Int. J. Appl. Math. Comput. Sci., 2019, Vol. 29, No. 2, 351–362
DOI: 10.2478/amcs-2019-0026

AN ACTIVE EXPLORATION METHOD FOR DATA EFFICIENT REINFORCEMENT LEARNING

DONGFANG ZHAO [a], JIAFENG LIU [a], RUI WU [a], DANSONG CHENG [a,*], XIANGLONG TANG [a]

[a] School of Computer Science and Technology
Harbin Institute of Technology, West Dazhi Street #92, Harbin 150001, China
e-mail: cdsinhit@hit.edu.cn

Reinforcement learning (RL) constitutes an effective method of controlling dynamic systems without prior knowledge. One of the most important and difficult problems in RL is the improvement of data efficiency. Probabilistic inference for learning control (PILCO) is a state-of-the-art data-efficient framework that uses a Gaussian process to model dynamic systems. However, it only focuses on optimizing cumulative rewards and does not consider the accuracy of a dynamic model, which is an important factor for controller learning. To further improve the data efficiency of PILCO, we propose its active exploration version (AEPILCO) that utilizes information entropy to describe samples. In the policy evaluation stage, we incorporate an information entropy criterion into long-term sample prediction. Through the informative policy evaluation function, our algorithm obtains informative policy parameters in the policy improvement stage. Using the policy parameters in the actual execution produces an informative sample set; this is helpful in learning an accurate dynamic model. Thus, the AEPILCO algorithm improves data efficiency by learning an accurate dynamic model by actively selecting informative samples based on the information entropy criterion. We demonstrate the validity and efficiency of the proposed algorithm for several challenging controller problems involving a cart pole, a pendubot, a double pendulum, and a cart double pendulum. The AEPILCO algorithm can learn a controller using fewer trials compared to PILCO. This is verified through theoretical analysis and experimental results.

Keywords: reinforcement learning, information entropy, PILCO, data efficiency.

1. Introduction

Reinforcement learning (RL) constitutes a developing field in machine learning, which is an efficient method for autonomous learning in robotics and control without prior knowledge (Sutton, 1988). RL is different from conventional supervised learning and unsupervised learning, which typically employ static training samples, in that it learns by interacting with an environment autonomously. Generally, several interactions are required to collect knowledge about an environment before the agent's learning to control. For a realistic dynamic system that is sensitive to time and increases in computation, a large number of interactions may lead to a security risk. Thus, the required number of interactions should be considered when RL is applied to actual robot systems.

A reinforcement learning agent explores the environment through interacting with it. The agent learns from these interactions and planning in order to achieve a certain goal. In reinforcement learning, there are four basic factors: the state (e.g., the position of the agent in the environment, the posture of the agent), the action (e.g., the magnitude of the force, the number of steps agent forward.), the transition probability and the reward function. The agent interacts with the environment according to a policy. The policy controls what action should be selected to interact. The reward function or the cost function rate the current policy.

RL can be formalized as a Markov decision process, consisting of the state, action, transition probability and reward function. An agent continually executes an action to interact with an environment and finally achieves an explicit task. This interaction behavior makes the agent translate from the present state to the next one according to a transition probability model and policy. The environment observation and feedback rewards obtained from interaction are used for learning the transition

*Corresponding author

probability and an appropriate policy until a predefined system target is achieved. Data efficient RL finds an appropriate policy that can maximize cumulative rewards with the minimal number of interactions (Deisenroth and Rasmussen, 2011). In data efficient RL, lack of prior knowledge about the environment is the most fundamental challenge in achieving data efficiency. It is difficult to select an optimal policy for the agent to control without an accurate dynamic model. In addition, the tradeoff between exploration and exploitation remains challenging for control systems.

Various effective algorithms for solving data efficient problems are available in the literature (Hayes and Demiris, 1994; Price and Boutilier, 2003; Bagnell and Schneider, 2001; Ng *et al.*, 2006). Previously proposed data efficiency algorithms mostly employ a model-based structure since it provides a natural advantage in achieving data efficiency compared with the model-free method. The Dyna papers are classical works in the model-based RL domain (Sutton, 1991; Silver *et al.*, 2008). There are several algorithms from the viewpoint of the model structure and stochastic optimal control (Fabisch and Metzen, 2014; Pan and Theodorou, 2014; Pan *et al.*, 2015). Alternatively, a few recent studies have combined model-based learning with deep nets (Gruslys *et al.*, 2017; Finn *et al.*, 2015; Finn and Levine, 2016; Levine *et al.*, 2016; Chebotar *et al.*, 2017; Nagabandi *et al.*, 2017; Ebert *et al.*, 2017). Moreover, the probabilistic inference for learning control (PILCO) algorithm is an excellent framework for achieving data efficiency (Deisenroth and Rasmussen, 2011; Deisenroth *et al.*, 2015).

PILCO uses a probability dynamic model instead of a single determinate one. In addition, this probability description learns the uncertainty of the dynamic model, which is an important challenge in model-based RL methods. However, PILCO focuses on maximizing cumulative rewards to learn optimal policy parameters and does not consider the accuracy of the dynamic model, which is an important factor when learning controllers. Moreover, in the policy improvement stage of PILCO, an agent minimizes the mean of an accumulated cost function to update policy parameters and then uses the parameters to simulate a new trajectory. This policy improvement strategy is essentially an exploitation-only algorithm, without exploration.

Motivated by the aforementioned limitations, we propose an active exploration PILCO (AEPILCO) algorithm. We improve the typical PILCO by considering the influence of a dynamic model. The key concept of the proposed algorithm is selecting a sample set that is helpful in training a dynamic model better. Owing to the accurate dynamic model, optimal policy parameters are learned and targets are achieved faster. To this aim information entropy is introduced to describe sample uncertainty. Samples with high uncertainty are more helpful in training an accurate dynamic model. Thus, data efficiency is achieved in terms of a carefully learned dynamic model. The improved data efficient performance of AEPILCO is verified through the simulation of several challenging control problems.

The rest of the paper is organized as follows. The typical PILCO framework is introduced in Section 2. Then, the AEPILCO algorithm is presented in Section 3. Experimental results and analysis are provided in Section 4. Finally, the conclusions are given in Section 5.

2. PILCO framework

PILCO considers a dynamic system with continuous state $\mathbf{x}$ and action $\mathbf{u}$. Formally, it considers dynamic systems

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{u}_t) + \boldsymbol{\omega}, \qquad \boldsymbol{\omega} \sim \mathcal{N}(0, \Sigma_\omega) \quad (1)$$

with continuous-valued states $\mathbf{x_t} \in \mathbb{R}^D$ and action $\mathbf{u_t} \in \mathbb{R}^F$, Gaussian system noise $\boldsymbol{\omega}$, and unknown transition dynamics f. The policy search objective is to find a policy/controller $\pi : \mathbf{x} \mapsto \pi(\mathbf{x}, \theta) = u$, which minimizes the expected long-term cost,

$$J^\pi(\boldsymbol{\theta}) = \sum_{t=0}^{T} \mathrm{E}_{\mathbf{x}_t}[c(\mathbf{x}_t)], \qquad \mathbf{x}_t \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0), \quad (2)$$

of following π for T steps, where $c(\mathbf{x}_t)$ is the cost of being in state $\mathbf{x}$ at time t, and the cost function is defined by the distance of current state to target one:

$$c(\mathbf{x}_t) = 1 - \exp\left(-\frac{1}{2\sigma_c^2} d(\mathbf{x}_t, \mathbf{x}_{target})^2\right) \in [0, 1]. \quad (3)$$

State transition is considered a Markov process. Given state $\mathbf{x}_t$, a dynamic system transfers to state $\mathbf{x}_{t+1}$ with action $\mathbf{u}_t$ according to dynamic model f, which describes the transition probability $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$. A state-action vector is defined as $\tilde{\mathbf{x}}_t = [\mathbf{x}_t, \mathbf{u}_t]$. The set of tuples $\langle \tilde{\mathbf{x}}_t, \mathbf{x}_{t+1} \rangle$ is defined as a sample. The state transition follows a dynamic model. Using the determined dynamic model, $\mathbf{x}_t$ is easy to be compute once $\tilde{\mathbf{x}}_{t-1}$ is known. The current sample can be described using the probability of the next state $p(\mathbf{x}_t)$, instead of $\tilde{\mathbf{x}}_{t-1}$.

