2014 International Joint Conference on Neural Networks (IJCNN)
July 6-11, 2014, Beijing, China

Causality Traces for Retrospective Learning in Neural Networks - Introduction of Parallel and Subjective Time Scales -

Katsunari Shibata
Dept. of Electrical and Electronic Engineering
Oita University
700 Dannoharu, Oita, JAPAN
Email: shibata@oita-u.ac.jp

***Abstract*—We live in the flow of time, and the sensor signals we get not only have a huge amount in space, but also keep coming without a break in time. As a general method for effective retrospective learning in neural networks (NNs) in such a world based on the concept of "subjective time", "causality trace" is introduced in this paper. At each connection in each neuron, a trace is assigned. It takes in the corresponding input signal according to the temporal change in the neuron's output, and is held when the output does not change. This enables to memorize only past important events, to hold them in its local memory, and to learn the past processes effectively from the present reinforcement or training signals without tracing back to the past. The past events that the traces represent are different in each neuron, and so autonomous division of roles in the time axis among neurons is promoted through learning. From the viewpoint of time passage, there are parallel, non-uniform and subjective time scales for learning in the NN. Causality traces can be applied to value learning with a NN, and also applied to supervised learning of recurrent neural networks even though the way of application is a bit different. A new simulation result in a value-learning task shows the outstanding learning ability of causality traces and autonomous division of roles in the time axis among neurons through learning. Finally, several useful properties and concerns are discussed.**

I. Introduction

We live in the flow of time, and the sensor signals we get not only have a huge amount in space, but also keep coming without a break in time. We, who live in such a world, have a great ability of retrospective learning in which the past processes should be updated appropriately from the present reward or other signals. Aiming to realize such a great ability in artificial learning systems, reinforcement learning and/or supervised learning for a recurrent neural network (RNN) have been proposed and used widely.

Supervised learning for an RNN, represented by BTPP (Back Propagation Through Time) [1] and RTRL (Real Time Recurrent Learning) [2], enables a learning system to memorize important past information or to generate useful dynamics through retrospective learning from the given training signals. In BPTT [1], which is the most popular, retrospective learning is realized by holding all or truncated past inputs and outputs with a constant interval of time to trace back to the past. Therefore, it needs a huge amount of memory, and is not suitable for practical use. In RTRL [2], instead of tracing back to the past, the contribution of every connection weight to every neuron is updated, and learning is performed on-line. However, $O(n^3)$ memory capacity and $O(n^4)$ computational cost are necessary where n is the number of neurons, and they are beyond the order of the number of connections $O(n^2)$. Therefore, it cannot be locally achieved and so is not practical.

Reinforcement learning [3], represented by TD (Temporal Difference)-Learning, enables a learning system to acquire past time series of values and actions through retrospective learning from the present reward or punishment. In reinforcement learning, TD-learning using eligibility traces TD(λ) [3] is one of the frameworks for realizing effective retrospective learning. Bakker et al. introduced the eligibility traces when an RNN was used in reinforcement learning [4], and at each local connection, one trace was implemented. However, the λ parameter that decides the decay of the past information in the trace is a constant. Therefore, the information is decayed constantly not depending on how the state is important.

Here, one big concern pops up when contrasting the learning in artificial systems with ours. Retrospective learning needs some sort of memory to hold some information about the past state. In the artificial ones, there seems to be a preconception that the time should pass uniformly also in the memory; the step-size is a constant in the discrete-time domain, while the time constant is a constant in the continuous-time domain. On the other hand, thinking about ourselves, we are not conscious of all the past states but conscious of only important states or events, which seems to enable effective learning. When someone goes abroad by plane taking one day, for most of the time, he/she sits in a seat on the plane, but can pay attention to a mistake made in a moment at the airport before the long flight. He/she does not remember all the states (sensor signals) at every second of the day, but can remember and learn his/her behaviors at the airport retrospectively. Considering the above for years, the author has wanted to develop a flexible and effective learning system with memories with a non-uniform internal time scale in which time passes slowly around important events, but passes fast otherwise. However, if we provide the knowledge about what the important events are to the system in advance, it loses flexibility. Therefore, it is aimed to develop a learning system in which both flexible time scales and the extraction of important events are learned together in parallel. Since the measure of importance is set up by the learner itself, the time scale is called "subjective".

978-1-4799-1484-5/14/$31.00 ©2014 IEEE

Many works have focused on the "time" in learning. In many of them, basic mechanisms for measuring time, such as pacemaker clock or decaying memory, have been directly discussed, and the Weber's law in the time domain often has been a target to be explained [5]-[7]. In the work by Nakahara et al. [8], internal time is introduced in the TD model, and an explanation is attempted for the choice reversal in the intertemporal choice tasks. However, most of the effort is put on the conversion of value learning between two different time systems, and it has not been mentioned how the internal time system is formed. Daw et al. proposed a model of dopamine response using a TD algorithm for a partially observable semi-Markov process [9]. Semi-Markov process provides us with the framework to deal with an event-driven process chain with non-constant intervals. However, the index of importance is just binary: event or not, and no way is given for how the events are discovered by learners. Yamashita et al. proposed the learning system named MTRNN (Multiple Timescale Recurrent Neural Network) which consists of two sub-networks with a different time constant, and a dynamics representing more abstract state is formed in the sub-network with the larger time constant [10]. However, since the time constant is a constant in each neuron, the states change slowly in the sub-network with the large time constant, and it is difficult to deal with both a long-lasting event and a short event effectively.

