Differential Trace in Learning of Value Function with a Neural Network

Katsunari Shibata and Shuji Enoki
Department of Electrical and Electronic Engineering,Oita University
700,Dannoharu,870-1192 Oita JAPAN
shibata@oita-u.ac.jp magic.hands1988@gmail.com

Abstract : Reinforcement learning has a fatal problem of slow learning. To solve this problem, Eligibility-trace has been widely used. However, since the trace throws away old information and takes the present information constantly not depending on whether the information is important or not, long-term learning and short-term learning are incompatible. In this paper, a novel approach called "Differential trace" is proposed, in which the trace is not updated constantly, but according to the change of each neuron's output in a neural network. In other words, the time axis is subjectively adjusted in each neuron. The characteristics of the Differential trace could be observed in the learning of state value in a simple task where one-dimensional continuous environment is divided into 100 states. The learning performance is better in total than the case of Eligibility trace with either of two decay rates.

1 Introduction

In recent years, reinforcement learning[1] which is a way to learn appropriate actions through trials and errors based on reward and punishment as a scalar signal, has been attracting attention because of its autonomous learning ability. It has been shown that through reinforcement learning, necessary functions emerge in a neural network (NN) that connects from sensors to motors[2]. However, reinforcement learning has a fatal problem of slow learning due to the exploration and also extraction of important information from a huge amount of spatio-temporal sensor signals. In the original algorithm, which is called "Onestep" method here, only the value function at the previous time step is learned from the present value and reinforcement signal, and that is also one major reason of slow learning. To solve this problem, Eligibility-trace (E-trace)[3][4] has been widely used that enables to update the values for the past series of states in real-time by holding the information about the visited states in the past. However, the past information in the E-trace decays constantly regardless of whether it is important or not. Therefore, if the trace is set to decay slowly in order to hold the past state, since taken information for a moment is relatively small than the whole information held in the trace, learning for quick motions becomes slow. On the other hand, if the trace decays quickly, the past information can be held only in a short period.

Here, for example, suppose that you are driving a car. Do you think as "go straight, go straight ..., go straight, turn right," with a fixed-time interval? No one might think as such. Rather, you must think as "now go straight, and turn right at the corner with the second signal" and so on. The thinking in the former way is obviously inefficient. Learning of action planning for one day with the sampling rate of 100 msec is very inefficient, while fine actions cannot be learned with the sampling rate of one minute. Accordingly, as in the latter example, important points should be focused and trivial points should be ignored. This means that time passes slowly for the important state and passes quickly for the trivial state. That is to say, if the time axis is adjusted subjectively, learning is expected to be more efficient.

When a robot learns something in the real world, it obtains sensor signals from various sensors, and they change from moment to moment. The quantity of information is huge, and so it is a difficult and intelligent task to know what is important among it. However, a neural network has an ability to represent important information in its hidden layer through learning without any directions from humans. Furthermore, the division of roles among hidden neurons progresses through learning. It can be said that each neuron represents the state subjectively. In the proposed method in this paper, the state change that is the time derivative of the output in each neuron is utilized to adjust how much present information is taken into the traces. The trace, the authors call "differential trace (D-trace)". In each neuron, when the output changes largely, considering that the state changes largely, the traces throw away the information about the old inputs and take the present inputs largely. When the output does not change so much, it is considered that the state does not change largely and there is no need to take the present inputs into the trace. In this way, each neuron takes the inputs in the traces only when its output changes, and keeps the previous traces when its output does not change. Furthermore, as mentioned, each neuron responds different events due to the division of roles among hidden neurons, and so the information held in the traces is different among the hidden neurons. The D-trace promotes the learning for the past event, and the learning promotes the effective memory of the past in the D-trace. The synergetic effect is expected to accelerate reinforcement learning when considering life-long learning where the knowledge or representation acquires through learning can be utilized in the following learning.

In this paper, the algorithm of D-trace is formulated comparing with the E-trace and its characteristics are observed in a simple task in which only value function is learned.

