ニューラルネットを用いた価値関数の学習における微分型トレースの提案

○榎修志 柴田克成 (大分大学)

Differential Trace in Learning of Value Function with a Neural Network

∗Shuji Enoki and Katsunari Shibata (Oita University)

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 time derivative 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 any case of Eligibility trace with a variety of decay rates.

Key Words: Differential Trace,Reinforcement Learning, Neural Network, Eligibility Trace,Temporal Difference Learning

1 はじめに

近年、強化学習 [1] は、試行錯誤を行って報酬と罰といった強化信号を基に目的に沿った行動を自律的に獲得する学習方法として注目を集めている。また、センサからモータまでをニューラルネット (NN) で構成し、強化学習で学習することによって必要な機能が内部に創発することが示されている [2]。しかし、これらの学習方法は、強化学習において状態や行動を探索するためや、ニューラルネット内で膨大な時空間の情報の中から重要な情報を抽出できるようになるまでに時間がかかるため、学習が遅いという致命的な欠点がある。また、元々のアルゴリズムでは、学習を行う際に学習主体となるエージェントが報酬を得ても 1 つ前の時刻の行動に対する状態価値しか更新を行うことができないため非常に効率が悪い。

この問題を解決するために、エージェントが過去に訪問した情報を Eligibility-trace(以下、E-trace)[3][4] に保持し、学習に用いることで、過去に訪問したそれぞれの状態に対して同時に価値の更新を行う方法がある。しかしながら、E-trace は、重要であるかどうかに関わらず刻々と入力の情報を取り入れ、過去に取り込んだ情報は常に指数関数的に減衰される。そのため、過去の状態を保持するためにトレースをゆっくり減衰するように減衰率 λ を設定すると、トレースに保持される情報の内、近い過去の情報が占める割合が相対的に小さくなるため、過去から現在までの大まかな学習は進むが、細かい時間間隔の学習はなかなか進まない。逆にトレースを早く減衰させるように減衰率 λ を設定すると、現在の時刻に近い過去の情報しか保持することができず、遠い過去の情報はほとんど残らない。よって、遠い過去に対する学習があまり進まないためトレースとしての効果が薄くなってしまう。

では、人間はどのように効率よく過去の情報を利用しているのだろうか。例えば、車を運転をしている時に、決まった時間ごとに「前に、前に、前に、右へ…」と考えるだろうか。まず、そのように考える人はいないと思われる。むしろ、「今は前に進み、そして次の信号を右に曲がろう」と考えるはずである。前者のように、一定時間間隔で行動を考えることは明らかに効率的ではない。例えば、時間幅を 100msec として 1 時間分の学習を行うのは大変効率が悪く、逆に 1 分間隔では細かい学習を行うことはできない。それに対して、後者の例のように重要なイベントが起きて状態が大きく変化したところを重点的に記憶して後の学習に利用し、そうでないところは記憶に残さないようにすれば、学習は非常に効率的になると考えられる。これは言い換えれば、重要な状態の変化があったときには時間の進みを遅くして重点的に、そうでない場合は時間の進みを速くして大雑把に学習することに相当する。つまり、人間のように主観的に時間の調整を行うことができれば、学習も効率的に行うことができると期待される。

何かイベントが起きるとき、NN 内の状態も大きく変化すると考えられる。例えば、信号機があれば、それを認識するニューロンの出力は大きく変化するだろう。そこで、各ニューロンの出力が大きく時間変化したとき、イベントが起こって状況が変化したと考え、トレースに保持されているそれまでの古い情報を破棄し、現在の情報に大きく入れ替える。逆に、出力がそれほど変化がないときは状況が変化していないと考え、そのときの情報は保持する必要がないとしてトレースに現在の情報を取り込まないようにする。このようにニューロンの出力の時間変化の大きさに応じてそのときの情報を保持すれば、重要なイベントでないところは意識しないで学習ができる。そこで、NN 内で各ニューロンの出力の時間変化、つまり、時間微分の情報を用いて、ニューロン自らが時間の調整をして必要なポイントに重点をおいて記憶し、学習に用いることを目指したものが提案する"微分型トレース"である。

学習前の NN は、通常、乱数を用いて初期重み値を設定するため、中間層ニューロンごとに異なった情報を表現するようになる。また、学習することによって、中間層ニューロンは NN が適切な出力をするために必要な情報を表現するようになり、各ニューロンごとに異なった情報を表現して相互に役割分担するようにな

