再帰型束ニューラルネットワークによる追加学習：樹状突起の計算能力について

Recurrent lattice neural network for incremental learning: On dendritic computing

浦上大輔†，大田宏之‡
Daisuke Uragami, Hiroyuki Ohta

†学習院大学，‡防衛医科大学校
Gakushuin University, National Defense Medical College
dduragami@gmail.com

Abstract

This article presents recurrent lattice neural network for incremental learning and we discuss a relation between partial ordered sets and the origin of distributed representation.

Keywords — Neural Network, Incremental Learning, Distributed representation, Lattice

1. 追加学習

追加学習では，可塑性と安定性のジレンマを克服することが求められる．すなわち，既に学習したデータを保持しつつ，追加されたデータを学習することが求められる．ニューラルネットワークは一般的に状態空間の連続的分散表現を採用しているため，追加されたデータの学習は既存の学習結果に影響を与える．Ohta らは，Winner – Take - All メソッドを採用することによって連続的分散表現を弱めることを提案している[1]．さらに，Negative Reinforcement のみで学習することによって，既存の学習結果が追加の学習内容によって上書きされないように，つまり促した．その結果，再帰型ニューラルネットワークにおいて時系列データの追加学習に成功している．一方，本研究では，追加された学習データに対しては新たな樹状突起を生成することによって，追加学習を実現する方法を提案する．

2. 束ニューラルネットワーク

神経細胞の樹状突起における情報処理の数理モデルとして、束代数に基づくニューラルネットワーク（lattice neural network, LNN）が提案されている[2]．脳の容積の大部分を占める樹状突起は高い計算能力を有すると推測されているが，それをモデル化した LNN は，単一のニューロンで XOR 問題を解けるなど, その計算能力は興味深い.

LNN のアーキテクチャを図 1 に示す. ホスト側のニューロンから D_1 ～D_K の複数の樹状突起が伸びていて，それぞれの樹状突起上で演算が行われるのが特徴である．具体的な演算の定義は次式で表現される．

$$\tau(\boldsymbol{x}) = \bigvee_{k=1}^{K} p_k \bigwedge_{i\in I} \bigwedge_{l\in L} r_{ik}^{l}(x_i + w_{ik}^{l}) \tag{1}$$

$$y(\boldsymbol{x}) = f\big(\tau(\boldsymbol{x})\big) \tag{2}$$

$$f(\tau) = \begin{cases} 0, & if\ \tau < 0 \\ 1, & if\ \tau \geq 0 \end{cases} \tag{3}$$

ここで，$\boldsymbol{x} = (x_1, \ldots, x_i, \ldots, x_n)$は入力，$w_{ik}^{l}$は結合の重みで$x_i, w_{ik}^{l} \in \mathbf{R} \cup \{-\infty, +\infty\}$．$p_k, r_{ik}^{l} = 1\ or\ -1$ はそれぞれ興奮性結合と抑制性結合を意味し，l=1 が興奮性，l=0 が抑制性の添え字である．yは出力である．∨{_}と∧{_}はそれぞれ束代数における演算で上限と下限であるが，ここでは全順序集合を扱っているので，それぞれ最大値と最小値に等しくなる．

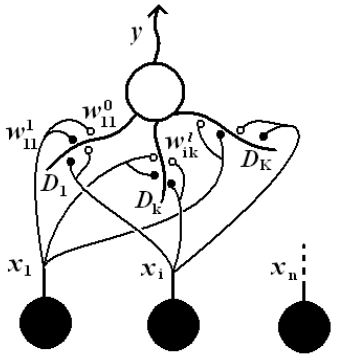

図 1　LNN のアーキテクチャ

LNN における学習はw_{ik}^{l}を調整することにより行われる．例えば図 2 のように，$\boldsymbol{x}=\left(x_1^{\xi},x_2^{\xi}\right)$, $y=1$ という入出力が学習データとして与えられたとしよう．このときw_{ik}^{l}を，$w_{1k}^{1}=-\left(x_1^{\xi}-\alpha\right), w_{1k}^{0}=-\left(x_1^{\xi}+\alpha\right), w_{2k}^{1}=-\left(x_2^{\xi}-\alpha\right), w_{2k}^{0}=-(x_2^{\xi}+\alpha)$のように調整すればよい．ここで$\alpha$は LNN の汎化能力に関係するパラメータで，一般的には各ニューロンや樹上突起で共通である必要はなく，このパラメータをどう調整するかが LNN の中心課題になる．

