Master’s Thesis

Deep Reinforcement Learning for Self-Triggered Control

Guidance

Professor Yoshito OHTA
Assistant Professor Kenji KASHIMA

Ibuki TAKEUCHI

Department of Applied Mathematics and Physics

Graduate School of Informatics

Kyoto University

February 2021

Abstract

One of the control methods for continuous-time systems is the sample-value control. This is a control method in which the system state is observed and new control inputs are communicated at periodic intervals. The disadvantage of the sample-valued control is that it requires communication at every interval even when the control performance can be maintained without redesigning the control inputs, which results in extra cost for communication.

In recent years, event-triggered control and self-triggered control have been focused as control methods for efficient communication and control input design. First of all, event-triggered control is a control method that observes the system state at fixed time intervals as in the case of sample-value control, and redesigns and communicates the control inputs only when the driving conditions are satisfied to achieve the desired control performance. Therefore, it can improve the efficiency in terms of communication cost compared with the sample value control.

Next, self-triggered control is described. In the self-triggered control, unlike the sample-value control and the event-triggered control, the periodic state observation is not performed. Instead, the designer itself decides the next trigger time and communicates the state observation and control input after that time. For the self-triggered control, several model-based design methods have been proposed, but these methods do not explicitly consider the communication cost over a long time of control.

In this paper, we formulate an optimal self-triggered control problem where communication cost is explicitly included, which has not been considered in previous studies. Then, we consider a policy gradient method to the problem formulated in this paper.

We also propose a reinforcement learning algorithm for approximate computation of the policy gradient. As a result of the implementation, for the linear system, we can improve the policy from the control law designed naively in the model based method. We also succeeded in improving the policy from periodic control for self-triggered control of nonlinear systems, which was not solved in the previous study.

Contents

1 Introduction

One of the control methods for continuous-time systems is the sample-value control. This is a control method in which the system state is observed and new control inputs are communicated at periodic intervals. The disadvantage of the sample-valued control is that it requires communication at every interval even when the control performance can be maintained without redesigning the control inputs, which results in extra cost for communication.

In recent years, event-triggered control and self-triggered control have been focused as control methods for efficient communication and control input design. First of all, event-triggered control is a control method that observes the system state at fixed time intervals as in the case of sample-value control, and redesigns and communicates the control inputs only when the driving conditions are satisfied to achieve the desired control performance. Therefore, it can improve the efficiency in terms of communication cost compared with the sample value control. For event-triggered control, several model-based design methods introduced in [1] have been proposed, and model-free methods using reinforcement learning such as [2] have also been proposed.

Next, self-triggered control is described. In the self-triggered control, unlike the sample-value control and the event-triggered control, the periodic state observation is not performed. Instead, the designer itself decides the next trigger time and communicates the state observation and control input after that time. For the self-triggered control, model-based design methods have been proposed in [3] and [4]. However, these methods do not explicitly consider the communication cost over a long time of control.

By the way, artificial intelligence is nowadays used in various situations, notably in automatic driving technology, and the development of research on the subject of artificial intelligence is remarkable. One of the concepts to realize artificial intelligence is reinforcement learning. Reinforcement learning is an algorithm that learns behaviors that optimize the long-term benefits by repeated trial and error. In addition, although not mathematically proven, reinforcement learning has been used to obtain meaningful results for nonlinear systems. In this paper, we investigate the usefulness of reinforcement learning as a method to realize the self-triggered control law.

The two main contributions of this research are

- To formulate the optimal self-triggered control problem for long-time costs explicitly considering communication cost, and to consider the policy gradient for it.
- To confirm the usefulness of reinforcement learning for self-triggered control not only for linear systems but also for non-linear systems.

2 Preliminaries

2.1 Background of Deep Reinforcement Learning

Consider a malkov decision process M given with tuple $M = \{S, A, TP, d_0, r, \gamma\}$. Here, S, A denotes state, action set, and $TP(s^{'}|s, a)$ express transition probability. Also, $d_0, r(s, a), \gamma \in [0, 1]$ are distribution of initial state, reward, discount factor respectively.

The purpose of reinforcement learning is to find a policy such that

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, J(\pi) \tag{1}$$

where evaluation function $J(\pi)$ and (state) value function $V^\pi(s)$ is given as following:

$$V^\pi(s) = \mathbb{E}_{TP}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)|_{a_t=\pi(s_t)}, s_0 = s\right] \tag{2}$$

$$J(\pi) = \mathbb{E}_{s_0 \sim d_0}[V^\pi(s_0)] \tag{3}$$

The expectation $\mathbb{E}_{TP}$ takes over the transition probability.

Let us define Q function, which is useful tool for analyzing reinforcement learning. Q function is given as

$$\begin{aligned} Q^\pi(s, a) &= r(s, a) + \gamma \mathbb{E}_{TP}\left[\sum_{t=1}^{\infty} \gamma^t r(s_t, a_t)|_{a_t=\pi(s_t)}\right] \\ &= r(s, a) + \gamma \mathbb{E}_{TP}[V^\pi(s')]. \end{aligned} \tag{4}$$

As shown in (4), Q function express the value when the agent select action a freely and choose action according to the policy π from the next step. Thus, the Q-function is also known as the action value function.

2.2 Policy Iteration

There is an algorithm for achieving (1), called the policy iteration method. It consists of repeating the following two steps.

1. Policy Evaluation: Find (or approximate) action value function $Q^\pi(s, a)$.

2. Policy Improvement: Update policy as $\pi(s) = \underset{a}{\operatorname{argmax}}\, Q^\pi(s, a)$.

It is known that the optimal policy π^* can be obtained by repeating the above two steps (Policy Improvement Theorem).

2.3 Algorithms adapted to the settings of state and action space

In the case that both the state space and the action space take discrete values, it is easy to obtain $\pi(s) = \underset{a}{\operatorname{argmax}}\, Q^\pi(s, a)$ by storing $Q^\pi(s, a)$ in a table.

Now, what about the case where the state space is continuous? Since the state s takes a continuous value, it cannot be stored in a table. Therefore, Minh et al.[6] took the approach of approximating $Q^\pi(s,a)$ by parametrizing it using a neural network. Since the action space is discrete, it is still possible to obtain $\underset{a}{\operatorname{argmax}} Q^\pi(s,a)$.

Finally, in the case where both state and action space is continuous, the problem is that it is very expensive to obtain $\underset{a}{\operatorname{argmax}} Q^\pi(s,a)$. Thus, up to this point, the policy π has been determined by the Q-function, but this approach cannot be taken when both spaces are continuous. Therefore, the policy function is often parameterized as π_θ and the parameter θ is updated by gradient mothod.

