Research Article

Kiseki Kameda* and Fuyuhiko Tanaka

Reinforcement learning with Gaussian process regression using variational free energy

https://doi.org/10.1515/jisys-2022-0205
received May 30, 2022; accepted December 14, 2022

Abstract: The essential part of existing reinforcement learning algorithms that use Gaussian process regression involves a complicated online Gaussian process regression algorithm. Our study proposes online and mini-batch Gaussian process regression algorithms that are easier to implement and faster to estimate for reinforcement learning. In our algorithm, the Gaussian process regression updates the value function through only the computation of two equations, which we then use to construct reinforcement learning algorithms. Our numerical experiments show that the proposed algorithm works as well as those from previous studies.

Keywords: reinforcement learning, Gaussian process, Q-learning, Bayesian inference

MSC 2020: 68

1 Introduction

Reinforcement learning is learning how to behave to maximize reward. One of the most striking examples of the application of reinforcement learning is AlphaGo [1,2], which defeated a professional Go player. Various function approximations have been used in reinforcement learning algorithms, one of which is based on Gaussian process regression [3]. Gaussian process regression is a Bayesian nonparametric method often used as a standard nonlinear regression model and has some desired properties such as a low possibility of overfitting and the ability to express estimation uncertainty.

Reinforcement learning algorithms are often based on estimating a value function [4]. In classical reinforcement learning, the model of the value function is represented in table form. In other words, the model is represented as a matrix of pairs of states and actions. However, as the set of states and actions becomes larger, estimating the value function becomes harder. We solve this problem by representing the value function by functional approximation [4,5]. One of the functional approximations used in reinforcement learning is Gaussian process regression, and its features are expected to be helpful. The principal advantage is the high degree of freedom of the model obtained by the kernel function. Through Bayesian learning, estimation uncertainty is expressed naturally.

However, due to its high computational cost, Gaussian process regression has not been suitable for reinforcement learning. In addition, the existing online algorithms in Gaussian process regression have

* **Corresponding author: Kiseki Kameda,** Department of Systems Innovation, Graduate School of Engineering Science, Osaka University, Osaka 560-8531, Japan, e-mail: kameda@sigmath.es.osaka-u.ac.jp
Fuyuhiko Tanaka: Center for Education in Liberal Arts and Sciences, Osaka University, Osaka 560-0043, Japan, e-mail: ftanaka.celas@osaka-u.ac.jp

been very complicated to implement. Thus, we need to develop a simple algorithm for online Gaussian process regression that is easier to implement for reinforcement learning.

Prior research on model-free reinforcement learning using Gaussian process regression includes state-action-reward-state-action (SARSA)-based methods and Q-learning-based methods. There are several methods based on SARSA, such as GP-SARSA [6,7] and iGP-SARSA [8]. However, these methods have problems, such as high computational costs. The GPQ [9] is an algorithm based on Q-learning and uses the sparse online Gaussian processes method [10]. These algorithms are based on complex methods to reduce the computational complexity of Gaussian process regression.

In this article, to overcome the above shortcomings, we propose a mini-batch-learnable variational free energy (VFE) method and a reinforcement learning algorithm based on the VFE method and Q-learning. The VFE method approximates the posterior distribution by variational inference. The offline VFE method [11] is widely used to reduce the computational complexity of Gaussian process regression.

The VFE method is one of the methods using inducing points and is expressed by a simple formula. The inducing points must be provided to estimate using this method. Choosing the inducing points may be difficult for some environments, but the method of using them is easier to estimate. For example, it is more difficult to select inducing points in higher-dimensional environments. The computational complexity of our method is the same as the offline VFE method. The computational complexity is $O(N)$, where N is the number of data. Mini-batch learning is more efficient than online learning in reinforcement learning.

Our main contributions are as follows:

- We extend the VFE method to allow online and mini-batch learning.
- We propose a reinforcement learning algorithm using the mini-batch-learnable VFE method.

In our experiments, we confirm that the proposed method can be learned just as well as the existing methods. We compare the proposed reinforcement learning algorithm with the algorithms in previous studies by considering a two-dimensional grid global environment as an example. Numerical results show that the proposed algorithm works as well as the current GPQ but is simpler and faster.

In Section 2, we explain the basics of reinforcement learning and Gaussian process regression in some detail. We then propose our reinforcement learning algorithm using the online Gaussian process regression with a mini-batch-learnable VFE method in Section 3. Section 4 shows the numerical results of experiments in a two-dimensional grid world. Finally, concluding remarks are given in Section 5.