る。それによって、各中間層ニューロンの出力の時間変化が異なるため、ニューロンごとに異なる過去の情報をトレースに保持することができ、別々の過去のイベントに対して並列に学習することができる。また、考慮する必要がない情報に対しては中間層ニューロンの出力の変化が学習を通して小さくなることが期待される。出力が変化しなくなればトレースはより過去の情報を保持できるようになり、より過去の状態に対して学習できるようになる。つまり、学習と微分型トレースとの相互作用により学習が加速することも期待できる。本論文では、E-trace の場合と比較しながら、微分型トレースの定式化を行うとともにシミュレーションを通してその特性を観察する。

2 微分型トレース

ここではリカレント構造のない階層型ニューラルネットを用いた価値関数の TD(Temporal Diffference) 学習の方法、E-trace を適用した際の学習方法、微分型トレースの定式化とそれを利用した学習方法を示す。また、ここから先ではトレースを用いない学習を Onestep 学習と呼ぶ。まず、Onestep 学習では式 (2) で定義される TD 誤差を 0 に近づけるために時刻 t での出力 $O_{N1,t}$ を変化させて誤差 E_t を減少させるように最急降下法によって NN 内の各重み値 w_{kji} の更新を行う。

$$E_t = \frac{1}{2}\, TDerr_t^2 \tag{1}$$

$$TDerr_t = r_{t+1} + \gamma\, O_{N1,t+1} - O_{N1,t} \tag{2}$$

$$\begin{aligned}
\Delta w_{kji,t} &= -\eta \cdot \frac{\partial E_t}{\partial O_{N1,t}} \cdot \frac{\partial O_{N1,t}}{\partial w_{kji}} \\
&= \eta \cdot TDerr_t \cdot \frac{\partial O_{N1,t}}{\partial w_{kji}} \\
&= \eta \cdot TDerr_t \cdot \frac{\partial O_{N1,t}}{\partial U_{kj,t}} \cdot \frac{\partial U_{kj,t}}{\partial w_{kji}} \\
&= \eta \cdot TDerr_t \cdot C_{kj,t} \cdot O_{k-1\ i,t} \\
&\quad \text{ただし、} C_{kj,t} = \frac{\partial O_{N1,t}}{\partial U_{kj,t}}
\end{aligned} \tag{3}$$

添字 k,j,i はそれぞれニューラルネット中の該当ニューロンのある層、層の中のニューロンの番号、そのニューロンに入る入力の番号を示している。N は出力層を表し、ここでは評価関数のみの学習を考えていることから O_{N1} は状態価値を出力する出力ニューロンのことを指す。$U_{kj,t}$、$O_{kj,t}$ は時刻 t における k 層目 j 番目のニューロンの内部状態とシグモイド関数を用いたその出力を表す。γ は割引率、η は学習係数を示す。

$C_{kj,t}$ は時刻 t における k 層目 j 番目のニューロンの出力ニューロンに対する貢献度を示し、Fig.1 のように誤差逆伝播 (BP,Error Back Propagation) 法 [1] の伝播誤差信号と同様に出力層で計算された C_{N1} を伝播させて中間層の C_{kj} を求めることができる。3 層の場合の具体

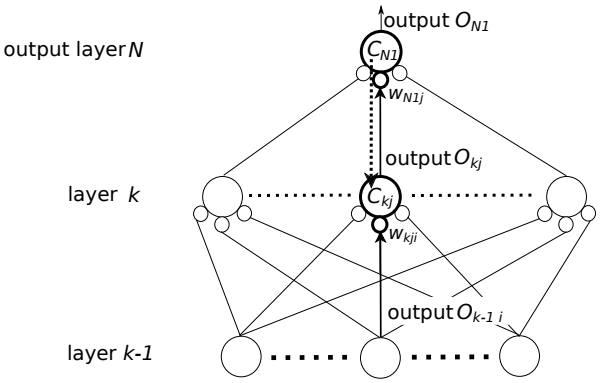


Fig. 1: The computation of contribution C_{kj} by back propagation.