Policy π is a function parameterized by $\boldsymbol{\theta}$. In PILCO, the policy representation is a deterministic Gaussian process with a fixed number of N basis functions. Here, 'deterministic' means that there is no uncertainty about the underlying function. Therefore, the deterministic Gaussian process is a degenerate model, which is functionally equivalent to a regularized RBF network. Our objective is to find a policy π^* which minimizes $J^\pi(\boldsymbol{\theta})$ in Eqn.(2).

In PILCO, the dynamic model is implemented as a Gaussian process, which is completely specified by its mean and covariance functions. According to the

definition of the Gaussian process transition probability, when a system is in a specific state $\mathbf{x}$, the predicted state $\mathbf{x}_*$ with action $\mathbf{u}$ follows the normal distribution,

$$p(\mathbf{x}_*|\mathbf{x},\mathbf{u}) = \mathcal{N}(m(\mathbf{x},\mathbf{u}), \Sigma(\mathbf{x},\mathbf{u})). \tag{4}$$

The squared exponential kernel function is selected as the covariance function

$$k(\mathbf{p},\mathbf{q}) = \sigma^2 \exp\Big(-\frac{\|\mathbf{p}-\mathbf{q}\|^2}{2l^2}\Big) + \delta_{pq}\sigma_\varepsilon^2, \tag{5}$$

where noise variance σ_ε, latent function variance σ and length-scale l are Gaussian process hyper-parameters, which must be learned. The initial training input is $D = \{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_n\}$. The corresponding training target is the next state set, $\{\mathbf{x}_2, \ldots, \mathbf{x}_{n+1}\}$, to which the system transits, and δ_{pq} is the Kronecker function, which is one if the two inputs, $\mathbf{p}, \mathbf{q}$, are equal and zero otherwise. An increasing number of samples is added to the training data set through constant interaction with the environment. Thus, an agent can update the hyper-parameters by retraining the dynamic model with the new training data to predict the next state. Moreover, PILCO assumes that action selection follows the normal distribution when the system is in a specific state,

$$p(\mathbf{u}|\mathbf{x}) = \mathcal{N}(\mu(\mathbf{x}), \sigma(\mathbf{x})). \tag{6}$$

In the initialization stage, assuming that the policy selection follows initial mean $\mu_0 = 0$ and covariance $\Sigma_0 = 1$, the system is executed in an actual environment for initial training dataset D. Subsequently, the agent uses this training dataset to learn the hyper-parameters of the Gaussian process. In the policy evaluation stage, policy parameter θ is evaluated by simulating t predicted steps using the learned dynamic model and parameterized policy function $\pi(\theta)$. Policy evaluation accumulates the cost function value in t predicted steps and obtains objective $J^\pi(\theta)$. Then, a policy gradient method is utilized to obtain the optimal policy parameters that will be used in the next trial. In order to implement the algorithm, PILCO makes certain assumptions and simplifications including the first-order Markov process and predicting one step forward when the agent simulates samples. Through experiments, the authors verified that the simplification of the calculation in PILCO would not affect controller learning significantly.

According to the one-step prediction with specific input derived by Deisenroth *et al.* (2015), the mean and variance are given by

$$\begin{aligned} m(\mathbf{x}_*) &= k(\mathbf{x}_*, \mathbf{x}_i)(\mathbf{K}+\sigma_\varepsilon^2\mathbf{I})^{-1}\mathbf{y}, \\ \Sigma(\mathbf{x}_*) &= k(\mathbf{x}_*, \mathbf{x}_*) \\ &\quad - k(\mathbf{x}_*, \mathbf{x}_i)(\mathbf{K}+\sigma_\varepsilon^2\mathbf{I})^{-1}k(\mathbf{x}_i, \mathbf{x}_*), \end{aligned} \tag{7}$$

where $k(\cdot,\cdot)$ is the covariance function, $\mathbf{K}$ is the covariance matrix of training inputs. $\mathbf{x}_i$ is an element of training inputs, $\mathbf{X} = \{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_n\}$, $\mathbf{y} = [\mathbf{x}_2 \ldots \mathbf{x}_{n+1}]$ is the training target, σ_ε is noise variance, which is learned in the dynamic model learning process, and $\mathbf{I}$ is the Kronecker delta matrix.

In the policy improvement stage, PILCO minimizes function $J^\pi(\theta)$ and the Broyden–Fletcher–Goldfarb–Shanno (BFGS) policy gradient method is utilized to obtain a new policy parameter (θ^*) by computing $\mathrm{d}J^\pi(\theta)/\mathrm{d}\theta$. Then, the current optimal policy is executed to generate new samples and train the new dynamic model. These processes are repeated until a predefined task is achieved.

PILCO benefits from its Gaussian process assumption and achieves data efficiency. However, the objective function, $J^\pi(\theta)$, in the policy evaluation stage depends only on the distance between the current state and the target without considering the accuracy of the dynamic model, which is an important factor for improving data efficiency.

3. Active exploration PILCO

This section describes the AEPILCO algorithm in detail. A typical Bayesian RL PILCO framework uses a Gaussian process to model a dynamic system and updates the policy parameter by minimizing the mean of accurate rewards, which are estimated by the distance between the current state and the target one. Then, the agent focuses on exploiting the current existing policy parameter to interact with the environment. However, PILCO does not consider the accuracy of the dynamic model. To resolve this problem, we propose the AEPILCO algorithm to achieve data efficiency by actively selecting samples. This is helpful for learning a more accurate dynamic model. Specifically, information entropy is utilized to describe long-term predicted samples. In the following subsections, we first describe the entropy-based sample description method and analysis. Subsequently, we examine how to use the information entropy method to achieve data efficiency in the policy evaluation and improvement stages. Finally, we describe the entire processes of the proposed algorithm and evaluate the difference between PILCO and AEPILCO.

3.1. Entropy-based sample description. Consider that a dynamic model is learned using interaction samples, which depend on a parameterized policy function. The policy parameter is updated depending on the reward accumulated in t simulation steps. Thus, the key to learning an accurate dynamic model is generating informative simulated samples in the policy evaluation stage.

We introduce information entropy to describe simulated samples. According to the Gaussian process assumption, transition probability $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$,

which describes the transition of the system from state $\mathbf{x}_t$ to state $\mathbf{x}_{t+1}$ with action $\mathbf{u}_t$, follows the normal distribution. The information entropy of predicted state distribution $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ can be described as $-\int p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) \log p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)\, \mathrm{d}\mathbf{x}_{t+1}$. This entropy describes the uncertainty of variable $\mathbf{x}_{t+1}$. High information entropy implies high uncertainty. Information entropy is used to describe the predicted sample. When predicting samples for efficient RL algorithms, the agent should select the samples with the largest uncertainty. Because the sample set with higher information entropy is helpful for learning a more accurate dynamic model, the learned model based on these high uncertainty samples exhibits a stronger generalization ability. Therefore, we select the sample $\tilde{\mathbf{x}}_t$, which can obtain the optimal predicted state $\mathbf{x}_{t+1}$, with the largest entropy. Thus, the information entropy criterion is

$$\mathbf{x}_H^* = \arg\max_{\mathbf{x}_{t+1}} \Big\{ -\int p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) \times \log p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)\, \mathrm{d}\mathbf{x}_{t+1} \Big\}, \quad (8)$$

where $\mathbf{x}_H^*$ is the optimal sample with the highest information entropy required for sampling, $\mathbf{x}_{t+1}$ ranges over all possible states, $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ describes the probability of state-action vector $\langle \mathbf{x}_t, \mathbf{u}_t \rangle$ yielding a transition to state $\mathbf{x}_{t+1}$. Based on the Gaussian process, normal distribution, and first-order Markov process assumptions, posterior probability $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)$ is specified by its mean and covariance function described in Eqn. (7).