2.4 Deterministic Policy Gradient Method

Silver et al.[7] finds the gradient for the evaluation function $J(\pi_\theta)$, even if the policy $\pi(s)$ is defined as deterministic policy. This gradient is known as deteministic policy gradient(DPG), and it is calculate as following theorem.

Theorem 1 (Deterministic Policy Gradient Theorem)**.** *The gradient for evaluation function $\nabla_\theta J(\pi_\theta)$ is exist and calculated as,*

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim\rho^{\pi_\theta}}[\nabla_\theta \pi_\theta(s) \nabla_a Q^{\pi_\theta}(s,a)|_{a=\pi_\theta(s)}] \tag{5}$$

where,

$$\rho^{\pi_\theta}(s) = \int_S \sum_{t=0}^{\infty} \gamma^t d_0(s_0) Pr(s_0 \to s, t, \pi_\theta) ds_0 \tag{6}$$

is discounted distribution.

Proof. First, we consider the gradient for $V^{\pi_\theta}(s)$.

$$\begin{aligned}
&\nabla_\theta V^{\pi_\theta}(s) \\
&= \nabla_\theta Q^{\pi_\theta}(s, \pi_\theta(s)) \\
&= \nabla_\theta [r(s,\pi_\theta(s)) + \gamma \int_S Pr(s \to s', 1, \pi_\theta) V^{\pi_\theta}(s') ds'] \\
&= \nabla_\theta \pi_\theta(s) \nabla_a r(s,a)|_{a=\pi_\theta(s)} \\
&\quad + \gamma \int_S (\nabla_\theta \pi_\theta(s) \nabla_a Pr(s \to s', 1, a)|_{a=\pi(s)} V^{\pi_\theta(s')} \\
&\quad + Pr(s \to s', 1, \pi_\theta) \nabla_\theta V^{\pi_\theta}(s')) ds' \\
&= \nabla_\theta \pi_\theta(s) \nabla_a [r(s,a) + \gamma \int_S Pr(s \to s', 1, \pi_\theta) V^{\pi_\theta}(s')]_{a=\pi_{\theta(s)}} ds' \\
&\quad + \gamma \int_S Pr(s \to s', 1, \pi_\theta) \nabla_\theta V^{\pi_\theta}(s') ds' \\
&= \nabla_\theta \pi_\theta(s) \nabla_a Q^{\pi_\theta}(s,a)|_{a=\pi_\theta(s)} + \gamma \int_S Pr(s \to s', 1, \pi_\theta) \nabla_\theta V^{\pi_\theta}(s') ds'
\end{aligned} \tag{7}$$

By using this relation recursively, we have,

$$\begin{aligned}\nabla_\theta V^{\pi_\theta}(s) &= \sum_{i=0}^{\infty} \int_S \cdots \int_S Pr(s \to s', 1, \pi_\theta) Pr(s' \to s'', 1, \pi_\theta) \cdots \\ &\qquad \gamma^i \nabla_\theta \pi_\theta(s'^{\cdots\prime}) \nabla_a Q^{\pi_\theta}(s'^{\cdots\prime}, a)|_{a=\pi_\theta(s'^{\cdots\prime})} ds'^{\cdots\prime} \ldots ds' \\ &= \sum_{i=0}^{\infty} \gamma^i \int_S Pr(s \to s', i, \pi_\theta) \nabla_\theta \pi_\theta(s) \nabla_a Q^{\pi_\theta}(s, a)|_{a=\pi_\theta(s')} ds'. \end{aligned} \tag{8}$$

Since $J(\pi) = \mathbb{E}_{s \sim d_0}[V^\pi(s)]$,

$$\begin{aligned}\nabla_\theta J(\pi_\theta) &= \nabla_\theta \int_S d_0(s) V^{\pi_\theta}(s) ds \\ &= \int_S d_0(s) \nabla_\theta V^{\pi_\theta}(s) ds \\ &= \int_S \rho^{\pi_\theta} \nabla_\theta \pi_\theta(s) \nabla_a Q^{\pi_\theta}(s, a)|_{a=\pi_\theta(s)} ds \end{aligned} \tag{9}$$

□

DDPG(Deep DPG) is a deep reinforcement learning algorithm which utilize this policy gradient. It adopts an Actor-Critic structure, and learns a critic network $Q(s, a|\omega)$ which approximates Q^{π_θ}, and an actor network $\pi(s|\theta) = \pi_\theta$ which represents a policy π, respectively. The update algorithm of actor and critic is described below.

DDPG uses mini-batch learning. First, update idea of critic is shown. The purpose of critic is to approximate Q^π. Because Q function is able to be decomposed like (4), $Q(s, a|\omega)$ should also be updated to satisfy this relation. For that, parameter ω is updated to the direction where it minimize Temporal Difference(TD) error:

$$Q(s, a|\omega) - \{r(s, a) + \gamma \mathbb{E}_{s'}[Q(s', \pi(s')|\omega)]\} \tag{10}$$

Since it is difficult to optimize for whole (s, a) at once, DDPG uses the mean squared error for the mini-batch E as the loss function and reduces it.

$$Loss = \frac{1}{N} \sum_{(s,a,s') \in E} \{Q(s, a|\omega) - (r(s, a) + \gamma Q(s', \pi(s')|\omega))\}^2 \tag{11}$$

Now, the above method of updating critic is supervised learning itself. Therefore, the data in the mini-batch must be i.i.d.. If the mini-batch E uses the data of the last N steps experienced by the agent, they are no longer independent. Hence, the agent stores the empirical data in a storage, called replay buffer, and randomly selects N data from it to make a mini-batch to increase the variance of it.

Next update law of actor are shown. Because actor is the representation of policy function $\pi(s)$, policy gradient is used for its update. However, since correct Q-function as in equation (5) cannot be used in DDPG, approximated policy gradient

$$\mathbb{E}_{s \sim \rho^\pi}[\nabla_\theta \pi_\theta(s) \nabla_a Q(s, a|\omega)|_{a=\pi_\theta(s)}] \simeq \nabla_\theta J(\pi_\theta) \tag{12}$$

is used. Furthurmore, an approximation to the expectation is made like following:

$$\mathbb{E}_{s\sim\rho^{\pi}}[\nabla_{\theta}\pi_{\theta}(s)\nabla_{a}Q(s,a|\omega)|_{a=\pi_{\theta}(s)}] \simeq \frac{1}{N}\sum_{s\in E}[\nabla_{\theta}\pi_{\theta}(s)\nabla_{a}Q(s,a|\omega)|_{a=\pi_{\theta}(s)}]. \tag{13}$$

Therefore, the accuracy of the approximation of the policy gradient may be greatly degraded by the accuracy of critic approximation and the distribution of mini-batch.