的な計算方法は

$$\begin{aligned}
C_{kj,t} &= \frac{\partial O_{N1,t}}{\partial U_{kj,t}} \\
&= \frac{dO_{N1,t}}{dU_{N1,t}} \cdot \frac{\partial U_{N1,t}}{\partial O_{kj,t}} \cdot \frac{dO_{kj,t}}{dU_{kj,t}} \\
&= C_{N1,t} \cdot w_{N1j} \cdot \frac{dO_{kj,t}}{dU_{kj,t}}
\end{aligned} \tag{4}$$

と表される。また、$\delta_{kj,t} = TDerr_t \cdot C_{kj,t}$ とすれば $\delta_{kj,t}$ を伝播誤差信号とする通常の BP 法による学習として表現できるが、ここではトレースを使った場合との関連を見やすくするため、あえて貢献度 $C_{kj,t}$ を用いて表現した。

E-trace は式 (5) のようにトレース e に過去の情報を蓄えて、現在の入力の代わりに用いることにより、過去に訪問した状態の価値を過去に遡ることなく更新する。

$$e_{kji,t} = \gamma\lambda\, e_{kji,t-1} + (1-\lambda)\, C_{kj,t} \cdot O_{k-1\ i,t} \tag{5}$$

$$\Delta w_{kji,t} = \eta \cdot TDerr_t \cdot e_{kji,t} \tag{6}$$

式 (5) からトレース e は、割引率 γ による減衰を除いて考え ($\gamma = 1$ とし)、Δt を時間の刻み幅とすると毎時刻 $C_{kj,t} \cdot O_{k-1\ i,t}$ を入力とし、$\Delta t/(1-\lambda)$ を時定数とする差分近似した一次遅れ系である。したがって、過去に訪問した状態に対する情報を減らしつつ現在に至るまでの情報を保持していく。そして、式 (6) によって現時刻の TD 誤差を掛けることで、過去に訪問した状態に対して状態価値を更新する。減衰率 λ は E-trace がどれだけ減衰するかを決定する定数であり、$\lambda = 0$ ならばトレースに過去の情報を保持しないので Onestep 学習と同様の学習となる。λ は定数なので、トレースにおける過去の状態に対する情報は指数関数的に減衰する。E-trace は通常、式 (5) の第二項に係数を付けない形で表現する。しかし、ここでは他の場合に対して互換性を確保するため、トレースに入力される $C_{kj,t} \cdot O_{k-1\ i,t}$ が長く時間変化がないときに一次遅れ系として入力の値に収束するように係数として $(1-\lambda)$ を乗算した。

微分型トレース d は、各ニューロンの出力の時間変化によって以下のように更新する。

$$d_{kji,t} = \gamma\,(1 - |\Delta O_{kj}|)\, d_{kji,t-1} + |\Delta O_{kj}|\, C_{kj,t} \cdot O_{k-1\ i,t} \tag{7}$$

$$\text{ただし、} \Delta O_{kj,t} = O_{kj,t} - O_{kj,t-1}$$

$$\Delta w_{kji,t} = \eta \cdot TD\_err_t \cdot d_{kji,t} \quad (8)$$

式 (7) を γ の影響を除いて考えると、保持していたトレースの値 $d_{kji,t-1}$ を、$(1 - |\Delta O|)$ を掛けることによって $|\Delta O|$ の分だけ破棄し、破棄した分だけ現在の入力 $C_{kj,t} \cdot O_{k-1\ i,t}$ を取り入れることになる。これを差分近似した一次遅れ系の式として見ると、$\Delta t/|\Delta O|$ が時定数となる。よって、$|\Delta O|$ が大きいほど時定数が小さくなることを意味し、時間の進みは早くなると捉えることができ、入力の値に速く近づくようになる。また、$|\Delta O|$ が小さいほど時定数が大きくなり、時間の進みは遅くなると捉えることができ、トレースの値はあまり変化せず、それまでの情報を保持する。Fig.2 に E-trace と微分型トレースの時間変化の例を示す。E-trace は現在の入力 $C_{kj,t} \cdot O_{k-1\ i,t}$ を刻々と取り込み、$C_{kj,t} \cdot O_{k-1\ i,t}$ の値が 0 ならば時間の経過とともにトレースの値は一様に減衰していくのに対して、微分型トレースは出力が時間変化したときに現在の入力をトレースに取り込み、出力が変化していないときはトレースは保持される。さらに、E-trace では各ニューロンごとに同じように過去の情報の保持を行うのに対して、微分型トレースは $|\Delta O|$ が各ニューロンごとに違うため、各ニューロンで保持する情報は異なり、ニューロンごとに別々のイベントに対して学習できる。

