2D1-2

再帰型ニューラルネットワークによる時系列データの追加学習

Incremental Learning of Time Series Data by Recurrent Neural Network

浦上 大輔[*1] 太田 宏之[*2] 西田 育弘[*2]
Daisuke Uragami Hiroyuki Ohta Yasuhiro Nishida

[*1] 東京工科大学 Tokyo University of Technology
[*2] 防衛医大学校 National Defense Medical College

A dilemma of plasticity and stability exists in the incremental learning by neural networks. This dilemma originates the fact that neural networks usually employ continuously distributed representation. To solve this dilemma, it is necessary to improve one neuron's discrimination ability to input configurations. We propose the Recurrent Neural Network that consisted of the neurons whose discrimination ability is improved by the maximum calculation on the dendrites. The proposed model succeeds in the incremental learning of time series data.

1. はじめに

時系列データの追加学習とは，既に保持している時系列データを可能な限り破壊せずに，新たな時系列データを蓄えることを指す．時系列データをニューラルネットワークに学習させるためには，ニューラルネットワーク内部に状態遷移モデルが構築されなくてはならない，追加学習の実現のためには，既存の状態遷移モデルを可能な限り破壊しないように，現在の入力データを手掛かりに既存のネットワーク上のどの内部状態に対応するのかを特定し，変更を加える状態および状態遷移を限定しなければならない．

これに対して，二つのアプローチが考えられる．一つは，これから学習しようとする時系列データが過去に学習した時系列データと類似しているか否かの分類器を用意し，その分類結果に応じて変更を加える状態遷移モデルを選択する方法である(Tani, 2004) ．しかし，このアプローチでは分類器によって追加学習課題の難度が軽減されているだけであり，ニューラルネットワークによって追加学習を可能にするという課題に対して，直接的な解法になっているとは言い難い．

もう一つは，内部状態そのものの弁別可能性を上げることで，状態遷移モデルに及ぶ変化の範囲を減少させる方法である．本論文では後者の立場を取った上で，既存の追加学習モデルの問題点(Chialvo and Bak, 1984; Ohta and Gunji, 2006) を解決する．Chialvo らおよび太田らのモデルでは，Winner-Take-All (Kohonen, 2001) と Weight Conservation (Royer 2003) が組み合わせて採用されており，その疎な状態空間によって弁別可能性の向上が図られている．またその上，Chialvo らのモデルでは罰を受けた時のみ興奮性シナプス結合を depress するというルールによって，更新対象の選択性を向上させている．一方，太田らのモデルでは罰を受けた際に inhibitory transmission を強化するルールを採用することで，更新対象の選択性を向上させている．WTA の生理学的根拠としては，相互抑制が想定されているものの，しかし，実際には相互抑制によって全てのニューロン群の中で発火するものを一つに抑えることは不可能である．また，太田らのモデルに含まれる inhibition の演算は生理学的に見て冗長であり不自然である．

我々は，これらの問題を樹状突起における局所的な演算を想定することによって包括的に解決する事を試みた．具体的には，興奮性と抑制性結合の両方の入力を受けた樹状突起の枝上における MAX 演算によって，更新対象の選択性の向上を含んだ弁別可能性の向上を図った．その生理学的妥当性については本論文では詳しく吟味しないが，樹状突起の枝間における MAX 演算については、Ritter らが別の文脈で提唱している (Ritter and Urcid, 2003)．

連絡先:浦上大輔，東京工科大学コンピュータサイエンス学部，
東京都八王子市片倉町 1404−1，uragami@cs.teu.ac.jp

2. モデル

2.1 Activation Algorithm

我々の提案するモデル，Recurrent Neural Network with Dendritic Computing (RNND) の発火のアルゴリズムは，以下の3つの関数からなる(Fig. 1).

$$\boldsymbol{y}(t) = f\big(\boldsymbol{x}(t); W^{(out)}\big) \quad (1)$$

$$\boldsymbol{x}(t) = g\big(\boldsymbol{u}(t), \boldsymbol{x}^{(pre)}(t);\ W^{(in)}, W^{(pre)}\big) \quad (2)$$

$$\boldsymbol{x}^{(pre)}(t) = h\big(\boldsymbol{x}(t-1)\big) \quad (3)$$

ここで，$\boldsymbol{u}(t)$ は時刻 t における入力層ベクトル，$\boldsymbol{x}(t)$ は状態層ベクトル、$\boldsymbol{x}^{(pre)}(t)$ は前状態層ベクトル，$\boldsymbol{y}(t)$ は出力ベクトルである．$W^{(in)}$ は入力層と状態層をつなぐ結合の重み行列、$W^{(pre)}$ は入力層と状態層間の結合に対する前状態層からの抑制性結合の重み行列，$W^{(out)}$ は出力層と状態層をつなぐ結合の重み行列である．

