Variational Policy Search using Sparse Gaussian Process Priors for Learning Multimodal Optimal Actions

Hikaru Sasaki a, *, Takamitsu Matsubara a

a Graduate School of Science and Technology, Division of Information Science, Nara Institute of Science and Technology, 8916-5 Takayamacho, Ikoma, Nara, Japan

Abstract

Policy search reinforcement learning has been drawing much attention as a method of learning a robot control policy. In particular, policy search using such non-parametric policies as Gaussian process regression can learn optimal actions with high-dimensional and redundant sensors as input. However, previous methods implicitly assume that the optimal action becomes unique for each state. This assumption can severely limit such practical applications as robot manipulations since designing a reward function that appears in only one optimal action for complex tasks is difficult. The previous methods might have caused critical performance deterioration because the typical non-parametric policies cannot capture the optimal actions due to their unimodality. We propose novel approaches in non-parametric policy searches with multiple optimal actions and offer two different algorithms commonly based on a sparse Gaussian process prior and variational Bayesian inference. The following are the key ideas: 1) multimodality for capturing multiple optimal actions and 2) mode-seeking for capturing one optimal action by ignoring the others. First, we propose a multimodal sparse Gaussian process policy search that uses multiple overlapped GPs as a prior. Second, we propose a mode-seeking sparse Gaussian process policy search that uses the student-t distribution for a likelihood function. The
effectiveness of those algorithms is demonstrated through applications to object manipulation tasks with multiple optimal actions in simulations.

Keywords: Reinforcement Learning, Policy Search, Gaussian Processes, Multimodality, Mode-seeking

2010 MSC: 00-01, 99-00

1. Introduction

Policy search reinforcement learning using non-parametric policy models is a promising approach for learning continuous robot control from high-dimensional, non-linear, and redundant sensory input. In particular, Gaussian process (or kernel) regression has successfully represented control policies in policy search \[1, 2\]. Since Gaussian processes (GPs) use an implicit (high-dimensional) feature space associated with a kernel function, such high-dimensional sensor input as image features can be directly managed by a reasonable amount of training data without designing human-engineered features, unlike typical policy search methods with parametric policy models (e.g., \[3\]) or deep neural networks that tend to require many samples \[4, 5, 6, 7, 8, 9, 10\]. For example, image-related, high-dimensional (800 dim.) features were successfully used as inputs \[2\].

However, the previously proposed methods implicitly assumed that optimal actions become unique for each state. This assumption can severely limit to such practical applications as robot manipulations since designing a reward function that appears only one optimal action for each state is often complicated for complex tasks. A reward function that appears multiple optimal actions may lead to poor performance in the previous methods. Multiple optimal actions may appear at particular states, although typical non-parametric policies cannot be captured due to their unimodality.

As an illustrative example of a robotic task, we consider a hand-posture adjustment task by a robotic arm (Fig. 1a). The objective is to grasp the object by rotating the robot’s wrist. We simply designed the reward function so that a positive value is given when the robot can grab and lift the object. Such
Figure 1: Illustrative example of robotic tasks with multiple optimal actions: (a) hand-posture adjustment task environment and (b) red solid lines indicate multiple optimal actions. Blue broken line and region indicate mean and standard deviation of learned policy with a unimodal policy model, which cannot capture optimal actions.

Figure 2: Two proposed methods for multiple optimal actions: (a) multimodal SGP-PS that can capture multimodality in optimal action using multiple models and (b) mode-seeking SGP-PS that can capture one optimal action and ignore others.

A reward function appears in multiple optimal actions at each state since the robot grabs the object from multiple wrist angles. Due to the multiple optimal actions, previous methods that assume unimodality in their policy model choose incorrect actions (Fig. 1b). Of course, for such simple tasks, we could elaborate the reward by adding additional terms. However, designing a reward function that appears only one optimal action is challenging when dealing with more complex tasks.

To alleviate this limitation, we propose novel approaches in a non-parametric policy search with multiple optimal actions and offer two different algorithms:
a multimodal sparse Gaussian process policy search (multimodal SGP-PS) and
a mode-seeking sparse Gaussian process policy search (mode-seeking SGP-PS).
Both methods employ sparse Gaussian processes \[11, 12\] as a prior of a policy
model to determine a robot action and derive update laws based on variational
Bayesian inference \[13, 14, 15\]. We introduce the following key ideas: 1) multi-
modality for capturing multiple optimal actions (Fig. 2a) and 2) mode-seeking
for capturing one optimal action by ignoring the others (Fig. 2b). Multimodal
SGP-PS employs a multimodal policy prior inspired by a recent extension of
Gaussian processes \[16\]. This prior distribution assumes that each component
is responsible for a global function over the common input space. Mode-seeking
SGP-PS employs a student-t distribution with outlier robustness as a likelihood
function, which facilitates the learned policy to capture one of the optimal ac-
tions by treating the others as outliers to be ignored. For deriving reasonable
policy update schemes, we use scale mixture representation to the student-t
distribution \[17, 18\].

We validated the effectiveness of these algorithms through robotic manip-
ulation tasks with multiple optimal actions in simulations: 1) hand-posture
adjustment task using a robot simulator, V-REP \[19\] and 2) table-sweeping
task using MuJoCo simulator \[20\]. The results of the hand-posture adjustment
task demonstrate that our methods can efficiently learn optimal actions even
with multiple optimal actions that previous methods cannot. The multimodal
SGP-PS captures multiple optimal actions with the multimodal policy model.
The mode-seeking SGP-PS learns an effective unimodal policy by seeking an
optimal action. In the table-sweeping task, we confirmed the performance of
our methods for more challenging situations.

Although a preliminary version of this paper was previously published \[21\],
it only focused on an approach with multimodality for alleviating the GP-PS
limitations and provided limited numerical verifications. Our paper significantly
expanded and generalized that previous paper with a mode-seeking approach.
We thoroughly investigated its effectiveness by comprehensively analysing the
experimental data, compared it with the neural-network-based methods, and
discussed the remaining issues and future directions.

The remainder of this paper is structured as follows. Section 2 summarizes related works. Section 3 presents our proposed methods, and Section 4 presents the simulation setup and results. Then several limitations and potential future works are discussed in Section 5. Finally, Section 6 concludes this paper.

2. Related Work

In this section, we describe related works in the following three categories: 1) non-parametric policy search, 2) multimodal reinforcement learning, and 3) robust reinforcement learning.

2.1. Non-parametric policy search

Non-parametric policy search uses a non-parametric model as a policy model. Bagnell et al. [22] embedded the function of the policy model in a reproducing kernel Hilbert space (RKHS) and used the policy gradient method to maximize the expected return, making it possible to learn non-parametric policies. However, that approach can only learn a policy with a discrete action. Therefore, a method was proposed for learning continuous policies by assuming a Gaussian distribution in the policy model, embedding the mean function in RKHS, and learning the policy gradient method [23]. For robot applications, the Cost-regularized Kernel Regression [1] method was applied to a ball-hitting task in table tennis. To avoid the difficulty of convergence in a gradient-based policy search, an EM-inspired non-parametric policy search was applied to a door-opening task [24]. A data efficient non-parametric policy search was proposed [2] that can learn a policy with a high-dimensional features.

