2020 IEEE/ASME International Conference on
Advanced Intelligent Mechatronics (AIM), Boston, USA
(Virtual Conference), July 6-9, 2020

Model-Based Robot Learning Control with Uncertainty Directed Exploration

Junjie Cao[1], Yong Liu[2], Jian Yang[3] and Zaisheng Pan[4]

Abstract— **The Robot with nonlinear and stochastic dynamic challenges optimal control that relying on an analytical model. Model-free reinforcement learning algorithms have shown their potential in robot learning control without an analytical or statistical dynamic model. However, requiring numerous samples hinders its application. Model-based reinforcement learning that combines dynamic model learning with model predictive control provides promising methods to control the robot with complex dynamics. Robot exploration generates diverse data for dynamic model learning. Model predictive control exploits the approximated model to select an optimal action. There is a dilemma between exploration and exploitation. Uncertainty provides a direction for robot exploring, resulting in better exploration and exploitation trade-off. In this paper, we propose Model Predictive Control with Posterior Sampling (PSMPC) to make the robot learn to control efficiently. Our PSMPC does approximate sampling from the posterior of the dynamic model and applies model predictive control to achieve uncertainty directed exploration. In order to reduce the computational complexity of the resulting controller, we also propose a PSMPC guided policy optimization algorithm. The results of simulation in the high fidelity simulator "MuJoCo" show the effectiveness of our proposed robot learning control scheme.**

I. Introduction

Constructing a dynamic model with transfer function or state-space model plays a significant role in controlling a robot or complex mechanical system. However, it is difficult to model and control a nonlinear or stochastic dynamic system, though linearization is usually adopted with bias hindering its performance. In recent years, with the rise of machine learning and deep learning, a significant research area in robotics centers on approaches where the robot dynamic model or controller is learned. The sample efficiency of robot learning control is the focus of these approaches.

When sufficient data are available, we can rely upon machine learning methods to learn an approximation of the dynamic system as a whole. The data for learning a statistical model should be sufficient and diverse which is corresponding to the persistent excitation for system identification of a physical model. Without learning a dynamic model, reinforcement learning provides a promising method for robot learning control via value function approximation. For learning a dynamic model or a value function, data is collected by robot exploring the environment. On the other hand, robot need to exploit the approximated model about the system. There is a dilemma in every action selection: should the robot take actions that maximize the objective based on its current knowledge about the system or instead investigate poorly understood states and actions to learn a better model and potentially improve future performance. This challenge is known as the exploration and exploitation trade-off.

Instead of fully uniform random exploration, most reinforcement learning methods use ϵ-greedy exploration to trade-off exploration and exploitation [1]. However, these methods are heuristic and known to be inefficient [2]. We call these methods undirected forms of exploration, without using any knowledge about the learned model. Having an understanding of what is not yet known or well understood is critical to effective exploration for better model learning. Inspired by human intelligence in exploration, with limited data and large uncertainty there is reason to explore, while near certainty naturally transfer to exploitation. So a natural solution to trade-off exploration and exploitation originates from tracking the uncertainty about the approximated model which directs where to explore. There are two types of uncertainty that exist in learning a statistic model: epistemic and aleatoric uncertainty. Epistemic uncertainty, also called model parameter uncertainty, results from a lack of training data in certain areas of the input domain. Aleatoric uncertainty originates from intrinsic randomness of the real model, maybe caused by the noise or other inherent property.

There are a lot of studies in Bayesian machine learning that dedicate to quantify model uncertainty. Bayesian neural network [3] may be a typical example, while it suffers from difficulty in training. Monte-Carlo Dropout [4] and Ensemble [5] aim to develop faster, more practical, approximate Bayesian inference schemes for neural networks. Neural network ensemble, as a practical method that is easy in implementation, has a long history dating back to [6], and remains popular today [5], [7]. It is assumed each individual neural network will converge to similar result where data has been observed, but predictions will be diverse elsewhere. Thus, the variance of these neural networks' predictions can be interpreted as epistemic uncertainty.

In this paper, we study the robot exploration and exploitation trade-off for learning a dynamic model and achieving optimal control. It is important to distinguish and quantify both kinds of uncertainty in approximated dynamic model. With epistemic uncertainty, data collection can be directed to diminish such uncertainty and a more accurate dynamic model can be build. Though aleatoric uncertainty cannot be removed and used for exploration, it can generally predict possible outcomes and represent the real dynamic more fully for better controlling it. With the guidance of the model-based controller, we propose an efficient policy optimization method, to improve the practicability of our learning control in real-time situations.

Junjie Cao[1], Yong Liu[2] and Zaisheng Pan[4] are with the Institute of Cyber Systems and Control, Zhejiang University, China.

