大脳新皮質学習における適応型シナプス配置法の検討

○青木 健　鈴ヶ嶺 聡哲　髙玉 圭樹　佐藤 寛之 （電気通信大学）

A Study of Self-adaptive Synapse Arrangement on Cortical Learning Algorithm

∗T. Aoki, S. Suzugamine, K. Takadama, and H. Sato
(The University of Electro-Communications)

Abstract– Cortical learning algorithm (CLA) is a time-series data prediction algorithm designed based on the human brain neocortex. CLA has a number of columns which are connected by synapses with input data, converts an input data to an internal column representation based on connection relation of synapses. Since the conventional CLA fixes the synapse relations between columns and input data, CLA cannot arrange the synapses according to input bias. Consequently, columns not used for the internal representation arise, and it causes the deterioration of prediction accuracy of CLA. To improve the prediction accuracy of CLA, in this work, we propose an improved CLA self-adaptively arranging synapses and verify its effectiveness on several time-series data which drastically change data tendency.

Key Words: Hierarchical temporal memory, Cortical learning algorithm, Time-series data prediction, Self-adaptive

1　はじめに

時系列データを予測する階層時間記憶 (Hierarchical Temporal Memory，以下 HTM) [1] という概念がある．HTM を具現化する方法の一つとして，大脳新皮質学習アルゴリズム (Cortical Learning Algorithm，以下 CLA) が提案されている [2, 3]．CLA は，人間の大脳新皮質の仕組みを模倣した学習アルゴリズムである．関連研究として，人工ニューラルネットワークの一種であるリカレントニューラルネットワークがあり，長短期記憶 (Long Short-Term Memory，以下 LSTM) [4] による時系列予測の有用性が知られている．CLA は大脳新皮質をモデル化するため，ニューロンを面状ではなく柱状に並べる点や，各ニューロンが複数の樹状突起を持つ点などが LSTM とは異なる．CLA は，タクシー乗車数の時系列予測において LSTM より高い予測精度を達成した報告 [5] があり，有望な時系列予測法の一つである．CLA は，予測器に複数のカラムを用いる．各カラムは入力データビット列との間にシナプスを持つ．CLA は，シナプスの接続関係にもとづいて，入力データをカラムの活性パターンで予測器の内部表現にする．従来の CLA は，アルゴリズムの複雑さ，パラメータの多さ，実行ごとの動作の不安定さに難点がある．これらを改善するために，我々はこれまでに簡素型 CLA[6] を提案し，その有効性を検証してきた．しかし，従来の CLA と簡素型 CLA は，入力データの偏りによってカラム群の活性頻度に偏りが生まれる．その結果，入力データの内部表現に活用されないカラムが生じ，予測精度が悪化する問題がある．これは，カラムと入力データビット列を繋ぐシナプス配置が固定されるため，入力の偏りに合わせてシナプスを配置できないことが原因である．

上記の問題に対して，我々は，カラムを有効活用して入力データをより精緻に内部表現化することで，時系列予測性能を高める目的から，カラムと入力データビットを繋ぐシナプスを，入力データに合わせて動的に再配置する方法を提案した [8]．先の報告 [8] では，時系列パターンが一定のテスト入力を予測する実験において，従来の CLA と LSTM，簡素型 CLA，簡素型 CLA に提案法を組み込んだ CLA の時系列予測性能を比較し，提案法の効果を明らかにした．本稿では，時系列パターンが途中で変化するテスト入力を予測する実験において，入力の変化に合わせて，提案法がシナプスの配置を変化でき，時系列予測性能を高める効果を持つことを確認する．

2　大脳新皮質学習アルゴリズム (CLA)[2]

2.1　概要

CLA は，人間の大脳新皮質の仕組みを模倣したアルゴリズムである．本稿では，NuPIC [3] による CLA を従来法とする．概念図を Fig. 1 に示す．CLA には，ニューロンをモデル化したセル，複数のセルを連ねたカラム，複数のカラムを並べたリージョンが存在する．セルの状態には，活性，予測，通常の 3 種類がある．また，カラムの状態には，活性と通常の 2 種類がある．各カラムは，シナプスの束であるセグメントを持ち，その各シナプスは，入力データの各ビットとの関係をつくる．また，各セルもセグメントを持ち，その各シナプスは，他のセルとの関係をつくる．カラムの各シナプスは，永続値 p_c を持つ．学習中に p_c を変化させ，p_c が閾値 T_c 以上の場合にシナプスを接続状態にし，それ