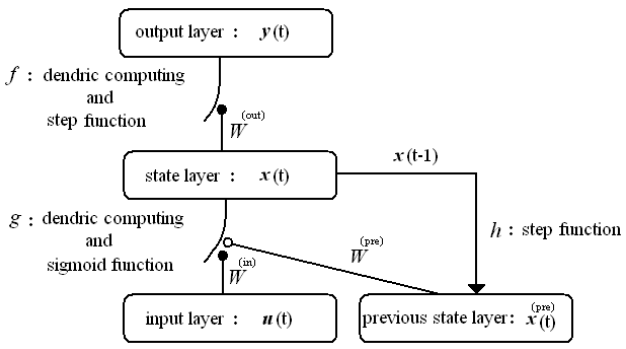

Fig. 1: The proposed architecture.

関数 *f, g, h* の詳細を説明する. 以下で、*i, j, k, p* はそれぞれ入力層, 状態層, 出力層, 前状態層のノードのインデックスである. まず, 関数 *h* の具体的な形は次の式(4)のようになる.

$$x_p^{(pre)}(t) = \begin{cases} 1, & x_p(t-1) > \theta^{(pre)} \\ 0, & x_p(t-1) \le \theta^{(pre)} \end{cases} \quad (4)$$

ここで $\theta^{(pre)}$は発火の閾値で , $\theta^{(pre)}$=0.8 とする.

関数 g は式(5)～(9)で表わされる.

$$P_{ji}(t) = \sum_p W_{jip}^{(pre)} x_p^{(pre)}(t) \quad (5)$$

$$R_{ji}(t) = W_{ji}^{(in)} u_i(t) \quad (6)$$

$$i^* = \arg max_i \left(R_{ji}(t) - P_{ji}(t) \right) \quad (7)$$

$$z_j^{(in)}(t) = R_{j i^*}(t) - P_{j i^*}(t) \quad (8)$$

$$x_j(t) = \begin{cases} \varsigma_a \left(z_j^{(in)}(t) - \theta^{(in)} \right), & z_j^{(in)}(t) > 0 \\ 0, & z_j^{(in)}(t) \le 0 \end{cases} \quad (9)$$

式(7)で$\arg max$ はインデックス i について $R_{ji}(t) - P_{ji}(t)$が最大となる i^*を抽出する操作であり, これが樹状突起上で行われる演算に相当する. 式(9)でς_aはゲインが a のシグモイド関数で, ここでは $a = 8$, $\theta^{(in)} = 0.5$ とする.

関数 f は式(10)～(12)で表わされる. ここで閾値 $\theta^{(out)} = 0.5$ とする.

$$j^* = \arg max_j \left(W_{kj}^{(out)} x_j(t) \right) \quad (10)$$

$$z_k^{(out)}(t) = W_{kj^*}^{(out)} x_{j^*}(t) \quad (11)$$

$$y_k(t) = \begin{cases} 1, & z_k^{(out)}(t) > \theta^{(out)} \\ 0, & z_k^{(out)}(t) \le \theta^{(out)} \end{cases} \quad (12)$$

以上のように, 入力層ベクトル $u(t)$ と前状態層ベクトル $x^{(pre)}(t)$ から状態層ベクトル $x(t)$ と出力層ベクトル $y(t)$ を決定する.

2.2 Learning Algorithm

入力 $u(t)$ に対する正しい出力を $y^{\mathrm{T}}(t)$ システムの出力を $y(t)$ とする. $y^{\mathrm{T}}(t)$ と $y(t)$ を比較することにより, 結合の重み $W^{(in)}$, $W^{(out)}$, $W^{(pre)}$ を増減する. 重みの増減のアルゴリズムは以下のように3つの場合に分けられる. ここで、$y_k(t)$=1 なる k を k^* とし, j^* は式 (10) によって k^* に対応するものが決定される. 同様に, i^* は式 (7) によって j^* に対応するものが決定され, p^* は $x^{(pre)}{}_{p^*}(t)$=1 なる p とする. また, $r^{(out)}$, $r^{(in)}$, $s^{(out)}$, $s^{(in)}$, $q^{(out)}$, $q^{(in)}$ は正定数である.