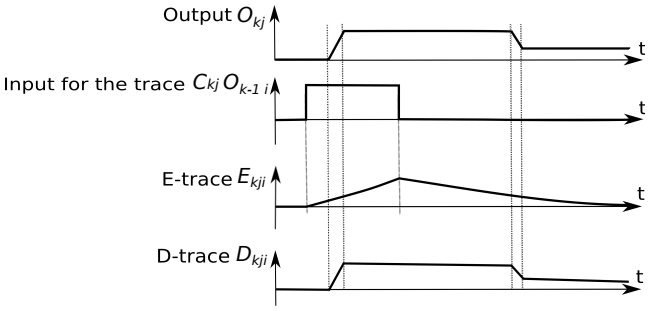


Fig. 2: Characteristics in temporal change between E-trace and D-trace.

3 シミュレーション設定

この章では E-trace、微分型トレースを用いた学習の特性を分かりやすく観察するためのタスクを示す。Fig.3 のように、エージェントは行動選択なしで 10,000 ステップかけて 1 次元のフィールドを連続的に左から右へ移動してゴールに向かい、ゴールに到達すると報酬 1.0 が与えられる。スタートからゴールまでを大きく 100 状態に分け、1 つの状態内を 100step かけて徐々に移動し、次の状態へと移動する。エージェントは 3 層の NN を用いて状態価値を学習する。100 に分けたそれぞれの状態に対して局所的に反応する 2 種類の入力を用意する。その 2 種類のうち一つはその状態にエージェントがいるときに 1、その状態の外にいるときは 0 を 2 値で入力する。もう一つは、その状態内でのエージェントの場所を示すように 100step かけて 0 から 1 に直線的に変化し、その状態にいないときは常に 0 を入力する。100 個の状態それぞれに対して 2 個の入力があるため、全部で 200 個の入力が NN に与えられる。このタスクでは各入力は局所的な情報しか表現していないことから、ニューラルネットの汎化が効きにくいため、トレースによる過去の状態の保持の効果が見やすい。

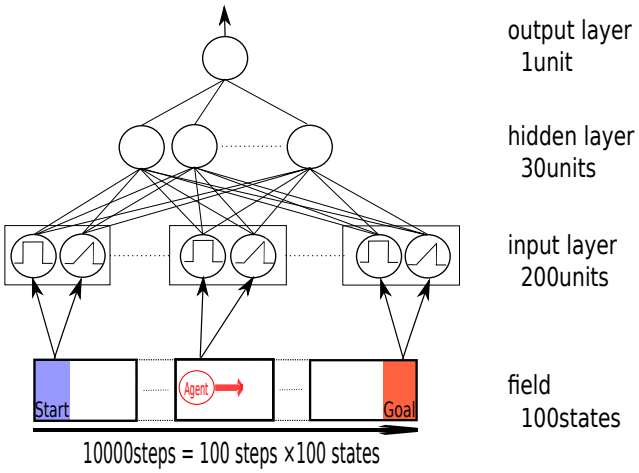


Fig. 3: A task to observe the characteristics of E-trace and D-trace.

学習に用いた各種パラメータを Table 1 に示す。学習係数は学習速度に影響するが、試行錯誤して学習が不安定にならない範囲で速く収束するものを採用した。Onestep 学習の場合は他の場合と同じ学習係数で学習を行ったところ、学習曲線にいくつか振動が観測されたため、他の場合よりも小さく設定した。割引率 γ は最初のステップで状態価値が 0.2、ゴールで 1.0 となるように設定した。ニューロンの出力関数は値域が-0.5 から 0.5 のシグモイド関数とし、出力の値域中の-0.4 から 0.4 を 0 から 1 に線形変換して状態価値とした。このとき、ニューロンの出力の値が-0.4 より小さい値、0.4 より大きい値である場合はそれぞれニューロンの出力の値を-0.4、0.4 とした。シミュレーション結果は ”Onestep 学習 ”, ”E-trace ”と ”微分型トレース ”で比較し、さらに ”E-trace ”の場合は減衰率を 0.99、0.995、0.999 の 3 種類の場合を観測した。

Table 1: Parameters used for learning

number of neurons in each layer	(input) 200-30-1 (output)	
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 connection weight	random [-1.0, 1.0]	
reward	1.0	

