Master's Thesis

# Deep Reinforcement Learning for Optimal 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**

In this thesis, we construct the mathematical models of students who write the master's thesis and develop efficient algorithms for writing the master's thesis based on the models. We show that the proposed algorithms generate the thesis 65536 times more efficiently than writing by oneself.

# Contents

# 1 Introduction

# 2 Preliminaries

## 2.1 Background of Deep Reinforcement Learning

Consider a malkov decision process $M$ given with tuple $M = \{S, A, T, d_0, r, \gamma\}$. Here, $S, A$ denotes state, action set, and $T(s^{'}|s, a)$ express transition probability. Also, $d_0, r(s, a), \gamma \in [0, 1]$ are distribution of initial state, reward, discount factor respectively.

Now, 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:

$$V^\pi(s) = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)|_{a_t=\pi(s_t)}, s_0 = s \tag{2}$$

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

Here, we define $Q$ function, which is useful tool for analyzing reinforcement learning. $Q$ function is given as

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

As showed in (4), $Q$ function express the value when agent select action $a$ freely and choose action according to the policy $\pi$ from 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 in 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 $\pi^*$ can be obtained by repeating the above two steps (Measure 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 $\pi(s) = \underset{a}{\operatorname{argmax}}\, Q^{\pi}(s,a)$, which appeared in the previous section, 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.[2] 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 whrere 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 $\pi$ 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 $\pi_\theta$ and the parameter $\theta$ is updated by gradient ascent.

### 2.4 Deterministic Policy Gradient Method

Silver et al.[3] 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:

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

where,

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

is discounted distribution.

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 $Q^{\pi_\theta}$, and an actor network $\pi(s|\theta) = \pi_\theta$ which represents a measure $\pi$, 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 $Q^{\pi}$. 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 $\omega$ is updated to the direction where it minimize Temporal Difference(TD) error:

$$\mathrm{TD} = Q(s,a|\omega) - \{r(s,a) + \gamma Q(s,\pi(s)|\omega)\} \tag{7}$$

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.

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 the mini-batch.

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

$$\mathbb{E}_{s\sim\rho^\pi}[\nabla_\theta \pi_\theta(s) \nabla_a Q(s,a|\omega)|_{a=\pi_\theta(s)}] \simeq \nabla_\theta J(\pi_\theta) \tag{8}$$

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

$$\mathbb{E}_{s\sim\rho^\pi}[\nabla_\theta \pi_\theta(s) \nabla_a Q(s,a|\omega)|_{a=\pi_\theta(s)}] \simeq \frac{1}{N}\sum_{s\in E}[\nabla_\theta \pi_\theta(s) \nabla_a Q(s,a|\omega)|_{a=\pi_\theta(s)}]. \tag{9}$$

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. Here, the control target is a continuous-time

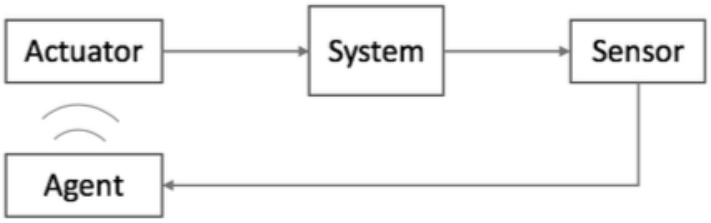


Figure 1: control system

system as in Equation (10)

$$\dot{s} = h(s,u) \tag{10}$$

Now, we consider a feedback control for system (10). 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 $\tau$ seconds determined by the agent itself. In order to express it mathematically, we assume that the agent's control law $\pi(s)$ 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 $\tau(sec)$. The input $u$ sent in the previous interaction is added until the time of the next interaction (ZOH control).

### 3.2 Optimal Self-Triggered Control

The optimal self-triggered control law $\pi^*$ is defined as follows

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\, J(\pi) \tag{11}$$

$$J(\pi) = \mathbb{E}_{s_0 \in d_0}[V^\pi(s_0)] \tag{12}$$

$$V^\pi(s_0) = \sum_{i=0}^{\infty} \gamma^i r_i^\pi \tag{13}$$

$$r_i^\pi = -\int_{T_i}^{T_{i+1}} s(t)^\top Q s(t) \mathrm{d}t + \tau_i a_i^\top R a_i + \lambda \tau_i,\ T_i = \sum_{l=0}^{i} \tau_l \tag{14}$$

where $i$ denote the number of interactions, and $a_i$ and $\tau_i$ be the outputs of the measures $\pi$ for the $i$th interaction, respectively.

## 4 Reinforcement Learning for Optimal Self-Triggered Control

In reinforcement learning, the agent is expected to perform interactions step by step. The self-triggered control problem considered in this paper, where the control target is given as a continuous-time system, satisfies the assumption of reinforcement learning by considering the communication with the actuator as one step.

### 4.1 Model Settings

In the self-triggered control, the agent needs to decide the input signal $u$ and the interval $\tau$ at each step. In this paper, the control law is given as a state feedback. Therefore, the policy function is given as follows:

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

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 2.
The activation function "original" shown in Fig 2 is defined as $0.99 \times \text{sigmoid} + 0.01$ to meet upper and lower limits of interval.