2 "Differential trace" vs. "Eligibility trace"

In this section, E-trace, D-trace and the learning of value function using the traces are formulated when they are implemented in a layered neural network(NN). Only the learning of value function is focused and action selection is not considered here. In the original "Onestep" reinforcement learning, each weight value

w in the neural network are updated to decrease TD(Temporal Difference) error as

$$E_t = \frac{1}{2}\, TDerr_t^2 \qquad (1)$$
$$where \;\; TDerr_t = r_{t+1} + \gamma\, O_{N1,t+1} - O_{N1,t}$$

$$\begin{aligned} \Delta w_{kji,t} = -\eta \cdot \frac{\partial E_t}{\partial w_{kji}} &= \eta \cdot\; TDerr_t \;\cdot\; \frac{\partial O_{N1,t}}{\partial w_{kji}} \\ &= \eta \cdot\; TDerr_t \;\cdot\; \frac{\partial O_{N1,t}}{\partial U_{kj,t}} \cdot \frac{\partial U_{kj,t}}{\partial w_{kji}} \\ &= \eta \cdot\; TDerr_t \;\cdot\; C_{kj,t} \cdot\; O_{k-1\;i,t} \qquad (2) \\ where\; C_{kj,t} &= \frac{\partial O_{N1,t}}{\partial U_{kj,t}} \end{aligned}$$

where subscript k, j, i indicate the layer number, the number of signal-receiving neuron, and the number of signal-sending neuron in the NN. N indicates the output layer, and there is only one output neuron in the layer that is learned to represent the state value. U and O is the internal state and output of a neuron, and $O = sigmoid(U)$ where $sigmoid$ is the sigmoid function as an output function. r is a given rewards, γ is a discount factor and η is a learning rate. $C_{kj,t}$ indicates the contribution of the neuron j in the layer k to the output, and that is similar to propagated error δ in BP (Error Back Propagation) but no error information is included. It can be computed through backward propagation from the output neuron in the same way as δ. In this algorithm only the present influence of the weight to the output neuron $\partial O_{N1,t}/\partial w_{kji}$ is considered and then the NN is updated so as to reduce the TD error for the present state.

On the other hand, using E-trace e that accumulates the past information in it as a discrete approximation of the first-order lag, the values for the past states also can be updated in real-time as

$$e_{kji,t} = \gamma\lambda\; e_{kji,t-1} + (1-\lambda)\; C_{kj,t} \cdot\; O_{k-1\;i,t} \qquad (3)$$
$$where\; \lambda \in [0, 1)$$

$$\Delta w_{kji,t} \;=\; \eta \cdot\; TDerr_t \;\cdot\; e_{kji,t} \qquad (4)$$

where λ is a constant to decide how fast the E-trace decays. If λ is large and close to 1.0, the E-trace decays slowly. Since λ is a constant, the E-trace decays constantly. When E-trace is formulated, generally, no coefficient is multiplied to the second term in Eq(3). However, here, $(1-\lambda)$ is multiplied to make the E-trace compatible with the other cases. By this, if $C_{kj,t} \cdot\; O_{k-1\;i}$ does not change for a long time, the E-trace converges to the value.

Finally, the D-trace d is updated according to the output change of each neuron instead of the constant λ as

$$d_{kji,t} = \gamma\; (1 - |\Delta O_{kj,t}|)\; d_{kji,t-1} + |\Delta O_{kj,t}|\; C_{kj,t} \cdot\; O_{k-1\;i,t} \qquad (5)$$
$$where \;\; \Delta O_{kj,t} \;=\; O_{kj,t} - O_{kj,t-1}$$

$$\Delta w_{kji,t} = \eta \cdot \ TD\_err_t \ \cdot d_{kji,t}. \tag{6}$$

When Eq(5) is seen as the difference approximation of first-order lag, large ΔO means small time constant, and the trace value is replaced largely by the present input. It can be considered that the time passes fast. When ΔO is small, the time constant is large, and the time passes slowly. In other words, it is possible to adjust the time axis flexibility according to the change of subjective state ΔO. Since ΔO is calculated in each neuron, it is known that the information that each trace holds is different among neurons.

