時系列予測のための
大脳新皮質学習アルゴリズムの研究

青木 健

電気通信大学 大学院 情報理工学研究科 情報学専攻

博士（工学）学位申請論文

2022 年 3 月

時系列予測のための
大脳新皮質学習アルゴリズムの研究

論文審査委員会

主査　佐藤　寛之　准教授
委員　西野　哲朗　教授
委員　髙玉　圭樹　教授
委員　庄野　逸　　教授
委員　高橋　裕樹　准教授

Abstract

Time-series data prediction is one of the most important topics in computational intelligence and essential information technology for appropriate decision-making. The effectiveness of a long short-term memory (LSTM) based on the recurrent neural network has been known in the domain of time-series data prediction. The cortical learning algorithm (CLA) is also a promising time-series data prediction method designed based on the human neocortex. A CLA predictor is composed of columns, cells, and synapses that represent relations among them. The CLA represents input data and its context by the patterns of columns and cells, respectively. CLA treats data as discrete value representation, whereas LSTM treats data as real value representation generally. An advantage of CLA is that its predictor can be updated online while receiving input data. This study aims to develop a methodology of CLA to improve the time-series data prediction accuracy and verify its effectiveness. Firstly, we propose (1) an adaptive synapse arrangement method. This method works to utilize columns that are not utilized for the data representations in the conventional CLA and improve the prediction accuracy on artificial and real-world electricity load predictions. Also, to accelerate the online predictor construction, we propose (2) a cell synapse update method of inactive cells. The conventional CLA updates the cell synapses when the prediction succeeds. The proposed method updates the cell synapses even when the prediction fails. To improve the inside data representation and correct the prediction using CLA, we propose (3) a double-layered predictor and show that the proposed method improves the prediction accuracy. Lastly, to transform inside representations of the CLA predictor into values that humans can understand directly, we also propose (4) a column-based decoder. The conventional CLA needs a learning process to build the decoder. The proposed method utilizes the existing column synapse network. The proposed method avoids not only the learning process but also accelerates prediction accuracy.

概要

時系列予測は，適切な意思決定のために不可欠な情報技術であり，計算知能の重要な研究トピックである．従来法として，ニューラルネットワークに基づく長短期記憶 (Long Short-Term Memory，以下 LSTM) の有用性が知られている．他方，高度な機能を司る大脳新皮質に注目してモデル化した大脳新皮質学習アルゴリズム (Cortical Learning Algorithm, 以下 CLA) がある．CLA の予測器は，カラム，セル，これらを関係付けるシナプスで構成される．カラムとセルの状態を変化させ，入力データと文脈を表現する．CLA は，データを離散表現するところが LSTM とは異なる．また，CLA はオンラインで予測器の内部構成を更新するところに利点がある．本論文では，時系列予測性能を高める大脳新皮質学習アルゴリズムの設計論を構築し，その効果を検証することを目的とした．まず，(1) 入力データ値の偏りにあわせて予測器を変更するシナプスの適応配置法を提案した．これにより，従来法では，入力データの表現に利用されなかったカラムを活用できるようになり，人工のテスト時系列データ，実世界の電力消費量データにおける予測性能が改善されることを明らかにした．次に，オンラインでの予測器の適応的構成を促進するため，(2) セルのシナプスネットワークの構築を促進させる不活性セルのシナプス更新法を提案した．従来法が，予測の成功時にシナプスを更新するのに対し，提案法は，予測の失敗時にもシナプスを更新する手段を導入し，予測精度が改善されることを示した．次に，予測器内におけるデータの表現力を高め，予測を補正するため，(3)CLA の予測器を階層化して予測を補正する方法について述べ，時系列予測性能が高まることを明らかにした．さらに，予測器の内部状態を数値に変換するデコードについて，(4) 学習を回避するカラムに基づくデコーダを提案した．従来法が，デコーダの構築に学習を要するのに対し，提案法は，既存のシナプスネットワークを活用することで，学習を回避してデコードできるだけでなく，テスト時系列データ，電力消費量の予測性能が高まることを示した．

目次

図目次

表目次

第 1 章

序論

1.1 研究背景

時系列予測は，適切な意思決定のために不可欠な情報技術であり，計算知能分野における重要な研究トピックのひとつである．例えば，今日までのウィルス感染者数の時系列データをもとに，明日の感染者数を出力することが，時系列予測である．感染者数の増減を予測できれば，適切な対策を施すことができる．従来法として，旧来のニューラルネットワークに基づくリカレントニューラルネットワーク [1, 2] を拡張した長短期記憶 (Long Short-Term Memory，以下 LSTM)[3] の有用性が知られている．一方，人間は，先を見据えた視点に立つフィードフォワードによって，時系列予測の成否から学習するといわれている [4]．このような人間の高度な機能を司る脳の部位を大脳新皮質という．大脳新皮質のはたらきをモデル化した階層時間記憶 (Hierarchical Temporal Memory)[4, 5] という概念がある．階層時間記憶を具現化した方法として，大脳新皮質学習アルゴリズム (Cortical Learning Algorithm, 以下 CLA)[6, 7, 8] が提案されている．タクシーの需要予測などにおいて，CLA が LSTM より高い予測精度を示すことが報告されている [9].

CLA の予測器は，カラム，セル，シナプスという要素で構成される．カラムとセルは，入力データとシナプスの関係によって状態を変化させる．CLA は，入力データを活性状態のカラムのパターンで離散表現する．また，入力データの文脈を活性状態のセルのパターンで離散表現する．予測は，予測状態のセルのパターンで離散表現する．このように，CLA は，データを離散表現するところが，一般的な LSTM とは異なる．また，CLA

は，データを受け取りながらオンラインで予測器の内部構成を更新するところに利点があり，有望な時系列予測法のひとつであると考えられる．

従来の CLA は，オンラインで更新する予測器の仕組みに起因して，予測精度が悪くなることがある．(1) 従来の CLA は，入力値の範囲をあらかじめ設定する必要がある．従来の CLA は，指定された範囲を均等に表現しようとする．指定された範囲の一部に偏った値が入力され続けると，値の差異を精緻に表現できず，予測精度が悪くなることがある．(2) CLA は，シナプスのネットワークを構築しながら予測するため，シナプスのネットワークが成熟するまでは，予測精度が悪くなることがある．(3) 人間の大脳新皮質には，階層構造がある．層と層との相互作用が，予測の補完に寄与していると考えられる [4]．一方，従来の CLA は，1 層構造であり，予測の補完機能がない．人間の大脳新皮質にあると考えられる予測の補完を CLA で実現できれば，予測精度のさらなる改善が期待できる．(4) CLA の予測器の内部状態は，人間が理解できる表現形式ではない．人間が理解できるように，CLA の予測器の内部状態をデコードする既存の仕組みがある．しかし，既存のデコーダは，学習を要する．そのため，学習中は，予測精度が悪くなる．

1.2 研究目的と方法

本研究では，CLA による時系列予測の精度を向上するため，予測器の構成とその取り扱いに関する方法論を構築し，その効果を検証することを大目的とする．大目的を達成するため，様々なデータへの対応，予測精度の向上，予測速度の向上の三つを目的として上述の従来の CLA における問題点を解決する．詳細として，以下の方法を構築し，効果を検証する．

(1) 様々なデータへの対応を目的として，入力値の偏りにあわせて適応的にシナプスを配置する方法論を構築する．従来法は，あらかじめ決めた入力値の範囲に均等にシナプスを配置し，その範囲の値を均一に予測器内で表現しようとする．これに対し，入力頻度が低いデータに紐づいたシナプスを，入力頻度が高いデータへオンラインで移動させることで，入力値にあわせて予測器内の表現粒度を変更する方法を提案する．(2) 予測精度の向上を目的として，シナプスネットワークの構築を促進する方法論を構築する．従来の CLA は，予測が成功したときに，成功をもたらしたシナプスを強化することで徐々にシナ

プスネットワークを構築していく．これに対し，予測が失敗したときにも，シナプスを更新する仕組みを導入した方法を提案する．(3) 同様に予測精度の向上を目的として，CLA の予測器を多層化する方法論を構築する．従来の CLA は，1 層構造である．これに対し，CLA の予測器を二つ重ね，上位の予測器が，下位の予測器の予測を補正するように相互作用させる方法を提案する．(4) 予測速度の向上を目的として，学習を回避して CLA の予測器の内部状態をデコードする方法論を構築する．従来の CLA は，デコードのために予測器の内部状態と入力値のマッチングを学習する．これに対し，入力値を予測器の内部状態にエンコードする既存の仕組みを利用し，予測器の内部状態を入力値にデコードする方法を提案する．

本研究で提案する方法の効果は，時系列データの予測精度によって検証する．まず，人工のテスト時系列データを検証に用いる．次に，New York ISO[10] で公開された実世界の時系列データである電力消費量を用いる．これにより，現実的な問題における予測性能の検証を可能とする．

1.3 関連研究

時系列予測に関する関連研究について，数理モデルによるアプローチとニューラルネットワークをベースとしたアプローチにわけて述べる．

1.3.1 数理モデルによるアプローチ

時系列データを予測する数理モデルの分類器がある．確率モデルに基づく方法として，観測されない隠れ状態を持つマルコフ過程である隠れマルコフモデル (Hidden Marcov model, 以下 HMM)[11] がある．これまでに，HMM に基づいて音声認識 [12] や株価予測 [13]，タンパク質構造を予測する研究 [14] がある．これらの研究では，時系列で変化するデータを HMM によって確率的に表現することで時系列を予測する．

動的ベイジアンネットワーク (Dynamic Baysian Network, 以下 DBN)[15, 16] を用いた，感情認識 [17, 18] や遺伝子予測の研究 [19] がある．これらの研究では，グラフ構造を持つ確率モデルであり，不確実性を有するいくつかの部分領域をモデル化することができるベイジアンネットワーク [20] を，時系列情報を含むように拡張した DBN によって時系

列を予測する．

サポートベクトルによって分類する学習器であるサポートベクターマシン (Support Vector Machine, 以下 SVM)[21] を用いた，感情認識 [22] や倒産予測 [23]，タンパク質細胞内局在予測の研究 [24] がある．これらの研究では，SVM によって汎化性を向上させるとともに，各時系列データとの誤差を最小化することで時系列を予測する．

1.3.2 ニューラルネットワークをベースとしたアプローチ

脳の仕組みに着想を得たアルゴリズムによって時系列データを予測する方法がある．脳の神経網を数理モデル化した人工ニューラルネットワーク (Artificial Neural Network, 以下 ANN) が代表的である．ANN の中でも，時系列データを予測するアルゴリズムとして，リカレントニューラルネットワーク (Recurrent Neural Network, 以下 RNN)[1, 2] がある．RNN は，ニューラルネットワークの学習に加えて，前の時点の中間層の情報も考慮して時系列予測する．この手法は，負荷予測 [2] や音声認識 [25]，為替レート予測 [26] などの研究に用いられている．RNN において，シナプスの重みを更新するための勾配降下法 [1, 27, 28, 29, 30, 31] や，過去のデータを反映するための時間遅れ [32, 33, 34, 35] など，時系列予測のための様々な研究がされている [3]．先に述べたタクシー需要予測 [9] においては，フィードフォワードニューラルネットワークの一種である ELM(online sequential Extreme Learning Machine)[36]，ランダムに接続される RNN である ESN(Echo state network)[37] が利用されている．RNN を改良した方法として長短期記憶 (Long Short-Term Memory，以下 LSTM) [3] があり，このアルゴリズムによる時系列予測の有用性が知られている．LSTM は，長い時系列の情報を伝播させるために，RNN における中間層の代わりに LSTM block を用いて時系列を予測する．この手法は，テキスト読み上げシステム [38] や，言語モデルの実装 [39]，フレーム音素分類 [40] などの研究に用いられている．

ANN には，抽出の計算式を畳み込みに置き換えることで，多層構造で入力の特徴量を抽出する畳み込みニューラルネットワーク (Convolutional Neural Network, 以下 CNN)[41] や，RNN のような仕組みを CNN を用いて構成することで，ある区間の時系列データをまとめて入力できるようになり，並列処理を可能にした QRNN(Quasi-RNN)[42]，PixelCNN[43] という画像認識のための CNN をベースに作られた，音声波

形を生成するためのディープニューラルネットワークである WaveNet[44]，CNN を組み合わせた階層性の構造を持ち，状態遷移を確率モデルとして扱うことで時系列を予測する RCN(Recursive Cortical Network)[45]，大脳新皮質の理論である予測符号化 (Predictive Coding)[46] に基づいて設計された CNN と LSTM を階層的に構築した PredNet(Predictive Coding Network)[47], RNN の一種で，中間層に相当する Resorvior と出力層に相当する Read out layer 間のシナプスについてのみ学習する RC(Reservior Computing)[48, 49] が存在する．また，自然言語処理で分野を中心に発展した技術として，エンコーダとデコーダとして RNN を用いた，ニューラルネットワークによる機械翻訳を実現する Sequence To Sequence[50]，エンコーダ・デコーダ構造において人間の目の仕組みを参考に作られた Attension 機構をベースとすることで，高速化と高精度化を実現した Transformer[51]，Transformer をベースとして文頭と文末の双方向からの学習，および文章同士の関係性の学習を実現した BERT(Bidirectional Encoder Representations from Transformers)[52] が存在する．Transformer においては，時系列データの予測にも適用された事例がある [53].

ANN 以外にも，脳の仕組みに着想を得た手法が存在する．複数の種類の細胞から構成され，それらが捉えた局所特徴量の統合によってパターン認識を行うネオコグニトロン (Neocognitron)[54] や，ノードごとにマルコフ確率場を形成し，階層構造化することでボトムアップおよびトップダウン入力を実現した，階層時間記憶 (Hierarchical Temporal Memory，以下 HTM) [4, 5] をもとに構成したアルゴリズムである Zeta1[55]，ベイジアンネットワークを核として，複数のモデルを統合したアルゴリズムである BESOM[56] がある．

1.3.3 CLA における関連研究

CLA を拡張する研究として，脳の嗅内皮質に存在し，空間の座標を表す格子細胞を CLA の内部で実現する研究 [57, 58, 59] がある．この研究では，格子細胞による処理を手書き文字の認識タスクである MNIST に応用し，その分類における有効性が示されている [57]．また，ニューラルネットワークに CLA の理論を組み込む研究がある [60]．この研究では，ディープラーニングの性能を改善することを目的として，ランダムフォレストと CLA を統合したアルゴリズムを用いて平均絶対パーセント誤差 (MAPE) の最小化に

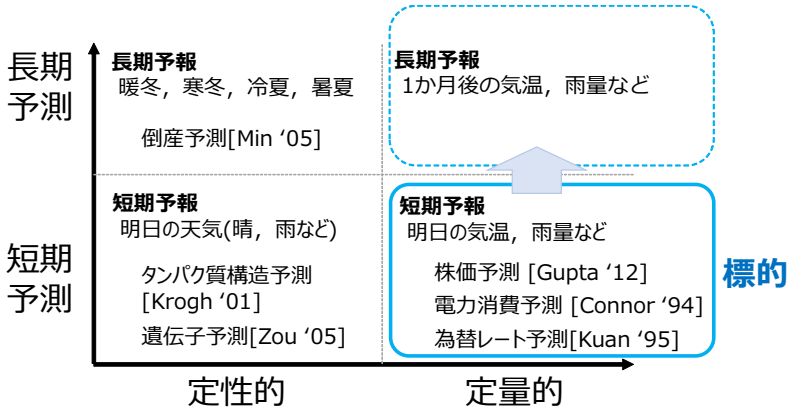


図 1.1: 本論文における問題の分類

取り組んでいる．このアルゴリズムはロボットの障害物回避にも応用され，その有効性が示されている [61].

1.4 研究の位置づけ

本研究の位置づけについて，取り扱う問題の観点から述べる．

初めに，時系列予測の問題の種類について分類したものを**図 1.1** に示す．横軸は予測結果を数値化することが可能かどうか，を表す定性的/定量的で分類し，縦軸は予測の時期を表す短期予測/長期予測で分類した．予測の時期については，長期の予測であるほど難しいと考えられる．例えば，**図 1.1** に示したような暖冬や寒冬，冷夏や暑夏を予測するのは定性的かつ長期予測の問題，晴や雨などの明日の天気を予測するのは定性的かつ短期予測の問題，一か月後の気温や雨量などを予測するのは定量的かつ長期予測の問題，明日の気温や雨量などを予測するのは定量的かつ短期予測の問題，と分類することができる．長期予測と短期予測の違いについては，データのサンプリング間隔などの加工方法にも依存するため，データによっては明確な分類は難しい．しかし，予測したいデータの要望によって加工が困難となるケースが考えられ，そのようなケースでは長期予測と短期予測に差異が生まれると考えられる．例えば，最高気温を予測したい，という要望に対しては，日ごとの最高気温の推移を予測する方法と，1 時間ごとの気温の推移を予測して最高気温を抽出する方法が考えられるが，1 時間ごとの気温の推移を予測したい，という要望に対

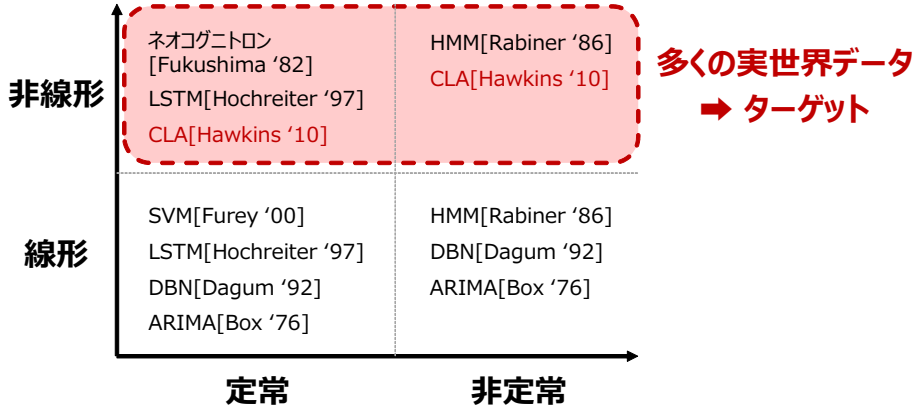


図 1.2: 代表的な予測技術の位置づけ

しては，日ごとの推移の予測では不十分となる．また，関連研究で取り扱われていた事例についても，**図 1.1** のように分類することができる．本研究では，定量的かつ短期予測の問題に取り組む．定量的かつ長期予測の問題も重要だが，長期予測は短期予測を繰り返すことで実現できると考え，前述の問題に取り組む．

また，1.3 節で述べた関連研究と本研究で着目する CLA を，取り扱う時系列データの特徴によって分類したものを**図 1.2** に示す．線形/非線形および定常/非定常の時系列データに適するかどうかで，それぞれの方法を分類した．時点 t のデータを $f(t)$ で表現したとき，線形の時系列データは，$f(t) = 2t$ のようなデータであり，加法性および斉次性が成り立つデータを指す．定常的なデータとは，時点 t にかかわらず，平均と分散が一定で，自己相関係数が時点の差のみに依存するデータを指す．

CLA は，非線形，かつ定常および非定常のデータに対応する特徴を有する．実世界の時系列データは，非線形かつ非定常となるデータが大半を占めている．非定常的な時系列データを変換することで，データに定常性を持たせる方法はいくつか存在するが，オンラインで予測することを考慮した場合，非定常的なデータも取り扱えることが望ましいと考えられる．そのため，CLA は，オンラインでの時系列予測を実現する手法として有望なアルゴリズムであると考えられる．また，この中で CLA と類似した特徴を有するアルゴリズムとして，DBN を含むベイジアンネットワーク，LSTM を含むニューラルネットワーク，ネオコグニトロンが挙げられる．

まず，ベイジアンネットワークは，大脳新皮質との類似性が報告されている．ベイジ

アンネットワークを用いた大脳新皮質のモデルも提案されている．CLA のもととなる HTM の概念は，階層的に配置されたノードの集まりであるベイジアンネットワークに例えることができる．しかし，HTM は，視覚野における不変表現の扱いを含む点がベイジアンネットワークと異なる．次に，ニューラルネットワークは，人間のニューロンをモデル化した手法であり，CLA もニューラルネットワークの一種とみなすことができる．ニューラルネットワークと CLA の相違点は，CLA は大脳新皮質の機能を模倣することが目標であり，より脳に近い特徴を考慮して設計されている点が挙げられる．例えば，ニューラルネットワークのシナプスの重みは連続値だが，CLA のシナプスは状態として離散値で表現される．最後に，ネオコグニトロンは，人間の視覚神経を参考に作られた方法である．初期に提案された深層学習モデルのひとつであり，ニューラルネットワークの一種といえる．そのため，ニューラルネットワークと同様の特徴を持っており，より CLA が大脳新皮質の特徴に近い点が異なる．

CLA は，6 層であるといわれる人間の大脳新皮質のうち，第 3 層に近い役割を持つといわれている [6]．このように，CLA は，人間の脳の類似性を高めるように設計されているが，その正しさと妥当性を評価することには困難があり，本論文での議論の対象にはしない．しかし，時系列予測の手段として，CLA は，ニューラルネットワークの一種である LSTM より良好な時系列予測性能を示す結果が報告されている [9]．CLA は，有望な時系列予測アルゴリズムのひとつであるといえる．ニューラルネットワークなど比較すると，CLA に関する研究は極めて少ない．1.3.3 節に示したように，実際の脳に存在する格子細胞を実現し，それを物体認識に応用するような研究や，ニューラルネットワークに CLA の原理を組み込むことで予測精度を向上させるような応用研究はあるものの，方法論に関する研究は，ほとんど存在しない．本論文では，CLA の方法論に着目し，時系列予測性能を高めることに取り組む．効果は，従来の CLA と提案する方法を組み込んだ CLA によって検証する．また，関連する従来法として，タクシー乗車数の CLA による予測に関する文献 [9] で用いられたニューラルネットワークに基づく方法から，LSTM を取り上げて比較対象にする．

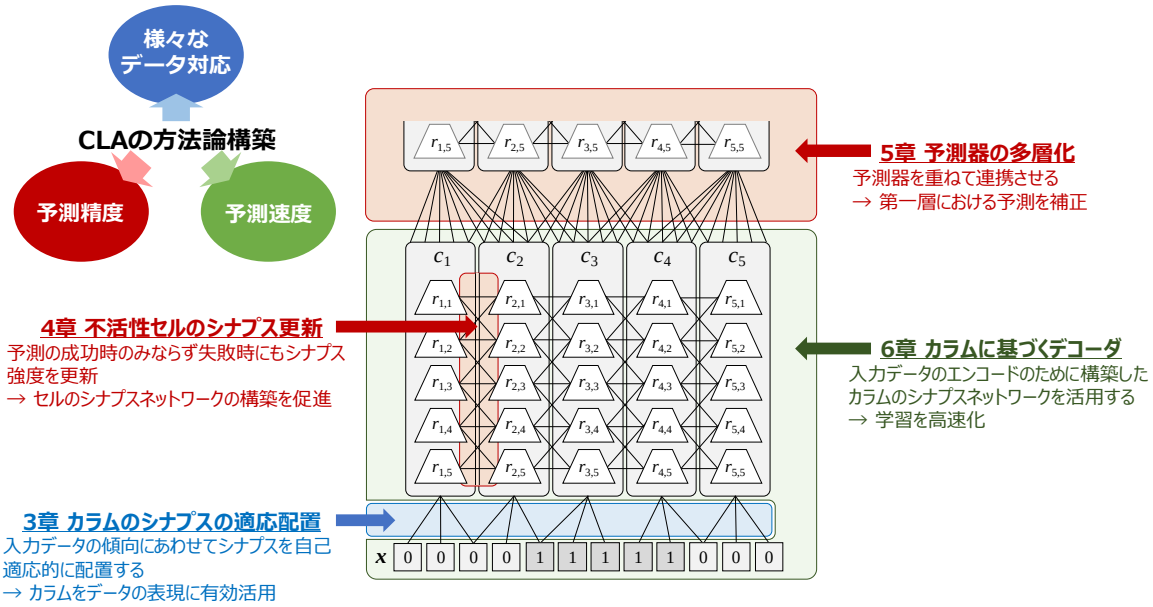


図 1.3: 本論文の全体像

1.5　本論文の構成

本章に続く第 2 章において，従来法である大脳新皮質学習アルゴリズム (CLA) による時系列予測について述べる．予測器の構成，手順の全体像，各手順の詳細について述べる．第 3 章以降において，本論文で提案する方法について述べる．本研究で提案する方法の全体像を**図 1.3** に示す．モノトーンで示した CLA の予測器に対し，各章で注目する部分を色付けした．CLA の予測器の詳細は，第 2 章で述べるが，ここでは，本論文の全体像を説明するため，予測器の構成要素を簡単に紹介する．各時点の入力データ $\boldsymbol{x}$ は，ビット列である．予測器は，カラム，セル，シナプスで構成される．c_i $(i = 1, 2, \ldots, n_c)$ が，カラムである．$r_{i,j}$ $(j = 1, 2, \ldots, n_r)$ が，セルである．カラムと入力データは，カラムのシナプスで結ばれる．セルとセルは，セルのシナプスで結ばれる．

3 章では，入力値の偏りにあわせて適応的にシナプスを配置することで様々なデータに対応するため，カラムのシナプスの適応配置法を提案する．予測器内のカラムを入力データの表現に有効活用することで，入力データをより詳細に表現する．**図 1.3** に示す予測器において，青色で示すカラムのシナプスに関する方法である．時系列データの予測におい

て，入力データによるシナプス配置の変化と予測誤差の変動を分析する．4 章では，シナプスネットワークの構築を促進することで予測精度を向上するため，不活性セルのシナプス更新法を提案する．これは，予測が成功したときのみならず，失敗したときにもシナプスを更新する仕組みである．**図 1.3** に示す予測器において，赤色で示すセルのシナプスに関する方法である．時系列データの予測において，この手法による予測精度への影響と予測器の内部表現の変化を解析する．5 章では，予測の補完をすることで予測精度を向上するために，予測器を多層化する方法を提案する．CLA の予測器を重ねることで，入力データの抽象表現を獲得し，それを予測の補正に反映する相互作用を実現する．**図 1.3** に示す予測器において，赤色で示す予測器と下位の予測器に関する方法である．時系列データの予測において，補正の効果とそれによる予測精度への影響を従来手法と比較して検証する．6 章では，学習を回避して CLA の予測器の内部状態をデコードすることで予測誤差の収束速度を向上するために，カラムに基づくデコーダを提案する．入力値を予測器の内部状態にエンコードする既存の仕組みを利用し，予測器の内部状態を入力値にデコードする．これにより，学習を回避して予測値を出力できるようにする．**図 1.3** に示す予測器において，緑色で示すカラムと入力データに関する方法である．時系列データの予測において，予測の収束速度の変化を確認する．7 章では，**図 1.3** に示す予測器において，全ての手法を適用した CLA の方法論を構築する．本論文で提案する四つの方法を統合し，その時系列予測性能を検証する．最後に，8 章では本論文をまとめ，今後の課題についても述べる．

第 2 章

大脳新皮質学習アルゴリズム

本章では，従来の大脳新皮質学習アルゴリズム (Cortical Learning Algorithm, CLA) について述べる．

2.1 予測器

CLA[8] の予測器を**図 2.1** に示す．CLA は，時点 t ごとにビット列 $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$ として入力値を受け取る．CLA の予測器は n_c 本のカラムで構成される．各カラム c_i $(i = 1, 2, \ldots, n_c)$ は，通常状態，活性状態の二つの状態を持つ．これは $c_i.st \in \{\text{Normal, Active}\}$ として表現される．カラムは，入力データを CLA の予測器内部で表現するための素子である．また，各カラム c_i は，n_{cy} 本のシナプスを持つ．各カラムのそれぞれのシナプス $c_i.y_k$ $(k = 1, 2, \ldots, n_{cy})$ は，あるデータビット x_i $(i = 1, 2, \ldots, n)$ に紐づけられる．このカラムのシナプスを用いて，入力ビット列とカラム群の関係が構築される．各カラム c_i は，n_r 個のセルを持つ．セルは，各文脈における入力データを CLA の予測器内部で表現するための素子である．このセルがニューロンに相当する．各セル $r_{i,j}$ $(j = 1, 2, \ldots, n_r)$ は，通常状態，活性状態，予測状態の三つの状態を持つ．これは，$r_{i,j}.st \in \{\text{Normal, Active, Predictive}\}$ として表現される．各セル $r_{i,j}$ は，セルのシナプス $r_{i,j}.y_k$ $(k = 1, 2, \ldots)$ を持つ．各セルのシナプスは，他のセルと紐づけられ，セル間の関係の構築に用いられる．ここで，カラム数 n_c，各カラムのシナプス数 n_{cy}，セル数 n_r は，パラメータである．

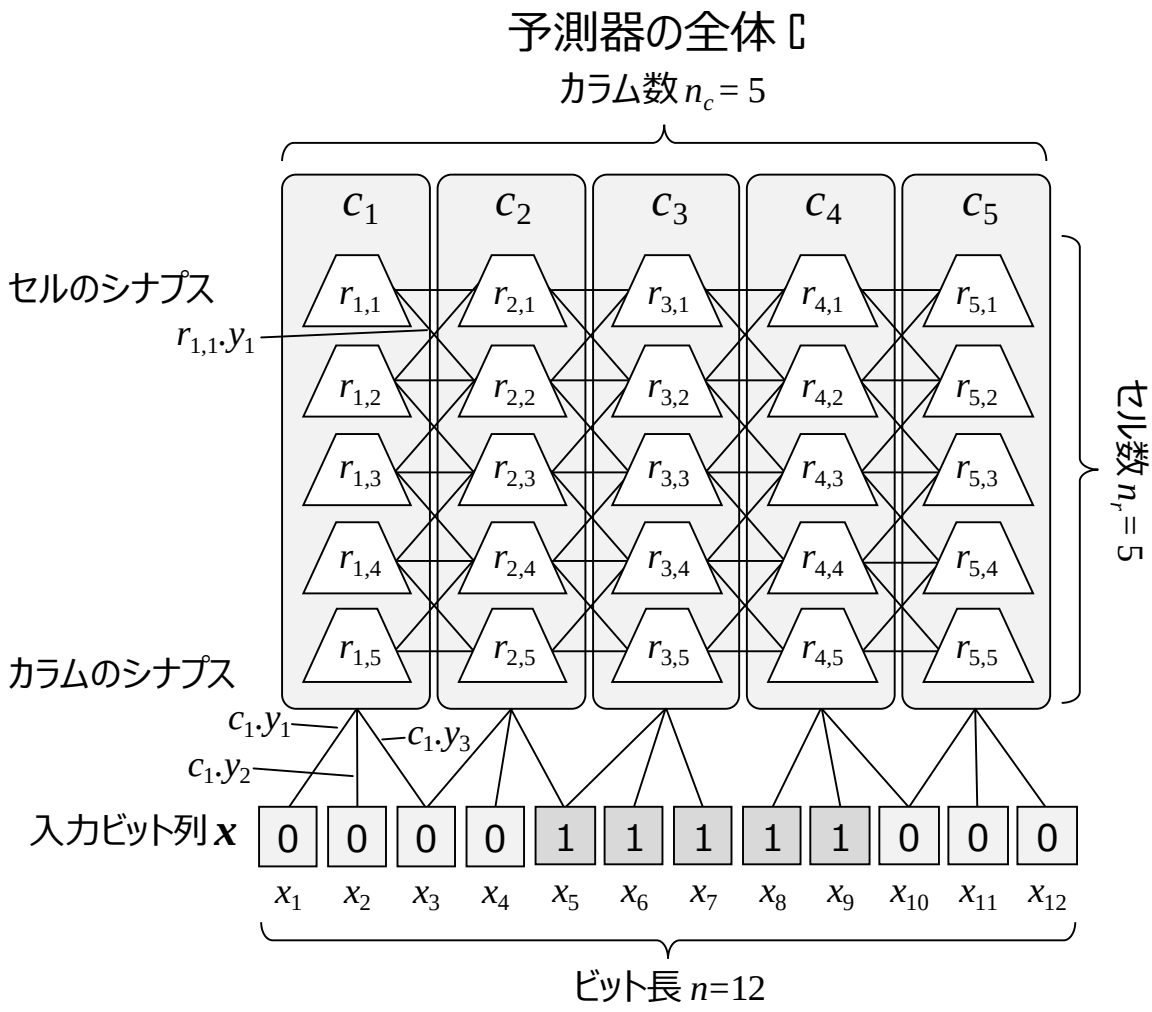


図 2.1: CLA の予測器構成

2.2　全体像

CLA の全体像の擬似コードを **Algorithm 1** に示す．擬似コード内で使用される変数とパラメータを表 **2.1** と **2.2** に示す．**Algorithm 1** について，以下で説明する．

1 行目において，CLA は，**Algorithm 1** に示した予測器 $\mathcal{C}$ に含まれる全てのカラム，セル，シナプスを初期化する．3–7 行目を繰り返すことで，各時点 t において予測器 $\mathcal{C}$ の内部状態を変化させる．繰り返す部分の処理について，図 **2.2** に示す．3 行目において，CLA は，入力データ $X(t)$ を受け取り，それをバイナリ値であるビット列 $\boldsymbol{x}$ にエンコー

Algorithm 1 大脳新皮質学習アルゴリズム (CLA)

1: $\mathcal{C} \leftarrow$ Initialization ▷ **Algorithm 2**
2: **for each** time-step t **do**
3: $X(t) \leftarrow$ Receiving input value
4: $\boldsymbol{x} \leftarrow$ Encoding $(X(t))$
5: $\mathcal{C} \leftarrow$ Spatial pooling $(\mathcal{C}, \boldsymbol{x})$ ▷ **Algorithm 3**
6: $\mathcal{C} \leftarrow$ Temporal pooling $(\mathcal{C})$ ▷ **Algorithm 4**
7: $\overline{X}(t+1) \leftarrow$ Decoding $(\mathcal{C})$
8: **end for**

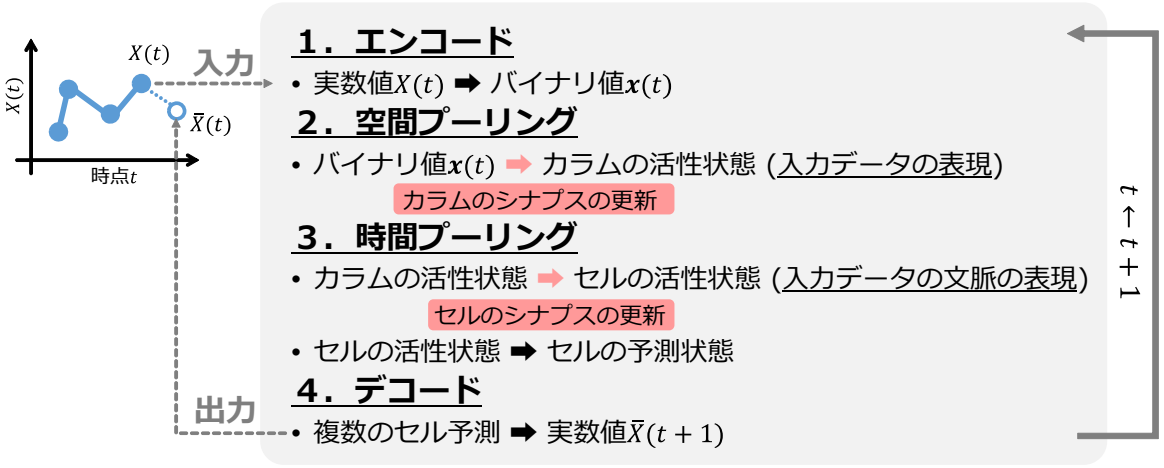


図 2.2: CLA の全体像

ドする．5 行目は，空間プーリングと呼ばれるプロセスで，入力データを全カラム群の中から活性状態となるカラムの部分集合として表現する．また，このプロセスでカラムのシナプスを更新する．6 行目は，時間プーリングと呼ばれるプロセスで，入力データの文脈を全セル群の中から活性状態となるセルの部分集合として表現する．このプロセスでは，セルのシナプスを更新する．また，CLA は，次時点 $t+1$ に入力されるデータを予測状態となるセルの部分集合として表現する．最後に 7 行目では，予測状態のセル群を入力データと同じ形式に変換する．CLA はこれらの手順を時点ごとに繰り返すことでオンライン学習をするアルゴリズムで，一般的なニューラルネットワークのように学習フェーズと評価フェーズがわかれていない．