Although the above studies focused on optimizing such policy models with high representation capability, scant attention was given to the relationship between the reward function design and the limitations of unimodality in the policy. A reward function that generates multiple optimal actions may degrade the performance in those methods. On the other hand, we focused on this
relationship and developed algorithms that can alleviate the limitations when the tackled task has multiple optimal actions.

2.2. Multimodal reinforcement learning

A multimodal policy search aims to learn a policy that can capture the multimodality of optimal actions. Many previous policy search-based methods employ a hierarchical policy model that consists of low-level policies that determine robot action and a high-level policy that determines which low-level policy is used [25, 26, 27, 28]. However, the proposed approaches employ such hierarchical parametric models as 1) a softmax gating function with linear Gaussian sub-policies or 2) Gaussian mixture models with a parametric policy model, both of which require hand-engineered features to cope with high-dimensional sensor input, unlike non-parametric methods.

Multimodality in optimal action has also been considered in neural network-based RL methods. For example, soft Q-learning learns a value function to acquire diverse behaviors by introducing an entropy term in the Bellman equation that promotes capturing multimodality in policies [29]. Soft actor-critic (SAC) is an extension of soft Q-learning; the learning performance is greatly improved [30, 31]. In SAC, the value function captures various behaviors, whereas the policy employs a unimodal model. Thus, it cannot explicitly capture multimodality in optimal actions. SAC-GMM with a GMM policy was also explored as an early version of SAC, but its performance is worse than SAC due to the algorithmic complexity [32]. Kalashnikov et al. proposed another method that selects an action from an action-value function using the cross-entropy method [33].

We propose a multimodal SGP-PS that employs a policy model inspired by the overlapping mixtures of Gaussian processes (OMGP) [16]. Multimodal SGP-PS learns a policy whose components in the mixture are global and overlapping in state space, unlike a typical mixture model that dictates the input space for capturing non-stationarity [34, 35, 36, 37]. We incorporate this feature into the policy search. We also employ a sparse Gaussian process as a prior
distribution for each low-level policy. Such a simpler structure with a non-parametric prior allows us to derive policy update algorithms which can cope with high-dimensional sensor input without requiring hand-engineered features.

2.3. Robust reinforcement learning

Robust reinforcement learning acquires a robust policy for several kinds of noise. Recent research has attracted attention by focusing on the errors between simulations and real environments in sim-to-real domains. Reinforcement learning (robust to state noise [38]) and robust reinforcement learning (robust to disturbances in actual environments) have been proposed [39]. In research that focuses on reward functions, methods have been proposed that learn reward functions from human demonstrations and action evaluations [40, 41] and reinforcement learning robust to noise in rewards [42, 43].

Other methods have been proposed for capturing one optimal action even when multiple optimal actions exist. Trust region policy optimization (TRPO) can seek optimal actions by conservatively updating policies [44]. In this method, a conservative policy update retards policy updates due to a characteristic of learning stabilization. A deep deterministic policy gradient (DDPG) also focuses on choosing one optimal action using a deterministic policy model [4]. These methods are implicit mode-seeking policy search methods. For them, it is commonly necessary to manually design many parameters in such neural network models as network structure and hyperparameters for each task.

We propose a mode-seeking SGP-PS, which effectively solves the issue of the multimodality of optimal actions. Using student-t distribution as a likelihood, a mode-seeking SGP-PS captures one optimal action. A non-parametric-based policy model enables efficient policy learning with few task-specific hyperparameters even with a small number of data samples.

3. Proposed Method

In this section, we derive two different non-parametric policy search algorithms: a multimodal SGP-PS and a mode-seeking SGP-PS. Each method is
derived by extending a sparse Gaussian process policy search (SGP-PS), which is a policy search that uses SGP as a policy prior.

3.1. Sparse Gaussian process policy search (SGP-PS)

The objective of policy search reinforcement learning is to learn the policy that maximizes the expected return by repeating the data collection using the current policy and improving the policy using the collected data under a Markov decision process. The environment consists of state $s_t$, action $a_t$, reward $r_t$, state transition probability $p(s_{t+1} \mid s_t, a_t)$, and initial state probability $p(s_1)$. The agent selects an action by following the current policy $\pi(a_t \mid s_t)$. After executing $T$ steps of action selection with $\pi$, the agent obtains trajectory $d = \{s_1, a_1, \cdots, s_{T-1}, a_{T-1}, s_T\}$, and its probability $p_\pi(d)$ is described:

$$p_\pi(d) = p(s_1) \prod_{t=1}^{T} \pi(a_t \mid s_t)p(s_{t+1} \mid s_t, a_t). \quad (1)$$

Return function $R(d)$ is defined as the sum of reward $r_t$ which is obtained at time step $t$ as $R(d) = \sum_{t=1}^{T} r_t$. The expected return is denoted by $J(\pi)$:

$$J(\pi) = \int R(d)p_\pi(d)dd. \quad (2)$$

Finally, the policy search goal is formulated:

$$\pi^* \leftarrow \arg \max_\pi J(\pi). \quad (3)$$

We derive a policy search that optimizes the SGP policy model by maximizing a lower bound on the expected return. We denote a policy model that determines the action from the state as $p(a_t \mid s_t) = N(a_t \mid f(s_t), \sigma^2)$, where $f$ is the mean function of the policy. We place a GP prior on mean function $f$:

$$f \sim GP(\mathbf{0}, k(s, s')) \quad (4)$$

where $k(s, s')$ is a kernel function.

As preparation for a policy search, we reduce the computational complexity by augmenting the prior distribution by introducing common pseudo inputs.
\( \tilde{S} = \{s_i\}_{i=1}^L \) and corresponding pseudo outputs \( \tilde{f} = \{\tilde{f}_i\}_{i=1}^L \). We place a GP prior on the pseudo outputs:

\[
p(\tilde{f} \mid \tilde{S}) = \mathcal{N}(\tilde{f} \mid 0, K_{\tilde{S}}),
\]

where \( K_{\tilde{S}} \) is the kernel gram matrix computed with \( \tilde{S} \). A prior distribution of \( f_t = f(s_t) \) is augmented:

\[
p(f_t \mid \tilde{f}, \tilde{S}, s_t) = \mathcal{N}(f_t \mid K_{s_t, \tilde{S}} K_{\tilde{S}}^{-1} \tilde{f}, \lambda_t),
\]

where \( \lambda_t = K_{s_t, \tilde{S}} K_{\tilde{S}}^{-1} K_{s_t, \tilde{S}} \), \( K_{s_t, \tilde{S}} = k(s_t, \tilde{S}) \).