Fig.1 illustrates the temporal change of E-trace and D-trace. It can be seen that when the output changes, the present input is taken into the D-trace, while the D-trace value is held when the output does not change except for the decay by the discount factor γ.

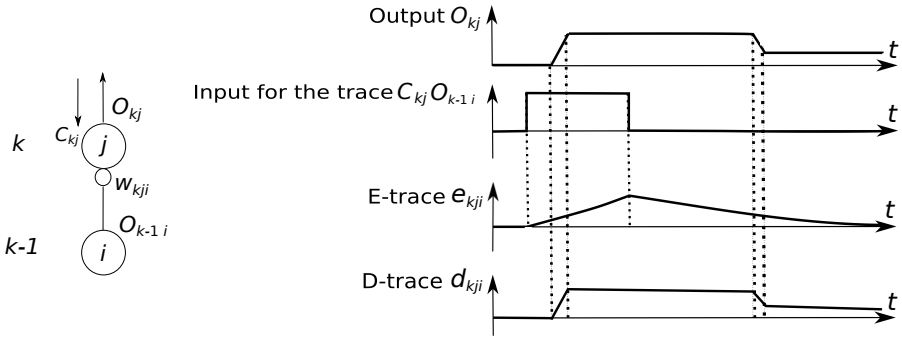


Fig. 1. Comparison of temporal change between E-trace and D-trace.

3 Task

In this section, a task in which the characteristics of each trace are easy to be seen is described. As shown in Fig.2, an agent just moves rightward from the left end to the goal at the right end constantly for 10,000 steps with no action selection. When it reaches the goal, reward 1.0 is given. The agent learns the state value during the episode using a 3-layer neural network. The way from start to goal is divided into 100 discrete states and it takes 100 steps to go through each state. There are two types of input signals that respond locally only when the agent is on one state. One type of them takes the value of 1 when the agent is on the corresponding state, and 0 otherwise. The other type inputs change its value linearly from 0 to 1 over 100 steps on the state so that the agent can identify the place in the state. In this task, since each input represents only local information and the generalization ability of the neural network does not work effectively, the effect of holding the past state in the traces can be seen easily.

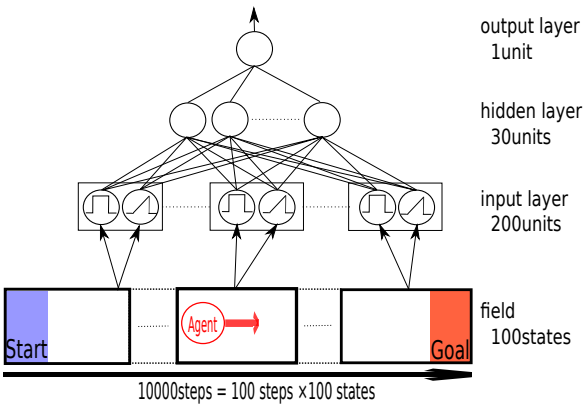


Fig. 2. A task to observe the characteristics of E-trace and D-trace.

Parameters used in this learning are shown in Table 1. The learning rate also influences the learning speed, and is roughly optimized through trials and errors. The reason why it is small in the case of Onestep is that some oscillation is observed when the learning is done with the same rate (2.0, 20.0) as the other cases. The discount factor γ was set so as that the state value is 0.2 at the first step and 1.0 at the goal. The sigmoid function whose value ranges from -0.5 to 0.5 is used. The value is linearly transformed to the range of [0.0, 1.0] from the range of [-0.4, 0.4] in the output of the NN. The value is limited from -0.4 to 0.4 even though it is less than -0.4 or more than 0.4 originally. Simulation results are compared among "Onestep","E-trace"and "D-trace", and in the E-trace case, the results for two different decay rates $\lambda = 0.999$ and $\lambda = 0.99$ are shown.

