Model-Based Policy Gradients with Parameter-Based Exploration by Least-Squares Conditional Density Estimation

Voot Tangkaratt[1], Syogo Mori[1], Tingting Zhao[1]
Jun Morimoto[2], and Masashi Sugiyama[1]

[1]Tokyo Institute of Technology, Japan.
{voot@sg., mori@sg., tingting@sg., sugi@}cs.titech.ac.jp

[2]ATR Computational Neuroscience Labs, Japan
xmorimo@atr.jp

Abstract

The goal of reinforcement learning (RL) is to let an agent learn an optimal control policy in an unknown environment so that future expected rewards are maximized. The model-free RL approach directly learns the policy based on data samples. Although using many samples tends to improve the accuracy of policy learning, collecting a large number of samples is often expensive in practice. On the other hand, the model-based RL approach first estimates the transition model of the environment and then learns the policy based on the estimated transition model. Thus, if the transition model is accurately learned from a small amount of data, the model-based approach is a promising alternative to the model-free approach. In this paper, we propose a novel model-based RL method by combining a recently proposed model-free policy search method called *policy gradients with parameter-based exploration* and the state-of-the-art transition model estimator called *least-squares conditional density estimation*. Through experiments, we demonstrate the practical usefulness of the proposed method.

Keywords

Reinforcement learning, transition model estimation, conditional density estimation.

1 Introduction

Reinforcement learning (RL) is a framework to let an agent learn an optimal control policy in an unknown environment so that expected future rewards are maximized (Kaelbling et al., 1996). The RL methods developed so far can be categorized into two types: *Policy iteration* where policies are learned based on value function approximation (Sutton and Barto, 1998; Lagoudakis and Parr, 2003) and *policy search* where policies are learned directly to maximize expected future rewards (Williams, 1992; Dayan and Hinton, 1997; Sutton et al., 2000; Kakade, 2002; Sehnke et al., 2010; Zhao et al., 2013).

1.1 Policy Iteration VS. Policy Search

A value function represents expected future rewards as a function of a state or a state-action pair. In the policy iteration framework, approximation of the value function for the current policy and improvement of the policy based on the learned value function are iteratively performed until an optimal policy is found. Thus, accurately approximating the value function is a challenge in the value function based approach. So far, various machine learning techniques have been employed for better value function approximation, such as least-squares approximation (Lagoudakis and Parr, 2003), manifold learning (Sugiyama et al., 2008), efficient sample reuse (Hachiya et al., 2009), active learning (Akiyama et al., 2010), and robust learning (Sugiyama et al., 2010a).

However, because policy functions are learned indirectly via value functions in policy iteration, improving the quality of value function approximation does not necessarily yield a better policy function. Furthermore, because a small change in value functions can cause a big change in policy functions, it is not safe to use the value function based approach for controlling expensive dynamic systems such as a humanoid robot. Another weakness of the value function approach is that it is difficult to handle continuous actions because a maximizer of the value function with respect to an action needs to be found for policy improvement.

On the other hand, in the policy search approach, policy functions are determined so that expected future rewards are directly maximized. A popular policy search method is to update policy functions via gradient ascent. However, a classic policy gradient method called REINFORCE (Williams, 1992) tends to produce gradient estimates with large variance, which results in unreliable policy improvement (Peters and Schaal, 2006). More theoretically, it was shown that the variance of policy gradients can be proportional to the length of an agent's trajectory, due to the stochasticity of policies (Zhao et al., 2012). This can be a critical limitation in RL problems with long trajectories.

To cope with this problem, a novel policy gradient method called *policy gradients with parameter-based exploration* (PGPE) was proposed (Sehnke et al., 2010). In PGPE, deterministic policies are used to suppress irrelevant randomness and useful stochasticity is introduced by drawing policy parameters from a prior distribution. Then, instead of policy parameters, hyper-parameters included in the prior distribution are learned from data. Thanks to this prior-based formulation, the variance of gradient estimates in PGPE is independent of the length of an agent's trajectory (Zhao et al., 2012). However, PGPE still suffers from an instability problem in small sample cases. To further improve the practical performance of PGPE, an efficient sample reuse method called *importance-weighted PGPE* (IW-PGPE) was proposed recently and demonstrated to achieve the state-of-the-art performance (Zhao et al., 2013).

1.2 Model-Based VS. Model-Free

The RL methods reviewed above are categorized into the *model-free* approach, where policies are learned without explicitly modeling the unknown environment (i.e., the transition probability of the agent in the environment). On the other hand, an alternative approach called the *model-based* approach explicitly models the environment in advance and uses the learned environment model for policy learning (Wang and Dietterich, 2003; Deisenroth and Rasmussen, 2011). In the model-based approach, no additional sampling cost is necessary to generate artificial samples from the learned environment model.

Model-based methods are the predominant approach for fast and data-efficient learning. For example, given a fixed budget for data collection, IW-PGPE requires us to determine the *sampling schedule* in advance. More specifically, we need to decide, e.g., whether many samples are gathered in the beginning or only a small batch of samples are collected for a longer period. However, optimizing the sampling schedule in advance is not possible without strong prior knowledge. Thus, we need to just blindly design the sampling schedule in practice, which can cause significant performance degradation. On the other hand, the model-based approach does not suffer from this problem because we can draw as many trajectory samples as we want from the learned transition model without additional sampling costs.

Another advantage of the model-based approach lies in *baseline subtraction.* In the gradient-based policy search methods such as REINFORCE and PGPE, subtraction of a baseline from a gradient estimate is a vital technique to reduce the estimation variance of policy gradients (Peters and Schaal, 2006; Zhao et al., 2013). If the baseline is estimated from samples that are statistically independent of samples used for the estimation of policy gradients, variance reduction can be carried out without increasing the estimation bias. However, such independent samples are not available in practice (if available, they should be used for policy gradient estimation), and thus variance reduction by baseline subtraction is practically performed at the expense of bias increase. On the other hand, in the model-based scenario, we can draw as many trajectory samples as we want from the learned transition model without additional sampling costs. Therefore, two statistically independent sets of samples can be generated and they can be separately used for policy gradient estimation and baseline estimation.

1.3 Transition Model Learning by Least-Squares Conditional Density Estimation

If the unknown environment is accurately approximated, the model-based approach can fully enjoy all the above advantages. However, accurately estimating the transition model from a limited amount of trajectory data in multi-dimensional continuous state and action spaces is highly challenging. Although the model-based method that does not require an accurate transition model was developed (Abbeel et al., 2006), it is only applicable to deterministic environments, which significantly limits its range of applications in practice. On the other hand, a recently proposed model-based policy search method called PILCO (Deisenroth and Rasmussen, 2011) learns a probabilistic transition model by the Gaussian process (GP) (Rasmussen and Williams, 2006), and explicitly incorporates long-term model uncertainty. However, PILCO requires states and actions to follow Gaussian distributions and the reward function to be a particular exponential form to ensure that the policy evaluation is performed in a closed form and policy gradients are computed analytically for policy improvement. These strong requirements make PILCO practically restrictive.

To overcome such limitations of existing approaches, we propose a highly practical policy-search algorithm by extending the model-free PGPE method to the model-based scenario. In the proposed model-based PGPE (M-PGPE) method, the transition model is learned by the state-of-the-art non-parametric conditional density estimator called *least-squares conditional density estimation* (LSCDE) (Sugiyama et al., 2010b), which can handle multi-modal distributions directly. LSCDE has various superior properties:

- It can directly handle multi-dimensional inputs and outputs.
- It achieves the optimal convergence rate (Kanamori et al., 2012).
- It has high numerical stability (Kanamori et al., 2013).
- It is robust against outliers (Sugiyama et al., 2012).
- Its solution can be analytically and efficiently computed just by solving a system of linear equations (Kanamori et al., 2009).
- Generating samples from the learned conditional density is straightforward.

Through experiments, we demonstrate that the proposed M-PGPE method is a promising approach.

The rest of this paper is structured as follows. In Section 2, we formulate the RL problem and review model-free RL methods including PGPE. We then propose the model-based PGPE method in Section 3, and experimentally demonstrate its usefulness in Section 4. Finally, we conclude in Section 5.

2 Problem Formulation and Model-Free Policy Search

In this section, we first formulate our RL problem and review existing model-free policy search methods.

2.1 Formulation

Let us consider a Markov decision problem consisting of the following elements:

- $\mathcal{S}$: A set of continuous states.
- $\mathcal{A}$: A set of continuous actions.
- $p(\boldsymbol{s})$: The (unknown) probability density of initial states.

- $p(\boldsymbol{s}'|\boldsymbol{s}, a)$: The (unknown) conditional probability density of visiting state $\boldsymbol{s}'$ from state $\boldsymbol{s}$ by action a.
- $R(\boldsymbol{s}, a, \boldsymbol{s}')$: The immediate reward function for the transition from $\boldsymbol{s}$ to $\boldsymbol{s}'$ by a.

Let $p(a|\boldsymbol{s}, \boldsymbol{\theta})$ be a policy of an agent parameterized by $\boldsymbol{\theta} \in \Theta$, which is the conditional probability density of taking action $a \in \mathcal{A}$ at state $\boldsymbol{s} \in \mathcal{S}$. Let

$$h := [\boldsymbol{s}_1, a_1, \ldots, \boldsymbol{s}_T, a_T, \boldsymbol{s}_{T+1}]$$

be a trajectory $h \in \mathcal{H}$, where $\mathcal{H} := \mathcal{S} \otimes \mathcal{A} \cdots \otimes \mathcal{S} \otimes \mathcal{A} \otimes \mathcal{S}$ is the trajectory space. Trajectory is a sequence of states and actions with finite length T generated as follows: First, the initial state $\boldsymbol{s}_1$ is determined following the initial-state probability density $p(\boldsymbol{s})$. Then action a_1 is chosen following policy $p(a|\boldsymbol{s}, \boldsymbol{\theta})$, and next state $\boldsymbol{s}_2$ is determined following the transition probability density $p(\boldsymbol{s}'|\boldsymbol{s}, a)$. This process is repeated T times.

Let $r(h)$ be the return for trajectory h, which is the discounted sum of future rewards the agent can obtain:

$$r(h) := \sum_{t=1}^{T} \gamma^{t-1} R(\boldsymbol{s}_t, a_t, \boldsymbol{s}_{t+1}), \tag{1}$$

where $\gamma \in (0, 1]$ is a discount factor. The expected return is given by

$$J(\boldsymbol{\theta}) := \int_{\mathcal{H}} r(h) p(h|\boldsymbol{\theta}) \mathrm{d}h, \tag{2}$$

where $p(h|\boldsymbol{\theta})$ is the probability density of observing trajectory h:

$$p(h|\boldsymbol{\theta}) = p(\boldsymbol{s}_1) \prod_{t=1}^{T} p(a_t|\boldsymbol{s}_t, \boldsymbol{\theta}) p(\boldsymbol{s}_{t+1}|\boldsymbol{s}_t, a_t). \tag{3}$$

The goal of RL is to find optimal policy parameter $\boldsymbol{\theta}^*$ that maximizes the expected return $J(\boldsymbol{\theta})$:

$$\boldsymbol{\theta}^* := \operatorname*{argmax}_{\boldsymbol{\theta} \in \Theta} J(\boldsymbol{\theta}). \tag{4}$$

2.2 REINFORCE

REINFORCE (Williams, 1992) is a classic method for learning the policy parameter $\boldsymbol{\theta}$ via gradient ascent:

$$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \varepsilon \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}), \tag{5}$$

where $\varepsilon > 0$ denotes the learning rate and $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ denotes the gradient of $J(\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$.

The gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ can be expressed as

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) &= \int_{\mathcal{H}} r(h)\nabla_{\boldsymbol{\theta}} p(h|\boldsymbol{\theta})\mathrm{d}h \\ &= \int_{\mathcal{H}} r(h)p(h|\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}} \log p(h|\boldsymbol{\theta})\mathrm{d}h \\ &= \int_{\mathcal{H}} r(h)p(h|\boldsymbol{\theta})\sum_{t=1}^{T}\nabla_{\boldsymbol{\theta}} \log p(a_t|\boldsymbol{s}_t,\boldsymbol{\theta})\mathrm{d}h, \end{aligned} \tag{6}$$

where we used

$$\nabla_{\boldsymbol{\theta}} p(h|\boldsymbol{\theta}) = p(h|\boldsymbol{\theta})\nabla_{\boldsymbol{\theta}} \log p(h|\boldsymbol{\theta}). \tag{7}$$

In the above expression, the probability density of trajectories, $p(h|\boldsymbol{\theta})$, is unknown. Suppose that we are given N trajectory samples $\{h_n\}_{n=1}^{N}$ for the current policy, where

$$h_n = [\boldsymbol{s}_1^n, a_1^n, \ldots, \boldsymbol{s}_T^n, a_T^n, \boldsymbol{s}_{T+1}^n]. \tag{8}$$

Then the expectation over $p(h|\boldsymbol{\theta})$ can be approximated by the empirical average over the trajectory samples $\{h_n\}_{n=1}^{N}$, i.e., an empirical approximation of the gradient $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})$ is given by

$$\nabla_{\boldsymbol{\theta}} \widehat{J}(\boldsymbol{\theta}) := \frac{1}{N}\sum_{n=1}^{N} r(h_n)\sum_{t=1}^{T}\nabla_{\boldsymbol{\theta}} \log p(a_t^n|\boldsymbol{s}_t^n,\boldsymbol{\theta}). \tag{9}$$

It is known (Peters and Schaal, 2006) that the variance of the above gradient estimator can be reduced by subtracting the baseline b:

$$\nabla_{\boldsymbol{\theta}} \widehat{J}^b(\boldsymbol{\theta}) := \frac{1}{N}\sum_{n=1}^{N} (r(h_n)-b)\sum_{t=1}^{T}\nabla_{\boldsymbol{\theta}} \log p(a_t^n|\boldsymbol{s}_t^n,\boldsymbol{\theta}), \tag{10}$$

where

$$b = \frac{\frac{1}{N}\sum_{n=1}^{N} r(h_n)\left\|\nabla_{\boldsymbol{\theta}} \log p(h_n|\boldsymbol{\theta})\right\|^2}{\frac{1}{N}\sum_{n=1}^{N}\left\|\nabla_{\boldsymbol{\theta}} \log p(h_n|\boldsymbol{\theta})\right\|^2}. \tag{11}$$

Let us consider the following Gaussian policy model with policy parameter $\boldsymbol{\theta} = (\boldsymbol{\mu}, \sigma^2)$:

$$p(a|\boldsymbol{s},\boldsymbol{\theta}) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(a-\boldsymbol{\mu}^\top\boldsymbol{\phi}(\boldsymbol{s}))^2}{2\sigma^2}\right), \tag{12}$$

where $^\top$ denotes the transpose, $\boldsymbol{\mu}$ is the Gaussian mean, σ is the Gaussian standard deviation, and $\boldsymbol{\phi}(\boldsymbol{s})$ is the vector of basis functions. Then the policy gradients are explicitly expressed as

$$\nabla_{\boldsymbol{\mu}} \log p(a|\boldsymbol{s},\boldsymbol{\theta}) = \frac{a-\boldsymbol{\mu}^\top\boldsymbol{\phi}(\boldsymbol{s})}{\sigma^2}\boldsymbol{\phi}(\boldsymbol{s}), \tag{13}$$

$$\nabla_{\sigma} \log p(a|\boldsymbol{s},\boldsymbol{\theta}) = \frac{(a-\boldsymbol{\mu}^\top\boldsymbol{\phi}(\boldsymbol{s}))^2-\sigma^2}{\sigma^3}. \tag{14}$$

REINFORCE is a simple policy-search algorithm that directly updates policies to increase the expected return. However, gradient estimates tend to have large variance even if it is combined with variance reduction by baseline subtraction. For this reason, policy update by REINFORCE tends to be unreliable (Peters and Schaal, 2006). In particular, the variance of gradient estimates in REINFORCE can be proportional to the length of the trajectory, T, due to the stochasticity of policies (Zhao et al., 2012). This can be a critical limitation when the trajectory is long.

2.3 Policy Gradients with Parameter-Based Exploration (PGPE)

To overcome the above limitation of REINFORCE, a novel policy-search method called *policy gradients with parameter-based exploration* (PGPE) was proposed (Sehnke et al., 2010). In PGPE, a deterministic policy (such as the linear policy) is adopted, and the stochasticity for exploration is introduced by drawing the policy parameter $\boldsymbol{\theta}$ from a prior distribution $p(\boldsymbol{\theta}|\boldsymbol{\rho})$ with hyper-parameter $\boldsymbol{\rho}$. Thanks to this per-trajectory formulation, the variance of gradient estimates can be drastically reduced.

In the PGPE formulation, the hyper-parameter $\boldsymbol{\rho}$ is learned via gradient ascent:

$$\boldsymbol{\rho} \leftarrow \boldsymbol{\rho} + \varepsilon \nabla_{\boldsymbol{\rho}} J(\boldsymbol{\rho}), \tag{15}$$

where $\varepsilon > 0$ denotes the learning rate and $\nabla_{\boldsymbol{\rho}} J(\boldsymbol{\rho})$ denotes gradients of the expected return with respect to $\boldsymbol{\rho}$. The expected return is represented as a function of $\boldsymbol{\rho}$:

$$J(\boldsymbol{\rho}) := \int_{\Theta} \int_{\mathcal{H}} r(h) p(h|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{\rho}) \mathrm{d}h \mathrm{d}\boldsymbol{\theta}. \tag{16}$$

Differentiating this with respect to $\boldsymbol{\rho}$, we have

$$\begin{aligned} \nabla_{\boldsymbol{\rho}} J(\boldsymbol{\rho}) &= \int_{\Theta} \int_{\mathcal{H}} r(h) p(h|\boldsymbol{\theta}) \nabla_{\boldsymbol{\rho}} p(\boldsymbol{\theta}|\boldsymbol{\rho}) \mathrm{d}h \mathrm{d}\boldsymbol{\theta} \\ &= \int_{\Theta} \int_{\mathcal{H}} r(h) p(h|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{\rho}) \nabla_{\boldsymbol{\rho}} \log p(\boldsymbol{\theta}|\boldsymbol{\rho}) \mathrm{d}h \mathrm{d}\boldsymbol{\theta}. \end{aligned} \tag{17}$$

Because of the per-trajectory formulation, trajectory samples in the PGPE framework are accompanied with policy parameters, i.e., $\{(h_n, \boldsymbol{\theta}_n)\}_{n=1}^N$. The probability density of trajectories $p(h|\boldsymbol{\theta})$ is unknown in the model-free methods. Then an empirical estimator of the above gradient (with baseline subtraction) based on the paired samples, is given as follows (Zhao et al., 2012):

$$\nabla_{\boldsymbol{\rho}} \widehat{J}^b(\boldsymbol{\rho}) := \frac{1}{N} \sum_{n=1}^{N} (r(h_n) - b) \nabla_{\boldsymbol{\rho}} \log p(\boldsymbol{\theta}_n|\boldsymbol{\rho}), \tag{18}$$

where

$$b = \frac{\frac{1}{N} \sum_{n=1}^{N} r(h_n) \left\| \nabla_{\boldsymbol{\rho}} \log p(\boldsymbol{\theta}_n|\boldsymbol{\rho}) \right\|^2}{\frac{1}{N} \sum_{n=1}^{N} \left\| \nabla_{\boldsymbol{\rho}} \log p(\boldsymbol{\theta}_n|\boldsymbol{\rho}) \right\|^2}. \tag{19}$$

Let us employ the linear deterministic policy, i.e., action a is chosen[1] as

$$a = \boldsymbol{\theta}^\top \boldsymbol{\phi}(\boldsymbol{s}) \tag{20}$$

for some basis function $\boldsymbol{\phi}$. The parameter vector $\boldsymbol{\theta}$ is drawn from the Gaussian prior distribution with hyper-parameter $\boldsymbol{\rho} = (\boldsymbol{\eta}, \boldsymbol{\tau})$. Here $\boldsymbol{\eta}$ denotes the Gaussian mean vector and $\boldsymbol{\tau}$ denotes the vector consisting of the Gaussian standard deviation in each element:

$$p(\theta_i|\rho_i) = \frac{1}{\sqrt{2\pi\tau_i^2}} \exp\left(-\frac{(\theta_i - \eta_i)^2}{2\tau_i^2}\right), \tag{21}$$

where θ_i, ρ_i, η_i, and τ_i are the ith elements of $\boldsymbol{\theta}$, $\boldsymbol{\rho}$, $\boldsymbol{\eta}$, and $\boldsymbol{\tau}$, respectively. Then the derivatives of $\log p(\boldsymbol{\theta}|\boldsymbol{\rho})$ with respect to η_i and τ_i are given as follows:

$$\nabla_{\eta_i} \log p(\boldsymbol{\theta}|\boldsymbol{\rho}) = \frac{\theta_i - \eta_i}{\tau_i^2}, \tag{22}$$

$$\nabla_{\tau_i} \log p(\boldsymbol{\theta}|\boldsymbol{\rho}) = \frac{(\theta_i - \eta_i)^2 - \tau_i^2}{\tau_i^3}. \tag{23}$$

2.4 Importance-Weighted PGPE (IW-PGPE)

A popular idea to further improve the performance of RL methods is to reuse previously collected samples (Sutton and Barto, 1998; Hachiya et al., 2009). Such a sample-reuse strategy is particularly useful when data sampling costs are high (e.g., robot control).

Importance-weighted PGPE (IW-PGPE) (Zhao et al., 2013) combines the sample-reuse idea with PGPE. Technically, IW-PGPE can be regarded as an off-policy extension of PGPE, where data collecting policies are different from the current policy (more precisely, in the PGPE formulation, data collecting policies and the current policy are drawn from prior distributions with different hyper-parameter values). Let $\boldsymbol{\rho}$ be the hyper-parameter for the current policy and $\boldsymbol{\rho}'$ be the hyper-parameter for a data collecting policy. Let us denote trajectory samples collected with hyper-parameter $\boldsymbol{\rho}'$ as $\{(\boldsymbol{\theta}'_n, h'_n)\}_{n=1}^{N'}$.