3 Problem Formulation

3.1 Self-Triggered Control

We consider control system like Fig 1.

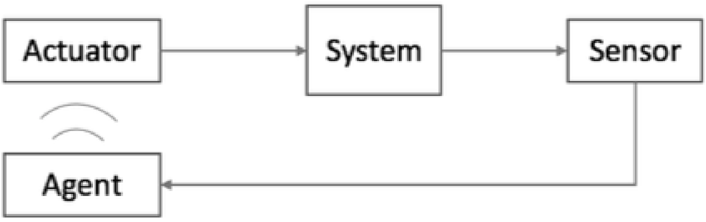

Figure 1: control system

Here, the control target is a continuous-time system as Equation (14)

$$\dot{s} = h(s,u) + \dot{w}. \tag{14}$$

Now, we consider a feedback control for system (14). In this paper, we call the observation of the state variable s and the sending of the input signal to the actuator as "interaction". In the self-triggered control, the agent does not make interaction continuously, but after a communication interval τ seconds determined by the agent itself. In order to express it mathematically, we assume that the agent's control law π is a vector-valued function consisting of two elements, where the first element represents the input u sent to the actuator, and the second element represents the interval τ seconds. The input u sent in the previous interaction is added until the time of the next interaction (Zero Order Hold control).

3.2 Previous Research for Self-Triggered Control

Previous researches in self-triggered control utilizing a Lyapunov function approach, such as [3] and [4], make one step optimization of the next triggering time on each interaction. However, this approach does not satisfy the optimality for the whole long time control episode.

In this research, we formulate an optimal self-triggered control problem in which communication is explicitly included in the cost. This makes it possible to consider the problem of finding a control policy with a long-term cost, instead of a one-step optimization.

3.3 Optimal Self-Triggered Control

In self-triggered control, the agent needs to decide the input signal u and the interval τ at each step. Thus, the action a in reinforcement learning corresponds to $\begin{bmatrix} u & \tau \end{bmatrix}^\top$. (In this paper, we equate a with the tuple (u, τ).)

Now, in this research, the control law is given as a state feedback. Therefore, the policy function is given as follows:

$$\pi(s) = \begin{bmatrix} u(s) & \tau(s) \end{bmatrix}^\top \tag{15}$$

In order to converge to the origin state as quickly as possible with the minimum input energy while reducing the frequency of communication, the agent aims to find a policy π^* that minimizes the following expected discounted cost

$$J(\pi) = \mathbb{E}_{s\sim d_0}[V^\pi(s)] \tag{16}$$

where

$$V^\pi(s) = \int_0^\infty e^{-\alpha t}\mathbb{E}_w[s(t)^\top Qs(t) + u(t)^\top Ru(t) + \beta C(t)|s(0) = s]\mathrm{d}t \tag{17}$$

and $C(t)$ is a boolean function which denotes the agent interact at time t.

If we separate the definite integral for each interval, we have

$$\begin{aligned} V^\pi(s) &= \sum_{i=0}^{\infty}\int_{t_i}^{t_{i+1}} e^{-\alpha t}\mathbb{E}_w[s(t)^\top Qs(t) + u_i^\top Ru_i + \beta C(t)|s(0) = s]\mathrm{d}t \\ &= \sum_{i=0}^{\infty} e^{-\alpha t_i}\left(\int_0^{\tau_i} e^{-\alpha t}\mathbb{E}_w[s(t)^\top Qs(t) + u_i^\top Ru_i|s(0) = s_i]\mathrm{d}t + \beta\right) \\ &= \sum_{i=0}^{\infty} e^{-\alpha t_i} r(s_i, \pi(s_i)). \end{aligned} \tag{18}$$

Here, t_i is the time of the i-th communication and s_i is the state at that time. Also, let $[u_i, \tau_i] = \pi(s_i)$, and let be the reward function $r(s_i, u_i, \tau_i)$ of each steps as

$$r(s_i, u_i, \tau_i) = \int_0^{\tau_i} e^{-\alpha t}\mathbb{E}_w[s(t)^\top Qs(t) + u_i^\top Ru_i|s(0) = s_i]\mathrm{d}t + \beta \tag{19}$$

Therefore, $V^\pi(s)$ satisfies the following Bellman equation.

$$V^\pi(s) = r(s, \pi_\theta(s)) + e^{-\alpha\tau}\mathbb{E}_{s'}[V^\pi(s'(s, \pi_\theta(s)))] \tag{20}$$

In addition, the action value function Q^π is the discounted accumulation cost when agent freely chooses an action in the first step and follows the policy π from the next step, which satisfies the following Bellman equation.

$$Q^\pi(s, u, \tau) = r(s, u, \tau) + e^{-\alpha\tau}\mathbb{E}_{s'}[Q^\pi(s'(s, u, \tau), \pi(s'(s, u, \tau)))] \tag{21}$$

4 Reinforcement Learning for Self-Triggered Control

In this section, we consider the application of reinforcement learning to find the optimal self-trigger policy π^*. Simply thinking, we can consider the reinforcement learning problem by taking the interaction as one step. Furthermore, DDPG may also be applied by approximating the Q-function, which satisfies the equation (21) using a critic network, to obtain the policy gradient. In this section, we discuss the validity of this approach.

4.1 Deterministic Policy Gradient for Self-Triggered Control

Since the discount factor in Equation (18) depends on τ at each step, it differs from the general reinforcement learning problem. In this subsection, we discuss how the DPG is affected by this difference.

Actually, due to the property of Q-functions such as (21), DPG cannot be computed as in (5). Since

$$\begin{aligned}\nabla_\theta V^{\pi_\theta}(s) &= \nabla_\theta Q^{\pi_\theta}(s, \pi_\theta(s)) \\ &= \nabla_\theta [r(s, \pi_\theta(s)) + e^{-\alpha\tau_\theta(s)}\mathbb{E}_{s'}[V^{\pi_\theta}(s')] \\ &= \nabla_\theta \pi_\theta(s)\nabla_a r(s,a)|_{a=\pi_\theta(s)} \\ &\quad + e^{-\alpha\tau_\theta(s)}\int_S \{\nabla_\theta\pi_\theta(s)\nabla_a Pr(s\to s', 1, a)|_{a=\pi(s)}V^{\pi_\theta}(s') \\ &\qquad\qquad + Pr(s\to s',1,\pi_\theta)\nabla_\theta V^{\pi_\theta}(s')\}ds' \\ &\quad + \int_S \nabla_\theta e^{-\alpha\tau_\theta(s)} Pr(s\to s', 1, \pi_\theta)V^{\pi_\theta}(s')ds', \end{aligned} \tag{22}$$