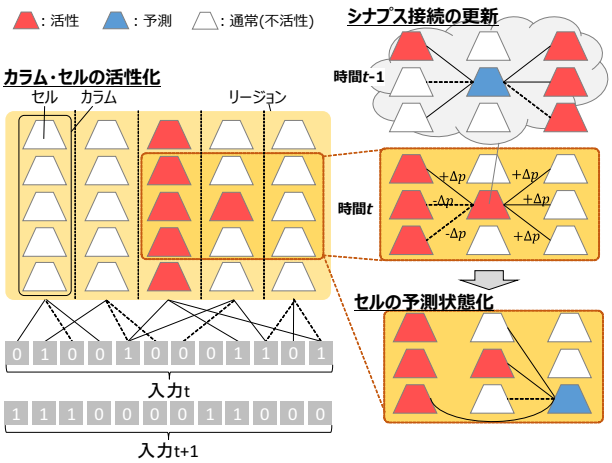


Fig. 1: CLA の概念図

以外の場合には切断状態にする．同様に，セルの各シナプスも永続値を持つ．

CLA は，入力データをビット列に変換し，空間プーリングによって活性状態となるカラムを決定する．その後，時間プーリングによってカラムの状態にもとづいて活性状態となるセルを決定する．さらに予測状態にするセルを決定し，その予測状態のセル群から予測値を算出する．以下，CLA の詳細について述べる．

2.2 初期化：カラムのシナプス生成

カラムと入力データビットを関係付けるシナプスを生成する．Fig. 2 **(a)** に例を示す．各カラムについて，まず，生成するシナプスの入力データビット列における中心位置を決定する．カラム数を N_c，入力データビット長を N_b とすると，i 番目のカラムが生成するシナプスの入力データビット列における中心位置 C_i は，次式で算出する．

$$C_i = \left\lfloor 1 + i\frac{N_b - 1}{N_c} \right\rfloor \quad (i = 1, 2, \ldots, N_c) \tag{1}$$

次に，中心位置 C_i の半径 R 内において密度 d でシナプスを生成する．生成されるシナプス数 N_{cs} は次式で与えられる．

$$N_{cs} = \lfloor d(2R + 1) \rfloor \tag{2}$$

次に，各シナプスの永続値 p_c を初期化する．初期シナプスの接続率をパラメータ CR とし，接続するシナプスの永続値に $[T_c, 1]$ の範囲でランダムな値を設定し，切断するシナプスの永続値に $[0, T_c)$ の範囲でランダムな値を設定する．

2.3 入力データ変換

Fig. 1 の左下部に示すように入力データを 0/1 のビット列に変換して，リージョンに入力する．本稿では，実数値の入力データを想定する．入力データごとに，全ビット長 N_b のうち，連続する w ビットが 1，それ以外を 0 のビット列にする変換処理を施す．入力データの最大値が max，最小値が min，入力値が $data$ $(= [min, max])$ の場合，入力データビット列中で 1 となる範囲 $[start, end]$ を次式で決定する．

$$start = \left\lfloor 1 + \frac{(data - min) \cdot (N_b - w)}{max - min} \right\rfloor \tag{3}$$

$$end = start + w - 1 \tag{4}$$

2.4 空間プーリング

Fig. 1 の左部において，0/1 のビット列に変換された入力データを受け取り，活性状態にするカラムを決定する処理を空間プーリングという．その手順について以下に述べる．

カラムの活性化

まず，1 の入力ビットと接続状態にあるシナプスを有するカラムを活性候補にする．次に，活性候補のカラム群から，活性状態にするカラム群を選択する．具体的には，各カラムにおける 1 の入力ビットと接続状態にあるシナプス数 $Overlap$ と，前回活性状態になってからの経過時間が長いほど大きくなるブースト値 $boost$ を乗算したスコア値 $Score = Overlap \cdot boost$ にもとづいて活性候補カラム群をランキングする．その後，その上位から上限数 AC^{max} までを活性状態にする．そして，活性状態のカラム群において，セグメント内のシナプスの永続値を更新する．具体的には，1 の入力ビットと対応するシナプスの永続値を増加させ，0 の入力ビットと対応するシナプスの永続値を減少させる．

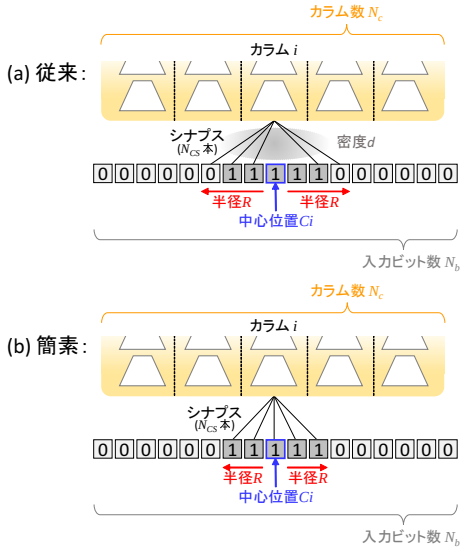


Fig. 2: シナプスの配置

バンプアップとブースト

バンプアップは，活性候補になる頻度が基準値を下回るカラムに対して，その全シナプスの永続値を Δp_b^+ 増加させる処理である．全カラム中の最大の活性候補頻度 ACF^{max} に対して，基準値 R^{ACF} は次式で算出する．

