Information Modelling and Knowledge Bases XXXII
M. Tropmann-Frick et al. (Eds.)

doi:10.3233/FAIA200839

Q-Learning as Failure

Kei TAKAHATA, and Takao MIURA
Dept.of Advanced Sciences, HOSEI University
Kajinocho 3-7-2, Koganei, Tokyo, Japan
Email:kei.takahata.6a@stu.hosei.ac.jp miurat@k.hosei.ac.jp

Abstract. Reinforcement Learning allows us to acquire knowledge without any training data. However, for learning it takes time. In this work, we propose a method to perform Reverse action by using Retrospective Kalman Filter that estimates the state one step before. We show an experience by a Hunter Prey problem. And discuss the usefulness of our proposed method.

Keywords. Reinforcement Learning, Q-Learning, Kalman Filter, Retrospective Kalman Filter, Reverse Action Learning

1. Introduction

Reinforcement Learning (RL) [1] [2] is a learning method in which any agent learns from interaction with its environment. It allows us to acquire knowledge without any training data. However, for learning it takes time. In a case of a robot, it takes heavy time to perform many experiences. Therefore, various methods aiming at reducing the amount of learning experiences have been proposed [4] [5]. These put focus on how to acquire quality knowledge efficiently. Unfortunately, there are few methods to improve knowledge proposed so far by using failure. For example in Fig.1, suppose that the agent acts twice to the left from the center position in the initial state and gets a reward of -100 where the circles in Figure 1 represent states, and the numbers represent rewards. Is it possible to improve knowledge by other choices? By the word "Reverse Action Learning(RAL)", we mean a learning method that agents select a reverse action and receive reverse rewards. In the case of Fig.1, the number of states and actions is small, therefore it could be possible to examine all the choices.However, in the case of complex tasks, it is not practical to keep all information. Therefore, the goal of this work is to propose RAL.

Retrospective behavior could have two interpretation, "compensation behavior" or "reverse behavior". "Compensation"[WIKI] refers to a type of defense mechanism in which people overachieve in one area to compensate for failures in another. So one can cover up, consciously or unconsciously, weaknesses, frustrations, desires, or feelings of inadequacy or incompetence in one life area through the gratification or (drive towards) excellence in another area. On the other hand, "reverse action" means a type of recovery mechanism whereby people can restore almost all the status back into the previous one to keep the environment consistently in any means of the situation before these actions.

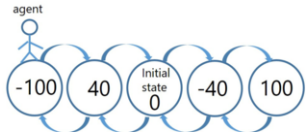

Figure 1. Example of state transition and rewards

Very often they differ from each other since compensation could be made by similar actions without any consideration of environment. The story looks like theory of transaction in databases [6]. We'd better say "roll back" for "reverse action". Here we stick to the situation "reverse" (or "roll back") behaviors since we might take strategic decisions every time we have to decide.

In this work, we model a method to perform RAL by using a Retrospective Kalman filter that estimates the state one step before. We discuss an experience by a Hunter Prey problem to see we show the usefulness of our proposed method.

The rest of the paper is organized as follows. In section 2 we give a denition of Reinforcement Learning. Section 3 discusses a background of Kalman filters for our learning. In section 4 we propose our approach of and section 5 concerns about experimental results to see how effective our approach works. In section 6 we conclude this investigation.

2. Reinforcement Learning and Q-Learning

2.1. Reinforcement Learning

Reinforcement learning (RL) is a learning method in which the agents obtains knowledge from interaction with its environment. Agents perform state perception and make a dicision. In RL, we don't give explicit correct answers and agents learn from rewards under the environment. Here, rewards include positive and negative. Agents goal is to find a policy that maximizes the total rewards.

An agent perceives its *current state* (or position at time t), selects an *action* for its behavior, to obtains *reward*, and then changes itself to the *next state*. When an agent takes an action a_t at time t, the next state s_{t+1} and the reward r_{t+1} at time $t+1$ depend on all the previous states and all the rewards in the agent history. Let s' and r be one of the possible next states and the rewards, then s_{t+1} and r_{t+1} can be described by means of conditional probabilities:

$$Pr\{s_{t+1} = s', r_{t+1} = r | s_t, a_t, r_t, s_{t-1}, \cdots, r_1, s_0, a_0\} \tag{1}$$

Markov probability model is a stochastic framework to model randomly changing systems. Here next state depends only on the current state, not on the previous states nor actions before. This is called *Markov property*. Generally this assumption enables reasoning and computation with the model efficient. In our case we must have:

$$Pr\{s_{t+1} = s', r_{t+1} = r | s_t, a_t\} \tag{2}$$

Assuming current state s_t and an action a_t under the assumption of Markov property, we can estimate the next state s_{t+1} and the reward r_{t+1} in a probabilistic manner as shown

in a formula (2). Repeating this process of (2), we may obtain all the future states, the actions and the whole rewards. In RL, it is possible to say that behaviors and the *value functions* (described later) depend only on current state.