2 Background

2.1 Reinforcement learning

In reinforcement learning, the environment is often modeled using the Markov decision process (MDP) [12]. The MDP is represented by $M = \langle S, A, R, p, \gamma \rangle$. Here, S and A denote the set of states and actions, $R : S \to \mathbb{R}$ is a function of immediate reward, $p : S \times A \times S \to [0, 1]$ is the transition probability, and $\gamma \in [0, 1)$ is the discount rate. Let $\pi : S \times A \to [0, 1]$ be the action rule of the agent called the policy. We have to estimate the value function from the trajectory $\{s_0, a_0, r_0, s_1, a_1, r_1, \ldots\}$ generated from the MDP and the policy. We define the discounted reward sum as $R_t = r_t + \gamma r_{t+1} + \cdots$. In reinforcement learning, the goal is to find the action rule that maximizes the reward, i.e., to find $\pi^* = \max_\pi E^\pi[R_0]$, where $E^\pi[\cdot]$ represents the expected value under the given policy π. To find π^*, we introduce a value function $Q^\pi(s, a) = E^\pi[R_0 | s_0 = s, a_0 = a]$. Since π^* can be obtained from the optimal value function $Q^* = \max_\pi Q^\pi(s, a)$, finding π^* reduces to finding Q^*. Q-learning [13] is often used as a method to find Q^*. If the value function is in table form, we can update the value function with $Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t[r_t + \gamma \max_a Q_t(s_{t+1}, a) - Q_t(s_t, a_t)]$. In the case of function approximation, the value of $Q(s_t, a_t)$ is updated to be closer to the value on the right-hand side.

2.2 Gaussian process regression

Gaussian process regression [3] is regarded as a Bayesian nonparametric regression, where we estimate the function $y = f(\boldsymbol{x})$ from the input variable $\boldsymbol{x}$ to the output variable y. Let $D = \{\boldsymbol{X}, \boldsymbol{Y}\}$, $\boldsymbol{X} = \{\boldsymbol{x}_n \in \mathbb{R}^p\}_{n=1}^N$, and $\boldsymbol{Y} = \{y_n \in \mathbb{R}\}_{n=1}^N$ be the data of N pairs of input and output variables. Assume that the output variable is accompanied by normally distributed noise, and the model is $y_i = f(\boldsymbol{x}_i) + \varepsilon_i$, $(i = 1, \ldots, N)$, $\varepsilon_i \overset{\text{iid}}{\sim} N(0, \sigma^2)$. The goal is to construct the predictive distribution of the output y_* for a new input $\boldsymbol{x}_*$. Let $f \sim N(0, \boldsymbol{K}_{NN})$ be the prior distribution. The covariance function is defined as $[\boldsymbol{K}_{NN}]_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, where $k(\cdot,\cdot)$ denotes a kernel function. In this article, we use the radial basis function (RBF) kernel $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \alpha^2 \exp[-||\boldsymbol{x}_1 - \boldsymbol{x}_2||^2/\beta^2]$. The predictive distribution is obtained from the formula for the posterior distribution of the normal distribution. The predictive distribution of y_* is

$$\begin{aligned} p(y_*|\boldsymbol{Y}, \boldsymbol{x}_*, \boldsymbol{X}) &= N(y_*|\mu_*, \Sigma_*), \\ \mu_* &= \boldsymbol{k}_{*N}(\boldsymbol{K}_{NN} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{Y}, \\ \Sigma_* &= k(\boldsymbol{x}_*, \boldsymbol{x}_*) + \sigma^2\boldsymbol{I} - \boldsymbol{k}_{*N}(\boldsymbol{K}_{NN} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{k}_{*N}^T, \end{aligned} \tag{1}$$

with $\boldsymbol{k}_{*N} = (k(\boldsymbol{x}_*, \boldsymbol{x}_1), \ldots, k(\boldsymbol{x}_*, \boldsymbol{x}_N))$. Only matrix multiplication is required to calculate the predictive distribution, and the computational cost is $O(N^3)$. One way to reduce the computational cost is to use inducing points. The partially independent training condition [14], the fully independent training condition [15], and the VFE [11] are well-known methods that use inducing points. In methods using inducing points, the computational cost is $O(NM^2)$, where M is the number of inducing points, and, in general, $N > M$.