3.2. Policy evaluation. In the AEPILCO framework, previous actual generated samples are used to learn a basic dynamic model. Subsequently, this dynamic model is utilized to predict a sequence of simulation samples, $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_t$. Our objective is to maximize the information entropy of the entire predicted state distribution. As every predicted state follows the normal distribution, the entropy of the multivariate normal distribution with probability $p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u_t})$ can be described as a continuous integration, which is given by

$$H(p) = -\underbrace{\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\cdots\int_{-\infty}^{\infty}}_{t} p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t) \times \log p(\mathbf{x}_{t+1}|\mathbf{x}_t, \mathbf{u}_t)\, \mathrm{d}\mathbf{x}_{t+1} \quad (9)$$

where $\mathbf{x}_{t+1}$ is an element of state set $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_t)'$ to be yet simulated.

According to the entropy expression for the multivariate normal distribution derived by Ahmed and Gokhale (1989), the entropy of the sample set in Eqn. (9) can be rewritten as

$$H(p) = \frac{N}{2} + \frac{N}{2}\ln(2\pi) + \frac{1}{2}\ln\left(|\mathbf{\Sigma}|\right), \quad (10)$$

where $\mathbf{\Sigma}$ is the covariance matrix of the predicted sample set, $|\cdot|$ denotes the determinant, and N is the number of samples. To maximize $H(\mathbf{X})$ in Eqn.(10), we only maximize $\ln\left(|\Sigma(\mathbf{X})|\right)$. Thus, the active exploration optimistic term is described as

$$J^E(\mathbf{X}) = \ln\left(|\mathbf{\Sigma}(\mathbf{X})|\right). \quad (11)$$

This term focuses on generating informative samples. Adding this term to the policy evaluation stage is helpful for learning an accurate dynamic model. Observe that a typical PILCO policy evaluation objective function only focuses on the distance between the current state and the target one without considering the accuracy of the dynamic model, which can contribute to improving data efficiency. We add Eqn. (11) to the policy evaluation of a typical PILCO. Then, the AEPILCO policy evaluation objective function is

$$J^\pi(\boldsymbol{\theta}) = \sum_{t=0}^{T} \mathrm{E}_{\mathbf{x}_t}[c(\mathbf{x}_t)] + \alpha \sum_{t=0}^{T} \ln\left[\mathbf{\Sigma}\left(\mathbf{x}_t\right)\right], \quad (12)$$

where α is parameter used to trade off exploration and exploitation. The first item is an objective function used in PILCO which is measured by the distance between the current state and the target one (Deisenroth *et al*., 2015). AEPILCO follows the definition of cost function $c(\mathbf{x}_t)$ in Eqn. (3).

The geometric distance from the state $\mathbf{x}_t$ to the target state $\mathbf{x}_{target}$ is denoted by d, and the parameter σ_c controls the width of the cost function according to the setup of each scenarios. We refer to the added optimization function item as the active exploration term because it helps in exploration. The covariance in Eqn. (11) contains the uncertainty information of the dynamic model. A typical PILCO is an exploitation-greedy algorithm. Thus, adding Eqn. (11) is helpful for balancing exploration and exploitation.

3.3. Policy improvement. We have shown how to achieve a policy improvement through the added active exploration item in Eqn. (12). In AEPILCO, the objective function in policy evaluation is the sum of the active exploration item and typical cost function, which is derived in PILCO. Thus, we must only compute the derivative of Eqn. (12) with respect to the policy parameter,

$$\frac{\mathrm{d}J^E(\boldsymbol{\theta})}{\mathrm{d}\boldsymbol{\theta}} = \frac{\mathrm{d}\ln\left|\mathbf{\Sigma}\left(\mathbf{x}\right)\right|}{\mathrm{d}\left|\mathbf{\Sigma}\left(\mathbf{x}\right)\right|} \frac{\mathrm{d}\left|\mathbf{\Sigma}\left(\mathbf{x}\right)\right|}{\mathrm{d}\boldsymbol{\theta}}, \quad (13)$$

where

$$\frac{\mathrm{d}\ln\left|\mathbf{\Sigma}\left(\mathbf{x}\right)\right|}{\mathrm{d}\left|\mathbf{\Sigma}\left(\mathbf{x}\right)\right|} = \frac{1}{\left|\mathbf{\Sigma}\left(\mathbf{x}\right)\right|}, \quad (14)$$

$$\frac{\mathrm{d}\left|\boldsymbol{\Sigma}\left(\mathbf{x}\right)\right|}{\mathrm{d}\boldsymbol{\theta}}=\frac{\mathrm{d}\left|\boldsymbol{\Sigma}\left(\mathbf{x}\right)\right|}{\mathrm{d}\boldsymbol{\Sigma}\left(\mathbf{x}\right)}\frac{\mathrm{d}\boldsymbol{\Sigma}\left(\mathbf{x}\right)}{\mathrm{d}\boldsymbol{\theta}}. \tag{15}$$

The matrix derivative with respect to the matrix is

$$\frac{\mathrm{d}\left|\boldsymbol{\Sigma}\left(\mathbf{x}\right)\right|}{\mathrm{d}\boldsymbol{\Sigma}\left(\mathbf{x}\right)}=\left|\boldsymbol{\Sigma}\right|\boldsymbol{\Sigma}^{-1}. \tag{16}$$

Thus, the derivative of the added active exploration term with respect to policy parameter $\boldsymbol{\theta}$ depends on $|\boldsymbol{\Sigma}(\mathbf{x})|$, $|\boldsymbol{\Sigma}|\,\boldsymbol{\Sigma}^{-1}$ and $\partial\boldsymbol{\Sigma}(\mathbf{x})/\partial\boldsymbol{\theta}$, as derived in Eqns. (14)–(16).

The covariance of predicted samples is derived in PILCO using the moment matching approximation method (Deisenroth *et al*., 2015), which exactly computes the first two moments of a predictive distribution.

The derivative of the covariance with respect to the policy parameter is similar to $\partial\boldsymbol{\mu}(\mathbf{x})/\partial\boldsymbol{\theta}$, which is derived in PILCO (Deisenroth *et al*., 2015). For $\partial\boldsymbol{\Sigma}(\mathbf{x})/\partial\boldsymbol{\theta}$, we compute the following derivative:

$$\begin{aligned}\frac{\mathrm{d}\boldsymbol{\Sigma}\left(\mathbf{x}_t\right)}{\mathrm{d}\boldsymbol{\theta}}&=\frac{\mathrm{d}\boldsymbol{\Sigma}}{\mathrm{d}p(\mathbf{u}_{t-1})}\frac{\mathrm{d}p(\mathbf{u}_{t-1})}{\mathrm{d}\boldsymbol{\theta}}\\&=\frac{\mathrm{d}\boldsymbol{\Sigma}}{\mathrm{d}\boldsymbol{\mu}_{\mathbf{u}}}\frac{\mathrm{d}\boldsymbol{\mu}_{\mathbf{u}}}{\mathrm{d}\boldsymbol{\theta}}+\frac{\mathrm{d}\boldsymbol{\Sigma}}{\mathrm{d}\boldsymbol{\Sigma}_{\mathbf{u}}}\frac{\mathrm{d}\boldsymbol{\Sigma}_{\mathbf{u}}}{\mathrm{d}\boldsymbol{\theta}},\end{aligned} \tag{17}$$

where $\partial\boldsymbol{\Sigma}/\partial\boldsymbol{\mu}_{\mathbf{u}}$ and $\partial\boldsymbol{\Sigma}/\partial\boldsymbol{\Sigma}_{\mathbf{u}}$ are propagated by long-term prediction and computed by the moment matching approximation method derived by Deisenroth *et al*. (2015).