Let us define a notion of *policy* that consists of all the pairs $(a, p(a))_s$ of action a and its probability $p(a)$ to each state s. The main goal of agents in RL is to obtain a policy π to solve issues of interests efficiently, or, in our case, to maximize the whole rewards in its life-time. To do that, we introduce value functions. Since we discuss Q-Learning (QL) [8], we define *action-value* functions called the Q-value. Q-value represents the expected value of whole rewards under the certain policy π.

2.2. Q-Learning

QL is a represemtative learning method of RL. By $Q(s,a)$, let us define the Q-value (the expected all the sum of rewards) at a state s with an action a. Formally let $r = r(s,a)$ be a reward at (s,a), $s' = Next(s,a)$ a next state to s. Also let a' be a next action at s' of the maximim $Q(s',a')$, that is, $a' = \max_{a' \in A(s')} Q(s',a')$ where $A(s')$ a set of possible actions at s'. Then let α be *learning rate* ($0 \leq \alpha \leq 1$), which means how large one learning step improves Q-values, and γ be *discount rate* which means the effect of valuing rewards received earlier higher than those received later. This may also be interpreted as the probability to succeed (or survive).
At every learning step, we keep updating Q-values as follows:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[\, r + \gamma \max_{a' \in A(s')} Q(s',a') - Q(s,a)]$$

Note that we update the Q-values so as to make $Q(s,a)$ close to $r + \gamma \max_{a' \in A(s')} Q(s',a')$.

Watkins [8] has shown the convergence of the Q-values if both learning rate and discount rate satisfy some constraints.

2.3. Reverse Action Learning

Let us discuss a case in which an agent receives a large negative reward. We assume that the reverse action allows us to improve the current situation. Under this assumption, we update efficiently the Q-value by Reverse Action Learning(RAL). If an agent recieves a large negative reward, an agent reduce the value taken to the negative reward actions by normal learning process. Moreover, an agent could improve the value of reverse actions by RAL process, that is, we expect that an agent learn how to avoid these actions.

3. Kalman Filter

Kalman filter (KF) is one of the well-known algorithm that estimates the state of a system from observation data with some noises. During KF process, whenever a new (temporal) observation data comes in, we improve estimation of state immediately. We estimate *a priori estimate* of a state X_k at time k, denoted by $\hat{X}_k^-$, by examining $X_{k-1}, X_{k-2}, \ldots$ Similarly we estimate *a posteriori estimate* of X_k, denoted by $\hat{X}_k$, by examining $Y_k, X_{k-1}, X_{k-2}, \ldots$ Given a covariance matrix P of state errors, we can think about a

priori estimate P_{k-1}^- and a posteriori estimate P_k. Let us note that by minimizing P_k (using minimum mean-square error), we can improve the precision of the estimation.

Now assume we have Kalman gain matrix G_k at k, there happen filtering processes:

EstimateStep	
$\hat{X}_k^-$	$= A\hat{X}_{k-1}$
P_k^-	$= AP_{k-1}A^T + BQB^T$
FilteringStep	
G_k	$= P_k^- C^T (CP_k^- C^T + R)^{-1}$
$\hat{X}_k$	$= \hat{X}_k^- + G_k(Y_k - C\hat{X}_k^-)$
P_k	$= (I - G_k C)P_k^-$

Let us remark that we have to give initialization values of $\hat{X}_{k-1}$ initial states, P_{k-1} initial a priori errors covariance, and Q, R noises covariance.

The KF improves the accuracy of state estimate by reducing the error covariance. *Kalman gain* represents the rate at which the state is updated from observations. For example, when a priori estimate covariance is large (a priori state estimate is not reliable) and when the observation noise is small (the observed value is reliable), a priori state estimate is largely updated because the observed value is more reliable. Therefore, the Kalman gain also increases. Meanwhile, when a priori estimate covariance is small and the observation noise is large, the Kalman gain is small because the state transition is more reliable than the observed values.