(L1) $y_{k^*}^{\mathrm{T}}(t) = 1$ の場合 (not miss):

$$W_{k^*j^*}^{(out)}(\mathrm{t+1}) = W_{k^*j^*}^{(out)}(t) + r^{(out)} * (1 + z_{k^*}^{(out)}(t)) \quad (13)$$

$$W_{j^*i^*}^{(in)}(\mathrm{t+1}) = W_{j^*i^*}^{(in)}(t) + r^{(in)} * (1 + z_{j^*}^{(in)}(t)) \quad (14)$$

(L2) $y_{k^*}^{\mathrm{T}}(t) = 0$ の場合 (miss) :

$$W_{k^*j^*}^{(out)}(\mathrm{t+1}) = W_{k^*j^*}^{(out)}(t) - s^{(out)} \quad (15)$$

$$W_{j^*i^*}^{(in)}(\mathrm{t+1}) = W_{j^*i^*}^{(in)}(t) - s^{(in)} \quad (16)$$

(L3) 全ての k について$y_k(t) = 0$ の場合(no output node fire) : $y_k^{\mathrm{T}}(t) = 1$ なる k を新たに k^* とし、その k^* に対応する j^* と i^* について,

$$W_{k^*j^*}^{(out)}(\mathrm{t+1}) = W_{k^*j^*}^{(out)}(t) + q^{(out)} * (1 + z_{k^*}^{(out)}(t)) \quad (17)$$

$$W_{j^*i^*}^{(in)}(\mathrm{t+1}) = W_{j^*i^*}^{(in)}(t) + q^{(in)} * (1 + z_{j^*}^{(in)}(t)) \quad (18)$$

ただし, 1つ前の時刻でシステムの出力が全て正しかった場合, (L2) と (L3) は実行されず, 代わりに下記の$(L2)^{\mathrm{p}}$ と $(L3)^{\mathrm{p}}$ が実行され, 前状態からの抑制性の結合が強化される.

$(L2)^{\mathrm{p}}$ $y_{k^*}^{\mathrm{T}}(t) = 0$ かつ全ての k について $y_k(t-1) = y_k^{\mathrm{T}}(t-1)$ の場合:

$$W_{j^*i^*p^*}^{(pre)}(\mathrm{t+1}) = W_{j^*i^*p^*}^{(pre)}(t) + s^{(pre)} \quad (19)$$

$(L3)^{\mathrm{p}}$ 全ての k について $y_k(t) = 0$ かつ全ての k について $y_k(t-1) = y_k^{\mathrm{T}}(t-1)$ の場合: $y_k^{\mathrm{T}}(t) = 1$ なる k を新たに k^* とし, その k^* に対応する j^* と i^* , $p \neq p^*$ なるすべての p について,

$$W_{j^*i^*p}^{(pre)}(\mathrm{t+1}) = W_{j^*i^*p}^{(pre)}(t) + q^{(pre)} \quad (20)$$

結合の重みの増減は次の式 (21) と式 (22) で表わされるように、重みの総量が保存されるように行われる。

$$\sum_k W_{kj}^{(out)}(t+1) = \sum_k W_{kj}^{(out)}(t) = \alpha^{(\mathrm{out})} \quad (21)$$

$$\sum_i \mathrm{W}_{ji}^{(in)}(\mathrm{t+1}) = \sum_i \mathrm{W}_{ji}^{(in)}(\mathrm{t}) = \alpha^{(in)} \quad (22)$$

したがって, ある結合の重みが増強されると関連する重みの結合は減少することになる. また, 結合の重みはそれぞれ $0 \le W_{kj}^{(out)} \le 1$, $0 \le W_{ji}^{(in)} \le 1$, $0 \le W_{jip}^{(pre)} \le 1$ を満たすようにする.

3. シミュレーションと結果

RNND が時系列データを追加学習可能であることシミュレーションで確認する. 学習用の入出力シーケンスは Fig.2 の3種類を用いた. 入力層と出力層とノードの個数は16, 中間層のノードの個数は32 とする.

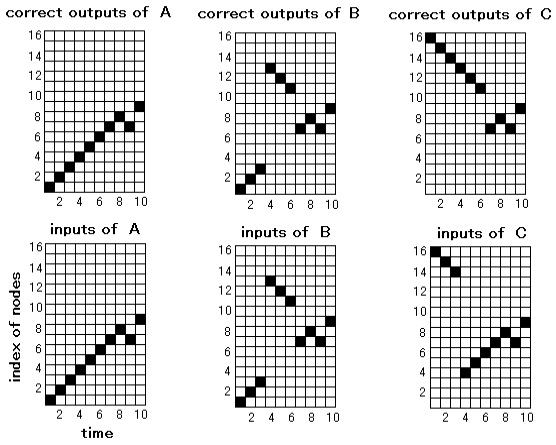

Fig. 2. The learning set of time series.