4 学習結果

各学習方法における学習曲線を Fig.4 に示す。縦軸は実際の状態価値と割引率から決まる理論値との差の絶対値をとり、1 試行通した絶対値の平均がそれぞれの場合で学習とともにどのように変化したかをそれぞれ示している。

まず、Onestep 学習はトレースを用いた各学習より学習が遅いことが分かる。E-trace(λ = 0.99) の場合は Onestep 学習より早く学習することが分かるが、他の

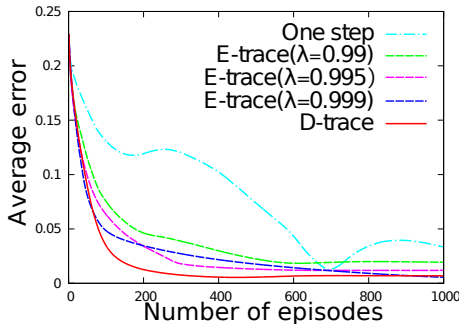
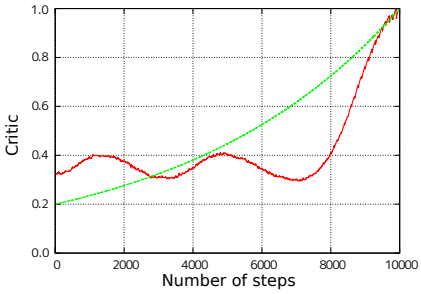


Fig. 4: Comparison of learning curves. The vertical axis indicates the difference between actual state value and ideal one.

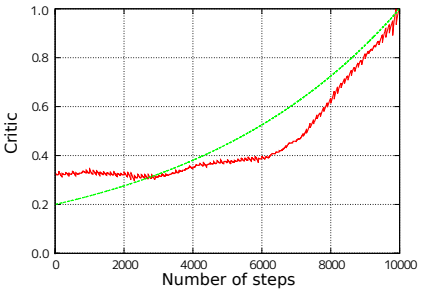
トレースを用いた学習よりは学習が遅い。E-trace(λ = 0.999) の場合は学習初期に関しては一番早く誤差が減っているが、100 試行前より誤差の減少の割合が小さくなっている。その後、再び約 700 試行以降で他の E-trace より誤差が小さくなっている。E-trace(λ = 0.995) の場合は約 200 試行から約 700 試行の間で他の E-trace より誤差が小さくなっているが、基本的には λ = 0.99 と λ = 0.999 の場合の中間的な変化をしている。一方、微分型トレースは学習初期では E-trace の中で一番学習が速い λ = 0.999 の場合より少し遅いが、ほぼ同じ割合で誤差は減少し、約 100 試行以降はどの場合よりも早く誤差が小さくなり、学習の収束が早いことが分かる。

Onestep 学習の誤差が試行約 700 回目以降で大きくなっているが、これは局所的な入力の影響で、一度理想値に近づいた後にさらに学習すると理想値から離れていく現象であることを確認した。また、よく見るとトレースを用いた場合にも、いったん誤差が減少した後にわずかながら上昇する傾向が見られる。学習係数を小さくすると、その影響は減少するが、学習速度が遅くなるためこのまま掲載した。

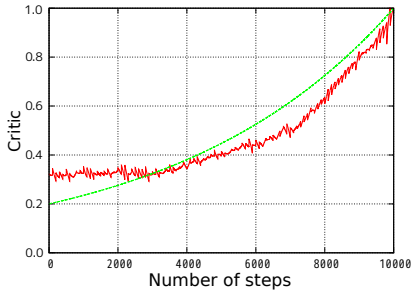
次に、それぞれの学習方法についてどのように学習が進んでいるかを観察するために、Fig.5 に 100 試行目の状態価値の分布を示す。Onestep 学習で Fig.5(a) のように報酬を得る最終 step から状態価値は形成されているが、この時点では 9000step 付近以前は報酬を得たことによる学習は全く進んでいない。しかし、他の場合と比べて、細かい振動が小さくなって滑らかに変化しており、各 step において、1 つ未来の状態価値による現在の状態価値の学習は進んでいることがわかる。それに対して、Fig.5(b) の E-trace (λ = 0.99) は Onestep 学習より全体の状態価値の学習が進んでいるように見えるが、やはり試行の初期の状態の学習は進んでいない。一方、Fig.5(d) の E-trace(λ = 0.999) と Fig.5(e) の微分型トレースの場合、大まかではあるが全体の状態価値が理想値に沿って形成されており、このため Fig.4 の学習曲線において両者の誤差の減りが大きかったと言える。しかしながら、E-trace(λ = 0.999) の場合は各状態単位での細かい振動の振幅が他の場合より大きい。初期重み値を変更したときにこの波形は変わることと、各トレースの振動パターンとの類似性から、この波形は初期重み値の影響によるものと考えられる。またこれは、「はじめに」でも述べたように、E-trace の減衰率 λ をゆっくり減衰するように設定すると、長い範囲での価