KTD [13] [12] has been proposed as a method for estimate parameters to RL over continuous states. KTD has a problem that it depends on initial parameters. Our proposed method uses Kalman filter for agent action selection, so it is fundamentally different from KTD.

4. Proposed method

4.1. Retrospective Kalman Filter

Now let us propose a new method to learn efficiently by reverse actions due to the Q-value update. We put a Failure Condition on our agents. Whenever the learning system detects Failure Condition due to the amount of Negative rewards, the system initiates RAL process. Otherwise the agent keeps learning. Let us discuss how to restore (recover) the original state of the agent. If the agent keeps both all transition histories and action histories, agent could put all the statues back to the original. However, because of memory limitation, we introduce Retrospective Kalman Filter(RKF) to recover any state one step before with few memory. Whenever we use KF, we can estimate a posteriori state estimate $\hat{X}_{k-1}$ and a posteriori estimate covariance P_{k-1} from a priori state estimate and a priori estimate covariance.

KF uses posteriori state estimate one step before $\hat{X}_{k-1}$ and posteriori estimate covariance one step before P_{k-1} in order to estimate a current a posteriori state estimate $\hat{X}_k$ and posteriori estimate covariance P_k. In RKF, we use a current a posteriori state estimate $\hat{X}_k$ and posteriori estimate covariance P_k in order to estimate posteriori state estimate one step before $\hat{X}_{k-1}$ and posteriori estimate covariance one step before P_{k-1}.

We can get a posteriori estimate covariance P_k from a priori estimate covariance P_{k-1}^- and Kalman gain G_k using Kalman Filter. Therefore, we can not get a priori estimate co-

variance P_{k-1}^{-} analytically from a posteriori estimate covariance P_k. We note that agents retain some priori estimate covariance P_{k-1}^{-} when learning and that they take reverse action. The number retaining a priori estimate covariance matrix is a hyperparameter. This is equal to the number of times that RAL can be performed. We define a retrospective Kalman filter below:

RetrospectiveFilteringStep	
G_k	$= P_k^{-} C^T (C P_k^{-} C^T + R)^{-1}$
$\hat{X}_k^{-}$	$= (I - G_k C)^{-1} (\hat{X}_k - G_k Y_k)$
RetrospectiveEstimateStep	
P_{k-1}	$= A^{-1}(P_k^{-} - BQB^T)(A^T)^{-1}$
$\hat{X}_{k-1}$	$= A^{-1}\hat{X}_k^{-}$

4.2. QLRKF

By the word "QLRKF", we mean a learning method that agents perform RAL. We use RKF to return agent's states. Agents keep learning by QLKF [9] unless agents do obey a Failure Condition. In QLKF, agents take an action under KF and provability ε, and greedy action with probability $(1-\varepsilon)$. If agents follow a Failure Condition, agents take a reverse action using the estimate by RKF with probability ε, and a reverse greedy action with probability $(1-\varepsilon)$.

5. Experiment

5.1. Hunter Prey Problem

In this work, we deal with the Hunter Prey problem which is a standard task of RL. Traditionally in the problem we discuss a discrete 2D space but here we assume a continuous 2D space of $m \times m$, $0 \leq x, y \leq m$ instead of the grid space. We assume one hunter and one prey in the space, the former agent always pursues (chases) the latter agent. The agents can't go outside the 2D space. Initially we put the two agents randomly in such a way that they keep at some distance off. We say *the prey is captured* when they stand closely with each other (say, less than ρ). Whenever the prey is captured, the hunter gets positive reward and the prey gets negative. Otherwise, the hunter gets negative and the prey positive.

There have been several kinds of assumption in hunter-prey games, but very often these preys take actions randomly without any knowledge. Here we assume both the agents might know positions of all the agents in the space, then we examine both cases of knowledgeable prey (i.e., the prey learns autonomously) and non-knowledgeable prey (actions are randomly selected). We assume the hunter learns by QL for the purpose of own Q-values improvement, considering prey's relative positions as states: based on RL, we assume discrete states, although these states are continuous. As for the prey, we assume the prey may learn by QL, or we assume take actions randomly without any knowledge.