The author proposed a novel method to solve the above problem in supervised learning of RNNs [11], [12], and also proposed separately a similar but different method in value learning using a layered neural network (NN) already [13]. Both use an NN, and have a common basic mechanism for memorizing the past important events effectively. In this paper, the mechanism is named "causality traces", and is introduced as a general way for effective retrospective learning of NNs. It is generally formulated both in discrete- and continuous-time domains, and the essential difference in usage between supervised learning of RNNs and value learning using layered NNs is clarified. A simple and new simulation result in a value learning shows the excellence of the proposed approach much more clearly than ever, and is the first to show that the causality traces promotes the division of roles in the time axis among neurons through learning compared with the case of eligibility trace. Finally, several useful properties and concerns for the novel approach are generally discussed.

II. Causality Traces

A. Concept of causality traces

As shown in the lower portion of Fig. 1, even though the author comes to his office almost everyday, he doesn't remember all the sensor signals at all the time, but remembers only important events such as "turn right at the intersection" or "go over the bridge". Such efficient memories seem to make us learning easy and effective. The problem is to define what the important events are. It is easy to think that 'state change' is defined as an important event. However, when replacing 'state' with 'sensor signals', the number of sensor signals is huge and each signal changes very often. Only by moving our head, the visual sensor signals change drastically. Then, as shown in the upper portion of Fig. 1, the 'state change' is defined as the output change in each neuron. A memory called "causality trace" is assigned at each connection. It takes

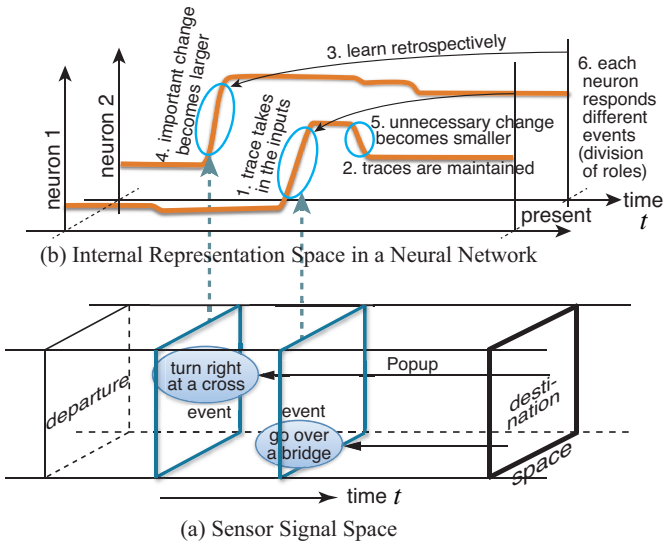


Fig. 1. Sensor signal space and internal representation space in an NN with the time axis. Important events for each neuron are defined from the change of its output. Causality traces take in the inputs according to the temporal change in the neuron's output.

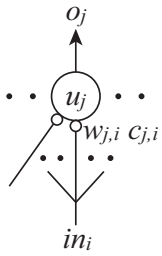


Fig. 2. A causality trace $c_{j,i}$ assigned at the i-th connection in the j-th neuron.

in the corresponding input signal according to the change of the neuron output, and holds its value when the output does not change. When a reinforcement or training signal is given, learning is performed retrospectively using the traces without tracing back to the past. As learning progresses, each neuron is expected to represent important events without being greatly influenced by trivial events. Therefore, two effects, that learning enables the causality traces to hold distant past events and that the traces enable learning for more distant past processes, work synergistically. The division of roles for events at different times is also expected among neurons as learning progresses.

B. Basic formulation of causality traces

As shown in Fig. 2, in order to take in and hold the causal inputs according to the state change in each neuron, a causality trace c is assigned at each connection, which can be a feedback connection in an RNN. Its time evolution is formulated in the continuous-time domain as

$$\frac{dc_{j,i}}{dt} = \frac{|do_j|}{dt}(in_i - c_{j,i}) \tag{1}$$

where $c_{j,i}$ is the causality trace assigned for the i-th input in the j-th neuron, o_j is the output of the neuron and in_i is the

i-th input. By omitting dt from both sides, it is rewritten as

$$dc_{j,i} = |do_j|(in_i - c_{j,i}). \tag{2}$$

This means that the trace change is independent from the external time coordinate system. In other words, in a situation where state changes slowly, the trace also changes slowly, and in a situation where state changes rapidly, the trace also changes rapidly without adjustment of any parameter such as a time constant. When it is formulated in the discrete-time domain, it is written as

$$\begin{aligned} c_{j,i,step} &= (1 - |\Delta o_{j,step}|)c_{j,i,step-1} \\ &\quad +|\Delta o_{j,step}|in_{i,step} \end{aligned} \tag{3}$$

where $\Delta o_{j,step} = o_{j,step} - o_{j,step-1}$. It can be also known that the equation does not depend on the step-size Δt.

On the other hand, the eligibility traces are usually used in reinforcement learning [3], but they are easily extended to a general method to hold the past information for retrospective learning in neural networks. The eligibility trace e takes an input signal constantly as a first-order lag and its time evolution is represented as follows for each of the continuous- and discrete-time domains.

$$\tau \frac{de_{j,i}}{dt} = in_i - e_{j,i} \tag{4}$$

$$\begin{aligned} e_{j,i,step} &= (1 - \frac{\Delta t}{\tau})e_{j,i,step-1} + \frac{\Delta t}{\tau} in_{i,step} \\ &= \lambda e_{j,i,step-1} + (1 - \lambda) in_{i,step} \end{aligned} \tag{5}$$

where Δt is the duration of one step, τ is a time constant and $\lambda (0 \leq \lambda < 1)$ is a constant representing the decay of the trace in the discrete-time domain. The second term in the right-hand side of the Eq. (5) is usually just $in_{i,step}$, but so as that the trace approaches the input itself when it is a constant, $(1 - \lambda)$ is put before $in_{i,step}$ in Eq. (5). Comparing Eq. (1) and Eq. (4), in the causality trace, $\frac{dt}{|do_j|} = 1/\frac{|do_j|}{dt}$ in Eq. (1) works as a 'time constant' though it is not a constant anymore. It means that the time passes fast when the output changes a lot and passes slowly when the output does not change so much. The neuron output o is not given from outside of the NN, but generated with the connection weights in the NN from the external inputs, and the weights are updated through learning. Therefore, it can be said that the time scale is "subjective".