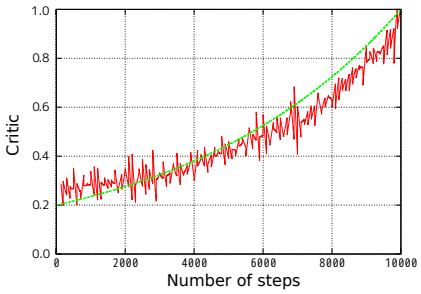
(a) Onestep



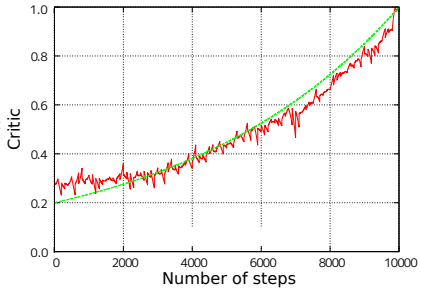
(b) E-trace (λ=0.99)



(c) E-trace (λ=0.995)



(d) E-trace (λ=0.999)

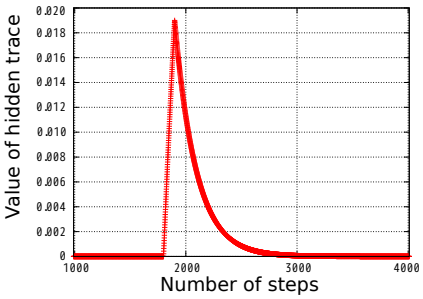


(e) D-trace

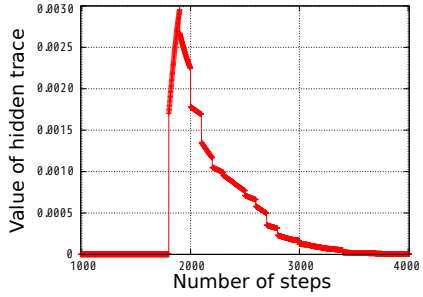
Fig. 5: Comparison of the state value at the 100th episode

値関数を大まかに形成するには良い。しかし、現時刻の情報を多く取り込まないため、トレースに保持された情報の中で近い過去の情報が占める割合が小さくなり、細かい時間間隔の学習が進まないことを示している。このため、Fig.4 の学習曲線で E-trace(λ = 0.999) の場合は、これ以後の誤差の減りが小さくなったものと考えられる。E-trace(λ = 0.995) の場合は E-trace (λ = 0.99) と E-trace (λ = 0.999) の中間的な学習のように見える。微分型トレースの場合は E-trace (λ = 0.999) とほぼ同じように全体の状態価値が理想値に沿って形成されており、さらに、細かい振動の振幅は E-trace (λ = 0.999) より小さく、細かい時間間隔の学習も進んでいることが分かる。