$$R^{ACF} = K \cdot ACF^{max} \tag{5}$$

ここで，K は係数パラメータである．

ブーストは，活性頻度が基準値を下回るカラムに対して，ブースト値 $boost$ を増加させることで $Score$ 値を高めて活性化を促進させる方法である．全カラム中の最大の活性頻度 AF^{max} に対して，基準値 R^{AF} は次式で算出する．

$$R^{AF} = K \cdot AF^{max} \tag{6}$$

ここで，K は係数パラメータである．i 番目のカラムの活性頻度を AF_i，最大ブースト値を $boost^{max}$ とすると，ブースト値 $boost_i$ は次式で算出する．

$$boost = \begin{cases} 1.0, & \text{if } AF > R^{AF} \\ \frac{AF(1 - boost^{max})}{R^{AF}} + boost^{max}, & \text{otherwise} \end{cases} \tag{7}$$

2.5 時間プーリング

Fig. 1 の右部において，活性状態のカラムに属するセルの中で，活性状態にするセルを決定し，さらに，次の入力を予測するために予測状態へ遷移させるセルを決定する処理を時間プーリングという．その手順について以下に述べる．

まず，活性状態の各カラムにおいて，活性状態にするセルを決定する．具体的には，活性状態のカラムの中に，$t - 1$ の入力データによって予測状態になったセ

ルが存在する場合，そのセルを活性状態に遷移させる．また，活性状態のカラムの中に予測状態のセルが存在しない場合は，そのカラム内の全てのセルを活性状態にする．次に，予測状態にするセルを決定する．具体的には，全てのセルの各セグメントにおいて，活性状態のセルと接続されているシナプスの数が Ac 以上存在する場合，そのセグメントが属するセルを予測状態にする．最後に，予測状態から活性状態となったセルに対して，そのセルを予測状態にしたセグメント内のシナプスの永続値を更新する．更新方法としては，予測状態となったときに活性状態のセルと接続されていたシナプスの永続値を増加させ，通常状態のセルと接続されていたシナプスの永続値を減少させる．

2.6 出力変換

時間プーリングによって得られた予測状態のセル群から予測値を算出する．各セルが予測状態になったときの予測値の情報をもとに，予測状態のセルの間で確率的に予測値 $predict(t)$ を求める．

3 簡素型 CLA[6)]

従来の CLA は，アルゴリズムの複雑さ，パラメータの多さ，実行ごとの動作の不安定さに難点がある．これを改善するため，我々はこれまでに簡素型 CLA を提案した [6)]．簡素型 CLA は，以下 (i)-(v) を導入する．

(i) シナプス生成密度の排除

従来法におけるシナプスの生成密度 d の概念を排除し，生成半径 R 内の全てにシナプスを生成する．Fig. 2 **(b)** に例を示す．各カラムのシナプス数を N_{cs} とし，シナプスの生成半径 R は次式で算出する．

$$R = \left\lfloor \frac{N_{cs} - 1}{2} \right\rfloor \tag{8}$$

これにより，シナプス配置のランダム性が排除される．また，従来のシナプス配置には，生成密度 d と半径 R の設定が必要だが，提案法は，シナプス数 N_{cs} か生成半径 R のいずれかを設定すれば良い．

(ii) シナプスの初期永続値の固定化

シナプスの初期の永続値をランダム値から固定値 P にする．これにより，シナプス接続のランダム性を排除できるため，特に初期のシナプス接続の不安定さが緩和され，予測を安定化できる．初期の永続値パラメータ P の導入と引き換えに，従来法における初期のシナプス接続率 CR を排除できるため，パラメータ数は変化しない．

(iii) バンプアップ無効化

バンプアップを無効化する．これにより，アルゴリズムが簡素化され，バンプアップに関するパラメータも排除される．

(iv) 加算式ブースト

従来の乗算式のスコア値の算出法に対して，次式で更新されるブースト値 $boost$ を用いて，加算式でスコア値 $Score = Overlap + boost$ を算出する．

$$boost = \quad 0.99 \cdot (1 - AF) \tag{9}$$

これにより，予測を妨げるシナプスの再接続を防止できる．また，$Overlap$ の少ないカラムが $Overlap$ の多いカラムより優先して活性化することがなくなり，予測精度の向上とカラムの活性パターンの安定化を期待できる．