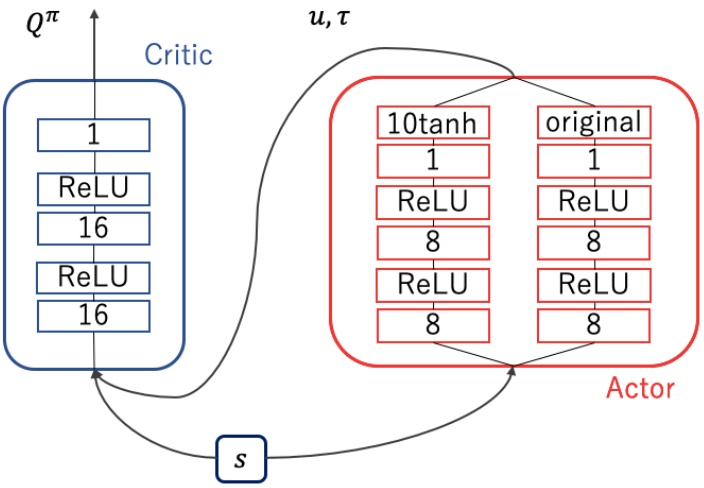


Figure 2: Agent Model

# 5 Consideration

## 5.1 Linear Case

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

$$\dot{s} = As + Bu = \begin{bmatrix} -1 & 4 \\ 2 & -3 \end{bmatrix} s + \begin{bmatrix} 2 \\ 4 \end{bmatrix} u. \tag{16}$$

### 5.1.1 Initial Policy

For learning efficiency, we use $\pi_{\text{init}}$ such that

$$\pi_{\text{init}} = \begin{bmatrix} -Ks & 0.01 \end{bmatrix} \tag{17}$$

where $K$ is a feedback gain calculated by Linear Quadratic Regulator. This policy stabilize the system (really??).

### 5.1.2 Learned Policy

## 5.2 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

$$\frac{\mathrm{d}}{\mathrm{dt}} \begin{pmatrix} \theta \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} \dot{\theta} \\ \frac{3g}{2l} \sin\theta + \frac{3}{ml^2} a \end{pmatrix}. \tag{18}$$

Therefore, for an inverted pendulum, the state variable $s$ is considered to be $\begin{pmatrix}\theta\\ \dot{\theta}\end{pmatrix}$.

### 5.2.1 Initial Policy

For learning efficiency, we use $\pi_{\text{init}}$ such that

$$\pi_{\text{init}} = \begin{bmatrix}-Ks & 0.2\end{bmatrix} \tag{19}$$

where $K$ is a feedback gain calculated by Linear Quadratic Regulator. This policy stabilize the system (really??).

### 5.2.2 Learned Policy

Fig 3 shows a control path with learned policy, stating from $s_0 = [3., 3.]$.

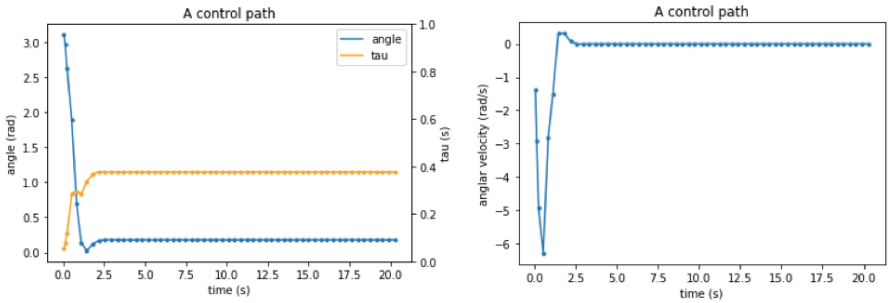


Figure 3: A control path with learned policy $\pi_{\text{RL}}$

The left panel of Figure 3 plots the change of angle and communication interval $\tau$ against the control time. The time of the communication is marked with a dot. From this figure, we can see that the communication interval is determined flexibly at each interaction.

Now, we confirm whether the learned policy $\pi_{\text{RL}}$ enlarge the evaluation function $J(\pi)$ compared from the initital policy $\pi_{\text{init}}$. Since $J(\pi) = \mathbb{E}_{s_0}[V^{\pi}(s_0)]$, we generate 500 initial states $s_0$ according to the initial state distribution $d_0$, and average the results of $V^{\pi}(s_0)$. We conduct this approximation of expectation 1000 times. From the result

$$J(\pi_{\text{init}}) = 4.23 \pm 0.083, \;\; J(\pi_{\text{RL}}) = 37.12 \pm 0.071, \tag{20}$$

we can confirm $J(\pi_{\text{init}}) < J(\pi_{\text{RL}})$. It can be concluded that the policy has been improved.

## 5.3 Optimality

Is the policy $\pi_{\text{RL}}$ described in the previous section an optimal? If it is an optimal solution, the policy gradient must be $\mathbf{0}$. Therefore from equation (5), a condition

$$\nabla_a Q^{\pi_{\text{RL}}}(s,a)|_{a=\pi_{\text{RL}}(s)} = \mathbf{0} \tag{21}$$

should be met.

