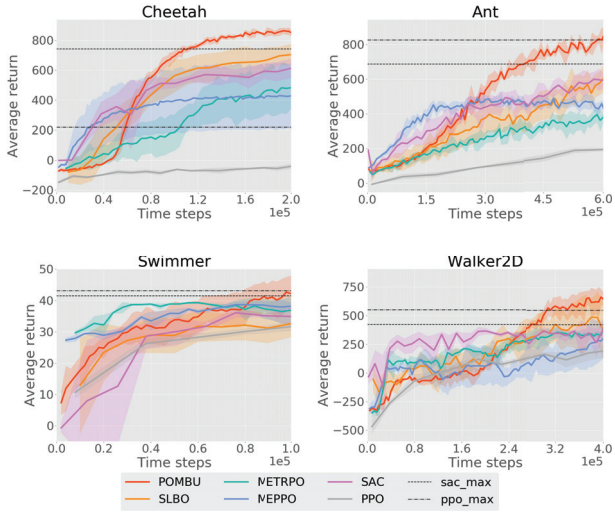


Figure 2: The training curve of our method and baselines. The horizons of all tasks are 200. The number of total steps is selected to ensure most model-based algorithms converge. We train the policy via PPO and SAC with at least 1 million samples and report the best averaged performance as "max".

Ablation Study

We provide an ablation study to show how the uncertainty benefits the performance. In our algorithm, we employ the uncertainty in policy optimization (controlled by α) and exploration (controlled by β). Therefore, we compare the performance with different α and β.

The results are shown in Figure 3 and 4. Setting α as 0.5 or 0.75 achieves the best final performance and the best robustness with 200K samples. Note that a large α may re-

	POMBU	MEPPO	METRPO	SLBO	SAC	PPO	SAC_max	PPO_max
Time (h)	12.05	10.17	6.35	3.91	0.87	0.04	4.18	0.19
Imagined	1.2e8	8e7	5e7	1e7	0	0	0	0
Real-world	2e5	2e5	2e5	2e5	2e5	2e5	9.89e5	9.78e5
Return	852 ± 21	449 ± 226	483 ± 136	704 ± 70	615 ± 64	-38 ± 16	741 ± 88	218 ± 63

Table 1: The performance, estimated wall-clock time and the number of used imagined samples and real-world samples in the HalfCheetah task (H=200). We conduct all experiments with one GPU Nvidia GTX 2080Ti.

Environment	POMBU	STEVE	MBMPO	SLBO	METRPO	PETS
Ant	$\mathbf{2010 \pm 91}$	552 ± 190	706 ± 147	728 ± 123	282 ± 18	1852 ± 141
HalfCheetah	3672 ± 8	$\mathbf{7965 \pm 1719}$	3639 ± 1186	1098 ± 166	2284 ± 900	2795 ± 880
Swimmer	$\mathbf{144.4 \pm 22.6}$	$\mathbf{149 \pm 81}$	85.0 ± 98.9	41.6 ± 18.4	30.1 ± 9.7	22.1 ± 25.2
Walker2d	-565 ± 129	-26 ± 328	-1546 ± 217	-1278 ± 428	-1609 ± 658	$\mathbf{260 \pm 537}$

Table 2: The performance of 200k time-step training. The horizons of all environments are 1000.

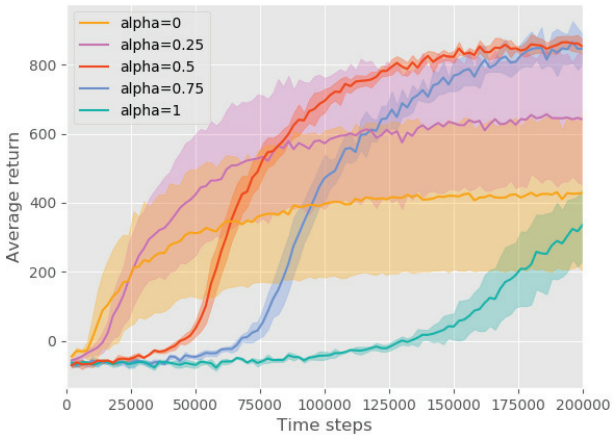


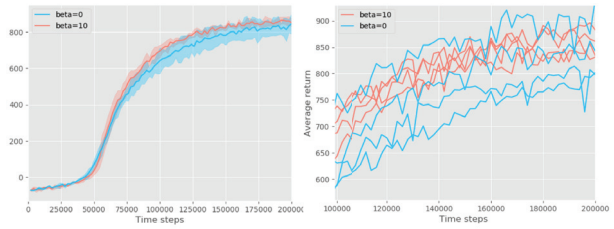
Figure 3: The development of the average return during training with different α in the Cheetah task (H=200).