学習のレートはそれぞれ $r^{(out)} = 0.2$ ， $r^{(in)} = 0.2$ ， $q^{(out)} = 0.1$ ， $q^{(in)} = 0.1$ ， $s^{(out)} = 0.01$ ， $s^{(in)} = 0.01$ ， $s^{(pre)} = 0.2$ ， $q^{(pre)} = 0.2$ に設定した. 初期条件については, $W_{kj}^{(out)}$ と $W_{ji}^{(in)}$ はそれぞれ式(21) と式(22) を満たす限りでランダムに設定した. ただし, 重みの総量を表すα については, $\alpha = 1$ とした. また, $W_{jip}^{(pre)} = 0$, $x_p^{(pre)} = 0$ とした.

3.1 Experiment 1

ここでは Fig.2 のパターンAとパターンBを学習用の入出力シーケンスとして用いる. AとBを比較すると, 3番目と4番目と5番目の入出力パターンが異なり, その他の入出力パターンが同一である. まず, Aを繰り返し 50 回(500 タイムステップ)学習し, その後にBを同様に繰り返し 50 回学習する. したがって, 学習は計 1000 タイムステップ行われる. Aを学習しているときにBを参照したり, Bを学習しているときにAを参照したりすることはない.

Fig.3 は学習曲線である. t =150 あたりでAの学習が完了し, t = 500 でBの学習が始まり, t =550 あたりでBの学習が完了していることがわかる.

Fig.4 は学習終了後に学習結果を確認するために, AとBを1回ずつシステムに入力した結果である. AとB共に正しく出力されていることがわかる. つまり, RNND はこのタイプの時系列を追加学習できることが確認できる. また Fig.4 の状態 x の発火パターンをみると, 異なる入力には異なる状態ノードが発火し, 状態ノードが重複して複数の入出力対を記憶することがないようになっていることがわかる. 特に、t=1014, 1015, 1016 において発火している状態ノードが, つまり追加された記憶を保持しているノードが, 他の時刻で発火しているノードと重複していないことに注意せよ.

Fig.5 を見れば、どのような仕組みで状態ノードの重複を回避しているかがわかる. 最初にAを学習する段階で, 初期状態で偶然に他より大きかった $W_{10,4}$ の結合が増強されると同時に, 重みの保存則により他の全ての $W_{10,i}$ が小さくなってくことが左上のグラフからわかる. その結果, 10番目の状態ノードは4番目の入力ノードの発火にのみに対応して発火することになる. 一方, 左下のグラフをみると, 2番目の状態ノードは, Aの学習には関与していないことがわかる. したがって4番目の入力ノードが発火した場合以外では, 相対的に2番目の状態ノードは10番目の状態ノードより励起することになる. だから, 追加のBの学習で2番目の状態ノードが10番目の状態ノードより使われやすくなる. Fig.5 の右の図は, Aの学習時に, 1つの状態ノードを2つの入力ノードがとりあうケースである. 式(14) および式(18) にあるように, 学習のレートに $(1 + z_{j^*}^{(in)}(t))$ の補正項が入っているので、最終的には1つの入力ノードが独占することになり, もう1つの入力ノードは別の状態ノードが割り当てられることになる. 以上をまとめると, どの経路が最も励起しているかを判別する樹上突起上の演算と, 重みの保存則により, 追加学習が可能になっていると言える. このケースでは, 前状態からの抑制は学習にほとんど関与しないが, 次節のケースでそれは重要な役割を担う.

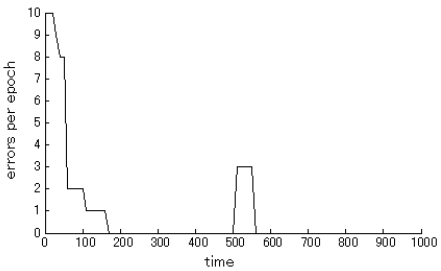

Fig. 3. Experiment 1. Learning curves.