We introduce the VFE method and explain this method in detail. Let $\boldsymbol{Z} = \{\boldsymbol{z}_j \in \mathbb{R}^p\}_{j=1}^M$ be the set of inducing points. The model for Gaussian process regression using inducing points is

$$p(\boldsymbol{Y}|\boldsymbol{f}) = N(\boldsymbol{Y}|\boldsymbol{f}, \sigma^2\boldsymbol{I}), \tag{2}$$

$$p(\boldsymbol{f}|\boldsymbol{u}) = N(\boldsymbol{f}|\boldsymbol{K}_{MN}^T\boldsymbol{K}_{MM}^{-1}\boldsymbol{u}, \boldsymbol{K}_{NN} - \boldsymbol{K}_{MN}^T\boldsymbol{K}_{MM}^{-1}\boldsymbol{K}_{MN}), \tag{3}$$

$$p(\boldsymbol{u}) = N(\boldsymbol{u}|\boldsymbol{0}, \boldsymbol{K}_{MM}), \tag{4}$$

with $\boldsymbol{u} = (f(\boldsymbol{z}_1), \ldots, f(\boldsymbol{z}_M))$, $[\boldsymbol{K}_{MM}]_{i,j} = k(\boldsymbol{z}_i, \boldsymbol{z}_j)$, and $[\boldsymbol{K}_{MN}]_{i,j} = k(\boldsymbol{z}_i, \boldsymbol{x}_j)$. For details, see [11]. Then, the predictive distribution of y_* is

$$p(y_*|\boldsymbol{Y}) = N(\mu(\boldsymbol{x}_*), k(\boldsymbol{x}_*)), \tag{5}$$

$$\mu(\boldsymbol{x}_*) = \boldsymbol{k}_{*M}\boldsymbol{K}_{MM}^{-1}\boldsymbol{\mu}, \tag{6}$$

$$k(\boldsymbol{x}_*) = k(\boldsymbol{x}_*, \boldsymbol{x}_*) - \boldsymbol{k}_{*M}\boldsymbol{K}_{MM}^{-1}(\boldsymbol{I} + \boldsymbol{\Sigma}\boldsymbol{K}_{MM}^{-1})\boldsymbol{k}_{*M}^T, \tag{7}$$

$$\boldsymbol{\mu} = \frac{1}{\sigma^2}\boldsymbol{K}_{MM}\left(\boldsymbol{K}_{MM} + \frac{1}{\sigma^2}\boldsymbol{K}_{MN}\boldsymbol{K}_{MN}^T\right)^{-1}\boldsymbol{K}_{MN}\boldsymbol{Y}, \tag{8}$$

$$\boldsymbol{\Sigma} = \boldsymbol{K}_{MM}\left(\boldsymbol{K}_{MM} + \frac{1}{\sigma^2}\boldsymbol{K}_{MN}\boldsymbol{K}_{MN}^T\right)^{-1}\boldsymbol{K}_{MM}, \tag{9}$$

with $\boldsymbol{k}_{*M} = (k(\boldsymbol{x}_*, \boldsymbol{z}_1), \ldots, k(\boldsymbol{x}_*, \boldsymbol{z}_M))$. In the VFE method, we can estimate the predictive distribution by computing these equations. If we used all data each time for mini-batch learning, the computational cost would be high. Therefore, in the next section, we will extend the method so that it can be estimated without changing the computational cost.

2.3 Related work

Here, we introduce previous research on reinforcement learning using Gaussian process regression. Reinforcement learning algorithms can be divided into two categories: model-based and model-free. In model-based reinforcement learning algorithms, Gaussian process regression can be used not only for the

value function but also for the environment model, such as [16–18]. Since the model of the environment is also estimated, the computational cost is higher, but it is not necessary to learn the model of the environment.

Next, we discuss model-free reinforcement learning algorithms. Reinforcement learning algorithms can be categorized into two types: on-policy learning and off-policy learning. SARSA [4] is a typical algorithm for on-policy learning. GP-SARSA [6,7] is an algorithm based on SARSA that learns a value function using a Gaussian process. iGP-SARSA [8] is a method that improves the exploration of GP-SARSA. On the other hand, Q-learning is a typical off-policy learning algorithm. GPQ [9] is a learning method based on Q-learning in which the value function is represented by a Gaussian process.