$\partial\boldsymbol{\Sigma}/\partial\boldsymbol{\mu}_{\mathbf{u}}$ and $\partial\boldsymbol{\Sigma}/\partial\boldsymbol{\Sigma}_{\mathbf{u}}$ in Eqn. (17) are the derivatives of the mean and covariance of the action with respect to the policy parameter. The derivation computation of Eqn. (17) depends on the presentation of the policy, which is introduced in PILCO. Following PILCO, AEPILCO represents the policy as a Gaussian, and a BFGS policy gradient method is utilized to obtain new policy parameters.

Algorithm 1 summarizes the entire process of AEPILCO. Compared with a typical PILCO (Deisenroth *et al*., 2015), AEPILCO adds an active exploration term in Step 6 to the objective function in the policy evaluation stage. The PILCO framework update policy depends on the evaluation of predicted simulations, which is calculated from the dynamic model. In AEPILCO, for learning a more accurate dynamic model, we extend the policy evaluation objective function by adding an optimization term, which is used to describe the predicted samples with information entropy. Moreover, adding the optimization term is also helpful for exploration as it can describe the sample variance. This feature is analyzed in Section 3.2. Thus, AEPILCO can use minimal interactions to achieve a predefined target using the accurate dynamic model, and by balancing exploration and exploitation.

Algorithm 1. Framework of ensemble learning for our system.

Init: Initialize a random policy π with parameter $\theta\sim\mathcal{N}(\mu_0,\Sigma_0)$. Execute the policy on a real system to gather training data D.

Repeat:

learn Gaussian process dynamic model f using all data.

Repeat:

prediction: simulate the system for $p(\mathbf{x}_1),\ \ldots,p(\mathbf{x}_t)$

policy evaluation: approximate inference; obtain $J^{\pi}(\theta)=\sum_{t=0}^{T}\mathrm{E}_{\mathbf{x}_t}[c(\mathbf{x}_t)]+\alpha\sum_{t=0}^{T}\ln\left[\Sigma\left(\mathbf{x}_t\right)\right]$.

policy improvement: BFGS-based policy improvement; obtain $\mathrm{d}J^{\pi}(\theta)/\mathrm{d}\theta$.

Return new policy parameter θ^* until convergence $\pi(\theta)\leftarrow\pi(\theta^*)$

Execute the policy on the actual system to gather training data.

until the current state of reaches the target state, the task is achieved.

4. Experiments and analysis

In this section, we evaluate the AEPILCO algorithm on several challenging control tasks including benchmark problems and high-dimensional state space problems. We utilize here the simulated scenarios provided by the PILCO software package (`http://mloss.org/software/view/508/`) to verify our algorithm. The package provides six simulators of implemented scenarios as demonstration. All of them focus on the fundamental of solving the differential equations of nonlinear dynamic systems. The detailed derivation of their physical model is provided in the PILCO software package. We use four of these predesigned scenarios with different dimensions of the state space to verify our algorithm. Firstly, the comparison experiment conducted to evaluate the dynamic model learned by AEPILCO and PILCO is presented in Section 4.1. The comparison is followed by a parameter selection experiment, which is described in Section 4.2. Lastly, the experiment and analysis performed to examine the time required by PILCO and AEPILCO to achieve a goal are presented in Section 4.3. All the experiments were performed on a PC with an i5 CPU (2.57 GHz), 8 GB RAM, and the Windows 10 operating system.

4.1. Scenarios setup and verification experiment. The verification experiment is designed to evaluate the effectiveness and data-efficiency of the AEPILCO algorithm. AEPILCO learns to control four challenging

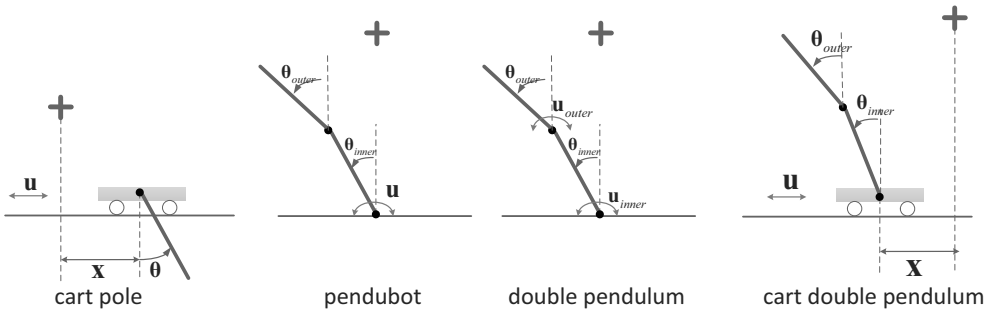


Fig. 1. Four scenarios used in our experiments.

Table 1. Dimensions of state action and policy parameter spaces.

	cart pole	pendubot	double pendulum	cart double pendulum
state space	$\mathbb{R}^4$	$\mathbb{R}^4$	$\mathbb{R}^4$	$\mathbb{R}^6$
action space	$\mathbb{R}$	$\mathbb{R}$	$\mathbb{R}^2$	$\mathbb{R}$
parameter space	$\mathbb{R}^{305}$	$\mathbb{R}^{305}$	$\mathbb{R}^{812}$	$\mathbb{R}^{1816}$

problems, involving a cart pole, a pendubot, a double pendulum, and a cart double pendulum. Figure 1 illustrates the structure of these four scenarios. The dynamic systems consist of a cart, a single/double-link pendulum, or both. Each scenario has a certain target, including maintaining the balance of the pendulum, maintaining the cart at a zero position, or both. Following PILCO, for each task, the target is to reach certain conditions and maintain them for at least 10 time steps.

To solve the four tasks in our experiment, a nonlinear policy is required. Following PILCO, AEPILCO represents the preliminary policy $\tilde{\pi}$ by

$$\tilde{\pi}(\mathbf{x}_*) = \sum_{i=1}^{N} k(\mathbf{m}_i, \mathbf{x}_*)(\mathbf{K} + \sigma_\pi^2 I)^{-1}\mathbf{t} = K(\mathbf{M}, \mathbf{x}_*)^T \alpha, \quad (18)$$

where $\mathbf{x}_*$ is a test input, $\alpha = (\mathbf{K} + 0.01I)^{-1}\mathbf{t}$, and t plays the role of a Gaussian processs training targets. $\mathbf{M} = [\mathbf{m}_1, \ldots, \mathbf{m}_N]$ are the centers of the (axis-aligned) Gaussian basis functions,

$$k(\mathbf{x}_p, \mathbf{x}_q) = \exp\left(-\frac{1}{2}(\mathbf{x}_p - \mathbf{x}_q)^T \Lambda^{-1}(\mathbf{x}_p - \mathbf{x}_q)\right), \quad (19)$$

The policy representation in (18) is called a deterministic Gaussian process with a fixed number of N basis functions. Here, 'deterministic' means that there is no uncertainty about the underlying function, that is, $\mathrm{var}_{\tilde{\pi}}[\tilde{\pi}(\mathbf{x}_*)] = 0$. Therefore, the deterministic Gaussian process is a degenerate model, which is functionally equivalent to a regularized RBF network.