Table. 1. Learning parameters

number of neurons in each layer	200-30-1	
learning rate (hidden → output)	Onestep	1.0
	E-trace,D-trace	2.0
learning rate (input → hidden)	Onestep	10
	E-trace,D-trace	20
initial weight of neurons	random [-1.0, 1.0]	
reward	1.0	

4 Learning result

Fig. 3 shows the learning curve for each case. The vertical-axis indicates the absolute value of the difference between actual state value and the ideal one that is decided from the discount factor γ. Each plot shows the average of the error over one episode.

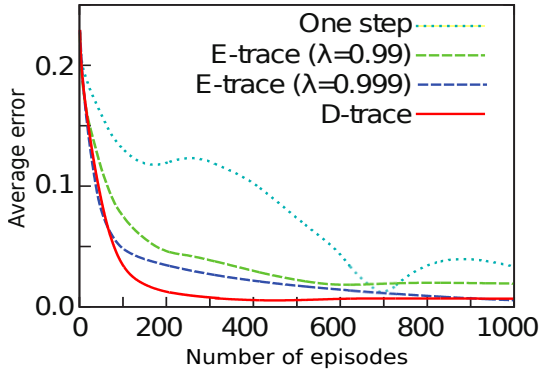
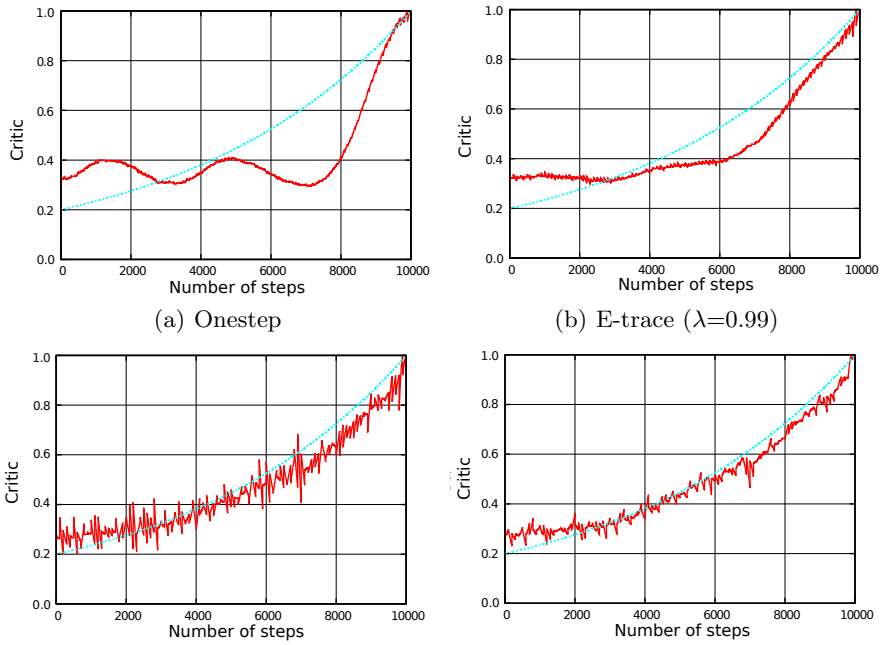


Fig. 3. Change of the average error between the network output and ideal state value during learning is compared among Onestep, E-trace and D-trace.

The learning speed is slower in Onestep learning than the cases of using a trace. In the case of E-trace($\lambda = 0.99$), learning is faster than in the Onestep learning, but is slower than the other two trace learning. As for the other two cases, learning is slightly faster in the case of E-trace($\lambda = 0.999$) at first, but, the difference in D-trace becomes smaller than the case of E-trace($\lambda = 0.999$). In the case of Onestep learning, the difference becomes larger again at around 700th episode after once it became small. Same phenomena are observed also in other cases although the increase is not so large. It was found that such phenomena occur by the influence of local representation of input signals and non-linearity in each neuron. Small learning rate decreases the influence, but learning becomes slow.

To show how the learning progresses in each case, Fig.4 shows the state value at the 100th episode. In the case of Onestep, the value is formed only after around the 7,000th step, and in the case of E-trace($\lambda = 0.99$), the range that the value is formed is a bit wider than the case of Onestep, but appropriate value does not reach in the early states.