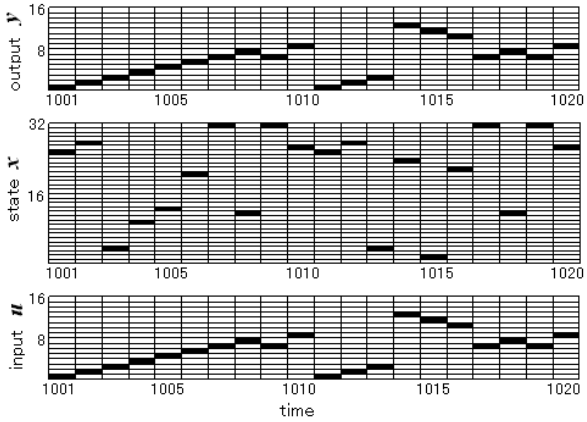

Fig. 4. Experiment 1. Results of the learning.

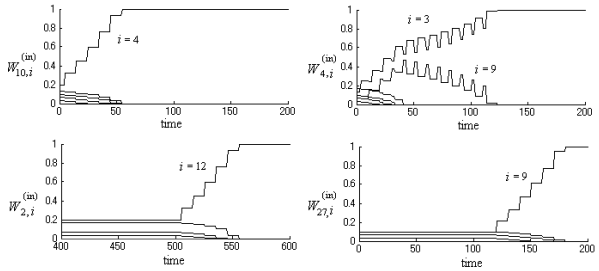

Fig. 5. Experiment 1. The evolution of the weight.

3.2 Experiment 2

ここでは Fig.2 のパターンAを学習した後に, パターンCを学習する. その他の条件は Experiment 1 と同じである. AとCを比較すると, 1～3番目では入出力が共に異なり, 4～7番目では入力が同じにもかかわらず出力が異なり, 8～10番目では入出力が共に同じである.

Fig.6 は学習曲線である. t =150 あたりでAの学習が完了し, t = 500 でCの学習が始まり, t =750 あたりでCの学習が完了していることがわかる.

Fig.7 は学習終了後に学習結果を確認するために, AとCを1回ずつシステムに入力した結果である. 出力 y をみると, AとC共に正しく出力されていることがわかる. つまり, RNND はこのタイプの時系列を追加学習できることが確認できる. 状態 x のグレースケールはシグモイド関数による発火の程度を表している. t=1014, 1015, 1016 において, 完全に発火しているノード(黒)の他にもう一つ, 中程度発火しているノード(灰色)がある. このノードは入力が同じである t=1004, 1005, 1006 において発火しているノードと同じである. これらのノードは前状態層からの抑制によって, 出力層の励起に寄与しない程度まで, 発火のレベルが抑えられている.

Fig.8 は, t=1004 を例にしてその様子を表している. 入力層で発火しいるノードのインデックス i=4 について, 状態層のインデックスj=7 とj＝27 の2つのノードにおいてポテンシャルRが高まっている. しかし, j＝27 のノードについては抑制のポテンシャルPも高まっているため, j=27 のノードの発火のレベルは小さくなる.

Fig.9 はこれらの結合が形成される学習過程を表している。まず, 入力層のノード(i=4)と状態層のノード(j=27)を繋ぐ結合が形成される(Fig.9-左上). 追加学習が始まると(t=500 以後), 前状態のノード(p=16)からの抑制性の結合が生成される(Fig.9-右上). その後(t=600 あたり), 状態層の別のノード(i=7)と入力層のノード(i=4)を繋ぐ結合が成長し(Fig.9-左下), それと同時

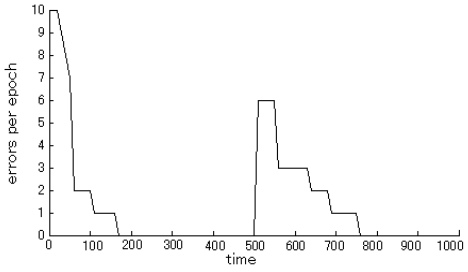

Fig. 6. Experiment 2. Learning curves.

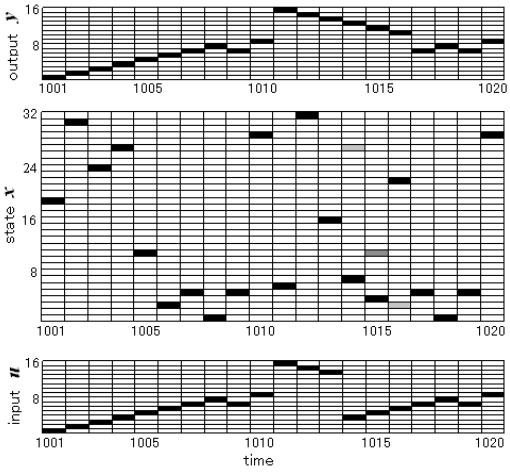

Fig. 7. Experiment 2. Results of learnig.