In AEPILCO, we combine the following parameters as a policy parameter vector θ. The parameters of the deterministic Gaussian process in (18) are the locations M of the centers ($D \times N$ parameters), the (shared) length-scales of the Gaussian basic function (D length-scale parameters per target dimension), and the N targets t per target dimension. In the case of multivariate controls, the basic function centers M are shared. Table 1 summaries the setup of four scenarios used in this paper: for each scenario, the dimensionality of the state and action spaces is listed together with the dimension of the policy space. Following PILCO, AEPILCO uses the BFGS algorithm to realize a policy improvement. The number of policy searches for each scenarios are 12, 30, 20 and 30, respectively. The number of basic functions for RBF controllers are 100, 150, 100 and 200, respectively.

Tables 2–5 show the comparison of the partial final states for four scenarios between PILCO and AEPILCO. In the cart pole problem, the target is determined by the cart position and the inner pendulum angle. The target is achieved when the cart position reaches zero and the angle of the pendulum reaches 0, $\pm\pi$, or $\pm 2\pi$. Table 2 shows that PILCO and AEPILCO require 7 and 8 trials to achieve the target, respectively. The cart double pendulum problem is a combination of a cart pole and a double pendulum. The target is achieved when the cart position is zero and the angles of two pendulums reach 0, $\pm\pi$, or $\pm 2\pi$. The results in Table 3 show that PILCO and AEPILCO require 34 and 26 trials to achieve the target, respectively. In the double pendulum problem, two actions are applied to the inner and outer pendulums. The target is achieved when the inner and outer pendulum angles reach 0, $\pm\pi$, or $\pm 2\pi$. The results in Table 4 show that PILCO and AEPILCO achieve this target in 10 and 6 trials, respectively. In the pendubot problem, one action is applied to the inner pendulum. Similarly to the double pendulum, the pendubot's final state is the one in which the inner and outer pendulum angles reach 0, $\pm\pi$, or $\pm 2\pi$. The results in Table 5 show that PILCO and AEPILCO require 15 and 10 trials to achieve the target, respectively.

Tables 3 and 5 show the results for trials 20 to 40 and trials 6 to 20, respectively, because the first 5 trials

and first 20 trials did not achieve the target. We use fixed parameter α in the experiments. In general, Tables 2–5 illustrate that the AEPILCO algorithm can successfully learn a good controller. For all tasks, AEPILCO requires fewer trials to achieve a target compared to PILCO and shows higher data efficiency. This is mainly because AEPILCO can learn a more accurate dynamic model, which is helpful for learning a controller. In contrast, PILCO focuses on minimizing the distance between the current state and the target without considering the accuracy of the dynamic model.

4.2. Dynamic model efficiency experiment. This experiment is designed to evaluate the dynamic model learned by AEPILCO. As the simulated samples are generated by the learned dynamic model and the actual executed samples are based on the actual dynamic model, the difference between the cost function values of the simulated and actual executed samples can reflect the similarity between the dynamic models. A small difference implies that the learned dynamic model is closer to the actual one.

Figure 2 shows the comparison of the cost function values for the cart pole, pendubot, double pendulum, and cart double pendulum problems. The cost function is defined by $c(\mathbf{x})$ in Eqn. (12). Following PILCO, the cost function is defined by the distance between the current state and target one. The predicted cost function value is computed using simulated samples in the simulation stage. The real cost function value is computed using the actual executed samples in a real execute stage. The horizontal axes in Fig. 2 represent the steps of the AEPILCO convergent trial. The vertical axes in Fig. 2 represent the corresponding cost function value at each step. We show the cost function values at each step in the convergent trial. The solid round and pentagram lines indicate the PILCO predicted mean simulation and actual executed cost function values, respectively. The hollow circle and left triangle lines indicate the AEPILCO predicted mean simulation and actual executed cost function values, respectively. In particular, as simulated samples depend on the mean and covariance, the predicted mean simulated values shown in Fig. 2 are the mean cost values of the simulated samples. In general, Fig. 2 intuitively shows that the hollow circle and left triangle lines are more similar compared to the solid round and pentagram lines. This implies that the dynamic model corresponding to the hollow circle and left triangle lines

Table 2. Partial final state of 13 trials for the cart pole problem. PILCO and AEPILCO require 8 and 7 trials to achieve the target, respectively.

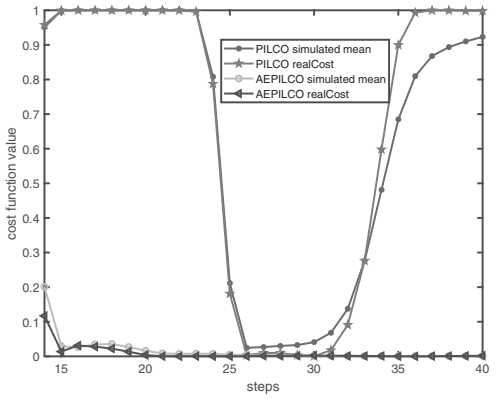
trials	x^{PILCO}	θ^{PILCO}	x^{AEPILCO}	θ^{AEPILCO}
trial 1	-16.4025	-0.38489	0.3049	-32.5808
trial 2	-0.0564	29.7306	0.6000	0.9902
trial 3	-0.1085	-0.6726	-0.0007	-0.1164
trial 4	-0.5821	-1.1992	0.9461	1.0190
trial 5	0.9344	6.2919	-0.0591	-3.1533
trial 6	0.06493	-0.7787	1.5453	-18.5497
trial 7	0.1033	9.0462	**-0.0204**	**-3.1587**
trial 8	**0.0245**	**3.0963**	**-0.0072**	**-3.1240**
trial 9	**-0.0502**	**3.0353**	**-0.0324**	**-3.1560**
trial 10	**-0.0218**	**3.1531**	**-0.0245**	**-3.1662**
trial 11	**-0.00651**	**3.1522**	**0.0158**	**-3.1099**
trial 12	**-0.0202**	**3.1308**	**0.0193**	**-3.1345**
trial 13	**-0.0512**	**3.0613**	**0.0066**	**-3.1706**

Table 3. Partial final state of trials 6C20 for the pendubot problem. PILCO and AEPILCO require 15 and 10 trials to achieve the target, respectively.

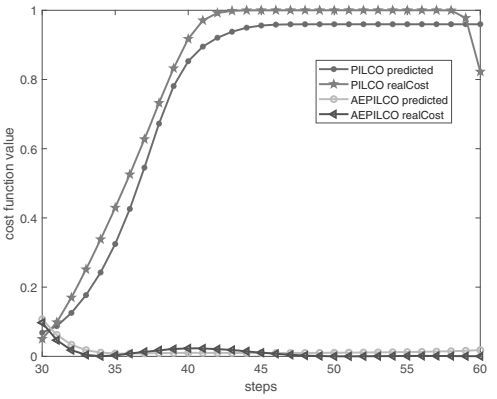
trials	$\theta^{\text{PILCO}}_{\text{inner}}$	$\theta^{\text{PILCO}}_{\text{outer}}$	$\theta^{\text{AEPILCO}}_{\text{inner}}$	$\theta^{\text{AEPILCO}}_{outer}$
trial 6	5.6452	-16.8567	0.1561	3.3439
trial 7	11.3038	-19.7802	-5.2519	-12.1740
trial 8	10.0991	-11.3890	0.1596	2.3933
trial 9	5.9804	-10.7086	1.2819	-4.9990
trial 10	1.1569	-1.3637	**-0.0100**	**-0.0279**
trial 11	-0.1333	-2.6692	**0.0496**	**-0.0111**
trial 12	3.8198	-3.1530	**0.0526**	**-0.0348**
trial 13	3.8299	-4.9931	**-0.0358**	**0.0506**
trial 14	5.5205	-0.0449	**-0.1157**	**0.0875**
trial 15	**6.0558**	**0.0308**	**-0.0182**	**0.0245**
trial 16	**6.0670**	**0.0311**	**-0.0593**	**0.0559**
trial 17	**6.1244**	**0.0211**	**-0.0732**	**0.0562**
trial 18	**6.1470**	**-0.0106**	**-0.0128**	**0.0323**
trial 19	**6.1727**	**0.0190**	**-0.0672**	**0.0390**
trial 20	**6.2578**	**-0.0083**	**-0.1000**	**0.0958**

Table 4. Partial final state of 15 trials for the double pendulum problem. PILCO and AEPILCO require 10 and 6 trials to achieve the target, respectively.