Next, we describe online learning for Gaussian process regression. GPQ uses the sparse online Gaussian processes method [10] to construct the algorithm. While the Gaussian regression algorithm used by these methods is complex, this article proposes an algorithm that can be updated with only two formulas. The difference between these Gaussian regression methods and the proposed method is the use of inducing points. Offline learning methods for Gaussian process regression with inducing points include VFE [11], fully independent training conditional (FITC) [15], and partially independent training conditional (PITC) [14]. However, reinforcement learning algorithms require online or mini-batch training. Online learning methods have been proposed for FITC and PITC without changing the computational cost [19].

In this article, we use the VFE, which is widely used and can be computed with a simple formula. We also propose mini-batch learning, which has not been proposed in previous studies. The relationship between the previous study and the proposed method is summarized in Table 1. We construct a Q-learning algorithm using our proposed mini-batch VFE. The difference from previous studies is that our proposed mini-batch VFE is used, but the original structure is the same as that of Q-learning.

3 Reinforcement learning with Gaussian process regression using VFE

We extend the VFE formulas (5)–(9) to be updatable online. We have used the methods of previous studies [19] as a guide. We rewrite the covariance function for an online update, where $\boldsymbol{\Sigma}_N$ for N pairs of data is given as follows:

$$\boldsymbol{\Sigma}_N = \boldsymbol{K}_{MM}\left(\boldsymbol{K}_{MM} + \frac{1}{\sigma^2}\boldsymbol{K}_{MN}\boldsymbol{K}_{MN}^T\right)^{-1}\boldsymbol{K}_{MM}. \tag{10}$$

Let $\boldsymbol{x}_+$ and $\boldsymbol{y}_+$ be the new input data. We transform the covariance function $\boldsymbol{\Sigma}_{N+1}$ with $N+1$ pairs of data as follows:

$$\boldsymbol{\Sigma}_{N+1} = \boldsymbol{K}_{MM}\left(\boldsymbol{K}_{MM} + \frac{1}{\sigma^2}\boldsymbol{K}_{M,N+1}\boldsymbol{K}_{M,N+1}^T\right)^{-1}\boldsymbol{K}_{MM} \tag{11}$$

$$= \boldsymbol{K}_{MM}\left(\boldsymbol{K}_{MM} + \frac{1}{\sigma^2}\boldsymbol{K}_{MN}\boldsymbol{K}_{MN}^T + \frac{1}{\sigma^2}\boldsymbol{K}_{M+}\boldsymbol{K}_{M+}^T\right)^{-1}\boldsymbol{K}_{MM}, \tag{12}$$

where $\boldsymbol{K}_{M+} = (k(\boldsymbol{x}_+, \boldsymbol{z}_1), \ldots, k(\boldsymbol{x}_+, \boldsymbol{z}_M))$.

Table 1: The relationship between the previous study and the proposed method

	Online learning	Mini-batch learning
FITC	(Bijl et al., 2015) [19]	—
PITC	(Bijl et al., 2015) [19]	—
VFE	Proposed method	Proposed method

By using the formula for the inverse of the sum of two matrices, we have

$$\boldsymbol{\Sigma}_{N+1} = \boldsymbol{\Sigma}_N - \frac{\boldsymbol{\Sigma}_N \boldsymbol{P} \boldsymbol{\Sigma}_N}{1 + \mathrm{tr}(\boldsymbol{\Sigma}_N \boldsymbol{P})}, \tag{13}$$

$$\boldsymbol{P} = \frac{1}{\sigma^2} \boldsymbol{K}_{MM}^{-1} \boldsymbol{K}_{M+} \boldsymbol{K}_{M+}^T \boldsymbol{K}_{MM}^{-1}. \tag{14}$$