sult in poorer performance in the early stage, because the uncertainty is high in the early stage and a large α tends to choose a small step size when uncertainty is high. Using $\beta = 10$ can improve the performance (larger mean and smaller standard deviation), which demonstrate the effectiveness of uncertainty-aware exploration.

Robustness Analyses

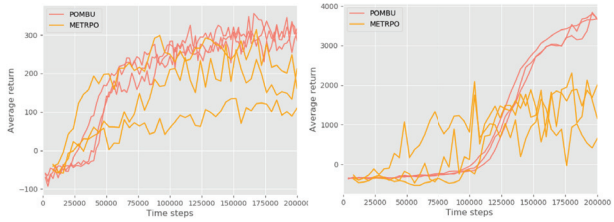
We demonstrate the excellent robustness of POMBU in two ways. First, we evaluate algorithms in noisy environments. In these environments, we add Gaussian noise to the observation with the standard deviation σ. This noise will affect the accuracy of the learned models. Second, we evaluate algorithms in long-horizon tasks. In these tasks, models need to generate long trajectories, and the error is further exacerbated due to the difficulty of longterm predictions.

We report the results in Figure 5. Experiments show that our algorithm achieves similar performance with different random seeds, while the performance of METRPO varies greatly with the random seeds. Moreover, in Figure 5, the worst performance of POMBU beats the best of METRPO.



(a) (b)

Figure 4: (a): The development of average return with $\beta = 10$ and $\beta = 0$. (b): The performance after 1e5 time-step training with different random seeds.



(a) (b)

Figure 5: The training curves of POMBU and METRPO with different random seeds. (a) Comparison in a noisy Cheetah task ($\sigma = 0.1$). (b) Comparison in a long-horizon Cheetah task ($H = 1000$).

This implies that our method has promising robustness, even in noisy environments and long-horizon environments.

7 Conclusion

In this work, we propose a Policy Optimization method with Model-Based Uncertainty (POMBU), which is a novel uncertainty-aware model-based algorithm. This method estimates uncertainty using a model ensemble and then optimizes policy Conservatively considering the uncertainty. Experiments demonstrate that POMBU can achieve comparable asymptotic performance with SAC and PPO while

using much fewer samples. Compared with other model-based methods, POMBU is robust and can achieve better performance. We believe that our approach will bring new insights into model-based reinforcement learning. An enticing direction for further work is the combination of our uncertainty estimation method with other kinds of models like Bayesian neural networks. Another exciting direction is to modify other advanced model-based algorithms like STEVE and PETS using our uncertainty estimation method.

Acknowledge

We would like to thank all the anonymous reviewers for their insightful comments. This work was supported in part by NSFC (61822604, 61836006, 61836011).

References

Abbeel, P.; Quigley, M.; and Ng, A. Y. 2006. Using inaccurate models in reinforcement learning. In *ICML*, 1–8.

Bagnell, J. A., and Schneider, J. G. 2001. Autonomous helicopter control using reinforcement learning policy search methods. In *ICRA*, volume 2, 1615–1620.

Bellemare, M.; Srinivasan, S.; Ostrovski, G.; Schaul, T.; Saxton, D.; and Munos, R. 2016. Unifying count-based exploration and intrinsic motivation. In *NIPS*, 1471–1479.

Buckman, J.; Hafner, D.; Tucker, G.; Brevdo, E.; and Lee, H. 2018. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *NIPS*, 8224–8234.

Chua, K.; Calandra, R.; McAllister, R.; and Levine, S. 2018. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *NIPS*, 4754–4765.