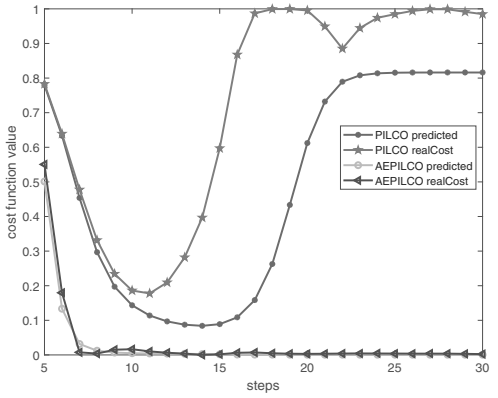
trials	$\theta^{\text{PILCO}}_{\text{inner}}$	$\theta^{\text{PILCO}}_{\text{outer}}$	$\theta^{\text{AEPILCO}}_{\text{inner}}$	$\theta^{\text{AEPILCO}}_{\text{outer}}$
trial 1	3.9865	1.8907	3.3673	5.7589
trial 2	5.3460	91.3749	3.4925	-3.8871
trial 3	1.7860	46.7438	-10.7224	-8.7774
trial 4	2.6169	0.5521	-2.8829	-12.8506
trial 5	3.8508	15.3192	-0.3987	0.3444
trial 6	4.2038	2.2230	**0.0430**	**0.0353**
trial 7	3.8352	10.7600	**-0.0407**	**0.0519**
trial 8	3.5209	12.6124	**-0.0396**	**0.0427**
trial 9	5.6984	6.4812	**-0.0538**	**0.0309**
trial 10	**6.0211**	**6.4821**	**-0.0221**	**0.0253**
trial 11	**6.1460**	**6.3697**	**-0.0421**	**0.0159**
trial 12	**6.1922**	**6.3447**	**-0.0469**	**0.0212**
trial 13	**6.2463**	**6.3320**	**-0.0052**	**0.0241**
trial 14	**6.2501**	**6.2999**	**-0.0258**	**0.0112**
trial 15	**6.2124**	**6.3446**	**-0.0650**	**0.0582**



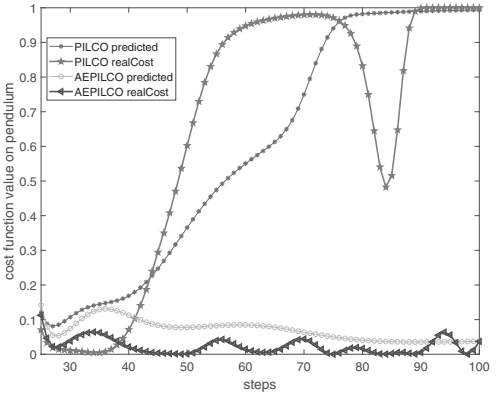
(a) cart pole



(b) pendubot



(c) double pendulum



(d) cart double pendulum

Fig. 2. Comparison in terms of the cost function value at the convergence trial with AEPILCO for the cart pole (trial #7) (a), pendubot (trial #10) (b), double pendulum (trial #6) (c) and cart double pendulum (trial #26) (d). The predicted cost function values are produced by simulation samples and the real cost function values are produced by real execute samples. The hollow circle and left triangle lines are more similar than the solid round and pentagram lines, which means the dynamic model producing the hollow circle and left triangle lines is more accurate.

is more accurate. Moreover, in Figs. 2(a)–2(c), even though the distance between the simulated and actual cost function values is larger for PILCO compared with AEPILCO, the solid round and pentagram lines show a common trend of change. In contrast, in Fig. 2(d), the actual cost function values obtained using PILCO are not smooth when the agent tends to convergence using AEPILCO. In addition, in Figs. 2(a)–2(b), the largest distance between the cost function values indicated by the pentagram and solid round lines is smaller than 0.2. In contrast, in Fig. 2(b), this distance is larger than 0.5 at most steps.

We use the Euclidean distance to quantitatively evaluate the distance between the predicted simulation and the actual executed cost function values. The comparison of the results of the high-dimensional problems is shown in Table 6. The results illustrate the distance between the actual executed cost and the simulated cost obtained using PILCO and AEPILCO, respectively. For all control problems, AEPILCO provides better performance in learning the dynamic model. This is mainly because our algorithm considers the accuracy of the dynamic model and adds the active exploration item in the policy evaluation stage. Assume that the dynamic model is learned using actual interaction samples, which depend on the policy. In the policy improvement stage, policy parameters are calculated according to the objective function, which is determined by the rewards of the simulation samples. In AEPILCO, the simulation samples are most informative owing to the information entropy criterion. Thus, we can obtain informative simulation samples and policy parameters; this is helpful in learning an accurate dynamic model.

4.3. Parameter selection experiment. This reported experiment is designed to analyze how parameter α in

Table 5. Partial final state of trials 20C40 for the cart double pendulum problem. PILCO and AEPILCO require 34 and 26 trials to achieve the target, respectively.

trials	x^{PILCO}	$\theta^{\text{PILCO}}_{\text{inner}}$	$\theta^{\text{PILCO}}_{\text{outer}}$	x^{AEPILCO}	$\theta^{\text{AEPILCO}}_{\text{inner}}$	$\theta^{\text{AEPILCO}}_{\text{outer}}$
trial 20	3.2609	3.4022	15.9468	-1.2791	3.0429	6.7714
trial 21	1.3206	3.5573	12.2109	-0.3146	0.2328	6.7714
trial 21	1.3206	3.5573	12.2109	-0.3146	0.2328	6.7714
trial 22	5.5277	3.4187	17.6096	-1.0893	5.3714	18.9294
trial 23	-2.0459	28.4699	-11.0816	-3.1812	-0.2534	5.8586
trial 24	5.5352	-17.5292	-11.8798	-0.1901	0.0484	6.2956
trial 25	4.2886	-17.3491	-9.0522	0.2354	-1.4979	-9.1185
trial 26	-5.6026	-5.2180	-2.4510	**0.0789**	**0.0057**	**6.2793**
trial 27	-2.8028	-3.2240	-29.2524	**-0.0100**	**-0.0549**	**6.2614**
trial 28	3.5142	-7.3158	-24.5851	**-0.0754**	**-0.0597**	**6.2410**
trial 29	3.9468	-0.3432	7.7067	**-0.0147**	**-0.1223**	**6.2110**
trial 30	3.2102	11.8316	7.2682	**-0.0111**	**-0.0093**	**6.2726**
trial 31	1.0757	9.0261	17.5730	**-0.0128**	**-0.0680**	**6.2313**
trial 32	3.3246	-5.3086	10.0210	**-0.0447**	**0.0304**	**6.2628**
trial 33	2.3570	-4.3986	2.0568	**-0.0899**	**0.0025**	**6.2457**
trial 34	**0.0908**	**-0.0207**	**6.1098**	**-0.0166**	**-0.0047**	**6.2522**
trial 35	**-0.0544**	**-0.0180**	**6.2091**	**-0.0101**	**-0.0902**	**6.2397**
trial 36	**0.0133**	**0.0185**	**6.3278**	**-0.0133**	**-0.0747**	**6.2313**
trial 37	**0.0221**	**0.0485**	**6.3555**	**-0.0517**	**0.0038**	**6.2743**
trial 38	**0.0129**	**0.0776**	**6.3229**	**-0.0136**	**-0.0892**	**6.2479**
trial 39	**0.0179**	**0.0367**	**6.3224**	**-0.0109**	**-0.1102**	**6.2198**
trial 40	**0.0021**	**0.0034**	**6.2780**	**-0.0165**	**-0.0483**	**6.2157**

Eqn. (12) affects convergence speed, to select appropriate parameters for each scenario, and evaluate the data efficiency of AEPILCO.