When the data collecting policy is different from the current policy, *importance sampling* is a useful technique to correct the estimation bias caused by differing distributions (Sugiyama and Kawanabe, 2012). More specifically, the gradient is estimated as

$$\nabla_{\boldsymbol{\rho}} \widehat{J}_b^w(\boldsymbol{\rho}) = \frac{1}{N'} \sum_{n=1}^{N'} w(\boldsymbol{\theta}'_n)(r(h'_n) - b) \nabla_{\boldsymbol{\rho}} \log p(\boldsymbol{\theta}'_n|\boldsymbol{\rho}), \tag{24}$$

where $w(\boldsymbol{\theta})$ is the *importance weight* defined as

$$w(\boldsymbol{\theta}) := \frac{p(\boldsymbol{\theta}|\boldsymbol{\rho})}{p(\boldsymbol{\theta}|\boldsymbol{\rho}')}, \tag{25}$$

[1]In the current formulation of PGPE, the action a is assumed to be a scalar. For problems with multi-dimensional actions, policies may be learned independently for each dimension.

and b is the baseline given by

$$b = \frac{\frac{1}{N'}\sum_{n=1}^{N'} r(h'_n) w^2(\boldsymbol{\theta}'_n) \left\|\nabla_{\boldsymbol{\rho}} \log p(\boldsymbol{\theta}'_n|\boldsymbol{\rho})\right\|^2}{\frac{1}{N'}\sum_{n=1}^{N'} w^2(\boldsymbol{\theta}'_n) \left\|\nabla_{\boldsymbol{\rho}} \log p(\boldsymbol{\theta}'_n|\boldsymbol{\rho})\right\|^2}. \tag{26}$$

In the IW-PGPE method, we employ the same linear deterministic policy model (20) as the original PGPE, and the policy parameter $\boldsymbol{\theta}$ is drawn from the Gaussian prior distribution with hyper-parameter $\boldsymbol{\rho} = (\boldsymbol{\eta}, \boldsymbol{\tau})$, where $\boldsymbol{\eta}$ denotes the Gaussian mean vector and $\boldsymbol{\tau}$ denotes the vector consisting of the Gaussian standard deviation of each element.

Through experiments, the IW-PGPE method was demonstrated to be the best performing algorithm in model-free RL approaches (Zhao et al., 2013). The purpose of this paper is to develop a model-based counterpart of PGPE, and demonstrate its usefulness.

3 Model-Based Policy Search

Model-based RL first estimates the transition model and then learns a policy based on the estimated transition model. Because one can draw as many trajectory samples as one wants from the learned transition model without additional sampling costs, the model-based approach can work well if the transition model is accurately estimated (Wang and Dietterich, 2003; Deisenroth and Rasmussen, 2011). In this section, we extend PGPE to a model-based scenario. We then review an existing model estimation method based on the Gaussian process (GP) (Rasmussen and Williams, 2006) and point out its limitations. Finally, we propose to use the state-of-the-art conditional density estimator called *least-squares conditional density estimation* (LSCDE) (Sugiyama et al., 2010b) in the model-based PGPE method.

3.1 Model-Based PGPE (M-PGPE)

PGPE can be extended to a model-based scenario as follows.

1. Collect transition samples $\{(\boldsymbol{s}_m, a_m, \boldsymbol{s}'_m)\}_{m=1}^M$.
2. Obtain a transition-model estimate $\widehat{p}(\boldsymbol{s}'|\boldsymbol{s}, a)$ from $\{(\boldsymbol{s}_m, a_m, \boldsymbol{s}'_m)\}_{m=1}^M$.
3. Initialize hyper-parameter $\boldsymbol{\rho}$.
4. Draw policy parameter $\boldsymbol{\theta}$ from prior distribution $p(\boldsymbol{\theta}|\boldsymbol{\rho})$.
5. Generate many trajectory samples $\{\widetilde{h}_n\}_{n=1}^{\widetilde{N}}$ from $\widehat{p}(\boldsymbol{s}'|\boldsymbol{s}, a)$ and current policy $p(a|\boldsymbol{s}, \boldsymbol{\theta})$.
6. Estimate baseline b and gradient $\nabla_{\boldsymbol{\rho}} \widehat{J}^b(\boldsymbol{\rho})$ from disjoint subsets of $\{\widetilde{h}_n\}_{n=1}^{\widetilde{N}}$.
7. Update hyper-parameter as $\boldsymbol{\rho} \leftarrow \boldsymbol{\rho} + \varepsilon \nabla_{\boldsymbol{\rho}} \widehat{J}^b(\boldsymbol{\rho})$, where $\varepsilon > 0$ denotes the learning rate.

8. Repeat Steps 4–7 until $\boldsymbol{\rho}$ converges.

Below, we consider the problem of approximating the transition probability $p(\boldsymbol{s}'|\boldsymbol{s},a)$ from transition samples $\{(\boldsymbol{s}_m, a_m, \boldsymbol{s}'_m)\}_{m=1}^M$, and review transition model estimation methods.

3.2 Gaussian Process (GP)

Here we review a transition model estimation method based on GP.

In the GP framework, the problem of transition probability estimation is formulated as the regression problem of predicting output $\boldsymbol{s}'$ given input $\boldsymbol{s}$ and a under Gaussian noise:

$$\boldsymbol{s}' = f(\boldsymbol{s}, a) + \varepsilon, \tag{27}$$

where f is an unknown regression function and $\varepsilon \sim \mathcal{N}(0, \sigma_\varepsilon^2)$ is independent Gaussian noise. Then, the GP estimate of the transition probability density $p(\boldsymbol{s}'|\boldsymbol{s},a)$ for an arbitrary test input $\boldsymbol{s}$ and a is obtained as the Gaussian distribution with mean and variance given by

$$\boldsymbol{k}^\top(\boldsymbol{K} + \sigma_\varepsilon^2 \boldsymbol{I}_M)^{-1}\boldsymbol{y} \quad \text{and} \quad k(\boldsymbol{s}, a, \boldsymbol{s}, a) - \boldsymbol{k}^\top(\boldsymbol{K} + \sigma_\varepsilon^2 \boldsymbol{I}_M)^{-1}\boldsymbol{k}, \tag{28}$$

respectively. Here, $\boldsymbol{I}_M$ denotes the M-dimensional identity matrix. $\boldsymbol{k}$ is the M-dimensional vector and $\boldsymbol{K}$ is the $M \times M$ Gram matrix defined by

$$\boldsymbol{k} = \begin{pmatrix} k(\boldsymbol{s}_1, a_1, \boldsymbol{s}, a) \\ \vdots \\ k(\boldsymbol{s}_M, a_M, \boldsymbol{s}, a) \end{pmatrix} \quad \text{and} \quad \boldsymbol{K} = \begin{pmatrix} k(\boldsymbol{s}_1, a_1, \boldsymbol{s}_1, a_1) & \dots & k(\boldsymbol{s}_M, a_M, \boldsymbol{s}_1, a_1) \\ \vdots & \ddots & \vdots \\ k(\boldsymbol{s}_1, a_1, \boldsymbol{s}_M, a_M) & \dots & k(\boldsymbol{s}_M, a_M, \boldsymbol{s}_M, a_M) \end{pmatrix}. \tag{29}$$

$k(\boldsymbol{s}, a, \boldsymbol{s}', a')$ denotes the covariance function, which is, e.g., defined by

$$k(\boldsymbol{s}, a, \boldsymbol{s}', a') = \theta \exp\left(-([\boldsymbol{s}^\top, a] - [\boldsymbol{s}'^\top, a'])\boldsymbol{\Theta}([\boldsymbol{s}^\top, a] - [\boldsymbol{s}'^\top, a'])^\top\right). \tag{30}$$

Here, θ and $\boldsymbol{\Theta}$ are hyper-parameters, and together with the noise variance σ_ε^2, the hyper-parameters are determined by evidence maximization (Rasmussen and Williams, 2006).

As shown above, the GP-based model estimation method requires the strong assumption that the transition probability density $p(\boldsymbol{s}'|\boldsymbol{s},a)$ is Gaussian. That is, GP is non-parametric as a regression method of estimating the conditional mean, but it is parametric (Gaussian) as a conditional density estimator. Such a conditional Gaussian assumption is highly restrictive in RL problems.

3.3 Least-Squares Conditional Density Estimation (LSCDE)

To overcome the restriction of the GP-based model estimation method, we propose to use LSCDE.

Let us model the transition probability $p(\boldsymbol{s}'|\boldsymbol{s}, a)$ by the following linear-in-parameter model:

$$\begin{aligned} q(\boldsymbol{s}'|\boldsymbol{s}, a) &:= \boldsymbol{\alpha}^\top \boldsymbol{\phi}(\boldsymbol{s}, a, \boldsymbol{s}') \\ &= \sum_{m=1}^{M} \alpha_m \exp\left(-\frac{\|\boldsymbol{s} - \boldsymbol{s}_m\|^2}{2\kappa^2}\right) \exp\left(-\frac{(a - a_m)^2}{2\kappa^2}\right) \exp\left(-\frac{\|\boldsymbol{s}' - \boldsymbol{s}'_m\|^2}{2\kappa^2}\right), \end{aligned} \tag{31}$$

where $\boldsymbol{\phi}(\boldsymbol{s}, a, \boldsymbol{s}')$ is the M-dimensional vector of basis functions and $\boldsymbol{\alpha}$ is the M-dimensional vector of parameters[2]. If M is too large, we may reduce the number of basis functions by only using a subset of samples as Gaussian centers. We may use different Gaussian widths for $\boldsymbol{s}$ and a if necessary.

The parameter $\boldsymbol{\alpha}$ in the model (31) is learned so that the following squared error is minimized:

$$G_0(\boldsymbol{\alpha}) := \frac{1}{2} \int_{\mathcal{S}} \int_{\mathcal{A}} \int_{\mathcal{S}} \Big(q(\boldsymbol{s}'|\boldsymbol{s}, a) - p(\boldsymbol{s}'|\boldsymbol{s}, a) \Big)^2 p(\boldsymbol{s}, a) \mathrm{d}\boldsymbol{s} \mathrm{d}a \mathrm{d}\boldsymbol{s}'. \tag{32}$$

This can be expressed as

$$\begin{aligned} G_0(\boldsymbol{\alpha}) &= \frac{1}{2} \int_{\mathcal{S}} \int_{\mathcal{A}} \int_{\mathcal{S}} q(\boldsymbol{s}'|\boldsymbol{s}, a)^2 p(\boldsymbol{s}, a) \mathrm{d}\boldsymbol{s} \mathrm{d}a \mathrm{d}\boldsymbol{s}' \\ &\quad - \int_{\mathcal{S}} \int_{\mathcal{A}} \int_{\mathcal{S}} q(\boldsymbol{s}'|\boldsymbol{s}, a) p(\boldsymbol{s}, a, \boldsymbol{s}') \mathrm{d}\boldsymbol{s} \mathrm{d}a \mathrm{d}\boldsymbol{s}' + C \\ &= \frac{1}{2} \int_{\mathcal{S}} \int_{\mathcal{A}} \int_{\mathcal{S}} \Big(\boldsymbol{\alpha}^\top \boldsymbol{\phi}(\boldsymbol{s}, a, \boldsymbol{s}') \Big)^2 p(\boldsymbol{s}, a) \mathrm{d}\boldsymbol{s} \mathrm{d}a \mathrm{d}\boldsymbol{s}' \\ &\quad - \int_{\mathcal{S}} \int_{\mathcal{A}} \int_{\mathcal{S}} \boldsymbol{\alpha}^\top \boldsymbol{\phi}(\boldsymbol{s}, a, \boldsymbol{s}') p(\boldsymbol{s}, a, \boldsymbol{s}') \mathrm{d}\boldsymbol{s} \mathrm{d}a \mathrm{d}\boldsymbol{s}' + C, \end{aligned} \tag{33}$$

where we used $p(\boldsymbol{s}'|\boldsymbol{s}, a) = p(\boldsymbol{s}, a, \boldsymbol{s}')/p(\boldsymbol{s}, a)$ in the second term and

$$C := \frac{1}{2} \int_{\mathcal{S}} \int_{\mathcal{A}} \int_{\mathcal{S}} p(\boldsymbol{s}'|\boldsymbol{s}, a) p(\boldsymbol{s}, a, \boldsymbol{s}') \mathrm{d}\boldsymbol{s} \mathrm{d}a \mathrm{d}\boldsymbol{s}'. \tag{34}$$