To derive a policy search algorithm, we define the probability of the trajectory using a SGP policy model:

\[
p(d, f, \tilde{f} \mid \tilde{S}) =
\]

\[
p(s_1) p(\tilde{f} \mid \tilde{S}) \prod_{t=1}^T p(a_t \mid f_t) p(f_t \mid \tilde{f}, S, s_t) p(s_{t+1} \mid s_t, a_t),
\]

where \( f = \{f_i\}_{i=1}^T \). For clarity, we omit the conditioning on \( S \) and \( \tilde{S} \) in the remainder of this section. The expected return is defined:

\[
J(\theta) = \int R(d)p(d, f, \tilde{f})d.ddf.dd\tilde{f}.
\]

The goal of a policy search is to find the hyperparameters of policy \( \theta \) to maximize the expected return. However, solving this optimization is difficult due to the analytical intractability of the integral. We derive the lower bound of expected return \( \log J_L(\theta) \) from Jensen’s inequality by introducing variational distribution \( q(d, f, \tilde{f}) \):

\[
\log \frac{J(\theta)}{J_{old}} = \log \int \frac{R(d) q(d, f, \tilde{f})}{J_{old}} \frac{p(d, f, \tilde{f})d.ddf.dd\tilde{f}}{q(d, f, \tilde{f})}
\]

\[
\geq \int \frac{R(d) q(d, f, \tilde{f})}{J_{old}} \frac{p(a \mid f)p(f \mid \tilde{f})p(\tilde{f})}{q(d, f, \tilde{f})}d.ddf.dd\tilde{f}
\]

\[
\equiv \log J_L(\theta, q),
\]
where $J_{\text{old}}$ is the expected return using sampler $p_{\text{old}}$ and $p_{\text{old}}$ is a sampler of the training data using the previous policy. We introduce variational distribution $q(d, f, \bar{f}) = p_{\text{old}}(d)p(f \mid \bar{f})p(\bar{f})$ and apply Monte Carlo approximation regarding expectation w.r.t. $p_{\text{old}}(d)$, and the lower bound becomes:

$$
\log J_L(\theta, q) \approx \int p(f \mid \bar{f})q(\bar{f}) \log \frac{p(\bar{a} \mid f)p(\bar{f})}{q(\bar{f})} df d\bar{f},
$$

(10)

where $p(\bar{a} \mid f)$ is the following return weighted likelihood using Gaussian distribution: $p(\bar{a} \mid f) = \mathcal{N}(\bar{a} \mid Wf, \sigma^2 I)$. $W$ is a weight based on the return express, $W = \text{diag}\left\{ \sqrt{\frac{R(\bar{d}_1) J_{\text{old}} E_1}, \ldots, \sqrt{\frac{R(\bar{d}_E) J_{\text{old}} E_1}} \right\}$, $d^e$ is $e$-th trajectory, and $\bar{a}$ is return weighted action samples, $\bar{a} = Wa$. The lower bound of the expected return (Eq. 10) is the lower bound of the return weighted marginal likelihood.

We derive an EM-like policy update law that maximizes the lower bound of expected return $J_L$ by alternately optimizing variational distribution $q(\bar{f})$, parameters $\theta$ by variational Bayesian inference.

The action selection for new state $s_*$ is determined by the predicted distribution of the SGP policy model using variational distribution $q(\bar{f})$ and hyperparameter $\theta$:

$$
p(a_* \mid s_*) \approx \int p(a_* \mid f_*)p(f_* \mid \bar{f}, s_*)q(\bar{f})df_*d\bar{f}.
$$

(11)

3.2. Multimodal sparse Gaussian process policy search (multimodal SGP-PS)

Next we derive the multimodal SGP-PS based on SGP-PS. To capture multiple optimal actions, we consider the following product of $M$ different GPs as a prior of function $f$ of the control policy:

$$
f \sim \alpha \prod_{m=1}^{M} \mathcal{GP}^{(m)}(0, k^{(m)}(s, s')).
$$

(12)

We assume a multimodality control policy that uses $M$ GPs and expresses the lower bound of the return weighted likelihood product of the $M$ lower bounds:

$$
\log J_L(\theta_M, q) \approx \prod_{m=1}^{M} \int p(f^{(m)} \mid \bar{f}^{(m)})q(\bar{f}^{(m)}) \log \frac{p(\bar{a} \mid f^{(m)})p(\bar{f}^{(m)})}{q(\bar{f}^{(m)})} df^{(m)}d\bar{f}^{(m)},
$$

(13)
where $\bar{f}^{(m)}$ and $f^{(m)}$ indicate the pseudo outputs and the function outputs of the $m$-th SGP. Optimizing the above lower bound w.r.t. policy parameters is, however, insufficient to search for multimodal behaviors; all the $M$ GPs tend to converge to the same solution since all the training data are commonly shared to train all the GPs. To allow exploration of different solutions for all the GPs, we introduce another latent variable, the so-called binary indicator matrix $Z$ whose $nm$ element is a binary variable that is associated with $m$-th GP [14]. Each row in $Z$ has one non-zero entry. We assume the prior on binary indicator matrix $Z$:

$$p(Z) = \prod_{n=1}^{N} \prod_{m=1}^{M} \Pi_{nm}^{Z_{nm}},$$

(14)

where $\Pi$ is the $N \times M$ probability matrix and $\Pi_{nm}$ indicates the probability that $a_n$ is generated by the $m$-th GP. Moreover, we extend the return weighted likelihood function:

$$p(\tilde{a} | \{f^{(m)}\}) = \int p(Z)p(\tilde{a} | \{f^{(m)}\}, Z)dz,$$

(15)

$$p(\tilde{a} | \{f^{(m)}\}, Z) = \prod_{n=1}^{N} \prod_{m=1}^{M} \mathcal{N}(\tilde{a}_n | W_{nn}f^{(m)}, \sigma^2)^{Z_{nm}}.$$

(16)

Due to the analytical intractability of the integral of $Z$, we apply Jensen’s inequality:

$$\log J'_{\mathcal{L}}(\theta_M, q) = \int p(\{f^{(m)}\} | \{\bar{f}^{(m)}\})q(\{\bar{f}^{(m)}\})q(Z).$$

$$\log \frac{p(\tilde{a} | \{f^{(m)}\}, Z)p(\{\bar{f}^{(m)}\})p(Z)}{q(\{f^{(m)}\})q(Z)}dzd\{f^{(m)}\}d\{\bar{f}^{(m)}\}dZ,$$

(17)

where $q(\{\bar{f}^{(m)}\}) = \prod_{m=1}^{M} q(\bar{f}^{(m)})$, $p(\{\bar{f}^{(m)}\}) = \prod_{m=1}^{M} p(\bar{f}^{(m)})$, and $p(\{f^{(m)}\} | \{\bar{f}^{(m)}\}) = \prod_{m=1}^{M} p(f^{(m)} | \bar{f}^{(m)})$.

Finally, we derive an EM-like iterative optimization scheme. We repeat the E-step that finds optimal variational distribution $q(\{\bar{f}^{(m)}\})$ and $q(Z)$ and the M-step that finds optimal parameter $\theta_M$ by alternatively maximizing $\log J'_{\mathcal{L}}$. 

11
Algorithm 1 Multimodal SGP-PS algorithm

1: Initialize $\theta_M, S$
2: while reward not converged do
3:   # Sample E trajectories
4:   for $e = 1 : E$ do
5:     Sample trajectory $d$ using policy $\pi$
6:   end for
7:   # Policy Improvement
8:   while $\log J_L'$ is not converged do
9:     # E step
10:    while $\log J_L'$ is not converged do
11:       Update $q(\{\bar{f}(m)\})$ with Eq. (A.2)
12:       Update $q(Z)$ with Eq. (A.5)
13:    end while
14:   # M step
15:   $\theta_M \leftarrow \arg \max \log J_L'$
16: end while
17: end while

We describe the derivation of the analytical update laws of the variational distributions and the analytical form of the objective function in Appendix A. A summary of the multimodal SGP-PS is given in Algorithm 1. Table 1 shows the computational complexity of each optimization, and the complexity of $q(\{\bar{f}(m)\})$ and $\log J_L'$ is reduced due to the pseudo inputs. Although the computational complexity of $q(Z)$ is increased, the overall complexity is reduced.