Clavera, I.; Rothfuss, J.; Schulman, J.; Fujita, Y.; Asfour, T.; and Abbeel, P. 2018. Model-based reinforcement learning via meta-policy optimization. *CoRL*.

Dearden, R.; Friedman, N.; and Andre, D. 1999. Model based bayesian exploration. In *UAI*, 150–159.

Deisenroth, M., and Rasmussen, C. E. 2011. Pilco: A model-based and data-efficient approach to policy search. In *ICML*, 465–472.

Depeweg, S.; Hernández-Lobato, J. M.; Doshi-Velez, F.; and Udluft, S. 2017. Learning and policy search in stochastic dynamical systems with bayesian neural networks. *ICLR*.

Haarnoja, T.; Zhou, A.; Abbeel, P.; and Levine, S. 2018. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *ICML*, volume 80, 1861–1870.

Hessel, M.; Modayil, J.; Van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M.; and Silver, D. 2018. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*.

Kakade, S., and Langford, J. 2002. Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, 267–274.

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

Kocijan, J.; Murray-Smith, R.; Rasmussen, C. E.; and Girard, A. Gaussian process model based predictive control. In *ACC*.

Kurutach, T.; Clavera, I.; Duan, Y.; Tamar, A.; and Abbeel, P. 2018. Model-ensemble trust-region policy optimization. *ICLR*.

Kuss, M., and Rasmussen, C. E. 2004. Gaussian processes in reinforcement learning. In *NIPS*, 751–758.

Levine, S., and Abbeel, P. 2014. Learning neural network policies with guided policy search under unknown dynamics. In *NIPS*, 1071–1079.

Levine, S.; Finn, C.; Darrell, T.; and Abbeel, P. 2016. End-to-end training of deep visuomotor policies. *JMLR* 17(1):1334–1373.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2016. Continuous control with deep reinforcement learning. *ICLR*.

Luo, Y.; Xu, H.; Li, Y.; Tian, Y.; Darrell, T.; and Ma, T. 2018. Algorithmic framework for model-based deep reinforcement learning with theoretical guarantees. *NIPS*.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M.; Fidjeland, A. K.; Ostrovski, G.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529.

Nagabandi, A.; Kahn, G.; Fearing, R. S.; and Levine, S. 2018. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *ICRA*, 7559–7566. IEEE.

O'Donoghue, B.; Osband, I.; Munos, R.; and Mnih, V. 2018. The uncertainty bellman equation and exploration. *ICML*.

Osband, I.; Aslanides, J.; and Cassirer, A. 2018. Randomized prior functions for deep reinforcement learning. In *NIPS*, 8617–8629.

Osband, I.; Blundell, C.; Pritzel, A.; and Van Roy, B. 2016. Deep exploration via bootstrapped dqn. In *NIPS*, 4026–4034.

Osband, I.; Van Roy, B.; Russo, D.; and Wen, Z. 2017. Deep exploration via randomized value functions. *arXiv preprint arXiv:1703.07608*.

Osband, I.; Van Roy, B.; and Wen, Z. Generalization and exploration via randomized value functions. *arXiv preprint arXiv:1402.0635*.

Ostrovski, G.; Bellemare, M. G.; van den Oord, A.; and Munos, R. 2017. Count-based exploration with neural density models. In *ICML*, 2721–2730.

Punjani, A., and Abbeel, P. 2015. Deep learning helicopter dynamics models. In *ICRA*, 3223–3230.

Rajeswaran, A.; Ghotra, S.; Ravindran, B.; and Levine, S. 2017. Epopt: Learning robust neural network policies using model ensembles. *ICLR*.

Schneider, J. G. 1997. Exploiting model uncertainty estimates for safe dynamic control learning. In *NIPS*. 1047–1053.

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; and Moritz, P. 2015. Trust region policy optimization. In *ICML*, 1889–1897.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. Mujoco: A physics engine for model-based control. In *IROS*, 5026–5033.

Wang, T.; Bao, X.; Clavera, I.; Hoang, J.; Wen, Y.; Langlois, E.; Zhang, S.; Zhang, G.; Abbeel, P.; and Ba, J. 2019. Benchmarking model-based reinforcement learning. *arXiv preprint arXiv:1907.02057*.