Because C is constant, we only consider the first two terms from here on:

$$\begin{aligned} G(\boldsymbol{\alpha}) &:= G_0(\boldsymbol{\alpha}) - C \\ &= \frac{1}{2} \boldsymbol{\alpha}^\top \boldsymbol{H} \boldsymbol{\alpha} - \boldsymbol{\alpha}^\top \boldsymbol{h}, \end{aligned} \tag{35}$$

[2]Note that LSCDE does not require explicit normalization to be consistent, thanks to the least-squares formulation. This highly contributes to simplifying the optimization problem as shown later.

where

$$\boldsymbol{H} := \int_{\mathcal{A}} \int_{\mathcal{S}} \overline{\boldsymbol{\Phi}}(\boldsymbol{s}, a) p(\boldsymbol{s}, a) \mathrm{d}\boldsymbol{s} \mathrm{d}a \in \mathbb{R}^{M \times M}, \tag{36}$$

$$\boldsymbol{h} := \int_{\mathcal{S}} \int_{\mathcal{A}} \int_{\mathcal{S}} \boldsymbol{\phi}(\boldsymbol{s}, a, \boldsymbol{s}') p(\boldsymbol{s}, a, \boldsymbol{s}') \mathrm{d}\boldsymbol{s} \mathrm{d}a \mathrm{d}\boldsymbol{s}' \in \mathbb{R}^{M}, \tag{37}$$

$$\overline{\boldsymbol{\Phi}}(\boldsymbol{s}, a) := \int_{\mathcal{S}} \boldsymbol{\phi}(\boldsymbol{s}, a, \boldsymbol{s}') \boldsymbol{\phi}(\boldsymbol{s}, a, \boldsymbol{s}')^\top \mathrm{d}\boldsymbol{s}' \in \mathbb{R}^{M \times M}. \tag{38}$$

Note that, for the Gaussian model (31), the (m, m')th element of $\overline{\boldsymbol{\Phi}}(\boldsymbol{s}, a)$ can be computed analytically as

$$\begin{aligned}\overline{\boldsymbol{\Phi}}_{m,m'}(\boldsymbol{s}, a) = (\sqrt{\pi}\kappa)^{\dim(\boldsymbol{s}')} \exp\left(-\frac{\|\boldsymbol{s} - \boldsymbol{s}_m\|^2 + \|\boldsymbol{s} - \boldsymbol{s}_{m'}\|^2}{2\kappa^2}\right) \\ \times \exp\left(-\frac{(a - a_m)^2 + (a - a_{m'})^2}{2\kappa^2}\right) \exp\left(-\frac{\|\boldsymbol{s}'_m - \boldsymbol{s}'_{m'}\|^2}{4\kappa^2}\right).\end{aligned} \tag{39}$$

Because $\boldsymbol{H}$ and $\boldsymbol{h}$ included in $G(\boldsymbol{\alpha})$ contain the expectations over unknown densities $p(\boldsymbol{s}, a)$ and $p(\boldsymbol{s}, a, \boldsymbol{s}')$, they are approximated by sample averages. Then we have

$$\widehat{G}(\boldsymbol{\alpha}) := \frac{1}{2} \boldsymbol{\alpha}^\top \widehat{\boldsymbol{H}} \boldsymbol{\alpha} - \widehat{\boldsymbol{h}}^\top \boldsymbol{\alpha}, \tag{40}$$

where

$$\widehat{\boldsymbol{H}} := \frac{1}{M} \sum_{m=1}^{M} \overline{\boldsymbol{\Phi}}(\boldsymbol{s}_m, a_m) \in \mathbb{R}^{M \times M}, \tag{41}$$

$$\widehat{\boldsymbol{h}} := \frac{1}{M} \sum_{m=1}^{M} \boldsymbol{\phi}(\boldsymbol{s}_m, a_m, \boldsymbol{s}'_m) \in \mathbb{R}^{M}. \tag{42}$$

By adding an ℓ_2-regularizer to $\widehat{G}(\boldsymbol{\alpha})$ to avoid overfitting, the LSCDE optimization criterion is given as

$$\widetilde{\boldsymbol{\alpha}} := \underset{\boldsymbol{\alpha} \in \mathbb{R}^M}{\operatorname{argmin}} \left[\widehat{G}(\boldsymbol{\alpha}) + \frac{\lambda}{2} \|\boldsymbol{\alpha}\|^2\right], \tag{43}$$

where λ (≥ 0) is the regularization parameter. Taking the derivative of the above objective function and equating it to zero, we can see that the solution $\widetilde{\boldsymbol{\alpha}}$ can be obtained just by solving the following system of linear equations:

$$(\widehat{\boldsymbol{H}} + \lambda \boldsymbol{I}_M) \boldsymbol{\alpha} = \widehat{\boldsymbol{h}}, \tag{44}$$

where $\boldsymbol{I}_M$ denotes the M-dimensional identity matrix. Thus, the solution $\widetilde{\boldsymbol{\alpha}}$ is given analytically as

$$\widetilde{\boldsymbol{\alpha}} = (\widehat{\boldsymbol{H}} + \lambda \boldsymbol{I}_M)^{-1} \widehat{\boldsymbol{h}}. \tag{45}$$

Because conditional probability densities are non-negative by definition, we modify the solution $\widetilde{\boldsymbol{\alpha}}$ as

$$\widehat{\boldsymbol{\alpha}} := \max(\mathbf{0}_M, \widetilde{\boldsymbol{\alpha}}), \tag{46}$$

where $\mathbf{0}_M$ denotes the M-dimensional zero vector and 'max' for vectors are applied in the element-wise manner.

Finally, we renormalize the solution in the test phase. More specifically, given a test input point $(\boldsymbol{s}, a)$, the final LSCDE solution is given as

$$\widehat{p}(\boldsymbol{s}'|\boldsymbol{s}, a) = \frac{\widehat{\boldsymbol{\alpha}}^\top \boldsymbol{\phi}(\boldsymbol{s}, a, \boldsymbol{s}')}{\int_{\mathcal{S}} \widehat{\boldsymbol{\alpha}}^\top \boldsymbol{\phi}(\boldsymbol{s}, a, \boldsymbol{s}')\mathrm{d}\boldsymbol{s}'}, \tag{47}$$

where, for the Gaussian model (31), the denominator in Eq.(47) can be analytically computed as

$$\int_{\mathcal{S}} \widehat{\boldsymbol{\alpha}}^\top \boldsymbol{\phi}(\boldsymbol{s}, a, \boldsymbol{s}')\mathrm{d}\boldsymbol{s}' = (\sqrt{2\pi}\kappa)^{\dim(\boldsymbol{s}')} \sum_{m=1}^{M} \alpha_m \exp\left(-\frac{\|\boldsymbol{s} - \boldsymbol{s}_m\|^2 + (a - a_m)^2}{2\kappa^2}\right). \tag{48}$$

LSCDE was proved to achieve the optimal non-parametric convergence rate to the true conditional density in the mini-max sense (Sugiyama et al., 2010b), meaning that no method can outperform this simple LSCDE method asymptotically.

Model selection of the Gaussian width κ and the regularization parameter λ is possible by cross-validation. A MATLAB® implementation of LSCDE is available from

'`http://sugiyama-www.cs.titech.ac.jp/~sugi/software/LSCDE/`'.

4 Experiments

In this section, we demonstrate the usefulness of the proposed method through experiments.

4.1 Continuous Chain Walk

For illustration purposes, let us first consider a simple continuous chain-walk task (Figure 1).

4.1.1 Setup

Let

$$s \in \mathcal{S} = [0, 10], \tag{49}$$

$$a \in \mathcal{A} = [-5, 5], \tag{50}$$

$$R(s, a, s') = \begin{cases} 1 & (4 < s' < 6), \\ 0 & (\text{otherwise}). \end{cases} \tag{51}$$

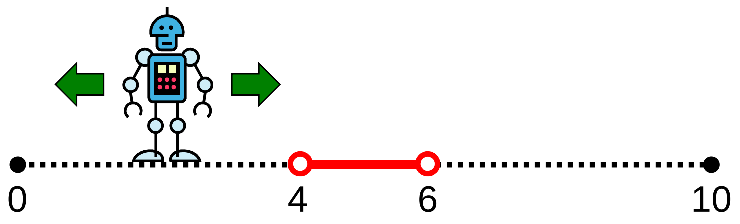


Figure 1: Illustration of continuous chain walk.

That is, the agent receives positive reward $+1$ at the center of the state space. We set the trajectory length at $T = 10$, the discount factor at $\gamma = 0.99$, and the learning rate at $\varepsilon = 0.1$. The initial mean parameter η is set at 0 and initial deviation parameter τ is set at 1. We use the following linear-in-parameter policy model in all the compared methods, i.e., M-PGPE and IW-PGPE:

$$\forall s \in \mathcal{S}, \qquad a = \sum_{i=1}^{6} \theta_i \exp\left(-\frac{(s - c_i)^2}{2}\right), \tag{52}$$

where $(c_1, \ldots, c_6) = (0, 2, 4, 6, 8, 10)$. If an action determined by the above policy is out of the action space, we pull it back to be confined in the domain. Thus, the action space itself is independent of the current position.

As transition dynamics, we consider two setups:

Gaussian: The true transition dynamics is given by

$$\forall s_t \in \mathcal{S}, a_t \in \mathcal{A}, \qquad s_{t+1} = s_t + a_t + \epsilon_t, \tag{53}$$

where ϵ_t is the Gaussian noise with mean 0 and standard deviation 0.3. If the next state is out of the state space, then we project it back to the domain.

Bimodal: The true transition dynamics is given by

$$\forall s_t \in \mathcal{S}, a_t \in \mathcal{A}, \qquad s_{t+1} = s_t \pm a_t + \epsilon_t, \tag{54}$$

where ϵ_t is the Gaussian noise with mean 0 and standard deviation 0.3, and the sign of a_t is randomly chosen with probability $1/2$.

We compare the following three policy search methods:

M-PGPE(LSCDE): The model-based PGPE method with transition model estimated by LSCDE.

M-PGPE(GP): The model-based PGPE method with transition model estimated by GP.

IW-PGPE: The model-free PGPE method with sample reuse by importance weighting[3] (Zhao et al., 2013).

Below, we consider the situation where the budget for data collection is limited to $N = 20$ trajectory samples.

4.1.2 LSCDE VS. GP

When the transition model is learned by the M-PGPE methods, all $N = 20$ trajectory samples are gathered randomly in the beginning at once. More specifically, the initial state s_1 and the action a_1 are chosen from the uniform distributions over $\mathcal{S}$ and $\mathcal{A}$, respectively. Then the next state s_2 and the immediate reward r_1 are obtained. Then the action a_2 is chosen from the uniform distribution over $\mathcal{A}$, and the next state s_3 and the immediate reward r_2 are obtained. This process is repeated until we obtain r_T. This gives a trajectory sample, and we repeat this data generation process N times to obtain N trajectory samples. These $N = 20$ trajectory samples give $M = 200$ transition samples for transition-model learning.