we have

$$\begin{aligned}&\nabla_\theta V^{\pi_\theta}(s) \\ &= \sum_{i=0}^{\infty}\int_S \cdots \int_S Pr(s_0\to s_1, 1, \pi_\theta)\cdots Pr(s_{i-1}\to s_i, 1, \pi_\theta) \\ &\qquad e^{-\alpha t_i}\nabla_\theta\pi_\theta(s_i)\nabla_a Q^{\pi_\theta}(s,a)|_{a=\pi_\theta(s_i)}ds_i ds_{i-1}\dots ds_1 \\ &\quad + \sum_{i=1}^{\infty}\int_S \cdots \int_S Pr(s_0\to s_1, 1, \pi_\theta)\cdots Pr(s_{i-1}\to s_i, 1, \pi_\theta) \\ &\qquad e^{-\alpha t_{i-1}}\nabla_\theta e^{-\alpha\tau_\theta(s_{i-1})}V^{\pi_\theta}(s_i)ds_i ds_{i-1}\dots ds_1 \\ &= \sum_{i=0}^{\infty}\int_S \cdots \int_S Pr(s_0\to s_1, 1, \pi_\theta)\cdots Pr(s_{i-1}\to s_i, 1, \pi_\theta) \\ &\qquad e^{-\alpha t_i}\nabla_\theta\pi_\theta(s_i)\nabla_a Q^{\pi_\theta}(s,a)|_{a=\pi_\theta(s_i)}ds_i ds_{i-1}\dots ds_1 \\ &\quad + \sum_{i=0}^{\infty}\int_S \cdots \int_S Pr(s_0\to s_1, 1, \pi_\theta)\cdots Pr(s_i\to s_{i+1}, 1, \pi_\theta) \\ &\qquad e^{-\alpha t_i}\nabla_\theta e^{-\alpha\tau_\theta(s_i)}V^{\pi_\theta}(s_{i+1})ds_{i+1} ds_i\dots ds_1, \end{aligned} \tag{23}$$

where

$$t_i = \begin{cases} 0 & (i = 0) \\ \sum_{k=0}^{i-1} \tau_\theta(s_{k-1}) & (otherwise) \end{cases}. \tag{24}$$

Now, since $J(\pi_\theta) = \mathbb{E}_{s_0 \sim d_0}[V^{\pi_\theta}(s_0)]$, we have deterministic policy gradient theorem for self-triggered control.

Theorem 2 (Deterministic Policy Gradient Theorem for Self-Triggered Control)**.** *The gradient for evaluation function* (16), (17) *is calculated as,*

$$\begin{aligned} &\nabla_\theta J(\pi_\theta) \\ &= \mathbb{E}_{s_0 \sim d_0}[\nabla_\theta V^{\pi_\theta}(s_0)] \\ &= \sum_{i=0}^{\infty} \int_S \cdots \int_S d_0(s_0) Pr(s_0 \to s_1, 1, \pi_\theta) \cdots Pr(s_i \to s_{i+1}, 1, \pi_\theta) \\ &\qquad e^{-\alpha t_i}\{\nabla_\theta \pi_\theta(s_i) \nabla_a Q^{\pi_\theta}(s, a)|_{a=\pi_\theta(s_i)} + \nabla_\theta e^{-\alpha\tau_\theta(s_i)} V^{\pi_\theta}(s_{i+1})\} ds_{i+1} ds_i \ldots ds_0. \end{aligned} \tag{25}$$

4.2 Value function for Self-Triggered Control

In DPG for reinforcement learning problems with a fixed discount factor, it is sufficient that the gradient of the critic $Q(s, a|\omega)$ with respect to a can correctly approximate that of the Q-function. However, in the case of reinforcement learning for self-triggered control considered in this paper, the value of $Q(s, \pi(s)|\omega)$ itself must also be correctly approximated. Therefore, we need to pay attention to the TD learning of critic.

In this section, we discuss whether the critic learned using TD learning can approximate the value of $Q^\pi(s, a)$. First of all, since the Bellman equation for the Q-function in self-triggered control should satisfy (21), the critic should set the TD error

$$TD = Q(s, u, \tau|\omega) - \{r(s, u, \tau) + e^{-\alpha\tau}\mathbb{E}_{s'}[Q(s'(s, u, \tau), \pi(s'(s, u, \tau))|\omega)]\} \tag{26}$$

to be zero for all (s, u, τ). In this section, we discuss the algorithm for learning such a critic.

Here, we create a mini-batch E from the dataset $D = (s, u, \tau, r, s)$ and learn critic by minimizing the MSE of the TD error for the mini-batch E. Therefore, if there is a bias of the distribution of (s, u, τ) in the empirical data set D, the accuracy of function approximation outside the distribution will obviously be low. From the equation (25), the approximation of $Q(s, \pi(s)|\omega)$ and $\nabla_a Q(s, a|\omega)|_{a=\pi(s)}$ is necessary for the calculation of the directional gradient, so we create a dataset D which contains s in the whole region of S and $[u, \tau] = \pi(s) + e$ (where e is a stochastic noise)

Algorithm 1 shows the learning algorithm for critics described in this section.
In the last part of this section, we compare $Q^\pi(s, \pi(s))$ for a self-triggered control law π with the critic $Q(s, \pi(s)|\omega)$ which approximates $Q^\pi(s, \pi(s))$ using the algorithm 1. Both $Q^\pi(s, \pi(s))$ and $Q(s, \pi(s)|\omega)$ are functions of state s. The state s is assumed to be two-dimensional, and the comparison between them is shown in Figure 2.

Algorithm 1 TD Learning for Critic Network

```
Sample $M$ states $s$ with equal probability from state space $S$.
for $r = 0$ to $R$ do
    For all sampled $s$, choose $[u, \tau] = \pi(s) + e$.
    Execute action $u$ for $\tau$ second to the environment.
    Receive $r$ and observe next state $s'$.
    Store $(s, u, \tau, r, s')$ to data set D.
end for
for epoch $= 0$ to $N$ do
    Select $m$ data pairs $(s, u, \tau, r, s')$ from $D$ and make a mini-batch $E$.
    Calculate gradient $g = \frac{\partial}{\partial \omega} \frac{1}{m} \sum_E \left(Q(s, u, \tau|\omega) - \{r + e^{-\alpha\tau} Q(s', \pi(s')|\omega)\}\right)^2$.
    Update $\omega$ with gradient $g$.
end for
```