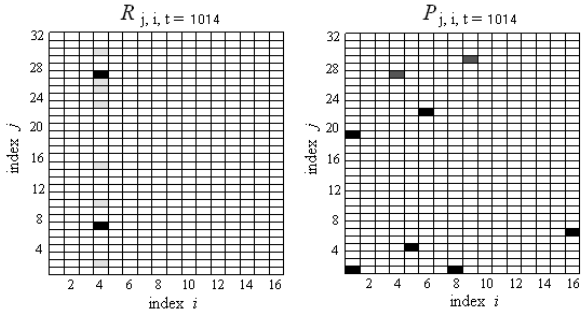

Fig. 8. Experiment 2. $P_{ji}(t)$ and $R_{ji}(t)$ of eq.(5) and eq.(6) at t=114.

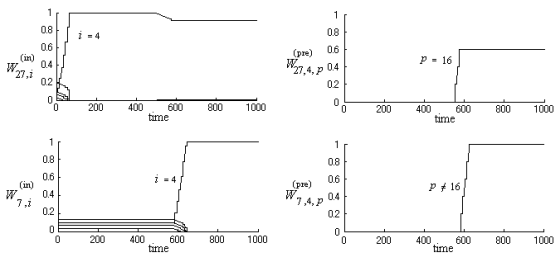

Fig. 9. Experiment 2. The evoluation of the weight

に前状態層の p=16 以外のノードからの抑制性の結合が成長する(Fig.9-右下).

4. 考察

これまで神経系の並列性に関する議論は, 分散表現という言葉によって表されるような, activation rule に従って生成された activated pattern に関するものが中心であった(Rumelhart, 1986) . しかし, 追加学習の実現においては, activation rule ではなくて learning rule(plasticity)の視点から, 並列性を考察しなくてはならない.

ここで複数のニューロン群によって生成される activated pattern から目を離し, ひとつのニューロンからみた可能な入力パターンに目を向けてみよう. あるひとつのニューロンは樹状突起において複数の興奮性・抑制性のシナプス結合を受ける. それらのシナプス結合を通じて EPSP, IPSP が発生し, 樹状突起や soma での演算処理を経て, 発火するか否かが決定され, 出力が行われる. そのニューロンから見て, そのニューロンを発火させる可能性のある EPSP, IPSP の組み合わせパターン群が規定できるが, そのパターン群を intrinsic distributed representation と名づけよう.

追加学習の実現には, この intrinsic distributed representation のサイズ(多様性)が重要な役割を果たす. すなわち、intrinsic distributed representation の多様性が小さければ, 更新対象となる intrinsic distributed representation の範囲が大きくなり, 結果として, そのニューロンの発火を決定するシナプス結合が大きく変化してしまう. 逆に intrinsic distributed representation の多様性が大きければ, 更新対象となる intrinsic distributed representation を小さくなり, 結果として, 更新されるシナプス結合が少なくなる.

我々のモデルでは, 樹状突起における MAX 演算を想定することで, intrinsic distributed representation の多様性を向上させ, 結果として時系列データの追加学習を実現した. 今後, 可塑性から見たニューラルネットワークの並列性を議論していくためには, この intrinsic distributed representation の多様性の定量化に関する検討が必要である.

参考文献

Chialvo, D., Bak, P.: Learning from mistakes, Neuroscience 90, pp.1137-1148, 1999.

Kohonen, T.: Self-organizing maps, New York: Springer, 2001.

Ohta, H., Gunji, Y.-P.: Recurrent neural network architecture with pre-synaptic inhibition for incremental learning, Neural Networks 19, pp.1106-1119, 2006.

Ritter, G. X., Urcid G.: Lattice algebra approach to single neuron computation, IEEE Trans on Neural Networks 14, No.2, pp.282-295, 2010.

Royer, S., Pare, D.: Conservation of total synaptic weight through balanced synaptic depression and potentiation. Nature 422, pp.518-522, 2003.

Rumelhart, D. E., Hinton, G. E., Williams, R. J.: Learning internal representations by error propagation, Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations, Cambridge, MA, USA: MIT Press, pp.318-362, 1986.

Tani, J., Ito, M., Sugita, Y.: Self-organization of distributedly represented multiple behavior schemata in a mirror system: reviews of robot experiments using RNNPB, Neural Networks 17, pp.1273-1289, 2004.