We analytically obtain the $M$ predictive distribution using variational distribution $q(\{\bar{f}(m)\})$ and parameter $\theta_M$ variational learning. The $m$-th predictive distribution of action $a_s^{(m)}$ corresponds to new state $s_*$:

$$p(a_s^{(m)} | s_*) \approx \int p(a_s^{(m)} | f_*) p(f_s | \bar{f}(m), s_*) q(\bar{f}(m)) df_s d\bar{f}(m).$$  \hspace{1cm} (18)
Table 1: Computational complexity of each optimization in multimodal SGP-PS: \(N\), \(M\), and \(L\) are amounts of training data, components, and pseudo inputs.

| Policy                        | \(q(\{F^{(m)}\})\) | \(q(Z)\) | \(\log J'_L\) |
|-------------------------------|----------------------|----------|---------------|
| Multimodal SGP-PS             | \(O(MNL^2)\)         | \(O(MNL^2)\) | \(O(MNL^2)\) |

Policy search is described in Appendix B.

We employ a softmax function that uses the negative variance of the predictive distribution:

\[
p(m) = \frac{\exp(-\sigma^*_m/\beta)}{\sum_{m=1}^{M} \exp(-\sigma^*_m/\beta)},
\]

where \(p(m)\) is the probability of using the \(m\)-th predictive distribution and \(\beta\) is a temperature parameter that controls the trade-off between exploration and exploitation.

### 3.3. Mode-seeking sparse Gaussian process policy search (mode-seeking SGP-PS)

In this section, we derive a mode-seeking sparse Gaussian process policy search based on SGP-PS. Mode-seeking means that the policy can capture an optimal action at each state by ignoring other optimal actions. To capture one optimal action, the mode-seeking SGP-PS employs student-t distribution as a likelihood and estimates the reliability of each data to ignore low-reliability data. The following is the probabilistic density function (PDF) of the student-t distribution:

\[
St(x \mid \mu, a, b) = \frac{b^a}{\Gamma(a)\sqrt{2\pi}} \left[ b + \frac{(x - \mu)^2}{2} \right]^{-a-1/2} \Gamma(a + 1/2),
\]

where \(\Gamma(\cdot)\) is a gamma function. Although the PDF is complicated, it can be described more simply by a scale mixture representation that uses Gaussian and
gamma distributions:

\[
\text{St}(x \mid \mu, a, b) = \int_0^\infty \mathcal{N}(x \mid \mu, \tau^{-1})\text{Gam}(\tau \mid a, b) d\tau, \quad (21)
\]

\[
\text{Gam}(\tau \mid a, b) = \frac{b^a}{\Gamma(a)} \tau^{a-1} \exp(-b\tau), \quad (22)
\]

where \(\tau\) is the precision of the Gaussian distribution and indicates the data reliability. We use gamma distribution for the precision \(\tau\) of the Gaussian distribution.

To learn a policy that captures one optimal action, we use the student-t distribution as the likelihood:

\[
p(\tilde{a} \mid f) = \text{St}(\tilde{a} \mid f) = \int p(\tilde{a} \mid f, T)p(T) dT, \quad (23)
\]

\[
p(\tilde{a} \mid f, T) = \prod_{n=1}^N \mathcal{N}(\tilde{a}_n \mid f_n, \tau_n^{-1}), \quad (24)
\]

\[
p(T) = \prod_{n=1}^N \text{Gam}(\tau_n \mid a, b), \quad (25)
\]

where \(T = \text{diag}\{\tau_1, \cdots, \tau_N\}\). The integration w.r.t \(f\) in Eq. (10) cannot be solved analytically since the likelihood is complicated. By introducing a scale mixture representation and variational distribution \(q(T)\) of the data reliability, we derive a new lower bound for the mode-seeking SGP-PS:

\[
\log J'_L(\theta_R, q) = 
\int p(f \mid \bar{f})q(\bar{f})q(T) \log \frac{p(\tilde{a} \mid f, T)p(T)p(\bar{f})q(\bar{f})q(T)}{q(\bar{f})q(T)} d\bar{f} d\bar{f} dT. \quad (26)
\]

We assume that \(q(T) = \prod_{n=1}^N q(\tau_n)\). We derive the update laws of \(q(\bar{f})\) and \(q(\tau_n)\) which maximize the lower bound of the expected return based on variational Bayesian inference. The analytical solution of the update laws and the lower bound are described in Appendix C. Alternately updating the variational distribution and the hyperparameters alternately, the mode-seeking SGP policy is learned. This method’s algorithm is described in Alg. 2. Table 2 shows the computational complexity of each optimization, and the complexity of \(q(\{\bar{f}^{(m)}\})\).
Algorithm 2 Mode-seeking SGP-PS algorithm

1: Initialize $\theta_R, S$
2: while reward not converged do
3:     # Sample $E$ trajectories
4:     for $e = 1 : E$ do
5:         Sample trajectory $d$ using policy $\pi$
6:     end for
7:     # Policy Improvement
8:     while $\log J'_L$ is not converged do
9:         # E step
10:            while $\log J'_L$ is not converged do
11:                Update $q(\bar{f})$ with Eq. (C.3)
12:                Update $q(T)$ with Eq. (C.2)
13:            end while
14:         # M step
15:         $\theta_R \leftarrow \arg \max_{\theta_R} \log J'_L$
16:     end while
17: end while

and $\log J'_L$ is reduced by the pseudo inputs. Although the computation amount of $q(T)$ increased, the overall complexity was reduced.

We analytically obtained the predictive distribution of the mode-seeking SGP policy for new state $s_*$ using variational distribution $q(\bar{f}), q(T)$, and parameters $\theta_R$, which are obtained in the learning:

$$p(a_* | s_*) \approx \int p(a_* | f_*, \tau)p(f_* | \bar{f}, s_*)q(\bar{f})q(\tau)d\bar{f}_s d\tau.$$

The analytical solution of the predictive distribution of mode-seeking SGP-PS is described in Appendix D.
Table 2: Computational complexity of each optimization in mode-seeking SGP-PS: \( N \) and \( L \) are amounts of training data and pseudo inputs.

| Optimization                  | \( q(\bar{f}(m)) \) | \( q(T) \) | \( \log J'_{L} \) |
|------------------------------|----------------------|-------------|-------------------|
| Mode-seeking SGP-PS          | \( O(NL^2) \)       | \( O(N^2) \) | \( O(N^2) \)     |

Figure 3: Learning performance of policy search for hand-posture adjustment task over ten experiments: Mean value and standard deviations are shown.

4. Experiments

We conducted experiments with two robot control tasks in simulations: hand-posture adjustment task and table-sweeping task. We investigated the effectiveness of our proposed methods in the hand-posture adjustment task and demonstrated scalability for more challenging situations in the table-sweeping task.