Using this formula, $\boldsymbol{\Sigma}$ can be updated online. Then, $\boldsymbol{\mu}$ can also be transformed as follows:

$$\boldsymbol{\mu}_{N+1} = \frac{1}{\sigma^2} \boldsymbol{K}_{MM} \left(\boldsymbol{K}_{MM} + \frac{1}{\sigma^2} \boldsymbol{K}_{M,N+1} \boldsymbol{K}_{M,N+1}^T \right)^{-1} \boldsymbol{K}_{M,N+1} \boldsymbol{y}_{N+1} \tag{15}$$

$$= \frac{1}{\sigma^2} \boldsymbol{\Sigma}_{N+1} \boldsymbol{\Sigma}_N^{-1} \boldsymbol{\mu}_N + \frac{1}{\sigma^2} \boldsymbol{\Sigma}_{N+1} \boldsymbol{K}_{MM}^{-1} \boldsymbol{K}_{M+} \boldsymbol{y}_+ \tag{16}$$

$$= \frac{1}{\sigma^2} \left(\boldsymbol{I} - \frac{\boldsymbol{\Sigma}_N \boldsymbol{P}}{1 + \mathrm{tr}(\boldsymbol{\Sigma}_N \boldsymbol{P})} \right) \boldsymbol{\mu}_N + \frac{1}{\sigma^2} \boldsymbol{\Sigma}_{N+1} \boldsymbol{K}_{MM}^{-1} \boldsymbol{K}_{M+} \boldsymbol{y}_+. \tag{17}$$

From this equation, we can update $\boldsymbol{\mu}_N$ from $\boldsymbol{\Sigma}_N$, $\boldsymbol{\Sigma}_{N+1}$, and $\boldsymbol{\mu}_N$. This update method allows us to learn with computational cost $O(NM^2)$, which is similar to offline learning. Furthermore, this method requires less memory because it does not store the input data but only keeps $\boldsymbol{\Sigma}_N$ and $\boldsymbol{\mu}_N$.

Next, we will extend this online VFE learning to allow mini-batch learning. In other words, we consider the case in which there is more than one piece of data coming in to be updated. Let D be the set of all data up to the present, and let D^+ be the set of data to be updated. The covariance function for a set of data $D \cup D^+$ is denoted by $\boldsymbol{\Sigma}_{N+}$. Then, $\boldsymbol{\Sigma}_{N+}$ is given as follows:

$$\boldsymbol{\Sigma}_{N+} = \boldsymbol{K}_{MM} \left(\boldsymbol{K}_{MM} + \frac{1}{\sigma^2} \boldsymbol{K}_{MN} \boldsymbol{K}_{MN}^T + \frac{1}{\sigma^2} \boldsymbol{K}_{M+} \boldsymbol{K}_{M+}^T \right)^{-1} \boldsymbol{K}_{MM}, \tag{18}$$

where $[\boldsymbol{K}_{M+}]_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j^+)$, $\boldsymbol{x}_i \in D$, $\boldsymbol{x}_j^+ \in D^+$. Since $\boldsymbol{K}_{M+}$ is a matrix, the inverse of the sum of the matrices is represented in a different form.

$$\begin{aligned} \boldsymbol{\Sigma}_{N+} &= \boldsymbol{\Sigma}_N - \frac{1}{\sigma^2} \boldsymbol{\Sigma}_N \boldsymbol{K}_{MM}^{-1} \boldsymbol{K}_{M+} \boldsymbol{Q}^{-1} \boldsymbol{K}_{M+}^T \boldsymbol{K}_{MM}^T \boldsymbol{\Sigma}_N, \\ \boldsymbol{Q} &= \boldsymbol{I} + \frac{1}{\sigma^2} \boldsymbol{K}_{M+}^T \boldsymbol{K}_{MM}^{-1} \boldsymbol{\Sigma}_N \boldsymbol{K}_{MM}^{-1} \boldsymbol{K}_{M+}. \end{aligned} \tag{19}$$

Using this update formula, $\boldsymbol{\Sigma}_{N+}$ can be updated. We transform $\boldsymbol{\mu}_{N+}$ in the same way as in online learning, where it is given as follows:

$$\boldsymbol{\mu}_{N+} = \frac{1}{\sigma^2} \left(\boldsymbol{I} - \frac{1}{\sigma^2} \boldsymbol{\Sigma}_N \boldsymbol{K}_{MM}^{-1} \boldsymbol{K}_{M+} \boldsymbol{Q}^{-1} \boldsymbol{K}_{M+}^T \boldsymbol{K}_{MM}^T \right) \boldsymbol{\mu}_N + \frac{1}{\sigma^2} \boldsymbol{\Sigma}_{N+} \boldsymbol{K}_{MM}^{-1} \boldsymbol{K}_{M+} \boldsymbol{y}_+. \tag{20}$$

By using (19) and (20), we can directly obtain $\boldsymbol{\Sigma}_{N+}$ and $\boldsymbol{\mu}_{N+}$ for the set of data $D \cup D^+$ from $\boldsymbol{\Sigma}_N$ and $\boldsymbol{\mu}_N$ for the set of data D. Furthermore, the number of elements in D^+ does not have to be a fixed value during the learning process. Both online learning and mini-batch learning have the same computational complexity and return the same estimated results as offline VFE formulas (5)–(9).