(v) シナプス中心位置の正規化

各カラムのシナプス生成時の中心位置を正規化する．従来の CLA では，入力データビット列の両端にシナプスを持つカラムが生じる．これを避けるため，提案法では，シナプスの中心位置を入力データビット列の内側にシナプスの生成半径 R 分寄せて配置する．カラム数を N_c，入力ビット長を N_b，シナプスの生成半径を R とする場合，i 番目のカラムが生成するシナプスの中心位置 C_i を次式で決定する．

$$C_i = \left\lfloor 1 + i\frac{N_b - 2R - 1}{N_c} + R \right\rfloor \quad (i = 1, 2, \ldots, N_c) \tag{10}$$

これにより，入力データビット列の両端をまたいだシナプス生成を回避でき，予測精度の向上を期待できる．

4 問題点：固定化したシナプス配置

従来の CLA[2)] と簡素型 CLA[6)] では，カラムが持つ入力データビットとのシナプスの位置が固定される．入力データビット列における 1 の出現率に偏りがある場合，リージョン内に活性状態にならないカラムが生じる．すなわち，入力データをリージョン内の内部表現にするために使用されない領域が生じる．入力データに対して適応的にシナプスを再配置することで，使用されなかったカラムを活用することができれば，入力データを精緻にリージョンの内部表現に変換でき，予測精度が高まると考えられる．

5 提案：シナプスの動的配置法

リージョン内に存在するカラムを有効活用して，入力データをより精緻に内部表現化することで時系列予測性能を高めるため，本稿では，カラムと入力データビットを繋ぐシナプスを，入力データに合わせて動的に再配置する方法を提案する．提案法は，簡素型 CLA をベースにする．従来の CLA は，動作の不安定さによって再配置するシナプスの決定が困難になるため，今回の提案法にとって最適でない．

5.1 考え方

提案法の考え方を Fig. 3 を用いて述べる．まず，活性頻度 A_l 以下の緑色で示すカラム群を移動元候補，活性頻度 A_h より高い赤色で示すカラム群を移動先候補にする．次に，赤色で示す移動先候補のカラム群の活性頻度の比に合わせて，緑色で示す移動元候補のカラム群が持つシナプスを，赤色で示す移動先候補のカラム群と同様のシナプス配置に再配置する．再配置したシナプスは，図において赤線で示している．

これにより，従来のシナプス配置では活性できなかった緑色のカラムが活性し，入力をより多くのカラム群で表現できるようになる．その結果，入力データの CLA における分解能が向上し，予測精度も改善される．

5.2 アルゴリズム

提案法の疑似コードを **Algorithm 1** に示す．提案法では，W，A_l，A_h はパラメータとする．提案するシ

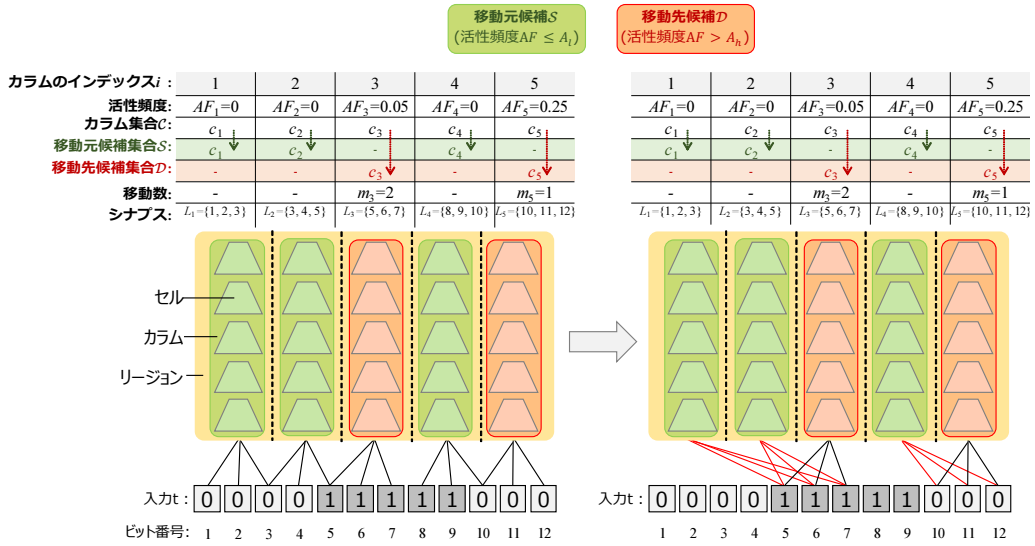


Fig. 3: 提案するシナプスの動的配置法

Algorithm 1 シナプスの動的配置法

Memo: 全カラム群 $\mathcal{C} = \{c_1, c_2, \ldots, c_{N_c}\}$，移動元候補集合 $\mathcal{S}$ $(\subseteq \mathcal{C})$，移動先候補集合 $\mathcal{D}$ $(\subseteq \mathcal{C})$，直近 W 入力時点におけるカラム i の活性頻度 AF_i^W，カラム i のシナプス移動数 m_i，カラム i のシナプス群 $\mathcal{L}_i$
Input: 移動元の閾値 A_l，移動先の閾値 A_h