CLA の手順の詳細について，それぞれ以下に述べる．

表 2.1: 擬似コードにおける変数

名前			説明
$X(t)$			時点 t における入力値
$\boldsymbol{x}$			入力ビット列，$\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$
$\mathcal{C}$			予測器全体，カラム集合 $\mathcal{C} = \{c_1, c_2, \ldots, c_{n_c}\}$
c_i			i 番目のカラム
	$.st$		カラム c_i の状態，$c_i.st \in$ {Normal, Active}
	$.ol$		c_i のオーバーラップ値，1 のビットと接続されたカラムのシナプス数
	$.bst$		カラム c_i のブースト値
	$.as$		カラム c_i の活性化スコア
	$.zf$		オーバーラップ値が 0，$c_i.ol = 0$ となる頻度
	$.af$		活性状態となる頻度
	$.af^w$		直近 w 時点において活性状態となる頻度
	$.y_k$		カラム c_i における k 番目のシナプス
		$.p$	カラム c_i における k 番目のシナプスの永続値
		$.a$	カラム c_i における k 番目のシナプスと紐づけられたビット，$c_i.y_k.a \in \{x_1, x_2, \ldots, x_n\}$
$r_{i,j}$			カラム c_i の j 番目のセル
	$.st$		セル $r_{i,j}$ の状態，$r_{i,j}.st \in$ {Normal, Active, Predictive}
	$.y_k$		セル $r_{i,j}$ における k 番目のシナプス
		$.p$	セル $r_{i,j}$ における k 番目のシナプスの永続値
		$.a$	セル $r_{i,j}$ における k 番目のシナプスと紐づけられたセル，$r_{i,j}.y_k.a \in \{r_{1,1}, \ldots, r_{n_c,n_r}\}$

2.3　初期化

Algorithm 1 の 1 行目で実行される初期化の詳細を **Algorithm 2** に示す．**Algorithm 2** について，以下で説明する．

2–14 行目において，CLA は，各カラム c_i $(i = 1, 2, \ldots, n_c)$ を初期化する．6–14 行目において，CLA は，カラムのシナプス $c_i.y_k$ $(k = 1, 2, \ldots, n_{cy})$ を初期化する．**図 2.3** と **図 2.4** に従来 CLA におけるカラムのシナプスの初期化の概念図を示す．

従来の CLA は，パラメータとして半径 r と密度 d を使用する．各カラムのシナプス数

表 2.2: 擬似コードにおけるパラメータ

名前	説明
n	データビット長
n_c	カラム数
n_r	各カラムのセル数
ρ	カラムのシナプスにおける接続率
r	カラムのシナプス生成における半径
d	カラムのシナプス生成における密度
n_{cy}	カラムのシナプス数
n_{ac}	活性状態となるカラム数
p^{bmp}	バンプアップされる永続値量
p_c^+, p_c^-	カラムのシナプスにおける永続値の増減量
θ_c	カラムのシナプスにおける接続閾値
p_r^+, p_r^-	セルのシナプスにおける永続値の増減量
θ_r	セルのシナプスにおける接続閾値
κ^{bmp}	バンプアップにおける係数
κ^{bst}	ブーストにおける係数
n_p	予測状態となるために必要な接続シナプス数
p^c	カラムのシナプスにおける初期永続値
w	適応的なシナプス配置における時点の間隔
Θ_l, Θ_h	適応的なシナプス配置における活性頻度の下限と上限
ol^{min}	活性状態となるカラムのオーバーラップ値の最小値

n_{cy} は，以下の式で計算される．

$$n_{cy} = \lfloor d \cdot (2r + 1) + 0.5 \rfloor . \tag{2.1}$$

各カラム c_i において，まず，以下の式を用いてカラムのシナプスを配置するための中心ビット位置 $m_i \in \{1, 2, \ldots, n\}$ を計算する．

$$m_i = \left\lceil (i - 0.5) \cdot \frac{n}{n_c} \right\rceil . \tag{2.2}$$

その後，中心ビット位置 m_i から半径 r 内のビットにおいて，密度 d でシナプスを配置する．具体的には，8 行目で，候補ビットのインデックス集合 $\mathcal{L}_i$ を選択する．ここで，候補

Algorithm 2 初期化

1: カラムの初期化
2: **for** $i \leftarrow 1, 2, \ldots, n_c$ **do**
3: $c_i.bst \leftarrow 1.0$ ▷ ブースト値
4: $c_i.af \leftarrow 0.0$ ▷ 活性頻度
5: カラムのシナプスの初期化
6: $n_{cy} \leftarrow \lfloor d \cdot (2r+1) + 0.5 \rfloor$ ▷ カラムのシナプス数
7: $m_i \leftarrow \lceil (i - 0.5) \cdot n / n_c \rceil$ ▷ 中心ビット位置
8: $\mathcal{L}_i \leftarrow \{m_i - r, \ldots, m_i, \ldots, m_i + r\}$ ▷ 候補データビット集合
9: **for** $k \leftarrow 1, 2, \ldots, n_{cy}$ **do**
10: $l \leftarrow$ Randomly pop an index from candidate data bit indices $\mathcal{L}_i$
11: $c_i.y_k.a \leftarrow x_l$ ▷ データビット x_l とシナプス $c_i.y_k$ を紐づける
12: $c_i.y_k.p \leftarrow \begin{cases} \text{rand } [\theta_c, 1), \text{ if rand } (0,1) \geq \rho, \\ \text{rand } (0, \theta_c), \text{ otherwise,} \end{cases}$ ▷ 永続値
13: **end for**
14: **end for**
15: セルの初期化
16: **for** $i \leftarrow 1, 2, \ldots, n_c$ **do**
17: **for** $j \leftarrow 1, 2, \ldots, n_r$ **do**
18: $r_{i,j}.st \leftarrow$ Normal
19: **end for**
20: **end for**

ビットのインデックス数 $|\mathcal{L}_i|$ はカラムのシナプス数 n_{cy} より大きいことに注意する．そして，$\mathcal{L}_i$ からビットのインデックスをランダムに n_{cy} 個選択してシナプスをそれぞれのビットに配置する．

各シナプス $c_i.y_k$ $(k = 1, 2, \ldots, n_{cy})$ は，接続か切断かの状態を決める永続値 $c_i.y_k.p$ を持つ．接続閾値 θ_c に基づき，$c_i.y_k.p \geq \theta_c$ のときにシナプスは接続され，$c_i.y_k.p < \theta_c$ のときにシナプスは切断される．12 行目において，初期永続値 $c_i.y_k.p$ が設定される．従来の CLA は，パラメータとして接続率 ρ を使用し，ρ の確率でランダム値 $[\theta_c, 1)$，$1 - \rho$ の確率でランダム値 $[0, \theta_c)$ に永続値を初期化する．

16–20 行目において，各カラム c_i $(i = 1, 2, \ldots, n_c)$ 内のセル $r_{i,j}$ $(j = 1, 2, \ldots, n_r)$ を初期化する．

2.4　エンコード

Algorithm 1 の 4 行目におけるエンコードの手順を以下に示す．

各時点 t において，実数の入力値 $X(t) \in [X^{\min}, X^{\max}]$ を受け取ることを仮定する．図

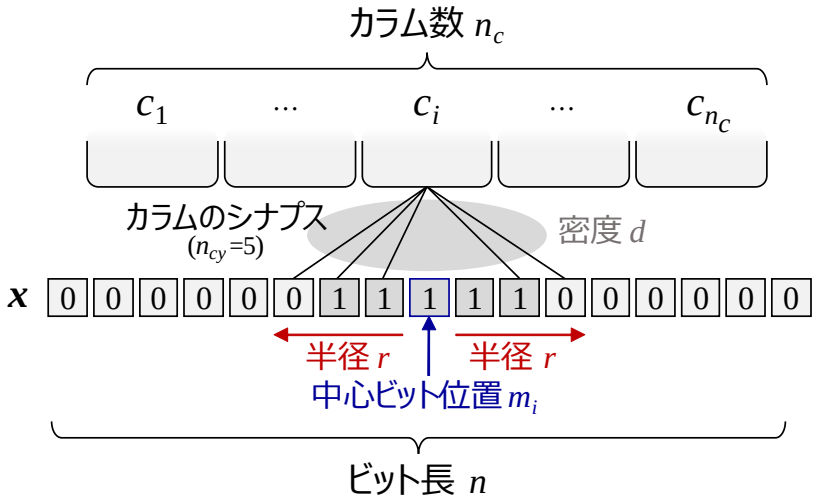


図 2.3: カラムのシナプス配置

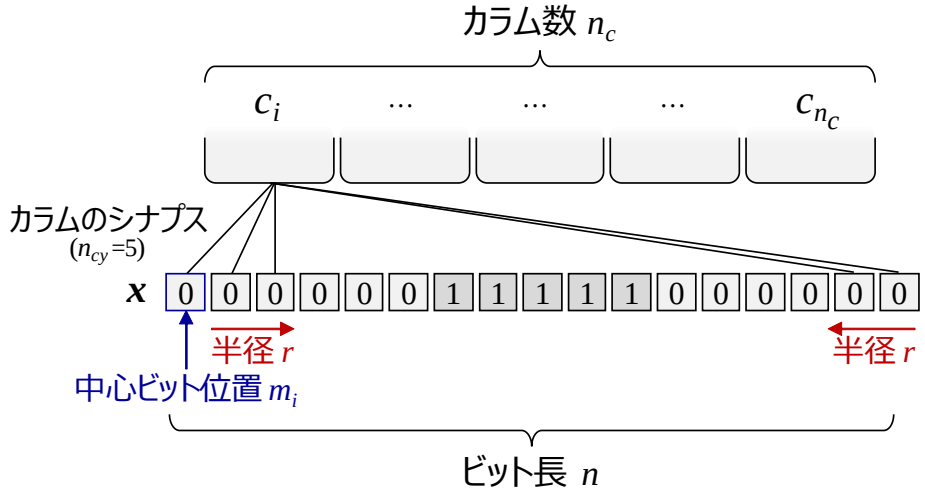


図 2.4: 中心ビット位置の決定

2.5 に示すように，CLA は，実数値 $X(t)$ をビット列 $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$ に変換する．実数値 $X(t)$ は，チャンクビットの位置に変換される．例えば，全ビット長 n に対して，連続した δ ビットが 1 の値となる．チャンクビット長 δ は，パラメータである．図 **2.5** における入力ビット列は，全ビット長 $n = 12$，チャンクビット長 $\delta = 5$ の例

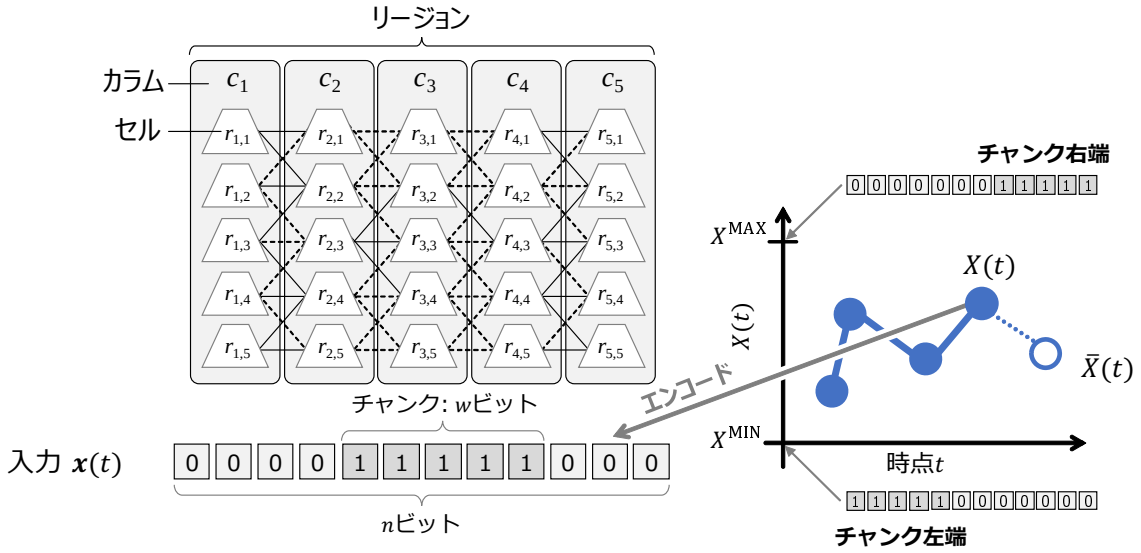


図 2.5: 入力のエンコード

である．各データビット x_i $(i = 1, 2, \ldots, n)$ は，以下の式で決定される．

$$x_i = \begin{cases} 1, & \text{if } h(X(t)) < i \leq h(X(t)) + \delta, \\ 0, & \text{otherwise}, \end{cases} \qquad (i = 1, 2, \ldots, n) \tag{2.3}$$

ここで，$h(X(t)) = 0.5 + \frac{(X(t) - X^{\min}) \cdot (n - \delta)}{X^{\max} - X^{\min}}$ である．これにより，入力値が最小値 $X^{\min}$ のときにチャンクの位置が左端，最大値 $X^{\max}$ のときにチャンクの位置が右端となるように変換される．

2.5　空間プーリング

Algorithm 1 の 5 行目で実行される空間プーリングを **Algorithm 3** に示す．空間プーリングの役割は，入力データビット列 $\boldsymbol{x}$ を，CLA 予測器の内部データ表現として，活性状態のカラム群のパターンに変換することである．**Algorithm 3** について，以下で説明する．

1–15 行目において，活性状態にするカラムの部分集合を選択する．まず，4 行目において，各カラム c_i について，接続状態のシナプスと紐づけられた入力ビット列 $\boldsymbol{x}$ が 1 であるビット数を，オーバーラップ値 $c_i.ol$ として計数する．次に，5 行目において，オーバーラップ値 $c_i.ol$ および後述するブースト値 $c_i.bst$ に基づき，活性化スコア $c_i.as$ を計

Algorithm 3 空間プーリング

1: カラムの活性状態化
2: **for** $i \leftarrow 1, 2, \ldots, n_c$ **do**
3: $c_i.st \leftarrow$ Normal
4: $c_i.ol \leftarrow |\{k \in \{1, 2, \ldots, n_{cy}\} \mid c_i.y_k.a = 1\}|$ ▷ オーバーラップ値
5: $c_i.as \leftarrow c_i.ol \times c_i.bst$ ▷ 活性化スコア
6: **end for**
7: **for** $i \leftarrow 1, 2, \ldots, n_c$ **do**
8: **if** $c_i.as$ is within the top n_{ac} scores among all columns **then**
9: $c_i.st \leftarrow$ Active
10: カラムのシナプスの更新
11: **for** $k \leftarrow 1, 2, \ldots, n_{cy}$ **do**
12: $c_i.y_k.p \leftarrow \begin{cases} c_i.y_k.p + p_c^+, & \text{if } c_i.y_k.a = 1, \\ c_i.y_k.p - p_c^-, & \text{otherwise } (c_i.y_k.a = 0). \end{cases}$
13: **end for**
14: **end if**
15: **end for**
16: バンプアップ
17: **for** $i \leftarrow 1, 2, \ldots, n_c$ **do**
18: **if** $c_i.zf > \theta^{bmp}$ **then**
19: **for** $k \leftarrow 1, 2, \ldots, n_{cy}$ **do**
20: $c_i.y_k.p \leftarrow c_i.y_k.p + p^{bmp}$
21: **end for**
22: **end if**
23: **end for**
24:
25: **for** $i \leftarrow 1, 2, \ldots, n_c$ **do**
26: $c_i.af \leftarrow \begin{cases} \frac{1}{t} \cdot (c_i.af \cdot (t-1) + 1), & \text{if } c_i.st = \text{Active}, \\ \frac{1}{t} \cdot (c_i.af \cdot (t-1)), & \text{otherwise} \end{cases}$
27: $c_i.bst \leftarrow \exp\left(\kappa^{bst} \cdot \left(\frac{\max_{j=1}^{n_c} c_j.af}{n_c} - c_i.af\right)\right)$
28: **end for**

算する．9 行目において，活性化スコア $c_i.as$ の降順に n_{ac} 本のカラムを選択して活性状態にする．**図 2.6** では，5 本のカラムのうち，赤字で示すオーバーラップ値が高い 2 本のカラムが活性状態になる例を示す．次に，10–13 行目において活性状態になったカラムにおいて，シナプスの永続値を更新する．**図 2.7** では，赤い 2 本の活性状態のカラムについて，入力データビットと紐づくシナプスを更新するところを例示した．具体的には，各活性状態のカラム c_i について，1 のデータビットと紐づけられたシナプスの永続値を p_c^+ 増加し，0 のデータビットと紐づけられたシナプスの永続値を p_c^- 減少する．その後，永続値が $c_i.y_k.p \geq \theta_c$ となったカラムのシナプス $c_i.y_k$ を接続状態にし，$c_i.y_k.p < \theta_c$ となったシナプスを切断状態にする．**図 2.7** では，赤い 2 本の活性状態のカラムについて，1 の

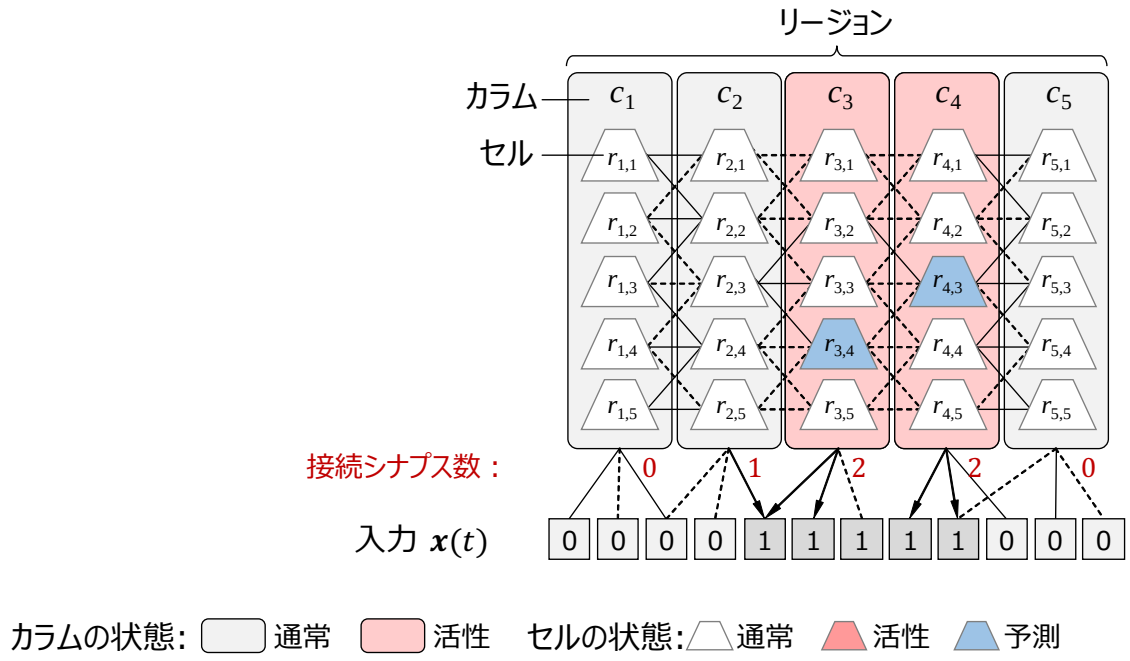


図 2.6: カラムの活性状態化

データビットと紐づく赤いシナプスの永続値が増加し，0 のデータビットと紐づく黒いシナプスの永続値が減少することになる．これにより，同じ値が入力された際に，同一のカラムが活性状態となるように学習する．

従来の CLA には，カラムの活性状態化を促進する二つの機構がある．バンプアップとブーストという．各カラム c_i が活性状態になる優先度が，5 行目においてオーバーラップ値 $c_i.ol$ とブースト値 $c_i.bst$ を掛け合わせて算出する活性化スコア $c_i.as$ で決定されることは，前述の通りである．

バンプアップは，17–23 行目において，オーバーラップ値 $c_i.ol$ を増加させる処理である．これにより，活性化スコア $c_i.as$ を増加させ，活性状態への遷移を促す．バンプアップでは，各カラム c_i について，オーバーラップ値 $c_i.ol$ が 0 になる頻度 $c_i.zf$ を用いる．ゼロ頻度 $c_i.zf$ は，現時点までの全入力時点のうち，1 のデータビットと接続されたシナプスを持たない $(c_i.ol = 0)$ 時点数の割合である．この頻度が大きいほど，入力の内部表現化に貢献するシナプス，もしくは接続状態のシナプス，の両方を持たないカラムであるといえる．バンプアップの閾値 θ^{bmp} は，ゼロ頻度 $c_i.zf$ $(i = 1, 2, \ldots, n_c)$ を用いた以下

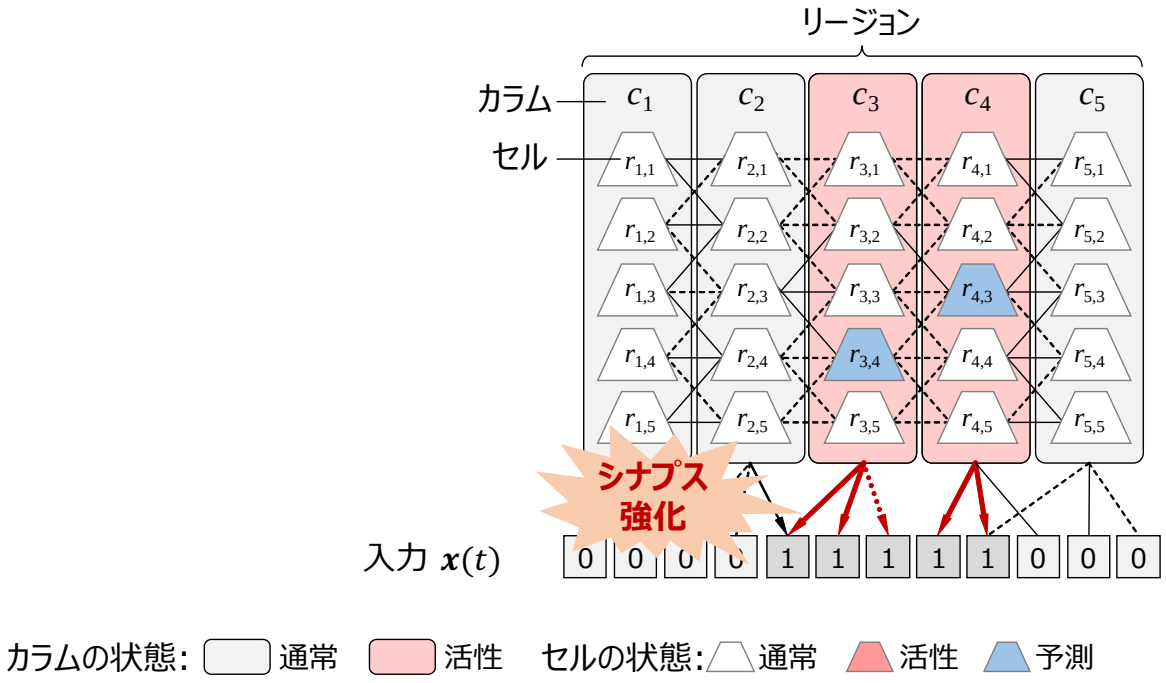


図 2.7: カラムのシナプスの更新

の式で算出する．

$$\theta^{bmp} = \kappa^{bmp} \cdot \max_{i=1}^{n_c} c_i.zf \tag{2.4}$$

ここで，κ^{bmp} は係数パラメータである．20 行目において，各カラム c_i が，$c_i.zf > \theta^{bmp}$ であれば，カラム c_i の全シナプスの永続値 $c_i.y_k.p$ $(k = 1, 2, \ldots, n_{cy})$ を p^{bmp} 増加する．このように，バンプアップは，接続状態になるシナプス数を増加させることで，オーバーラップ値を増加させようとする．これが，活性化スコアの増加につながり，最終的に．活性状態への遷移を促すことになる．

ブーストは，25–28 行目においてブースト値 $c_i.bst$ を増加させることで，活性化スコア $c_i.as$ を増加させる．これにより，活性状態への遷移を促す．ブーストは，各カラム c_i について，現時点までの全入力時点のうち，活性状態 ($c_i.st$ =Active) になった時点数を表す活性頻度 $c_i.af$ を用いる．各時点において，活性頻度 $c_i.af$ を継続して更新するため，従来の CLA は，26 行目の式を用いる．ブースト値 $c_i.bst$ は，活性頻度 $c_i.af$

Algorithm 4 時間プーリング

```
1: セルの活性状態化
2: for i ← 1, 2, ..., n_c do
3:   if c_i.st = Active then
4:     for j ← 1, 2, ..., n_r do
5:       if r_{i,j}.st = Predictive then
6:         r_{i,j}.st ← Active
7:         Generate synapses from r_{i,j} to active cells at time t − 1
8:         セルのシナプスの更新
9:         for k ← 1, 2, ... do
10:          r_{i,j}.y_k.p ← Update performance value
11:        end for
12:      end if
13:    end for
14:    if Column c_i has no active cell then
15:      for j ← 1, 2, ..., n_r do
16:        r_{i,j}.st ← Active
17:      end for
18:      Generate synapses from a cell to active cells at time t − 1
19:    end if
20:  end if
21: end for
22: セルの予測状態化
23: for i ← 1, 2, ..., n_c do
24:   for j ← 1, 2, ..., n_r do
25:     if r_{i,j} has more than n_p synapses connected to active cells then
26:       r_{i,j}.st ← Predictive
27:     end if
28:   end for
29: end for
```

$(i = 1, 2, \ldots, n_c)$ を用いた以下の式で算出する．

$$c_i.bst = \exp\left(\kappa^{bst} \cdot \left(\frac{\max_{j=1}^{n_c} c_j.af}{n_c} - c_i.af\right)\right), \tag{2.5}$$

ここで，κ^{bst} は係数パラメータである．これにより，ブーストは活性頻度 $c_i.af$ が低いカラムのブースト値 $c_i.bst$ を増加させる．これが活性化スコアの増加につながり，最終的に活性状態への遷移を促すことになる．

2.6　時間プーリング

Algorithm 1 の 6 行目から実行される時間プーリングを **Algorithm 4** に示す．時間プーリングには，二つの役割がある．一つ目は，入力データの文脈をセルの活性状態のパ

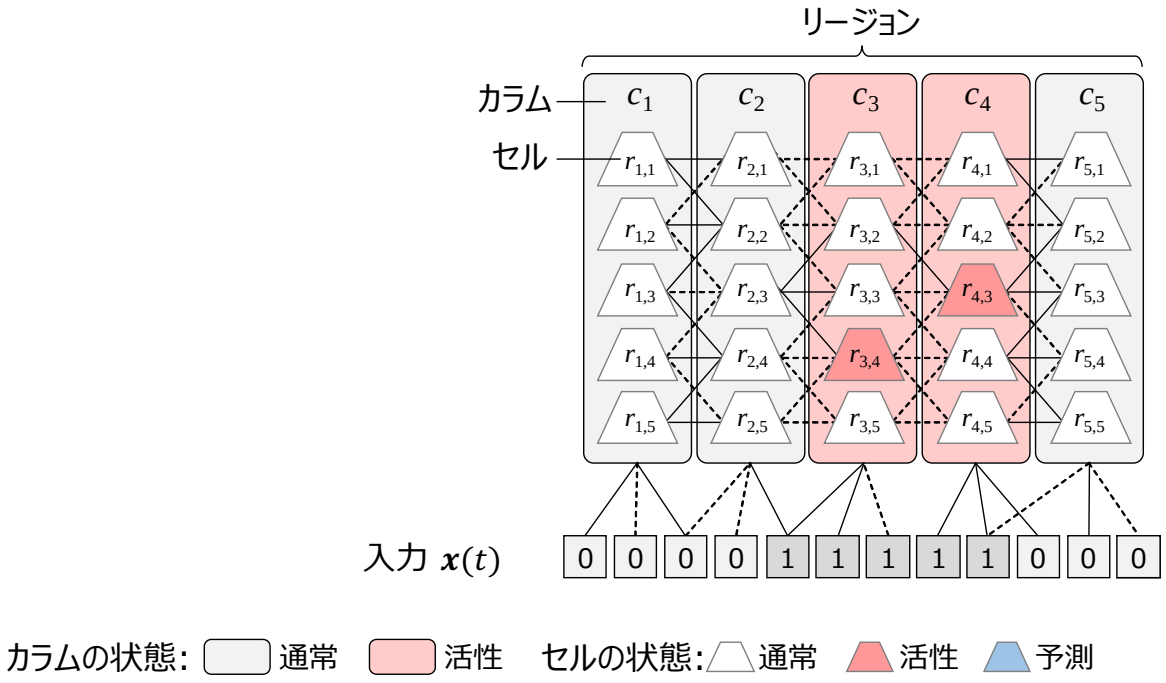


図 2.8: セルの活性状態化

ターンで表現することである．二つ目は，次時点 $t+1$ で入力されるデータをセルの予測状態のパターンとして表現することである．**Algorithm 4** について以下で説明する．

まず，2–21 行目において，時間プーリングは，活性状態のカラム群から，活性状態にするセルを選択する．具体的には，$c_i.st$ = Active の活性状態のカラム c_i において，前時点 $t-1$ で予測状態になったセルを活性状態にする．**図 2.8** では，カラム c_3 と c_4 が活性状態であり ($c_3.st$ =Active，$c_4.st$ =Active)，カラム c_3 内のセル $r_{3,4}$ と，カラム c_4 内のセル $r_{4,3}$ が前時点 $t-1$ において予測状態だった例を示す．**図 2.8** では，これらのセル $r_{3,4}$ と $r_{4,3}$ が活性状態になる例を示している．セルの予測状態から活性状態への遷移は，予測の成功を意味する．活性状態のカラム c_i に予測状態のセルが存在しないとき (14 行目)，カラム c_i の全てのセルを活性状態にする．次に，8–11 行目において，予測状態から活性状態に遷移したセルのシナプスの永続値を更新する．前時点 $t-1$ において，活性状態のセルと紐づけられたシナプスの永続値を p_r^+ 増加させる．また，前時点 $t-1$ において，通常状態のセルと紐づけられたシナプスの永続値を p_r^- 減少させる．**図 2.9** の例では，予測状態から活性状態になったセル $r_{2,3}$ について，前時点 $t-1$ においてセル $r_{2,3}$ を

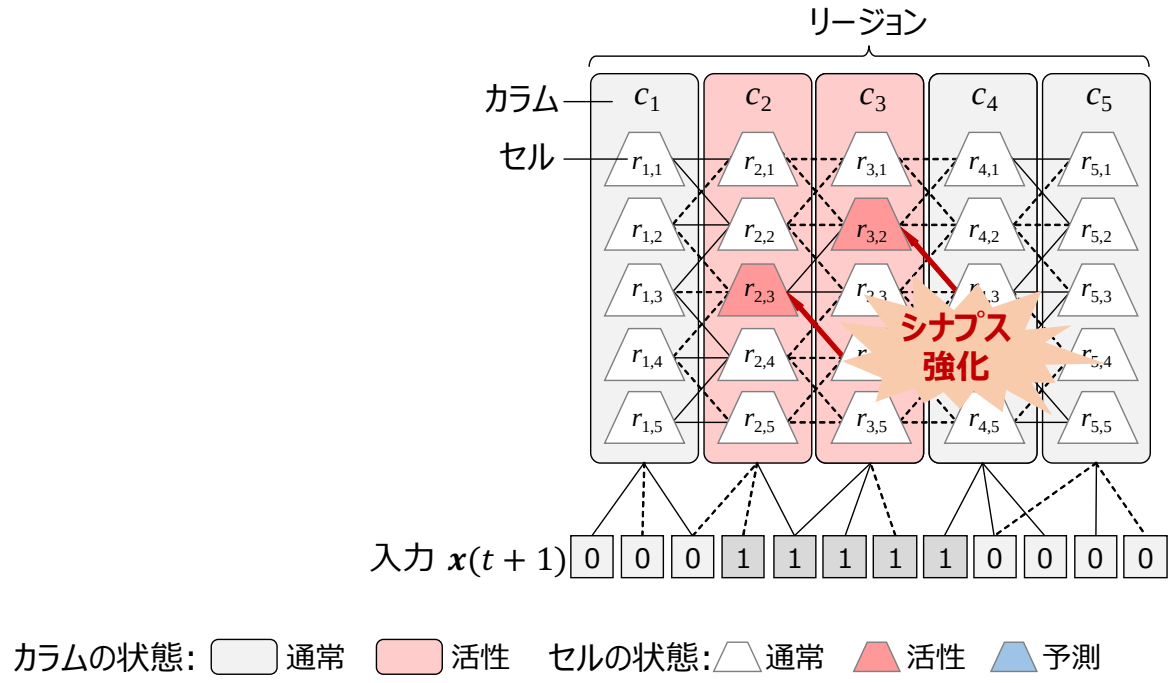


図 2.9: セルのシナプスの更新

予測状態にしたセル $r_{4,3}$ とのシナプスの永続値を高めている．$r_{3,2}$ についても，同様である．その後，セルのシナプス $r_{i,j}.y_k$ の永続値が $r_{i,j}.y_k.p \geq \theta_r$ であれば接続状態になり，$r_{i,j}.y_k.p < \theta_r$, であれば切断状態になる．ここで θ_r はセルのシナプスの接続閾値である．そして，7 行目および 18 行目において，活性状態となったセルから前時点 $t-1$ において活性状態だったセルにシナプスを生成する．**図 2.10** にセル $r_{1,5}$ と $r_{4,2}$ のシナプスの配置例を示した．なお，シナプスが配置される範囲に空間的な制限は無い．各セルは，全てのセルにシナプスを配置できる．これにより，活性状態となったセルが，同じ入力データの文脈で予測状態となるようにシナプスを構築する．

次に，23–29 行目において，予測状態にするセルを選択する．これは，現状の活性状態のセルのパターンから，予測状態のセルのパターンを作る処理である．予測状態のセルのパターンは，次時点 $t+1$ の予測値を予測器の内部で表現する．具体的には，活性状態のセルと n_p 本以上の接続されたシナプスを持つセルを予測状態にする．この動作の例を**図 2.11** に示す．セル $r_{3,4}$ と $r_{4,3}$ が活性状態になっている．これらのセルと接続状態のシナプスを有する $r_{2,3}$ と $r_{3,2}$ が，予測状態になる例を示している．

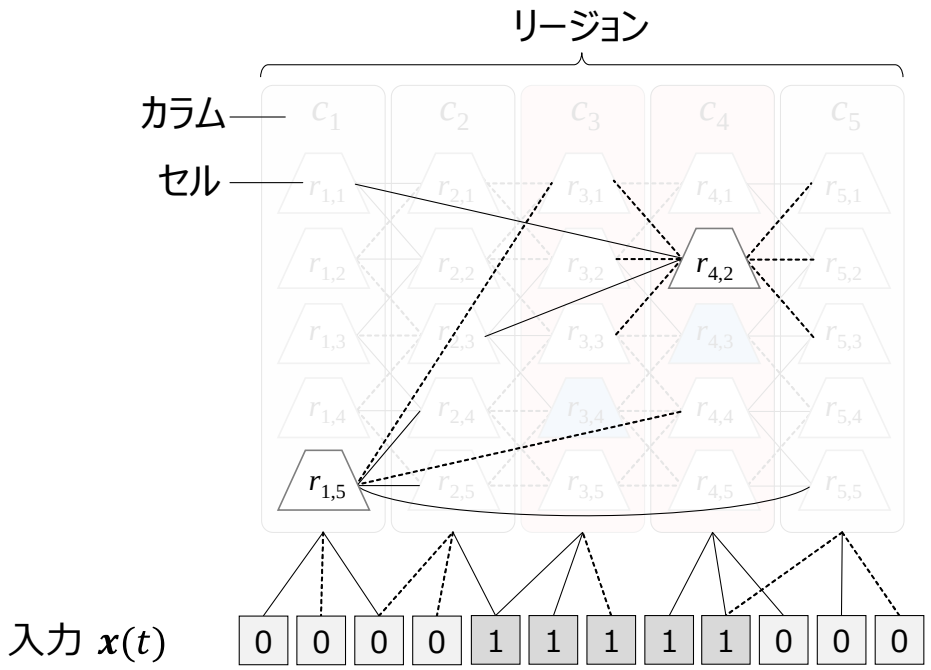


図 2.10: セルのシナプスの配置例

2.7 デコード

予測状態のセル群のパターンは，次時点の予測値を表現する．予測器全体のセル群に対して，わずかな予測状態のセル群の組み合わせで予測値を表現する．これは，人間が直接理解できない表現形式である．**Algorithm 1** の 7 行目は，予測状態のセル群のパターンを，入力ビット列と同じ形式にデコードする処理である．従来 CLA におけるデコーダとして，入力ビット列と予測セルパターンの関係を学習する疎分散表現分類器 (sparse distributed representations classifier, 以下 SDRC)[62] を使用する．予測器内部にまばらに存在する予測状態のセル群に対応する実数値を見つけるため，「Sparse Distributed Representations Classifier」と呼ばれる．**図 2.12** に SDRC の概要図を示す．

SDRC は，現時点 t までに入力された入力ビット列の集合 $\mathcal{X} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \dots\}$ を保持する．$\mathcal{X}$ の初期サイズは 0 である．例えば，**図 2.12** に示す $\boldsymbol{x}^1$ は，現時点 t までに入力されたある入力ビット列である．もし，同じ入力ビット列が入力された場合は，$\mathcal{X}$ に追加しない．新しい入力ビット列のみが $\mathcal{X}$ に追加さる．$\mathcal{X}$ に含まれるビット列が重複すること