Fig. 3 shows an example of temporal change in the causality trace compared with the case of eligibility trace. They are computed from an input and an output generated for demonstration. It is seen that when the neuron output (b) changes a lot at $t = T_2$ or $t = T_3$, the causality trace (c) also changes a lot towards the neuron input (a). On the other hand, the eligibility trace with a large λ (d) always moves slowly towards the input (a) not depending on the output (b), while the eligibility trace with a small λ (e) moves a lot to the input (a) when the input changes a lot. A typical difference between causality trace and eligibility trace can be seen in the effect of the input between T_3 and T_4. At $t = T_5$, the eligibility trace is large not depending on the value λ, while the causality trace is small. That is because the duration between T_3 and T_4 is short, but the output changes significantly.

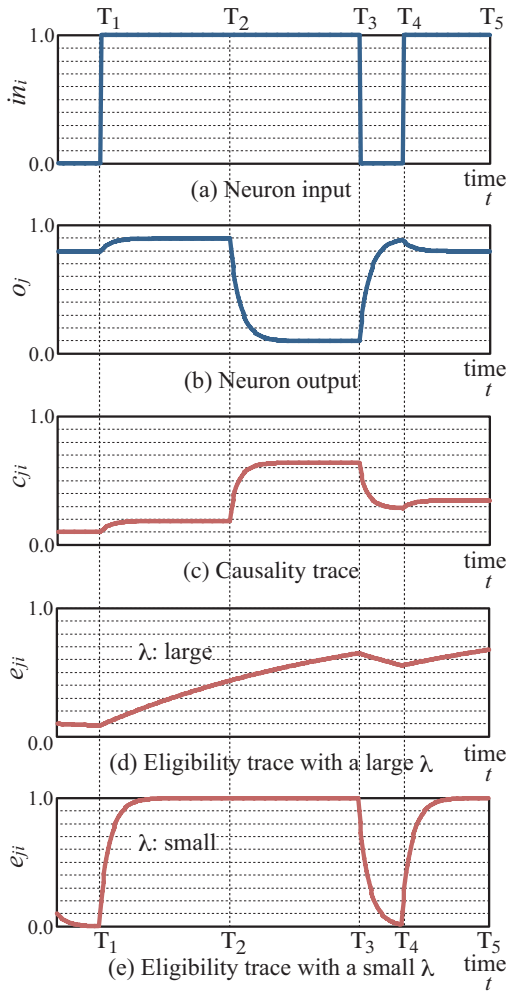


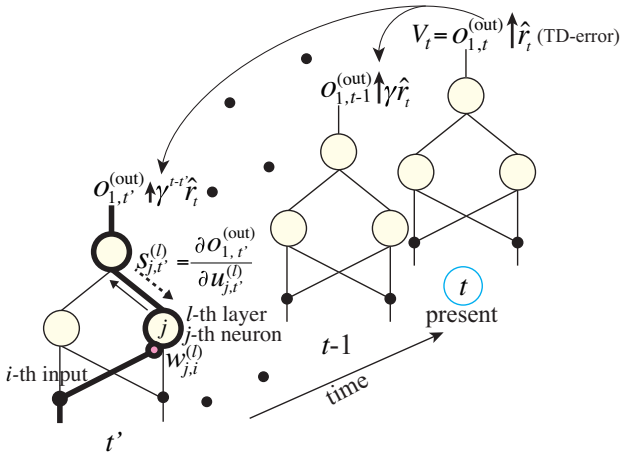
Fig. 3. Comparison of the temporal change between causality trace and eligibility trace.

C. Causality traces in value learning

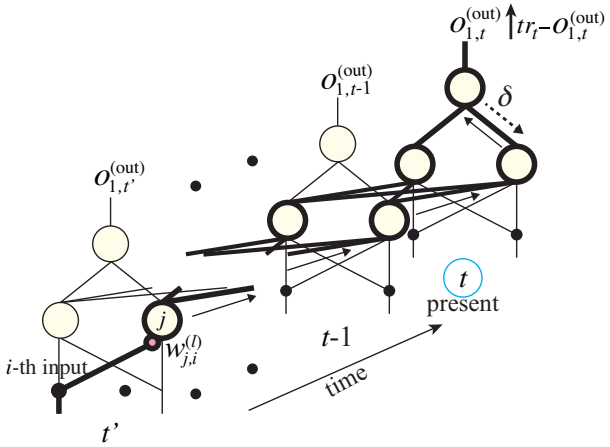
In value learning using a layered neural network (NN) whose output represents state or action value, the causality traces can be used on behalf of the eligibility traces. Here, for simplicity, discrete-time domain and the step size $\Delta t = 1$ are assumed, and also the static neuron model is used as

$$o_{j,t}^{(l)} = f(u_{j,t}^{(l)}), \quad u_{j,t}^{(l)} = \sum_i w_{j,i}^{(l)} o_{i,t}^{(l-1)} \tag{6}$$

where $o_{j,t}^{(l)}$ and $u_{j,t}^{(l)}$ are the output and internal state of the j-th neuron in the l-th layer at time t, and $o_{i,t}^{(1)}$ means i-th external input signal. $f()$ indicates an output function, and a sigmoid function is used in this paper. As shown in Fig. 4(a), retrospective learning is necessary to learn not only the present output (value) $o_{1,t}^{(\mathrm{out})}$, but also the past outputs $o_{1,t'}^{(\mathrm{out})} (t' = 0, 1, .., t-1)$ simultaneously with the discounted TD-error $\gamma^{t-t'} \hat{r}_t$ where γ is a discount factor $(0 \leq \gamma < 1)$ and $\hat{r}_t$ is the TD-error. The TD-error is represented as $\hat{r}_t = r_{t+1} + \gamma o_{1,t+1}^{(\mathrm{out})} - o_{1,t}^{(\mathrm{out})}$ where r is the given reward. To learn the past output $o_{1,t'}^{(\mathrm{out})}$ at the present time t, the sensitivity of the internal state $u_j^{(l)}$ to the