Jian Yang[3] is with the China Research and Development Academy of Machinery Equipment, Beijing, China.

978-1-7281-6794-7/20/$31.00 ©2020 IEEE

II. Related Work

In recent years, learning from demonstration has been successfully used in the field of robot learning control with applications in autonomous helicopter [8], playing table tennis [9] and manipulator [10]. Demonstration provides a good guidance for robot learning. While it is difficulty to access numerous demonstrations and to teach a robot to accomplish a complex task only with the demonstration. Reinforcement learning also provides a promising method for robot learning control in stochastic environments. For example, algorithm based on actor-critic methods [11] and policy gradients [12] have shown success in learning very complex robot skills in high-dimensional state spaces, such as robotic locomotion [13]. Other than learning a value approximation or a policy function, a lot of works in robot learning control rely on the dynamic model approximation [14]–[16]. However, these methods don't take into account the robot exploration for learning a better dynamic model efficiently, which is one of the contributions in our work.

Reinforcement learning algorithms with undirected exploration have achieved impressive performance in robot learning control, such as ϵ-greedy [1] and parameter randomization [17]. These exploration strategies are inefficient without direction. A lot of works [18], [19] use state novelty to direct the exploration and achieve human level performance in computer games. Upper Confidence Bound (UCB) is a prominent uncertainty directed exploration strategy that originates from bandit problem [20] and also achieves impressive success in reinforcement learning [21]. Thompson Sampling [22], also called Posterior Sampling, is also a principled exploration strategy directed by uncertainty. It samples actions according to their posterior probability of being optimal. Thompson Sampling often provides better empirical results than UCB [23]. Osband et al. successfully extended Thompson sampling to value based reinforcement learning [24]. In this paper, we resort to Thompson Sampling in the context of model-based learning control. Different from the directed exploration with information gain of learned dynamic model, presented in VIME [25], our model-based exploration strategy is directed by the uncertainty of trajectory value estimation.

Contrary to many previous works in model-based learning control [8], [10], [14] that don't take into account the representation of uncertainty, PILCO [26] approximates the dynamic model with Gaussian Process and achieves high sample-efficiency in learning a controller. In order to improve the scalability of PILCO, Gal et al. used Bayesian drop-out in neural network to track parametric model uncertainty [27]. However, there are criticisms and a counterexample about dropout as a way to make neural networks Bayesian [5]. Instead, we will represent the uncertainty of our approximated dynamic model with neural network ensemble.

III. Robot Control with Dynamic Model Learning

In this section, we first validate robot learning control with machine learning method by presenting an example of linear dynamic control. Second, we introduce how to approximate a nonlinear dynamic system using neural network. Based on the learned nonlinear dynamic, Model Predictive Control can be applied to achieve optimal control. Then, we augment the neural network dynamic model with ensemble and distribution output to represent the epistemic and aleatoric uncertainty respectively. The resulting dynamic model is an approximate and practical implementation of Bayesian neural network. In this paper we use x to represent the state of the robot system and u to represent the control command or the action taken by the robot.

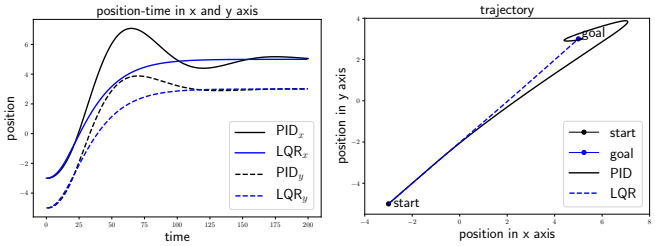

Fig. 1: Point mass position control comparisons: PID and LQR with leaned linear dynamic.

A. Optimal Control with Linear Dynamic Regression

As a key component of system identification, parameter estimation provides a basic method for robot learning control without knowing the full dynamic system beforehand. Take a simple point mass in horizontal plane for example, where the point mass is actuated to move with the force in the same plane. The physical model of the point mass can be approximated using a linear system with the state space model: $\dot{x} = Ax + bu$, where A, b represent the parameters and x is the state of the point mass, including its position and velocity. And u is the force applied in the point mass. Rewriting the state space model in discrete format: $x_{t+1} = A'x_t + b'u_t$ we can get a linear function with (x_t, u_t) as the input and x_{t+1} as the output. By controlling the point mass with varying force and collecting state and force values (x_t, u_t, x_{t+1}) during a period, a complete data set can be constructed to represent the dynamic system fully. Even though with data noise in (x_t, u_t, x_{t+1}), the parameters of the discrete state space model, i.e. A' and b', can be estimated with linear regression. Based on the learned linear dynamic model, model-based control can be applied straightforward.