We evaluate the effect of parameter α for four scenarios. We select a fixed α for each task. Figure 3 shows the comparison of convergence speed for different values of α for the cart pole pendubot, double pendulum, and cart double pendulum problems. The horizontal axes represent the number of trials. The vertical axes represent the mean cost value for the last several steps at each trial. The solid round lines show PILCO with no exploration. Other lines denote the results obtained using AEPILCO with fixed α. The mean cost decreases as the number of trials increases. The agent achieves the target state once the mean cost in the last few steps of each trial approaches zero.

Figure 3(a) shows the mean cost for the final 16 steps of the trials with $\alpha = 0$, 0.1, 0.2, 0.3, and 0.4. When α equals 0.3, the cart pole agent can achieve the target in 4 trials while it cannot do so even within 15 trials for other cases. Figure 3(b) shows the mean cost for the final 30 steps of the trials with $\alpha = 0$, 0.002, 0.004, 0.005, and 0.006. When it equals 0.005, the pendubot agent can achieve the target in approximately 10 trials, while it achieves the target in more than 20 trials for other cases. Figure 3(c) shows the mean cost for the final 20 steps of the trials with $\alpha = 0$, 0.0005, 0.001, 0.0015, and 0.002. When α equals 0.001 and 0.0015, the double pendulum agent can achieve the target in 6 and 11 trials, respectively, while it cannot achieve the target within 20 trials in other cases. Figure 3(d) shows the mean cost for the final 70 steps of the trials with $\alpha = 0$, 0.001, 0.0001, 0.0002, and 0.0005. The last three parameters in Fig. 3(d) are better than $\alpha = 0$. The cart double pendulum agent cannot do so within 40 trials when $\alpha = 0.001$.

We consider the cost for the final few steps at each trial because the agent requires a few steps to achieve the target, and the number of required steps is different in each problem. Moreover, AEPILCO does not normalize the active exploration optimization item. Therefore, the optimal α is different for different problems.

According to the derivation of AEPILCO in Section 3, a larger α represents more exploration. However, excessive exploration may lead to divergence. Therefore, we use the following adaptive parameter selection function:

$$\alpha = \alpha_0 e^{-\omega t}, \tag{20}$$

where α_0 is the initial value of α, t is the number of trials, and ω is set according to different scenarios. This adaptive parameter selection function allows more agent exploration in the first few trials and more exploitation when the agent approaches the target. Here, we set $\alpha_0 = 0.1$, and the parameters ω for the four scenarios are 0.2, 500, 100 and 1000, respectively.

Table 7 shows the trials required to achieve the targets in each scenario. The bold numbers are the trials

Table 6. Comparison of dynamic model distances on four high dimension problems.

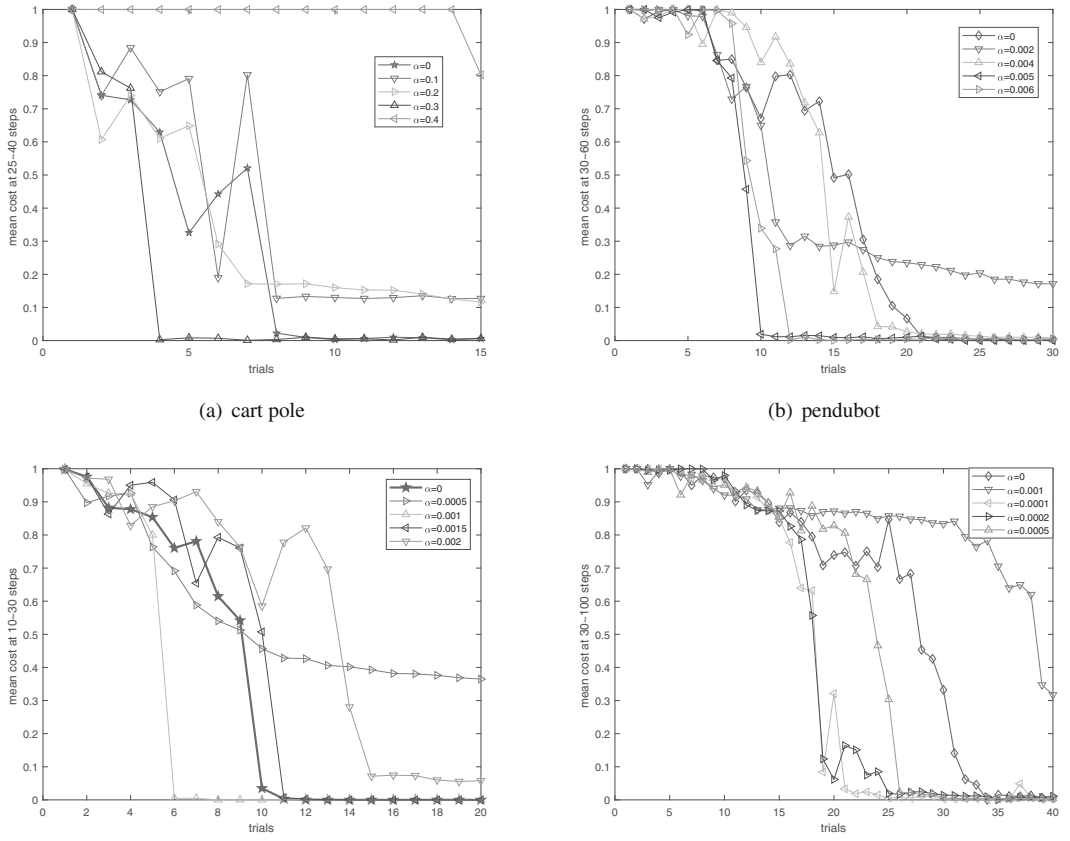
	cart pole	pendubot	double pendulum	cart double pendulum
PILCO (Deisenroth *et al*., 2015)	0.3874	0.3433	1.7250	1.9974
AEPILCO	**0.1013**	**0.0635**	**0.0787**	**0.2946**

Table 7. Results of convergent trials obtained using a typical PILCO, AEPILCO with fixed α, and AEPILCO with variable α for four high-dimensional problems.

	cart pole	pendubot	double pendulum	cart double pendulum
PILCO (Deisenroth *et al*., 2015)	8	15	10	34
AEPILCO with fixed α	**7**	**10**	**6**	**26**
AEPILCO with variable α	**4**	**9**	**5**	**24**
best improvement	50%	40%	50%	29%

Table 8. Comparison of the total times between the PILCO and AEPILCO algorithms (first two rows). Comparison of the average times for each trial between the PILCO and AEPILCO algorithms (last two rows).

	cart pole	pendubot	double pendulum	cart double pendulum
PILCO total	≈27 s	≈56 s	≈ 38 s	≈ 134 s
AEPILCO total	≈ **18 s**	≈ **41 s**	≈ **23 s**	≈ **112 s**
PILCO average	≈3.4 s	≈3.7 s	≈ 3.8 s	≈ 3.9 s
AEPILCO average	≈ **4.5 s**	≈ **4.6 s**	≈ **4.6 s**	≈ **4.7 s**



(a) cart pole

(b) pendubot

(c) double pendulum

(d) cart double pendulum

Fig. 3. Comparison of parameter α for the cart pole (a), pendubot (b), double pendulum (c) and cart double pendulum (d). Subfigures show the mean cost function values in each trial.

required by AEPILCO. The first row shows the results of PILCO. They illustrate that the AEPILCO algorithm performs better than PILCO, which is a no-exploration algorithm. In addition, AEPILCO with variable α achieves better performance compared with other cases in all scenarios. Moreover, data efficiency is reflected in Fig. 2. In Figs. 2(a), 3(c), and 2(d), when the cost function value obtained using PILCO jumps from 0.01 to 1, AEPILCO has already achieved the problem target state and the cost function value tends to 0. In Fig. 2(b), AEPILCO achieves the convergence state for the pendubot problem, while the cost function value obtained using PILCO has not converged yet. The last row in Table 7 summarizes the best improvement scale compared with a typical PILCO.