Let us describe how we construct 2D space in a discrete manner. Assume a hunter stands at a position considered as a center relatively, and we divide the 2D space into 8 areas where each area is further divided into two areas, a close area and the farther, as well as a center area. So there are 17 (relative) areas in total. Let us illustrate an example

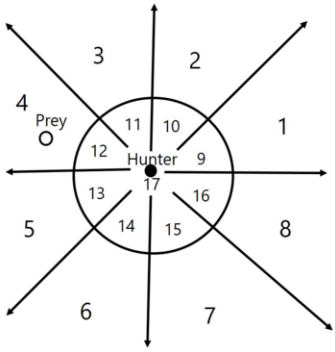

Figure 2. Position Area

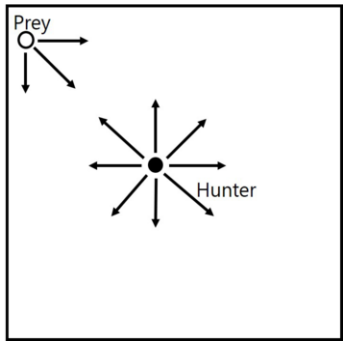

Figure 3. Actions of Hunter and prey

in a figure 2 where a hunter look at a prey in its area 4. There are 9 actions in the problem as shown in a figure 3: 8 directions and *stay*. For example, when a hunter stands at a center and a prey stands at the left upper corner, the hunter can select one of 9 actions. However, the prey can't move up or left any more because of the field boundary and can select one of 5 choices[1].

Let us summarize the movement of the agent:

(a) put a hunter and a prey in a space initially.
(b) a prey perceives a hunter and selects an action.
(c) a hunter perceives a prey and selects an action.
(d) both get rewards according to their results.
(e) a prey perceives a hunter and learn.
(f) a hunter perceives a prey and learn.
(g) if a hunter captures a prey, go to (a). Go to (b) otherise. We keep utilizing both the Q-values and the Kalman gain in any case.

In our proposed method, if the hunter has moved a certain number, but has not been able to catch the prey, hunter perform RAL. Let us note a certain number is hyperprameter and equal the number that hunter can perform reverse action. Let us describe the flow during RAL:

(a) a hunter perceives a prey and selects an action.
(b) a prey perceives a hunter and selects an action.
(c) a hunter and a prey get rewards.
(d) a hunter perceives a prey and learn.
(e) a prey perceives a hunter and learn.

5.2. Preliminaries

Here we mention our experimental results to see how well our approach works. We examine a Hunter Prey problem with one hunter and one prey. We discuss two cases in which the prey dose not learn and prey learns. In each case, hunter learns with ε-greedy (comparison method 1), QLKF (comparison method 2), and QLRKF (proposed method)

[1]The prey can select one of *right, right down, down* and *stay*.

and evaluates. We evaluate the number of capture steps. We consider that the number of capture steps indicates the quality of knowledge, and that the smaller the number of steps being better learning method.

When the hunter learns with QLKF (comparison method 2) and QLRKF (proposal method), the hunter uses a Kalman filter that estimates the position of the prey. The hunter takes an action aproaching the estimated position by KF with probability ε.

The field consists of a continuous 2D space $[0,1] \times [0,1]$. In the learning process, we assume the distance of 0.8 between the two agents initially, and the prey can be captured within distance $\rho = 0.1$. The reward 100 is given to the hunter when the hunter captures the prey, and -1 otherwise. Hunters receive a reward of 1 during RAL by our proposed method. Note this reward is the reverse of normal learning. The hunter performes RAL on condition that the hunter do not captures the prey. Therefore, we do not consider the reverse reward at capture. On the contrary, the prey gets the reward -80 when the prey is captured and 1 otherwise. As well as the hunter, the prey receive a -1 reward during the hunter performes RAL by our proposed method.

Here we assume $\alpha = 0.1$ (learning rate) and $\gamma = 0.9$ (discount rate) of QL for both agents. And we assume $\varepsilon = 0.1$ for both agents. We set the number that holds P_{k-1}^{-} of our proposed method to 50.

For KF processing, we initialize covariance matrices: $P_0 = 10^4 I$ a covariance matrix of state estimation error, $V_0 = 0.05I$ a covariance matrix of process noises and $0.9 \times 0.999^{learningcount} I$ a covariance matrix of observation noises. We set the covariance matrix of the observation noise as $0.9 \times 0.999^{learningcount} I$, so that the error decreases as the number of learning increases.