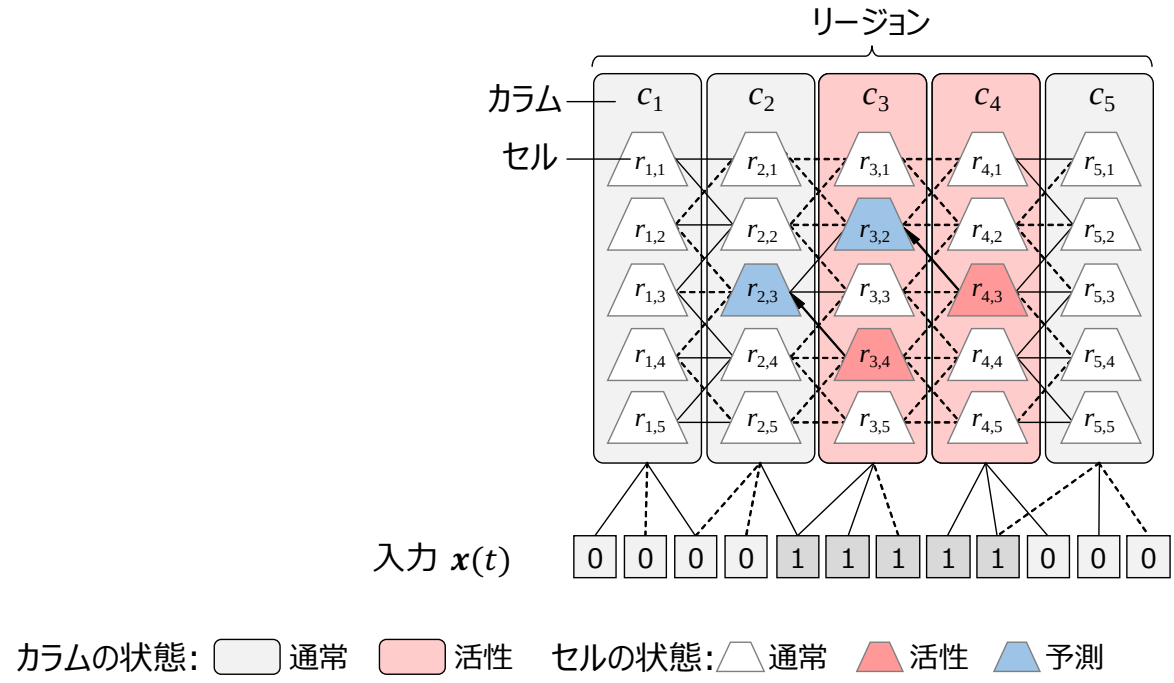


図 2.11: セルの予測状態化

はない．そのため，デコーダが出力しうる予測ビット列の数は，その時点までに入力された入力ビット列と同数であり，有限個となる．

各時点 t において，SDRC は予測セルパターンを $\mathcal{P} = \{P_1, P_2, \ldots, P_{n_c \times n_r}\}$ として表現する．この表現では，**図 2.12** に示すように，全てのカラム内のセルを $r_{1,1}, \ldots, r_{1,n_r}, r_{2,1}, \ldots, r_{2,n_r}, \ldots, r_{n_c,1}, \ldots, r_{n_c,n_r}$ のように一列に並べ，予測状態のセルを 1，他のセルを 0 として $\mathcal{P}$ を構成する．例えば**図 2.12** において，セル $r_{1,1}$ は予測状態であるため，対応する要素が，$P_1 = 1$ になる．同様に，セル $r_{1,2}$ は，予測状態ではないため，対応する要素が $P_2 = 0$ になる．セル $r_{1,3}$ は，予測状態であるため，対応する要素が $P_3 = 1$ になるといった具合である．

SDRC は，入力ビット列の集合と予測セルパターンの関係を構築するために，重み行列 W を使用する．W のサイズは $|\mathcal{X}| \times |\mathcal{P}|$ である．$|\mathcal{X}|$ は，$\mathcal{X}$ におけるビット列の数である．$|\mathcal{P}|$ $(=n_c \times n_r)$ は，CLA の予測器におけるセルの合計数を表す．時点 t の予測セルパターン $\mathcal{P}$ において，活性化レベル a^j を各入力ビット列 $\boldsymbol{x}^j$ $(j = 1, 2, \ldots, |\mathcal{X}|)$ に対して

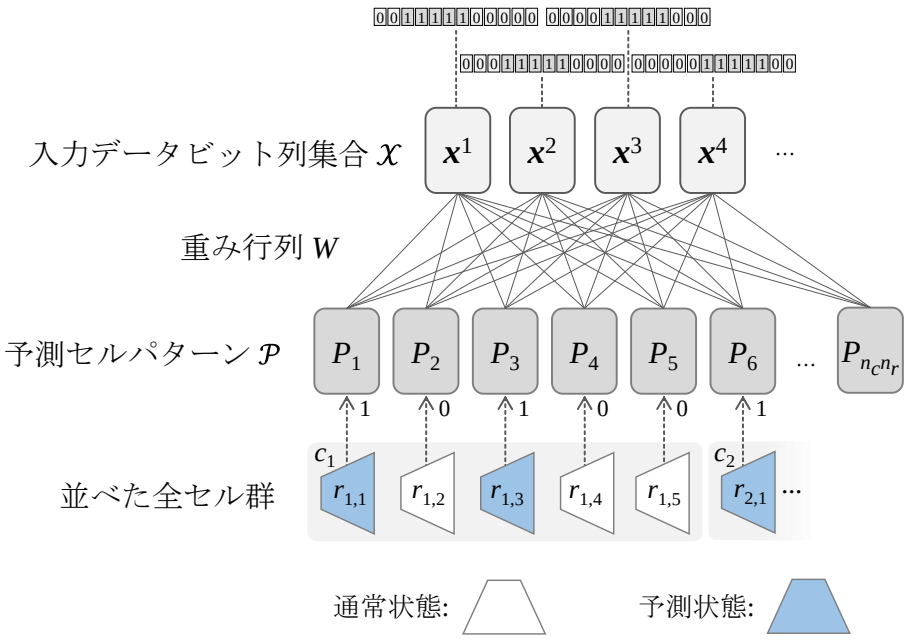


図 2.12: SDRC デコーダ

以下の式で計算する．

$$a^j = \sum_{i=1}^{|\mathcal{P}|} W_{i,j} \cdot P_i \quad (j = 1, 2, \ldots, |\mathcal{X}|) \tag{2.6}$$

活性化レベルが最も高いビット列 $\overline{\boldsymbol{x}} = \mathrm{argmax}_{\boldsymbol{x}^j \in \mathcal{X}}\, a^j$ を，デコードした結果の予測ビット列として出力する．

次に，ビット列集合 $\mathcal{X}$ を，次時点 $t+1$ の入力ビット列 $\boldsymbol{x}(t+1)$ を用いて更新する．次時点 $t+1$ の $\boldsymbol{x}(t+1)$ を 1，それ以外を 0 とする目標分布ベクトル $\boldsymbol{z} = (z_1, z_2, \ldots, z_{|\mathcal{X}|})$ を以下の式で生成する．

$$z_j = \begin{cases} 1, & \text{if } \boldsymbol{x}^j = \boldsymbol{x}(t+1), \\ 0, & \text{otherwise}, \end{cases} \quad (j = 1, 2, \ldots, |\mathcal{X}|). \tag{2.7}$$

その後，重み行列 W を目標分布ベクトル $\boldsymbol{z}$ を用いた以下の式で更新する．

$$W_{i,j} = W_{i,j} + \alpha \left\{ z_j - \frac{e^{a^j}}{\sum_{k=1}^{|\mathcal{X}|} e^{a^k}} \right\} \ (j = 1, 2, \ldots, |\mathcal{X}|), \tag{2.8}$$

ここで，α は学習パラメータである．この更新により，ある時点 t での CLA で予測状態となったセルと，次時点 $t+1$ の入力ビット列 $\boldsymbol{x}(t+1)$ との重みを大きくする．その結果，同じセル群が予測状態となったとき，$\boldsymbol{x}(t+1)$ を予測値として出力することができる．

第 3 章

カラムのシナプスの適応配置

従来の CLA は，入力データビット列に対して，均一にカラムのシナプスを配置する．偏った値が入力されると，値の差異を精緻に表現できず，予測精度が悪くなる．本章では，様々なデータへの対応を目的として，入力値の偏りにあわせて，適応的にシナプスを配置する方法を提案する．CLA の予測器において，本章で取り組む部分を**図 3.1** に青色で示した．本章は，CLA におけるカラムのシナプスの取り扱いに注目し，CLA-AC (CLA with Adaptive Column-synapses) を提案する．

3.1 概要

本章では，従来の CLA におけるカラムのシナプスの問題点に焦点を当てる．

3.1.1 シードによる CLA の不安定性

従来の CLA はシードごとに予測精度に差がある．これは，従来の CLA におけるカラムのシナプスの初期設定が，アルゴリズムの振る舞いを不安定にするためである．ここでは，カラムのシナプスの初期設定の問題点を三つ述べる．

(i) 確率的な配置： 従来の CLA は，カラムのシナプスを確率的に配置する．具体的には，各カラム c_i について，中心ビット位置 m_i から半径 r ビットの範囲で，密度 d になるように n_{cy} 本のシナプスをランダムに配置する．例を**図 3.2 (a)** に示す．カラム c_i のシナプスの配置先は，中心ビット位置 m_i の半径 $r = 3$ ビット内の 7 ビットから，ラン

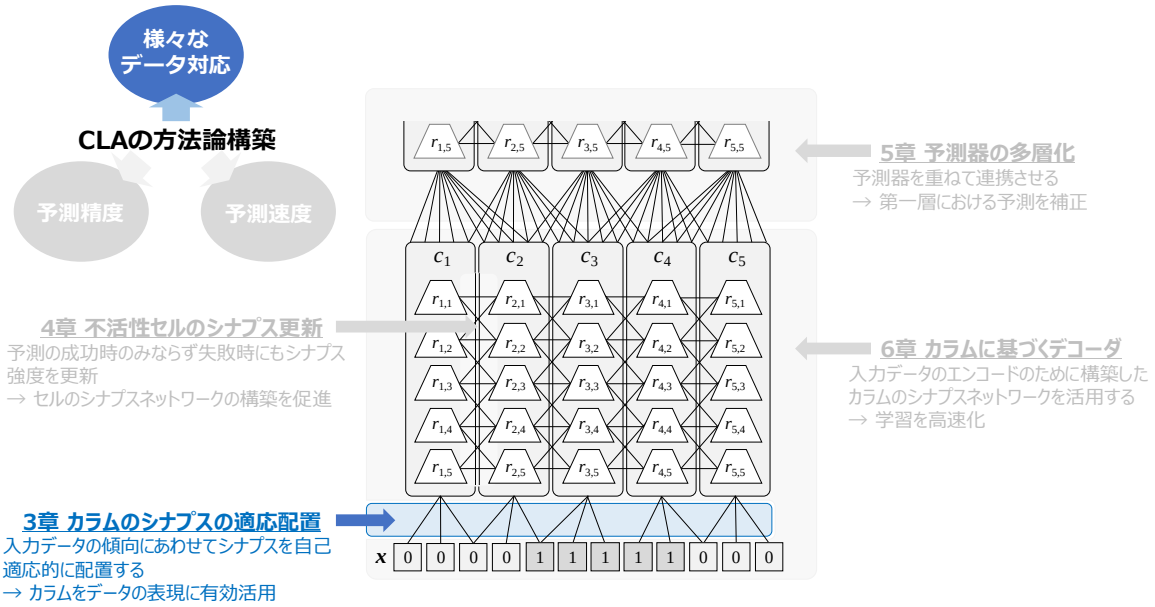
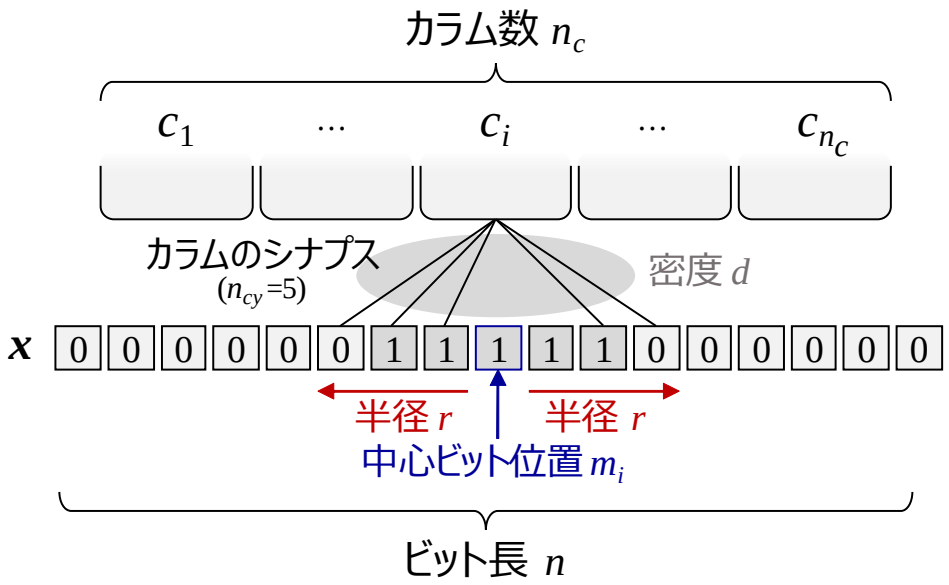


図 3.1: 3 章の位置づけ

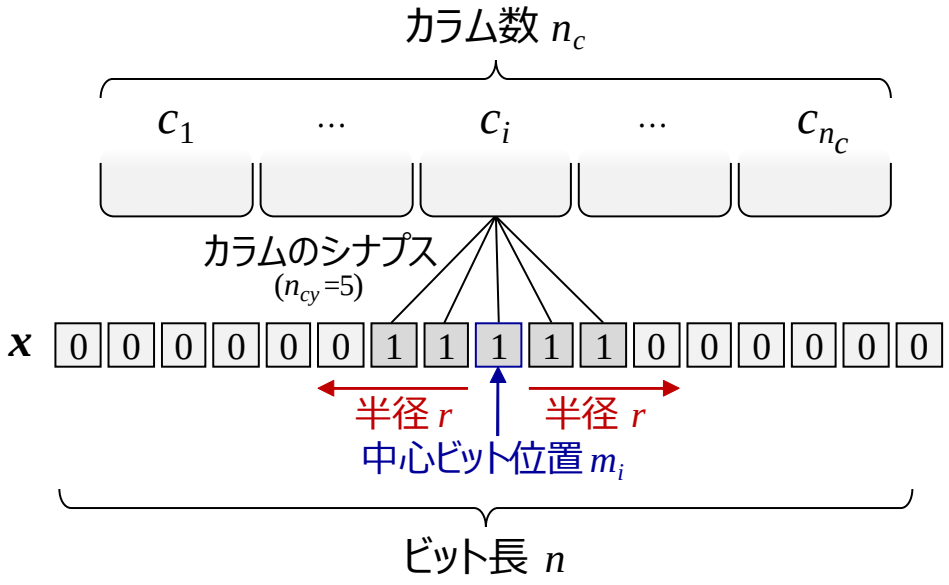
ダムに $n_{cy} = 5$ ビット選ばれる．2 ビットには，シナプスが配置されないことがわかる．このように，従来の CLA では，シナプスの配置が決定論的ではない．その結果，アルゴリズムの挙動と結果が実行ごとに変化する．

(ii) ランダムな初期永続値： 従来の CLA は，各シナプスの永続値の初期値をランダム値にする．このランダム性によって，アルゴリズムの挙動と結果が，実行ごとに変化する．また，永続値をランダムにすると，シナプスごとに接続閾値 θ_c との差異がばらつくことになる．永続値の初期値が，接続閾値 θ_c に近ければ，永続値の更新のたびに，シナプスの接続状態と切断状態が頻繁に切り替わることがある．一方，永続値の初期値が，接続閾値 θ_c から遠ければ，シナプスの接続状態と切断状態を切り替えるために，永続値の更新を多数回繰り返す必要がある．これがカラムのシナプスネットワークの形成に時間がかかる原因になり，予測精度の悪さの要因になる．

(iii) シナプスの分離： 従来の CLA では，端のカラムが入力ビット列の両端にシナプスを分離して配置する．例を**図 3.3 (a)** に示す．左端のカラム c_i が，入力ビット列の左端と右端にシナプスを分離して配置することがわかる．両端の入力ビットは，全く異なる入力値 $X(t)$ に対応している．そのため，このように分離したシナプス配置のカラムのオーバーラップ値 $c_i.ol$ は，他のカラムと比べて低くなる．その結果，端のカラムは，活



(a) 従来の CLA (確率的なシナプス配置)

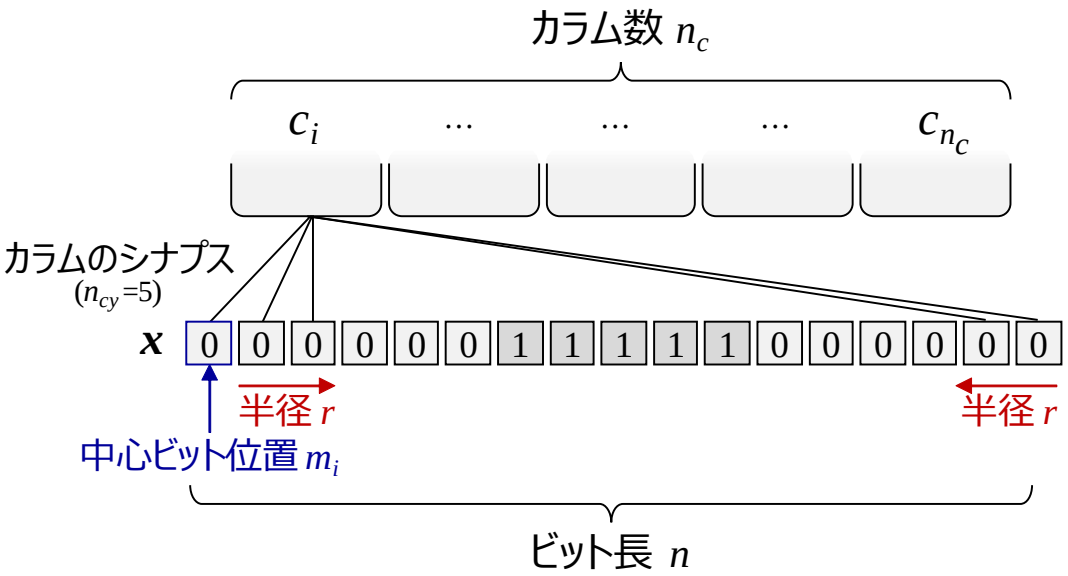


(b) 提案する CLA (決定論的なシナプス配置)

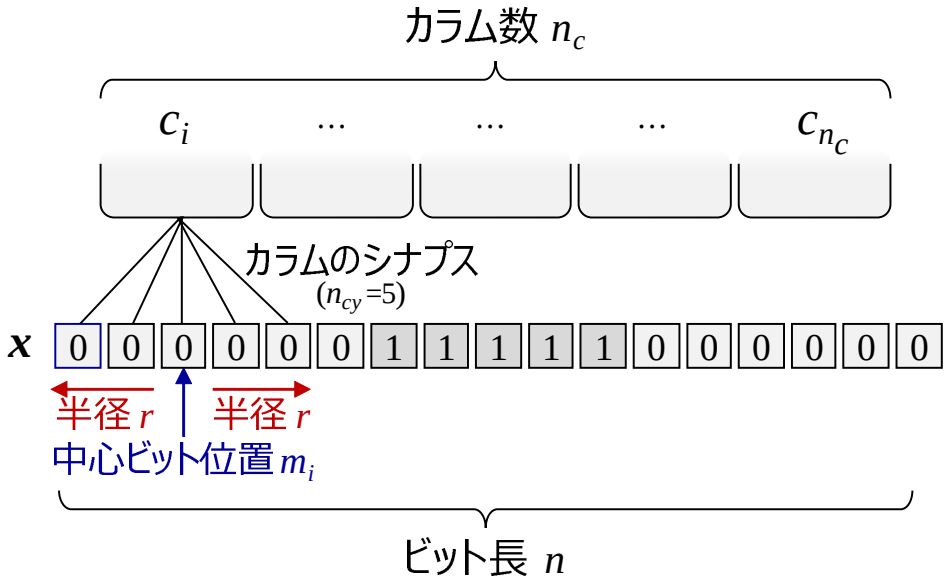
図 3.2: 確率的および決定論的なシナプス配置

性状態に至ることが困難になり，入力ビット列の内部表現に利用されづらくなる．これが，予測精度の悪さの要因になる．従来の CLA が，中心ビット位置 m_i を入力ビット列の左端や右端に分離して配置することが原因である．

これらの初期設定は，活性状態になるカラムの位置を空間的に分散させる効果がある．



(a) 従来の CLA (分離されたシナプス配置)



(b) 提案する CLA (集約されたシナプス配置)

図 3.3: 分離および集約されたシナプス配置

これにより，障害などにより一部のカラムが破損したときでも予測性能を維持しやすい [63]．これは，人間の脳を模倣して CLA が設計されたことによる．カラムがハードウェアとして実装される場合は，破損に対して予測を維持する効果が，これらの初期設定に期待できるが，予測器をコンピュータ上でソフトウェア実装する本研究の前提では，破損を

想定しないため，この効果は不要である．むしろ，これらの初期設定は，アルゴリズムの動作に不安定性をもたらすため，本研究では決定論的な手段を導入する．

3.1.2 入力の偏りによって使用されないカラム

CLA において，入力値の範囲はあらかじめ設定する必要がある．また，実際に予測する値は少なからず偏りがあるケースがほとんどを占める．一方，従来の CLA は，カラムのシナプスを初期化によって固定して配置する．従来の CLA の実行中にカラムのシナプス配置が変わることはない．その結果，入力されない値にシナプスを固定配置したカラム群は，活性状態にならない．すなわち，これらのカラム群は，入力データを予測器の内部表現にするために利用されない．これにより，用意した予測器のサイズで想定した性能よりも予測精度が悪化する．入力データの偏りにあわせてカラムのシナプスを適応的に再配置できれば，入力データの内部表現化に活用されないカラムを減らすことができる．これにより，入力データを精緻に内部表現でき，予測精度が向上することが期待される．

3.2 方法

3.2.1 カラムのシナプスの初期化

3.1.1 節の (i)–(iii) で述べた従来の CLA におけるカラムのシナプスの初期化の問題に対処するため，提案する CLA では，以下について変更する．本章で提案する CLA-AC は，従来の **Algorithm 2** を **Algorithm 5** で代替する．従来の CLA からの変更点は，青色で示した．

(i) 決定論的な配置： 従来の CLA は，密度パラメータ d を用いて確率的にシナプスを配置する．その結果，**図 3.2 (a)** に示すように，半径 r ビット内でシナプスを配置するデータビットと配置しないデータビットが生じる．これに対して，提案する CLA は，決定論的にシナプスを配置する．**図 3.2 (b)** に示すように，半径 r 内の全てのビットに対してシナプスを配置する．

Algorithm 5 初期化 (提案する CLA)

```
1: カラムの初期化
2: for i ← 1, 2, ..., n_c do
3:   c_i.bst ← 1.0                                   ▷ ブースト値
4:   c_i.af ← 0.0                                    ▷ 活性頻度
5:   カラムのシナプスの初期化
6:   r ← ⌊(n_cy − 1)/2⌋                             ▷ 半径ビット数
7:   m_i ← ⌈(i − 0.5) · (n − 2r)/n_c + r⌉           ▷ 中心ビット位置
8:   L_i ← {m_i − r, ..., m_i, ..., m_i + r}       ▷ データビット集合
9:   for k ← 1, 2, ..., n_cy do
10:    l ← Pop an index from data bit indices L_i
11:    c_i.y_k.a ← x_l          ▷ データビット x_l とシナプス c_i.y_k を紐づける
12:    c_i.y_k.p ← p^c                               ▷ 永続値の固定
13:  end for
14: end for
15: セルの初期化
16: for i ← 1, 2, ..., n_c do
17:  for j ← 1, 2, ..., n_r do
18:    r_{i,j}.st ← Normal
19:  end for
20: end for
```

提案する CLA は，以下の式で半径 r を求める．

$$r = \left\lfloor \frac{n_{cy} - 1}{2} \right\rfloor \tag{3.1}$$

ここで，n_{cy} は各カラムのシナプス数である．**Algorithm 5** において，半径 r は 6 行目で算出する．次に，各カラム c_i について，8 行目において，入力データビットを選択する．10–11 行目において，選択した入力データビットにシナプスを配置する．この決定論的なシナプス配置は，アルゴリズムの振る舞いの安定性を高める．従来の CLA には，半径 r と密度 d の二つがパラメータとして必要だったが，提案する CLA には，シナプス数 n_{cy} のみがパラメータとして必要となる．

(ii) 初期永続値の固定： 従来の CLA は，**Algorithm 2** の 12 行目で示すように，シナプスの永続値をランダム値で初期化する．提案する CLA は，**Algorithm 5** の 12 行目に示すように，シナプスの永続値を固定値 p^c で初期化する．これにより，シナプス接続のランダム性を回避し，予測精度の安定性を向上させる．提案する CLA には，パラメータとして初期永続値 p^c が必要だが，従来の CLA に必要なシナプスの初期接続率 ρ が不要になる．そのため，必要とするパラメータ数は変わらない．

(iii) シナプスの集約： 従来の CLA は，**図 3.3 (a)** に示すように，端のカラムのシナプ

スが，入力データビット列の両端に分離して配置される．提案する CLA は，**図 3.3 (b)** に示すように，各カラムのシナプスを集約して配置する．具体的には，各カラム c_i のシナプス配置の中心ビット位置 m_i を変更する．提案する CLA は，入力データビット列の両端から半径 r ビットには，中心ビット位置 m_i を配置しないようにする．提案する CLA において，カラム c_i の中心ビット位置 m_i は，以下の式で算出する．

$$m_i = \left\lceil (i - 0.5) \cdot \frac{n - 2r}{n_c} + r \right\rceil . \tag{3.2}$$

3.2.2 シナプスの適応配置

3.2.2.1 概要

3.1.2 節で言及した従来の CLA においてシナプス配置が固定される問題に対応するため，入力データの傾向にあわせて動的にシナプスを配置する方法を提案する．提案するシナプスの適応配置を含む空間プーリングの擬似コードを **Algorithm 6** に示す．提案する CLA-AC は，従来の空間プーリング **Algorithm 3** を **Algorithm 6** で代替する．変更点は，青色で示した．シナプスを適応配置する **Algorithm 6** の 49 行目については，**Algorithm 7** を用いる．

3.2.2.2 カラムのシナプスの適応配置の例

提案するシナプスの適応配置の例を**図 3.4** に示す．**図 3.4 (a)** は，提案法によるシナプスの適応配置の前の状態である．**図 3.4 (b)** は，適応配置後の状態である．提案法は，直近の w 時点における各カラム c_i $(i = 1, 2, \ldots, n_c)$ の活性頻度 $c_i.af^w$ を活用する．まず，活性頻度が低いカラムを移動元カラムとして選択する．**図 3.4** では，青色の c_1，c_2，c_4 の 3 本のカラムが，移動元カラムとなる．次に，活性頻度が高いカラムを移動先カラムとして選択する．**図 3.4** では，赤色の c_3 と c_5 の 2 本のカラムが，移動先カラムになる．提案法は，青色の移動元カラムのシナプスについて，赤色の移動先カラムのシナプスが接続されたデータビットに再配置する．例えば，青色の移動元カラム c_1 と赤色の移動先カラム c_3 に注目する．**図 3.4 (a)** の通り，青色の移動元カラム c_1 のシナプス $c_1.y_k$ $(k = 1, 2, 3)$ は，再配置前はデータビット番号 1，2，3 に紐づけられた $c_1.y_1.a = x_1$，$c_1.y_2.a = x_2$，$c_1.y_3.a = x_3$ の 3 本である．これに対して**図 3.4 (b)** のように，提案法は，赤色の移動先

Algorithm 6 空間プーリング (提案する CLA)

カラムの活性状態化
for $i \leftarrow 1, 2, \ldots, n_c$ **do**
  $c_i.st \leftarrow$ Normal
  $c_i.ol \leftarrow |\{k \in \{1, 2, \ldots, n_{cy}\} \mid c_i.y_k.a = 1\}|$ ▷ オーバーラップ値
  $c_i.as \leftarrow c_i.ol + c_i.bst$ ▷ 活性化スコア
end for
for $i \leftarrow 1, 2, \ldots, n_c$ **do**
  if $c_i.ol \geq ol^{min}$ **then**
    if $c_i.as$ is in top n_{ac} score among all columns **then**
      $c_i.st \leftarrow$ Active
    end if
  end if
end for
追加のシナプス配置機構
if $|\{c_i \in \mathcal{C} \mid c_i.st = \text{Active}\}| < n_{ac}$ **then**
  $m \leftarrow n_{ac} - |\{c_i \in \mathcal{C} \mid c_i.st = \text{Active}\}|$
  for $j \leftarrow 1, 2, \ldots, m$ **do**
    $c_j \leftarrow$ Randomly pick a column from normal columns
    $c_j.st \leftarrow$ Active
    $k \leftarrow 1$ ▷ シナプス配置
    for $l \leftarrow 1, 2, \ldots, n$ **do**
      if $x_l = 1$ **then**
        $c_j.y_k.a \leftarrow x_l$
        $c_j.y_k.p \leftarrow p^c$
        $k \leftarrow k + 1$
      end if
    end for
  end for
end if
カラムのシナプスの更新
for $i \leftarrow 1, 2, \ldots, n_c$ **do**
  if $c_i.st =$ Active **then**
    $c_i.y_k.p \leftarrow \begin{cases} c_i.y_k.p + p_c^+, & \text{if } c_i.y_k.a = 1, \\ c_i.y_k.p - p_c^-, & \text{otherwise } (c_i.y_k.a = 0). \end{cases}$
  end if
end for
バンプアップ
for $i \leftarrow 1, 2, \ldots, n_c$ **do**
  if $c_i.zf < \theta^{bmp}$ **then**
    $c_i.y_k.p \leftarrow c_i.y_k.p + p^{bmp}$ $(k = 1, 2, \ldots, n_{cy})$
  end if
end for
ブースト
for $i \leftarrow 1, 2, \ldots, n_c$ **do**
  $c_i.af \leftarrow \begin{cases} 0.5 \cdot (c_i.af + 1), & \text{if } c_i.st = \text{Active}, \\ 0.5 \cdot (c_i.af), & \text{otherwise.} \end{cases}$
  $c_i.bst \leftarrow \kappa^{bst}(1 - c_i.af)$
end for
シナプスの適応配置
if $t \mod w = 0$ **then**
  Adaptive Synapse Arrangement ▷ **Algorithm 7**
end if

Algorithm 7 シナプスの適応配置

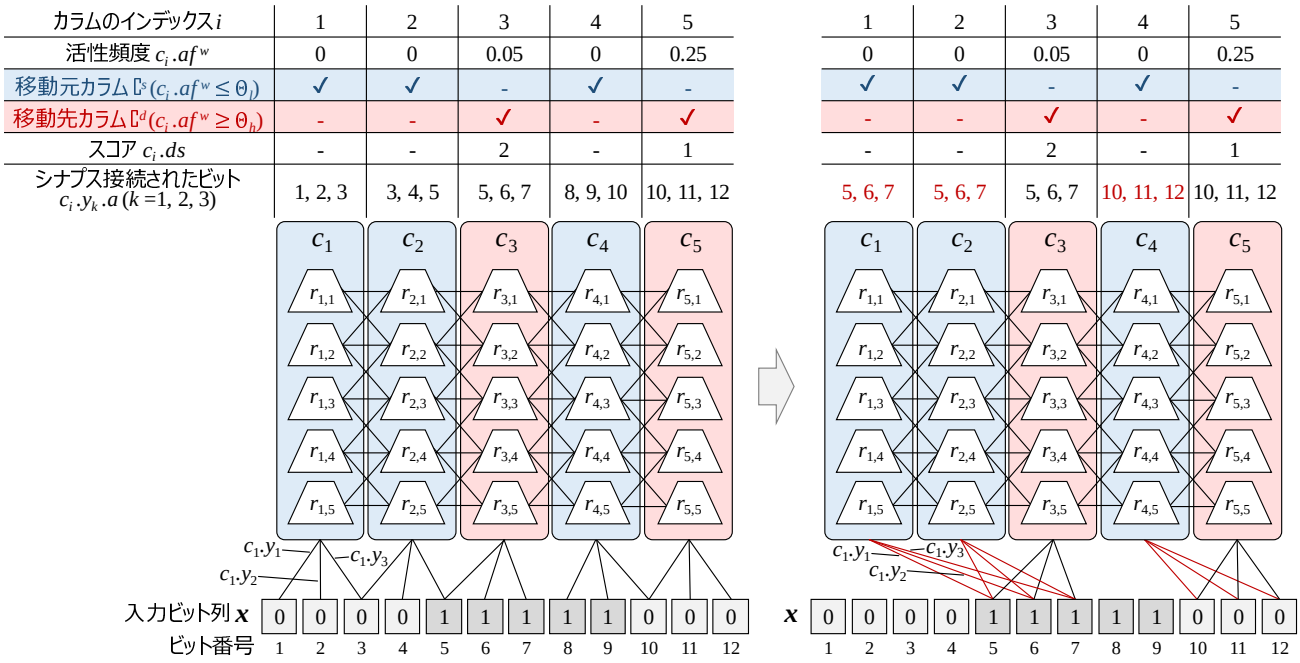
1: **Step 1**：移動元カラムと移動先カラムの選択
2: $\mathcal{C}^s \leftarrow \{c_i \in \mathcal{C} \mid c_i.af^w \leq \Theta_l\}$ ▷ 移動元カラム
3: $\mathcal{C}^d \leftarrow \{c_i \in \mathcal{C} \mid c_i.af^w > \Theta_h\}$ ▷ 移動先カラム
4: **Step 2**：移動先カラムのスコア算出
5: **for each** $c_d \in \mathcal{C}^d$ **do**
6: $\quad c_d.ds \leftarrow |\mathcal{C}^s| \cdot \frac{c_d.af^w - \Theta_h}{\sum_{c_j \in \mathcal{C}^d}(c_j.af^w - \Theta_h)}$ ▷ 移動先カラムのスコア
7: **end for**
8: **Step 3**：適応的なシナプスの再配置
9: **Step 3–a**：整数部の処理
10: **for each** $c_d \in \mathcal{C}^d$ **do**
11: $\quad$ **loop** $\lfloor c_d.ds \rfloor$ times ▷ 整数部 $\lfloor c_d.ds \rfloor$ の処理
12: $\quad\quad c_s \leftarrow$ Pop a column from column set $\mathcal{C}^s$
13: $\quad\quad$ **for** $k \leftarrow 1, 2, \ldots, n_{cy}$ **do** ▷ シナプス配置
14: $\quad\quad\quad c_s.y_k \leftarrow c_d.y_k$
15: $\quad\quad$ **end for**
16: $\quad$ **end loop**
17: $\quad c_d.ds \leftarrow c_d.ds - \lfloor c_d.ds \rfloor$ ▷ 小数部$c_d.ds$ の算出
18: **end for**
19: **Step 3–b**：小数部の処理
20: $\xi \leftarrow$ rand $(0, 1)$
21: **repeat**
22: $\quad c_d \leftarrow$ Pop a column from column set $\mathcal{C}^d$
23: $\quad \xi \leftarrow \xi + c_d.ds$
24: $\quad$ **if** $\xi \geq 1$ **then**
25: $\quad\quad c_s \leftarrow$ Pop a column from column set $\mathcal{C}^s$
26: $\quad\quad$ **for** $k \leftarrow 1, 2, \ldots, n_{cy}$ **do** ▷ シナプス配置
27: $\quad\quad\quad c_s.y_k \leftarrow c_d.y_k$
28: $\quad\quad$ **end for**
29: $\quad\quad \xi \leftarrow \xi - 1$
30: $\quad$ **end if**
31: **until** $\mathcal{C}^d \neq \emptyset$

カラム c_3 と紐づけられたデータビット番号 5，6，7 とのシナプスに再配置する．その結果，c_1 のシナプスは $c_1.y_1.a = x_5$，$c_1.y_2.a = x_6$，$c_1.y_3.a = x_7$ となる．

3.2.2.3　アルゴリズム

Algorithm 7 に示すカラムのシナプスの適応配置法の手順を以下に述べる．

ステップ 1：移動元カラムと移動先カラムの選択： まず，2 行目において，活性頻度が下限の閾値 Θ_l 以下であるカラムを移動元カラム群 $\mathcal{C}^s$ として選択する．選ばれた移動元カラム群 $\mathcal{C}^s$ は，これまでに活性状態になった回数が少ないことになる．これは，移動元カ



(a) 配置前　　(b) 配置後

図 3.4: 適応的なカラムのシナプス配置

ラム群 $\mathcal{C}^s$ のシナプス群が配置されたデータビットにおいて，1 の出現率が低い状態にあることを示す．次に，3 行目において，活性頻度が上限の閾値 Θ_h より高いカラムを移動先カラム群 $\mathcal{C}^d$ として選択する．選ばれた移動先カラム群 $\mathcal{C}^d$ は，これまでに活性状態になった回数が多いことになる．これは，移動先カラム群 $\mathcal{C}^d$ のシナプス群が配置されたデータビットにおいて，1 の出現率が高い状態にあることを示す．