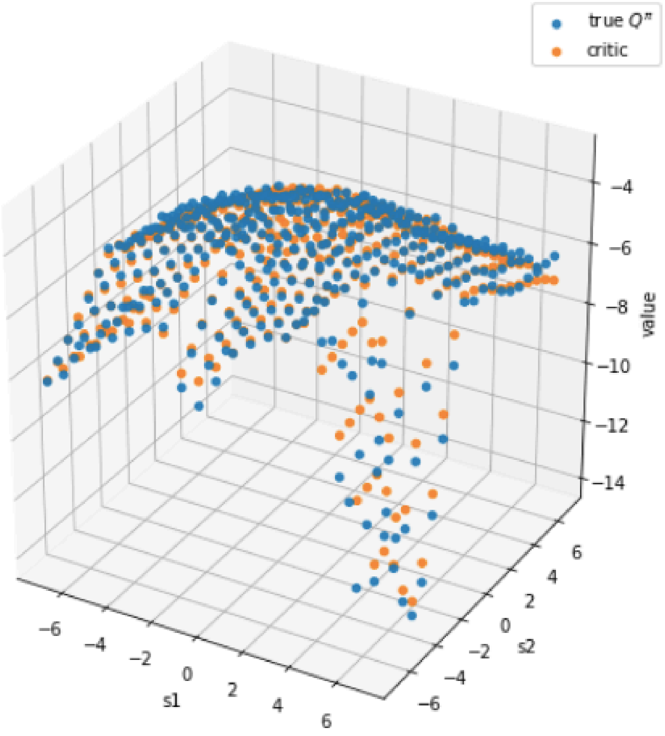

Figure 2: Approximation of $Q^\pi(s, \pi(s))$

In Figure 2, the blue points indicate the true Q^π and the orange points indicate the critics. The true Q^π is obtained by simulation. From Figure 2, we can see that the critic learned by Algorithm 1 is a good approximation of Q^π.

4.3 Ideal Algorithm

We start by considering an ideal algorithm for computing the exact policy gradient (25) without considering the computational complexity. First, for an initial state s_0, we perform P episodes of control for T seconds. In each step, we store $(s_i, u_i, \tau_i, r_i, t_i)$ in a set. The difference with DDPG is that the time at each step is also stored. When let T_i be the set $\{i \mid t_i \leq T\}$. for each control path, we can approximate $\nabla_\theta V^{\pi_\theta}(s_0)$ by computing

$$\sum_{i \in T_i} e^{-\alpha t_i} \{\nabla_\theta \pi_\theta(s_i) \nabla_a Q(s, a|\omega)|_{a=\pi_\theta(s_i)} + \nabla_\theta e^{-\alpha \tau_\theta(s_i)} Q(s_{i+1}, \pi_\theta(s_{i+1})|\omega)\} \tag{27}$$

and averaging it over P paths. We can approximate the policy gradient by generating M initial states s_0 from the initial state distribution d_0 and taking the average of this calculation for each of them. If we take P, N and M to be infinitely large, and if $Q(s, a|\omega)$ is a good approximation of $Q^{\pi_\theta}(s, a)$, we can calculate the correct policy gradient.

In algorighm 2, the reinforcement learning method with ideal calculation of policy gradient at each step.

Algorithm 2 Ideal Algorithm for Self-Triggered Control RL

```
Initialize actor $\pi_\theta(s)$ and critic $Q(s, u, \tau|\omega)$.
Make target networks $\pi_{\theta'}(s)$ and critic $Q(s, u, \tau|\omega')$ by cloning actor and critic respectively.
Learn critic $Q(s, u, \tau|\omega)$ with algorithm 1.
for $epoch = 0$ to $N$ do
  for $m = 0$ to $M$ do
    Initialize $s_0 \sim d_0$.
    for episode $= 0$ to $P$ do
      Initialize episode memory $E$.
      while $t \leq T$ do
        Select $[u, \tau] = \pi_\theta(s)$.
        Execute action $u$ for $\tau$ second to the environment.
        Receive $r$ and observe next state $s'$.
        Store tuple $(s, u, \tau, r, s', t)$.
      end while
      Calculate (27) with episode memory $E$.
    end for
    Take the average of (27) over P paths, and let it be $V^{\pi_\theta}(s_0)$.
  end for
end for
Take the average of $V^{\pi_\theta}(s_0)$ over the generated $s_0$ and let it be policy gradient $g$.
Update actor with approximated policy gradient $g$.
Update target network.
```

4.4 Practical Algorighm

As I mentioned before, the above algorithm does not take into account the problem of computational complexity. Therefore, from now on, we consider an efficient method to approximate the policy gradient. The most important point is the state distribution of the mini-batch which takes the sample mean to approximate equation (25).

Assuming that the update of the actor is very gradual, the replay buffer stores the experience gained by policies similar to the current policy. Thus, for each experience $(s_i, u_i, \tau_i, r_i, t_i)$, if we create a mini-batch E by sampling the experience with probability $e^{-\alpha t_i}$, we can expect that the sample mean

$$\frac{1}{N}\sum_{(s,s')\in E}\{\nabla_\theta \pi_\theta(s)\nabla_a Q(s,a|\omega)|_{a=\pi_\theta(s)} + \nabla_\theta e^{-\alpha\tau_\theta(s)}Q(s,\pi_\theta(s)|\omega)\} \tag{28}$$

for the mini-batch E will approximate (25) well. This is because the distribution of the mini-batch E is discounted for time t. Algorithm 3 utilize this idea.

Algorithm 3 Practical Algorithm for Self-Triggered Control RL

```
Initialize actor $\pi_\theta(s)$ and critic $Q(s,u,\tau|\omega)$.
Make target networks $\pi_{\theta'}(s)$ and $Q(s,u,\tau|\omega')$ by cloning actor and critic respectively.
for episode = 0 to $M$ do
  Initialize $s_0 \in d_0$.
  Set $i = 0, t_i = 0$.
  while $t_i \leq T$ do
    Select $[u_i, \tau_i] = \pi_\theta(s_i) + e_i$.
    Execute action $u_i$ for $\tau_i$ second to the environment.
    Receive $r_i$ and observe $s_{i+1}$.
    Store $(s_i, u_i, \tau_i, r_i, s_{i+1}, t_i)$ to the replay buffer.
    Make mini-batch $E$ considering probability $e^{-\alpha t_i}$.
    Update critic $\omega$ to decrease

      $L = \sum_{(s,u,\tau)\in E} Q(s,u,\tau|\omega) - \{r(s,u,\tau) + e^{-\alpha\tau}Q(s', \pi_{\theta'}(s')|\omega')\}.$

    Calculate approximated policy gradient using (28).
    Update actor with approximated policy gradient.
    Update target network.
  end while
end for
```

We refer to this approach as the proposed method.