When a hunter likes to estimate the next state x_{t+1}, y_{t+1} of the prey at time t by KF, it observes the state x_t, y_t of the prey by position sensors. Let us note the hunter estimates the current state using the previous observation using KF framework with process noise V_t and observation noise W_t like a state equation 3 and an observation equation 4:

$$\begin{pmatrix} x_{t+1} \\ y_{t+1} \end{pmatrix} = \begin{pmatrix} 1\ 0 \\ 0\ 1 \end{pmatrix} \begin{pmatrix} x_t \\ y_t \end{pmatrix} - \begin{pmatrix} 1\ 0 \\ 0\ 1 \end{pmatrix} \begin{pmatrix} hv_{xt} \\ hv_{yt} \end{pmatrix} + \begin{pmatrix} 1\ 0 \\ 0\ 1 \end{pmatrix} V_t \tag{3}$$

$$\begin{pmatrix} x_t \\ y_t \end{pmatrix} = \begin{pmatrix} 1\ 0 \\ 0\ 1 \end{pmatrix} \begin{pmatrix} x_t \\ y_t \end{pmatrix} + W_t \tag{4}$$

Here hv_{xt}, hv_{yt} represent velocity of the hunter at time t.

5.3. Evaluation Criteria

Let us discuss evaluation criteria, and capturing process. First in this experiment, we say *one step* when both agents take every actions (behaviors), and *one interval* for 100 steps. This means, during one interval in the learning process, the hunter learns (updates Q-values) 100 times. We also say *one episode* for 100 intervals, or equivalently 10,000 steps. We examine a capturing steps in every interval. We initialize Q-values at the beginning of each episode (10,000 steps). We examine 10 episodes for every experiment and take the average steps for capturing of 10 episodes.

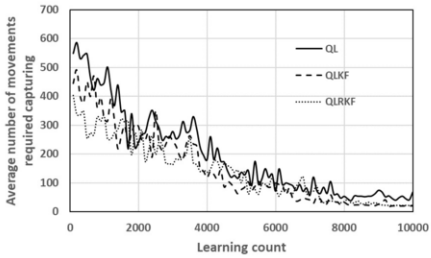

Figure 4. Relationship between learning count and capture step number (When only hunter is learning)

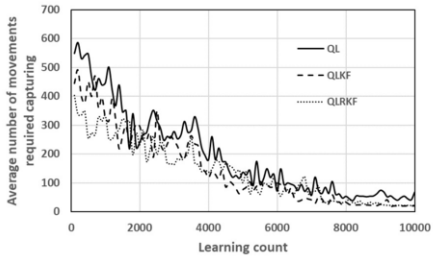

Figure 5. Relationship between learning count and capture step number (When both hunters and prey are learning)

In this experiment, we examine both learning and capturing processes in an interleaved manner. That is, we do learning in one interval then capturing the prey 10 times using the Q-values, and we continue learning further. At each interval, we stop learning and execute the capturing: the hunter chases the prey 10 times. Everytime we take counts of the steps[2] until the hunter captures the prey, and take an average count as *capturing step* of this interval. The chasing manner in the capturing process is the same as in the learning process. However, during the capturing process of our proposed method, the hunter does not take a reverse action, and take an action based on QLKF. That is, when learning with ε-greedy during learning process, the capturing process is also ε-greedy. (Hereafter, in the case of an agent takes an action based on ε-greedy, we denote as QL.)

Note that we should have some discussion of how to evaluate results of learning and capturing. Clearly the smaller steps we need for capturing, the better knowledge we have. Also, the less steps we need to get convergence (described later), the better parameters we have for learning. The former concerns about capturing quality while the latter about learning efficiency. Then we also introduce a notion of *harmonic average h* as a combined criteria of both learning and capturing where $h = 2pq/(p+q)$, 1 p means a earning step and q a capturing step. The less h goes, the better performance we have.

5.4. Results

In tables 1 and 2, we show the results of capturing steps depending on "who learn by QL, QLKF and QLRKF". We also illustrate overviews of the tables in figures 4 and 5 respectively.

[2]Up to 1,000 steps.