ステップ２：移動先カラムのスコア算出： 各移動先カラム $c_d \in \mathcal{C}^d$ において，スコア $c_d.ds$ を 6 行目で計算する．スコア $c_d.ds$ は，移動元カラムの総数 $|\mathcal{C}^s|$，移動先カラムの総数 $|\mathcal{C}^d|$，そのカラムの活性頻度 $c_d.af^w$ から計算される．活性頻度 $c_d.af^w$ が高いほど，スコア $c_d.ds$ も大きくなる．

ステップ３：適応的なシナプスの再配置： シナプスの再配置は，二つのプロセスにわかれる．まず，**ステップ３–a** で，移動先カラムのスコア $c_d.ds$ の整数部について処理する．次に，**ステップ３–b** で，スコア $c_d.ds$ の小数部について処理する．

ステップ３–a は，10–18 行目において整数部を扱う．各移動先カラム $c_d \in \mathcal{C}^d$ につい

て，$c_d.ds$ の整数部である $\lfloor c_d.ds \rfloor$ 個の移動元カラムを $\mathcal{C}^s$ から選択する．そして，それらのカラムが持つシナプスを，移動先カラム c_d が持つシナプスが紐づけられたデータビットに再配置する．また，移動先カラム c_d のシナプスの永続値も，同様に再配置したシナプス群にコピーする．

ステップ3–b は，20–31 行目において小数部を扱う．まず，移動先カラムの選択バイアスを回避するため，ランダムな初期値 ξ を用いる．次に，残りの移動先カラム集合 $\mathcal{C}^d$ からある移動先カラム c_d を選び，そのカラムのスコアの小数部を ξ に加算する．そして，ξ が 1 以上になったとき，ある移動元カラム c_s のシナプスを再配置し，ξ から 1 を引く．これを移動元カラム集合 $\mathcal{C}^d$ が空になるまで繰り返す．

シナプスの適応配置の結果として，シナプスが配置されない入力データビットが生じる．入力データの傾向が変化し，シナプスが配置されないデータビットが 1 になったとき，このままでは，入力データを予測器の内部で表現できない．これに対処するため，提案する CLA には，**Algorithm 6** の 14–29 行目の処理がある．まず，カラムが活性状態になるために必要な最小のオーバーラップ値 ol^{min} を設定する．次に，活性状態となるカラム群が n_{ac} より少ないとき，追加のカラム群をランダムに選択する．そして，そのカラムのシナプスを 1 のデータビットに再配置し，それらを活性状態にする．

3.2.3 活性化スコア

入力データビット列において，1 になるデータビットに偏りが生じると，提案法では，シナプスの配置先が同一のカラムが増加する．これらのカラムは，オーバーラップ値が等しくなる．この場合，同じ入力値だが，出現する文脈が異なるケースにおいて，活性状態になるカラムのパターンが不安定になる．オーバーラップ値が同じカラムに対して，従来の CLA は，これまでに活性状態になった回数を考慮して活性化スコアを求める．提案する CLA は，これまでに活性状態になった回数とタイミングを考慮して活性化スコアを求めることで，上記の問題を回避する．

提案する CLA は，従来の CLA と異なる方法で活性化スコア $c_i.as$ を算出する．まず，

提案する CLA は，各カラム c_i について，以下の式で活性頻度 $c_i.af$ を更新する．

$$c_i.af = \begin{cases} 0.5 \cdot (c_i.af + 1), \text{if } c_i.st = \text{Active}, \\ 0.5 \cdot (c_i.af), \qquad \text{otherwise}. \end{cases} \tag{3.3}$$

従来の CLA は，現在の時点 t までに活性状態になった回数が多いほど，カラム c_i の活性頻度 $c_i.af$ が高くなる．一方，提案する CLA は，現在の時点 t から直近の時点での活性回数が多いほど，活性頻度 $c_i.af$ が高くなる．活性頻度 $c_i.af$ の値域は，従来の CLA，提案する CLA ともに，$[0, 1]$ である．提案する CLA は，ブースト値を以下の式で計算する．

$$c_i.bst = \kappa^{bst}(1 - c_i.af), \tag{3.4}$$

κ^{bst} は 1.0 未満の係数で，ブースト値 $c_i.bst$ を 1.0 未満にするための係数である．提案する CLA の活性化スコア $c_i.as$ は，以下の式で計算される．

$$c_i.as = c_i.ol + c_i.bst. \tag{3.5}$$

このように，提案する CLA は，同じオーバーラップ値 $c_i.ol$ であるカラムにおいて，それらのカラムが活性状態となった回数と，そのタイミングを考慮したブースト値 $c_i.bst$ に基づいて活性化スコア $c_i.as$ を算出する．

3.3　実験設定

提案法の効果を検証するため，時系列テストデータを用いて予測精度を比較する．また，提案法による CLA 内部のシナプス配置がどのように変化するか，入力に合わせてその配置を追従できるか確認するため，複数の入力データを用いてそのシナプス分布を確認する．

3.3.1　アルゴリズム

従来手法と比較するため，従来の CLA に提案法を導入した提案 CLA-AC，従来 CLA[8]，LSTM[3] を比較する．従来の LSTM では，ネットワークの最適化アルゴリズムとして Adam，RMSprop，SGD を用いる．

3.3.2 入力データ

まず，基本の入力データとして，周期的な時系列データである正弦波 $X_s(t)$ と，正弦波の合成波 $X_c(t)$ を用いた．また，非周期的な時系列データとして，ロジスティック写像 $X_l(t)$ を用いた．時点 t における入力データは，以下の式で定義される．

$$X_s(t) = \frac{1}{2} \cdot \sin\left(\frac{(t-1)\cdot\pi}{50}\right) + 0.5, \tag{3.6}$$

$$X_c(t) = \frac{1}{2} \cdot \sum_{k\in\{1,3,5,7,9\}} \frac{1}{k} \cdot \sin\left(\frac{(t-1)\cdot k\cdot\pi}{50}\right) + 0.5, \tag{3.7}$$

$$X_l(t) = \begin{cases} 0.4, & \text{if } t = 1, \\ 3.6 \cdot X_l(t-1) \cdot \{1 - X_l(t-1)\}, & \text{otherwise.} \end{cases} \tag{3.8}$$

入力時点は，$t \in [1, 10^5]$ に設定した．

次に，入力データの傾向が変化する例として，全時点 $t \in [0, 10^5]$ の中間にあたる 50,000 時点で，3 種類の入力データ $X_s(t)$，$X_c(t)$，$X_l(t)$ を他の入力データに変化させる計 6 種類の入力データを用いる．この入力では，前後で大きく異なる入力を予測できるように学習する必要がある．また，その入力データ傾向の変化前後における予測誤差とシナプス分布の変化について観察する．

最後に，実世界データとして，New York ISO [10] で公開された電力消費の予測タスクを用いる．入力データは [66] のケースと同様に，外れ値だと考えられるスパイクを削除し，欠損値を補完し，1 時間ごとの窓で移動平均を算出したデータから，2007 年 5 月 1 日から 2019 年 2 月 28 日までのニューヨーク市の 15 分間隔の電力消費データを獲得した．この入力では，最後の 28 日間，すなわち，2019 年 2 月の予測精度を比較する．

実験では，ある時点 t の入力データを入力として，次時点 $t+1$ の入力データを予測する，ということを各入力時点に対して繰り返した．

3.3.3 パラメータ

CLA の共通パラメータとして，カラム数は $n_c = 2,048$，各カラムのセル数は $n_r = 32$ に設定した．初期化において，入力データの値域は $[X^{\min}, X^{\max}] = [-0.01, +1.01]$，データビット長は $n = 421$，チャンク長は $\delta = 21$ とした．空間プーリングにおいて，活性状

態となるカラム数は $n_{ac} = 40$，カラムのシナプスの永続値の増加量は $p_c^+ = 0.05$，減少量は $p_c^- = 0.025225$，接続閾値は $\theta_c = 0.1$ に設定した．時間プーリングにおいて，予測状態となるために必要な活性状態のセルとの接続数は $n_p = 15$，セルのシナプスの永続値の増加量は $p_r^+ = 0.1$，減少量は $p_r^- = 0.1$，接続閾値は $\theta_r = 0.5$ に設定した．ここでは，基本的に NuPIC のライブラリ [8] で設定されたデフォルトのパラメータを使用する．データビット長 n については，ライブラリでは $n = 396$ だが，本実験では $n = 421$ に設定した．これは，入力データビットのパターン数を 400 とし，予測誤差からどのくらいのビット数分のずれが生じたのか判断しやすくするためである．

従来 CLA において，シナプス生成半径は $r = 13$，密度は $d = 0.8$ とした．その結果，式 (2.1) より，各カラムのシナプス数は $n_{cy} = 21$ となる．また，カラムのシナプスの初期接続率は $\rho = 0.5$ とした，そして，バンプアップにおける係数は $\kappa^{bmp} = 0.001$，バンプアップの永続値増加量は $p^{bmp} = 0.01$，ブーストの係数は $\kappa^{bst} = 1.0$ に設定した．ここでは，ライブラリのパラメータと比較して，シナプス生成半径 r を $r = 16$ から $r = 13$ に変更した．これは，各カラムのシナプス数 n_{cy} を入力データのチャンク長 δ と同数とすることで，カラムが複数の入力に対応するケースを減らすためである．

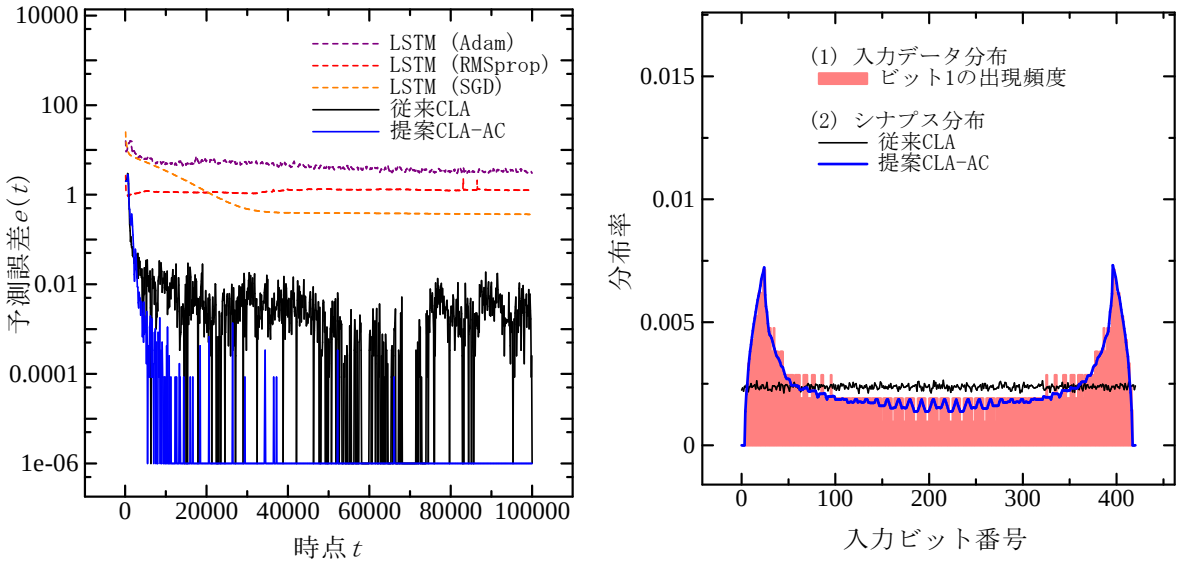
提案 CLA-AC において，各カラムのシナプス数は従来の CLA と同等となるように $n_{cy} = 21$，固定初期永続値は $p^c = 0.2$ に設定した．シナプスの適応配置において，以下のパラメータ群 $w = 1,000$，$\Theta_l = 0$，$\Theta_h = 0.02$，$ol^{min} = 17$，$\kappa^{bst} = 0.99$ を用いた．

従来の LSTM において，ネットワークの最適化に Adam[64]，RMSprop[65]，SGD を用いた．また，二つの中間層のユニット数は，それぞれ 100 とした．一つ目の層は，線形関数を活性化関数として用いた全結合層で，二つ目の層は，tanh 関数を活性化関数として用いた LSTM 層とした．出力層は，ユニット数 1 の活性化関数を用いない全結合層とした．損失関数として，平均二乗誤差を使用した．

3.3.4　評価指標

予測精度を評価するため，次式で求める 100 時点ごとの予測誤差 $e(t)$ を用いる．

$$e(t) = \sum_{\tau=t-99}^{t} |\bar{X}(\tau+1) - X(\tau+1)|, \tag{3.9}$$



(a) 予測誤差　　(b) ビット列におけるシナプス分布

図 3.5: 正弦波 $X_s(t)$ における結果

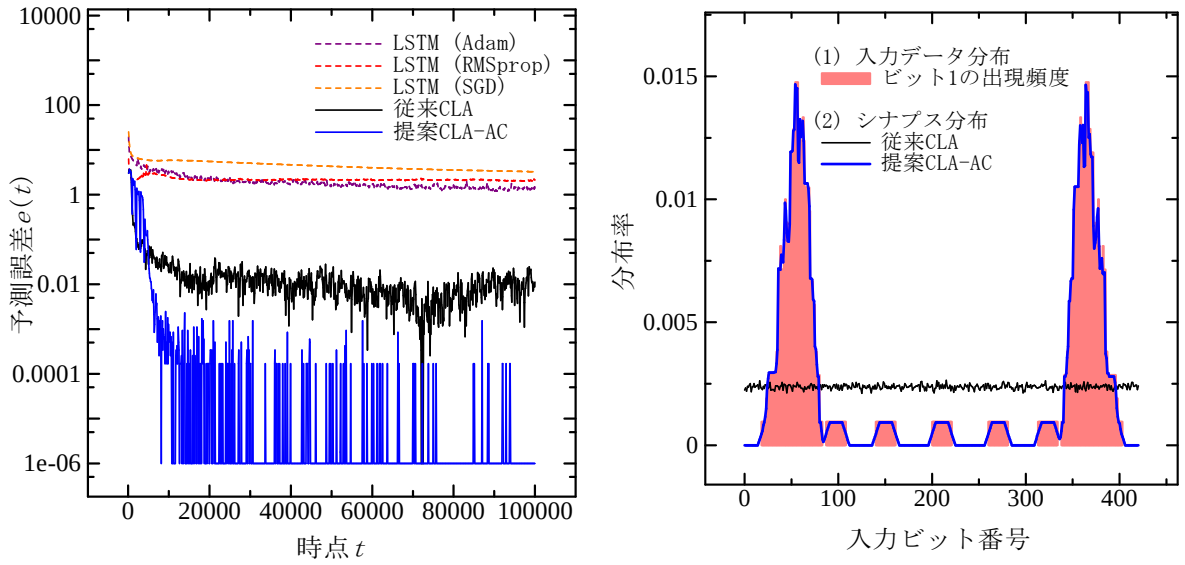
ここで，$\bar{X}(t+1)$ は，次時点 $t+1$ の予測値である．$X(t+1)$ は，次時点 $t+1$ の実際の入力値である．$e(t)$ が小さいほど，予測精度が高いと判断する．本章では，$t \in \{100, 200, 300, 400, \ldots, 10^5\}$ とし，$e(t)$ を算出する．

また，提案するカラムのシナプスの適応配置の効果を確認するため，入力ビットが 1 となる頻度の分布と入力ビット列への CLA のシナプス分布を比較する．この分布が近いほど，入力データの傾向に合わせたシナプス配置を実現したと判断できる．

3.4　実験結果と考察

3.4.1　予測誤差の比較

従来 CLA，三つの従来の LSTM，提案法である提案 CLA-AC の結果を比較したグラフを**図 3.5** と **3.6** と **3.7** に示す．**図 3.5 (a)**，**3.6 (a)**，**3.7 (a)** は，それぞれ三つの入力における予測誤差の推移である．縦軸は対数目盛であり，10^{-6} 以下の予測誤差は 10^{-6} としてプロットした．また，**図 3.5 (b)**，**3.6 (b)**，**3.7 (b)** において，入力ビットが 1 となる頻度の分布と入力ビット列へのシナプス分布を示した．横軸は，1 から $n = 421$ までの入力ビット番号を表す．赤色の棒グラフは，各入力データビットが 1 になる確率を表

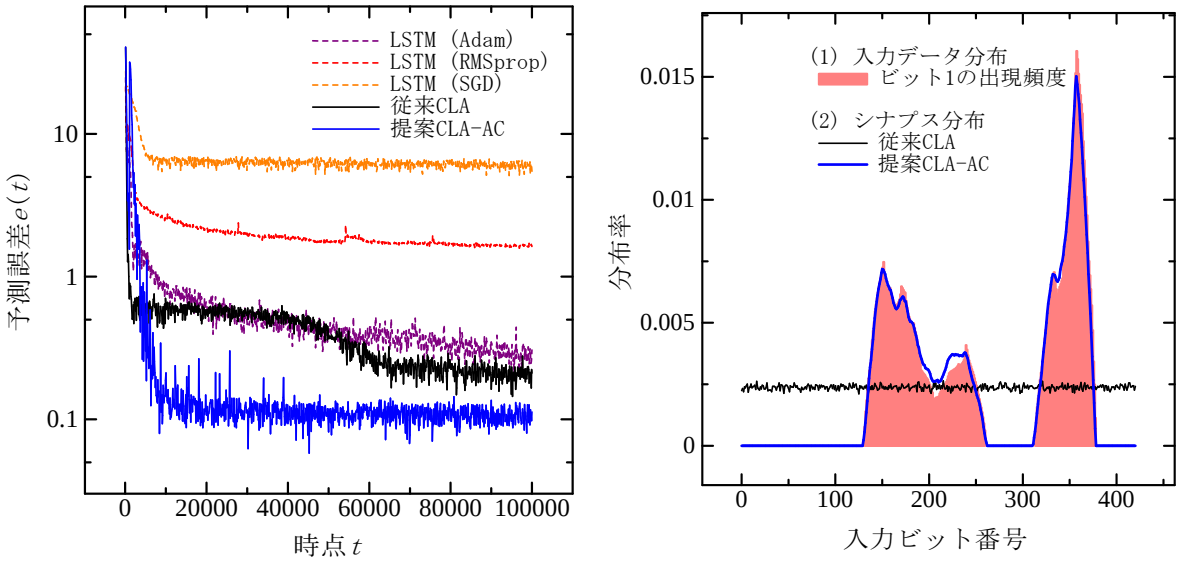


(a) 予測誤差　　(b) ビット列におけるシナプス分布

図 3.6: 正弦波の合成波 $X_c(t)$ における結果

す．2 本の折れ線グラフは，全入力データビットへのカラムの全シナプスにおける，各入力データビットへのカラムのシナプス数の割合である．縦軸は，棒グラフおよび折れ線グラフにおける割合である．

まず，正弦波 $X_s(t)$ の結果について述べる．予測誤差の推移を示した**図 3.5 (a)** より，二つの CLA の予測誤差が，三つの LSTM より低いことがわかる．また，提案 CLA-AC は，従来 CLA より低い予測誤差を示すことがわかる．入力ビットにおける 1 の出現率と入力ビット列に紐づけられたシナプス分布を示した**図 3.5 (b)** より，時系列データ $X_s(t)$ におけるビット列の 1 の出現率には偏りがあることがわかる．また，黒色の折れ線グラフより，従来 CLA のシナプス分布は，データビット列に対してほぼ一様であることがわかる．これは，従来の CLA が，入力データビット列における 1 の出現率を考慮しないためである．また，従来 CLA のシナプス分布は，カラムのシナプス配置の仕組みにランダム性が含まれる影響により，完全に一様にはならない．1 の出現率が高いデータビットに紐づけられたシナプスは，カラムを活性状態にすることへの貢献度が高い．一方，1 の出現率が低いデータビットに紐づけられたシナプスは，カラムを活性状態にすることへの貢献度が低い．これに対して，青色の折れ線グラフより，提案 CLA-AC のシナプス分布は，ビット列の 1 の出現率に近いことがわかる．このように，適応的に配置されたシナプス



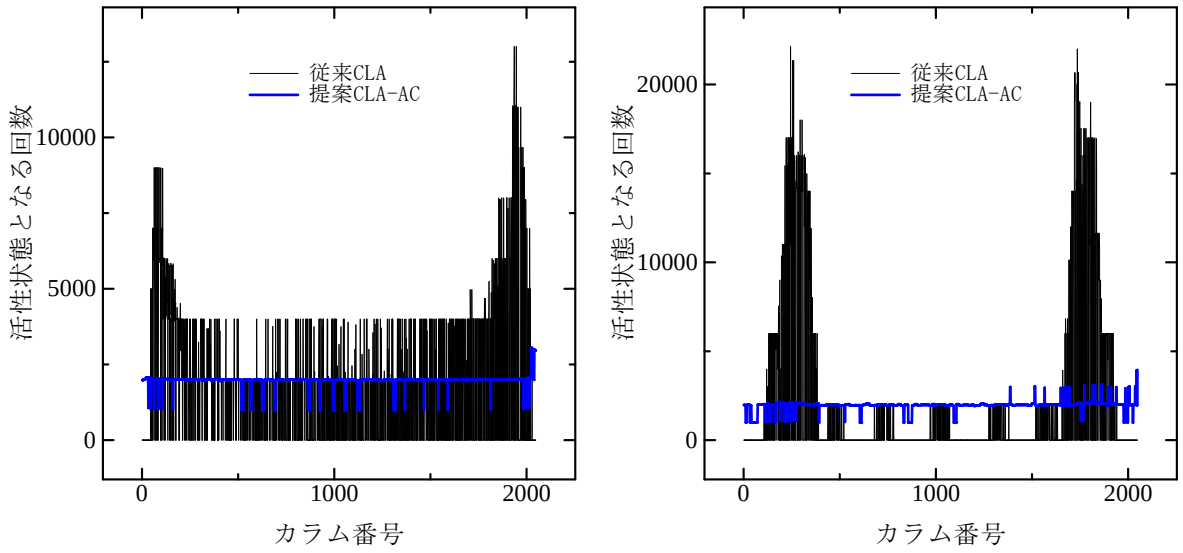
(a) 予測誤差　　(b) ビット列におけるシナプス分布

図 3.7: ロジスティック写像 $X_l(t)$ における結果

は，入力データの表現にカラムを活用しやすくする．

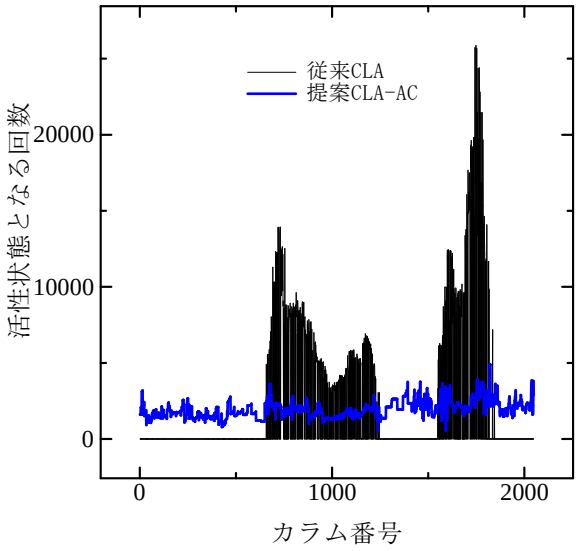
次に，正弦波の合成波 $X_c(t)$ の結果について述べる．予測誤差の推移を示した**図 3.6 (a)** より，二つの CLA が，三つの LSTM より低い予測誤差を達成することがわかる．また，提案 CLA-AC の予測誤差は，従来 CLA より低いことがわかる．**図 3.6 (b)** に入力ビットが 1 となる確率の分布と入力データビットに対するシナプス分布を示す．この結果から，正弦波の合成波においては，ある範囲の入力データビットは 1 にならず，その確率が 0 となることがわかる．このように，偏りのある入力データに対して，提案 CLA-AC は，1 にならないデータビットにシナプスを配置しないことがわかる．また，提案 CLA-AC は，ビット列が 1 になる確率にあわせてシナプスを配置することがわかる．

ロジスティック写像 $X_l(t)$ の結果について述べる．予測誤差の推移を示した**図 3.7 (a)** より，従来 CLA の予測誤差と比較して，Adam を用いた従来の LSTM の方が，予測誤差が低い時点が存在することがわかる．しかし，提案 CLA-AC は，他の方法より予測誤差が低く，安定した予測誤差の推移を示すことがわかる．また，**図 3.7 (b)** に入力ビットが 1 となる確率の分布と入力ビット列におけるシナプスの分布を示す．この結果から，1 となる確率が 0 になるビットがあることがわかる．これにより，従来 CLA は一部のカラム群を活性状態にすることができず，それらのカラムを入力データの内部表現に活用でき



(a) 正弦波 $X_s(t)$

(b) 正弦波の合成波 $X_c(t)$



(c) ロジスティック写像 $X_l(t)$

図 3.8: カラムが活性状態となる回数

ない．これに対して，提案 CLA-AC は，1 の出現率にあわせてシナプスを配置できることがわかる．このシナプスの再配置は，カラムを活用しやすくし，予測精度の改善に貢献する．

表 3.1 に学習が進んだ入力後半にあたる 50000–100000 時点での予測誤差の合計を示す．この結果から，三つ全ての入力において，提案 CLA-AC が今回比較した方法の中で

表 3.1: 入力後半にあたる 50,000–100,000 時点における予測誤差合計

方法	正弦波 $X_s(t)$	正弦波の合成波 $X_c(t)$	ロジスティック写像 $X_l(t)$
LSTM (Adam)	3773.78	1578.59	395.53
LSTM (RMSprop)	1119.88	1939.73	1670.2
LSTM (SGD)	555.9	3888.33	5610.31
従来 CLA	0.9877	3.79	122.34
提案 CLA-AC	0.0004	0.01	53.92

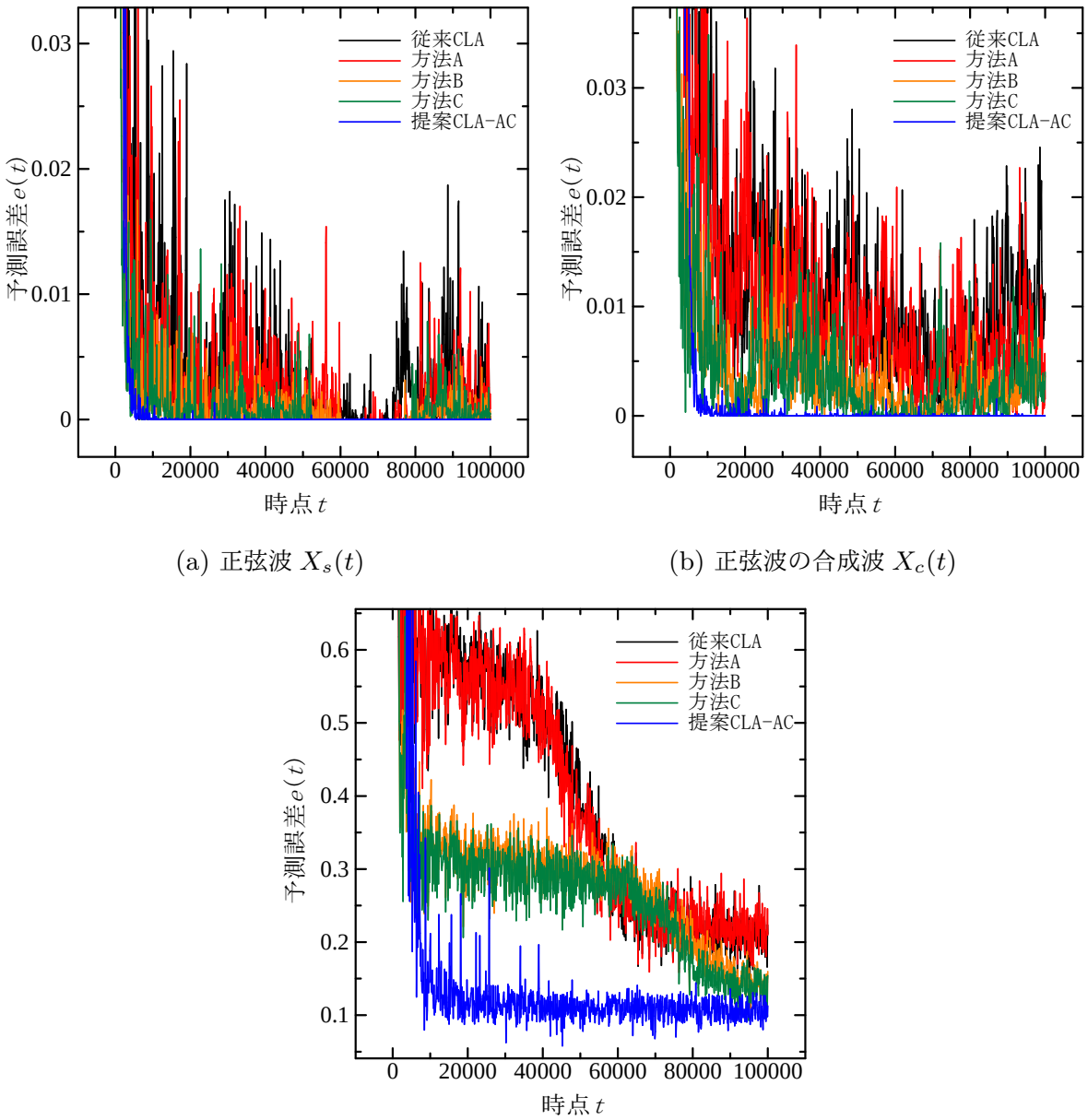
表 3.2: 提案法の要素ごとの効果を評価するためのアルゴリズム

	従来 CLA	方法 A	方法 B	方法 C	提案 CLA-AC
(i) 決定論的な配置	-	✓	✓	✓	✓
(ii) 初期永続値の固定	-	-	✓	✓	✓
(iii) シナプスの集約	-	-	-	✓	✓
シナプスの適応配置	-	-	-	-	✓

最も低い予測誤差を達成したことがわかる．

それぞれのカラムが活性状態になった回数を**図 3.8** に示す．従来 CLA は，カラムごとに活性状態になる回数に偏りがあることがわかる．その結果，内部データ表現に使われないカラムが生じ，予測精度の低下を引き起こす．これに対して，提案 CLA-AC は，入力データを表現するために活性状態となる頻度がほぼ平均化され，全てのカラムが，一様に近い回数で活性状態になることがわかる．これにより，内部データ表現にカラムが活用しやすくなり，予測精度を改善できる．

以上の結果から，入力データの傾向に基づく適応的なシナプスの配置によって，提案 CLA-AC は他のアルゴリズムより低い予測誤差を達成することが明らかとなった．



(a) 正弦波 $X_s(t)$

(b) 正弦波の合成波 $X_c(t)$

(c) ロジスティック写像 $X_l(t)$

図 3.9: 提案法の要素ごとの予測誤差の推移

3.4.2　提案する CLA の各構成要素の効果

提案 CLA-AC におけるアルゴリズムの各構成要素の効果を評価するため，**表 3.2** に示す 5 種類の方法を比較した．三つの方法 A–C は，それぞれ提案 CLA-AC から方法を削ったものである．方法 A は，従来 CLA にシナプスの初期化における (i) 決定論的な配

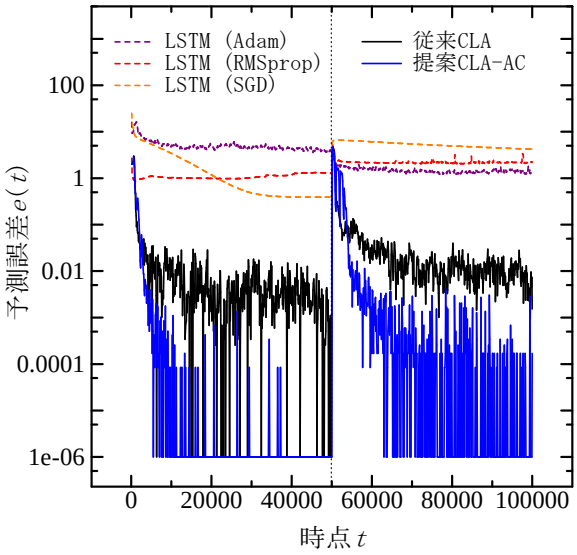
置を適用した方法である．方法 B は，方法 A に (ii) 初期永続値の固定を組み込んだ方法である．方法 C は，方法 B に (iii) シナプスの集約を適用した方法である．また，提案 CLA-AC は，方法 C にシナプスの適応配置を導入した方法である．

図 3.9 (a) に正弦波 $X_s(t)$ を入力としたときの予測誤差を示す．この結果から，シナプスの初期化において (i) 決定論的な配置を用いた方法 A は，従来 CLA より低い予測誤差を達成することがわかる．また，(ii) 初期永続値の固定を適用した方法 B は，方法 A よりも予測誤差が低いことがわかる．さらに，(iii) シナプスの集約を適用した方法 C は，方法 B と比較して予測誤差を改善することがわかる．最後に，シナプスの適応配置を導入した提案 CLA-AC は，方法 C より予測誤差が低くなることがわかる．**図 3.9 (b)** に正弦波の合成波 $X_c(t)$ を入力としたときの予測誤差を示す．この結果から，正弦波 $X_s(t)$ を入力としたときの結果と同様の傾向の結果が見られることがわかる．そのため，提案した方法を適用することによって予測誤差が減少することが示される．**図 3.9 (c)** にロジスティック写像 $X_l(t)$ を入力としたときの結果を示す．ここでも，前の二つの入力のときと同様の傾向の結果であることがわかる．方法 A–C は，学習前半の早い時点において予測精度を改善する効果が見られ，適応的なカラムのシナプス配置は学習後半において予測精度を改善する効果が見られる．

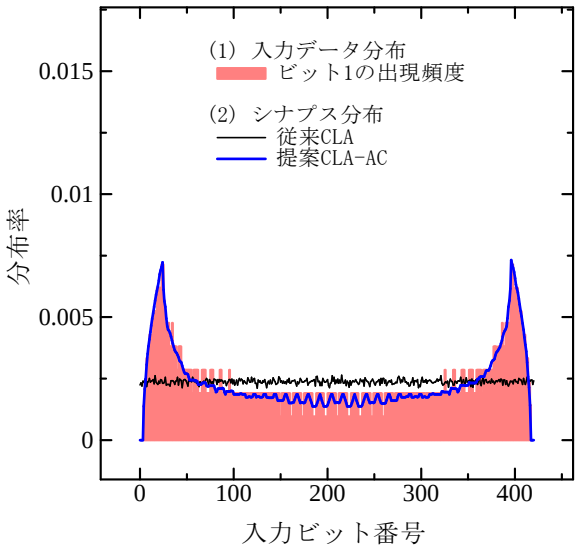
以上の結果から，提案 CLA-AC のアルゴリズムの各構成要素が，予測精度の改善に貢献することが明らかになった．

3.4.3 時系列入力データの変化に対する結果

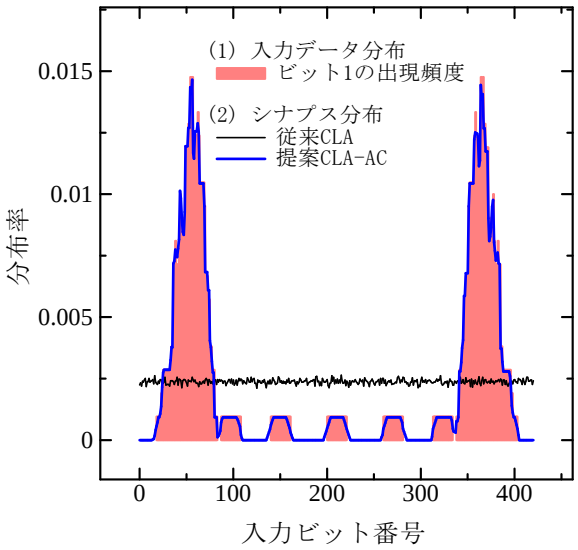
入力データの傾向が変化するときの提案 CLA-AC の効果について検証する．入力データが，正弦波 $X_s(t)$ から正弦波の合成波 $X_c(t)$ に変化するケースでの結果を**図 3.10** に示す．反対に，正弦波の合成波 $X_c(t)$ から正弦波 $X_s(t)$ に入力が変化するケースでの結果を**図 3.11** に示す．予測誤差の推移を示した**図 3.10 (a)** と**図 3.11 (a)** より，入力データが変化する中間の時点において，提案 CLA-AC の予測誤差が，一時的に増加することがわかる．しかし，提案 CLA-AC の予測誤差は，比較したアルゴリズムの中で最終的に最も低くなることがわかる．次に，入力データビットが 1 となる確率の分布とビット列に紐づけられたシナプスの分布を観察する．**図 3.10 (b)** と**図 3.11 (b)** に入力データが変化する前の結果を示す．同様に，**図 3.10 (c)** と**図 3.11 (c)** に入力データが変化した後の結果