Step 1: 移動元および移動先候補となるカラムの選択
$\mathcal{S} \leftarrow \mathcal{D} \leftarrow \emptyset$
for each $c_i \in \mathcal{C}$ **do**
  if $AF_i^W \leq A_l$ **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup c_i$
  end if
  if $AF_i^W > A_h$ **then**
    $\mathcal{D} \leftarrow \mathcal{D} \cup c_i$
  end if
end for
Step 2: 各移動先候補に対する移動カラム数の決定
for each $c_i \in \mathcal{D}$ **do**
  $m_i \Leftarrow |\mathcal{S}| \cdot \frac{AF_i^W - A_h}{\sum_{n=1}^{|\mathcal{D}|}(AF_i^W - A_h)}$
end for
Step 3: シナプスの移動
for each $c_i \in \mathcal{D}$ **do**
  for $loop = 1$ **to** $\lfloor m_i \rfloor$ **do**
    $s = \mathrm{argmin}_{c_j \in \mathcal{S}}\, j$
    $\mathcal{S} \leftarrow \mathcal{S} \backslash c_s$
    $\mathcal{L}_s \leftarrow \mathcal{L}_i$
  end for
  $m_i \leftarrow m_i - \lfloor m_i \rfloor$
end for
$I_m = 0$
while $\mathcal{D} \neq \emptyset$ **do**
  $d = \mathrm{argmin}_{c_j \in \mathcal{D}}\, j$
  $I_m = I_m + m_d$
  if $I_m \geq 1$ **then**
    $s = \mathrm{argmin}_{c_j \in \mathcal{S}}\, j$
    $\mathcal{S} \leftarrow \mathcal{S} \backslash c_s$
    $\mathcal{L}_s \leftarrow \mathcal{L}_d$
    $I_m = I_m - 1$
  end if
  $\mathcal{D} \leftarrow \mathcal{D} \backslash c_d$
end while

ナプスの再配置は，実行周期パラメータ W に対して，データの入力時点 $t = \{t : t \bmod W = 0 \wedge t \neq 0\}$ のときに実行する．これは，シナプスを再配置するために時間的な間隔を持たせることで，移動元と移動先のシナプスの見積もり精度を高め，重要なシナプスの移動を防ぐためである．また，$A_l \leq A_h$ となるようにこれらの値を設定する．これは，あるカラムが移動元候補と移動先候補に重複して選ばれることを避けるためである．以下，再配置の手順について述べる．

Step 1: 移動元および移動先候補となるカラムの選択

活性頻度 A_l 以下のカラムを移動元候補 $\mathcal{S}$，活性頻度 A_h より高いカラムを移動先候補 $\mathcal{D}$ にする．これは，入力データビットにおいて，1 の出現頻度が低く，カラムの活性に対する貢献度が低いシナプスを有するカラムと，1 の出現頻度が高く，カラムの活性に対する貢献度が高いシナプスを有するカラムを識別するためである．また，疑似コードに示すように，各カラムの活性頻度は直近 W 入力時点における活性頻度を用いる．これは，入力データの傾向が大きく変化するケースを想定したとき，全入力での活性頻度を用いると，変化後の入力分布を正確に捉えられないためである．

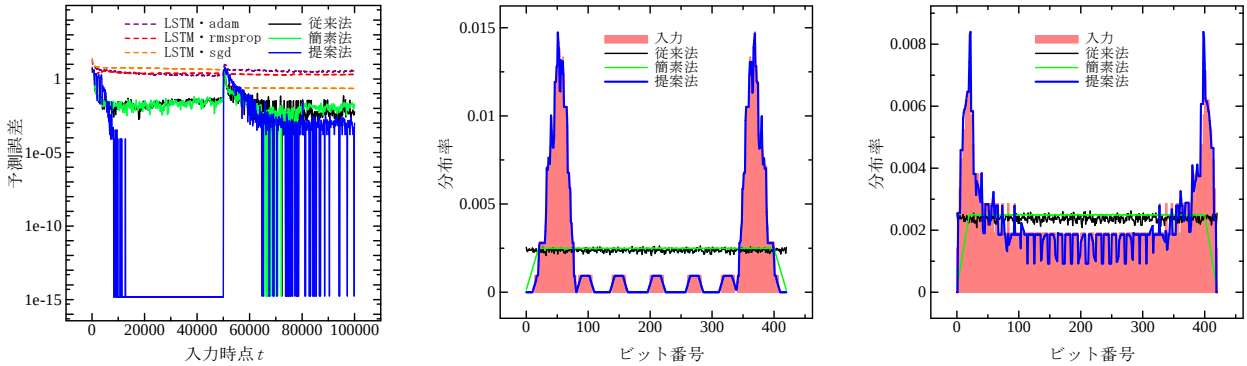
Step 2: 各移動先候補に対する移動カラム数の決定

移動元候補のカラム数 $|\mathcal{S}|$ と，移動先候補のカラム数 $|\mathcal{D}|$ およびそれらの活性頻度 AF^W から，各移動先候補 c_i に移動すべきカラム数 m_i を計算する．これは，移動先候補のカラム群の活性頻度の比をもとに，相対的に移動カラム数を決定することで，合理的に移動元候補のカラム群を配置するためである．