The point mass can be controlled to move and stop at a fixed position with PID controller without knowing the exact value of A', b'. On the other hand, we can construct a quadratic cost function about the state and control output, then Linear Quadratic Regulator (LQR) can be applied with learned linear dynamic model. We simulate such simple example in MuJoCo physical simulator [28], with the results illustrated in Fig. 1. It is obvious that optimal control with learned dynamic has better performance compared with PID controller without extensive controller tuning.

B. Model Predictive Control with Learned Dynamic Model

Neural network provides a powerful function to approximate the dynamic model for robot control. The learned neural network dynamic model can be parameterized as $\hat{x}_{t+1} = \hat{f}_\theta(x_t, u_t)$, in which θ represents the parameter of the neural network and $\hat{f}$ is a nonlinear function approximation. Such approximated dynamic model takes as input the current state x_t and action u_t, and outputs the predicted next state $\hat{x}_{t+1}$. However, this form of dynamic model function can be difficult to train when the states x_t and x_{t+1} are too similar and its differences can not indicate the underlying dynamics well [14]. It is more benefit to learn a nonlinear function of the change in state s_t over the time step duration of Δ_t [14]. As a result, the next state prediction is:

$$\hat{x}_{t+1} = x_t + \hat{f}_\theta(x_t, u_t). \tag{1}$$

To construct a data set $\mathcal{D}$ with size $|\mathcal{D}|$, we can control the robot with a series of random commands and collect the trajectory $\tau = (x_0, u_0, \cdots, x_{T-2}, u_{T-2}, x_{T-1})$. The data set contains (x_t, u_t) and $x_{t+1} - x_t$ as the input and target output value of the nonlinear function. Then supervised learning can be used to train a neural network to approximate the dynamic model. The model learning is to minimize the mean square error (MSE):

$$J^{MSE}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(x_t, u_t, x_{t+1}) \in \mathcal{D}} \frac{1}{2} \left[(x_{t+1} - x_t) - \hat{f}_\theta(x_t, u_t) \right]^2 \tag{2}$$

To control robot as expected, we can define a cost function about the robot state and the action: $C(x_t, u_t)$. Unlike in LQR, the nonlinear cost function here usually is not quadratic. To find a sequence of commands $\mathbf{U}^{(H)} = (u_0, \cdots, u_{H-1})$ to control the robot over a finite horizon H, the cumulative cost should be minimized:

$$\mathbf{U}^{(H)} = \arg\min_{\mathbf{U}^{(H)}} \sum_{t=0}^{H-1} C(\hat{x}_t, u_t) \tag{3}$$

where $\hat{x}_0 = x_0, \hat{x}_{t+1} = \hat{x}_t + \hat{f}_\theta(\hat{x}_t, u_t)$.

We can not get an analytic solution of Eq. (3), due to the dynamic and cost functions being nonlinear. Random sampling shooting method [29] first generates a lot of candidates of action sequences randomly, then predicts the corresponding state sequences using the learned dynamic model. The candidate with the lowest expected cumulative cost will be chosen as the solution. Instead of random sampling, we use Cross-Entropy Method (CEM) [30], which samples action sequences from a distribution closer to previous action sequence samples that yielded low cumulative cost. It is more reasonable to sample the action sequences with a plan horizon $h \leq H$ instead of the whole horizon H for diminishing the cumulative error of the state propagation. Rather than executes all planned actions with length h in open-loop, robot should take only the first action. After receiving new state information x_{t+1}, replanning the optimal action sequence at the next time step construct a closed-loop control. This control scheme is known as model predictive control (MPC).

C. Uncertainty Representation in Nonlinear Dynamic Model

Though with good performance in linear dynamic learning control, linear regression has difficulty in approximating a stochastic and nonlinear dynamic model. The nonlinear dynamic model learning process requires numerous data generated in sequence during the control process. Before substantial data is available, neural network with powerful expressive ability tends to overfitting and results in high prediction error in front of the states that robot has never experienced before. Along with the prediction error, uncertainty exist in the partially learned dynamic model. Uncertainty is critical to prediction and control with learned model, while it is ignored in neural network model learning. To overcome the problem of overfitting and represent the uncertainty of learned dynamic model, we resort to the Bayesian deep learning methods.

1) Aleatoric Uncertainty Representation: In a general regression setup, the neural network takes a vector of input and generates a corresponding prediction. While, we force the neural network to output the parameter of a probability distribution, from which the prediction is sampled. The inherent randomness of a stochastic dynamic is reflected on the output distribution which is reasonable to be assumed as Gaussian. Then our proposed neural network model outputs the estimated mean and variance of the distribution, i.e. the differences between states x_t and x_{t+1} are normal distributed:

$$\hat{\Delta} = \hat{x}_{t+1} - x_t \sim \mathcal{N}(\hat{\mu}(x_t, u_t), \hat{\sigma}(x_t, u_t)|\theta), \tag{4}$$

where θ is the parameter of neural network. This neural network model with distribution output is able to capture the intrinsic randomness or aleatoric uncertainty of the dynamic system. While we assume a Gaussian distributed randomness in our example, the method can be applied to other parametric distributions in the same way.