4.1. Hand-posture adjustment task

4.1.1. Settings

We experimented with the simple robot task shown in Fig. 1a to confirm the performance of our proposed methods on the task with the multiple optimal actions. The task’s environment consisted of a UR5 robot arm and a red, 7\( \times \)12\( \times \)10 cm\(^3\) cube. The robot arm has a two-fingered gripper and can rotate its wrist in one revolution. The task aims to grasp the red cube by its longer
Figure 4: Robot behavior controlled with a policy learned by multimodal SGP-PS in hand-posture adjustment task.

Figure 5: (a) Learned policy for hand-posture adjustment task by multimodal SGP-PS with two components. Black dots indicate pseudo input. (b) and (c) show posterior probability of the data association of each data point shown in (a).

side; due to its physical constraint, the robot cannot grasp its shorter side. The robot can grasp the object in two ways: by rotating its wrist left or right. So, the task has two optimal actions.

The state is the orientation of the red cube. The action is the rotation angle.
of the robot wrist between $-\pi$ rad to $\pi$ rad. After determining the action, the robot moves to grasp the object with its wrist angle decided as an action. Since each episode is terminated in one step, the length of the trajectory is $T = 1$. Return function $R(d)$ is binary. $R(d) = 100$ if the robot grasps the cube, otherwise $R(d) = 0$.

We confirm the performances of our methods to the multiple optimal actions by comparing them with a unimodal SGP-PS that employs a standard SGP as a policy model. We set the number of components of the multimodal SGP-PS
to two and three to confirm its performance with different numbers of components to the number of multiple optimal actions. We used an isotropic-squared exponential kernel in each method. In this experiment, each method explored $E = 100$ episodes and learned a policy using the data of 100 episodes and the best 80 episodes as sample reuse after exploration. The number of pseudo-inputs of each method was set: $L = 20$.

4.1.2. Results

Figure 3 shows the mean and the standard deviation of the return over ten experiments using multimodal SGP-PS with two and three components, mode-seeking SGP-PS, and unimodal SGP-PS. The policies learned by our proposed methods indicate higher performance than that obtained by unimodal SGP-PS. This result suggests that the expansions of a policy model in our proposed methods are valid for the task with the multiple optimal actions. Moreover, multimodal SGP-PS can learn an appropriate policy even when the number of components is redundant. Fig. 4 shows the robot behavior using a learned
multimodal SGP policy which, can select an appropriate action for the object’s angle. Fig. 5a shows a learned policy by multimodal SGP-PS with two components. The learned multimodal SGP policy can captures multiple optimal actions indicated by the black dotted line by properly estimating data associations. To capture two optimal actions, component 1 captures two dotted lines, and component 2 captures the middle line. Figs. 5b and 5c show the optimized posterior distribution of $Z$, which is the data points assigned to each component. In the crossing point of the two components, the posterior of the data association indicates low probability. Fig. 6a shows a learned policy by multimodal SGP-PS with three components. Each component of the learned policy captures one of three dotted lines. Figs. 6b, 6c, and 6d show the optimized posterior distribution of $Z$. Similar to the case of the multimodal SGP-PS with two components, the posterior of the data association indicates a low probability of the two components’ crossing points. Fig. 7a shows a learned policy by mode-seeking SGP-PS. The policy acquired by it captured one of the optimal actions at each state, coincidentally in a similar way as with two components in Fig. 5a. Fig. 7b shows the posterior distribution of reliability $T$. The mode-seeking SGP-PS learned a unimodal policy by estimating the reliability of the data and ignoring low-reliability data.

In summary, the simulation results in the hand-posture adjustment task suggest the effectiveness of our proposed methods for learning a policy in tasks with multiple optimal actions.

4.2. Table-sweeping task

4.2.1. Settings

The aim of the table-sweeping task is to have the fetch robot sweep the five objects on the table individually by robot’s end-effector in the environment shown in Fig. 8. In this task, we learned an individual action policy based on the number of objects by exploiting the fact that this task can be naturally decomposed into subtasks according to the number of objects. Thus, five individual policies are learned. We constructed the table-sweeping task envi-
The task environment consists of the fetch robotic arm, a 40-cm diameter table, and five cylindrical objects with diameters of 7 cm, and heights of 2.5 cm. State consists of the two-dimensional Cartesian position of the end-effector and five objects, and the two-dimensional Cartesian relative position of the five objects related to the position of the end-effector. Each axis of the Cartesian coordinate is shown in Fig. 8a. Therefore, a state is represented by a 22-dimensional vector consisting of positions of the end-effector and five objects and relative positions of five objects. The definition of a state is relied on [45]. We set that the position and the relative position in a state corresponding to swept objects are assigned 0. An action is a two-dimensional vector and each dimension specifies the desired end-effector movement in each axis of two-dimensional Cartesian coordinates on the table. The absolute value of each dimension of action is limited to 4 cm or less so that even an optimal policy takes multiple steps to achieve the task. Return function $R(d)$ rewards the
trajectory data with 10 when an object is swept and punishes $0.1 \times T$ for the length of the episode $T$.

We set that the end-effector’s position is initialized to the center of the table after sweeping an object. The positions of the five objects are initialized by uniformly sampling from two patterns shown in Fig. 8. The five objects are arranged at 72-degree intervals on a circle with a diameter of 12 cm around the z-axis, but there is 32-degree difference between two patterns. The task is terminated when an object is swept or when the length of the trajectory has reached to $T = 20$. After the robot swept an object, the end-effector moves to the center of the table to execute the next task.

To compare our methods, we employed unimodal SGP-PS, SAC [30], and TRPO [44]. SAC and TRPO are state-of-the-art reinforcement learning methods that learn neural-network policies for continuous action space. The hyperparameters of SAC and TRPO are described in Tables E.3 and E.4.

We used a squared exponential kernel with automatic relevance determination in the SGP-based methods. The number of pseudo-inputs of each method is set to $L = 20$. We set the number of objects on the table as the number of components of the multimodal SGP-PS. For a fair comparison, the SGP-based methods and the NN-based methods as TRPO and SAC commonly update their policies using every 1000 steps worth of exploration episodes.

4.2.2. Results

Figure 9 shows the mean and standard deviation of the total reward over ten experiments using multimodal SGP-PS, mode-seeking SGP-PS, unimodal SGP-PS, SAC, and TRPO. The learning experiment was conducted individually for each number of objects. In the task with one object, the performance of unimodal SGP-PS is comparable multimodal SGP-PS and mode-seeking SGP-PS. However, as the number of objects increases, multimodal SGP-PS and mode-seeking SGP-PS outperform unimodal SGP-PS by effectively handling multiple optimal actions.

In the SAC algorithm, although the value function captured multiple optimal
actions, the policy employs a unimodal model. Then it needs to seek one of the optimal actions among multiple ones. The policy’s performance may become unstable because mode-seeking is not stable at the beginning of learning. Even though TRPO improves the policy monotonically; however, the performance is inadequate. It needs more data to learn a suitable policy in tasks with multiple optimal actions may be due to neural-network-based policies with many parameters. This result resembles one that was previously reported [2].