On the other hand, in the cases of E-trace($\lambda = 0.999$) and D-trace, the rough shape of the value is formed over whole the episode. However, large high-frequency components are seen in the case of E-trace($\lambda = 0.999$). It is confirmed that the shape is influenced by the initial connection weights. This means that E-trace with a large λ is good at forming a rough shape of value function in a long range, but is not good at forming appropriate value change in a short range.



(a) Onestep (b) E-trace (λ=0.99)

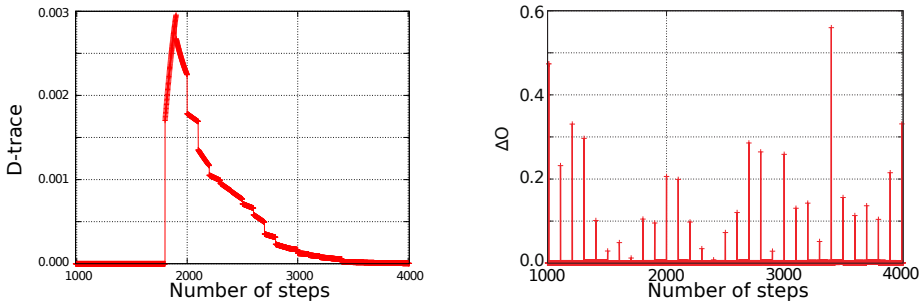
(c) E-trace (λ=0.999) (d) D-trace

Fig. 4. Comparison of the state value at the 100th episode.

In the case of D-trace, high-frequency components are not so large, and both local and global learning seem to be progressing. Due to the limited number of pages, the result is not shown, but in the case of E-trace for an intermediate decay rate $\lambda = 0.996$, high-frequency component is at the same level as in the case of D-trace, but the global shape is closer to the case of $\lambda = 0.99$.

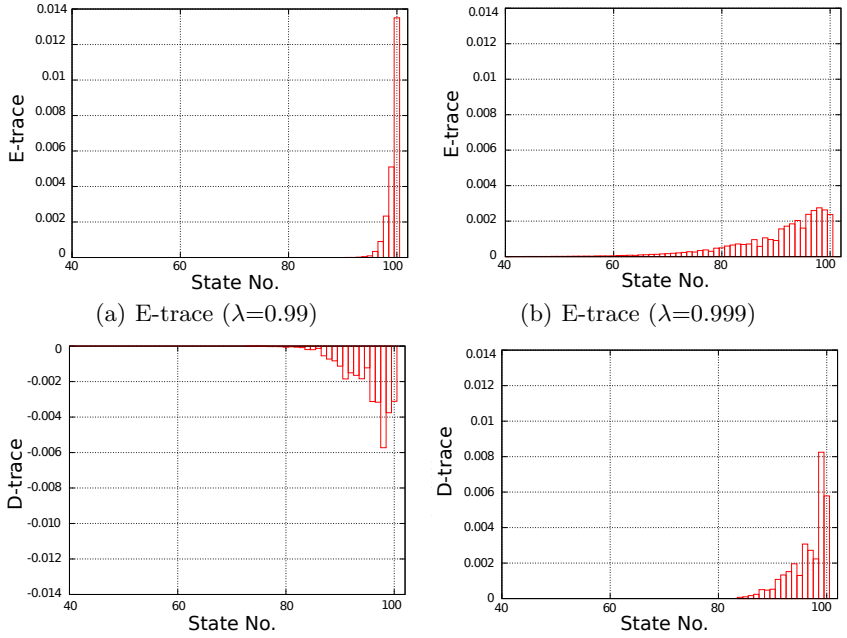
Fig. 5 shows how D-trace changes as time goes by together with ΔO that is the temporal change in the output of the hidden neuron. Each hidden neuron has one trace for each input, and 200 traces in total. Since the trace shown in Fig. 5 is for the binary input from the 18th state, the value largely increased when the agent reached the 18th state at the 1801st step. After that, the trace increased for 100 steps because the input is 1.0, but the increase ratio is not so large due to the small ΔO. After leaving the state, the value mainly decreased at every 100 step when a state transition occurs. The reduced value is not constant, but depends on the value of ΔO.