We propose an algorithm for Q-learning using this mini-batch-learnable VFE method. The algorithm for learning GP based on the supervised data obtained by Q-learning has been proposed in GPQ [9]. While the GPQ is based on the sparse online Gaussian process regression algorithm [10], we use the formula proposed earlier. The proposed algorithm is presented as Algorithm 1. Lines 2–6 of Algorithm 1 are the same as in Q-learning, and equations (19) and (20) of the proposed method are used in the updating part of the value function in lines 7–10.

Q-learning with mini-batch-learnable VFE

For the first data, use the offline VFE formulas (5) through (9).
for each time step t **do**
  Choose a_t from s_t, using ε-greedy exploration
  Take action a_t, observe r_t, s_{t+1}
  $y_t = r + \gamma \max_a Q(s_{t+1}, a)$
  $x_t = \langle s_t, a_t \rangle$
  Add (y_t, x_t) to D^+
  if $|D^+| = N_{\text{batch}}$ **then**
    Compute $\boldsymbol{\mu}_{N+}$ and $\boldsymbol{\Sigma}_{N+}$ according to (20) and (19)
    Reset D^+
  end if
end for

To learn with the mini-batch-learnable VFE method, a set of inducing points must be given. The inducing points should be evenly distributed across the product set of states and actions. Kernel functions can be selected according to the environment. In our algorithm, the supervised data are generated in the same way as in normal Q-learning. The generated supervised data are then used to learn the value function in the Gaussian process regression. The batch size should be changed depending on the environment. Empirically, it is more efficient to consider a large batch size in a complex environment. The larger the batch size, the faster the learning proceeds for the number of data. However, note that too large a batch size may slow the convergence of the proposed algorithm. The computational cost of Algorithm 1 is $O(NM^2)$. Since the proposed algorithm only computes two formulas in the update, we expect the proposed algorithm to be faster than the GPQ method.

4 Experiments

In this section, we present several experiments that show that the proposed algorithm can perform as well as existing algorithms. We use two-dimensional grids [9] for our experiments. The state of this environment is represented by a 5×5 grid. Agents in this environment start at (1, 1) and can move in four directions: up, down, left, and right. The transition is noisy, i.e., with a probability of 0.1, that the agent remains in its current state. The agent can obtain reward 1 in state (5, 5). This environment has been used in previous experiments [9]. We use the GPQ method and tabular Q-learning for comparison.

In this experiment, γ is set to 0.99. For all methods, we used the ε-greedy algorithm, the exploration rate of which is $1/t^{0.3}$. The learning rate used in Q-learning is set to $0.5/t^{0.1}$. We use the same RBF kernel as $k(x_1, x_2) = \exp[-(x_1 - x_2)^2/2]$ and variance $\sigma^2 = 1$ for both methods. We set the GPQ parameter, β_{tol}, to 0.75 and the kernel budget to 25. The batch size N_{batch} is set to 16. The same estimation can be done with $N = 1$, i.e., with online learning, but the calculation time increases. The number of inducing points in the proposed method is 36, and the inducing points are arranged in a grid pattern in this experiment. This is the best combination of parameter settings for each algorithm in our experiments.

We perform ten independent experiments for each method. The average number of steps required to reach the goal is shown in Figure 1. We can see that each algorithm can learn the optimal behavior. Figure 2 shows that the resulting value estimates are similar for both the proposed method and GPQ. Numerically, there is no significant difference in the speed of convergence for each algorithm, and these algorithms perform similarly. The proposed algorithm computes only two equations without branches, and this experiment shows that for the same data size, our method terminates more than twice as fast as the GPQ method. The computational cost of the online Gaussian process regression algorithm used by GPQ is $O(N)$. Since the