In general, for all scenarios, AEPILCO with variable α provides the best performance. This is mainly because our algorithm selects informative simulations and policy parameters at each trial, which results in learning an accurate dynamic model. Moreover, the adaptive parameter selection method increases agent exploration at the beginning of interaction and increases exploitation when the agent approaches the target. In terms of computational efficiency, the computational complexity of AEPILCO does not increase significantly compared with a typical PILCO because adding the active exploration term to the policy evaluation stage corresponds to computing one more inverse of the covariance matrix in Eqn. (11). Considering the small size of the matrix, computational complexity does not increase significantly. The comparison results of the time required by PILCO and AEPILCO to achieve a goal are shown in Table 8. The first two rows show the total time required to achieve the target in each scenario. The average time for each trial is shown in the last two rows. Even though the average time required by AEPILCO is larger than that required by PILCO, the total time required by AEPILCO is smaller compared to PILCO owing to the high data efficiency of AEPILCO.

5. Conclusions

In this paper, AEPILCO, which is an active exploration version of the data-efficient PILCO framework, was proposed to further improve data efficiency. Our algorithm utilizes an information entropy criterion to select the most informative sample set to learn a more accurate dynamic model. Compared with the data-efficient PILCO framework, our algorithm considers the accuracy of the dynamic model. This is helpful for improving data efficiency in a controller learning task. Moreover, the AEPILCO algorithm can balance exploration and exploitation because the active exploration item in the policy evaluation objective function consists of the covariance of predicted samples, which describes the amount of exploration. In summary, the contribution of AEPILCO includes a more accurate dynamic model and a higher balance between exploration and exploitation. These can effectively improve data efficiency. The simulation results obtained for several challenging control problems verify the effectiveness and data efficiency of the AEPILCO algorithm.

Also, AEPILCO has some limitations. The Gaussian process model is hard to scale to high dimensions. When a dimension is large, the computation is demanding. The policy function form, policy parameters and the reward function form are specific in AEPILCO. These limitations result from the Gaussian process. A Bayesian neural network is a good substitute. The simulation process should be derived carefully with a Bayesian neural network model. This is a prospective future research direction.

Acknowledgment

This work was supported by the National Science Foundation Council of China under the grants 61672190 and 61370162.

References

Ahmed, N.A. and Gokhale, D. (1989). Entropy expressions and their estimators for multivariate distributions, *IEEE Transactions on Information Theory* **35**(3): 688–692.

Bagnell, J.A. and Schneider, J.G. (2001). Autonomous helicopter control using reinforcement learning policy search methods, *IEEE International Conference on Robotics and Automation, Seoul, South Korea*, Vol. 2, pp. 1615–1620.

Chebotar, Y., Hausman, K., Zhang, M., Sukhatme, G., Schaal, S. and Levine, S. (2017). Combining model-based and model-free updates for trajectory-centric reinforcement learning, *arXiv:*1703.03078.

Deisenroth, M.P., Fox, D. and Rasmussen, C.E. (2015). Gaussian processes for data-efficient learning in robotics and control, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **37**(2): 408–423.

Deisenroth, M. and Rasmussen, C.E. (2011). PILCO: A model-based and data-efficient approach to policy search, *Proceedings of the 28th International Conference on Machine Learning (ICML-11), Bellevue, WA, USA*, pp. 465–472.

Ebert, F., Finn, C., Lee, A.X. and Levine, S. (2017). Self-supervised visual planning with temporal skip connections, *arXiv:*1710.05268.

Fabisch, A. and Metzen, J.H. (2014). Active contextual policy search, *Journal of Machine Learning Research* **15**(1): 3371–3399.

Finn, C. and Levine, S. (2016). Deep visual foresight for planning robot motion, *arXiv:*1610.00696.

Finn, C., Tan, X.Y., Duan, Y., Darrell, T., Levine, S. and Abbeel, P. (2015). Deep spatial autoencoders for visuomotor learning, *arXiv:*1509.06113.

Gruslys, A., Azar, M.G., Bellemare, M.G. and Munos, R. (2017). The reactor: A sample-efficient actor-critic architecture, *arXiv:*1704.04651.

Hayes, G. and Demiris, J. (1994). A robot controller using learning by imitation, *International Symposium on Intelligent Robotic Systems* **676**(5): 1257–1274.

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

Nagabandi, A., Kahn, G., Fearing, R.S. and Levine, S. (2017). Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning, *arXiv:*1708.02596.

Ng, A., Coates, A., Diel, M., Ganapathi, V., Schulte, J., Tse, B., Berger, E. and Liang, E. (2006). Autonomous inverted helicopter flight via reinforcement learning, *in* M.H. Ang Jr. and O. Khatib (Eds.), *Experimental Robotics IX*, Springer, Berlin/Heidelberg, pp. 363–372.

Pan, Y. and Theodorou, E.A. (2014). Probabilistic differential dynamic programming, *Advances in Neural Information Processing Systems* **3**: 1907–1915.

Pan, Y., Theodorou, E.A. and Kontitsis, M. (2015). Sample efficient path integral control under uncertainty, *Advances in Neural Information Processing Systems* **2015**: 2314–2322.

Price, B. and Boutilier, C. (2003). Accelerating reinforcement learning through implicit imitation, *Journal of Artificial Intelligence Research* **19**: 569–629.

Silver, D., Sutton, R.S. and Müller, M. (2008). Sample-based learning and search with permanent and transient memories, *International Conference on Machine Learning, Helsinki, Finland*, pp. 968–975.

Sutton, R.S. (1988). Learning to predict by the methods of temporal differences, *Machine Learning* **3**(1): 9–44.

Sutton, R.S. (1991). Dyna, an integrated architecture for learning, planning, and reacting, *ACM Sigart Bulletin* **2**(4): 160–163.

Dongfang Zhao received the BSc degree from Southwest University, Chongqing, China, in 2012, and the MSc degree from the Harbin Institute of Technology, China, in 2015. She is currently a PhD candidate at the School of Computer Science and Technology, Harbin Institute of Technology. Her research interests is model based reinforcement learning.

Jiafeng Liu received his PhD degree from the Harbin Institute of Technology, China, in 1996. He is currently an associate professor at the School of Computer Science and Technology, Harbin Institute of Technology. His research interests cover image and video analysis, optimal character recognition, pattern recognition, machine learning and artificial intelligence. He has published over 40 papers in refereed international journals.

Rui Wu is an associate professor. He received the PhD degree in computer application technology from the Harbin Institute of Technology in 2010. His a research interests include computer vision, character recognition, robot intelligence, and embedded systems.

Dansong Cheng received the BSc and PhD degrees from the Harbin Institute of Technology, China, in 1997 and 2009, respectively, and the MSc degree in communication engineering from the Chiba Institute of Technology, Chitanma, Japan, in 2001. Since 2002, he has been with the Harbin Institute of Technology, where he became an associate professor in 2012. His current research interests include machine learning, medical image processing, and pattern recognition.

Xianglong Tang received his PhD degree from the Harbin Institute of Technology, China, in 1995. He is currently a professor at the School of Computer Science and Technology and the director of the Research Center of Pattern Recognition, both in the Harbin Institute of Technology. His main research interests are focused on Chinese character recognition, medical imaging and biometrics, computer vision and pattern recognition. He has published over 80 papers in refereed international journals.

Received: 18 July 2018
Revised: 28 December 2018
Re-revised: 31 January 2019
Accepted: 31 January 2019