Step 3: シナプスの移動

各移動先候補のカラムにおいて，**Step 2** で求めたカラム数だけ，移動元候補のカラムにおけるシナプスを削除し，移動先候補が有するシナプスが配置された入力データビットにシナプスを再配置する．また，そのシナプスが有する永続値 p_c も移動先候補が有するシナプスと同様にする．これは，移動先候補のカラムが持つシナプスの情報をより多く保持することで，より適切なシナプス配置を早く実現するためである．

疑似コードにおいては，**Step 2** で求めた移動数 m_i の整数部を処理する部分と，小数部を処理する部分にわかれている．これは，移動元候補のカラム集合 $\mathcal{S}$ が持つシナプスを全て移動することを考えると，求めた移動数に対して一定の処理を行うことは困難であるため，各移動先候補で確実に必要と考えられる整数部を先に処理し，その後に移動数の小数部を重要度として



(a) 予測誤差の推移　(b) 学習前半の分布率　(c) 学習後半の分布率

Fig. 4: $x_1(t)$ における結果

扱うことで，残りの移動元候補を割合的に移動先候補へ振りわけるためである．

6　実験設定

提案するシナプスの動的配置法の効果を検証するため，従来の CLA[2]，簡素型 CLA[6]，提案する CLA(簡素型 CLA+シナプスの動的配置法) の 3 種類の CLA に，最適化アルゴリズムに adam，rmsprop，sgd をそれぞれ用いた 3 種類の LSTM を加えた 6 種類の方法を比較する．テスト時系列データは，先の報告 [8] で用いた正弦波，正弦波の合成波，ロジスティック写像をそれぞれ組み合わせた，以下の $x_1(t), x_2(t), x_3(t)$ を用いる．

$$x_1(t) = \begin{cases} \frac{1}{2} \cdot \sum_{k \in \{1,3,5,7,9\}} \frac{1}{k} \cdot \sin\left(\frac{(t-1)\cdot k \cdot \pi}{50}\right) + 0.5, & t \leq 50000 \\ \frac{1}{2} \cdot \sin\left(\frac{(t-1)\cdot \pi}{50}\right) + 0.5, & \text{otherwise} \end{cases} \tag{11}$$

$$x_2(t) = \begin{cases} 3.6 \cdot x_2(t-1) \cdot \{1 - x_2(t-1)\}, & t \leq 50000 \\ \frac{1}{2} \cdot \sum_{k \in \{1,3,5,7,9\}} \frac{1}{k} \cdot \sin\left(\frac{(t-1)\cdot k \cdot \pi}{50}\right) + 0.5, & \text{otherwise} \end{cases} \tag{12}$$

$$x_3(t) = \begin{cases} \frac{1}{2} \cdot \sin\left(\frac{(t-1)\cdot \pi}{50}\right) + 0.5, & t \leq 50000 \\ 3.6 \cdot x_3(t-1) \cdot \{1 - x_3(t-1)\}, & \text{otherwise} \end{cases} \tag{13}$$

ここで，t は入力時点である．また，$x_2(1) = 0.4$ とした．

入力時点 $t \in [1, 10^5]$ とした．CLA において，カラム数 $N_c = 2048$，1 カラム当たりのセルの数は 32 に設定した．シナプスの生成のパラメータは $R = 13$，$d = 0.8$ に設定した．簡素型 CLA における初期永続値 $P = 0.21$ に設定した．入力変換処理において，$[min, max] = [0, 1]$ とし，$N_b = 421$，$w = 21$ に設定した．空間プーリングのパラメータとして，活性状態にするカラムの上限数は $AC^{max} = 40$ にした．シナプスの接続と切断を決定する閾値は $T_c = 0.1$ に設定した．また，バンプアップとブーストの基準値を決定する係数は $K = 0.001$，ブースト値の最大値は $boost^{max} = 2.0$ に設定した．時間プーリングのパラメータとして，予測状態となるために必要な接続されている活性状態のセルの個数は $Ac = 15$ にした．提案法では，$W = 1000$，$A_l = 0$，$A_h = 0.02$ を用いた．LSTM には，Keras[7] を利用した．中間層は線形層と LSTM 層の 2 層とし，ユニット数はそれぞれ 100 とした．損失関数は平均二乗誤差を用いた．