(a) 予測誤差



(b) 入力とシナプスの分布：50,000 時点

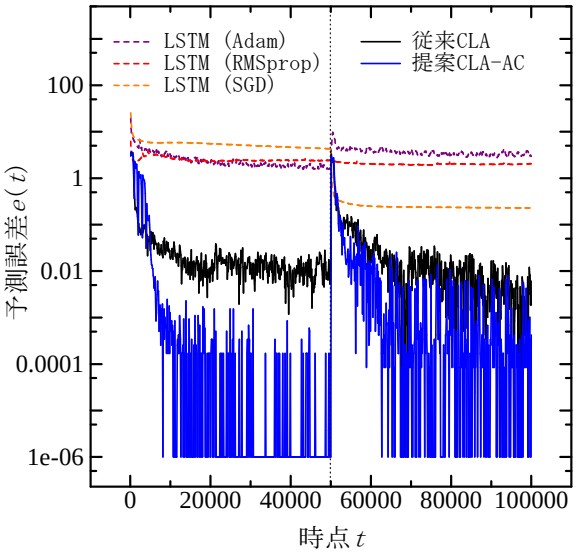


(c) 入力とシナプスの分布：100,000 時点

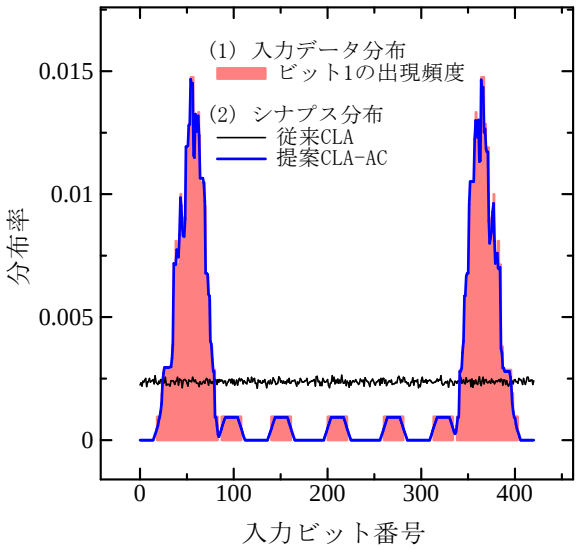
図 3.10: $X_s(t)$ から $X_c(t)$ へ変化する時系列データでの結果

を示す．この結果から，赤色の棒グラフで示した入力データビットが 1 となる確率が，入力データが変化する前後で大きく異なることがわかる．また，青色の折れ線グラフで示した提案 CLA-AC のシナプス分布も入力データが変化する前後で異なり，その分布は入力データビットが 1 となる確率に近似することがわかる．

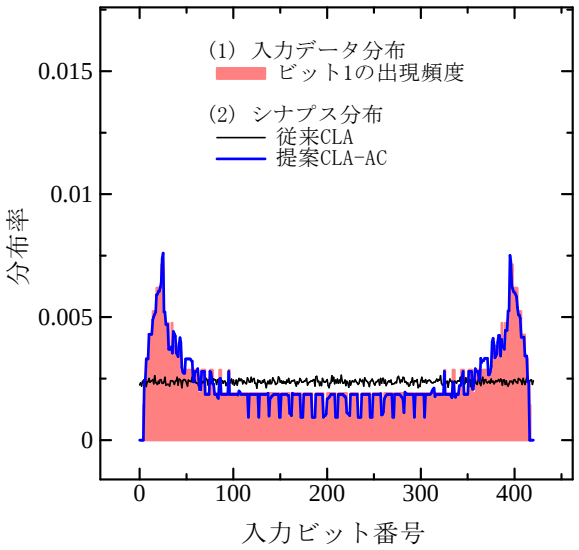
正弦波 $X_s(t)$ からロジスティック写像 $X_l(t)$ に入力が変化するケースでの結果を図



(a) 予測誤差



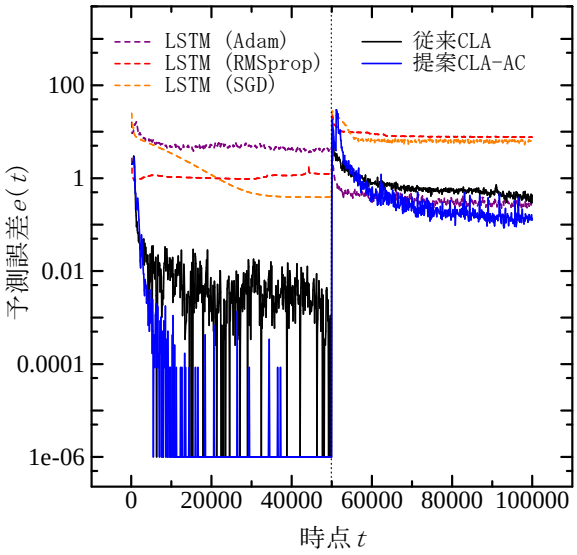
(b) 入力とシナプスの分布：50,000 時点



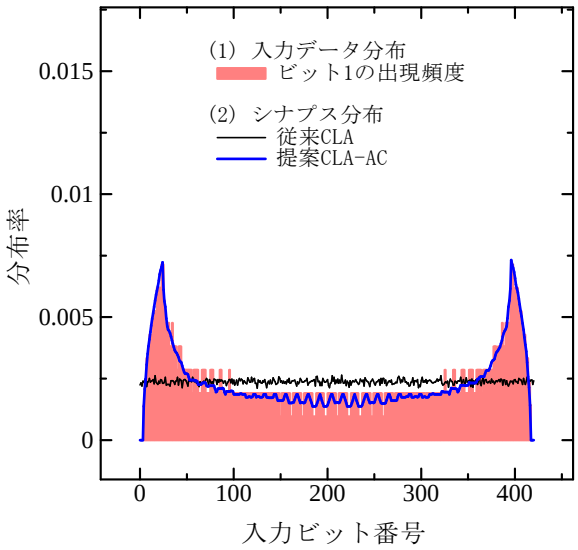
(c) 入力とシナプスの分布：100,000 時点

図 3.11: $X_c(t)$ から $X_s(t)$ へ変化する時系列データでの結果

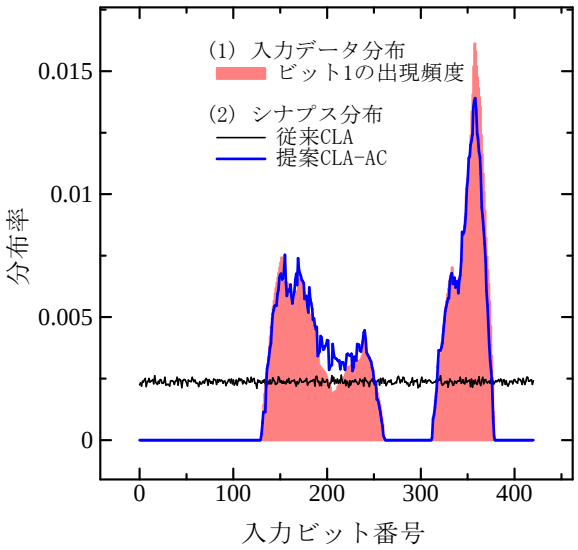
3.12 に示す．反対に，ロジスティック写像 $X_l(t)$ から正弦波 $X_s(t)$ に入力が変化するケースでの結果を**図 3.13** に示す．同様に，**図 3.14** に入力が正弦波の合成波 $X_c(t)$ からロジスティック写像 $X_l(t)$ に変化するケースでの結果を示す．反対のケースとして，ロジスティック写像 $X_l(t)$ から正弦波の合成波 $X_c(t)$ に入力が変化するときの結果を**図 3.15** に示す．これらの結果から，前述の結果と同様に，提案 CLA-AC が入力ビット列におい



(a) 予測誤差



(b) 入力とシナプスの分布：50,000 時点

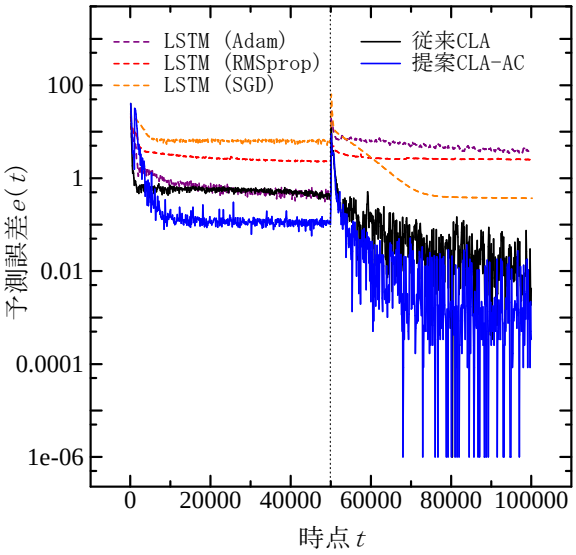


(c) 入力とシナプスの分布：100,000 時点

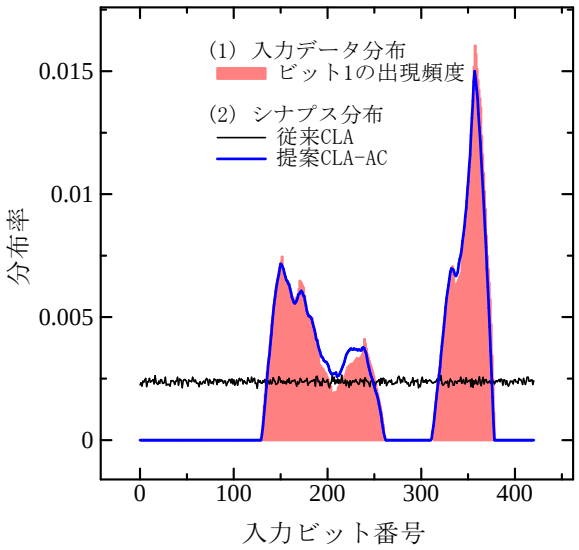
図 3.12: $X_s(t)$ から $X_l(t)$ へ変化する時系列データでの結果

て 1 となる確率の分布にあわせてシナプスを適応的に配置することで，最も低い予測誤差を達成することがわかる．

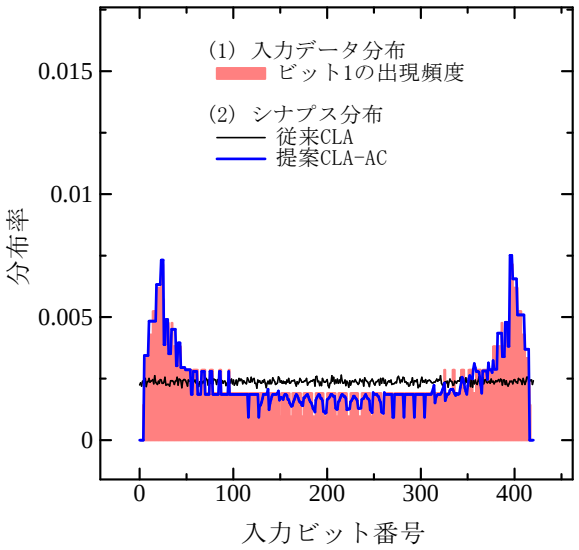
以上の結果より，提案 CLA-AC は入力データにあわせて適応的にシナプスを配置でき，従来 CLA よりも高い予測精度を達成することが示された．



(a) 予測誤差



(b) 入力とシナプスの分布：50,000 時点

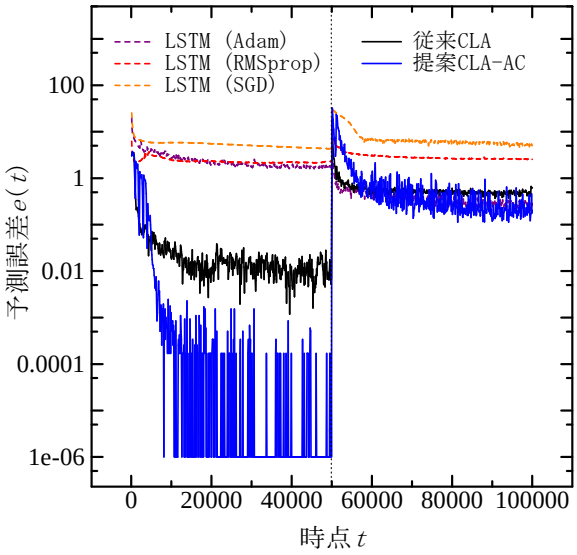


(c) 入力とシナプスの分布：100,000 時点

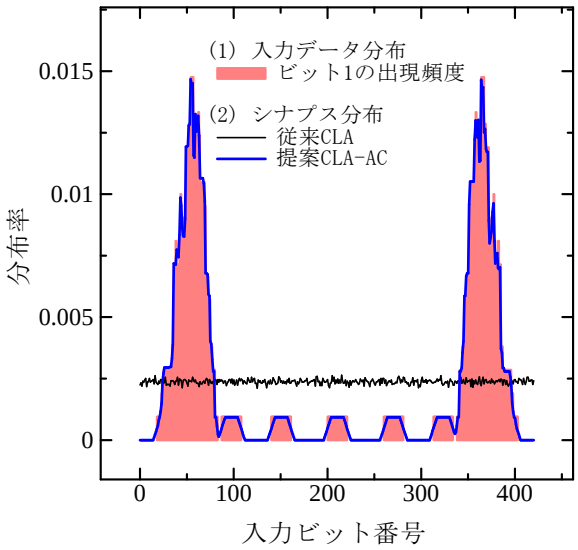
図 3.13: $X_l(t)$ から $X_s(t)$ へ変化する時系列データでの結果

3.4.4　実世界の消費電力量の予測

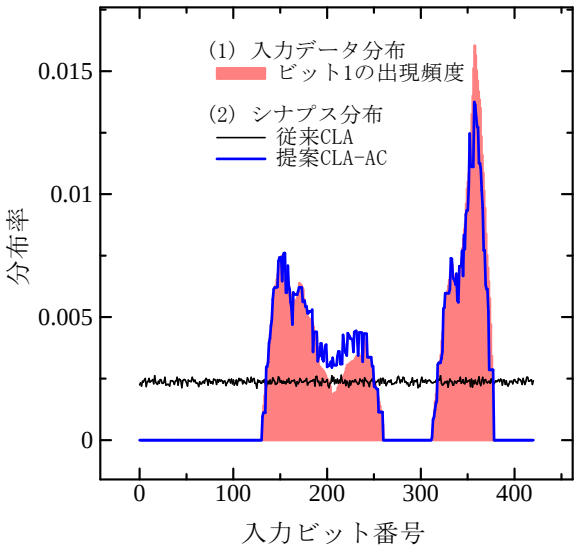
実世界データの予測として，New York ISO [10] で公開された電力消費の予測タスクについて，予測性能を比較する．本節では，提案 CLA-AC におけるシナプスの適応配置の時点間隔 w は 1 年周期を考慮して 35,040(= 365 日 × 24 時間 × 4 データ点，15 分ごと



(a) 予測誤差



(b) 入力とシナプスの分布：50,000 時点

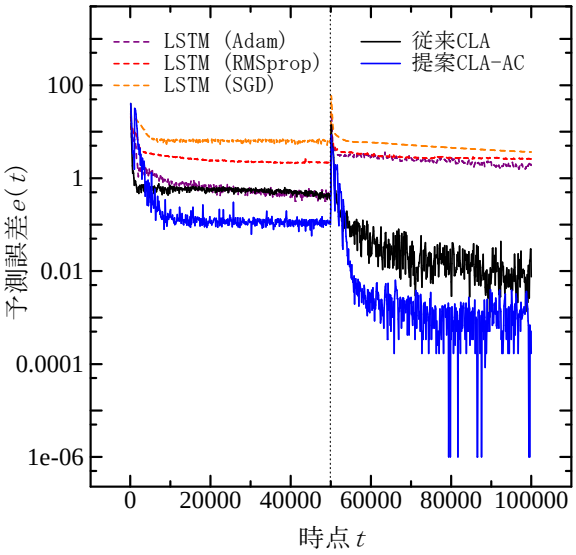


(c) 入力とシナプスの分布：100,000 時点

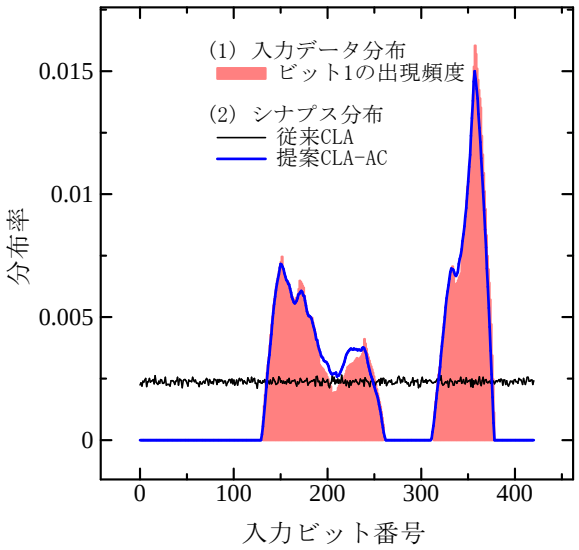
図 3.14: $X_c(t)$ から $X_l(t)$ へ変化する時系列データでの結果

に 1 データ) に設定し，他は前節で用いた実験設定と同じパラメータを用いる．**図 3.16** に実世界の電力消費予測の結果を示す．

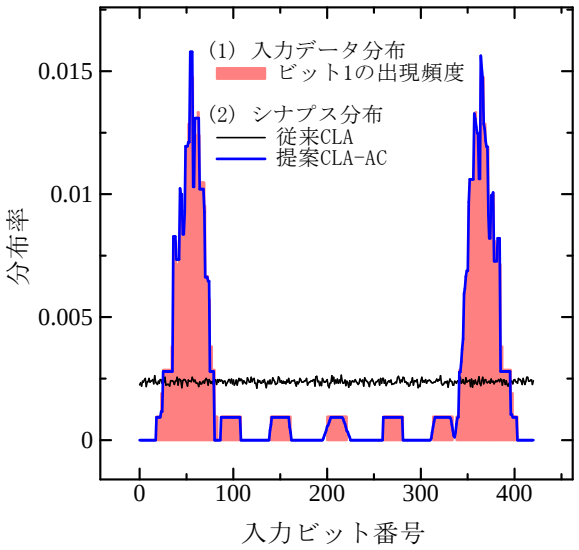
まず，2019 年 2 月にあたる最後の 28 日間での予測誤差合計を**図 3.16 (a)** に示す．この結果から，LSTM の性能は，最適化アルゴリズムによって差があることがわかる．SGD を用いた方法が，三つの従来の LSTM の中で最も高い予測精度を示すことがわかる．一



(a) 予測誤差



(b) 入力とシナプスの分布：50,000 時点

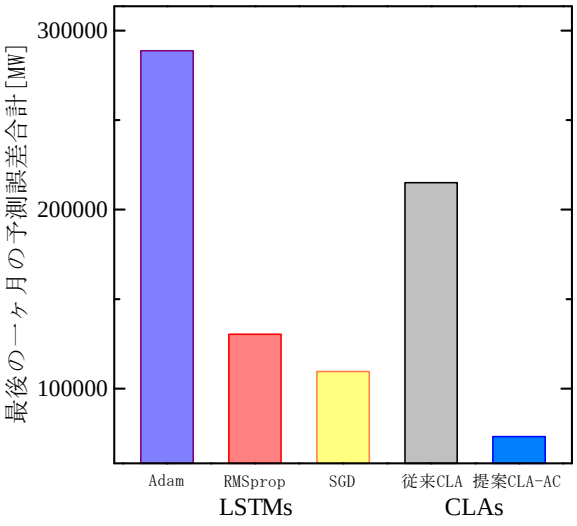


(c) 入力とシナプスの分布：100,000 時点

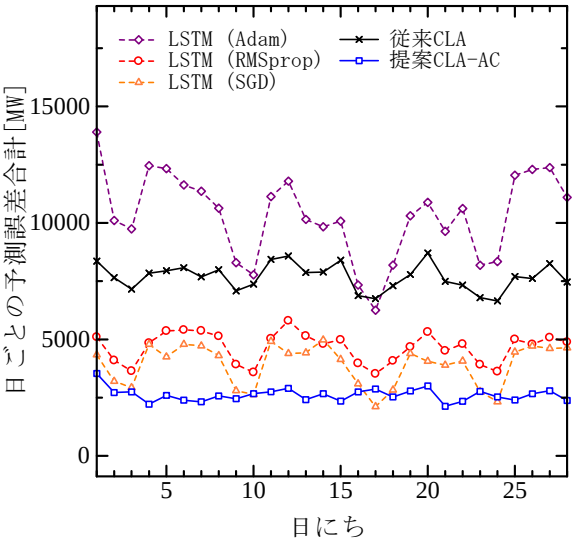
図 3.15: $X_l(t)$ から $X_c(t)$ へ変化する時系列データでの結果

方，提案 CLA-AC は，比較した 5 種類のアルゴリズムの中で予測誤差が最も低くなることがわかる．

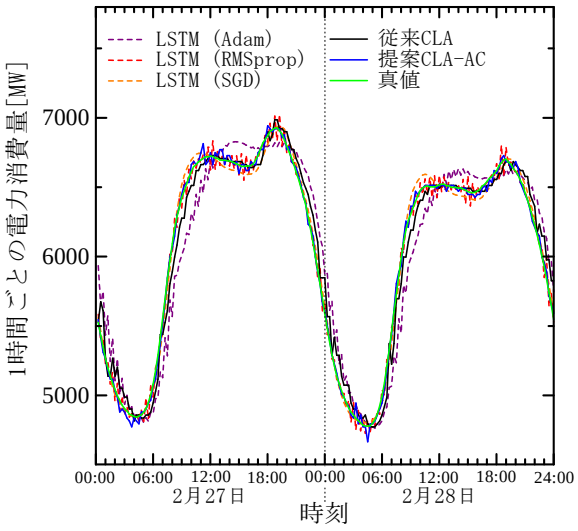
次に，**図 3.16 (b)** に最後の 28 日間における日ごとの予測誤差の推移を示す．この結果から，提案 CLA-AC が，5 種類のアルゴリズムの中でほとんどの日で最も低い予測誤差を達成することがわかる．三つの従来の LSTM に注目すると，七日のうち二日連続し



(a) 予測誤差合計



(b) 日ごとの予測誤差



(c) 最後の二日間の真値と予測値

図 3.16: 実世界の電力消費予測の結果

て予測誤差が減少しており，週ごとに周期があることがわかる．この連続した二日間は土曜日と日曜日にあたる．このことから，従来 LSTM は，休日の時系列パターンに適合するものの，平日の予測精度を低下させることがわかる．一方，提案 CLA-AC は休日と平日の予測精度の差が少なく，一週間にわたって安定した予測を実現することがわかる．これにより，**図 3.16 (a)** に示す通り，提案法は最も低い予測誤差合計を達成したと考えられる．

最後に，**図 3.16 (c)** に 2 月 27 日と 28 日の実際の電力消費量と予測値の推移を示す．電力消費は 1 日に 2 度のピークが存在することがわかる．具体的には，12 時あたりと 19 時あたりである．Adam を用いた従来の LSTM と従来 CLA では，特にその 2 度のピークの時間帯において予測誤差が増える．従来 CLA はピークの位置は予測できているが，Adam を用いた従来の LSTM はピーク位置の予測が困難となる．そのような急激に電力消費量が変化する時間帯において，RMSprop と SGD を用いた従来の LSTM と提案 CLA-AC は，Adam を用いた従来の LSTM と従来 CLA よりも予測精度が高くなる．SGD を用いた従来の LSTM は，安定して滑らかな電力消費の推移の予測を実現する．しかし，一つ目のピークである 12 時あたりの予測値が実際の値より大きくなり，二つのピークの間にあたる 12–19 時の予測値が実際の値より小さくなる．ピーク間の時間帯において，RMSprop を用いた従来の LSTM と提案 CLA-AC は SGD を用いた従来 LSTM よりも高い予測精度を実現する．そして，RMSprop を用いた従来の LSTM による日中の電力消費量の予測値は，不安定で，変動が激しくなる傾向がある．提案 CLA-AC による電力消費の予測値も同様に変動が激しいが，RMSprop を用いた従来の LSTM よりもその変動は小さい．

以上の結果より，実世界の電力消費量を予測するタスクにおいて，提案 CLA-AC は本節で比較した 5 種類のアルゴリズムの中で最も良い予測精度を達成することがわかる．

3.5 結言

本章では，従来の CLA におけるカラムのシナプスの取り扱いに着目し，初期シナプス配置方法と，入力値の偏りにあわせて適応的にカラムのシナプスを配置する方法を提案した．複数の人工テスト時系列データと実世界の電力消費量を用いた実験の結果，提案法を導入した提案 CLA-AC が，従来 CLA と従来の LSTM より予測誤差が低くなることを示した．また，提案法が，入力データの分布にあわせてシナプスを適応的に配置できることを確認した．

第 4 章

不活性セルのシナプス更新

CLA は，シナプスのネットワークをオンラインで構築しながら時系列予測する．本章では，予測精度の向上を目的として，セルのシナプスネットワークの構築を促進するため，不活性セルのシナプス更新法を提案する．CLA の予測器において，本章が取り組む部分を**図 4.1** に赤色で示した．本章は，CLA におけるセルのシナプスの取り扱いに注目し，CLA-IU (CLA with Inactive-cell synapse Update) を提案する．

4.1 概要

例えばノイズを含むような入力において，完璧な予測を実現することは困難である．このような入力において，CLA は予測精度が悪化する．直接的な原因は，入力がノイズを含むことだが，予測を成功したときにしかシナプスネットワークを更新しないことが，CLA の仕組みにおける問題点である．本章では，この点に着目する．

CLA におけるセルの予測状態からの状態遷移を**図 4.2** に示す．ここで，オーバーラップ値が 1 以上 ($c_i.ol \geq 1$) のカラム c_i を活性候補と呼ぶことにする．活性候補のカラム集合の部分集合が，活性状態のカラム集合になる．また，カラムが活性状態となる条件は，2.5 節で述べた従来 CLA と同様に，活性化スコア $c_i.as$ が高い上位 n_{ac} 本のカラムである．

ケース I において，予測状態のセルを内包するカラムが活性候補になった後に活性状態になったとき，従来 CLA は，内包される予測状態のセルのシナプスの永続値を更新し，

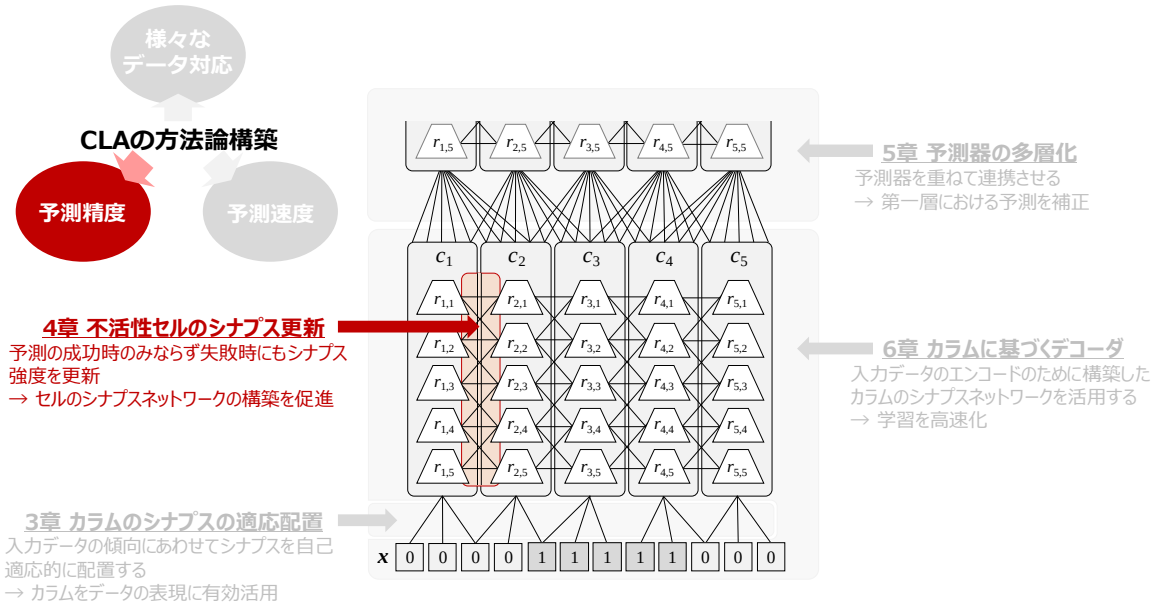


図 4.1: 4 章の位置づけ

シナプスの接続強度に反映する．一方，ケース II において，予測状態のセルを内包するカラムが活性候補になった後に活性状態にならなかったとき，従来 CLA は，それらの予測状態のセルのシナプスの永続値を更新しない．また，ケース III においても，それらのカラムが活性候補にならなかったとき，従来 CLA は，それらの予測状態のセルで永続値を更新しない．従来 CLA は，ケース I のみでシナプスネットワークを更新する．ケース II と III においてシナプスの永続値を更新しないことで，予測のために必要なシナプスネットワークの構築が不十分となり，予測精度が悪化する一因となる．これらのケース II と III において適切にシナプスを更新することで，セルのシナプスネットワークの構築を促進し，CLA の予測精度を改善できる可能性がある．

4.2　方法

CLA におけるセルのシナプスネットワークの構築を促進するため，予測状態から通常状態に遷移した不活性セルのシナプスの永続値を更新する三つの方法を提案する．**図 4.2** に示す通り，方法 A と方法 B は，それぞれケース II とケース III に適用される．方法 C は，ケース II と III の両方に適用される．セルのシナプスの永続値は，時間プーリングに

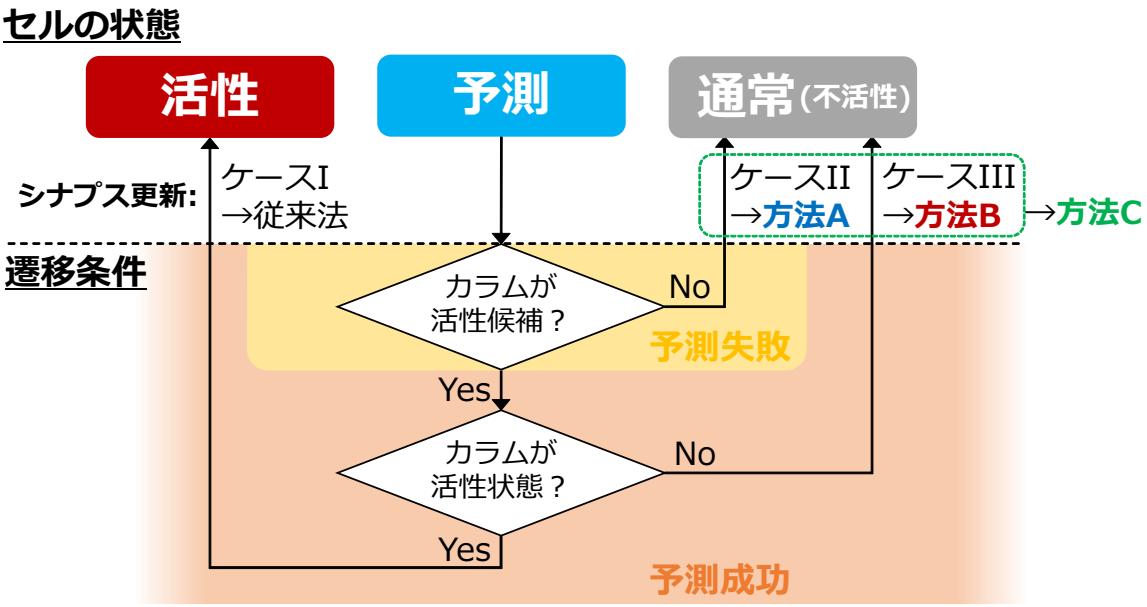


図 4.2: セルの予測状態からの状態遷移

おいて更新される．提案する CLA-IU は，従来の CLA における **Algorithm 4** の代わりに，**Algorithm 8** を適用する．従来の **Algorithm 4** からの差異を青字で示した．提案法では，**Algorithm 8** の 11 行目において，不活性セルのシナプスの永続値の更新が追加される．

4.2.1 3 種類の更新方法

不活性セルのシナプスの永続値を更新する三つの方法 A，B，C について述べる．

方法 A： ケース II のセルにおいてシナプスの永続値を更新する．ケース II の予測状態から通常状態への遷移は，属するカラムがその入力で活性状態になる可能性は全く無く，入力に対して的外れなカラム内に予測状態のセルが存在したために起きた遷移である．そのため，予測の失敗を意味する．よって方法 A は，不活性セルを予測状態にしたシナプスの永続値を弱める．具体的には，前時点 $t-1$ において活性状態だったセルとのシナプスの永続値を p_r^+ 下げる．また，前時点 $t-1$ において通常状態だったセルとのシナプスの永続値を p_r^- 上げる．不活性セル $r_{i,j}$ のシナプス $r_{i,j}.y_k$ の永続値 $r_{i,j}.y_k.p$ は，次式で更

Algorithm 8 時間プーリング (提案 CLA-IU)

```
1: セルの活性状態化
2: for i ← 1, 2, ..., n_c do
3:   if c_i.st = Active then
4:     for j ← 1, 2, ..., n_r do
5:       if r_{i,j}.st = Predictive then
6:         r_{i,j}.st ← Active
7:         Generate synapses from r_{i,j} to active cells at time t − 1
8:         セルのシナプスの更新
9:         for k ← 1, 2, ... do
10:          r_{i,j}.y_k.p ← Update performance value
11:          r_{i,j}.y_k.p ← Update performance value for inactive cells
12:        end for
13:      end if
14:    end for
15:    if Column c_i has no active cell then
16:      for j ← 1, 2, ..., n_r do
17:        r_{i,j}.st ← Active
18:      end for
19:      Generate synapses from a cell to active cells at time t − 1
20:    end if
21:  end if
22: end for
23: セルの予測状態化
24: for i ← 1, 2, ..., n_c do
25:   for j ← 1, 2, ..., n_r do
26:     if r_{i,j} has more than n_p synapses connected to active cells then
27:       r_{i,j}.st ← Predictive
28:     end if
29:   end for
30: end for
```

新する．

$$r_{i,j}.y_k.p = \begin{cases} r_{i,j}.y_k.p - p_r^+, & \text{if } r_{i,j}.y_k.a.st = \text{Active at } t-1, \\ r_{i,j}.y_k.p + p_r^-, & \text{otherwise } (r_{i,j}.y_k.a.st = \text{Normal at } t-1), \end{cases} \tag{4.1}$$

ここで，右辺の $r_{i,j}.y_k.p$ は更新前の永続値，左辺の $r_{i,j}.y_k.p$ は更新後の永続値を表すことに注意する．また，$r_{i,j}.y_k.a$ は，注目する不活性セル $r_{i,j}$ のシナプス $r_{i,j}.y_k$ に紐づいたセルである．$r_{i,j}.y_k.a.st$ は，そのセルの状態を示す．

方法 B： ケース III のセルにおいて，シナプスの永続値を更新する．ケース III の予測状態から通常状態への遷移は，活性状態になるカラムを活性候補のカラム群から選択する過程で起きる遷移である．オーバーラップ値 $c_i.ol$ が 0 ではないため，属するカラムが活性状態となる可能性はあったが，属するカラムの活性頻度が高いためにブースト値 $c_i.bst$ が

低くなった，もしくは他のカラムのブースト値 $c_i.bst$ が高いことで活性化スコア $c_i.as$ のランキングが変動し，以前に同じ値が入力された際と比べて活性状態となるカラム群が変化した可能性がある．そのため，それらのセルの予測自体は成功していると判断する．よって方法 B では，そのセルを予測状態にしたシナプスの接続を強める．具体的には，前時点 $t-1$ において活性状態だったセルとのシナプスの永続値を p_r^+ 上げる．また，前時点 $t-1$ において通常状態だったセルとのシナプスの永続値を p_r^- 下げる．不活性セル $r_{i,j}$ のシナプス $r_{i,j}.y_k$ の永続値 $r_{i,j}.y_k.p$ は，次式で更新する．

$$r_{i,j}.y_k.p = \begin{cases} r_{i,j}.y_k.p + p_r^+ & \text{if } r_{i,j}.y_k.a.st = \text{Active at } t-1, \\ r_{i,j}.y_k.p - p_r^- & \text{otherwise } (r_{i,j}.y_k.a.st = \text{Normal at } t-1), \end{cases} \tag{4.2}$$

ここで，右辺の $r_{i,j}.y_k.p$ は更新前の永続値，左辺の $r_{i,j}.y_k.p$ は更新後の永続値を表す．

方法 C： ケース II とケース III それぞれにおいて，方法 A と方法 B の両方を適用する．

4.2.2 期待される効果

ケース I のみでセルのシナプスの永続値を更新する従来の CLA は，シナプスが学習する機会が少なく，シナプスネットワークの構築が不十分になる．提案する方法 A，B，C では，ケース II とケース III において，セルのシナプスネットワークを更新する機会を新たに設ける．その結果，シナプスが学習する機会が増える．これにより，CLA におけるセルのシナプスネットワークの構築が促進される．ケース II に適用される方法 A は，予測の失敗をセルのシナプスネットワークの構築に反映する．これにより，誤った予測状態のセルの生成を抑える効果を期待できる．ケース III に適用される方法 B は，入力データにおけるノイズなどの影響で活性候補になったものの，活性状態には至らなかったカラムに含まれる予測状態のセル，すなわち予測に成功したセルをシナプスネットワークの構築に活用する．これにより，正しい予測状態のセルの生成を促進する効果を期待できる．方法 C は，方法 A と B を組み合わせることで，二つの方法の相乗効果を期待できる．