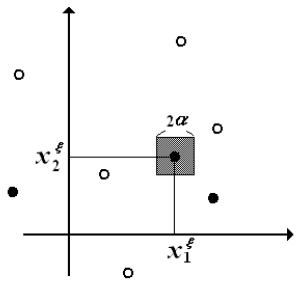

図 2 学習データの例

3. LNN による時系列の追加学習

先述の式(1)を次式のように時系列データが学習可能な Recurrent　LNN へと拡張した．

$$\tau\left(x(t),x(t-1)\right)=$$
$$\mathrm{V}_{k=1}^{K}\left\{\left(p_k\wedge_{i\in I}\wedge_{l\in L}r_{ik}^{l}\left(x_i\left(t\right)+w_{ik}^{l}\right)\right)\right.$$
$$\left.\mathrm{V}\left(p_k^{*}\wedge_{i\in I}\wedge_{l\in L}r_{ik}^{*l}\left(x_i\left(t-1\right)+w_{ik}^{*l}\right)\right)\right\} \qquad (4)$$

Recurrent　LNN は，汎化能力に関係するパラメータαを十分に小さくすれば，Ohta らモデルと同様の追加学習課題をこなすことができる．

Ohta らのモデルでは，1. Winner – Take – All メソッドで分散表現を弱めることにより，既存の学習結果と追加の学習内容の共存を可能にする．2. Negative Reinforcement のみで学習することによって，既存の学習結果が追加の学習内容によって上書きされないようする．これに対応して LNN では，

1. V{_}の演算，つまり最大値をとる演算が Winner – Take – All に相応する。
2. 追加の学習内容に対しては新たな樹状突起を生成することにより，既存の学習結果の上書きを防ぐ．

という仕組みで追加学習を実現している．

4. 分散表現と束

ここまでの議論では，入力x_iや結合の重みw_{ik}^{l}の値域は実数，すなわち全順序集合であった．例えば図 3 の左のように，全順序集合では，$\mathrm{V}\{a,b\}$の演算結果は$\{a,\ b\}$の最大値＝a となる．つまり，$\mathrm{V}\{\mathrm{a},\mathrm{b}\}$の演算結果に b の値は影響を与えないことなり，このことが a（という記憶領域）と b（という記憶領域）の独立性を担保している．

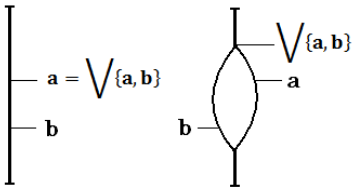

図 3　全順序集合と束

一方，図 3 の右ように順序がつけられない領域が存在する（半順序集合，束）場合，$\mathrm{V}\{\mathrm{a},\mathrm{b}\}$の演算結果は a の値と b の値の両方に依存することになる．これが分散表現の起源である．

Uragami らは全順序集合とより一般的な束が動的に入れ替わる仕組みを考案し，それをセルオートマトンに実装している[3]．同様のアーキテクチャを LNN に適用することは容易く，興味深い結果をもたらすであろう．

参考文献

[1] Ohta, H., Gunji, Y.-P., (2006) “Recurrent neural network architecture with pre-synaptic inhibition for incremental learning”, Neural Networks 19, pp.1106-1119.

[2] Ritter G.X., Urcid G, (2003) “Lattice algebra approach to single neuron computation”, IEEE Trans on Neural Networks 14, No.2, pp.282-295.

[3] Uragami, D., Gunji, Y.-P., (2008) “Lattice-driven cellular automata implementing local semantics”, Physica D 237, pp.187-197.