In the implementation of LSCDE, the Gaussian width κ and the regularization parameter λ are selected by cross-validation from the following candidate values:

$$\kappa \in \{0.0316, 0.0487, 0.0750, 0.1155, 0.1778, 0.2738, 0.4217, 0.6494, 1.00\}, \tag{55}$$

$$\lambda \in \{0.0010, 0.0032, 0.0100, 0.0316, 0.1000, 0.31620, 1.0000, 3.1623, 10.0000\}. \tag{56}$$

Figure 2 and Figure 5 illustrate the true transition dynamics and its estimates obtained by LSCDE and GP in the Gaussian and bimodal cases, respectively. The squared error of the estimated transition models in Figure 2(b) and Figure 5(b) is computed by Eq.(35) with the expectations approximated by the averages over test samples:

$$\widehat{G} = \frac{1}{2n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} \int_{\mathcal{S}} q(\boldsymbol{s}'|\boldsymbol{s}_i, a_i)^2 \mathrm{d}\boldsymbol{s}' - \frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} q(\boldsymbol{s}'_i|\boldsymbol{s}_i, a_i), \tag{57}$$

where n_{test} denotes the number of test samples. In the experiments, we set $n_{\text{test}} = 10201$. Note that Figure 2(b) and Figure 5(b) are different in the scale of error values since the constant C is ignored. It is also worth mentioning that the integral in the first term of Eq.(57) can be computed analytically for LSCDE with the Gaussian kernel model and GP.

Figure 2 shows that both LSCDE and GP can learn the entire profile of the true transition dynamics well in the Gaussian case. The squared error of the learned models in Figure 2(b) show that GP provides smaller squared error in this Gaussian case. On the other hand, Figure 5 shows that LSCDE can still successfully capture the entire profile of the true transition model well even in the bimodal case, but GP fails to capture the

[3]We have also tested the plain PGPE method without importance weighting, but this did not perform well in our preliminary experiments. For this reason, we decided to omit the results.

bimodal structure. The squared error shown in Figure 5(b) further illustrates that LSCDE estimates the transition model more accurately than GP in this bimodal case.

Based on the estimated transition models, we learn policies by the M-PGPE method. More specifically, we generate 1000 artificial trajectory samples for policy gradient estimation and another 1000 artificial trajectory samples for baseline estimation from the learned transition model. Then policies are updated based on these artificial trajectory samples. We repeat this policy update step 100 times. For evaluating the return of a learned policy, we use 100 additional test trajectory samples which are not used for policy learning. Figure 3 and Figure 6 depict the average performance of learned policies over 100 runs for the Gaussian and bimodal cases, respectively. The results show that the GP-based method performs very well in the Gaussian case, but LSCDE still exhibits reasonably good performance. In the bimodal case, on the other hand, GP performs poorly and LSCDE gives much better policies than GP. Furthermore, the performance of GP drops after some iterations due to its inaccurate estimation of the bimodal transition model.

Since GP is a Gaussian parametric method as a conditional density estimator, it works very well if the parametric assumption is (approximately) correct (see Figure 2(d)). However, if the parametric model is strongly misspecified, it tends to produce a highly biased solution (see Figure 5(d)). On the other hand, LSCDE is non-parametric and thus it can flexibly approximate *any* conditional densities (see Figure 5(c)). However, its statistical efficiency tends to be lower than parametric approaches; nevertheless, Figure 2(c) shows that LSCDE still performs reasonably well even in the Gaussian case.

4.1.3 Model-Based VS. Model-Free

Next, we compare the performance of M-PGPE with the model-free IW-PGPE method.

For the IW-PGPE method, we need to determine the schedule of collecting 20 trajectory samples under the fixed budget scenario. More specifically, the "sampling schedule" indicates that a small batch of k trajectory samples are gathered $20/k$ times for different k values. $k = 20$ means that all 20 trajectory samples are gathered following the initial data-collecting policy. If $k < 20$, the first batch of k trajectory samples are gathered following the initial data-collecting policy; then the parameter of the current policy is updated and this is used as the data-collecting policy to gather the next batch of k trajectory samples; these parameter update and data collection steps are iterated until the sampling budget runs out.

First, we illustrate how the choice of sampling schedules affects the performance of IW-PGPE. Figure 4 and Figure 7 show expected returns averaged over 100 runs under the sampling schedule that a batch of k trajectory samples are gathered $20/k$ times for different k values. Figure 4 shows that the performance of IW-PGPE depends heavily on the sampling schedule, and gathering $k = 20$ trajectory samples at once is shown to be the best choice in the Gaussian case. Figure 7 shows that gathering $k = 20$ trajectory samples at once is also the best choice in the bimodal case.

Although the best sampling schedule is not accessible in practice, we use this optimal sampling schedule for IW-PGPE. Figure 3 and Figure 6 also include the returns obtained by IW-PGPE averaged over 100 runs as functions of the policy update iterations. These graphs show that IW-PGPE can improve the policies only in the beginning, because all trajectory samples are gathered at once in the beginning. The performance of IW-PGPE may be further improved if it is possible to gather more trajectory samples, but this is prohibited under the fixed budget scenario. On the other hand, return values of M-PGPE constantly increase throughout iterations, because artificial trajectory samples can be kept generated without additional sampling costs. This illustrates a potential advantage of model-based RL methods.

Note that, in our implementation of IW-PGPE, policy update is performed 100 times after observing each batch of trajectory samples, because we empirically observed that this performs better than the conventional way of performing policy update only once (note that no additional sampling costs are necessary during this policy update process). On the other hand, policies are updated only once for each batch of samples in M-PGPE, because we can gather as many samples as we want without additional sampling costs.

4.2 Humanoid Robot Control

Finally, we evaluate the performance of M-PGPE on a practical control problem of a simulated upper-body model of the humanoid robot *CB-i* (Cheng et al., 2007) (see Figure 8(a)). We use its simulator for experiments (see Figure 8(b)). The goal of the control problem is to lead the end-effector of the right arm (right hand) to the target object.

4.2.1 Setup

The simulator is based on the upper-body of the CB-i humanoid robot, which has 9 joints for shoulder pitch, shoulder roll, elbow pitch of the right arm, shoulder pitch, shoulder roll, elbow pitch of the left arm, waist yaw, torso roll, and torso pitch.

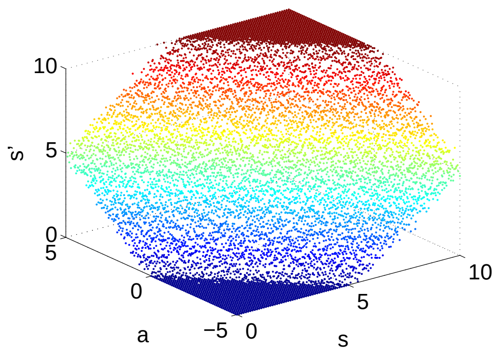
At each time step, the controller receives a state vector from the system and sends out an action vector. The state vector is 18-dimensional and real-valued, which corresponds to the current angle in degree and the current angular velocity for each joint. The action vector is 9-dimensional and real-valued, which corresponds to the target angle of each joint in degree.

At each time step, given a state-action pair, the physical control system calculates torques at each joint by using a proportional-derivative (PD) controller as

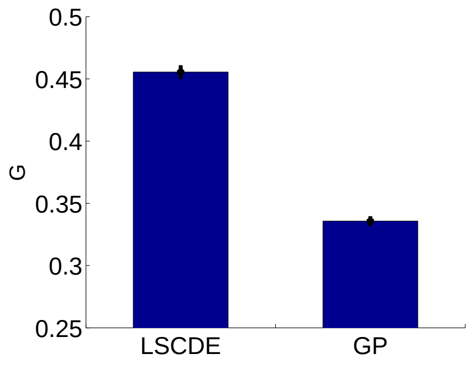
$$\tau_i = K_{p_i}(a_i - s_i) - K_{d_i}\dot{s}_i, \tag{58}$$

where s_i, $\dot{s}_i$, and a_i denote the current angle, current angular velocity, and target angle of the ith joint, respectively. K_{p_i} and K_{d_i} denote the position and velocity gains for the ith joint, respectively. In the experiments, we set $K_{p_i} = 2000$ and $K_{d_i} = 100$ for all joints except the elbow pitch joints for which we set $K_{p_i} = 200$ and $K_{d_i} = 10$.

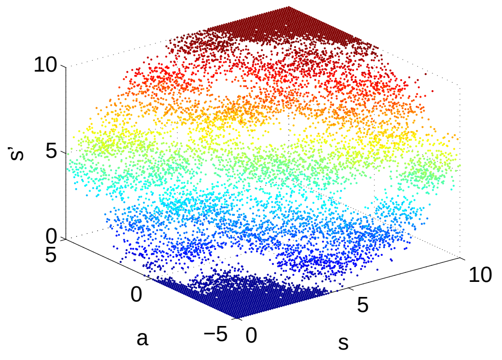
We simulate a noisy control system by perturbing action vectors with independent bimodal Gaussian noise. More specifically, for each action element, we add Gaussian



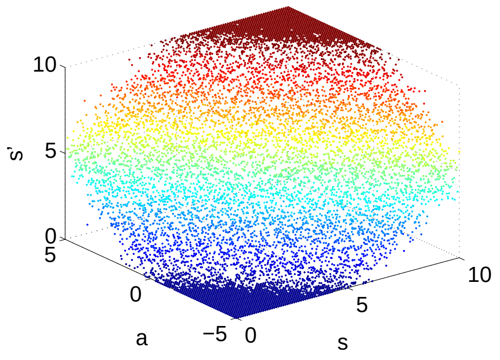
(a) True transition



(b) Squared error



(c) Transition estimated by LSCDE



(d) Transition estimated by GP

Figure 2: Gaussian transition dynamics and its estimates by LSCDE and GP. $\mathrm{argmax}_{s'} p(s'|s,a)$ is plotted as a function of s and a.

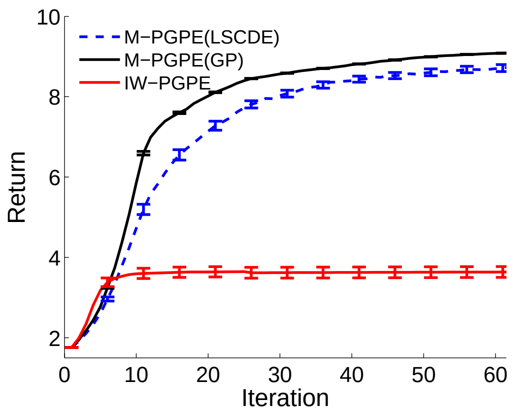


Figure 3: Averages and standard errors of returns of the policies over 100 runs obtained by M-PGPE with LSCDE, M-PGPE with GP, and IW-PGPE for Gaussian transition.