In the training process, instead of using the mean squared error (MSE) loss function Eq. (2), we take into account the variance that we want to estimate along with the mean value. The optimization can be achieved using a maximum likelihood (ML) approach, where we take the negative log-likelihood function of the Gaussian distribution as the loss function: $-\log p(\Delta|x_t, u_t)$. After averaging over multiple samples and ignoring constants, we get the mean negative log-likelihood (MNLL) as the objective function:

$$\mathcal{J}^{MNLL}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(x_t, u_t, x_{t+1}) \in \mathcal{D}} \left[\frac{\log \hat{\sigma}^2}{2} + \frac{(\Delta - \hat{\mu})^2}{2\hat{\sigma}^2} \right], \tag{5}$$

where $\Delta = x_{t+1} - x_t$ represents the true difference between successive states and $\hat{\mu}, \hat{\sigma}$ are the estimated mean and variance with input (x_t, u_t).

2) Epistemic Uncertainty Representation: Instead of training a single neural network, we are going to train an ensemble of M networks with different random initialization. By training these neural networks with the same data, we expect all models to predict similarly in areas of sufficient training data, and the prediction will be completely different

where there is scarce or no data available. The epistemic uncertainty can be estimated with the variance of mean predictions of those neural networks, which will diminish when more training data are available. From the perspective of Bayesian neural network, the initialized ensemble neural networks can be interpreted as a prior. And random selection from the trained neural networks can be viewed as sampling from the posterior.

D. State Propagation with Learned Dynamic Model

Based on the ensemble neural networks with distribution outputs, proposed in Section III-C, we can perform model predictive control as in Section III-B with the only difference in state propagation. The overall model predictive control process is summarized in Algorithm 1.

State propagation plays an important role in optimizing Eq. (3). With one deterministic neural network approximating the dynamic model, state propagation is simple and straightforward. The neural network outputs the next prediction with the combination of previous state prediction and action $(\hat{x}_t, u_t)$ as input. For the neural network model that outputs a distribution, we maintain a population of P particles for propagating the output distribution. The propagation begins by creating P particles from the current state: $x^p_{t=0} = x_0 \forall p$. Then each particle is propagated by sampling from the output distribution: $x^p_{t+1} \sim \mathcal{N}(\hat{\mu}(x^p_t, u_t), \hat{\sigma}(x^p_t, u_t)|\theta)$, where x^p_t is the current particle and u_t the action. For neural networks in ensemble, we keep them independent in propagating particles. With such state propagation strategy the aleatoric uncertainty can be maintained and propagated.

Algorithm 1 Model Predictive Control.

Input: Approximated dynamic: f_θ and plan horizon h
Output: Average first actions: $\frac{1}{n}\sum_{i=1}^{n} \mathbf{U}_i^0$

1: Random sample $\mathbf{U}^{(h)}_{i=\{1,\cdots n\}}$;
2: **repeat**
3: Propagate states with f_θ and $\mathbf{U}^{(h)}_{i=\{1,\cdots n\}}$ by particles;
4: Evaluate $\mathbf{U}^{(h)}_{i=\{1,\cdots n\}}$ with Eq. (3) and Select $\mathbf{U}^{(h)}_{elites}$;
5: Gaussian approximation of $\mathbf{U}^{(h)}_{elites}$: $\mathcal{N}(\mu_{\mathbf{U}}, \sigma_{\mathbf{U}})$;
6: Sample $\mathbf{U}^{(h)}_{i=\{1,\cdots n\}}$ from $\mathcal{N}(\mu_{\mathbf{U}}, \sigma_{\mathbf{U}})$;
7: **until** l times of iteration

IV. Robot Learning Control with Uncertainty Directed Exploration

As presented in previous section, dynamic model learning is the key ingredient for robot learning control. The dynamic model learning process requires many data collected by robot exploration. In order to learn a dynamic model that represents all situations the robot will encounter, robot takes random exploration in the method presented in Section III. However, the random exploration by taking random action is inefficient with a lot of repetition and uninformative data collection. In this section, we first demonstrate a Multi-Armed Bandit problem where uncertainty directs the decision-making process to be more efficient without exploring where it is familiar to the agent. Then, we propose a practical exploration strategy in model-based learning control, inspired by the Thompson Sampling [22] in Multi-Armed Bandit problem.