Table 4. Harmonic Average (only Hunter learned)

Learning	QL	QLKF	QLRKF
100	169.1	163.3	162.9
500	522.2	474.3	406.5
1000	617.2	491.8	476.3
1500	561.3	446.9	486.4
2000	429.5	516.9	351.4
2500	559.2	614.2	209.7
3000	508.7	454.3	337.7
3500	521.7	490.6	268.1
4000	344.1	230.2	298.3
4500	332.1	169.9	286.4
5000	266.8	152.0	232.1
5500	188.7	207.4	171.1
6000	141.8	196.4	182.7
6500	177.9	92.1	185.1
7000	113.8	82.1	86.6
7500	126.5	58.6	101.9
8000	106.8	65.6	75.6
8500	107.6	46.5	56.5
9000	147.7	54.3	78.9
9500	80.2	41.5	51.8
10000	135.3	41.6	80.6
(Average)	310.2	240.6	213.9
(StdDev)	184.5	178.0	127.0

Table 5. Harmonic Averages(Both learned)

Leaning	QL	QLKF	QLRKF
500	537.7	456.2	371.6
1000	507.9	426.8	239.8
1500	737.0	311.7	248.2
2000	688.4	124.6	228.2
2500	580.1	282.6	177.6
3000	503.3	238.4	148.0
3500	307.3	139.5	97.5
4000	242.7	135.6	67.3
4500	209.8	71.1	69.8
5000	232.6	134.3	76.8
5500	190.3	62.6	58.2
6000	174.6	61.3	63.0
6500	140.6	60.8	63.8
7000	165.2	61.2	59.4
7500	118.1	61.4	64.7
8000	263.6	59.2	70.1
8500	305.9	58.6	60.0
9000	292.2	60.8	54.2
9500	180.2	63.5	66.1
10000	83.8	60.6	54.4
(Average)	311.2	148.5	125.6
(StdDev)	185.0	126.4	94.2

Table 1. Capturing Steps (only Hunter learned)

Learning	QL	QLKF	QLRKF
500	546.4	451.1	342.4
1000	446.3	326.1	312.6
1500	345.2	262.6	290.2
2000	240.6	296.8	192.6
2500	314.8	350.1	109.4
3000	277.9	245.8	178.9
3500	281.9	263.8	139.4
4000	179.8	118.5	154.9
4500	172.4	86.6	147.9
5000	137.1	77.1	118.8
5500	96.0	105.7	86.9
6000	71.8	99.8	92.7
6500	90.2	46.4	93.9
7000	57.4	41.3	43.6
7500	63.8	29.4	51.3
8000	53.7	32.9	38.0
8500	54.1	23.3	28.4
9000	74.5	27.2	39.6
9500	40.3	20.8	26.0
10000	68.1	20.8	40.5

Table 2. Capturing Steps (Both learned)

Learning	QL	QLKF	QLRKF
500	581.7	419.4	295.6
1000	340.4	271.3	136.2
1500	488.5	173.9	135.3
2000	415.7	64.3	121.0
2500	328.2	149.8	92.0
3000	274.7	124.2	75.9
3500	160.7	71.2	49.4
4000	125.2	68.9	34.0
4500	107.4	35.8	35.2
5000	119.1	68.1	38.7
5500	96.8	31.5	29.2
6000	88.6	30.8	31.7
6500	71.1	30.5	32.1
7000	83.6	30.7	29.9
7500	59.5	30.8	32.5
8000	134.0	29.7	35.2
8500	155.8	29.4	30.1
9000	148.5	30.5	27.2
9500	91.0	31.9	33.2
10000	42.1	30.4	27.3

Table 3. Total number of Capturing Steps

Learning	QL	QLKF	QLRKF
(only Hunter learned)	19467	14916(0.77)	12695(0.65)
(Both learned)	19868	9308(0.47)	7806(0.39)

It is clear that our proposed approach (QLRKF) outperforms the comparison methods (QL, QLKF), ie, we have less steps (the better efciency). A table 3 contains the comparison of capturing steps in QL, QLKF, QLRKF cases. In the total number of capturing steps of only Hunter learned, we see an improvement by using QLRKF during learning process to 65% than QL, and 85% than QLKF. As for the case of both hunter and prey learned, we see an improvement to 40%(QL) and 84%(QLKF).

Let us illustrate the detailed comparison using the harmonic averages in tables 4 and 5. In the averages of only Hunter learned, QLRKF becames to 75% of QL and 97% of QLKF. As for the case of both hunter and prey learned, QLRKF becames to 40%(QL) and 85%(QLKF). Let us see the result the standard deviation of the harmonic averages. In the standard deviation of only Hunter learned, we see an improvemed to 70%. Moreover, as for the case of both hunter and prey learned, we see an improvement to 50% than QL and 75% than QLKF. This result means that learning progresses stably.