Next, it is observed how each trace holds the past information. Fig. 6 shows the trace values of one hidden neuron at the final step in the 100th episode for the 4 cases. Fig. 6 shows the traces for the 100 binary inputs that take always 1.0 when the agent is on the corresponding state. Each trace takes the product of contribution to the output C_{2j} and the input $O_{1,i}$. The contribution C_{2j} depends on the weight to the output and also the derivative of output function in the



(a) change of a D-trace (b) change of ΔO

Fig. 5. Change of a D-trace and ΔO of one hidden neuron in the 1000th episode. The D-trcae is for the input whose value is 1.0 at the 18th state.



(a) E-trace (λ=0.99) (b) E-trace (λ=0.999)

(c) D-trace in the 2nd neuron (d) D-trace in the 25th neuron

Fig. 6. The trace values for the binary inputs from the 100 states in one hidden neuron at the last(10000th) step in the 100th episode.

hidden and output neurons. Therefore, they influence the trace values, but the rough tendency can be seen about which input signal is taken more.

It can be seen that in the case of $\lambda = 0.99$, the value of E-trace for the states close to the goal is large, but that has decayed for the distant states from the goal. On the other hand, in the case of $\lambda = 0.999$, the value is not so large even for the states close to the goal, but the decay is slow and so the trace

for the distant state from the goal also has some value. D-trace holds the past information, but in Fig. 6(c), the trace for the input from the 98th state, which is the 3rd state from the goal, takes a larger value comparing with the other trace. The profile is different from that in a different hidden neuron in Fig.6(d). This means that each hidden neuron can hold different past information, and that is supposed to be one reason for the efficient learning.

5 Conclusion

In this paper, a novel approach for learning of value function called "Differential trace (D-trace)"was proposed. That enables to hold the important information for each neuron and to learn the value function for past states efficiently in real time. In the learning of a simple task in a one-dimensional continuous environment with many local sensors, the characteristics of D-trace was observed, and it was confirmed that the learning performance is better in total than the case of Eligibility trace in either case of fast decay or slow decay.

In the real world, there is a vast amount of information, and so subjective adjustment of time axis must be required to extract important events and to learn effectively. Accordingly, D-trace has a very large potential in the autonomous learning in the real world, and further investigation is strongly demanded. Similar concept has been already introduced in the learning of a recurrent neural network, and it was shown that it works even though the computational cost and necessary memory capacity is as small as $O(N^2)$ where N is the number of neurons[5][6]. The integration of the method and D-trace is expected.

Acknowledgment

This work was supported by JSPS Grant-in-Aid for Scientific Research #23500245.

References

1. Rumelhart, D. E. et al., Learning Internal Representation by Error Propagation, Parallel Distributed Processing, MIT Press, Vol. 1, pp. 318-364 (1986)
2. Shibata, K., Emergence of Intelligence through Reinforcement Learning with a Neural Network, Advances in Reinforcement Learning, Abdelhamid Mellouk (Ed.), In-Tech, pp.99-120 (2011)
3. Sutton, R.S. & Barto, A. G., Reinforcement Learning, MIT Press, pp163-192 (1988)
4. Bakker,B., Zhumatiy,V, Gruener,G.& Schmidhuber,J., A robot that reinforcement-learns to identify and memorize important previous observation. Intelligent Robots and Systems, pp230-235 (2003)
5. Shibata, K., Ito, K. & Okabe, Y., Simple Learning Algorithm for Recurrent Networks to Realize Short-Term Memories, Proc. of IJCNN (Int'l Joint Conf. on Neural Networks), pp. 2367-2372 (1988)
6. Samsudin, M. F., Hirose, T., & Shibata, K., Practical Recurrent Learning (PRL) in the Discrete Time Domain, Neural Information Processing of Lecture Notes in Computer Science, Vol. 4984, pp. 228-237