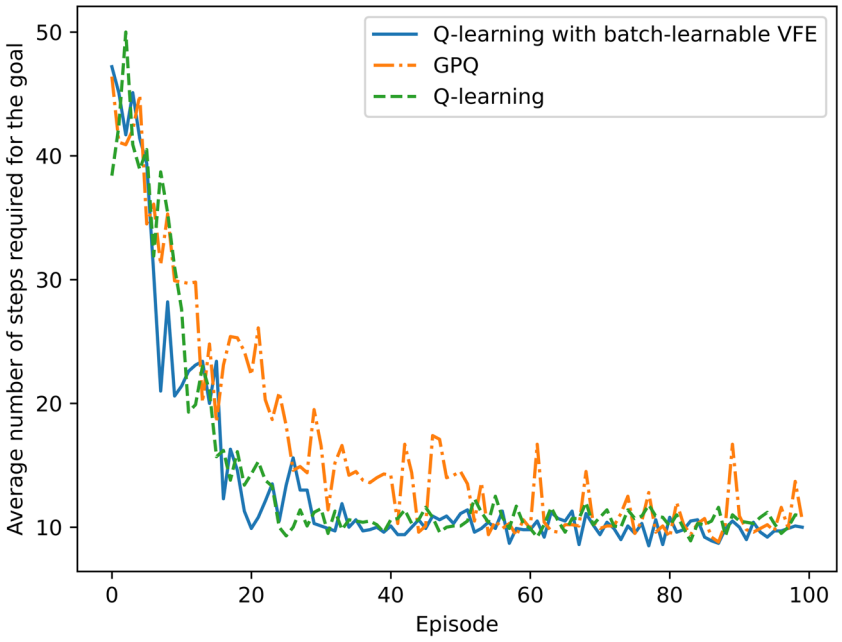

Figure 1: Average number of steps required to reach the goal over ten independent trials.

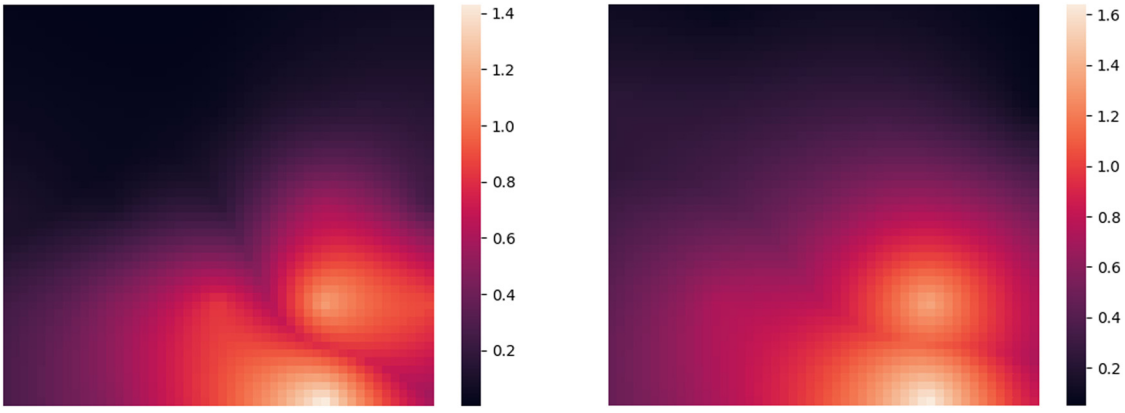

Figure 2: The resulting value estimates of the proposed method (left) and the resulting value estimates of GPQ (right) after 100 episodes. The goal is located at the bottom right.

computational cost of both methods is $O(N)$ for the number of data, the difference in execution time depends on the size of the hyperparameters and the number of formulas to be calculated.

A limitation of the proposed method comes from the selection of inducing points, which must be given before learning. In a complex problem, the performance of our algorithm depends on the set of inducing points. Increasing the number of inducing points gives a wider estimated range of states and actions, but the time required for learning is longer. This problem can be solved by changing the scale of the data if the range of states and actions is not large.

5 Conclusion

In reinforcement learning, an algorithm has been proposed in which the value function is represented by Gaussian process regression. Gaussian process regression is expected to have advantages because of its

high expressivity by kernel functions and Bayesian learning. However, the algorithms proposed in previous studies use complex online Gaussian process regression methods. We propose an online or mini-batch Gaussian process regression with VFE and inducing points for easier learning with Gaussian process regression. This method of Gaussian regression requires only the computation of two equations. We then construct a Q-learning algorithm using these two equations. Our experiments show that the algorithm can learn as well as those from previous studies.

The advantage of our algorithm is that it is easy to implement while expressing the value function in Gaussian process regression. In addition, our algorithm uses mini-batch learning, and experiments show that it can be estimated more efficiently than online learning.