5 Consideration

In this section, we study the effectiveness of the reinforcement learning approach to the optimal self-triggered control problem. We conduct numerical experiments and review the results for the cases of linear and nonlinear control systems, respectively. In both cases, the communication interval is allowed to be $0.01(s) \sim 10.0(s)$.

5.1 Evaluation Criteria

In this section, we use the valuation function $J(\pi)$ as a criterion to evaluate the policy π. $J(\pi)$ was the expectation of the value function $V^{\pi}(s)$ with respect to the initial state distribution s_0.In this paper, we assume that the initial state distribution d_0 is a uniform distribution on the state space S in both linear and nonlinear cases. Then, the state space S is discretized into a grid, and the value function $V^{\pi}(s)$ for each state s on the grid is calculated by simulation and averaged to approximate the valuation function $J(\pi)$. In order to take into account the effect of system noise, $V^{\pi}(s)$ is the average of the long-time costs of several simulations for each state s.

Note that we considered the minimization problem of the evaluation function (16) in section 3.3, but from now on, we consider it as the maximization problem of the evaluation function multiplied by -1. In other words, the evaluation value of a policy π is defined as

$$\tilde{J}(\pi) = -J(\pi), \tag{29}$$

and the higher the value, the better the policy.

5.2 Linear System

First, we adopt reinforcement learning to self-triggered control for linear system. The control object is

$$\dot{s} = As + Bu + D\dot{w} = \begin{bmatrix} -1 & 4 \\ 2 & -3 \end{bmatrix} s + \begin{bmatrix} 2 \\ 4 \end{bmatrix} u + \begin{bmatrix} 0.6 \\ 0.3 \end{bmatrix} \dot{w} \tag{30}$$

where $\dot{w}$ is wiener process noise. Here, the input signal u is limited to $-10 \sim 10$. And let the state spase be $S = \{s \in \mathbb{R}^2 | s_0 \in [-7, 7], s_1 \in [-7, 7]\}$. If an element of state s exceed the range, it will be clipped.

5.2.1 Initial Policy

For the comparison with the control performance with that of a naively designed model-based self-triggered control law, we use $\pi^{\mathrm{MB}}(s)$ such that

$$\pi_{\mathrm{MB}}(s) = \underset{u,\tau}{\operatorname{argmin}} \left\{ u^2 - \lambda\tau + {s'_e}^{\top} P s'_e + P\Sigma \right\} \tag{31}$$

where $s'_e = \mathbb{E}_w[s'(s, u, \tau)]$, $\Sigma = Var_w[s'(s, u, \tau)]$ are expectation and variance of next state respectively, and P is a unique solution of algebraic riccati equation.

In order to use π_{MB} as an initial policy for reinforcement learning, we represent π_{MB} in a neural network by supervised learning of $\pi_{\mathrm{MB}}(s)$ for M randomly generated states s in the state space S. We refer to this network as $\hat{\pi}_{\mathrm{MB}}$.

The evaluation value of $\hat{\pi}_{\mathrm{MB}}$ is

$$\tilde{J}(\hat{\pi}_{\mathrm{MB}}) \simeq -19.605850573533203. \tag{32}$$

Figure 3 shows the control path with initial policy $\hat{\pi}_{\mathrm{MB}}$ stating from $s_0 = [3., 3.]$.

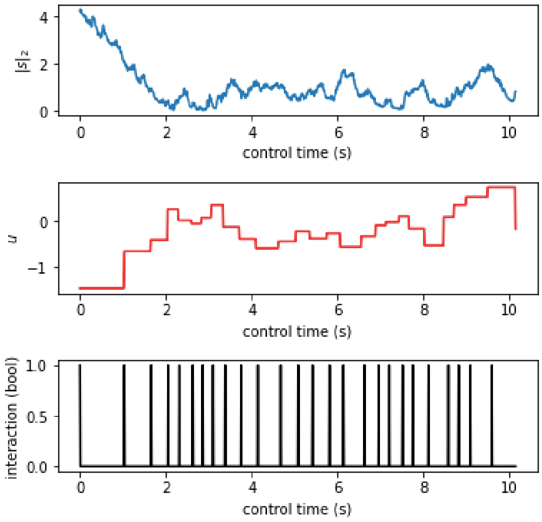

Figure 3: A control path with learned policy $\hat{\pi}_{\mathrm{MB}}$

In Figure 3, the norm of state s, the control signal u and the boolean which denotes whether agent interact with environment at time t second are shown from top to bottom.

5.2.2 Result of Proposed Method

First, we consider the results of reinforcement learning using the proposed method 1 with $\hat{\pi}^{\mathrm{MB}}$ as the initial policy. Figure 4 shows the path controlled by the policy π^{L}_{prop} from the initial state $s_0 = \begin{bmatrix} 3 & 3 \end{bmatrix}$.

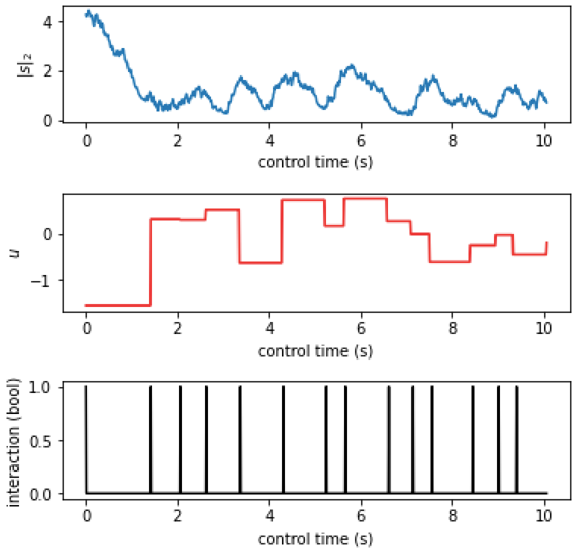

Figure 4: A control path with learned policy π^L_{prop}

The evaluation value of this policy π^L_{prop} is

$$\tilde{J}(\pi^L_{\text{prop}}) \simeq -11.986561081591695. \tag{33}$$

Thus, we can see the improvement of policy from π_{MB}.

The change of the value of the evaluation function $J(\pi_\theta)$ as the policy parameter θ is updated is shown in Figure 5.