4.3 実験設定

提案する CLA-IU の効果を検証するため，従来 CLA と，従来 CLA に方法 A，方法 B，方法 C をそれぞれ適用した三つの提案 CLA-IU をあわせた 4 種類の CLA について，複

数のテスト入力データを用いて予測精度を比較する．また，提案法による CLA 内部のセルの状態遷移の変化を確認するため，そのセル数も計数する．

4.3.1　テスト入力データ

本章では，2 種類のテスト入力データを用いる．まず，以下の式で表される正弦波を用いる．

$$X_1(t) = \sin\left(\frac{(t-1)\cdot\pi}{50}\right), \tag{4.3}$$

正弦波の周期は，$t = 100$ である．次に，以下の式で表される入力データを用いる．

$$X_2(t,\delta) = X_1(t) + noise(\delta), \tag{4.4}$$

ここで，$noise(\delta)$ は $[-\delta, +\delta]$ の値域をとる一様乱数である．$X_2(t,\delta)$ の場合，$noise(\delta)$ を含む $X_1(t)$ を予測することを目的とする．また，$\delta = \{0.01, 0.02, 0.03\}$ の 3 種類を用いる．この入力では，ノイズにより正弦波の周期ごとに現れる入力が変化するため，予測状態から通常状態へ遷移するセル数が増えると考えられる．これに対する提案法の効果を検証する．

実験では，ある時点 t のテスト入力データを入力として，次時点 $t+1$ のテスト入力データを予測する，ということを各入力時点に対して繰り返した．

4.3.2　パラメータ

CLA のパラメータとして，学習器のカラム数は $n_c = 2,048$ 個，各カラムそれぞれのセル数は $n_r = 32$ 個にした．各実験における入力時点は $t \in [1, 10^4]$ とした．空間プーリングのパラメータとして，各カラムのシナプス生成半径は $r = 16$ とし，シナプスの生成確率は $d = 0.8$ とした．各入力時点 t における活性カラムの最大数は $n_{ac} = 40$ に設定した．永続値の増加量は $p_c^+ = 0.05$，減少量は $p_c^- = 0.025225$ とした．また，各シナプスの接続・切断を決める閾値は $\theta_c = 0.1$ とした．時間プーリングのパラメータとして，永続値の増減量はそれぞれ $p_r^+ = 0.1$ と $p_r^- = 0.1$，各シナプスの接続・切断を決める永続値の閾値は $\theta_r = 0.5$ に設定した．また，予測セルを決定する際に用いる，必要な活性セルとの接続シナプス数は $n_p = 15$ にした．本章では，全てのパラメータについて，NuPIC のラ

イブラリ [8] と同一の値を用いた．

4.3.3 評価指標

予測精度を評価するための指標として予測誤差を用いる．予測誤差は，予測値 $\overline{X}(t+1)$ と実際の入力値 $X(t+1)$ の差である．以下の式で 100 入力時点ごとの予測誤差を合計する．

$$e(t) = \sum_{\tau=t-99}^{t} |\overline{X}(\tau+1) - X(\tau+1)|. \tag{4.5}$$

$e(t)$ は $t \in \{100, 200, 300, 400, \ldots, 10^4\}$ の間隔で算出する．

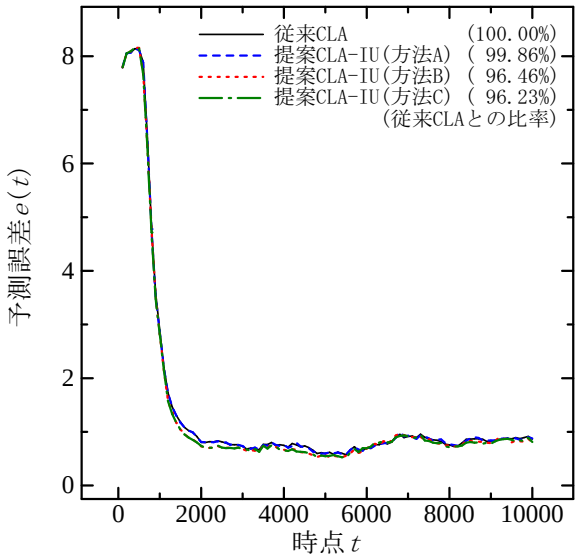
また，**図 4.2** に示したケース II とケース III の状態遷移セル数を計数する．ケース II は，予測の失敗によりセルが予測状態から通常状態に遷移するため，状態遷移セル数が少ないときに良い結果であると判断する．一方，ケース III は，活性候補のカラムが活性状態にならなかったことで，その内部の予測状態のセルが通常状態に遷移するため，予測自体は成功しているといえる．そのため，状態遷移セル数が多いときに良い結果であると判断する．

4.4 実験結果と考察

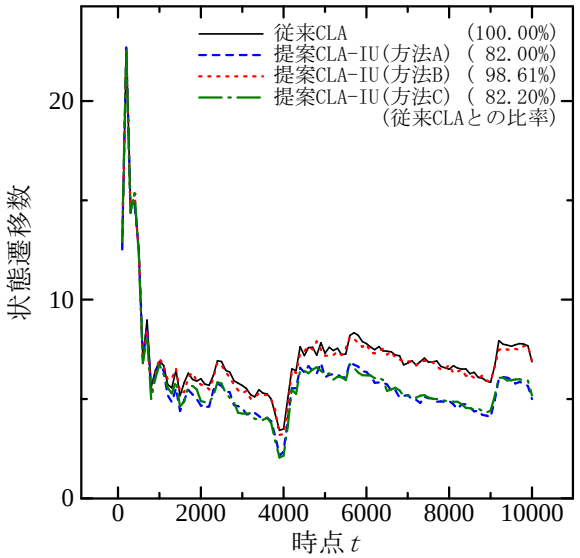
4.4.1 ノイズを含まない時系列データにおける結果

ノイズを含まない入力 $X_1(t)$ における結果を**図 4.3** に示す．予測誤差の推移を**図 4.3 (a)** に示す．4 種類の CLA について，予測誤差の合計値の従来法比をグラフの凡例に付記した．この結果から，三つの提案 CLA-IU が従来 CLA より低い予測誤差を示すことがわかる．また，方法 C が，4 種類の CLA の中で最も低い予測誤差を達成することがわかる．以上の結果より，ケース II とケース III においてシナプスを更新する提案法が，予測精度の向上に貢献することが明らかになった．

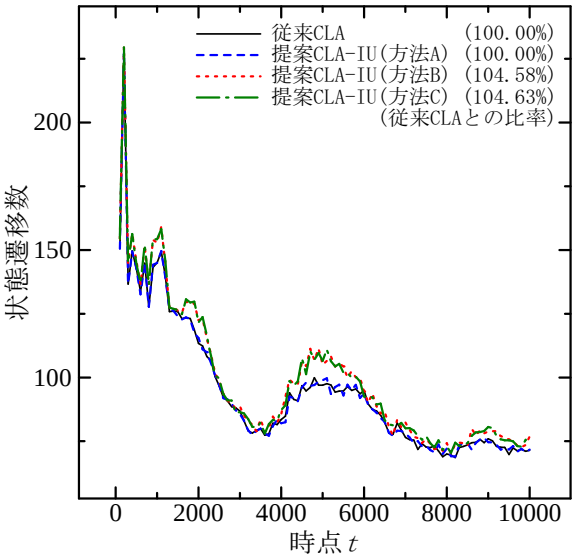
ケース II における状態遷移数の推移を**図 4.3 (b)** に示す．ケース II の状態遷移は，予測の失敗を意味するため，状態遷移数が少ない方が良い．この結果から，三つの提案 CLA-IU が，従来 CLA より少ない状態遷移数を示すことがわかる．また，ケース II に対応した方法 A の状態遷移セル数が，方法 B より少ないことがわかる．以上の結果より，



(a) 予測誤差



(b) ケース II の状態遷移セル数

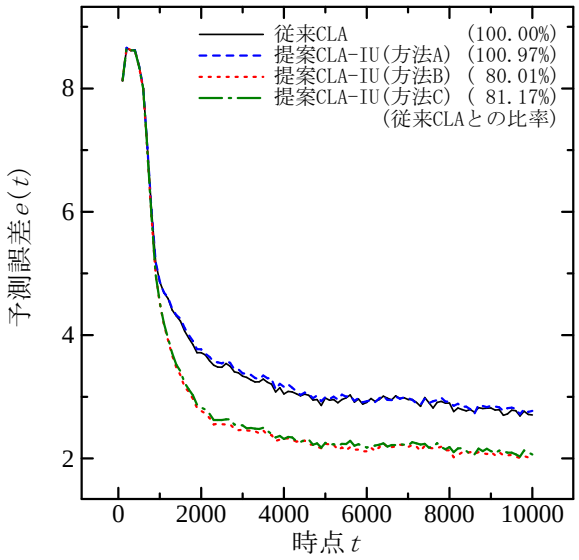


(c) ケース III の状態遷移セル数

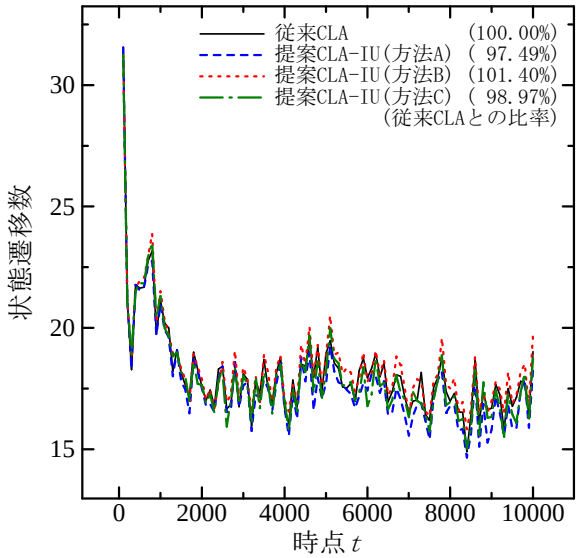
図 4.3: ノイズを含まない入力 $X_1(t)$ における結果

予測が失敗したときにシナプス接続を弱める方法 A が，予測に失敗したセル数を減らすといえる．

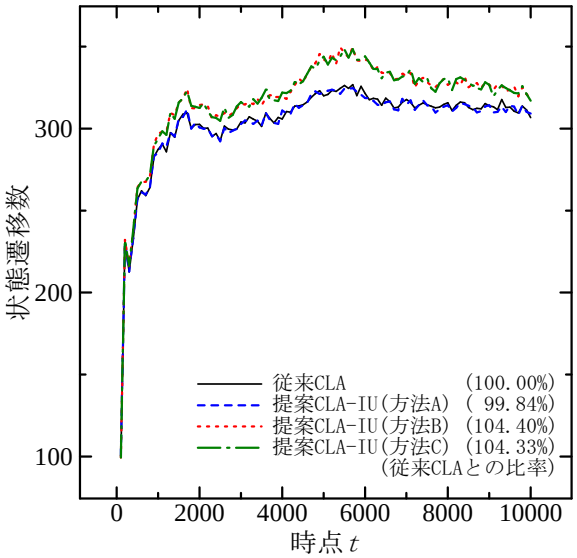
ケース III の状態遷移数の推移を**図 4.3 (c)** に示す．ケース III の状態遷移は，予測状態のセルの属するカラムが活性候補になったものの，最終的に活性状態に遷移しなかったために起きる．そのため，セルの予測自体は成功したといえる．よって，状態遷移数は多



(a) 予測誤差



(b) ケース II の状態遷移セル数

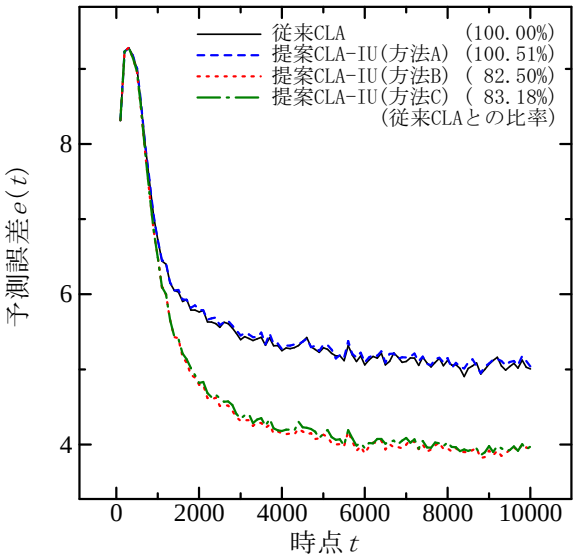


(c) ケース III の状態遷移セル数

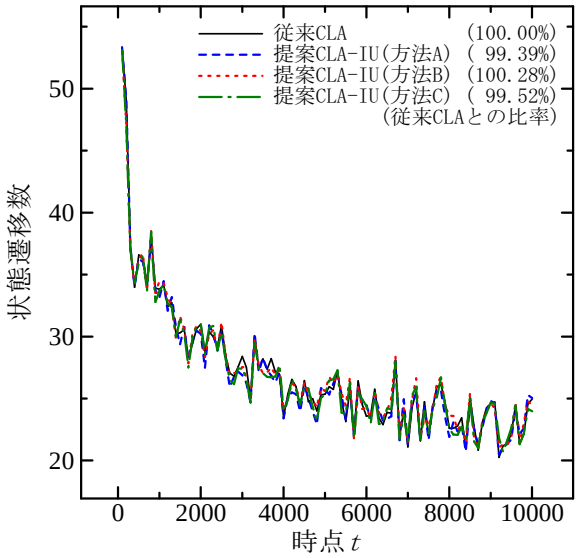
図 4.4: ノイズ $\delta = 0.01$ を含む入力 $X_2(t, 0.01)$ における結果

い方が良いと判断する．この結果から，提案 CLA-IU の方法 B と C の状態遷移数が，従来 CLA と方法 A より多いことがわかる．以上の結果から，ケース III の活性候補のカラムに属する予測自体は成功したセルにおいて，シナプス接続を強めることで，方法 B は予測に成功したセルを増やすといえる．

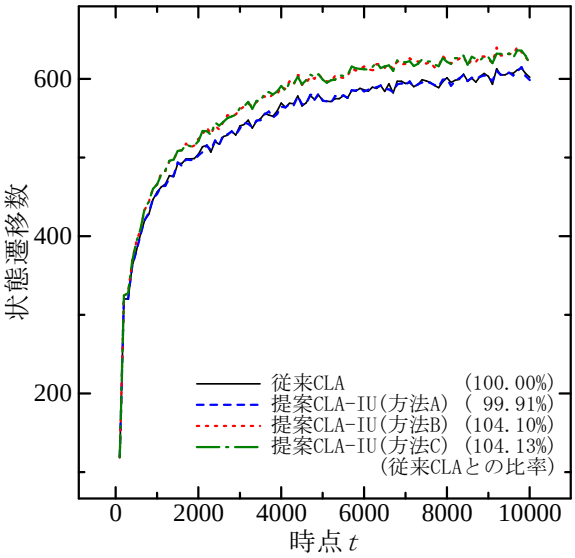
以上より，提案 CLA-IU の方法 A は予測に失敗したセルを減らし，方法 B は予測に成



(a) 予測誤差



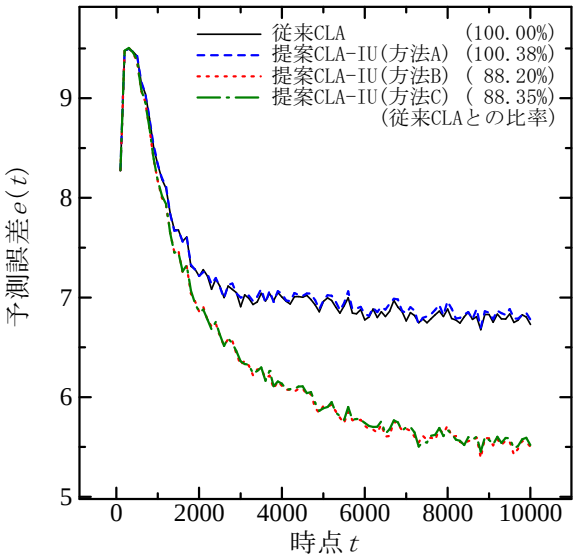
(b) ケース II の状態遷移セル数



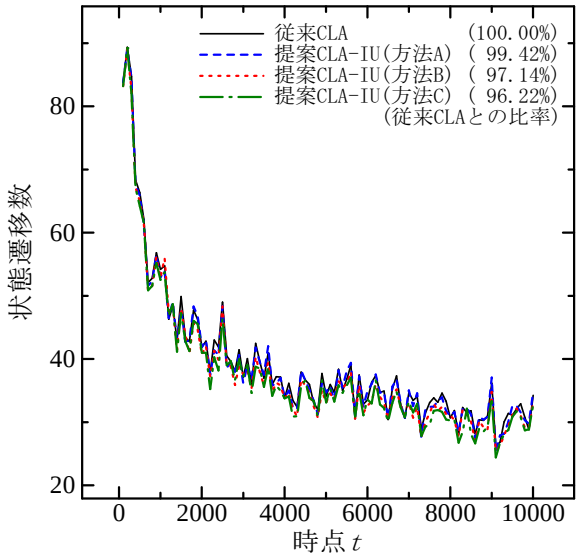
(c) ケース III の状態遷移セル数

図 4.5: ノイズ $\delta = 0.02$ を含む入力 $X_2(t, 0.02)$ における結果

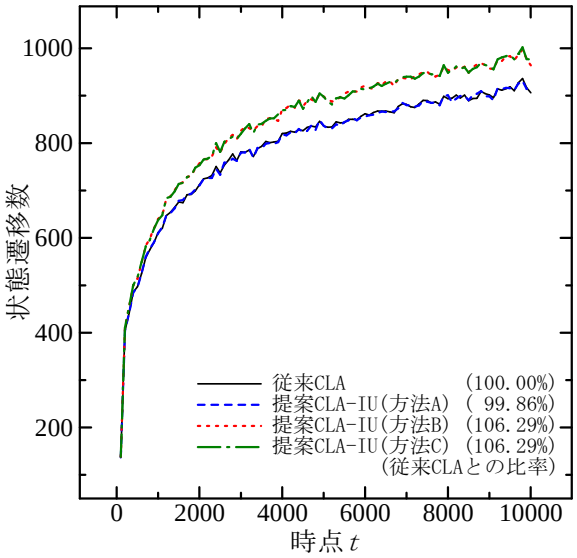
功したセルを増やすことがわかった．また，方法 C は，これらを組み合わせることでノイズを含まない入力 $X_1(t)$ において，最も低い予測誤差を達成したといえる．



(a) 予測誤差



(b) ケース II の状態遷移セル数



(c) ケース III の状態遷移セル数

図 4.6: ノイズ $\delta = 0.03$ を含む入力 $X_2(t, 0.03)$ における結果

4.4.2　ノイズを含む時系列データにおける結果

まず，ノイズ $\delta = 0.01$ を含む入力 $X_2(t, 0.01)$ における結果を**図 4.4** に示す．予測誤差の推移を**図 4.4 (a)** に示す．この結果から，提案 CLA-IU の方法 B が予測精度の改善

に大きく貢献することがわかる．次に，ケース II の状態遷移数の推移を**図 4.4 (b)** に示す．この結果から，提案 CLA-IU の方法 A が予測の失敗を意味するケース II の状態遷移数を減少させる傾向が見て取れる．しかし，この結果による予測誤差への影響は見られない．そして，ケース III の状態遷移数の推移を**図 4.4 (c)** に示す．この結果から，提案 CLA-IU の方法 B が予測の成功を意味するケース III の状態遷移数を増加させることがわかる．この結果は，**図 4.4 (a)** において，予測誤差の減少に貢献することがわかる．

次に，ノイズ $\delta = 0.02$ を含む入力 $X_2(t, 0.02)$ における結果を**図 4.5**，ノイズ $\delta = 0.03$ を含む入力 $X_2(t, 0.03)$ における結果を**図 4.6** に示す．**図 4.5 (a)** と**図 4.6 (a)** より，$X_2(t, 0.01)$ を入力としたときと同様に，提案 CLA-IU の方法 B が予測誤差を低減させることがわかる．また，**図 4.5 (b)** と**図 4.6 (b)** より，提案 CLA-IU の方法 A がケース II の状態遷移数を減らすことがわかる．しかし，ノイズを含まない $X_1(t)$ を入力とした**図 4.3 (b)**，および小さなノイズ $\delta = 0.01$ を含む $X_2(t, 0.01)$ を入力とした**図 4.4 (b)** よりも，予測の失敗を意味する状態遷移数を減らす効果は弱いことがわかる．これは，予測を失敗する要因として，ノイズによって値が変動した影響が強くなり，単純に予測を失敗したと判断することが逆効果となるケースが増えた，と考えられる．一方，最も大きいノイズ $\delta = 0.03$ を含む入力 $X_2(t, 0.03)$ における結果である**図 4.6 (b)** では，方法 B が状態遷移数の減少に貢献し，方法 C が最も状態遷移数が少ないことがわかる．これは，ノイズが原因で予測状態から活性状態にならなかったセルでも学習が進むことで，予測の成功に貢献するシナプスの構築が促進され，結果的に予測の失敗を意味するシナプスの構築を阻害したと考えられる．そして，**図 4.5 (c)** と**図 4.6 (c)** より，$X_2(t, 0.01)$ を入力としたときと同様に，提案 CLA-IU の方法 B がケース III の状態遷移数を増やすことがわかる．この予測の成功を意味する状態遷移を増やすことで，**図 4.5 (a)** と**図 4.6 (a)** における予測精度の改善に貢献したと考えられる．

以上より，ノイズを含む時系列データにおいても，提案 CLA-IU の方法 A は予測の失敗を減らし，方法 B は予測の成功を増やす効果があることが確かめられた．また，方法 B が予測精度の改善に貢献することが明らかになった．

4.5 結言

本章では，従来の CLA におけるセルのシナプスの取り扱いに着目し，セルのシナプスネットワークの構築を促進するため，不活性セルのシナプス更新法を提案した．提案法は，予測状態から通常状態に遷移する不活性セルにおいて，シナプスの永続値を更新する．方法 A は，予測に失敗した場合，予測状態を作り出したシナプスの永続値を弱める．方法 B は，予測に成功したものの次時点のデータ表現に利用されなかった場合，予測状態を作り出したシナプスの永続値を強める．方法 C は，方法 A と B を同時に実行する．これらの方法によって，従来 CLA と比べてシナプスの永続値を更新する機会が増加し，シナプスネットワークの構築が促される．テスト時系列データを用いた実験の結果，提案 CLA-IU の方法 A が，シナプスの永続値を弱めることで予測に失敗したセルを減らすことを示した．また，方法 B が，シナプスの永続値を強めることで予測に成功したセルを増やすことを示した．ノイズを含まない時系列データにおいて，方法 A と B が予測精度の改善に貢献することを示した．ノイズを含む時系列データにおいては，特に方法 B が CLA の予測精度を改善することを示した．

今回は，各時点で独立したノイズを付与した時系列データを取り扱った．実世界問題においては，時間によってノイズの大きさが変化する時系列データがある．実世界の電力消費量でいえば，昼の活動時間帯は日々の変動量が大きく，夜の非活動時間帯は日々の変動量が小さいといったことがある．時間によってノイズの大きさが変化する時系列データに対する性能検証と方法の構築については，今後の課題とする．

第 5 章

予測器の多層化

人間の大脳新皮質には，階層構造がある．層と層との相互作用が，予測の補完や補正に寄与していると考えられる [4]．一方，従来の CLA は，1 層構造であり，予測の補完や補正機能はない．人間の大脳新皮質にあると考えられる予測の補正を CLA で実現できれば，予測精度のさらなる改善が期待できる．本章では，予測を補正することで，予測精度を向上させる方法を構築する．予測を補正するために，CLA の既存の空間プーリングによって，時系列データの内部表現の特徴を抽出して補正に利用する．このために，予測器を積み重ねる．提案法は，CLA の予測器を二つ重ね，上位の予測器が，下位の予測器の予測を補正するように相互作用させる．CLA の予測器において，本章が取り組む部分を**図 5.1** に赤色で示した．本章は，CLA における予測器を積み重ね，相互に連携させることに注目し，CLA-DL (CLA with Double Layers) を提案する．

5.1 概要

従来の CLA では，入力の複雑さによらず予測誤差が増大することがある．その一因として，予測器内部のデータ表現が不安定になるケースが挙げられる．例えば，従来の CLA では，活性状態と接続状態のシナプスを n_p 本以上持つセルを予測状態にする．この予測セル数が活性状態のカラム数 n_{ac} と同数で安定したとき，活性状態となる各カラムにそれぞれひとつずつ予測セルが存在するため，一意の予測を実現したといえる．一方，一意の予測を実現できない場合，予測セルを持たない活性状態のカラムが現れ，その内部のセル

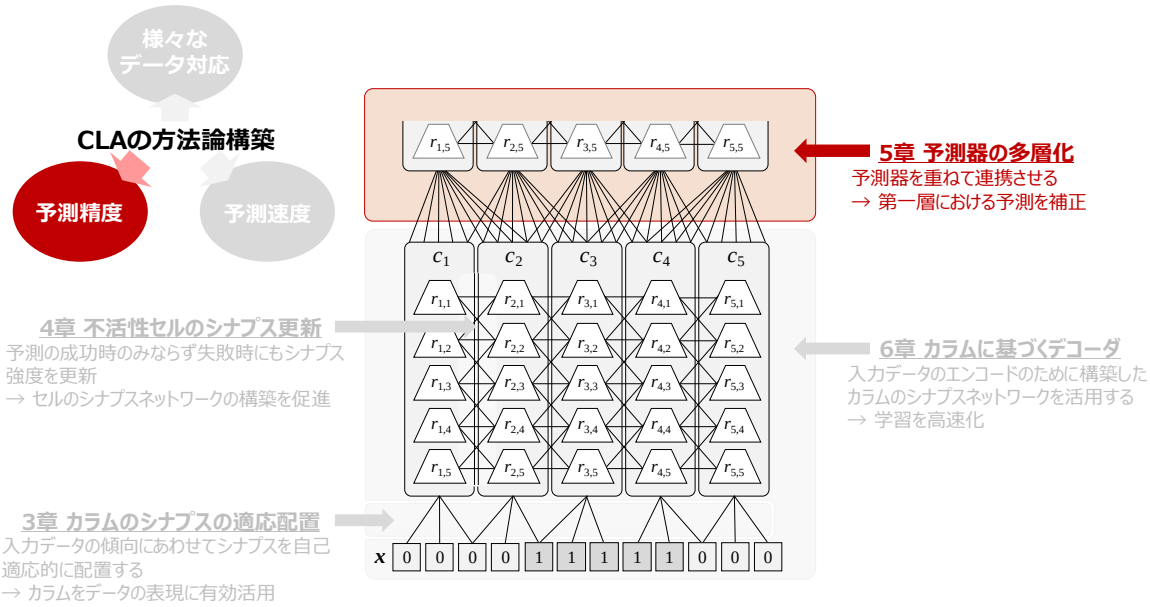


図 5.1: 5 章の位置づけ

Algorithm 9 提案する CLA-DL

1: $\mathcal{C} \leftarrow$ Initialization ▷ **Algorithm 2**
2: $\mathcal{C}_2 \leftarrow$ Initialization ▷ **Algorithm 2**
3: **for each** time-step t **do**
4: $X(t) \leftarrow$ Receiving input value
5: $\boldsymbol{x} \leftarrow$ Encoding $(X(t))$
6: $\mathcal{C} \leftarrow$ Spatial pooling $(\mathcal{C}, \boldsymbol{x})$ ▷ **Algorithm 3**
7: $\mathcal{C} \leftarrow$ Temporal pooling - cell activation $(\mathcal{C})$ ▷ **Algorithm 10**
8: 第二層における学習
9: $\mathcal{C}_2 \leftarrow$ Spatial pooling $(\mathcal{C}_2, \mathcal{C})$ ▷ **Algorithm 3**
10: $\mathcal{C}_2 \leftarrow$ Temporal pooling $(\mathcal{C}_2)$ ▷ **Algorithm 4**
11: $\mathcal{C}' \leftarrow$ Column-based Decoder $(\mathcal{C})$
12: $\mathcal{C} \leftarrow$ Temporal pooling - cell predictivation$(\mathcal{C}_2, \mathcal{C}')$ ▷ **Algorithm 11**
13: $\overline{X}(t+1) \leftarrow$ Decoding $(\mathcal{C})$
14: **end for**

が全て活性状態となる．その結果，予測状態のセル数も増減して不安定となり，予測誤差が増大する．これを解決する方法のひとつとして，本章では CLA の予測器を複数用意して重ね，内部の予測表現を補正する方法を検討する．

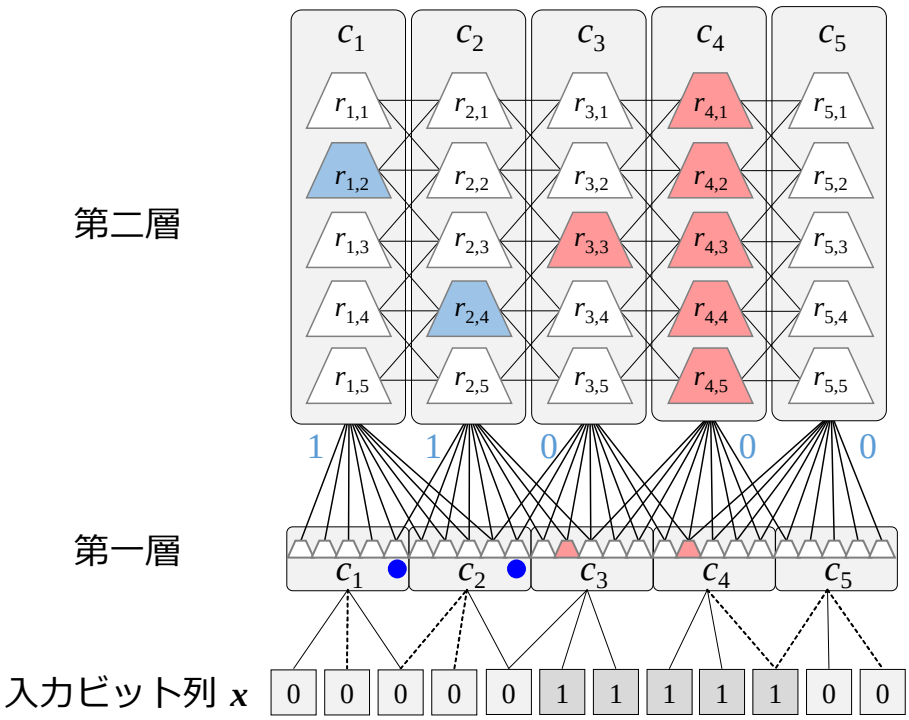


図 5.2: CLA-DL

5.2 方法

5.2.1 予測器

図 5.2 に提案する CLA-DL の予測器の構成を示す．提案法は，**図 2.1** に示した従来の CLA のリージョンを重ねて構成する．下部の層を第一層と呼び，上部の層を第二層と呼ぶ．図のように，第一層と第二層のそれぞれの予測器の構成は，同一である．

第一層と第二層は，第二層におけるカラムのシナプスに相当する層間シナプスで接続される．各層間シナプスは，第二層のカラムから第一層のセルに紐づけられる．層間シナプスは，第一層のセルの情報を第二層へ伝達する．また，層間シナプスは，第二層の予測情報を第一層へ伝達する役割も持つ．CLA-DL において，第一層の役割は従来の CLA と類似している一方，第二層は第一層のセルの状態に影響を及ぼす役割を持つ．

Algorithm 10 時間プーリング-セルの活性状態化

```
1: セルの活性状態化
2: for i ← 1, 2, ..., n_c do
3:   if c_i.st = Active then
4:     for j ← 1, 2, ..., n_r do
5:       if r_{i,j}.st = Predictive then
6:         r_{i,j}.st ← Active
7:         Generate synapses from r_{i,j} to active cells at time t − 1
8:         セルのシナプスの更新
9:         for k ← 1, 2, ... do
10:          r_{i,j}.y_k.p ← Update performance value
11:        end for
12:      end if
13:    end for
14:    if Column c_i has no active cell then
15:      for j ← 1, 2, ..., n_r do
16:        r_{i,j}.st ← Active
17:      end for
18:      Generate synapses from a cell to active cells at time t − 1
19:    end if
20:  end if
21: end for
```

Algorithm 11 時間プーリング-セルの予測状態化

```
1: セルの予測状態化
2: for i ← 1, 2, ..., n_c do
3:   for j ← 1, 2, ..., n_r do
4:     if r_{i,j} is predicted on C′ then
5:       θ_r ← 0.5 · θ_r
6:       if r_{i,j} has more than 0.5 · n_p synapses connected to active cells then
7:         r_{i,j}.st ← Predictive
8:       end if
9:     else
10:      if r_{i,j} has more than 2 · n_p synapses connected to active cells then
11:        r_{i,j}.st ← Predictive
12:      end if
13:    end if
14:  end for
15: end for
```

5.2.2　アルゴリズム

提案する CLA-DL の擬似コードを **Algorithm 9** に示す．変更点は，青色で表記する．各時点 t において，4–5 行目で，第一層は入力データ x を受け取る．まず，6 行目で，第一層において，空間プーリングによってカラム群を活性状態にする．また，7 行目で，時

間プーリングによってセル群を活性状態にする．アルゴリズムは，**Algorithm 10** に示す通りである．これらの処理は，従来の CLA における処理と同じである．**図 5.2** の例では，第一層における空間プーリングの結果として，カラム c_3 と c_4 を活性状態にして入力データ x を表現している．また，第一層における時間プーリングの結果として，セル $r_{3,2}$ と $r_{4,2}$ を活性状態にしてデータの文脈を表現している．次に，第一層における活性セルのパターンを層間シナプスを用いて第二層へ伝達する．この手順では，第一層における通常状態のセルを 0，活性状態のセルを 1 として扱い，擬似コードの 9 行目で第二層において従来の CLA と同様の操作で空間プーリングを実行する．これにより，第一層における活性セルのパターンを第二層における活性状態のカラム集合で表現する．**図 5.2** の例においては，第一層のセル $r_{3,2}$ と $r_{4,2}$ を 1，他のセルを 0 とする．それを第二層が受け取り，カラム c_3 と c_4 が活性状態となる．そして，擬似コードの 10 行目で第二層において従来の CLA と同様の操作で時間プーリングを実行し，あるセル群を活性状態とした後にあるセル群を予測状態とする．**図 5.2** では，第二層のカラム c_3 と c_4 のセルが活性状態となり，その後にセル $r_{1,2}$ と $r_{2,4}$ が予測状態となる例を示している．最後に，11 行目で第二層の予測セル情報を層間シナプスを用いて第一層へ伝達し，12 行目で第一層において予測セルを決定する際にその第二層からの予測を考慮する．本章では，この操作をフィードバックと呼ぶ．

5.2.3 フィードバック

提案する CLA-DL は，第二層からの予測情報を考慮して，第一層で予測状態となるセル群を決定する．第二層から予測された第一層のセルについて，CLA-DL はそのセルが予測状態となる条件を緩和する．**図 5.2** の例においては，第一層のセル $r_{1,5}$ と $r_{2,5}$ が第二層から予測されている．提案する CLA-DL は，これらのセルが予測状態となる条件を緩和し，他のセルについて条件を強化する．CLA-DL は，二つの方法で第一層において予測状態となるセルを補正する．フィードバックを含むセルの予測状態化について，擬似コードを **Algorithm 11** に示す．

まず，二つの方法で第一層においてセルが予測状態となる条件を緩和する．一つ目の方法として，**Algorithm 11** の 5 行目に示す通り，セルのシナプスの接続閾値 θ_r を緩和する．提案する CLA-DL は，**図 5.2** における第二層から予測された第一層のセル $r_{1,5}$ と