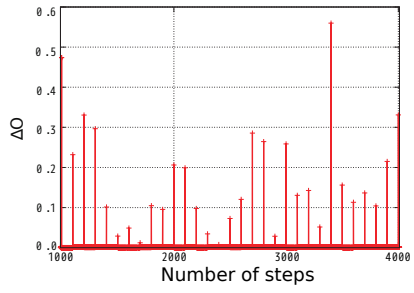
次に実際に微分型トレースの値が 1 試行中にどのように変化するかを観察する。それぞれの中間層ニューロンは各入力ごとに 1 つトレースがあることから 1 つのニューロンに合計 200 個のトレースがある。ここでは学習終了後のある中間層ニューロンにおける状態 18(step1801 から step1900) に 2 値で反応する入力に対する微分型トレースと、比較のため、E-trace(λ = 0.995) の値の時間変化を Fig.6 に示す。状態 18 にエージェントが訪問するまで入力は 0 であるために、どちらの場合もトレースの値は 0 となっている。E-trace の場合は、その後、入力に 1 が入ると上昇し、状態が 19 に移り変わることで入力が 0 になるとトレースの値は指数関数的に滑らかに減少していることがわかる。一方、微分型トレースの場合は step1801 で当該入力が 0 から 1 になり、1 つ前の状態 17 に反応する入力は 1 から 0 に大きく変化するため、$|\Delta O|$ も Fig.6(c) のように大きく変化する。これより、Fig.6(b) のように step1801 でトレースの値は大きく増加している。状態の切り変わり以外のときも、状態内で連続的に変化する入力があるために $|\Delta O|$ は小さいが 0 ではない値を持つ。2 値で反応する入力は状態 18 にエージェントが訪問しているときに 1 となるので、$C_{kj,t} \cdot O_{k-1\ i,t} = C_{kj,t}$ をトレースの入力として次の状態に遷移するまでトレースの値は少しずつ増加する。状態 18 から次の状態へ移動した後は 2 値で反応する入力は 0 であるため、トレースの値は減少する。その後、100step ごとに大きく減少しているが、それは状態の切り替わりで入力が大きく変化することで Fig.6(c) のようにこのニューロンの出力が大きく変化するからである。ただし、変化量 $|\Delta O|$ は変化する入力との重み値によるため、その値はばらついている。これらのことから、新しい状態に入ると古い状態の情報は忘れていき、同一状態内で入力があまり変化しないときは、トレースの変化も小さいことが分かる。また、$C_{kj,t}$ の値は出力ニューロンとの重み値によっては負となることもあり、その場合はニューロンへの入力が正の値でもトレースの値は負となる。



(a) Actual chage of an E-trace in one hidden neuron when λ = 0.995



(b) Actual chage of a D-trace in one hidden neuron

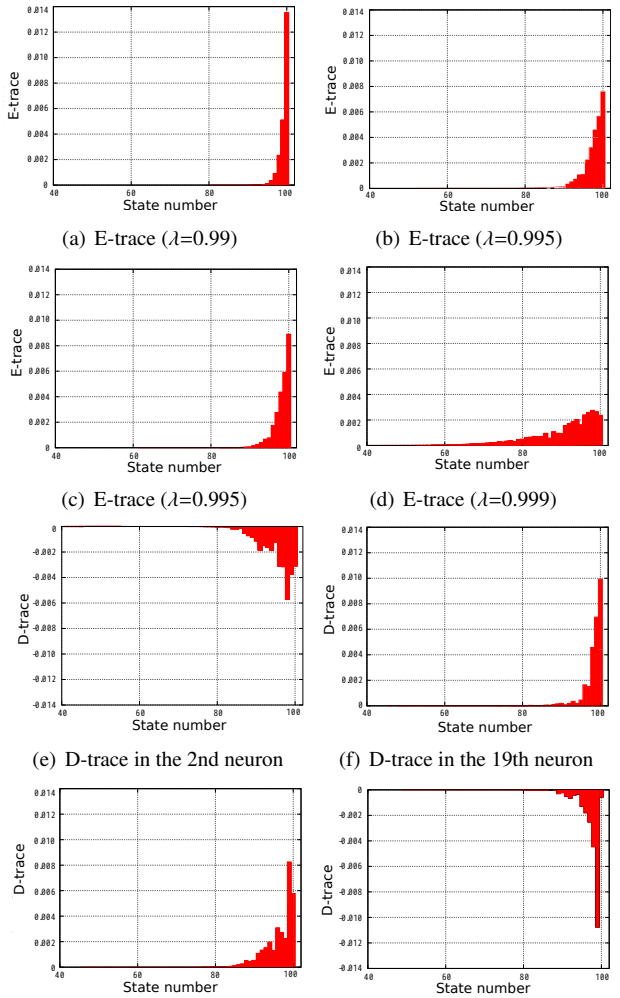


(c) change of ΔO in one hidden neuron

Fig. 6: The change of E-trace, D-trace and ΔO in the one episode.