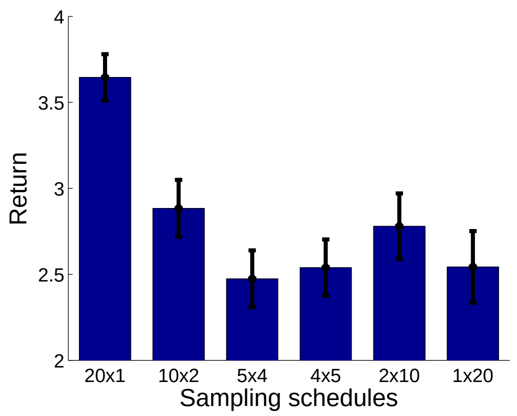
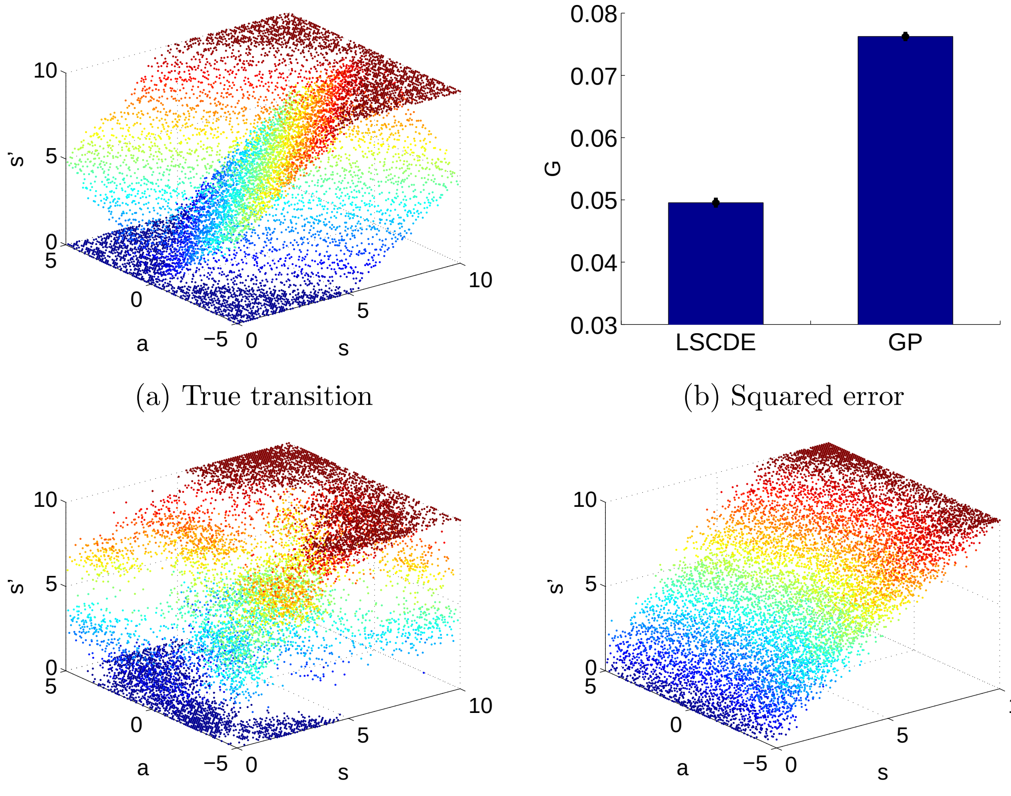


Figure 4: Averages and standard errors of returns obtained by IW-PGPE over 100 runs for Gaussian transition dynamics with different sampling schedules (e.g., 5×4 means gathering $k = 5$ trajectory samples 4 times).



(a) True transition (b) Squared error

(c) Transition estimated by LSCDE (d) Transition estimated by GP

Figure 5: Bimodal transition dynamics and its estimates by LSCDE and GP. $\text{argmax}_{s'} p(s'|s, a)$ is plotted as a function of s and a.

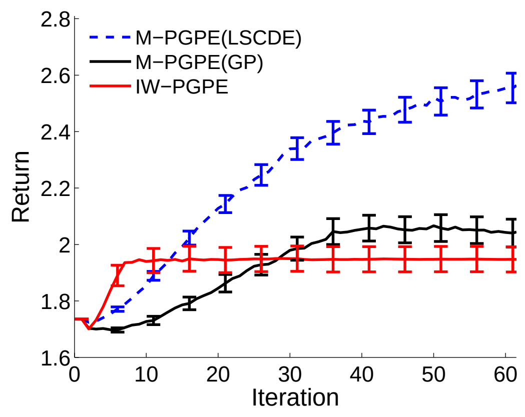


Figure 6: Averages and standard errors of returns of the policies over 100 runs obtained by M-PGPE with LSCDE, M-PGPE with GP, and IW-PGPE for bimodal transition.

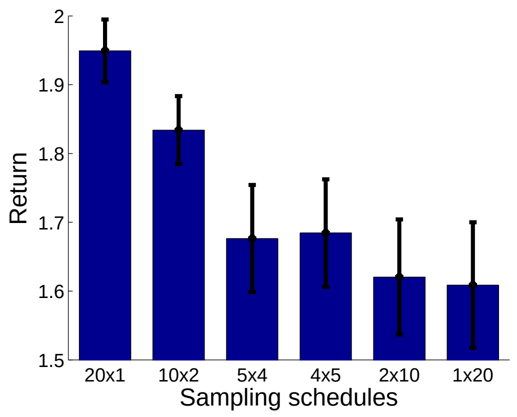
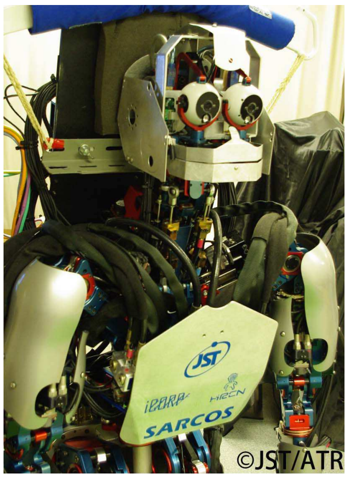


Figure 7: Averages and standard errors of returns obtained by IW-PGPE over 100 runs for bimodal transition with different sampling schedules (e.g., 5×4 means gathering $k = 5$ trajectory samples 4 times).



(a) CB-i

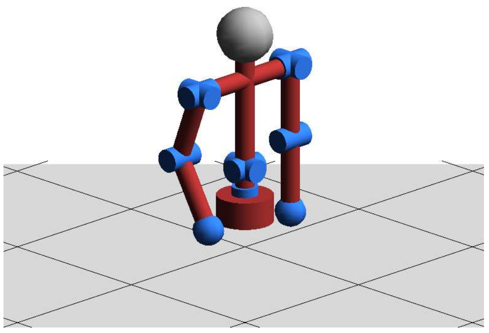

(b) Simulator of the CB-i upper-body

Figure 8: Humanoid robot CB-i and its upper-body model.

noise with mean 0 and standard deviation 3 with probability 0.6, and Gaussian noise with mean -5 and standard deviation 3 with probability 0.4.

The initial posture of the robot is fixed to be standing up straight with arms down. The target object is located in front-above of the right hand, which is reachable by using the controllable joints. The reward function at each time step is defined as

$$r_t = \exp(-10d_t) - 0.000005 \min\{c_t, 1000000\}, \tag{59}$$

where d_t is the distance between the right hand and the target object at time step t. c_t is the sum of control costs for each joint, i.e., $c_t = \sum_{i=1}^{J} \tau_i^2$, where J is the total number of used joints. The coefficient 0.000005 is multiplied to keep the values of the two terms in the same order of magnitude. The deterministic policy model used in M-PGPE and IW-PGPE is defined as Eq.(20). Note that, for this multi-dimensional action problem, the policy for each joint is learned independently. We set the trajectory length at $T = 100$, the discount factor at $\gamma = 0.9$, and the learning rate at $\varepsilon = 0.1/\|\nabla_\rho \hat{J}(\boldsymbol{\rho})\|$.

The initial mean parameter of the Gaussian prior is randomly chosen from a standard normal distribution, i.e., $\eta_0 \sim N(0, 1)$. The initial deviation parameter of the Gaussian prior is set at 1, i.e., $\tau_0 = 1$. Note that we have no prior knowledge of the choice of the initial parameters. Choosing a good initial parameter is especially important for IW-PGPE, since poor initialization may lead to a significant difference between the data collection distribution and the target distribution, and thus IW-PGPE will produce gradient estimators with large variance. On the other hand, M-PGPE does not suffer this problem. In the implementation of LSCDE, the Gaussian width κ and the regularization parameter λ are chosen by cross-validation from the following candidate values:

$$\kappa, \lambda \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10\}. \tag{60}$$

4.2.2 Experiment with 2 Joints

First, we only use 2 joints among the 9 joints, i.e., we allow only the right shoulder pitch and right elbow pitch to be controlled, while the other joints are fixed to the initial posture. This means that 4-dimensional states and 2-dimensional actions are considered here. However, since the simulator calculates 18-dimensional humanoid dynamics, the other joints can slightly move due to movements of the controlled joints. Therefore, to some extent, the learning algorithms are evaluated in the partially observable environment. Under this setup, we compare the performance of M-PGPE(LSCDE), M-PGPE(GP), and IW-PGPE.

We suppose that the budget for data collection is limited to $N = 50$ trajectory samples. For the M-PGPE methods, all trajectory samples are collected at first using the uniformly random initial states and policy. More specifically, the initial state is chosen from the uniform distribution over $\mathcal{S}$. At each time step, the ith element of the action vector, a_i, is chosen from the uniform distribution on $[s_i - 5, s_i + 5]$. In total, we have 5000 transition samples for model estimation. Then, we generate 1000 artificial trajectory samples for policy gradient estimation and another 1000 artificial trajectory samples for baseline estimation from the learned transition model and update the control policy based on these artificial trajectory samples. For the IW-PGPE method, we performed preliminary experiments to determine the optimal sampling schedule (Figure 9), showing that collecting $k = 5$ trajectory samples $50/k$ times yields the highest average return. We use this sampling schedule for performance comparison with the M-PGPE methods. Moreover, the policy update is performed 100 times after observing each batch of samples for the IW-PGPE method, which we confirmed to work better than just updating the policy only once. Thus, policy updates are performed 1000 times in total under this sampling schedule. On the other hand, in M-PGPE, policies are updated only once for each batch of samples, since no additional sampling costs are necessary to gather samples from the learned model.

The returns obtained by each method averaged over 10 runs are plotted in Figure 10, showing that M-PGPE(LSCDE) tends to outperform both M-PGPE(GP) and IW-PGPE. In Figure 10, the IW-PGPE curve is adjusted to have the same horizontal scale as M-PGPE. The top horizontal axis indicates the policy update iterations for IW-PGPE, while the bottom horizontal axis indicates the policy update iterations for M-PGPE.

Figure 11 shows the minimum distance and discounted control cost averaged over 10 runs. The minimum distance plot in Figure 11(a) shows that the learned policies from M-PGPE(LSCDE) and M-PGPE(GP) get closer to the target than that of IW-PGPE, and M-PGPE(LSCDE) gets slighly closer to the target than M-PGPE(GP). The discounted control cost in Figure 11(b) shows that M-PGPE(LSCDE) and M-PGPE(GP) use smaller discounted torques than IW-PGPE to reach the target object.

Figure 12 illustrates an example of the reaching motion with 2 joints obtained by M-PGPE(LSCDE) at the 60th iteration policy. This shows that the learned policy successfully leads the right hand to the target object within only 13 steps in this noisy control system.