(a) Value learning in a feedfoward neural network



(b) Supervised learning in a recurrent neural network (RNN)

Fig. 4. The trained output is different between the value learning in a feedforward NN and supervised learning in a recurrent neural network (RNN). The thick lines indicate the signal flow along which $w^{(l)}_{j,i}$ at $t-\tau$ influences the trained output.

output $o^{(\text{out})}_1$ at time t' represented as $s^{(l)}_{j,t'} = \frac{\partial o^{(\text{out})}_{1,t'}}{\partial u^{(l)}_{j,t'}}$, has to be included in the trace and so the eligibility trace is updated at time t as

$$e^{(l)}_{j,i,t} = \gamma\lambda e^{(l)}_{j,i,t-1} + (1-\lambda)s^{(l)}_{j,t}o^{(l-1)}_{i,t} \tag{7}$$

referring to [4]. The sensitivity $s^{(l)}_{j,t}$ is computed by back propagation by expanding it as

$$s^{(l)}_{j,t} = \frac{\partial o^{(\text{out})}_{1,t}}{\partial u^{(l)}_{j,t}} = \frac{do^{(\text{out})}_{1,t}}{du^{(\text{out})}_{1,t}} \frac{\partial u^{(\text{out})}_{1,t}}{\partial o^{(l)}_{j,t}} \frac{do^{(l)}_{j,t}}{du^{(l)}_{j,t}}. \tag{8}$$

By replacing λ with $1-|\Delta o^{(l)}_{j,t}|$, the causality trace is updated as

$$c^{(l)}_{j,i,t} = \gamma(1-|\Delta o^{(l)}_{j,t}|)c^{(l)}_{j,i,t-1} + |\Delta o^{(l)}_{j,t}|s^{(l)}_{j,t}o^{(l-1)}_{i,t}. \tag{9}$$

The weight is updated at time t for each case of eligibility trace and causality trace as

$$\Delta w^{(l)}_{j,i,t} = \eta\hat{r}_t(e^{(l)}_{j,i,t} \text{ or } c^{(l)}_{j,i,t}) \tag{10}$$

where η is a learning rate [13].

D. Causality traces in supervised learning of recurrent neural networks (RNNs)

Here, it is explained how the causality traces are used for supervised learning in an RNN. As an example for explanation, an Elman-type RNN is considered in which neurons in the hidden layer have feedback connections with each other. Unlike RTRL that also enables on-line learning in RNNs, the eligibility trace or causality trace holds only local influence of each connection weight to the neuron where the connection exists, and so either the memory capacity or computational cost falls in the order of the number of connection weights. As shown in Fig. 4(b), in RNNs, the past input signals influence the present hidden neurons' outputs through the feedback connections, and the output of the RNN is derived as a static mapping of them. Accordingly, only the hidden neurons employ the causality traces, and the output neuron updates its connection weights as well as the regular BP. In contrast to the case of value learning mentioned in the previous sub-section, the past output is not learned from the present training signal. Therefore, each causality trace in each hidden neuron takes in and holds just the product of the input signal and the derivative of the neuron as

$$c^{(l)}_{j,i,t} = (1-|\Delta o^{(l)}_{j,t}|)c^{(l)}_{j,i,t-1} + |\Delta o^{(l)}_{j,t}|f'(u^{(l)}_{j,t})o^{(l-1)}_{i,t}. \tag{11}$$

N is assumed to be the number of external input signals, and also it is assumed that the suffix i indicates an external inputs when $i \leq N$, and indicates a feedback input signals from a hidden neuron at the previous time step when $i > N$. Then, for the feedback input signals, the time evolution of each causality trace is written as

$$c^{(l)}_{j,i,t} = (1-|\Delta o^{(l)}_{j,t}|)c^{(l)}_{j,i,t-1} + |\Delta o^{(l)}_{j,t}|f'(u^{(l)}_{j,t})o^{(l)}_{i-N,t-1}. \tag{12}$$

Each weight in each hidden neuron is updated using propagated error signal δ as follows [12].

$$\Delta w^{(l)}_{j,i,t} = \eta\delta^{(l)}_{j,t}c^{(l)}_{j,i,t}. \tag{13}$$

In BPTT, the error signal for the past time t' computed from the square error $E_t = \frac{1}{2}(tr_t - o^{(\text{out})}_{1,t})^2$ at the present time t where tr_t is the given training signal is represented as

$$\delta^{(l)}_{j,t'\leftarrow t} = -\frac{\partial E_t}{\partial u^{(l)}_{j,t'}} = (tr_t - o^{(out)}_{1,t})\frac{\partial o^{(out)}_{1,t}}{\partial u^{(l)}_{j,t'}}, \tag{14}$$