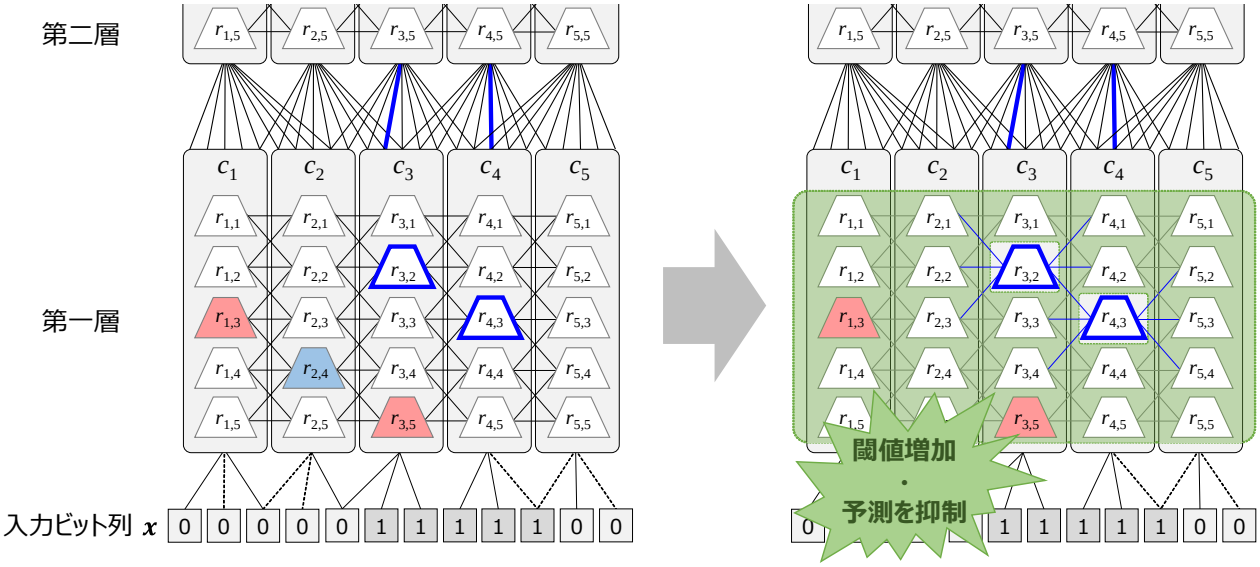


図 5.3: 抑制フィードバック

$r_{2,5}$ において，シナプスの接続閾値を半減する．第二層から予測されていない他のセルについては，従来の接続閾値である θ_r が使用される．各セルが予測状態となるためには，活性状態のセルと接続されたシナプスを n_p 本以上有する必要があるが，接続閾値を半分に減少させることで接続状態であるシナプスが増え，そのセルが予測状態となる可能性を上げることができる．二つ目の方法として，5 行目に示す通り，予測状態となるための閾値をより直接的に減少する．具体的には，提案する CLA-DL は予測状態となるために必要な活性状態のセルとの接続シナプス数 n_p を減らす．**図 5.2** に示すような第二層から予測された第一層のセル $r_{1,5}$ と $r_{2,5}$ において，必要な接続シナプス数を半減する．これにより，もし活性状態のセルとの接続シナプス数が n_p より少ないとしても，第二層から予測された第一層のセルを予測状態になりやすくする．

次に，第二層から予測されていない第一層のセルにおいて，CLA-DL はそれらのセルが予測状態となるための条件を強化する．具体的には，擬似コードの 10 行目に示す通り，予測状態となるために必要な活性状態のセルとの接続シナプス数 n_p を増やす．例を**図 5.3** に示す．第二層から予測された第一層のセル $r_{3,2}$ と $r_{4,3}$ 以外において，必要なシナプス数 n_p を倍化する．これにより，第一層において間違った可能性のある予測を抑制する

ことができる．フィードバックを導入した後の予測に必要な接続シナプス数 n_p は，以下のようになる．

$$n_p \leftarrow \begin{cases} 0.5 \cdot n_p, & \text{if 第二層から予測されたセル,} \\ 2 \cdot n_p, & \text{otherwise,} \end{cases} \tag{5.1}$$

5.2.4 層間シナプスの配置

第二層のカラムのシナプスにあたる各層間シナプスは，第一層のセルと第二層のカラムを紐づける．提案する CLA-DL は，この層間シナプスを時点が進むごとに動的に配置する．具体的には，第二層の空間プーリングにおいて活性状態となるカラム数が n_{ac} より少ないとき，活性頻度の低い第二層のカラムと第一層の活性状態のセル群の間に層間シナプスを新しく配置する．

この動的な層間シナプスの配置は，10^4 入力時点が経過した後に開始する．これは，下層の内部状態が不安定なときに配置を開始しても，有効なシナプスネットワークを構築できないと考えられるためである．

5.2.5 期待される効果

提案する CLA-DL において，フィードバックによって学習効率が改善し，CLA の学習速度を向上する効果が期待される．また，もし第一層における予測が適切に機能しないとき，第二層からのフィードバックによって第一層の予測が補正され，予測精度が改善する効果が期待される．

5.3 実験設定

提案する CLA-DL の効果を検証するため，様々な入力における予測性能を比較する．また，学習器内部のセルの状態遷移数を計数することで，フィードバックによって予測がどのように補正されるのか確認する．

5.3.1　アルゴリズム

まず, CLA における性能差を比較する．従来手法として, 従来 CLA[8], 簡素型 CLA[67] を用いる．また，提案法として多層の CLA である提案 CLA-DL を用いる．提案 CLA-DL のベースとして，簡素型 CLA を用いる．簡素型 CLA は，3.2.1 節に記した方法で，初期永続値の固定化とカラムのシナプス配置の集約によって，カラムのシナプスの初期配置を決定論的にする方法である．各カラムのセル数は，従来手法である従来の CLA および簡素型 CLA では $n_r = 16$，提案法である提案 CLA-DL では第一層，第二層ともに $n_r = 16$ とした．

次に，関連手法も含めた方法での性能を比較する．比較手法として，提案法である提案 CLA-DL と，簡素型 CLA を用いた．各カラムのセル数は提案 CLA-DL では第一層，第二層ともに $n_r = 16$ とし，簡素型 CLA は $n_r = 16$ と $n_r = 32$ の 2 種類を使用した．これにより，提案 CLA-DL における第一層のみでのセル数の従来法，第一層と第二層をあわせたセル数の従来法の両方との比較ができる，また，関連手法として三つの RNN をベースとした LSTM を用いる．それぞれ，Adam，RMSprop，SGD をそれぞれネットワーク最適化アルゴリズムとして用いた．

5.3.2　ベンチマーク時系列データ

周期的なベンチマーク時系列データとして，正弦波 $X_s(t)$ と正弦波の合成波 $X_c(t, m)$ を使用する．これらの時系列データは以下の式で定義される：

$$X_s(t) = \frac{1}{2} \cdot \sin\left(\frac{(t-1)\cdot\pi}{50}\right) + \frac{1}{2}, \tag{5.2}$$

$$X_c(t, m) = \frac{1}{2} \cdot \sum_{k=1}^{m} \frac{1}{2k-1} \cdot \sin\left(\frac{(t-1)\cdot(2k-1)\cdot\pi}{50}\right) + \frac{1}{2}, \tag{5.3}$$

正弦波の合成波において，次数 $m = \{2, 3, 4, 5\}$ を用いる．次数が増えるほど，類似したパターンが増えて予測が難しくなる．入力時点は $t \in [1, 10^5]$ とする．入力の値域は $[X^{\min}, X^{\max}] = [-0.01, 1.01]$ に設定する．

実験では，ある時点 t のベンチマーク時系列データを入力として，次時点 $t+1$ のベンチマーク時系列データを予測する，ということを各入力時点に対して繰り返した．

表 5.1: 実験における CLA のパラメータ

名前	値
データビット列の長さ n	421
チャンクの長さ w	21
カラム数 n_c	2048
カラムのシナプス数 n_{cy}	21
カラムのシナプスの接続閾値 θ_c	0.1
活性状態となるカラム数 n_{ac}	40
カラムのシナプスの永続値増加量 p_c^+	0.05
カラムのシナプスの永続値減少量 p_c^-	0.025225
セルのシナプスの接続閾値 θ_r	0.5
セルが予測状態となるために必要な活性状態のセルとの接続数 n_p	15
セルのシナプスの永続値増加量 p_r^+	0.1
セルのシナプスの永続値減少量 p_r^-	0.1

5.3.3 評価指標

時系列データの予測精度を評価する指標として，以下の式で定義される予測誤差 $e(t)$ を用いる．

$$e(t) = \sum_{\tau=t-99}^{t} |\bar{X}(\tau+1) - X(\tau+1)|, \tag{5.4}$$

ここで，$\bar{X}(t+1)$ は次時点 $t+1$ の予測値，$X(t+1)$ は次時点 $t+1$ の真値を表す．$e(t)$ は 100 時点ごとの真値と予測値の差の合計となる．$e(t)$ が小さいほど，予測精度が良いといえる．本章では，$e(t)$ を $t \in \{100, 200, 300, 400, \ldots, 10^5\}$ の範囲で計算する．

また，提案法の効果を検証するため，CLA において時点 $t \in [99,001, 100,000]$ での予測状態のセル数を確認する．このセル数が活性状態となるカラム数 n_{ac} に近いほど，次時点でのパターンを予測できている，すなわち予測が一意であるといえる．提案法である提案 CLA-DL においては，入力を受け取る第一層でのセル数を確認した．

5.3.4　パラメータ

三つの CLA において，データビット列の長さは $n = 421$ ビット，チャンクの長さは $w = 21$ ビットに設定する．空間プーリングにおいて，カラム数は $n_c = 2048$，カラムのシナプス数は $n_{cy} = 21$ とする．カラムのシナプスの接続閾値は $\theta_c = 0.1$，各時点で活性状態となるカラム数は $n_{ac} = 40$，永続値の増減量はそれぞれ $p_c^+ = 0.05$，$p_c^- = 0.025225$ とした．時間プーリングにおいて，セルのシナプスの接続閾値は $\theta_r = 0.5$，予測状態となるために必要な活性状態のセルとの接続シナプス数は $n_p = 15$ とした．また，永続値の増減量はそれぞれ $p_r^+ = 0.1$，$p_r^- = 0.1$ とした．提案 CLA-DL において，第一層，第二層ともに上記のパラメータを使用する．**表 5.1** に実験で使用した CLA のパラメータを示す．本章では，第 3 章と同様に，予測のビット単位での差異を判別しやすくするために $n = 421$，カラムが単一の入力ビット列に対応しやすくするために $n_{cy} = 21$ とした．他のパラメータについては，NuPIC のライブラリ [8] で設定されたデフォルトのパラメータを使用した．

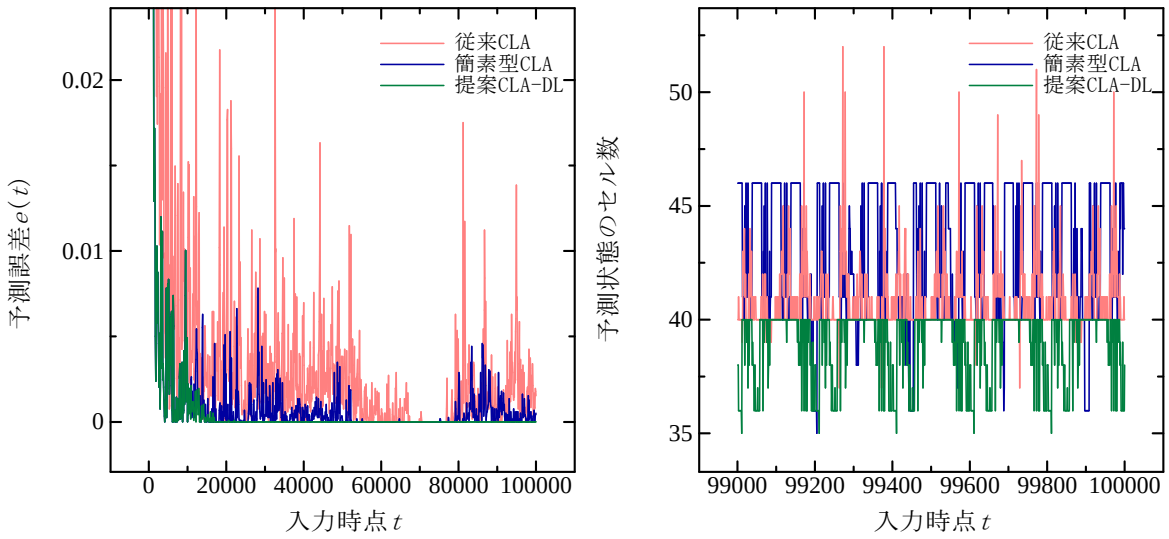
関連手法として，ネットワーク最適化アルゴリズムに Adam[64]，RMSprop[65]，SGD をそれぞれ適用した三つの LSTM を用いる．実装は Keras[68] によるものを用いる．第一層のニューラルネットワーク層は全結合層で，第二層の LSTM 層は活性化関数に tanh 関数を用いる．出力層は全結合のニューラルネットワーク層とする．損失関数として，平均二乗誤差を使用する．

5.4　実験結果と考察

5.4.1　CLA における結果

CLA-DL の効果を正弦波 $X_s(t)$ を用いて検証する．予測誤差の推移を**図 5.4 (a)**，CLA の時点 $t \in [99,001, 100,000]$ における予測状態のセル数を**図 5.4 (b)** に示す．

まず**図 5.4 (a)** より，簡素型 CLA の予測誤差が従来 CLA より低く安定することがわかる．また，提案法である提案 CLA-DL の予測誤差が簡素型 CLA よりも低く，CLA の中で最も高い予測精度を達成することがわかる．この結果から，正弦波において多層化は予測精度を改善する効果が示されたといえる．



(a) 予測誤差の推移　　(b) 予測状態となるセル数

図 5.4: 正弦波 $X_s(t)$ における CLA の比較

次に，**図 5.4 (b)** より，従来 CLA の予測状態のセル数は不安定で，そのセル数が 50 個を超える時点が存在することがわかる．これは，従来 CLA の予測が不安定で，誤差が増大する一因になると考えられる．また，簡素型 CLA における遷移は安定しているが，そのセル数は 46 個となることが多い．この安定性により予測精度が向上したが，予測セル数が活性状態となるカラム数 $n_{ac} = 40$ より多く，その予測は一意ではない時点が多く存在するため，予測誤差を 0 に収束させることはできなかったと考えられる．これに対して，提案 CLA-DL は予測状態となるセル数が活性状態となるカラム数と同数の 40 個となる時点が多いことがわかる．この結果から，提案 CLA-DL は時系列データの内部表現の特徴を抽出した第二層から第一層を補正できることがわかる．また，補正によって一意の予測を実現し，予測精度を改善したと考えられる．

5.4.2　CLA と LSTM における結果

提案 CLA-DL の性能について，様々な入力を用いて従来手法と比較する．正弦波 $X_s(t)$ を入力としたときの予測誤差の推移を**図 5.5** に示す．また，正弦波の合成波 $X_c(t, 2)$ を入力としたときの予測誤差の推移を**図 5.6**，$X_c(t, 3)$ の結果を**図 5.7**，$X_c(t, 4)$ の結果を**図 5.8**，$X_c(t, 5)$ の結果を**図 5.9** にそれぞれ示す．グラフの縦軸は対数目盛となっている．

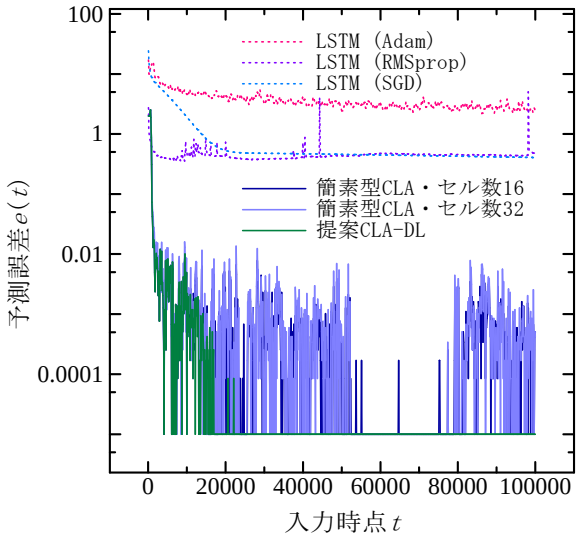


図 5.5: 正弦波 $X_s(t)$

表示のため，予測誤差 $e(t)$ が 10^{-5} 以下の場合，グラフ内では 10^{-5} に丸めて表示する．

まず**図 5.5** より，三つの CLA の予測誤差は，異なる最適化アルゴリズムを用いた三つの LSTM の予測誤差よりも小さいことがわかる．また，異なるセル数に設定した二つの簡素型 CLA において，予測誤差に大きな差が無いことがわかる．これは，$n_r = 16$ か $n_r = 32$ かにかかわらず正弦波 $X_s(t)$ を予測するためのセルが十分に存在し，セル数によって精度の改善が望めないことを示すと考えられる．一方，提案 CLA-DL は二つの簡素型 CLA よりも低い予測誤差を達成することがわかる．この結果から，多層化によって従来の CLA では達成できない予測誤差の低減を実現したといえる．

次に，**図 5.6–5.9** より，三つの LSTM において，入力によって最善となる方法が異なることがわかる．これは，最適化アルゴリズムに予測誤差が依存していることを示している．一方，全ての入力において三つの CLA の予測誤差はそれらの LSTM の予測誤差よりも小さくなることがわかる．また，入力によってその推移は異なるものの，二種類の簡素型 CLA における予測誤差はどの入力でも明らかな差は無いことがわかる．これにより，正弦波の合成波 $X_c(t, n)$ でも十分なセル数が存在するといえる．そして，提案 CLA-DL は全ての入力において，二種類の簡素型 CLA よりも特に収束後の予測誤差が小さくなることがわかる．この結果から，様々な入力において提案 CLA-DL は予測精度を改善することがわかる．

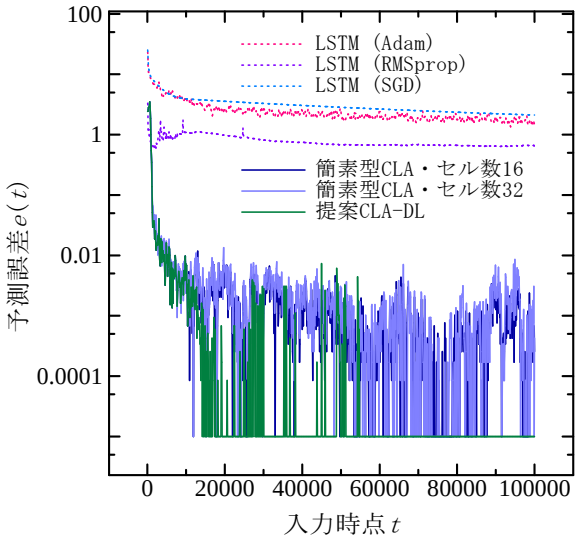


図 5.6: 2 次の正弦波の合成波 $X_c(t, 2)$

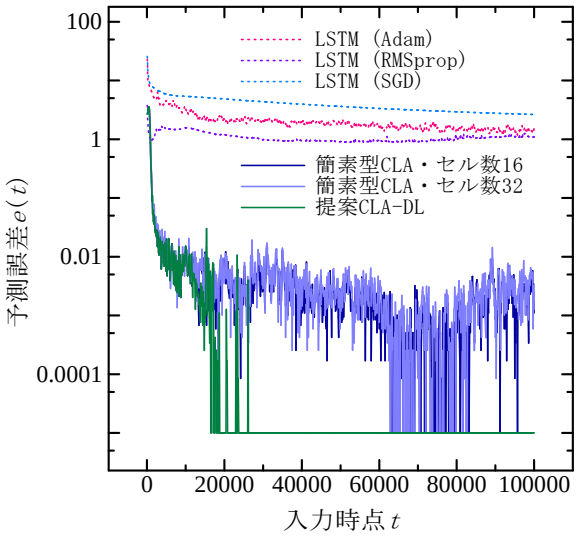


図 5.7: 3 次の正弦波の合成波 $X_c(t, 3)$

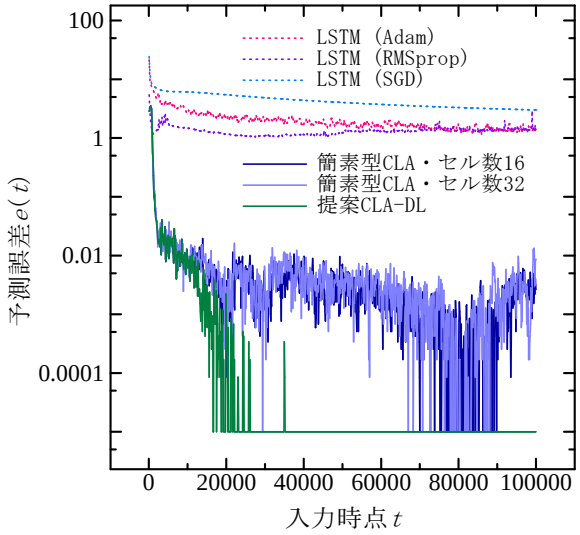


図 5.8: 4 次の正弦波の合成波 $X_c(t, 4)$

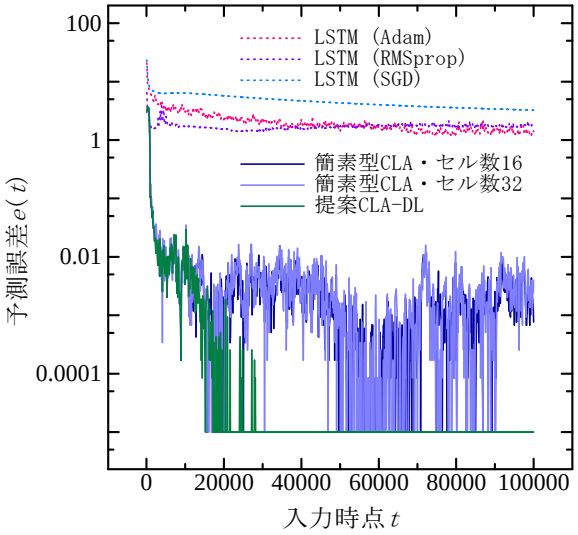


図 5.9: 5 次の正弦波の合成波 $X_c(t, 5)$

以上の結果より，提案 CLA-DL はフィードバックによって予測を補正し，内部表現として一意の予測を実現することで従来の CLA や関連手法である LSTM よりも低い予測誤差を達成することが示された．

5.5　結言

本章では，CLA における予測器を積み重ね，相互に連携させることに着目し，CLA を多層化する方法を提案した．提案する CLA は，第一層で入力データを受け取り，入力データと入力データの文脈を第一層で内部表現する．第一層における入力データの文脈表現は，第二層へと伝達され，抽象表現として第二層で表現される．また，第二層における抽象予測は，第一層における予測の補正および促進のため，第一層へ反映される．ベンチマーク時系列データを使用した実験の結果，提案法である提案 CLA-DL が従来の単一層の CLA と，関連手法である LSTM よりも低い予測誤差を達成することを示した．

第 6 章

カラムに基づくデコーダ

CLA の予測器の内部状態は，人間が理解できる表現形式ではない．人間が理解できるように，CLA の予測器の内部状態をデコードする既存の仕組みがある．従来の CLA におけるデコーダは，予測器の内部状態と入力値のマッチングを学習する．学習中は，予測精度が悪くなる．本章では，予測誤差の収束速度の向上を目的として，学習を回避して，CLA の予測器の内部状態をデコードする方法を提案する．空間プーリングにおいて，入力値を予測器の内部状態にエンコードするカラムのシナプスネットワークが構築される．エンコードのために構築されたカラムのシナプスネットワークを活用し，予測器の内部状態を入力値にデコードする方法を提案する．CLA の予測器において，本章が取り組む部分を**図 6.1** に緑色で示した．本章は，CLA におけるカラムと入力データの関係の取り扱いに注目し，CLA-CD (CLA with Column-based Decoder) を提案する．

6.1 概要

従来の CLA は，入力にかかわらず予測精度の収束に時間がかかる問題がある．その原因として，2.7 節で述べた従来の CLA における SDRC デコーダに着目する．SDRC デコーダには，以下の問題点がある．一つ目は，従来の SDRC デコーダが，学習を要する点である．従来の SDRC は，CLA における予測に関する内部表現をデータビット列にデコードするため，入力データビット列のパターンと予測セルのパターンの関係を学習する必要がある．そのため，CLA が正しく予測を内部表現できたとしても，SDRC の学習が

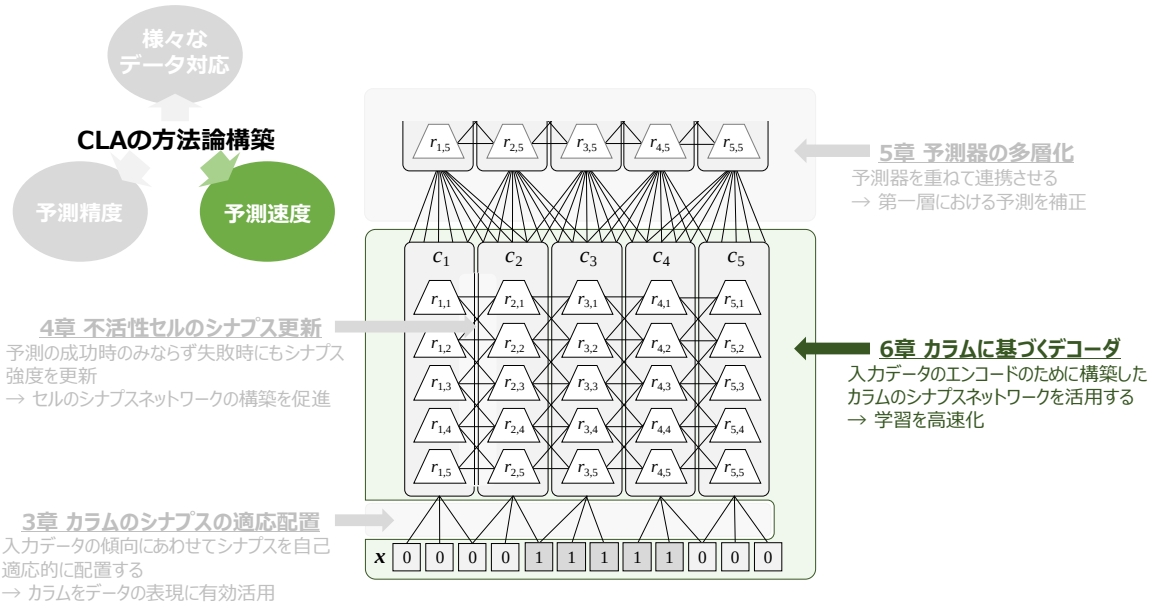


図 6.1: 6 章の位置づけ

不十分な状況では，予測の内部表現を適切にデータビット列にデコードできない．二つ目は，従来の SDRC デコーダが，学習パラメータ α を必要とする点である．α を大きくすると，学習の速度は向上する．しかし，予測精度は不安定になる．学習速度と予測の安定性は，トレードオフの関係にある．そのため，学習速度と予測の安定性の両方を同時に満足するような学習パラメータ α を見出す必要がある．

このように，従来の CLA は，学習の多重構造がある．まず，入力データと入力データの文脈について，カラム，セル，シナプスで表現できるように学習する．次に，入力データと入力データの文脈を予測器で内部表現できるようになった後，その内部表現を入力データの形式で表現できるようにするために，従来の SDRC によるデコーダでさらに学習する．このような学習の多重構造を回避して，CLA の予測器の内部表現を入力データの形式にデコードできることが望ましい．

Algorithm 12 提案する CLA-CD

1: $\mathcal{C} \leftarrow$ Initialization ▷ **Algorithm 2**
2: **for each** time-step t **do**
3: $X(t) \leftarrow$ Receiving input value
4: $\boldsymbol{x} \leftarrow$ Binarization $(X(t))$
5: $\mathcal{C} \leftarrow$ Spatial pooling $(\mathcal{C}, \boldsymbol{x})$ ▷ **Algorithm 3**
6: $\mathcal{C} \leftarrow$ Temporal pooling $(\mathcal{C})$ ▷ **Algorithm 4**
7: $\overline{X}(t+1) \leftarrow$ Column-based Decoder $(\mathcal{C})$
8: **end for**

6.2 方法

6.2.1 コンセプト

従来の SDRC デコーダの問題を解決して CLA の予測精度を改善するため，本章ではカラムベースのデコーダを提案する．提案法は，空間プーリングにおけるエンコードの過程で構築された既存のシナプスネットワークを活用して予測データビット列を決定する．これにより，提案法は SDRC デコーダで必要だった追加の学習コストとパラメータ設定の問題を回避する．提案する CLA-CD の擬似コードは **Algorithm 12** のようになる．7 行目に青色で示す通り，従来の SDRC デコーダの代わりに提案法を用いる．

6.2.2 アルゴリズム

提案するカラムベースのデコーダの概念図を**図 6.2** に示す．

ステップ 1 ： 各カラム c_i $(i = 1, 2, \ldots, n_c)$ について，予測状態であるセル数を $c_i.n_p$ として計数する．

図 6.2 において例を示す．カラム c_1 には予測状態のセルが含まれないため $c_1.n_p = 0$ となる．カラム c_2 には予測状態のセルがひとつ含まれるため $c_2.n_p = 1$ となる．カラム c_3 には予測状態のセルが二つ含まれるため $c_3.n_p = 2$，というように計数する．

ステップ 2 ： 各入力データビット x_j $(j = 1, 2, \ldots, n)$ について，接続状態のシナプスで紐づけられたカラムに含まれる予測セルの合計数を d_j $(j = 1, 2, \ldots, n).$ として計

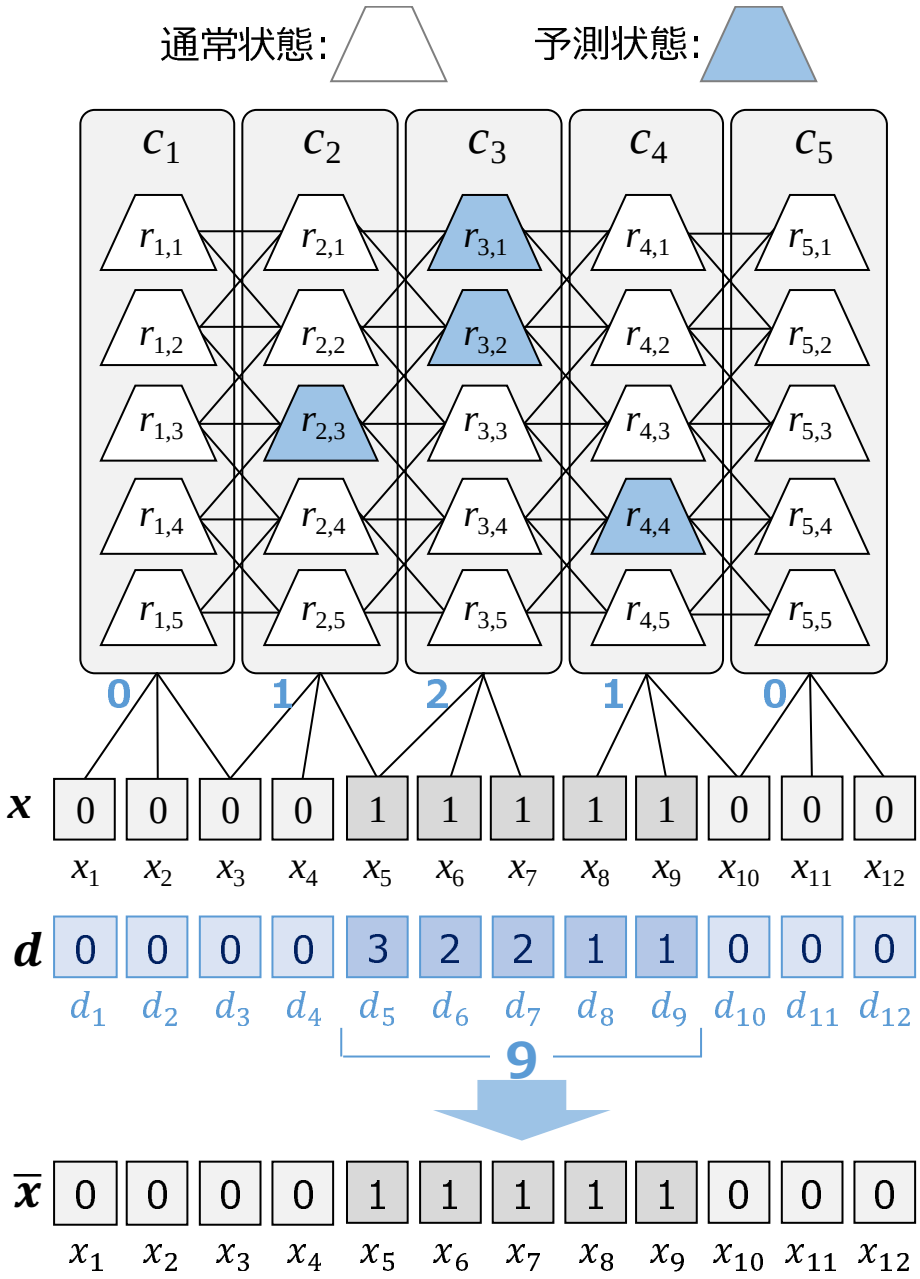


図 6.2: 提案する CLA-CD：カラムベースのデコーダ

算する．

図 6.2 において例を示す．1–4 番目のビットにおいて，シナプス接続されたカラム内に予測セルが含まれないため，$d_1 = d_2 = d_3 = d_4 = 0$ となる．一方，5 番目のビットにおいては，2 本のカラム c_2 と c_3 と接続状態のシナプスで紐づけられ，それらのカラムは $c_2.n_p = 1$，$c_3.n_p = 2$ と予測セルを含む．そのため，これらを合計

して $d_5 = 3$ となる．

ステップ 3： 現入力時点 t までに入力された入力ビット列集合 $\mathcal{X} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \dots\}$ について，以下の式を満たす予測ビット列候補集合 $\mathcal{X}'$ を求める．

$$\mathcal{X}' = \underset{\boldsymbol{x} \in \mathcal{X}}{\operatorname{argmax}} \sum_{j=1}^{n} d_j x_j \tag{6.1}$$

ステップ 4： 予測ビット列候補集合 $\mathcal{X}'$ における中央値を求め，それを次時点 $t+1$ における予測ビット列 $\overline{\boldsymbol{x}}$ として出力する．式は以下のようになる．

$$\overline{\boldsymbol{x}} = \operatorname{median}(\mathcal{X}') \tag{6.2}$$

図 6.2 において，$\mathcal{X}$ から最も $\sum_{j=1}^{n} d_j x_j$ が大きいビット列を選択し，$\overline{\boldsymbol{x}}$ とする例を示す．ビット列 $\overline{\boldsymbol{x}}$ は提案するカラムベースのデコーダの出力となる．

カラムベースのデコーダの基礎的な実装が，先行研究 [69] で利用されている．この既存手法と，本章の提案法の違いを述べる．まず，**ステップ 1** において，本章での提案法は各カラムの予測セル数を計数するが，既存手法は各カラムに予測セルを含むかどうかの二値で計算する．また，**ステップ 2** において，提案法は各データビットに接続されたカラム内の予測状態のセル数を合計するが，既存手法は各データビットに接続された予測状態のセルを含むカラム数を計数する．その結果，**ステップ 3** において，本章での提案法は $\mathcal{X}$ からより詳細なデータビット列のランキングを獲得することができる．さらに**ステップ 4** において，提案法は複数のデータビット列の $\sum_{j=1}^{n} d_j x_j$ が同じ値となるケースも考慮できる．

6.3 実験設定

提案する CLA-CD の効果を検証するため，人工のベンチマーク時系列データと実世界の時系列データを用いて予測精度を比較する．比較手法として，従来 CLA，従来 CLA-CD，簡素型 CLA，提案 CLA-CD の 4 種類の CLA と，関連手法である LSTM を用いる．

6.3.1　入力時系列データ

人工のベンチマーク時系列データとして，周期的なデータである正弦波 $X_s(t)$ と正弦波の合成波 $X_c(t)$ を用いる．また，非周期的な入力データとして，ロジスティック写像 $X_l(t,\alpha)$ を用いる．時点 t における各入力値は，それぞれ以下の式で計算される．

$$X_s(t) = \frac{1}{2} \cdot \sin\left(\frac{(t-1)\cdot\pi}{50}\right) + \frac{1}{2}, \tag{6.3}$$

$$X_c(t) = \frac{1}{2} \cdot \sum_{k\in\{1,3,5,7,9\}} \frac{1}{k} \cdot \sin\left(\frac{(t-1)\cdot k\cdot\pi}{50}\right) + \frac{1}{2}, \tag{6.4}$$

$$X_l(t,\alpha) = \begin{cases} 0.4, & \text{if } t = 1, \\ \alpha \cdot X_l(t-1)\cdot\{1 - X_l(t-1)\}, & \text{otherwise.} \end{cases} \tag{6.5}$$

ロジスティック写像については，$X_l(t, 3.6)$ と $X_l(t, 4.0)$ の 2 種類を用いる．入力時点の範囲は $t \in [1, 10^5]$ に設定した．時系列入力データにおいて，その値域は $[X^{\min}, X^{\max}] = [-0.01, 1.01]$ とした．

実世界の時系列予測問題として，UCI Machine Learning Repository で公開されている 2011–2014 年の電力消費量のデータセットを用いた [70]．これは 2011–2014 年における 370 件の顧客の電力消費量を含む時系列データである．データは 15 分間隔で，単位は [kW] である．本章では，2014 年 12 月の MT 002 という顧客のデータを用いる．12 月 1 日から 31 日までのデータ数は 2976 である (= 31 日 × 24 時間 × 4 データ点)．

実験では，ある時点 t の入力時系列データを入力として，次時点 $t+1$ の入力時系列データを予測する，ということを各入力時点に対して繰り返した．

6.3.2　評価指標

予測精度を評価するための指標として，以下の式で計算される 100 時点ごとの予測誤差 $e(t)$ を用いる．

$$e(t) = \sum_{\tau=t-99}^{t} |\bar{X}(\tau+1) - X(\tau+1)|, \tag{6.6}$$

表 6.1: 提案法の効果を評価するためのアルゴリズム

	従来 CLA	従来 CLA -CD	簡素型 CLA	提案 CLA -CD
3.2.1 節に記した方法	-	-	✓	✓
SDRC デコーダ	✓	-	✓	-
提案するカラムベースのデコーダ	-	✓	-	✓