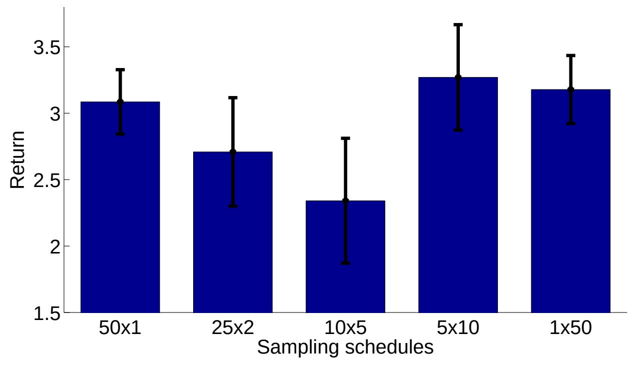


Figure 9: Averages and standard errors of returns obtained by IW-PGPE over 10 runs for the 2-joint humanoid robot simulator for different sampling schedules (e.g., 5×10 means gathering $k = 5$ trajectory samples 10 times).

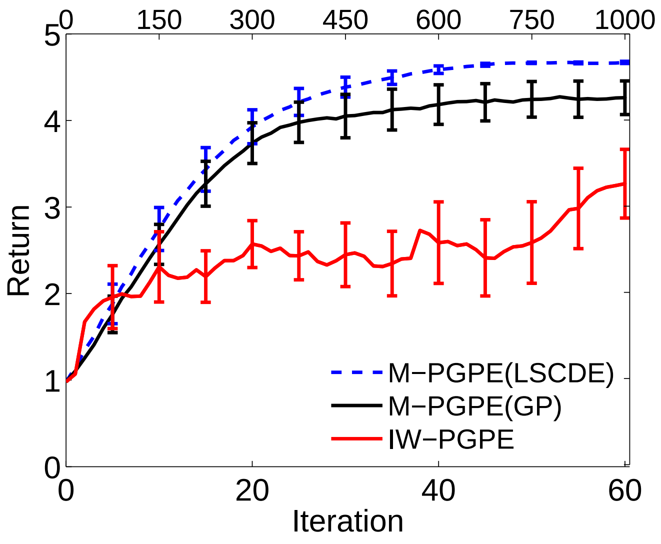
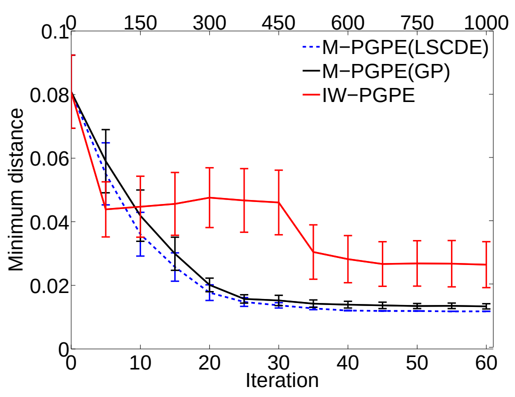
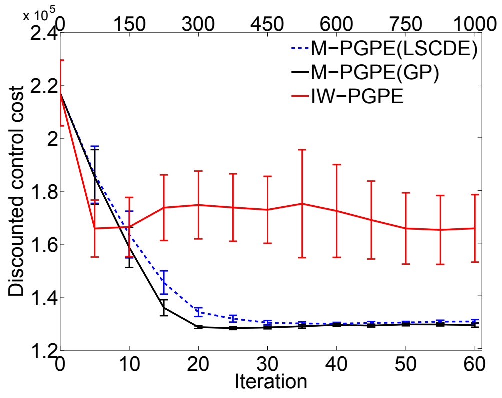


Figure 10: Averages and standard errors of obtained returns over 10 runs for the 2-joint humanoid robot simulator. All methods use 50 trajectory samples for policy learning. In M-PGPE(LSCDE) and M-PGPE(GP), all 50 trajectory samples are gathered in the beginning and the environment model is learned; then 2000 artificial trajectory samples are generated in each update iteration. In IW-PGPE, a batch of 5 trajectory samples are gathered for 10 iterations, which was shown to be the best sampling scheduling (see Figure 9). Note that policy update is performed 100 times after observing each batch of trajectory samples, which we confirmed to perform well. The bottom horizontal axis is for the M-PGPE methods, while the top horizontal axis is for the IW-PGPE method.



(a) Minimum distance



(b) Discounted control cost

Figure 11: Averages and standard errors of the minimum distance and discounted control cost over 10 runs for the 2-joint humanoid robot simulator. The bottom horizontal axis is for the M-PGPE methods, while the top horizontal axis is for the IW-PGPE method.

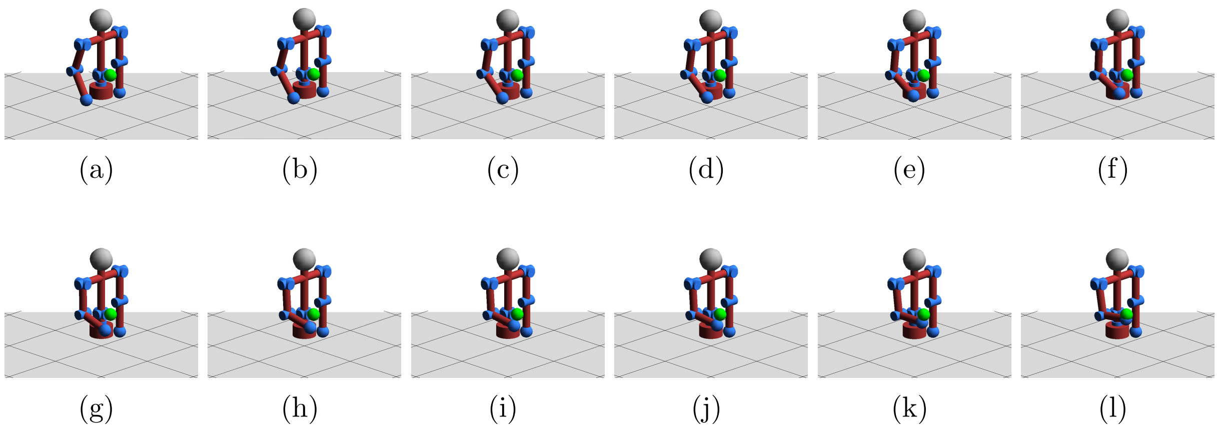


Figure 12: Example of arm reaching with 2 joints using a policy obtained by M-PGPE at the 60th iteration (some intermediate steps are not shown here).

4.2.3 Experiment with 9 Joints

Finally, we evaluate the performance of the proposed method on the reaching task with 9 joints, i.e., all joints are allowed to move. In this experiment, we compare learning performance between M-PGPE(LSCDE) and IW-PGPE. We do not include M-PGPE(GP) since it is outperformed by M-PGPE(LSCDE) on the previous 2-joint experiments, and furthermore the GP-based method requires an enormous amount of computation time.

The experimental setup is essentially the same as the 2-joint experiments, but we have a budget for gathering $N = 1000$ trajectory samples for this complex and high-dimensional task. The position of the target object is moved to far left, which is not reachable only by using 2 joints. Thus, the robot is required to move other joints to reach the object with the right hand. We randomly choose 5000 transition samples for Gaussian centers for M-PGPE(LSCDE). The sampling schedule for IW-PGPE was set at gathering 1000 trajectory samples at once, which is the best sampling schedule according to Figure 13. The returns obtained by M-PGPE(LSCDE) and IW-PGPE averaged over 30 runs are plotted in Figure 14, where the IW-PGPE curve is elongated to have the same horizontal scale as M-PGPE; the top horizontal axis indicates the policy update iterations for IW-PGPE, while the bottom horizontal axis indicates the policy update iterations for M-PGPE. The graph shows that M-PGPE(LSCDE) tends to outperform IW-PGPE in this challenging robot control task.

Figure 15 shows the minimum distance and discounted control cost averaged over 10 runs. The minimum distance plot in Figure 15(a) shows that the learned policies from IW-PGPE get closer to the target object than that of M-PGPE(LSCDE), while the discounted control cost plot in Figure 15(b) shows that M-PGPE(LSCDE) uses much smaller discounted torques than IW-PGPE. Note that, as shown in Figure 14, the return (the weighted discounted combination of the distance and control cost) for M-PGPE(LSCDE) is better than that for IW-PGPE, and either strongly optimizing the distance or control cost depends on the choice of the balancing constant in the definition of the reward function (see Eq.(59)).

Figure 16 shows a typical example of the reaching motion with 9 joints obtained by M-PGPE(LSCDE) at the 1000th iteration. The images show that the policy learned by M-PGPE(LSCDE) leads the right hand to the distant object successfully within 14 steps.

5 Conclusions and Discussions

We extended the model-free PGPE method to a model-based scenario, and proposed to combine it with the model estimator called LSCDE. Under the fixed sampling budget, appropriately designing a sampling schedule is critical for the model-free IW-PGPE method, while this is not a problem for the proposed model-based PGPE method. Through experiments, we confirmed that GP-based model estimation is not as flexible as the LSCDE-based method when the transition model is not Gaussian, and the proposed model-based PGPE based on LSCDE was overall demonstrated to be promising.

We promoted the use of model-based methods in this paper, particularly because we

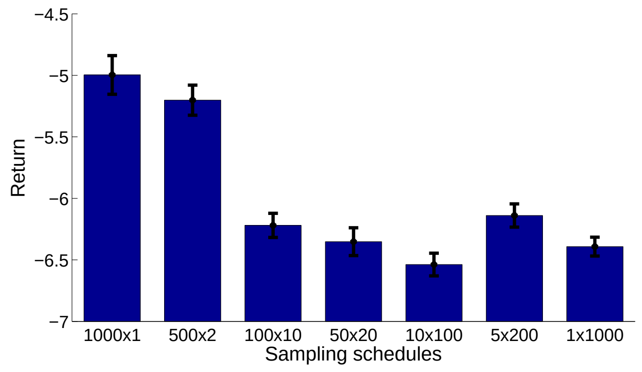


Figure 13: Averages and standard errors of returns obtained by IW-PGPE over 30 runs for the 9-joint humanoid robot simulator for different sampling schedules (e.g., 100×10 means gathering $k = 100$ trajectory samples 10 times).