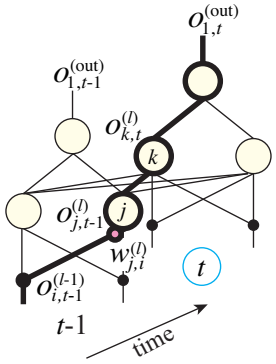
and is propagated backward from t to t'. However, the propagated error signal δ here [11] is computed as

$$\delta^{(l)}_{j,t} = (tr_t - o^{(\text{out})}_{1,t})f'(u^{(\text{out})}_{1,t})w^{(\text{out})}_{1,j} + \sum_k v^{(l)}_{k,j,t-1}\delta^{(l)}_{k,t-1}. \tag{15}$$

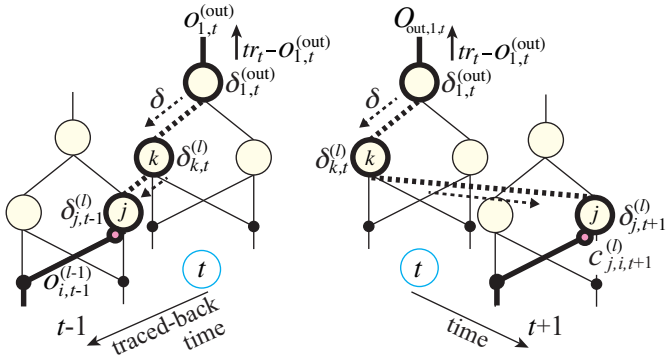
$v^{(l)}_{k,j}$ holds the product of connection weight $w^{(l)}_{k,j}$ and the past $f'(u^{(l)}_k)$ when the output change was large. Therefore, the way of updating v is similar to that for the causality trace c, and $v^{(l)}_{k,j}$ is computed as

$$v^{(l)}_{k,j,t} = (1-|\Delta o^{(l)}_{k,t}|)v^{(l)}_{k,j,t-1} + |\Delta o^{(l)}_{k,t}|w^{(l)}_{k,j+N}f'(u^{(l)}_{k,t}). \tag{16}$$

The error signal δ is similar to that in BPTT, but there are two major differences. At first, it does not include the derivative $f'(u^{(l)}_{j,t})$ of the neuron because the causality trace



(a) One contribution path in forward computation



(b-1) BPTT (b-2) C-trace-based learning

(b) Backward computation for the contribution as in (a)

Fig. 5. The difference in error signal propagation between BPTT (b-1) and causality-trace-based learning (b-2).

includes it as in Eq. (11) to hold the derivative when the neuron's output changed.

The second big difference is that the time does not turn back, but goes on by the propagation through the feedback connections. Let us consider the update of the weight $w^{(l)}_{j,i}$ based on its contribution at $t-1$ to the present network output $o^{(out)}_{1,t}$ through the k-th hidden neuron at time t as shown in Fig. 5(a). In BPTT, as shown in Fig. 5(b-1), δ_{t-1} is computed from the present δ_t, and the weight is updated according to the product of the $\delta^{(l)}_{j,t-1}$ and the past input signal $o^{(l-1)}_{i,t-1}$. The learning is not on-line in the meaning that updates of the weights for the times $t-1, t-2, ...$ are done individually at time t. However, in the learning using the traces, as shown in Fig. 5(b-2), even though the error signals trace backward through the same feedback connections, the time goes on and δ_{t+1} is computed from δ_t. After that, the weight is updated according to the product of the $\delta^{(l)}_{j,t+1}$ and $c^{(l)}_{j,i,t+1}$ at time $t+1$ expecting that the causality traces hold the information about the past important events. Accordingly, the retrospective learning can be done on-line such that Eq. (13) has only suffix t for time. The algorithm is still being explored and is not exactly the same as in [11], [12], and should be further examined in the future. When eligibility traces are used, $|\Delta O^{(l)}_{j,t}|$ is replaced with $(1-\lambda)$.

III. Simulation

In this section, a simulation where state value (critic) is learned with a layered NN is introduced to show the effective learning ability by using the causality traces clearly. As shown in Fig. 6, there is a special one-dimensional field that is divided into 100 regions. An agent takes 200 steps to go through an odd-numbered region, and takes just 2 steps to go through an even-numbered region. There are 300 input signals, and each of them has a non-zero value only in one of the 100 regions. There are three types of input signals. One of them takes a constant value in the corresponding region. In the other two types, the value is increased from 0.0 to 1.0 or decreased from 1.0 to 0.0 with a constant slope in the region. The number of hidden neurons is 30. In the output layer, there is only one neuron that learns to represent the state value. Here, no exploration is done, and the agent always just moves one step to the right. At the 10,100th step, the agent reaches the goal, which is the right end of the field, and gets a reward of 1.0. The discount factor γ is set so as that the ideal state value at the first step is 0.2. The purpose of this task is to form the correct state value (critic), and the convervence speed is compared among the cases of one-step learning (no trace or it can be said as eligibility trace with $\lambda = 0.0$), eligibility trace with various λs, and causality trace.

The task is similar to that in [13], but the sizes of all the regions are not uniform to show the effectiveness of causality traces much more clearly. Furthermore, after exploring the initial input-hidden connection weights, it was found that small initial weights are better for learning. Therefore, all of them are set to 0.0 instead of random numbers from -1.0 to 1.0 for the case of eligibility trace. However, in the case of causality trace, when the weight is set to 0.0, the output of each hidden neuron does not change and so the causality traces do not change at all. Then small random values from -0.1 to 0.1 are used for the initial weights. Nevertheless, the output changes in hidden neurons are still small, and then the change of each output ΔO in Eq. (9) is normalized by the value range that the neuron experienced in the past. The initial hidden-output connection weights are set to a random number from -1.0 to 1.0 as in [13]. To optimize the combination of the learning rate for hidden layer and that for output layer, they were increased separately and gradually by being double from 1.0, and the optimal one was found with which the mean product of global and local errors described later at the 50th episode over 10 simulation runs is the minimum under the condition that no hidden output oscillates. The following graphs are shown for the closest case to the average for each case.