Fig. 10 shows the behavior of a policy learned by the multimodal SGP-PS. For each object, our methods can learn a policy to sweep an object by deciding action multiple steps.

Fig. 11 shows five trajectories generated by the policy learned by multimodal SGP-PS, mode-seeking SGP-PS, and unimodal SGP-PS on the task with five objects. The multimodal SGP policy learned the multiple optimal actions for sweeping an object in each initial state by the nature of multimodality. However, it could learn two or three optimal actions out of five due to limited data.
explored. The mode-seeking SGP policy learned one of the optimal actions by ignoring others successfully, while the unimodal SGP policy could not learn any optimal actions properly. We confirmed that each learned policy obtained different optimal actions according to the initial positions of objects.

In summary, all of our simulation results in the table-sweep task suggest the effectiveness of our methods for learning policies with multiple optimal actions. We confirmed that our methods more effectively learn a policy than SAC and TRPO in a task environment with a small number of samples and multiple optimal actions.

5. Discussion

We compared multimodal SGP-PS and mode-seeking SGP-PS. A learned policy by multimodal SGP-PS earned a higher return in simulations. However, it needs to decide the number of components affecting the performance before learning. Therefore, we could select a suitable one from the multimodal SGP-PS and the mode-seeking SGP-PS depending on whether the number of multimodalities of the function is known.
In a mode-seeking SGP-PS, if there is a clear difference in the size of data among multiple optimal actions in the training data, learning tends to proceed faster since it fits the model. If the multiple optimal actions are observed almost evenly, the mode-seeking SGP-PS may struggle to capture one among them; in that case, the multimodal SGP-PS tends to be faster and more stable than the mode-seeking SGP-PS. Our experimental results may support the above discussion: For the table-sweeping task with more than three objects, the multimodal SGP-PS converged learning faster than the mode-seeking SGP-PS.
For the table-sweeping task with one or two objects, the mode-seeking SGP-PS outperformed the multimodal SGP-PS.

Our methods can compute the predictive distribution to determine an action at any state since they employ GP-based policy models. Our methods explore state-action space by sampling the action using the predictive distribution, i.e., on-policy learning. In the multimodal SGP-PS, which has multiple components, a softmax function with the uncertainties of all the predictive distributions computes the probability of the component selection.

Although GP provides several advantages for our method, it has limitations in computational complexity. For tasks that require handling huge amounts of data, our methods require much computational complexity of learning and predictive distribution. Perhaps our methods are unsuitable for long-horizon tasks.

To apply our methods to complex tasks, we must handle raw images and need more data-efficient algorithms. Features need to be designed to handle raw images as a state in a non-parametric policy search. A kernel function that incorporates the structure of convolutional neural networks and deep learning technology specializing in image processing was proposed [46]. By introducing it into our method, we can learn a policy that uses images as input. To improve data efficiency, we can introduce a policy update with limited information loss [2]. The multimodal SGP-PS needs to decide the number of components since we use overlapping mixtures of GPs inspired by a previous work [16]. We can also infer the number of components [47].

6. Conclusions

We proposed a multimodal SGP-PS and a mode-seeking SGP-PS, which are non-parametric policy search methods for a task with multiple optimal actions. The multimodal SGP-PS employs a multimodal policy prior inspired by OMGP to learn a policy that can capture multiple optimal actions. The mode-seeking SGP-PS learns a unimodal policy that captures one optimal action by employing
an outlier-robust likelihood function. We derived the updating laws of both methods based on variational Bayesian inference.

To investigate the performance of our methods, we conducted two manipulation tasks: 1) a hand-posture adjustment task and 2) a table-sweeping task. We confirmed that our methods can learn suitable policies in an environment with multiple optimal actions.

Appendix A. Derivation of analytical solutions in multimodal SGP-PS

The analytical solution of variational distribution \( q(f^{(m)}) \) is obtained by solving the following equation:

\[
\log q(f^{(m)}) = \int \left\{ \log p(\tilde{a} \mid \{f^{(m)}\}, Z)p(\{f^{(m)} \mid \{\tilde{f}^{(m)}\}\})p(Z) \cdot \right. \\
\left. p(\{\tilde{f}^{(m)}\}) \right\} q(\{\tilde{f}^{(i\backslash m)}\})q(Z)d\{\tilde{f}^{(i\backslash m)}\}dZ + C \quad (A.1)
\]

Here \( \{\tilde{f}^{(i\backslash m)}\} \) indicates all the pseudo outputs without \( \tilde{f}^{(m)} \). The following is the analytical solution of \( q(f^{(m)}) \):

\[
q(f^{(m)}) = \mathcal{N}(\tilde{f}^{(m)} \mid \mu^{(m)}, \Sigma^{(m)}), \\
\mu^{(m)} = K_{S}^{(m)} Q_{L}^{(m)} K_{S}^{(m)} W B^{(m)} W a, \\
\Sigma^{(m)} = K_{S}^{(m)} Q_{L}^{(m)} K_{S}^{(m)}, \\
Q_{L}^{(m)} = K_{S}^{(m)} + K_{S}^{(m)} W B^{(m)} W K_{S}^{(m)} W K_{S}^{(m)}, \quad (A.2) \\
B^{(m)} = \text{diag} \left\{ \frac{\Pi_{nm}}{\lambda_{nm}^{(m)} + \sigma^{2}} \right\}. \quad (A.3)
\]

The analytical solution of variational distribution \( q(Z) \) is obtained by solving
the following equation:

$$
\log q(Z) = \int \left\{ \log p(\tilde{a} \mid \{f^{(m)}\}, Z) p(\tilde{f}^{(m)} \mid f^{(m)}) p(Z) \right\} \\
\cdot p(\{\tilde{f}^{(m)}\}) q(\{\tilde{f}^{(m)}\}) \cdot d\{\tilde{f}^{(m)}\} + C.
$$

(A.4)

The following is the analytical solution of $q(Z)$:

$$
q(Z) = \prod_{n=1, m=1}^{N, M} \hat{\Pi}_{nm}^{Z_{nm}},
$$

(A.5)

$$
\hat{\Pi}_{nm} = \Pi_{nm} \exp(b_{nm}),
$$

$$
b_{nm} = -\frac{1}{2} \log 2\pi (\lambda_n^{(m)} + \sigma^2)
\left( \frac{W_{nn} a_n - W_{nn} K_{s_s S}^{(m)^{-1}} \mu^{(m)}}{2(\lambda_n^{(m)} + \sigma^2)} \right)^2
\left( \frac{W_{nn} K_{s_s S}^{(m)^{-1}} \mu^{(m)}}{2(\lambda_n^{(m)} + \sigma^2)} \right)
\left( \frac{W_{nn} K_{s_s S}^{(m)^{-1}} \mu^{(m)}}{2(\lambda_n^{(m)} + \sigma^2)} \right) W_{nn}.
$$