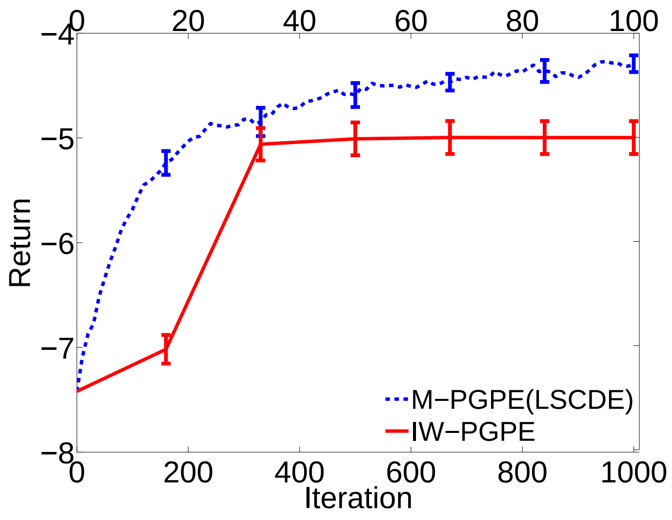
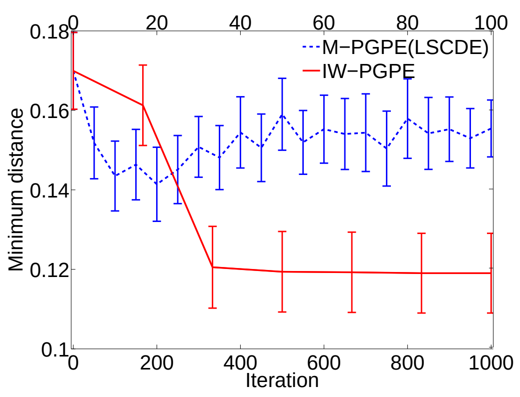
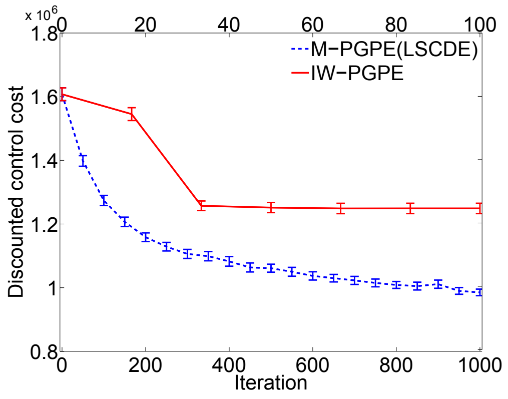


Figure 14: Averages and standard errors of obtained returns over 30 runs for the humanoid robot simulator with 9 joints. Both methods use 1000 trajectory samples for policy learning. In M-PGPE(LSCDE), all 1000 trajectory samples are gathered in the beginning and the environment model is learned; then 2000 artificial trajectory samples are generated in each update iteration. In IW-PGPE, a batch of 1000 trajectory samples are gathered at once, which was shown to be the best scheduling (see Figure 13). Note that policy update is performed 100 times after observing each batch of trajectory samples. The bottom horizontal axis is for the M-PGPE methods, while the top horizontal axis is for the IW-PGPE method.



(a) Minimum distance



(b) Discounted control cost

Figure 15: Averages and standard errors of the minimum distance and discounted control cost over 30 runs for the 9-joint humanoid robot simulator. The bottom horizontal axis is for the M-PGPE methods, while the top horizontal axis is for the IW-PGPE method.

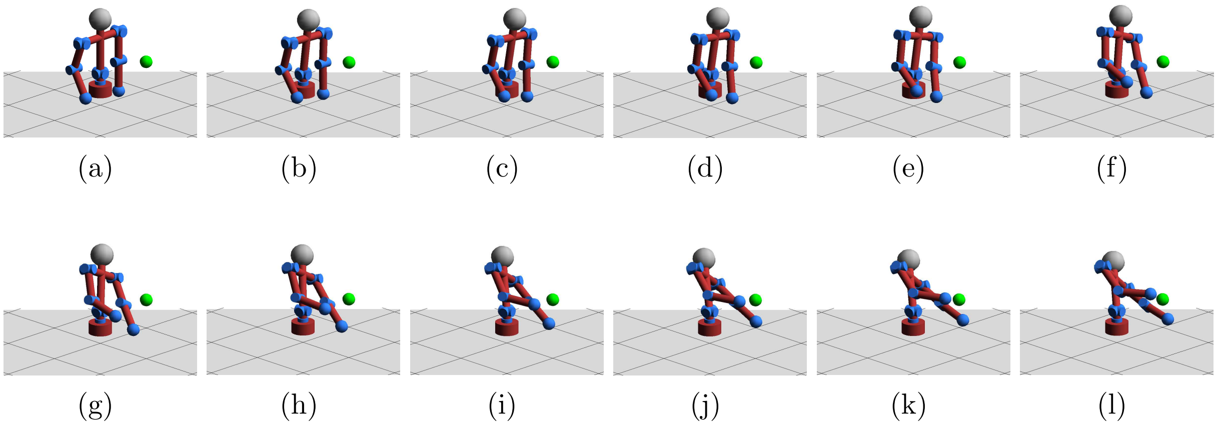


Figure 16: Example of arm reaching with 9 joints using a policy obtained by M-PGPE(LSCDE) at the 1000th iteration (some intermediate steps are not shown here).

do not have to consider the issue of sampling schedules. On the other hand, model-free methods were shown to perform well if prior knowledge in the form of demonstration of the policy is available (Kober and Peters, 2011). In robotics, the model-free approach seems to be more preferred because accurately learning the transition dynamics of complex robots is challenging (Deisenroth et al., 2013), but the model-based approach is advantageous in that no interaction with the real robot is required once the transition model has been learned and the learned transition model can be utilized for further simulation. Actually, choice of either going with model-free or model-based methods is not only an ongoing research theme in machine learning, but also a big debatable topic in neuroscience[4]. Therefore, further discussion would be necessary to more deeply understand pros and cons of the model-based and model-free approaches. Combining (Kupcsik et al., 2013) or switching (Deisenroth et al., 2013) the model-free and model-based approaches are also interesting directions to be further investigated.

There are many practical situations where multimodality is a problem. A well-known example is the mobile robot global localization, where the dynamics of the global localization is usually multimodal due to the symmetry of the environment and ambiguity of detected features (Liu et al., 2007). For example, the robot is placed somewhere in the environment without the knowledge of its global location. After state transition, the robot finds out that the current position is next to a door according to the information from observations and sensors. However, if there are 2 doors in the environment, the distribution of the sensed position of the robot cannot be uniquely determined in general. Multimodality due to such ambiguity is conceivable in the many real-world problems because of imperfect state information. On the other hand, if the robot is highly certain about its position, it would follow a unimodal distribution centered around the true position of the robot (Fox et al., 1999).

Although LSCDE was shown to be promising, it still performs rather poorly in high-dimensional problem (e.g., experiments in Section 4.2.3). A promising approach to coping with this problem is to apply dimension reduction for supervised learning. For example, the use of the information-based method called *least-squares dimension reduction* (Suzuki and Sugiyama, 2013) before estimating the transition model by LSCDE will be investigated in our future work.

In the current formulation of PGPE, the action is assumed to be a scalar, and the policy for each dimension is learned independently for multi-dimensional actions. However, multi-dimensional actions may have some correlation. In our future work, we will therefore extend PGPE to be able to directly handle multi-dimensional actions. The use of multi-task learning (Evgeniou and Pontil, 2004) would be a promising direction.

Acknowledgments

VT was supported by the JASSO scholarship, TZ was supported by the MEXT scholarship, JM was supported by MEXT KAKENHI 23120004, and MS was supported by the

[4]For example, see "`http://decisions.naist.jp/english/index.html`".

FIRST project.

References

P. Abbeel, M. Quigley, and A. Y. Ng. Using inaccurate models in reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning*, pages 1–8, 2006.

T. Akiyama, H. Hachiya, and M. Sugiyama. Efficient exploration through active learning for value function approximation in reinforcement learning. *Neural Networks*, 23(5): 639–648, 2010.

G. Cheng, S. Hyon, J. Morimoto, A. Ude, J. G. Hale, G. Colvin, W. Scroggin, and S. C. Jacobsen. CB: A humanoid research platform for exploring neuroscience. *Advanced Robotics*, 21(10):1097–1114, 2007.

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

M. P. 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*, pages 465–473, 2011.

M. P. Deisenroth, G. Neumann, and J. Peters. A survey on policy search for robotics. *Foundations and Trends in Robotics*, 2(1-2):1–142, 2013.

T. Evgeniou and M. Pontil. Regularized multi-task learning. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD2004)*, pages 109–117. ACM, 2004.

D. Fox, W. Burgard, and S. Thrun. Markov localization for mobile robots in dynamic environments. *Journalof Artificial Intelligence Research*, 11, 1999.

H. Hachiya, T. Akiyama, M. Sugiyama, and J. Peters. Adaptive importance sampling for value function approximation in off-policy reinforcement learning. *Neural Networks*, 22 (10):1399–1410, 2009.

L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

S. Kakade. A natural policy gradient. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, pages 1531–1538, 2002.

T. Kanamori, S. Hido, and M. Sugiyama. A least-squares approach to direct importance estimation. *Journal of Machine Learning Research*, 10:1391–1445, Jul. 2009.

T. Kanamori, T. Suzuki, and M. Sugiyama. Statistical analysis of kernel-based least-squares density-ratio estimation. *Machine Learning*, 86(3):335–367, 2012.

T. Kanamori, T. Suzuki, and M. Sugiyama. Computational complexity of kernel-based density-ratio estimation: A condition number analysis. *Machine Learning*, 90(3):431–460, 2013.

J. Kober and J. Peters. Policy search for motor primitives in robotics. *Machine Learning*, 84(1-2):171–203, July 2011. ISSN 0885-6125.

A.G. Kupcsik, M.P. Deisenroth, J. Peters, and G. Neumann. Data-efficient generalization of robot skills with contextual policy search. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, pages 1401–1407, 2013.

M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

Zhibin Liu, Zongying Shi, Mingguo Zhao, and Wenli Xu. Mobile robots global localization using adaptive dynamic clustered particle filters. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2007*, pages 1059–1064, 2007.

J. Peters and S. Schaal. Policy gradient methods for robotics. In *Processing of the IEEE/RSJ Internatinal Conference on Intelligent Robots and Systems*, pages 2219–2225, 2006.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning.* MIT Press, Cambridge, MA, USA, 2006.

F. Sehnke, C. Osendorfer, T. Rückstiess, A. Graves, J. Peters, and J. Schmidhuber. Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559, 2010.

M. Sugiyama and M. Kawanabe. *Machine Learning in Non-Stationary Environments: Introduction to Covariate Shift Adaptation.* MIT Press, Cambridge, MA, USA, 2012.

M. Sugiyama, H. Hachiya, C. Towell, and S. Vijayakumar. Geodesic Gaussian kernels for value function approximation. *Autonomous Robots*, 25(3):287–304, 2008.

M. Sugiyama, H. Hachiya, H. Kashima, and T. Morimura. Least absolute policy iteration—A robust approach to value function approximation. *IEICE Transactions on Information and Systems*, E93-D(9):2555–2565, 2010a.

M. Sugiyama, I. Takeuchi, T. Suzuki, T. Kanamori, H. Hachiya, and D. Okanohara. Least-squares conditional density estimation. *IEICE Transactions on Information and Systems*, E93-D(3):583–594, 2010b.

M. Sugiyama, T. Suzuki, and T. Kanamori. Density ratio matching under the Bregman divergence: A unified framework of density ratio estimation. *Annals of the Institute of Statistical Mathematics*, 64(5):1009–1044, 2012.

R. S. Sutton and G. A. Barto. *Reinforcement Learning: An Introduction.* MIT Press, Cambridge, MA, USA, 1998.

R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In S.A. Solla, T.K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems 12*, pages 1057–1063, 2000.

T. Suzuki and M. Sugiyama. Sufficient dimension reduction via squared-loss mutual information estimation. *Neural Computation*, 25:725–758, 2013.

X. Wang and T. G. Dietterich. Model-based policy gradient reinforcement learning. In *Proceedings of the 20th International Conference on Machine Learning*, pages 776–783, 2003.

Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 1992.

T. Zhao, H. Hachiya, G. Niu, and M. Sugiyama. Analysis and improvement of policy gradient estimation. *Neural Networks*, 26:118–129, 2012.

T. Zhao, H. Hachiya, V. Tangkaratt, J. Morimoto, and M. Sugiyama. Efficient sample reuse in policy gradients with parameter-based exploration. *Neural Computation*, 25: 1512–1547, 2013.