A. Uncertainty Directed Exploration in Multi-Armed Bandit

Multi-Armed Bandit is a static sequential decision-making problem, where an agent chooses an action a from a given set $\mathcal{A}$ repeatedly over n rounds. For each chosen action, the environment reveals a reward $r_t = R(a)$. The reward function $R(a)$ is stochastic and unknown. We can approximate the reward function with the ensemble neural networks introduced in Section III-B. To illustrate the performance of uncertainty directed exploration, we train 5 neural networks with 6 and 7 data points, shown in Fig. 2, where we represent the epistemic uncertainty with the standard deviation of the ensemble networks' output means. As shown in the left of Fig. 2, with scarce data available our ensemble neural networks have high discrepancy. After choosing action 5 where our model has the highest uncertainty, it is obvious that the model uncertainty reduced sharply, which is beneficial to make better decision.

To trade-off exploration and exploitation, we can construct an acquisition function: $\mu(a) + \kappa\sigma(a)$, where μ and σ represent the mean and variance of the mean predictions of ensemble neural networks. Choosing action that maximize the acquisition function can trade-off exploration and exploitation by tuning the parameter κ. The acquisition function $\mu(a)+\kappa\sigma(a)$ is known as the Upper Confidence Bound [20].

Thompson Sampling [22], known as Posterior Sampling, provides an elegant approach that tackles the exploration-exploitation dilemma by maintaining a posterior over approximated reward function. At every time-step, Thompson Sampling draws one sample from the posterior distribution, then rank the actions according to the sampled reward function, after that the highest-ranked action will be taken. In this work, we use trained ensemble neural networks to approximate the posterior.

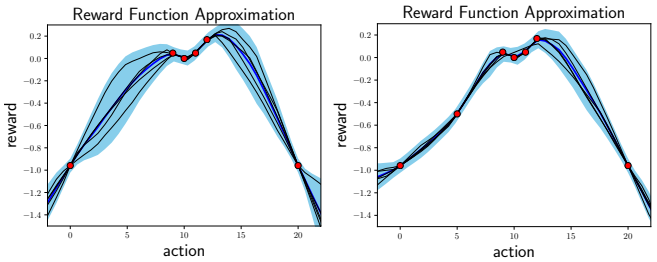

Fig. 2: Reward function approximated with 5 neural networks. Blue thick line and shadow represent the mean and stand deviation of 5 predictions. Dots represent the available data.

B. Model-Based Exploration with Thompson Sampling

The key idea in Thompson sampling is to maintain a posterior distribution over the reward model parameters

and to choose action that maximize the reward function sampled from it. We formulate the optimal control in finite horizon with dynamic model learning as Multi-Armed Bandit Problem, where Thompson Sampling can be applied for exploration and exploitation trade-off.

Given the approximated dynamic model $\hat{f}_\theta$ and the initial state of the robot system x_0, we can predict all the succeeding states with the action sequence that robot will take $\{u_0, \cdots, u_{T-1}\}$. Then we define trajectory value as the cumulative reward a robot will receive over the whole finite horizon H:

$$V_\theta(\mathbf{U}^{(H)}) = \sum_{t=0}^{H-1} R(\hat{x}_t, u_t), \tag{6}$$

where $\hat{x}_0 = x_0, \hat{x}_{t+1} = \hat{x}_t + \hat{f}_\theta(\hat{x}_t, u_t)$. $V_\theta(\mathbf{U}^{(H)})$ is a function about the whole action sequence the robot will take, and is corresponding to the reward function in Multi-Armed Bandit.

The trajectory value function accumulates the epistemic and aleatoric uncertainty of the learned dynamic model at every time step $t \in [0, H-1]$. So the distribution over dynamic model constructs the distribution over trajectory value. To apply Thompson Sampling, we sample from the posterior of trajectory value by sampling a dynamic model from its posterior. The process works as follows: (1) Sampling a dynamic model from its posterior construct a posterior estimation about the trajectory value which is a function of action sequence $\mathbf{U}^{(H)}$; (2) Maximize the trajectory value function Eq. (6), constructed by the sampled dynamic model parameter θ, with Algorithm 1. In our practical algorithm, we use M neural networks in ensemble to approximate the distribution over dynamic. The robot selects $j \sim \text{Unif}([1, .., M])$ at the beginning, then follows the model predictive control, introduced in Section III, with the j'th approximated dynamic model for the whole episode with time horizon H.

C. Guided Policy Optimization

In model predictive control, action must be optimized at every time steps with the dynamic model learned before. The action sequence optimization in model predictive control is time-consuming and hinders its application in real-time control. However, model-free reinforcement learning methods usually optimize a value function or policy function from which action can be obtained directly without additional optimization. It is benefit to optimize a policy function that outputs the action with the state as input.