Fig. 7 shows learning curves for the two types of errors. Part (a) shows the mean absolute difference from the ideal value over all the 10,100 steps, and shows how fast the output can be close to the ideal value globally. Part (b) shows the mean absolute TD-error over all the 10,100 steps, and that mainly shows the smoothness between the values of the neighbor regions. Fig. 8 shows the state value (critic) curves in one episode for 4 cases after 30 episodes of learning. In the case of one-step learning (a) or eligibility trace with a small λ (b), it is easily seen that the propagation of the value from the goal state is slow and in Fig. 7(a), the difference from the ideal value was not reduced so fast. However, since the curve of the value is not so fluctuated, the TD-error did not become

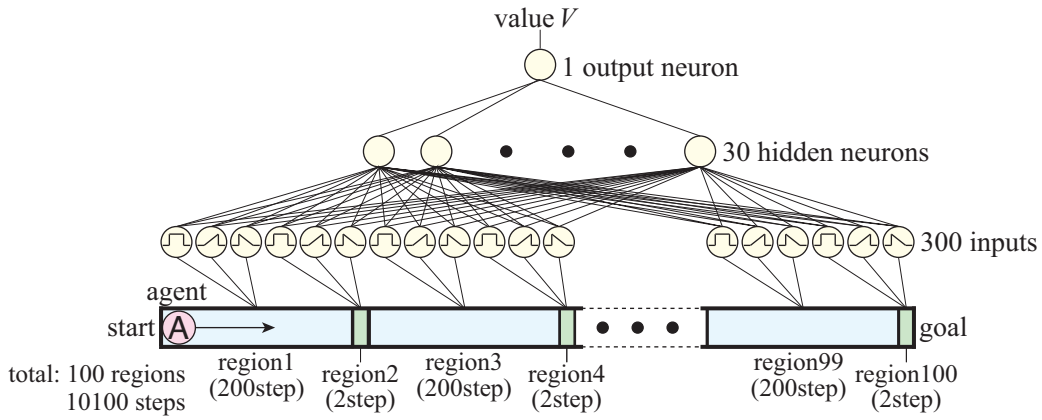


Fig. 6. One-dimensional task field with non-uniform regions and an NN to compute the state value. An agent moves towards the goal state without any exploration factors.

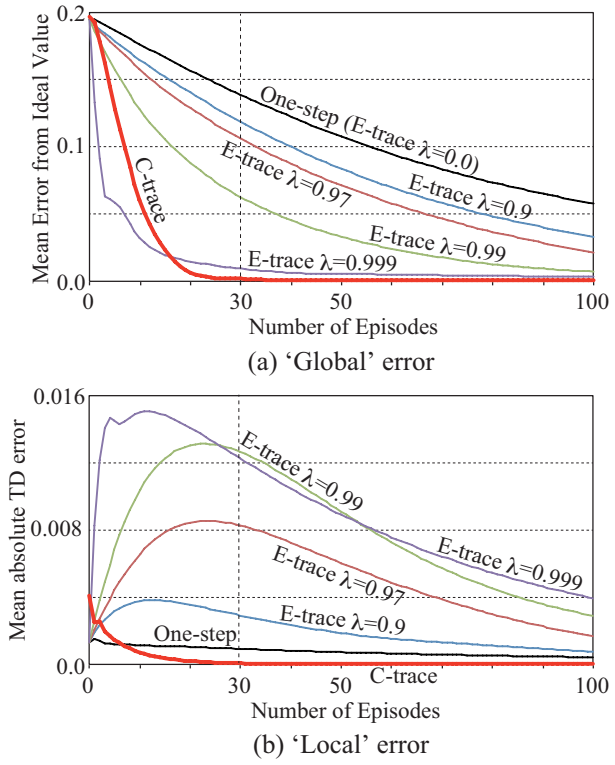


Fig. 7. Learning curves. (a) Mean error from the ideal value that shows global property, and (b) mean absolute TD-error that shows local property.

so large as shown in Fig. 7(b). On the other hand, when Fig. 8(c) is seen, which shows the case of the eligibility trace with a large λ, the rough shape of the value is close to the ideal one, but the periodical pulse train can be seen. That happens because in each odd-numbered state, the agent spent for as a short duration as only two steps, and so the traces do not take in the inputs so much compared with the even-numbered states. As the result, in Fig. 7(a), the output approaches the ideal value immediately, but the TD-error becomes large soon after learning began. Before learning, since the input-hidden connection weights are all 0.0, the output is a constant and so the TD-error is small. In the case of causality trace, as shown in Fig. 8(d), the value is very smooth and almost the same as the ideal curve. Before learning, the TD-error is larger than the other cases because the input-hidden weights have a non-zero value, but is decreased soon as shown in Fig. 7(b), and in Fig. 7(a), it is seen that the global shape of the value approaches the ideal rapidly even though the approach is faster at first in the case of eligibility trace with $\lambda = 0.999$.

Next, to examine whether or not each neuron represents different past events, the correlation of the input-hidden weight vectors between hidden neurons is observed. The eight hidden neurons that have a larger absolute weight to the output neuron are picked up, and the mean correlation coefficient between any two of the eight neurons is observed during learning. For a fair comparison, initial input-hidden connection weights are decided randomly from -0.1 to 0.1 in the eligibility trace case as well, and the same initial connection weights were set in all the cases. When the sign of the weight to the output neuron is different between the two hidden neurons, the sign of the correlation coefficient is inverted. Fig. 9 shows the change in the mean correlation coefficient for some cases. Since the hidden neurons try to represent important information to reduce the TD error, the correlation increases during learning in any cases. However, the correlation in the case of causality trace is significantly smaller than the other cases. The result did not change so much due to the initial connection weights that are decided randomly.