Now, during the reinforcement learning process, the policy keeps changing at each step, so $Q^{\pi}(s,a)$, which is the approximation target of critic, also keeps changing. In addition, the empirical data used for critic's learning includes data of experience with policies different from the one to be evaluated. Thus, there is no guarantee that $Q^{\pi_{\mathrm{RL}}}(s,a)$ is approximated correctly at each step. Therefore, we learned $Q^{\pi_{\mathrm{RL}}}(s,a)$ by supervised learning of critic using only the data generated by interaction with a fixed policy. Let $Q^{\pi_{\mathrm{RL}}}_{\mathrm{ap}}(s,a)$ denotes this approximated function.

If $\pi_{\mathrm{RL}}$ is the optimal policy, $\nabla_a Q^{\pi_{\mathrm{RL}}}_{\mathrm{ap}}(s,a)|_{\pi_{\mathrm{RL}}}$ should be $\mathbf{0}$. However, when this is calculated in practice, we have

$$\nabla_a Q^{\pi_{\mathrm{RL}}}_{\mathrm{ap}}(s,a) = \left[0.03, 0.07\right]. \tag{22}$$

Therefore, we find $\pi_{\mathrm{RL}}$ is not the optimal policy.

### 5.4 方策の最適性

前節で述べた方策 $\pi_{\mathrm{RL}}$ は最適解なのだろうか. もし最適解であれば方策勾配が $\mathbf{0}$ となる必要があるため, 式 (5) より,

$$\mathbb{E}_{s\sim\rho^{\pi_{\mathrm{RL}}}}[\nabla_\theta \pi_{\mathrm{RL}}(s) \nabla_a Q^{\pi_{\mathrm{RL}}}(s,a)|_{a=\pi_{\mathrm{RL}}(s)}] = \mathbf{0} \tag{23}$$

となるべきである.

方策 $\pi_{\mathrm{RL}}$ が式 (23) を満たしているかを調査するために, $Q^{\pi_{\mathrm{RL}}}(s,a)$ を求めたい. そこで, 価値関数 $V^{\pi_{\mathrm{RL}}}(s) = \sum_{i=0}^{\infty}$ の近似値をシミュレーションによって以下のように求め,

$$V^{\pi_{\mathrm{RL}}}_{\mathrm{ap}}(s) = \sum_{i=0}^{N} \gamma^i r(s_i, \pi_{\mathrm{RL}}(s_i)) \tag{24}$$

$V^{\pi_{\mathrm{RL}}}_{\mathrm{ap}}$ を用いて $Q^{\pi_{\mathrm{RL}}}$ の近似値を以下のようにして求める.

$$Q^{\pi_{\mathrm{RL}}}_{\mathrm{apd}}(s,a) = r(s,a) + \gamma V^{\pi_{\mathrm{RL}}}_{\mathrm{ap}}(s') \tag{25}$$

$\gamma < 1$ なので, シミュレーションのステップ数 $N$ を大きく取っておけば, $Q^{\pi_{\mathrm{RL}}}_{\mathrm{apd}}$ はいい近似となる. ただし, 方策勾配の計算には $\nabla_a Q^{\pi_{\mathrm{RL}}}(s,a)$ を計算する必要があるが, $Q^{\pi_{\mathrm{RL}}}_{\mathrm{apd}}$ は $(s,a)$ における値を計算することしかできない. そこで, 適当な関数近似器を $Q^{\pi_{\mathrm{RL}}}_{\mathrm{apd}}(s,a)$ を教師データとして教師あり学習することで関数 $Q^{\pi_{\mathrm{RL}}}$ を近似する. この関数を $Q^{\pi_{\mathrm{RL}}}_{\mathrm{ap}}$ とする.

さて, $Q^{\pi_{\mathrm{RL}}}_{\mathrm{ap}}$ を用いて方策勾配を計算すると

$$\mathbb{E}_{s\sim\rho^{\pi_{\mathrm{RL}}}}[\nabla_\theta \pi_{\mathrm{RL}}(s) \nabla_a Q^{\pi_{\mathrm{RL}}}(s,a)|_{a=\pi_{\mathrm{RL}}(s)}] \neq \mathbf{0} \tag{26}$$

となり, 方策 $\pi_{\mathrm{RL}}$ は最適方策ではないことがわかった.

## 6 Conclusion

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

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

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

[4] 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.

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

[6] 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.

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


# A Appendix

This is an appendix. This is a citation [**?**].

Table 1: This is a table.

| | A | B |
|---|---|---|
| C | 70 | 80 |
| D | 100 | 0 |

Master's Thesis

# Deep Reinforcement Learning for Optimal 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 Optimal Self-Triggered Control

Ibuki TAKEUCHI

**Abstract**

In this thesis, we construct the mathematical models of students who write the master's thesis and develop efficient algorithms for writing the master's thesis based on the models. We show that the proposed algorithms generate the thesis 65536 times more efficiently than writing by oneself.