We approximate the policy π with a neural network that outputs a distribution of action: $\pi = p(u|x;\gamma)$, where γ is the parameter of neural network. We define the model predictive control as a planner $u = \mathbb{P}(x) = \frac{1}{M}\sum_{j=1}^{M}\mathbb{P}_j(x)$, where we average the results of model predictive control with different dynamic model $f_{\{\theta_1,\cdots,\theta_M\}}$. Then the policy function can be learned with the maximum likelihood, where we are maximizing the probability (likelihood) of planned action in the output distribution of the policy. The loss function is the mean negative log-likelihood:

$$J_\pi^{MNLL}(\gamma) = \frac{1}{|\mathcal{D}'|}\sum_{x\in\mathcal{D}'} -\log p(\mathbb{P}(x)|x;\gamma), \tag{7}$$

where $\mathcal{D}'$ is the integrated data set containing the states that robot encountered with the policy π and the exploration strategy proposed in Section IV-B. The integrated data set can also be used to train the dynamic model in replace of the $\mathcal{D}$ in Eq. (5). With the combination of whose states, the policy optimization process prevents the distribution mismatch problem [31]. The overall algorithm proposed in this paper is summarized in Algorithm 2.

Algorithm 2

```
Initialize data set D' with robot random exploration.
Train M neural networks f_{θ_1,···,θ_M} with Eq. (5).
repeat
    Select j ~ Unif([1, ··· , M]).
    for t = 0, ··· , H − 1 do
        Get optimal u_t from MPC (Algorithm 1) with f_{θ_j};
        Take action u_t and augment D' with (x_t, u_t);
    end for
    for t = 0, ··· , H − 1 do
        Sample u_t ~ π(u|x_t);
        Take action u_t and augment D' with (x_t, u_t);
    end for
    Optimize f_{θ_1,···,θ_M} with D' and Eq. (5);
    Optimize π with sampled x from D' and Eq. (7);
until k times of iteration
```

V. Experiments

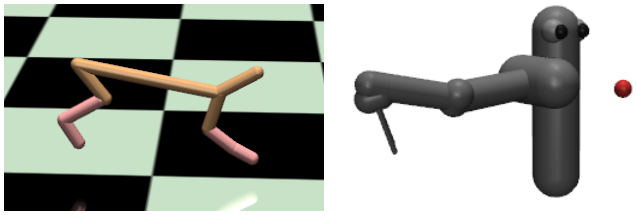

Fig. 3: Simulated "Cheetah" and "7-dof Robot Arm".

A. Experiments Description

The robots in experiments are simulated in MuJoCo [28], including "Cheetah" and "Robot Arm", shown in Fig. 3.

Cheetah is a robot constrained in forward-vertical plane. It can be represented with a state vector of 18 dimensions, including the position and velocity of its center of mass, the angles and angular velocities of its 6 joints. Cheetah can control its 6 joints with torque, i.e. its action is a vector with 6 dimensions. The goal of Cheetah is to run forward as far as possible in 1000 time steps. So the cost function is defined as the $C(x,u) = -x_{vfwd} + 0.1 \times \|u\|^2$, where x_{vfwd} represents the forward velocity which is included in state x.

The Robot Arm is with 7 degrees of freedom. The state of the robot system is a vector with 17 dimensions, including the position of the goal, the angles and angular velocities of all its 7 joints. What the controller outputs is the torque of all joints. The goal is to control its end effector to a prescribed random goal position in 150 time steps. We define the cost function as: $C(x,u) = \|g(x) - x_{goal}\|^2 + 0.01 \times \|u\|^2$, where $g(x)$ is the position of the end effector which can be calculated according to the angles of 7 joints and x_{goal} is the random position of goal which is also included in the x.

All neural networks are full connected in our experiments. We use the cumulative reward as the evaluation: $\sum_{t=0}^{H-1} R(x,u) = \sum_{t=0}^{H-1} -C(x,u)$. We compare our algorithms with related works from two aspects: the samples (steps) and the wall clock time needed for robot learning.

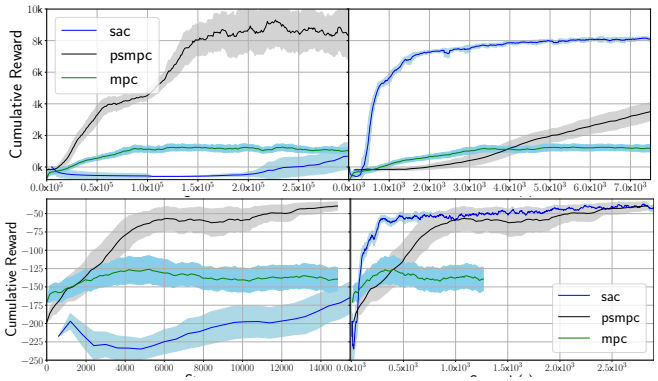

Fig. 4: The cumulative rewards during the learning process of "Cheetah" (up) and "Robot Arm" (down), compared from two aspects: samples (left) and wall clock time (right). Curves are the mean values. Shadows illustrate the standard deviation in 5 independent tests.