The eligibility traces take in the inputs constantly with λ, though only the sensitivity s to the network output is different in Eq. (7) among the hidden neurons. On the other hand, the causality traces in one neuron is more likely to take different events at different times from the other hidden neurons because ΔO is different among hidden neurons in Eq. (9), and that must help to promote the division of roles in the time axis.

IV. Discussion

A. Validity of time scale

It is also a big advantage in causality trace that we are no longer bothered by choosing an appropriate time constant manually. The time scale is determined by the output change in each neuron. The important thing is the validity of the time

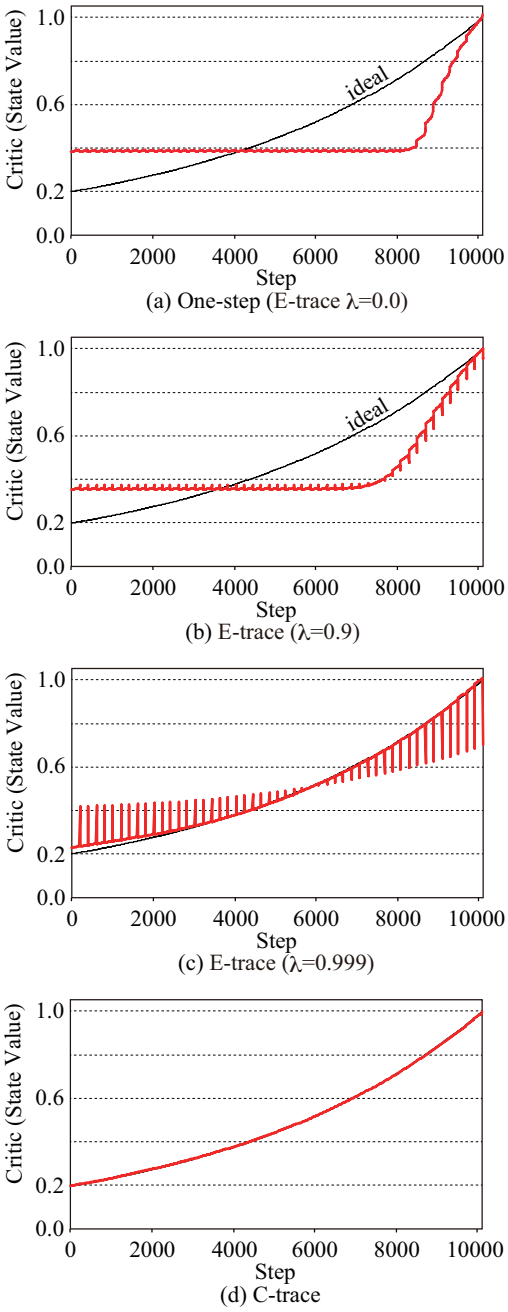


Fig. 8. Comparison of the shape of state value change after 30 episodes of learning.

scale. The underlying concept is that if the output changes from one end to the other in the value range, there is no need to hold the information before the change from the viewpoint of the cause-and-effect relation. This concept is the origin of the time-scale-independent learning. It is natural and matches the fact that when a big event happens, we are likely to forget the past events.

When a multi-layer neural network is employed, the higher hidden neurons, which are closer to the output layer, become to represent more abstract information through learning. The abstract information is expected not to be influenced so much by the frequent changes in the sensor signals as inputs to the

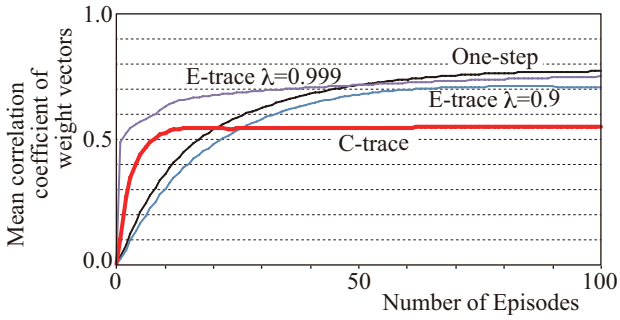


Fig. 9. Change of the mean correlation coefficient of input-hidden weight vectors between any two of the eight hidden neurons that have a large connection weight to the output neuron.

network. Therefore, they can hold abstract information in the distant past, while the lower hidden neurons, which are closer to the input layer, cannot hold their inputs due to the influence of the frequent changes in the sensor signals.

It also seems to match the property of the causality traces that when we have not yet learned to distinguish important events from trivial changes in our sensor signals, we can hardly notice the relation between a distant past cause and its effect at present, but after learning of important events, we can easily notice it.

B. Causality between neurons

In the simulation in this paper, a static neuron model, in which the output is computed only from the present inputs including feedback inputs, was used. In a layered NN, the signal flow between neurons is one way from lower to higher, and then lower neurons are always 'cause' and are not 'effect'. In an RNN, the delay for passing a feedback connection enables to detect the cause and effect between neurons by the causality traces.

When using a dynamical neuron model, the input signals cause the increase or decrease of its output according to its differential equation of the first-order lag. Since causality traces can extract the causes for effects directly, it is expected that cause and effect relations between two neurons are detected more efficiently by the traces than the case of using a static neuron.

Furthermore, STDP [14], which is a learning mechanism observed in biological organs, seems to extract the cause-and-effect relation between two neurons from the relative timing of their spikes, and it is interesting to think the relation to the causality traces.