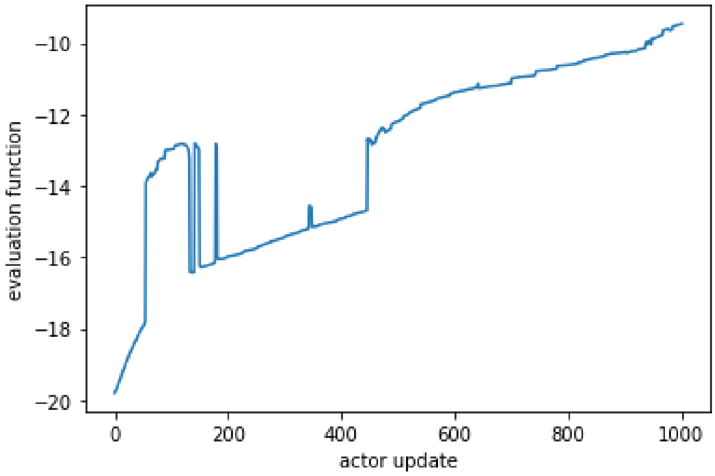

Figure 5: Policy improvement on linear case

Figure 5 shows an example of successful learning. However, since the calculation of the policy gradient depends on the approximation accuracy of the critic according to the equation (28), we often observed a sharp deterioration of the policy using the proposed method. Therefore, the learning accuracy of critic is a future work.

5.3 Non-Linear Case

In this subsection, we investigate whether the self-triggered control law can be learned by reinforcement learning even when the control target is extended to non-linear systems, especially control affine systems. We consider an inverted pendulum, whose state-space representation is

$$\frac{\mathrm{d}}{\mathrm{dt}}\begin{bmatrix}\theta \\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}\dot{\theta} \\ \frac{3g}{2l}\sin\theta + \frac{3}{ml^2}a\end{bmatrix} + \begin{bmatrix}1 \\ 1\end{bmatrix}\dot{w}. \tag{34}$$

where $\dot{w}$ is wiener process noise. Therefore, for an inverted pendulum, the state variable s is considered to be $\begin{pmatrix}\theta & \dot{\theta}\end{pmatrix}^\top$.

As in the linear case, the input signal u is limited to $-10 \sim 10$. And let the state spase be $S = \{s \in \mathbb{R}^2 | s_0 \in [-\pi, \pi], s_1 \in [-2\pi, 2\pi]\}$. If angle $\|\theta\| > \pi$, θ will be replaced by the equivalent angle which satisfies $\|\theta\| \leq \pi$. And if Angular velocity $\|\dot{\theta}\| > 2\pi$, $\|\dot{\theta}\|$ will be clipped.

5.3.1 Initial Policy

The initial policy π_{init} used in this case is

$$\pi_{\text{init}}(s) = \begin{bmatrix}-Ks & 0.2\end{bmatrix} \tag{35}$$

where K is a feedback gain calculated by Linear Quadratic Regulator for linearized system around $s = \mathbf{0}$. Figure 6 shows the control path with initial policy π_{init} stating from $s_0 = [3., 3.]$.

The evaluation value of π_{init} is

$$\tilde{J}(\pi_{\text{init}}) \simeq -62.492721990335504. \tag{36}$$

In Figure 6, the angle of pendulum θ rad, the torque u N·m and the boolean which denotes whether agent interact with environment at time t second are shown from top to bottom.

5.3.2 Result of Proposed Method

First, we show the results of reinforcement learning by the proposed method 1. Figure 7 shows the control path by the obtained policy π_{prop}^N stating from $s_0 = [3., 3.]$.

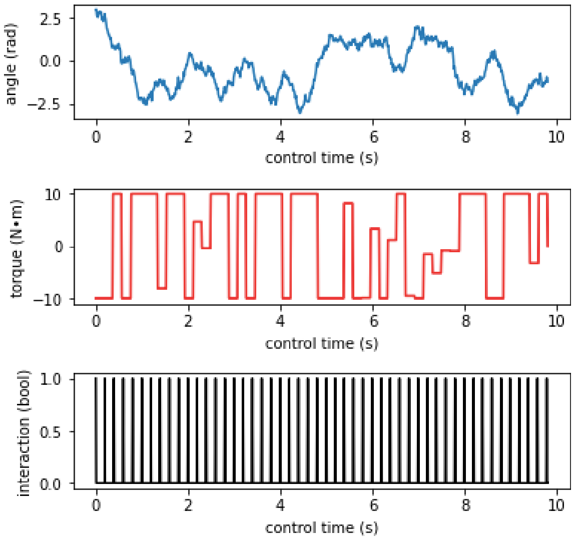

Figure 6: A control path with initial policy π_{init}

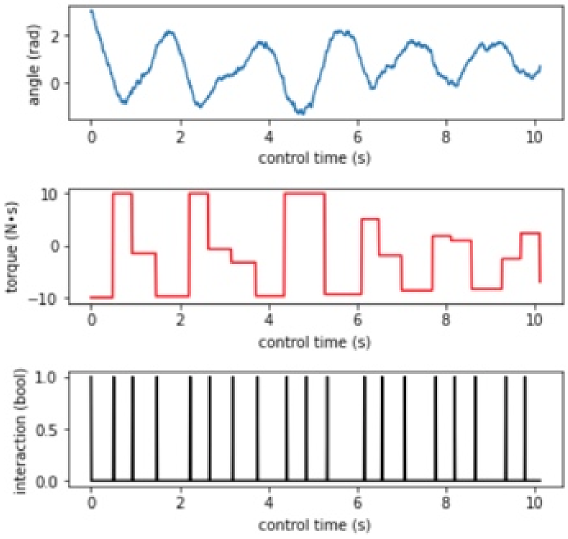

Figure 7: A control path with learned policy π^N_{prop}

The evaluation value of π^N_{prop} is

$$\tilde{J}(\pi^N_{\text{prop}}) \simeq -33.49714598533895. \tag{37}$$

Thus, we can confirm the improvement of policy.

The change of the value of the evaluation function $J(\pi_\theta)$ as the policy parameter θ is updated is shown in Figure 8.

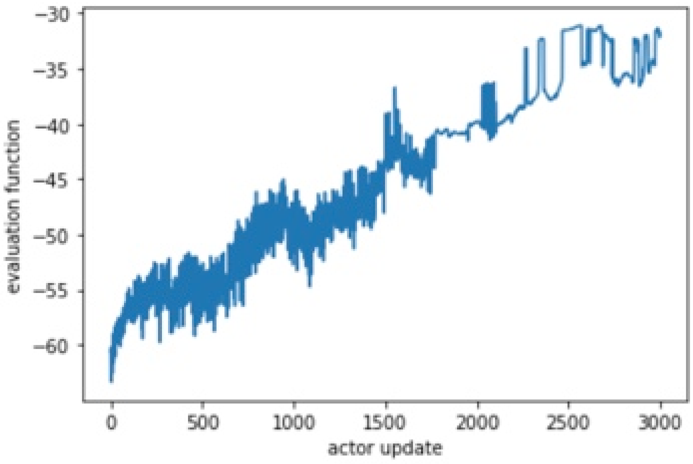

Figure 8: Policy improvement on non-linear case

6 Conclusion

In this paper, we formulate an optimal self-triggered control problem where communication cost is explicitly included, which has not been considered in previous studies. Then, we consider a reinforcement learning approach to the problem.