次に、各中間層ニューロンにおいてそれぞれのトレースが過去の情報をどのように保持しているかを観察した。Fig.7 は 100 試行目のゴール到着時の各状態に局所的に 2 値で反応する 100 個の入力に対するトレースの値を,

E-trace の減衰率 λ = 0.99、λ = 0.995、λ = 0.999 の場合、微分型トレースの場合のそれぞれについてトレースの値の例を示す。λ = 0.99 の場合、ゴールに近い状態の E-trace の値は大きいが、ゴールから遠い状態に反応する入力に対しては大きく減衰している。一方、λ = 0.999 の場合、ゴールに近い状態に反応する入力に対する E-trace の値は大きくないがトレースの値はゆっくり減衰しているため、ゴールから遠い状態の入力まで記憶されていることが分かる。貢献度 C_{kj} は式 (4) から、出力ニューロンでの $dO_{N1,t}/dU_{N1,t}$、出力-中間層間の重み値 w_{Nji}、中間層ニューロンでの $dO_{kj,t}/dU_{kj,t}$ によって値が決まる。この内、出力-中間層間の重み値 w_{Nji} は試行中に大きく変化しないとしても、出力ニューロンの値 $O_{N1,t}$、中間層ニューロンの出力の値 O_{kj} が各 step で違うために E-trace のトレースの値が完全な単調減少にはならない。微分型トレースの場合は、Fig.7(e) の中間層 2 番目のニューロンがトレースの最大値が状態 98 に対するものであり、ゴールに対する状態のトレースの値がそれほど大きくないなど、Fig.7(e)、Fig.7(f)、Fig.7(g)、Fig.7(h) の各中間層ニューロンのトレースを比較すると、それぞれ傾向が異なり、ニューロンごとに保存している情報が E-trace の場合よりさらに異なることが分かる。このことから、ニューロンごとに異なる情報を保持することができ、E-trace のいずれの場合よりも効果的な学習ができたと考えられる。



(a) E-trace (λ=0.99) (b) E-trace (λ=0.995)

(c) E-trace (λ=0.995) (d) E-trace (λ=0.999)

(e) D-trace in the 2nd neuron (f) D-trace in the 19th neuron

(g) D-trace in the 25th neuron (h) D-trace in the 28th neuron

Fig. 7: The trace values for the inputs from the 100 states in one hidden neuron at the 10000th step in the 100th episode.

前述のように、ニューロンごとにトレースが保持する情報が異なるのは、初期重み値によるものと学習によるものが考えられる。ここでは、0 から学習しているために初期重み値による影響が強いと考えられるが、様々な学習を積み重ねる場合には、前の学習で獲得されたニューロン間の役割分担と、影響の小さい入力を無視する効果によって、今回の結果よりさらに大幅に学習が加速する可能性があると期待される。この効果の確認は今後の課題である。

5 結論

本論文では微分型トレースと呼ぶ価値関数の学習のための新たなトレースを用いる手法を提案した。各ニューロンごとに出力の変化が大きいときに、たくさんの情報を取り込み、出力の変化が小さいときに前の情報を保持する微分型トレースは、ニューロン自らが主観的に時間軸を調整して、各ニューロンで重要な情報を保持することが可能であり、イベント単位で効率的に過去の状態の状態価値を学習することができる。

また、一次元の連続的な環境を移動するタスクにおいて、局所的に反応する信号を入力し、状態価値を学習させて微分型トレースの特性を Eligibility-trace(E-trcae) の場合と比較した。E-trcae の場合は減衰率 λ の値によって、近い過去に重点をおいて学習するか、遠い過去まで学習するかを調節できるが、微分型トレースを用いた場合は、そのようなパラメータを調節することなく、減衰率 λ を変化させた E-trace のいずれの場合より学習性能は総合的に優れていた。また、微分型トレースでは各ニューロンごとで表現する情報が異なることを示し、そのことから E-trace の場合と比較した学習性能の優位性につながる可能性を示唆した。

実環境では情報量は膨大であり、さらにそれらが時系列データとして時々刻々と入力される。このような中で学習するためには、重要な情報を抽出し、記憶し、学習に用いることが大変重要となる。時間軸上でフラットに過去の情報を記憶するのでは非常に効率が悪いが、微分型トレースは時間軸を主観的かつ自律的に調整する画期的な方法であり、今後、実環境において力を発揮するものと期待される。

筆者らのグループではリカレントニューラルネットの学習にも、各ニューロンの時間微分値を利用して過去の情報を記憶することで、計算コスト、メモリ容量ともにニューロンの数が N 個あるとすると $O(N^2)$ で済む学習方法 [5)6)] を提案しており、今後、両者を融合させることも考えていきたい。

謝辞

この研究は日本学術振興会科学研究費補助金 (No.23500245) の援助を受けた。

参考文献

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.), InTech, 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