C. Parallelism and autonomous division of roles

Even though there are many neurons in an NN, it is not useful if all of them work similarly. Therefore, autonomous division of roles is very important so as that the parallelism of an NN shows its excellent inherent ability. An NN that learns according to the steepest-descent type learning has originally has the ability of autonomous division of roles in space among hidden neurons. In each neuron, the representation that is useful to generate appropriate network outputs but has not

grown so much in the other neurons grows through learning. That enables a variety of functions to emerge in an NN through reinforcement learning [15]. It is a significant result in this paper that the difference of the causality traces among hidden neurons promotes the representation of past events that the other neurons do not represent, and the division of roles in the time axis is achieved.

D. Influence of noises

If high frequency noises are added to the input signals, the output of each hidden neuron also must change frequently. That results in unnecessary intake of input signals to the causality traces, and reduces the effectiveness drastically. It seems to be a serious problem especially in the early phase of learning. However, it is expected that noise influence is reduced through learning in higher layers that is closer to the output neurons because the NN learns to reduce the noise effect in the network output. In reinforcement learning, noises as exploration factors are added to the motion (actor) outputs, and the fluctuations in the sensor signals caused by the exploration is concerned. It should be investigated how serious the influence of the exploration is.

E. Adjustment of discount factor

It is considered that when the time scale is flexibly adjusted for a long-duration or short-duration task, the discount factor in value learning should be also adjusted because the discount factor can be considered to represent the decay of the value due to the passage of time. In the simulation of value learning in this paper, the discount factor was given in advance. It remains as a future problem.

F. Causality traces for reinforcement learning with an RNN

As shown in this paper, the basic idea is the same between value learning with a layered neural network and supervised learning with an RNN. Another future work is to develop a way of utilizing causality traces that works efficiently in reinforcement learning using an RNN.

G. Unification of memory for output generation and that for learning

The traces are used only in the backward computation for learning. On the other hand, the memory in the forward computation for output generation is stored as the outputs of neurons. Both are the memory to store the past important information. Therefore, the author thinks that the memory for forward computation and that for backward computation might be unified.

V. Conclusion

In this paper, a general idea for retrospective learning named "Causality Trace" was introduced to make learning in time-axis more effective in neural networks. It was shown that the traces are updated without being influenced by an external time scale, but are updated subjectively and in parallel in each neuron in a neural network. In the learning of state value (critic) in a task where an agent can pass through in a short time in one type of regions and takes a long time to pass through in the other type of regions, "causality traces" showed outstanding learning ability. It was also shown that division of roles in time axis among neurons is promoted through learning.

Two major future works are to examine the ability of "causality trace" in reinforcement learning where actions are learned with trials and errors and to integrate the trace for value learning and that for learning of an RNN to work in reinforcement learning with an RNN effectively.

Acknowledgment

This work was supported by JSPS KAKENHI Grant Number 23500245. The author expresses his gratitude to the reviewers for their helpful comments.

References

[1] D. E. Rumelhart, et al., "Learning Internal Representation by Error Propagation", *Parallel Distributed Processing*, MIT Press, **1**, 318-364, 1986

[2] R. J. Williams & D. Zipser, "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks", *Neural Computation*, **1**, 270-280,1989

[3] R. S. Sutton & A. G. Barto, *Reinforcement Learning: An Introduction*, A Bradford Book, The MIT Press,1998

[4] B. Bakker, V. Zhumatiy, G. Gruener & J. Schmidhuber, "A Robot that Reinforcement-Learns to Identify and Memorize Important Previous Observations", *Proc. of IROS (IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems) 2003*, 430-435, 2003

[5] J. Gibbon, "Origins of scalar timing", *Learn. Motiv.*, **22**: 3-38, 1991

[6] J. E. R. Staddon, "Interval timing: memory, not a clock", *Trends in Cog. Sci.*, **9** (7): 312-314, 2005

[7] C. V. Buhusi & W. H. Meck, "What makes us tick? Functional and neural mechanisms of interval timing", *Nature Reviews Neurosci.*, **6**(10): 755-765, 2005

[8] H. Nakahara & S. Kaveri, "Internal-time temporal difference model for neural value-based decision making", *Neural Computation*, **22**(12): 3062-106, 2010

[9] N. D. Daw, A. C. Courville & D. S. Touretzky "Representation and timing in theories of the dopamine system", *Neural Computation*, **18**(7):1637-1677, 2006

[10] Y. Yamashita & J. Tani, "Emergence of Functional Hierarchy in a Multiple Timescale Neural Network Model: a Humanoid Robot Experiment", *PLoS Comput. Biol.*, **4**(11), 2008

[11] K. Shibata, K. Ito & Y. Okabe, "Simple Learning Algorithm for Recurrent Networks to Realize Short-Term Memories", *Proc. of IJCNN (Int'l Joint Conf. on Neural Networks) '98*, 2367-2372, 1998

[12] M. F. Samsudin & K. Shibata, "Improvement of Practical Recurrent Learning Method and Application to a Pattern Classification Task", *Proc. of ICONIP (Int'l Conf. on Neural Information Processing) 08*, **5507**: 631-638, 2009

[13] K. Shibata & S. Enoki, "Differential Trace in Learning of Value Function with a Neural Network", *Proc. of RiTA (Robot Intelligence Technology and Applications) 2012*, 55-64, 2012

[14] H. Markram, J. Lubke, M. Frotscher & B. Sakmann, "Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs", *Science*, **275**: 213-215, 1997

[15] K. Shibata, "Emergence of Intelligence through Reinforcement Learning with a Neural Network", *Advances in Reinforcement Learning*, Abdelhamid Mellouk (Ed.), InTech, 99-120, 2011