5.5. Discussion

We change the number of times that the hunter can perform revers actions with QLRKF to 20, 50, and 80, and examine the results of capturing steps. In tables 6 and 7, we show the results of capturing steps. We also illustrate overviews of the tables in figures 6 and 7 respectively. Let us compare the total number of capturing steps to 50 times and the others(20, 80 times). When the prey does not learn, it deteriorates to 120% in the case of the hunter perform RAL 20 times, and improves to 75% in the case of the learning is 80 times. As for the case of both hunter and prey learned, it deteriorated to 220% in the case of 20 times, and improved to 93% in the case of 80 times. We see that our process goes better everytime.

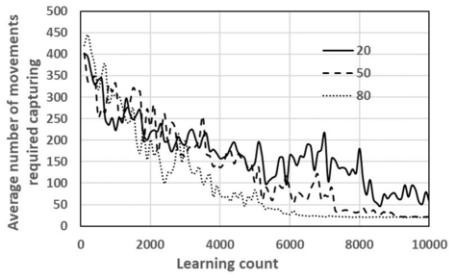

Figure 6. Capturing Steps with respect to the number of RAL (only Hunter learned)

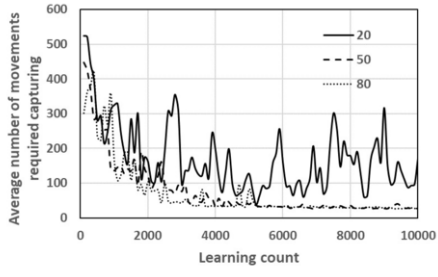

Figure 7. Capturing Steps with respect to the number of RAL (Both learned)

Table 6. Capturing Steps with respect to the number of RAL (only Hunter learned)

Learning	20	50	80
500	340.87	255.06	317.13
1000	222.79	331.42	302.41
1500	248.24	321.37	274.67
2000	217.9	231.54	200.91
2500	195.94	197.54	124.38
3000	189.51	162.41	198.83
3500	205.96	255.78	97.06
4000	157.18	135.94	71.78
4500	173.24	166.76	64.27
5000	147.86	99.51	83.88
5500	112.45	59.87	39.15
6000	165.12	82.91	26.41
6500	190.25	62.12	23.03
7000	215.91	71.73	20.62
7500	136.16	35.29	21.14
8000	180.38	34.06	21.15
8500	52.85	26.38	20.57
9000	76.67	24.82	21
9500	94.36	21.37	20.4
10000	56.83	22.29	19.39

Table 7. Capturing Steps with respect to the number of RAL (Both learned)

Learning	20	50	80
500	280.8	295.63	234.62
1000	324.7	136.22	165.73
1500	284.99	135.28	107.87
2000	162.72	120.99	104.12
2500	195.17	92.05	100.71
3000	98.18	75.89	42.75
3500	124.92	49.44	34.61
4000	195.77	33.96	32.73
4500	98.25	35.18	35.22
5000	126.89	38.71	83.36
5500	84.8	29.23	34.4
6000	158.09	31.69	33.55
6500	106.94	32.08	32.6
7000	102.99	29.85	31.17
7500	301.55	32.48	29.73
8000	178.92	35.21	29.42
8500	64.19	30.1	31.41
9000	316.93	27.18	30.14
9500	135.52	33.18	26.17
10000	172.26	27.26	25.55

Table 8. Harmonic Averages with respect to the number of RAL (only Hunter learned)

Learning	20	50	80
500	405.4	337.8	388.1
1000	364.4	497.8	464.4
1500	426.0	529.3	464.3
2000	393.0	415.0	365.1
2500	363.4	366.1	237.0
3000	356.5	308.1	372.9
3500	389.0	476.7	188.9
4000	302.5	262.9	141.0
4500	333.6	321.6	126.7
5000	287.2	195.1	165.0
5500	220.4	118.5	77.7
6000	321.4	163.6	52.6
6500	369.7	123.1	45.9
7000	418.9	142.0	41.1
7500	267.5	70.2	42.2
8000	352.8	67.8	42.2
8500	105.0	52.6	41.0
9000	152.0	49.5	41.9
9500	186.9	42.6	40.7
10000	113.0	44.5	38.7
(Average)	297.1	235.0	165.7
(StdDev)	100.1	151.4	143.5

Table 9. Harmonic Averages with respect to the number of RAL (Both learned)