The lower bound of the marginal likelihood is also obtained analytically:

$$
\log J^*_L =
\sum_{n=1, m=1}^{N, M} \frac{1}{2} a^T W \left( B^{(m)^{-1}} + W K_{s_s S}^{(m)} K_{s_s S}^{(m)^{-1}} W \right) a
$$

$$
\left. + \sum_{n=1, m=1}^{N, M} \log| R^{(m)} |_{nn} - \text{KL}(q(Z) \mid\mid p(Z)) \right.
$$

$$
\left. - \frac{1}{2} \sum_{n=1, m=1}^{N, M} | \hat{\Pi} |_{nm} \log 2\pi (\lambda_n^{(m)} + \sigma^2), \right. \quad \text{A.6}
$$

$$
R^{(m)} = \text{chol} \left( I + B^{(m)^{-1/2}} W K_{s_s S}^{(m)} K_{s_s S}^{(m)^{-1}} W B^{(m)^{-1/2}} \right).
$$

(A.7)
Appendix B. Predictive distribution of multimodal SGP-PS

Compute the predictive distribution using the variational distribution instead of the true posterior distribution:

\[
p(a_\star | s_\star) \approx \int p(a_\star | \bar{f}(m)s_\star)q(\bar{f}(m))d\bar{f}(m) = \mathcal{N}(a_\star | \mu_\star, \sigma_\star^2),
\]

(B.1)

\[\mu_\star = K_{s,\star} Q^{-1}_{\bar{f}} K_{\bar{f}} W B^T \bar{f} \cdot W_{\bar{f}}, \]

\[\sigma_\star^2 = K_{s,\star} - K_{s,\star} K_{\bar{f}} Q^{-1}_{\bar{f}} (K_{\bar{f}} W B^T \bar{f} \cdot W_{\bar{f}}) + \sigma^2.\]

Appendix C. Derivation of analytical solutions in mode-seeking SGP-PS

The analytical solution of variational distribution \(q(\tau_n)\) is obtained by solving the following equation:

\[
\log q(\tau_n) = \int \{ \log p(\bar{a} | f, T)p(f | \bar{f})p(T) p(\bar{f}) \} \cdot q(\{\tau_{i \backslash n}\})q(\bar{f})d\{\tau_{i \backslash n}\}d\bar{f} + C,
\]

(C.1)

where \(\{\tau_{i \backslash n}\}\) indicates all precision variables without \(\tau_n\). The following is the analytical solution of \(q(\tau_n)\):

\[
q(\tau_n) = \text{Gam}(\tau_n | a_n, b_n),
\]

(C.2)

\[a_n = \nu + 1, \]

\[b_n = \frac{W_{2 \times 2}^2 (a_n - A_n \mu)^2}{2} + \frac{\text{tr}(A_n^T W_{2 \times 2}^2 A_n C)}{2} + \frac{W_{2 \times 2}^2 \lambda_n + \nu \sigma^2}{2},\]

\[A = K_{\bar{f}} K_{\bar{f}}^{-1}.\]

The analytical solution of variational distribution \(q(\bar{f})\) is obtained by solving
the following equation:

\[ q(\bar{f}) = \mathcal{N}(\bar{f} \mid \mu, C), \]  
\[ \mu = K_s \Sigma^{-1} K_{s, s} W T W a, \]  
\[ C = K_s \Sigma^{-1} K_s, \]  
\[ \Sigma = K_{s, s} W T W K_{s, s} + K_s, \]  
\[ \mathbf{T} = \text{diag}\{a_n/b_n\}. \]

The lower bound of the marginal likelihood is also obtained analytically:

\[
\log J'_L = - \frac{N}{2} \log 2\pi - \frac{1}{2} \log |\hat{T}^{-1}| - \frac{1}{2} (a - A\mu)^T W \hat{T} W (a - A\mu) - \frac{1}{2} \text{tr}(A^T W \hat{T} W A C) - \text{KL}(q(\bar{f}) \mid \mid p(\bar{f})) - \text{KL}(q(T) \mid \mid p(T)) - \frac{1}{2} \text{tr}(W \hat{T} W A).
\]

Appendix D. Predictive distribution of mode-seeking SGP-PS

Compute the predictive distribution using the variational distribution instead of the true posterior distribution:

\[
p(a_* \mid s_*) \approx \int p(a_* \mid f, \tau)p(f \mid \bar{f})q(\bar{f})q(\tau)d\bar{f}d\tau
\]

\[ = \mathcal{N}(y_* \mid \mu_*, \sigma_*^2), \]  
\[ \mu_* = K_{s, s} \Sigma^{-1} K_{\bar{s}, s} W \hat{T} W a, \]  
\[ \sigma_*^2 = K_s - K_{s, s} \Sigma^{-1} K_{\bar{s}, s} + K_{s, s} \Sigma^{-1} K_{\bar{s}, s} + \sigma^2. \]

Appendix E. Hyperparameters of SAC and TRPO in the table-sweeping task

This section describes the hyperparameters of SAC and TRPO.
Table E.3: SAC hyperparameters

| Parameter                  | Value       |
|---------------------------|-------------|
| optimizer                 | Adam        |
| learning rate             | $3 \times 10^{-4}$ |
| discount ($\gamma$)       | 0.99        |
| replay buffer size        | $10^6$      |
| number of hidden layers   | 2           |
| number of hidden units per layer | 128 |
| number of sample per minibatch | 256 |
| nonlinearity              | ReLU        |

Table E.4: TRPO hyperparameters

| Parameter                  | Value       |
|---------------------------|-------------|
| optimizer                 | Adam        |
| learning rate             | $1 \times 10^{-3}$ |
| discount ($\gamma$)       | 0.995       |
| Stepsize (KL)             | 0.01        |
| replay buffer size        | $10^6$      |
| number of hidden layers   | 2           |
| number of hidden units per layer | 64 |
| number of sample per minibatch | 256 |
| nonlinearity              | Tanh        |

References

[1] J. Kober, E. Öztop, J. Peters, Reinforcement learning to adjust robot movements to new situations, in: International Joint Conference on Artificial Intelligence (IJCAI), 2011, pp. 2650–2655.

[2] H. van Hoof, G. Neumann, J. Peters, Non-parametric policy search with limited information loss, Journal of Machine Learning Research 18 (73) (2017) 1–46.
[3] J. Kober, J. A. Bagnell, J. Peters, Reinforcement learning in robotics: A survey, International Journal of Robot Research 32 (11) (2013) 1238–1274.

[4] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, D. Wierstra, Continuous control with deep reinforcement learning, arXiv:1509.02971 [cs, stat].

[5] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, K. Kavukcuoglu, Asynchronous methods for deep reinforcement learning, in: International Conference on Machine Learning (ICML), 2016, pp. 1928–1937.

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

[7] S. Gu, E. Holly, T. Lillicrap, S. Levine, Deep Reinforcement Learning for Robotic Manipulation with Asynchronous Off-Policy Updates, arXiv:1610.00633 [cs].

[8] S. Levine, C. Finn, T. Darrell, P. Abbeel, End-to-end training of deep visuomotor policies, Journal of Machine Learning Research 17 (39) (2016) 1–40.

[9] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, Trust region policy optimization, in: International Conference on Machine Learning (ICML), Vol. 37, 2015, pp. 1889–1897.

[10] Y. Tsurumine, Y. Cui, E. Uchibe, T. Matsubara, Deep reinforcement learning with smooth policy update: Application to robotic cloth manipulation, Robotics and Autonomous Systems 112 (2019) 72–83.
[11] E. Snelson, Z. Ghahramani, Sparse Gaussian processes using pseudo-inputs, in: Advances in Neural Information Processing Systems (NIPS), 2005, pp. 1257–1264.