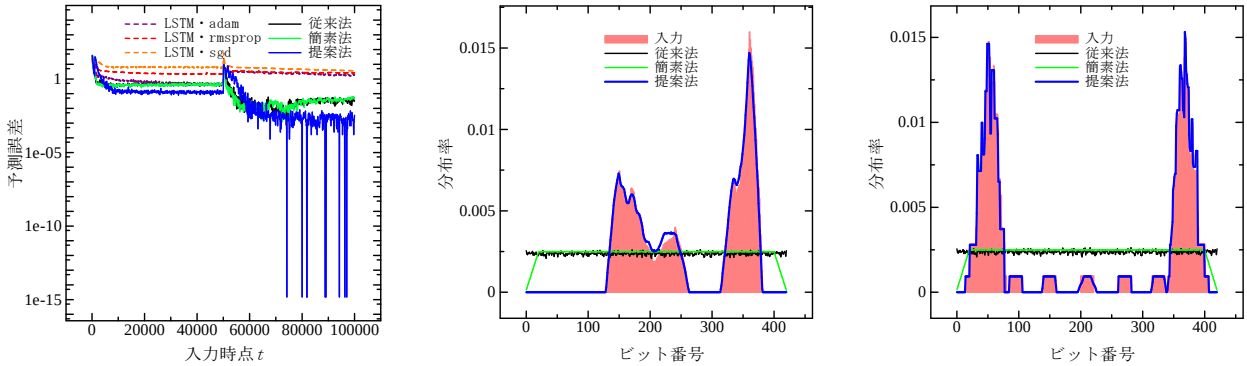
評価尺度として予測誤差を用いる．予測誤差は，時刻 t における CLA による予測値 $predict(t)$ と，時刻 $t+1$ の実際の入力データ $input(t+1)$ の差分である．予測誤差が小さいほど，予測が正確であると判断する．

7　実験結果と考察

まず，正弦波の合成波から正弦波に切り替わる $x_1(t)$ における結果について議論する．予測誤差の推移を Fig. 4a に示す．この結果から，入力の変化の前後によらず，3 種類の CLA における予測誤差が，3 種類の LSTM における予測誤差より低く推移することがわかる．また，3 種類の CLA の中で，提案する CLA は予測誤差を低く推移させ，最も予測精度が良好といえる．次に，入力データビット列における 1 の分布率と，入力変化前 ($t = 50,000$) および全入力後 ($t = 100,000$) のカラムが持つシナプスの分布率について議論する．入力時点 $t = 50,000$ における結果を Fig. 4b に示す．双方の形状が近いほど，カラムを入力データの内部表現に有効活用できる．この結果から，提案する CLA のシナプスの分布率が，入力データビット列における 1 の分布率に近いことがわかる．そして，入力時点 $t = 100,000$ における結果を Fig. 4c に示す．この結果から，提案する CLA は，変化した入力に合わせて，入力データビット列における 1 の分布率に近いシナプスの配置を実現できたことがわかる．

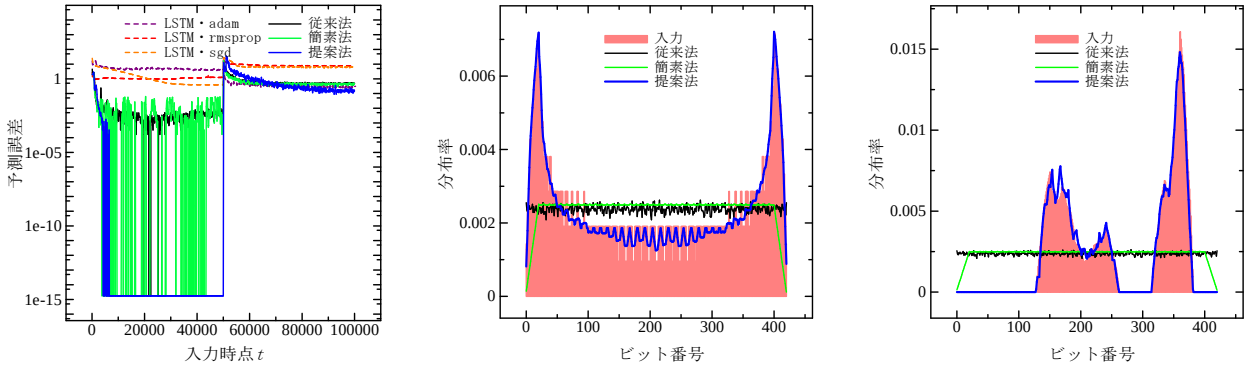
次に，ロジスティック写像から正弦波の合成波に切り替わる $x_2(t)$ における結果について議論する．予測誤差の推移を Fig. 5a に示す．この結果から，学習後半では，3 種類の CLA における予測誤差が，3 種類の LSTM における予測誤差より低く推移していることがわかる．その上，提案する CLA は予測誤差を低く安定して推移させることができ，最も予測精度が良好といえる．しかし，学習前半では，最適化アルゴリズムとして adam を用いた LSTM の予測誤差が，最終的に従来の CLA や簡素型 CLA と同等の結果を示すことがわかる．一方，提案する CLA の予測誤差はさらに低く安定しており，最も良い予測精度を示しているといえる．次に，入力データビット列に対する 1 の出現率とシナプスの分布率について，入力時点 $t = 50000$ における結果を Fig. 5b，$t = 100000$ における結果 Fig. 5c に示す．この結果から，提案する CLA は，変化した入力に合わせて，入力データビット列における 1 の分布率に近いシナプスの配置を実現できたことがわかる．