B. Efficient Learning Control with Uncertainty Directed Exploration

The main algorithm proposed in this paper is the Model Predictive Control with Thompson Sampling (Posterior Sampling), called PSMPC. The plain Model Predictive Control (MPC), where a neural network dynamic model is learned but without uncertainty directed exploration, is compared with our PSMPC. Soft Actor-Critic (SAC) is a state-of-the-art model-free reinforcement learning method, which aims at learning a policy with policy entropy bonus for exploration. To demonstrate the sample efficiency of model-based method we also compare our PSMPC with SAC.

The learning curves of both robots trained with different algorithms are illustrated in Fig. 4. We do comparisons from the aspects of samples and time needed for training. For better illustration, we cut out a part of curves and the whole experiments are demonstrated by putting the left and right of Fig. 4 together. It is obvious that our PSMPC achieves the competitive final results and is more sample efficient than MPC and SAC. As illustrated in the left of Fig. 4, our PSMPC is more sample efficient than MPC, by means of the Thompson Sampling for exploration and exploitation trade-off. And model-based methods especially our PSMPC has huge advantage in sample efficiency compared with model-free method.

C. Model Predictive Control Guided Policy Optimization

Though be sample efficient, our PSMPC takes more wall clock time to achieve the good performance compared with SAC, as shown in the right of Fig. 4. The reason is that our PSMPC need to plan at every time step with the approximated dynamic model. The planning is actually an optimization of action sequence. Such controller may not be applicable on real systems with limited computational power. Instead, SAC calculate an action with an approximated policy without additional optimization at every time step.

Taking computational limitations of our resulting controller into account, we propose to distill the planning of model predictive control into a neural network approximated policy. With our PSMPC guided policy optimization algorithm, the resulting policy has the similar performance compared with that of our PSMPC. The policy optimization process is shown in Fig. 5. With the resulting policy, our robot learning control has the similar computational complexity as model-free reinforcement learning during controller execution, while keeping sample efficient in the learning process.

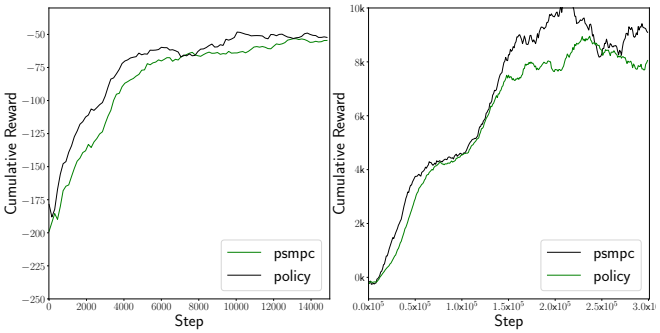

Fig. 5: The performance comparisons between PSMPC and the guided policy. "Robot Arm" (left) and "Cheetah" (right).

VI. Conclusion and Discussion

In this paper we present Model Predictive Control with Posterior Sampling (PSMPC) as an algorithm for efficient robot learning control with complex dynamics. Combining dynamic model learning and model predictive control builds the basic framework of our PSMPC. Our contributions include three parts. **First**, our approximated dynamic model is an approximation of Bayesian neural network.; **Second**, to trade-off *exploration* for better dynamic model learning and *exploitation* of approximated model for optimal control, PSMPC uses posterior sampling to achieve trajectory value uncertainty directed exploration; **Third**, our proposed PSMPC guided policy optimization method distills the model-based controller into a neural network policy, which outputs optimal actions directly with states as input without additional optimization.

From the experiments presented above, our method has shown applicability for robot learning control in systems with high-dimensional state spaces, contact-rich environment dynamics, and complex nonlinear dynamics that pose a considerable modeling challenge. The experiments that compare our PSMPC with MPC demonstrate that it is our uncertainty directed exploration that makes the model-based learning control more sample-efficient and achieve better asymptotic performance. However, the exploration in model-based learning control had been a largely under explored domain, which is an exciting direction for future work to improve the asymptotic performance of model-based method. Our exploration strategy may provide some enlightenment for system identification.

Compared with model-free methods, apart from the sample efficiency, model-based methods are also beneficial in that the learned model can be used by robot to accomplish various tasks, by simply changing the reward or cost function, without additional training. This is compelling especially when looking toward application to real robots. The improved sample efficiency and transferability would bring model-based learning control out of the simulation world and into the realm of reality. Our experiments also demonstrate the benefit of neural network in policy approximation. Combination of policy optimization and model-based control deserves further research to make robot learning process more efficient which are important for application in real time robot control.