Learning	20	50	80
500	359.6	371.6	319.4
1000	490.2	239.8	284.3
1500	479.0	248.2	201.3
2000	301.0	228.2	197.9
2500	362.1	177.6	193.6
3000	190.1	148.0	84.3
3500	241.2	97.5	68.5
4000	373.3	67.3	64.9
4500	192.3	69.8	69.9
5000	247.5	76.8	164.0
5500	167.0	58.2	68.4
6000	308.1	63.0	66.7
6500	210.4	63.8	64.9
7000	203.0	59.4	62.1
7500	579.8	64.7	59.2
8000	350.0	70.1	58.6
8500	127.4	60.0	62.6
9000	612.3	54.2	60.1
9500	267.2	66.1	52.2
10000	338.7	54.4	51.0
(Average)	296.0	125.6	116.4
(StdDev)	126.4	94.2	100.4

Let us illustrate the detailed comparison using the harmonic averages in tables 8 and 9. Similar to the total number of capturing steps, we compare in the case of 50 times and the others. In the case of only Hunter learned, it deteriorates to 126% in the case of 20 times, and improves to 70% in the case of the learning is 80 times. As for the case of both hunter and prey learned, it deteriorated to 235% in the case of 20 times, and improved to 92% in the case of 80 times. From these results, it can be seen that the more times the RAL can be performed, the better the results. Let us see the result the standard deviation of the harmonic averages. In the case of only Hunter learned, the standard deviation is 66% with 20 times, and 94% with 80 times. As for the case of both hunter and prey learned, the standard deviation is 134% with 20 times, and 106% with 80 times. Therefore, we cannot say that the standard deviation is affected by the number of RAL.

6. Conclusion

In this work, we have proposed a new method for reverse action by using Retrospective Kalman Filter that estimates the state one step before. In the total number of capturing steps of only Hunter learned, we see an improvement by using QLRKF during learning process to 65% than QL, and 85% than QLKF. As for the case of both hunter and prey learned, we see an improvement to 40% than QL and 84% than QLKF. In the the arithmetic averages of the harmonic averages of only Hunter learned, we see an improvemement by using QLRKF during learning process to 75% than QL, and 97% than QLKF. Moreover, as for the case of both hunter and prey learned, we see an improvement to 40% than QL and 85% than QLKF.

References

[1] Sutton, Richard S., and Andrew G. Barto. Reinforcement learning: An introduction. Vol. 1. No. 1. Cambridge: MIT press, 1998.
[2] Leslie Pack Kaelbling, Michael L. Littman, Andrew W. Moore. “Reinforcement Learning: A Survey.” CoRR cs.AI/9605103 (1996)
[3] Hado van Hasselt. “Double Q-learning.” NIPS 2010: 2613-2621
[4] Marco A, Wiering, and Hado van Hasselt. “Ensemble Algorithms in Reinforcement Learning.” IEEE Trans. Systems, Man, and Cybernetics, Part B 38(4): 930-936 (2008)
[5] Vukosi Ntsakisi Marivate, Michael L. Littman. “An Ensemble of Linearly Combined Reinforcement-Learning Agents.” AAAI (Late-Breaking Developments) 2013
[6] Raghu Ramakrishnan and Johannes Gehrke, Database Management Systems 3rd Edition. 2002.
[7] Benjamin Eysenbach, Shixiang Gu, Julian Ibarz, Sergey Levine. “Leave no Trace: Learning to Reset for Safe and Autonomous Reinforcement Learning.” CoRR abs/1711.06782 (2017)
[8] Watkins, Christopher JCH, and Peter Dayan. ”Q-learning.” Machine learning 8.3-4 (1992): 279-292.
[9] Kei Takahata, Takao Miura. “Reinforcement Learning using Kalman Filters.” IEEE International Conference on Cognitive Informatics and Cognitive Computing(ICCICC) 2019
[10] Takadama, K.:MultiAgent Learning, Corona-Sha, 2004 (in Japanese)
[11] Adati, S. and Maruta, I. : Fundamentals of Kalman Filter, Tokyo Denki University Press, 2012 (in Japanese)
[12] Takehiro Kitao, Masato Shirai, and Takao Miura: “Model Selection Based on Kalman Temporal Differences Learning.” CIC 2017: 41-47
[13] Matthieu Geist, Olivier Pietquin: “Kalman Temporal Differences.” J. Artif. Intell. Res. 39: 483-532 (2010)