Our proposed algorithm has the limitation that inducing points must be given before learning. In the case of an environment where inducing points are difficult to give, for example, when the behavior or state is high-dimensional, it becomes difficult to use our proposed algorithm.

Improving the proposed method by taking advantage of Gaussian process regression will be conducted in a future study. Even though we have been able to learn with Gaussian process regression, we have not been able to take full advantage of Gaussian process regression. Exploration for reinforcement learning is important in gathering useful data for updating the value function, and we believe that the ability to express uncertainty in the value function can be used for exploration. In addition, the choice of inducing points also affects the estimation. How inducing points are selected in reinforcement learning algorithms will also be investigated in a future study. Finally, experiments were conducted using the classical reinforcement learning environment that has been used in previous studies. Experiments in other environments should be the subject of future research.

Acknowledgment: This work was supported by JST SPRING (Grant Number JPMJSP2138), and JSPS KAKENHI (Grant Number JP19K11860).

Conflict of interest: The authors declare that they have no conflict of interest.

References

[1] Silver D, Huang A, Maddison C, Guez A, Sifre L, Driessche G, et al. Mastering the game of GO with deep neural networks and tree search. Nature. 2016;529:484–9.
[2] Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, et al. Mastering the game of GO without human knowledge. Nature. 2017;550:354–9.
[3] Rasmussen CE, Williams CKI. Gaussian processes for machine learning. Cambridge, MA, USA: MIT Press; 2006.
[4] Sutton SR, Barto GA. Reinforcement learning: an introduction. 2nd edition. Cambridge, MA, USA: MIT Press; 2018.
[5] Szepesvári C. Algorithms for reinforcement learning. Synthesis lectures on artificial intelligence and machine learning. San Rafael, CA, USA: Morgan and Claypool Publishers; 2010.
[6] Engel Y, Mannor S, Meir R. Bayes meets Bellman: The Gaussian process approach to temporal difference learning. In: International Conference on Machine Learning; 2003. p. 154–61.
[7] Engel Y, Mannor S, Meir R. Reinforcement learning with Gaussian processes. In: International Conference on Machine Learning; 2005. p. 201–8.
[8] Chung JJ, Lawrance RJN, Sukkarieh S. Gaussian processes for informative exploration in reinforcement learning. In: 2013 IEEE International Conference on Robotics and Automation; 2013. p. 2633–9.
[9] Chowdhary G, Liu M, Grande R, Walsh T, How J, Carin L. Off-policy reinforcement learning with Gaussian processes. IEEE/CAA J Automat Sinica. 2014;1(3):227–38.
[10] Csató L, Opper M. Sparse online Gaussian processes. Neural Comput. March 2002;14(3):641–68.
[11] Titsias M. Variational learning of inducing variables in sparse Gaussian processes. In: Artificial intelligence and statistics. Cambridge, MA, USA: MIT press; 2009. p. 567–74.
[12] Puterman LM. Markov decision processes: Discrete stochastic dynamic programming. Hoboken, NJ, USA: John Wiley and Sons, Inc., 1994.
[13] Watkins CJCH, Dayan P. Q-learning. In: Machine learning; 1992. p. 279–92.

[14] Quinonero-Candela J, Rasmussen CE. A unifying view of sparse approximate Gaussian process regression. J Machine Learn Res. 2005;6:1939–59.
[15] Snelson E, Ghahramani Z. Sparse Gaussian processes using pseudo-inputs. In: Advances in neural information processing systems. Cambridge, MA, USA: MIT press; 2006. p. 1257–64.
[16] Rasmussen CE, Kuss M. Gaussian processes in reinforcement learning. In: Advances in neural information processing systems. Cambridge, MA, USA: MIT press; 2003. p. 751–9.
[17] Deisenroth MP, Rasmussen CE. PILCO: a model-based and data-efficient approach to policy search. In: International Conference on Machine Learning; 2011. p. 465–72.
[18] Jung T, Stone P. Gaussian processes for sample efficient reinforcement learning with RMAX-like exploration. In: Proceedings of the European Conference on Machine Learning; 2010. p. 601–16.
[19] Bijl H, Wingerden J, Schön BT, Verhaegen M. Online sparse Gaussian process regression using FITC and PITC approximations. IFAC-PapersOnLine. 2015;48(28):703–8.