REFERENCES

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

[2] I. Osband, B. V. Roy, and Z. Wen, "Generalization and exploration via randomized value functions," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 2377–2386.

[3] D. J. MacKay, "A practical bayesian framework for backpropagation networks," *Neural computation*, vol. 4, no. 3, pp. 448–472, 1992.

[4] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *international conference on machine learning*, 2016, pp. 1050–1059.

[5] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped dqn," in *Advances in Neural Information Processing Systems 29*, D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, Eds. Curran Associates, Inc., 2016, pp. 4026–4034.

[6] T. Heskes, "Practical confidence and prediction intervals," in *Advances in neural information processing systems*, 1997, pp. 176–182.

[7] T. Pearce, A. Brintrup, M. Zaki, and A. Neely, "High-quality prediction intervals for deep learning: A distribution-free, ensembled approach," in *International Conference on Machine Learning*, 2018, pp. 4072–4081.

[8] P. Abbeel, A. Coates, and A. Y. Ng, "Autonomous helicopter aerobatics through apprenticeship learning," *The International Journal of Robotics Research*, vol. 29, no. 13, pp. 1608–1639, 2010.

[9] S. Calinon, F. D'Halluin, E. L. Sauser, D. G. Caldwell, and A. G. Billard, "Learning and reproduction of gestures by imitation," *Robotics and Automation Magazine IEEE*, vol. 17, no. 2, pp. 44–54, 2010.

[10] J. Matas, S. James, and A. J. Davison, "Sim-to-real reinforcement learning for deformable object manipulation," 2018.

[11] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," *arXiv preprint arXiv:1506.02438*, 2015.

[12] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*, ser. ICML'15, 2015, pp. 1889–1897.

[13] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. ARTICLE, p. eaau5872, 2019.

[14] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7559–7566.

[15] S. Depeweg, J. M. Hernández-Lobato, F. Doshi-Velez, and S. Udluft, "Uncertainty decomposition in bayesian neural networks with latent variables," *arXiv preprint arXiv:1706.08495*, 2017.

[16] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," in *Advances in Neural Information Processing Systems*, 2018, pp. 4754–4765.

[17] M. Plappert, R. Houthooft, P. Dhariwal, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, and M. Andrychowicz, "Parameter space noise for exploration," in *International Conference on Learning Representations*, 2018.

[18] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *International Conference on Machine Learning (ICML)*, ser. ICML'17, vol. 2017, 2017.

[19] H. Tang, R. Houthooft, D. Foote, A. Stooke, O. X. Chen, Y. Duan, J. Schulman, F. DeTurck, and P. Abbeel, "Exploration: A study of count-based exploration for deep reinforcement learning," in *Advances in Neural Information Processing Systems*, ser. NIPS'17, 2017, pp. 2753–2762.

[20] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine learning*, vol. 47, no. 2-3, pp. 235–256, 2002.

[21] R. Y. Chen, S. Sidor, P. Abbeel, and J. Schulman, "Ucb exploration via q-ensembles," 2017.

[22] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3/4, pp. 285–294, 1933.

[23] O. Chapelle and L. Li, "An empirical evaluation of thompson sampling," in *Proceedings of the 24th International Conference on Neural Information Processing Systems*, ser. NIPS'11, 2011, pp. 2249–2257.

[24] I. Osband, D. Russo, and B. Van Roy, "(more) efficient reinforcement learning via posterior sampling," in *Advances in Neural Information Processing Systems*, 2013, pp. 3003–3011.

[25] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel, "Vime: Variational information maximizing exploration," in *Advances in Neural Information Processing Systems*, 2016, pp. 1109–1117.

[26] M. Deisenroth and C. E. Rasmussen, "Pilco: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on machine learning*, 2011, pp. 465–472.

[27] Y. Gal, R. McAllister, and C. E. Rasmussen, "Improving pilco with bayesian neural network dynamics models," in *Data-Efficient Machine Learning workshop, ICML*, vol. 4, 2016.

[28] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.

[29] A. V. Rao, "A survey of numerical methods for optimal control," *Advances in the Astronautical Sciences*, vol. 135, no. 1, pp. 497–528, 2009.

[30] Z. I. Botev, D. P. Kroese, R. Y. Rubinstein, and P. L'Ecuyer, "The cross-entropy method for optimization," in *Handbook of statistics*. Elsevier, 2013, vol. 31, pp. 35–59.

[31] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, ser. Proceedings of the fourteenth international conference on artificial intelligence and statistics, 2011, pp. 627–635.