最後に，正弦波からロジスティック写像に切り替わる $x_3(t)$ における結果について議論する．予測誤差の推移を Fig. 6a に示す．この結果から，学習前半では，3 種類の CLA における予測誤差が，3 種類の LSTM における予測誤差より低く推移していることがわかる．その上，提案する CLA は予測誤差を低く安定して推移させることができ，最も予測精度が良好といえる．しか



(a) 予測誤差の推移　(b) 学習前半の分布率　(c) 学習後半の分布率

Fig. 5: $x_2(t)$ における結果



(a) 予測誤差の推移　(b) 学習前半の分布率　(c) 学習後半の分布率

Fig. 6: $x_3(t)$ における結果

し，学習後半では，最適化アルゴリズムとして adam を用いた LSTM の予測誤差が，最終的に従来の CLA や簡素型 CLA よりも低くなることがわかる．一方，提案する CLA の予測誤差はさらに低く安定しており，最も良い予測精度を示しているといえる．次に，入力データビット列に対する 1 の出現率とシナプスの分布率について，入力時点 $t = 50000$ における結果を Fig. 6b，$t = 100000$ における結果 Fig. 6c に示す．この結果から，提案する CLA は，変化した入力に合わせて，入力データビット列における 1 の分布率に近いシナプスの配置を実現できたことがわかる．

これらの結果から，提案法は，入力データに合わせてシナプスを再配置することで，入力変化から一定時間経過後の予測誤差を低く安定化できることがわかる．

8　まとめ

本稿では，CLA の内部表現にカラム群を有効活用し，入力データをより精緻に内部表現化することで時系列予測性能を向上することを目的として，カラムと入力データビット列を繋ぐシナプスを，入力データの偏りに合わせて動的に再配置する方法を提案した．従来の CLA および簡素型 CLA は，偏った入力値を受け取ると，カラムの活性頻度にも偏りが生じる．そのため，入力データを内部表現に変換する際，カラム群を最大限に活用することができない．提案法は，入力データビット列における 1 の出現頻度が低く，カラムの活性に対する貢献度が低いシナプスを，1 の出現頻度が高いデータビットの位置へと移動することで，カラム群を有効活用できるようにした．時系列パターンが途中で変化する 3 種類のテスト入力を予測した実験の結果，学習開始および入力変化から一定時間経過後の予測誤差を低く抑えることができ，予測精度を向上させることがわかった．また，変化前後のそれぞれの入力に合わせて，入力データビット列における 1 の出現率に近いシナプス配置を実現することで，カラム群を有効活用できることがわかった．

今後の課題は，提案法を用いた際に，再配置直後の予測精度が悪い問題を改善することである．また，より複雑な入力における提案法の効果も検証する．

参考文献

1) S. Ahmad and J. Hawkins : Properties of Sparse Distributed Representations and their Application to Hierarchical Temporal Memory，Numenta，1/18（2015）

2) J. Hawkins，A. Subutai，and D. Dubinsky : Hierarchical temporal memory including HTM cortical learning algorithms，Technical report，Numenta，Inc（2010）

3) NuPIC，https://github.com/numenta/nupic (2017/9/12 アクセス)

4) S. Hochreiter and J. Schmidhuber : Long Short-Term Memory，*Neural Computation*，Vol. 9，No. 8，1735/1780（1997）

5) Y. Cui，S. Ahmad，and J. Hawkins : Continuous Online Sequence Learning with anUnsupervised Neural Network Model，*Neural Computation*，Vol. 28，Issue. 11，2474/2504 (2016)

6) 青木 健，高玉 圭樹，佐藤 寛之 : 大脳新皮質アルゴリズムの簡素化と予測精度向上に関する検討，計測自動制御学会 システム・情報部門 学術講演会 2018(SSI 2017)，135/140 (2017)

7) Keras，https://github.com/keras-team/keras (2018/5/1 アクセス)

8) 青木 健，鈴ヶ嶺 聡哲，高玉 圭樹，佐藤 寛之 : 大脳新皮質学習におけるシナプスの動的再配置に関する検討，計測自動制御学会 システム・情報部門 学術講演会 2018(SSI2018)，in USB memory，6 pages（2018）