[12] M. K. Titsias, Variational learning of inducing variables in sparse Gaussian processes, in: International Conference on Artificial Intelligence and Statistics (AISTATS), 2009, pp. 567–574.

[13] G. Neumann, Variational inference for policy search in changing situations, in: International Conference on Machine Learning (ICML), 2011, pp. 817–824.

[14] S. Levine, V. Koltun, Variational policy search via trajectory optimization, in: Advances in Neural Information Processing Systems (NIPS), 2013, pp. 207–215.

[15] K. S. Luck, J. Pajarinen, E. Berger, V. Kyrki, H. B. Amor, Sparse latent space policy search, in: Conference on Artificial Intelligence (AAAI), 2016, p. 1911–1918.

[16] M. Lázaro-Gredilla, S. V. Vaerenbergh, N. D. Lawrence, Overlapping mixtures of Gaussian processes for the data association problem, Pattern Recognition 45 (4) (2012) 1386–1395.

[17] J. Vanhatalo, P. Jylänki, A. Vehtari, Gaussian process regression with student-t likelihood, in: Advances in Neural Information Processing Systems (NIPS), 2009, pp. 1910–1918.

[18] P. Jylänki, J. Vanhatalo, A. Vehtari, Robust Gaussian process regression with a student-t likelihood, Journal of Machine Learning Research 12 (2011) 3227–3257.

[19] E. Rohmer, S. P. N. Singh, M. Freese, V-rep: A versatile and scalable robot simulation framework, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2013, pp. 1321–1326.
[20] E. Todorov, T. Erez, Y. Tassa, Mujoco: A physics engine for model-based control, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012, pp. 5026–5033.

[21] H. Sasaki, T. Matsubara, Multimodal policy search using overlapping mixtures of sparse Gaussian process prior, in: IEEE/RSJ International Conference on Robotics and Automation (ICRA), 2019, pp. 2433–2439.

[22] J. A. D. Bagnell, J. Schneider, Policy search in reproducing kernel hilbert space, Tech. Rep. CMU-RI-TR-03-45, Carnegie Mellon University, Pittsburgh, PA (November 2003).

[23] G. Lever, R. Stafford, Modelling Policies in MDPs in Reproducing Kernel Hilbert Space, in: International Conference on Artificial Intelligence and Statistics (AISTATS), 2015, pp. 590–598.

[24] N. A. Vien, P. Englert, M. Toussaint, Policy search in reproducing kernel hilbert space, in: International Joint Conference on Artificial Intelligence (IJCAI), 2016, pp. 2089–2096.

[25] S. Calinon, P. Kormushev, D. G. Caldwell, Compliant skills acquisition and multi-optima policy search with EM-based reinforcement learning, Robotics and Autonomous Systems 61 (4) (2013) 369–379.

[26] C. Daniel, G. Neumann, O. Kroemer, J. Peters, Hierarchical relative entropy policy search, Journal of Machine Learning Research 17 (93) (2016) 1–50.

[27] A. Kupcsik, M. P. Deisenroth, J. Peters, A. P. Loh, P. Vadakkepat, G. Neumann, Model-based contextual policy search for data-efficient generalization of robot skills, Artificial Intelligence 247 (2017) 415 – 439.

[28] T. Osa, M. Sugiyama, Hierarchical policy search via return-weighted density estimation, in: Conference on Artificial Intelligence (AAAI), 2018, pp. 3860–3867.
[29] T. Haarnoja, H. Tang, P. Abbeel, S. Levine, Reinforcement learning with
deep energy-based policies, in: International Conference on Machine Learn-
ing (ICML), 2017, pp. 1352–1361.

[30] T. Haarnoja, A. Zhou, P. Abbeel, S. Levine, Soft actor-critic: Off-policy
maximum entropy deep reinforcement learning with a stochastic actor, in:
International Conference on Machine Learning (ICML), 2018, pp. 1861–
1870.

[31] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar,
H. Zhu, A. Gupta, P. Abbeel, S. Levine, Soft Actor-Critic Algorithms and
Applications, arXiv:1812.05905 [cs, stat]

[32] T. Haarnoja, Acquiring diverse robot skills via maximum entropy deep rein-
forcement learning, Ph.D. thesis, University of California, Berkeley (2018).

[33] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang,
D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, S. Levine, Scal-
able deep reinforcement learning for vision-based robotic manipulation, in:
Conference on Robot Learning (CoRL), 2018, pp. 651–673.

[34] V. Tresp, A Bayesian committee machine, Neural Computation 12 (11)
(2000) 2719–2741.

[35] C. E. Rasmussen, Z. Ghahramani, Infinite mixtures of Gaussian process
experts, in: Advances in Neural Information Processing Systems (NIPS),
2001, pp. 881–888.

[36] E. Meeds, S. Osindero, An alternative infinite mixture of Gaussian process
experts, in: Advances in Neural Information Processing Systems (NIPS),
2006, pp. 883–890.

[37] C. Yuan, C. Neubauer, Variational mixture of Gaussian process experts,
in: Advances in Neural Information Processing Systems (NIPS), 2008, pp.
1897–1904.
[38] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, P. Abbeel, Domain randomization for transferring deep neural networks from simulation to the real world, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2017, pp. 23–30.

[39] L. Pinto, J. Davidson, R. Sukthankar, A. Gupta, Robust adversarial reinforcement learning, in: International Conference on Machine Learning (ICML), 2017, pp. 2817–2826.

[40] J. Fu, K. Luo, S. Levine, Learning robust rewards with adversarial inverse reinforcement learning, in: International Conference on Learning Representations (ICLR), 2018.

[41] P. F. Christiano, J. Leike, T. Brown, M. Martic, S. Legg, D. Amodei, Deep reinforcement learning from human preferences, in: Advances in Neural Information Processing Systems (NIPS), 2017, pp. 4299–4307.

[42] M. Sugiyama, H. Hachiya, H. Kashima, T. Morimura, Least absolute policy iteration for robust value function approximation, in: IEEE/RSJ International Conference on Robotics and Automation (IROS), 2009, pp. 2904–2909.

[43] J. Wang, Y. Liu, B. Li, Reinforcement Learning with Perturbed Rewards, arXiv:1810.01032 [cs, stat].

[44] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, Trust Region Policy Optimization, in: International Conference on Machine Learning (ICML), 2015, pp. 1889–1897.

[45] M. Plappert, M. Andrychowicz, A. Ray, B. McGrew, B. Baker, G. Powell, J. Schneider, J. Tobin, M. Chociej, P. Welinder, V. Kumar, W. Zaremba, Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research, arXiv:1802.09464 [cs].
[46] A. Garriga-Alonso, C. E. Rasmussen, L. Aitchison, Deep convolutional networks as shallow Gaussian processes, in: International Conference on Learning Representations (ICLR), 2019.

[47] J. C. Ross, J. G. Dy, Nonparametric mixture of Gaussian processes with constraints, in: International Conference on Machine Learning (ICML), 2013, pp. 1346–1354.