First, from the configuration of the evaluation function, we confirm that the deterministic policy gradient theorem for general reinforcement learning is not directly applicable, and then we derive a policy gradient theorem that is compatible with the formulated optimal self-triggered control problem.

In this paper, we also propose a reinforcement learning algorithm for approximate computation of the policy gradient. As a result of the implementation, for the linear system, we can improve the policy in the sense of the formulated evaluation function for the control law designed naively in the model base. We also succeeded in improving the policy for self-triggered control of nonlinear systems, which was not solved in the previous study.

However, the computational complexity and the way of saving the empirical data are important issues to be solved in the future, because they greatly affect the results of the calculation of the policy gradient.

Acknowledgments

The author would like to express his sincere gratitude to Professor Yoshito Ohta and Assistant Professor Kenji Kashima for their helpful advices.

References

[1] W. P. M. H. Heemels, K. H. Johansson, and P. Tabuada. “ An intro- duction to event-triggered and self-triggered control. ” *In Proc. of the 51st IEEE International Conference on Decision and Control*, 2012.

[2] D. Baumann, J. J. Zhu, G. Martius, and S. Trimpe. “Deep Reinforcement Learning for Event-Triggered Control. ” *In Proc. of the 57th IEEE International Conference on Decision and Control*, 2018.

[3] T. Gommans, D. Antunes, T. Donkers, P. Tabuada, and M. Heemels. “Self-triggered linear quadratic control. ” *Automatica*, vol. 50, no. 4, pp. 1279-1287, 2014.

[4] G. Yang, C. Belta, and R. Tron. “ Self-triggered Control for Safety Critical Systems Using Control Barrier Functions.” *In American Control Conference (ACC) Philadelphia, USA*, 2019.

[5] C. J. Watkins, and P. Dayan. Q-learning. *Machine Learning*, vol. 8, no. 3-4, pp. 279-292, 1992.

[6] V. Minh, K. Kavukcouglu, D. Silver. et al.. “ Human-level control through deep reinforcement learning.” *Nature 518*, pp.529-533, 2015.

[7] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, et al.. “ Deterministic Policy Gradient Algorithms.” *ICML Beijing, China.*, 2014, Beijing.

[8] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N.Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. *International Conference on Learning Representations*, 2015.

[9] I. Grondman, L. Busoniu, G. A. Lopes, and R. Babuska. “ A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradient.” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291-1307, 2012.

[10] T. Degris, M. White and R. Sutton. “ Off-Policy Actor-Critic.” *ICML Edinburgh, United Kingdom*, 2012.

[11] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. ”Policy gradient methods for reinforcement learning with function approximation.” *In Advances in Neural Information Processing Systems*, 2000.

[12] D. P. Kingma and J. Ba. “ Adam: A Method for Stochastic Optimization.” *arXiv preprint arXiv: 1412.6980*, 2014.

A Appendix

A.1 Model Settings

We use DDPG as reinforcement learning algorithm. As described in section 2, actor and critic is expressed as neural networks respectively. Experiments have shown that learning diverges when using a general network, so here we use the special network. The architecture of 2 networks is in Fig 9.

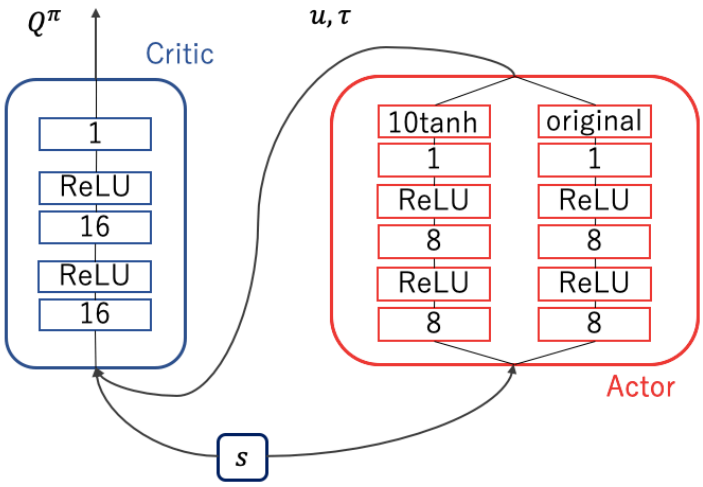

Figure 9: Agent Model

The activation function ”original” shown in Fig 9 is defined as $0.99 \times \text{sigmoid} + 0.01$ to meet upper and lower limits of interval described in the next section.

Master's Thesis

Deep Reinforcement Learning for Self-Triggered Control

Guidance

Professor	Yoshito OHTA
Assistant Professor	Kenji KASHIMA

Ibuki TAKEUCHI

Department of Applied Mathematics and Physics

Graduate School of Informatics

Kyoto University

February 2021

Deep Reinforcement Learning for Self-Triggered Control

Ibuki TAKEUCHI

February 2021

Deep Reinforcement Learning for Self-Triggered Control

Ibuki TAKEUCHI

Abstract

One of the control methods for continuous-time systems is the sample-value control. This is a control method in which the system state is observed and new control inputs are communicated at periodic intervals. The disadvantage of the sample-valued control is that it requires communication at every interval even when the control performance can be maintained without redesigning the control inputs, which results in extra cost for communication.

In recent years, event-triggered control and self-triggered control have been focused as control methods for efficient communication and control input design. First of all, event-triggered control is a control method that observes the system state at fixed time intervals as in the case of sample-value control, and redesigns and communicates the control inputs only when the driving conditions are satisfied to achieve the desired control performance. Therefore, it can improve the efficiency in terms of communication cost compared with the sample value control.

Next, self-triggered control is described. In the self-triggered control, unlike the sample-value control and the event-triggered control, the periodic state observation is not performed. Instead, the designer itself decides the next trigger time and communicates the state observation and control input after that time. For the self-triggered control, several model-based design methods have been proposed, but these methods do not explicitly consider the communication cost over a long time of control.

In this paper, we formulate an optimal self-triggered control problem where communication cost is explicitly included, which has not been considered in previous studies. Then, we consider a policy gradient method to the problem formulated in this paper.

We also propose a reinforcement learning algorithm for approximate computation of the policy gradient. As a result of the implementation, for the linear system, we can improve the policy from the control law designed naively in the model based method. We also succeeded in improving the policy from periodic control for self-triggered control of nonlinear systems, which was not solved in the previous study.