ここで，$\bar{X}(t+1)$ は次時点 $t+1$ の予測値，$X(t+1)$ は次時点 $t+1$ における実際の入力値を表す．$e(t)$ は，それぞれ $t \in \{100, 200, 300, 400, \ldots, 10^5\}$ で計算される．$e(t)$ が小さいほど予測精度が良いといえる．

また，従来法である SDRC デコーダと，提案法であるカラムベースのデコーダの出力を比較するため，各時点の入力データビット列と予測データビット列のハミング距離を計測した．この値が小さいほど入力と予測が一致しているといえる．

6.3.3 アルゴリズム

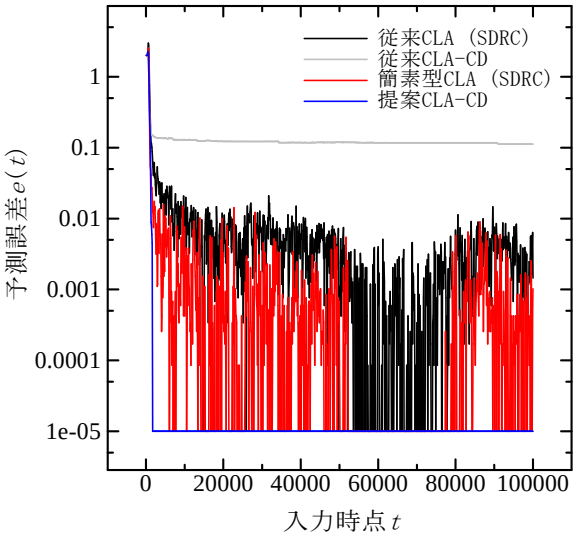
まず，四つの CLA について比較する．一つ目として，NuPIC のライブラリ [8] で公開されている従来 CLA を用いる．二つ目として，従来 CLA に提案するカラムベースのデコーダを適用した従来 CLA-CD を用いる．三つ目として，簡素型 CLA[67] を用いる．これは，従来 CLA でカラムのシナプスについて簡単化した手法である．簡素型 CLA は，3.2.1 節に記したシナプス位置の正規化とカラムのシナプスの永続値の初期値固定により，データビットと紐づけられるカラムのシナプスを決定論的に設定することで CLA の安定性を向上する [67]．従来 CLA と簡素型 CLA では，SDRC デコーダ [62] を用いる．四つ目として，簡素型 CLA に提案するカラムベースのデコーダを適用した提案 CLA-CD を用いる．4 種類のアルゴリズムについて，適用した方法の比較を**表 6.1** に示す．表に示す通り，従来 CLA と従来 CLA-CD は学習器の構成はそれぞれ同一である．異なる点は，デコーダとして SDRC デコーダを用いているか，それとも提案するカラムベースのデコーダを用いてるか，のみである．これにより，従来 CLA と従来 CLA-CD を比較することで，従来 CLA において SDRC デコーダの代わりに提案するカラムベースのデコーダを用

いることによる効果が確認できる．同様に，簡素型 CLA と提案 CLA-CD を比較することで，簡素型 CLA において SDRC デコーダの代わりに提案するカラムベースのデコーダを用いることによる効果が確認できる．

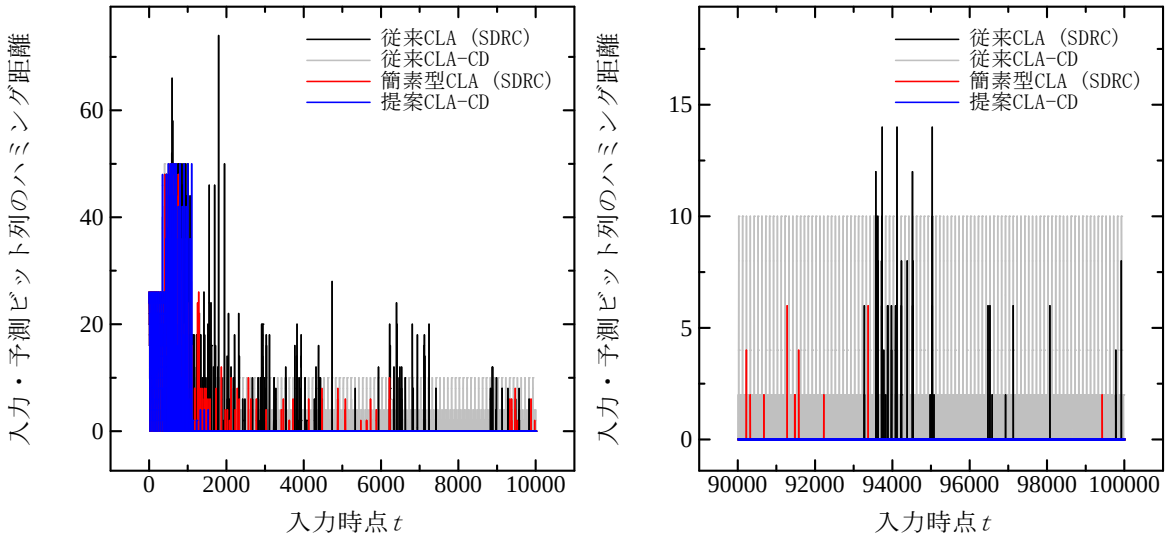
次に，三つの CLA と三つの LSTM[3] をあわせた 6 種類の方法を比較する．CLA について，**表 6.1** における従来 CLA，簡素型 CLA，提案 CLA-CD の 3 種類を用いる．また，ネットワークの最適化アルゴリズムとして Adam[64]，RMSprop[65]，SGD をそれぞれ適用した 3 種類の LSTM を用いる．LSTM は，Keras[68] による実装を活用する．LSTM の入力窓サイズは 1 に設定した．隠れ層は 2 層で，ユニット数はそれぞれ 100 とした．一つ目の層は線形関数を活性化関数とした全結合のニューラルネットワーク層を用いた．二つ目の層は tanh 関数を活性化関数とした LSTM 層を用いた．また，出力層として 1 ユニットで活性化関数なしの全結合層を用いた．損失関数は平均二乗誤差を用いた．

6.3.4　パラメータ

入力データとして，入力データビット数は $n = 421$，チャンク長は $w = 21$ に設定した．空間プーリングに関するパラメータとして，カラム数は $n_c = 2048$，各カラムのシナプス数は $n_{cy} = 21$，カラムのシナプスにおける永続値の閾値は $\theta_c = 0.1$，活性状態となるカラム数は $n_p = 40$，永続値の増減量はそれぞれ $p_c^+ = 0.05$，$p_c^- = 0.00025$ とした．時間プーリングに関するパラメータとして，各カラムのセル数は $n_r = 32$，セルのシナプスにおける永続値の閾値は $\theta_r = 0.5$，セルが予測状態となるために必要な接続シナプス数は $n_{ar} = 15$，永続値の増減量はそれぞれ $p_c^+ = 0.1$，$p_c^- = 0.1$ とした．従来の CLA における SDRC のパラメータとして，学習率は $\alpha = 0.75$ に設定した．本章では，ベースとして NuPIC のライブラリ [8] で設定されたデフォルトのパラメータを使用した．第 3 章と同様に，予測のビット単位での差異を判別しやすくするために $n = 421$，カラムが単一の入力ビット列に対応しやすくするために $n_{cy} = 21$ にデフォルト値から変更した．また，カラムのシナプスの切断状態への変化が提案するカラムベースのデコーダの出力値に大きく影響するため，カラムのシナプスの永続値の減少量を $p_c^- = 0.00025$ とすることで予測の安定化をはかった．



(a) 予測誤差の推移



(b) 学習開始時のハミング距離　　(c) 学習終了時のハミング距離

図 6.3: 正弦波 $X_s(t)$ における CLA の比較

6.4　実験結果と考察

6.4.1　CLA における結果

図 6.3 に正弦波 $X_s(t)$ を入力としたときの CLA における結果を示す．**図 6.3 (a)** は予測誤差の推移である．表示のため，予測誤差 $e(t)$ が 10^{-5} 以下の場合は 10^{-5} に丸めた．

また，解析として，各時点の入力データビット列と予測データビット列のハミング距離を計算した．**図 6.3 (b)** は学習開始時にあたる $t \in [1, 10,000]$，**図 6.3 (c)** は学習終了時にあたる $t \in [90,001, 100,000]$ 時点での結果である．

図 6.3 (a) より，従来 CLA において SDRC デコーダの代わりにカラムベースのデコーダを用いた従来 CLA-CD は，SDRC デコーダを用いた従来 CLA よりも予測精度が悪化することがわかる．従来 CLA のカラムのシナプスが確率的でランダムに配置されることにより，提案法のステップ 2 で算出する各ビットの予測セルの合計数 d_j が不安定となり，カラムベースのデコーダの出力も不安定となったと考えられる．これに対して，2.7 節で述べた SDRC デコーダはカラムのシナプス配置を活用しないため，従来 CLA でも一定の予測精度を達成できたと考えられる．一方，簡素型 CLA は従来 CLA よりも低い予測誤差を達成する．また，簡素型 CLA において SDRC デコーダの代わりに提案法であるカラムベースのデコーダを用いた提案 CLA-CD が，比較手法の中で最も予測誤差が低くなる．簡素型 CLA によってカラムのシナプス配置が決定論的となり，各ビットの予測セルの合計数 d_j が安定し，カラムベースのデコーダが期待する効果を発揮したと考えられる．

図 6.3 (b) より，簡素型 CLA は従来 CLA よりもハミング距離が小さくなることがわかる．また，その傾向は学習終盤である**図 6.3 (c)** においても継続する．しかし，従来 CLA および簡素型 CLA は断続的にハミング距離が増大する時点が現れる．一方，従来 CLA-CD では，**図 6.3 (b)** の $2,000$ 時点以降から，**図 6.3 (c)** の学習終了時まで，そのハミング距離は安定する．しかし，正弦波 $X_s(t)$ の周期に合わせて変化する傾向が見られる．これが**図 6.3 (a)** において予測誤差がある程度大きい状態で収束した原因である．これに対して，提案 CLA-CD では $2,000$ 時点以降においてハミング距離が常に 0 となる．これにより，提案 CLA-CD は最も良い予測精度を達成したといえる．また，その収束速度は従来 CLA および簡素型 CLA よりも早い．提案するカラムベースのデコーダが，従来法である SDRC デコーダにおいて必要とした学習コストを必要としないため，従来法における問題点を解決し，この収束速度を実現したと考えられる．

6.4.2　CLA と LSTM における結果

6.4.2.1　ベンチマークデータにおける結果

図 6.4 と**図 6.5** にそれぞれ正弦波 $X_s(t)$ と正弦波の合成波 $X_c(t)$ を入力としたときの

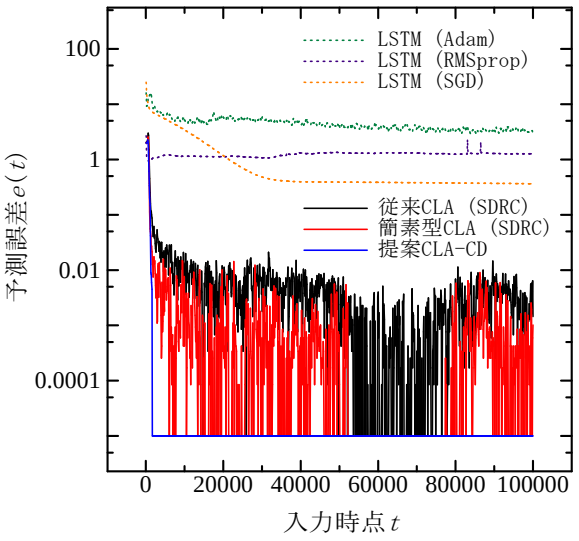


図 6.4: 正弦波 $X_s(t)$

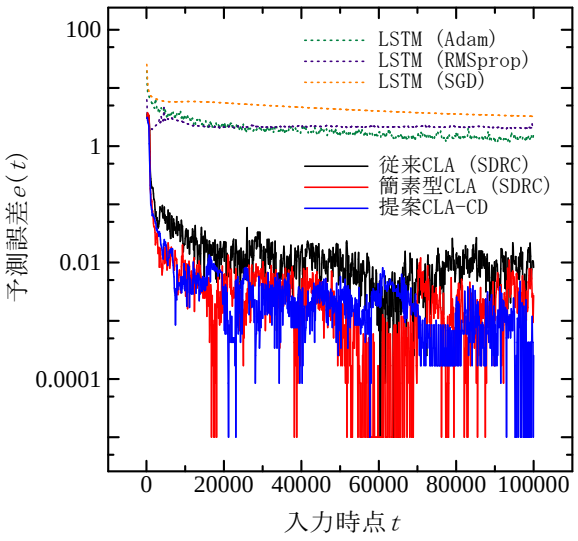


図 6.5: 正弦波の合成波 $X_c(t)$

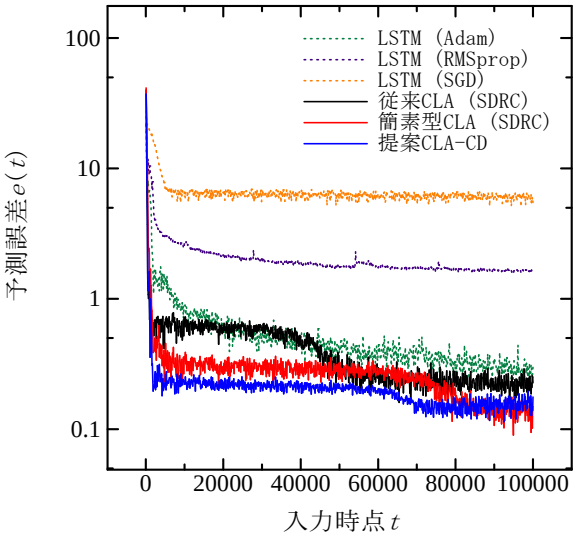


図 6.6: ロジスティック写像 $X_l(t, 3.6)$

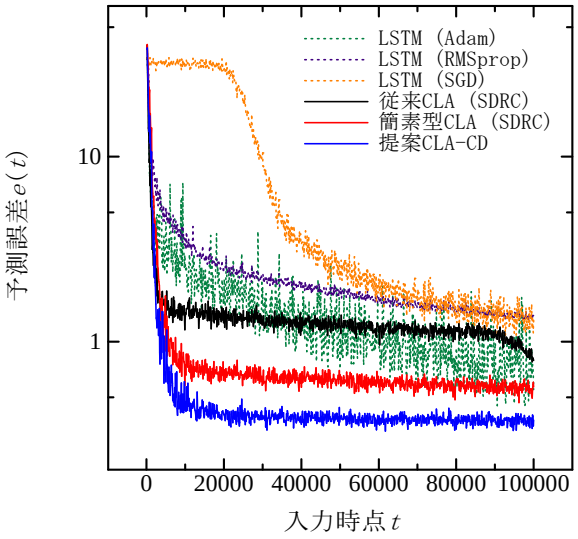


図 6.7: ロジスティック写像 $X_l(t, 4.0)$

予測誤差の推移を示す．また，**図 6.6** と**図 6.7** に 2 種類のロジスティック写像 $X_l(t, 3.6)$ および $X_l(t, 4.0)$ を入力としたときの予測誤差の推移をそれぞれ示す．ここで表示のため，予測誤差 $e(t)$ が 10^{-5} 以下の場合は 10^{-5} に丸めた．

まず，従来の LSTM において，最適化アルゴリズムの違いによって予測誤差が変化することがわかる．一方，全体の傾向として，ロジスティック写像 $X_l(t, 4.0)$ における従来 CLA と Adam を用いた LSTM の比較を除き，LSTM の予測誤差は，CLA の予想誤差よりも高い．

表 6.2: 電力消費予測における予測誤差合計

方法	予測誤差合計
従来の CLA (SDRC)	3323.64
簡素型 CLA (SDRC)	3303.52
提案する CLA	2570.86

次に，従来のデコーダである SDRC を用いた二つの CLA において，簡素型 CLA は，全ての時系列ベンチマークデータで従来 CLA より低い予測誤差を達成することがわかる．また，提案するカラムベースのデコーダを適用した提案 CLA-CD が，SDRC デコーダを用いた簡素型 CLA よりさらに予測誤差が低いことがわかる．全体の傾向として，カラムベースのデコーダを適用した提案 CLA-CD において，予測誤差の減少が他の 2 種類の CLA よりも速い．特に正弦波 $X_s(t)$ と 2 種類のロジスティック写像 $X_l(t, 3.6)$ および $X_l(t, 4.0)$ を入力としたときのそれぞれの結果である**図 6.4**，**図 6.6**，**図 6.7** において，二つの従来の CLA が入力時点が進む中で徐々に予測誤差を減少させる一方，提案 CLA-CD が予測誤差を学習初期で明らかに素早く減少させることがわかる．従来手法である SDRC デコーダが，入力ビット列のパターンと予測状態となるセルのパターン間の関係を決定する重み行列 W の調整に学習期間を必要とすることが原因だと考えられる．一方，提案するカラムベースのデコーダは元から存在する空間プーリングで構築した入力データビットとカラムとの関係を活用するため，この学習コストを削減することができる．これにより，予測誤差の素早い低減に貢献する．しかし，**図 6.6** において，$80,000$ 時点以降から提案 CLA-CD の予測誤差が不安定となり，低い予測誤差を達成するものの，そのばらつきが増大することがわかる．これは，入力であるロジスティック写像 $X_l(t, 3.6)$ は非周期的なデータであり，入力時点数が増えるほど新しい入力値が現れるため，それを内部表現化するための学習器のサイズが不足し，予測表現が混同したことが考えられる．

以上の結果から，提案するカラムベースのデコーダが，予測誤差の減少に貢献し，従来の SDRC デコーダを用いた際よりも小さい予測誤差を達成できることが明らかになった．

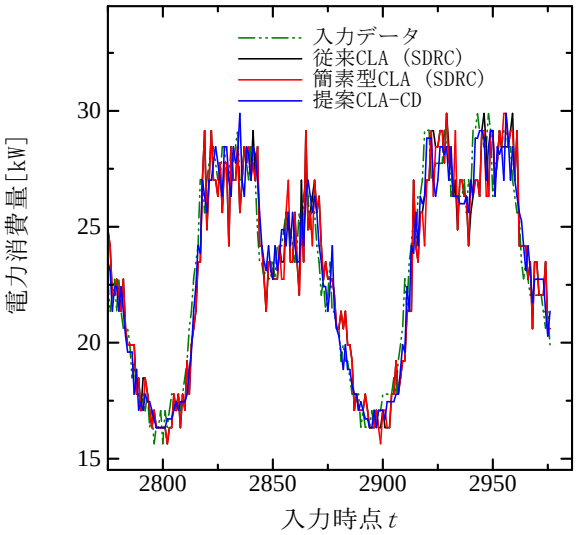


図 6.8: 電力消費予測

6.4.2.2　実世界のデータにおける結果

実世界の電力消費量を予測する問題において，提案するカラムベースのデコーダの効果を検証する．SDRC デコーダを用いる従来 CLA，SDRC デコーダを用いる簡素型 CLA，簡素型 CLA に提案するカラムベースのデコーダを適用した提案 CLA-CD の 3 種類の方法を比較する．

図 6.8 に従来 CLA，簡素型 CLA，提案 CLA-CD における最後の 200 時点での入力値と予測値を示す．全ての時点間隔は 15 分である．また，**表 6.2** に三つの CLA の予測誤差合計を示す．この結果から，簡素型 CLA は従来 CLA よりも予測精度が良いことがわかる．また，提案 CLA-CD は簡素型 CLA よりも低い予測誤差を達成する．今回予測した電力消費量は，日にち単位や週単位などの周期性を持つものの，各時点における値はノイズを含むケースと同様に変動する．そのため，ある時刻における予測値は分散し，複数の値を示す予測表現が生成されると考えられる．このような入力に対して，提案するカラムベースのデコーダは，複数の予測表現から最頻値を抽出し，その中の中央値を獲得することができる．その結果，予測精度を改善したと考えられる．以上より，提案するカラムベースのデコーダは実世界の電力消費量を予測する問題においても予測精度の改善に貢献することが明らかとなった．

6.5　結言

本章では，従来の CLA におけるカラムと入力データの関係の取り扱いに着目し，従来の SDRC デコーダの学習を回避して，CLA の予測器の内部状態をデコードする方法を提案した．提案法は，エンコードのために構築されたカラムのシナプスネットワークを活用し，予測器の内部状態を入力値にデコードする．提案するデコーダは，入力データとカラムの間で構築済みのシナプスを活用するため，従来の SDRC デコーダで必要になる学習コストとパラメータチューニングを回避できる．人工のベンチマーク時系列データを入力とした実験の結果，提案するカラムベースのデコーダは予測誤差を素早く低減し，従来の SDRC デコーダよりも予測誤差を小さくすることを示した．また，実世界の電力消費量を予測する問題を用いた実験の結果，ここでも提案法が予測誤差の減少に貢献することを示した．

第 7 章

方法の統合と相互作用

実世界の電力消費量を予測する問題において，本論文で提案した四つの方法の組み合わせについて，効果を検証する．

7.1 実験設定

7.1.1 アルゴリズム

比較手法として，6 種類の LSTM と 4 種類の CLA を比較した，まず LSTM として，6.3.3 節で述べた 3 種類の最適化アルゴリズムである Adam，RMSprop，SGD に対して，窓サイズを 1 と 100 の 2 通りにそれぞれ設定して適用した．窓サイズ以外の層数やユニット数については，6.3.3 節で述べた設定を用いた．また，CLA として，6.3.3 節で述べた従来 CLA と，**表 7.1** に示した 3 種類の提案 CLA を用いた．一つ目として，第 3 章で提案したカラムのシナプスの適応配置，第 4 章で提案した不活性セルのシナプス更新，第 6 章で提案したカラムに基づくデコーダを導入した CLA を用いた．セル数は 32 に設定した．二つ目として，一つ目の CLA において，セル数を 16 に変更した CLA を用いた．この 2 種類の CLA を比較することで，セル数による予測精度の変化を見ることができる．三つ目として，二つ目の CLA に第 5 章で提案した予測器の多層化を導入した CLA を用いた．

次に解析として，本論文で提案したカラムのシナプスの適応配置，不活性セルのシナプス更新，カラムに基づくデコーダの 3 方法の有無の組み合わせで CLA の予測性能を比較した．基準となる方法として，3.2.1 節に記した方法を導入した簡素型 CLA を用いた，こ

表 7.1: 本論文における提案法を組み合わせたアルゴリズム

	提案 CLA (単一層・セル数 32)	提案 CLA (単一層・セル数 16)	提案 CLA (多層・セル数 16 & 16)
第 3 章 カラムのシナプスの適応配置 CLA-AC	✓	✓	✓
第 4 章 不活性セルのシナプス更新 CLA-IU	✓	✓	✓
第 5 章 予測器の多層化 CLA-DL	—	—	✓
第 6 章 カラムに基づくデコーダ CLA-CD	✓	✓	✓
セル数	32	16	16(第一層) 16(第二層)

れは，カラムのシナプスの適応配置を導入した CLA-AC，およびカラムに基づくデコーダを導入した CLA-CD においてもベースとして簡素型 CLA を用いたためである．また，従来法である従来 CLA とも予測精度を比較した．

7.1.2　入力時系列データ

入力時系列データとして，New York ISO [10] で公開された電力消費の予測タスクを用いる．この入力データは [66] のケースと同様に，外れ値だと考えられるスパイクを削除し，欠損値を補完し，1 時間ごとの窓で移動平均を算出したデータから，2007 年 5 月 1 日から 2019 年 2 月 28 日までのニューヨーク市の 15 分間隔の電力消費データを獲得した．この入力では，最後の 28 日間，すなわち，2019 年 2 月の予測精度を比較する．

実験において，CLA では，ある時点 t の入力時系列データを入力として，次時点 $t+1$ の入力時系列データを予測する，ということを各入力時点に対して繰り返した．LSTM では，窓サイズが 1 の場合は CLA と同様の入力を用いた．窓サイズが 100 の場合，ある時点 $\{t, t+1, t+2, \ldots, t+99\}$ のデータを入力として，次時点 $t+100$ のデータを予測した．

7.1.3 評価指標

予測精度を評価するため，次式で求める予測誤差 $e(t)$ を用いる．

$$e(t) = |\bar{X}(\tau+1) - X(\tau+1)|, \tag{7.1}$$

ここで，$\bar{X}(t+1)$ は，次時点 $t+1$ の予測値である．$X(t+1)$ は，次時点 $t+1$ の実際の入力値である．$e(t)$ が小さいほど，予測精度が高いと判断する．本章では，最後の 28 日間での $e(t)$ を合計して比較する．また，その 28 日間における日ごとの予測誤差も算出して比較する．

7.1.4 パラメータ

従来 CLA は，セル数以外のパラメータについて，第 3 章で用いたパラメータ群と同一の値を用いた．提案 CLA に導入した提案法は，各章に示したパラメータ群を用いた．提案法を組み合わせる上で，提案法のアルゴリズムを一部調整した．まず，第 4 章で提案した不活性セルのシナプス更新において，カラムを活性候補とする条件をオーバーラップ値が 1 以上 ($c_i.ol \geq 1$) から，オーバーラップ値が 17 以上 ($c_i.ol \geq 17$) に変更した．第 3 章で提案したカラムのシナプスの適応配置において，カラムが活性状態になるために必要な最小のオーバーラップ値 $ol^{min} = 17$ と設定されていることを考慮し，その条件に適応させるためである．次に，第 5 章で提案した予測器の多層化において，フィードバックの強度を弱めた．具体的には，シナプスの接続閾値 θ_r，第二層から予測された第一層のセルにおいて，予測状態となるために必要な活性状態のセルとの接続シナプス数 n_p を，半減である 0.5 倍から 0.9 倍に変更した．また，第二層から予測されていない第一層のセルにおける n_p を，倍化である 2 倍から 1.11 倍に変更した．実世界の電力消費量は予測が難しく，第二層における予測の正確性は落ちることを考慮し，その正確性に合わせたフィード

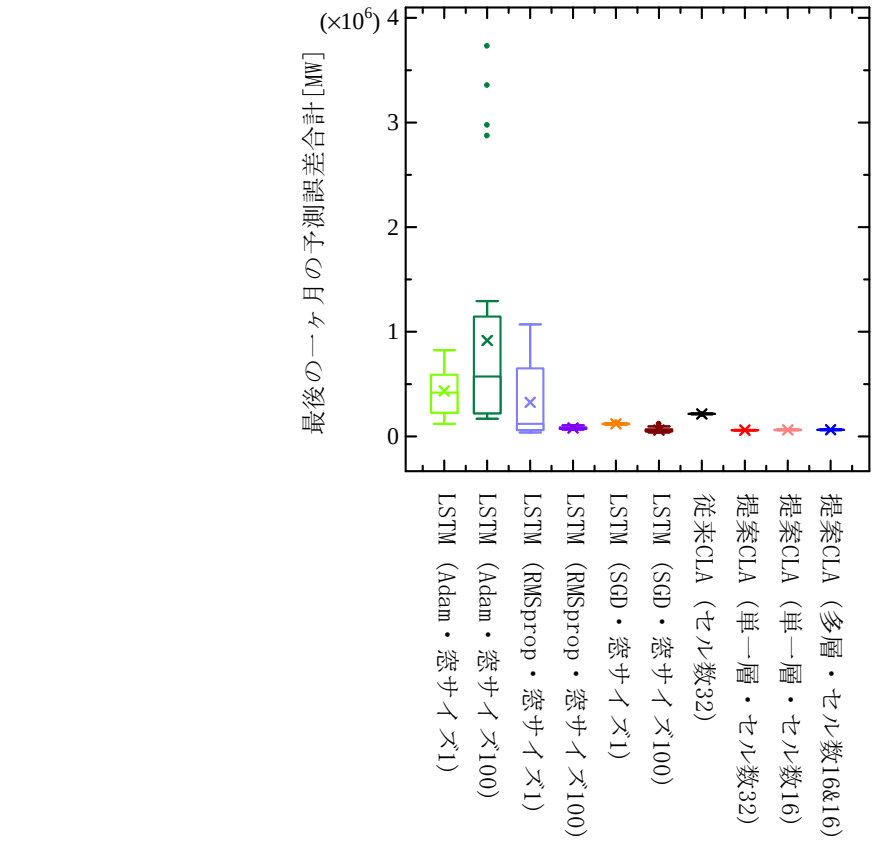
バック強度を実現するためである．

7.2　実験結果と考察

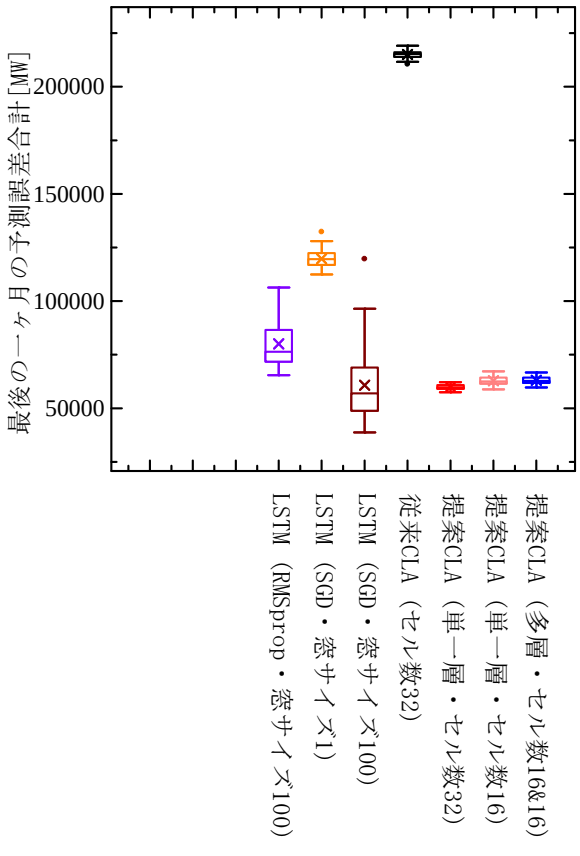
7.2.1　CLA と LSTM における結果

図 7.1 に 2019 年 2 月にあたる最後の 28 日間での予測誤差合計を箱ひげ図として出力した結果を示す．図内において，各方法の平均値は×印，外れ値は●印で表示した．まず，全体の結果を示した**図 7.1 (a)** から，比較手法である LSTM は最適化アルゴリズムおよび窓サイズによって大きく予測誤差が変化することがわかる．最適化アルゴリズムとして Adam を用いた LSTM は，窓サイズを大きくした際に予測誤差が増大する．パラメータはデフォルトの値を用いているが，それが適切でない可能性が考えられる．次に，一部を拡大した結果を示した**図 7.1 (b)** から，LSTM の中では最適化アルゴリズムとして SGD を用いた，窓サイズ 100 のときに最も予測誤差が小さくなることがわかる．また，4 種類の CLA の比較から，本論文で提案した方法を導入することで予測精度を大きく改善できることがわかる．そして，さらに一部を拡大した結果を**図 7.1 (c)** に示す．この結果から，CLA において，多層の提案 CLA は，セル数 16 の単一層の提案 CLA とほぼ同等の予測精度を示すことがわかる．平均値および中央値はわずかに多層の提案 CLA の方が大きいが，シードによるばらつきが小さくなる．多層化することで予測の補正を実現し，より安定した結果を獲得できたと考えられる．単一層の提案 CLA において，学習器のサイズであるセル数を 32 よりも 16 に設定した方が予測誤差が小さくなることがわかる．今回用いたデータは推移の傾向が曜日ごとに変化する上，季節ごとに全体の傾向も変化するため，膨大なパターンが存在し，より多くのセル数を必要としたと考えられる．また，セル数 32 の単一層の提案 CLA は，平均値で比較した場合，関連手法である LSTM よりも予測誤差が小さい．一方，中央値で比較した場合，LSTM の方が予測誤差が大きい．**図 7.1 (b)** の通り，シードによるばらつきは提案 CLA の方が小さく，外れ値も存在しないため，提案 CLA は安定して良い予測精度を達成するといえる．

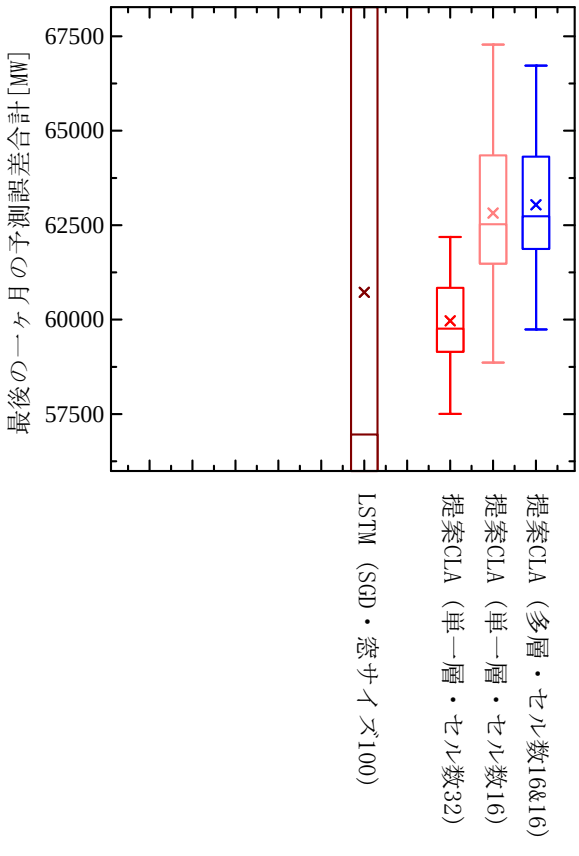
最後の 28 日間における日ごとの予測誤差の推移を，**図 7.2** に示す．まず，全体の結果を示した**図 7.2 (a)** から，最適化アルゴリズムとして Adam を用いた関連手法である LSTM は，予測誤差の推移に週ごとに周期性があり，一部の曜日に適合することがわか



(a) 予測誤差合計の箱ひげ図：全体

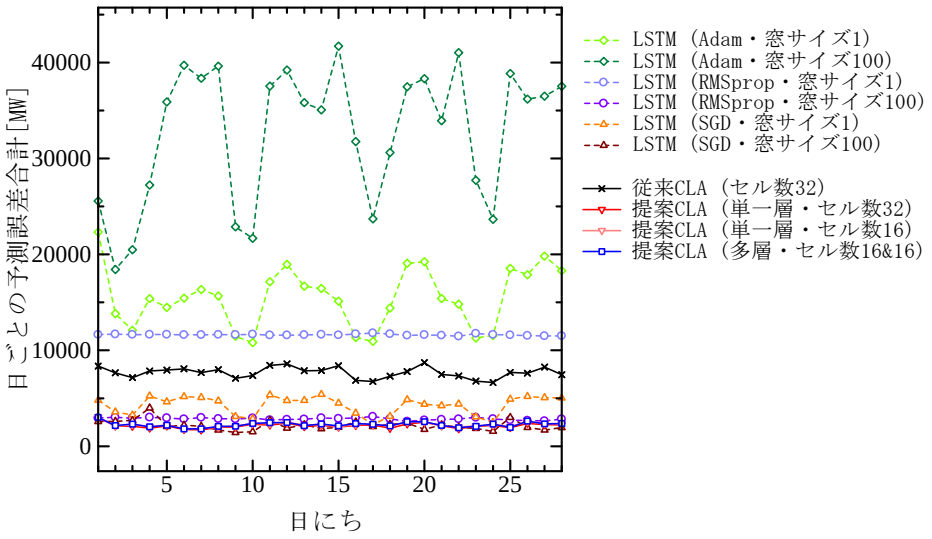


(b) 予測誤差合計の箱ひげ図：拡大その 1

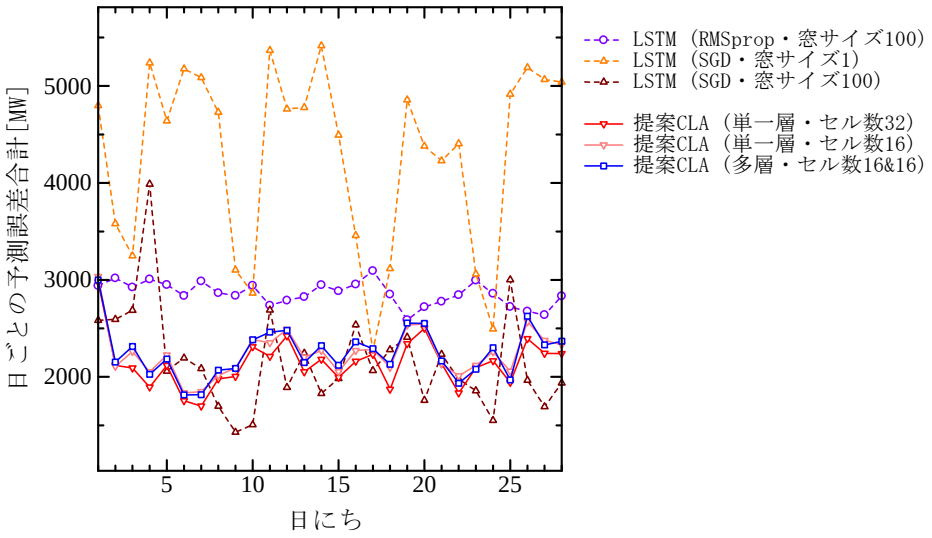


(c) 予測誤差合計の箱ひげ図：拡大その 2

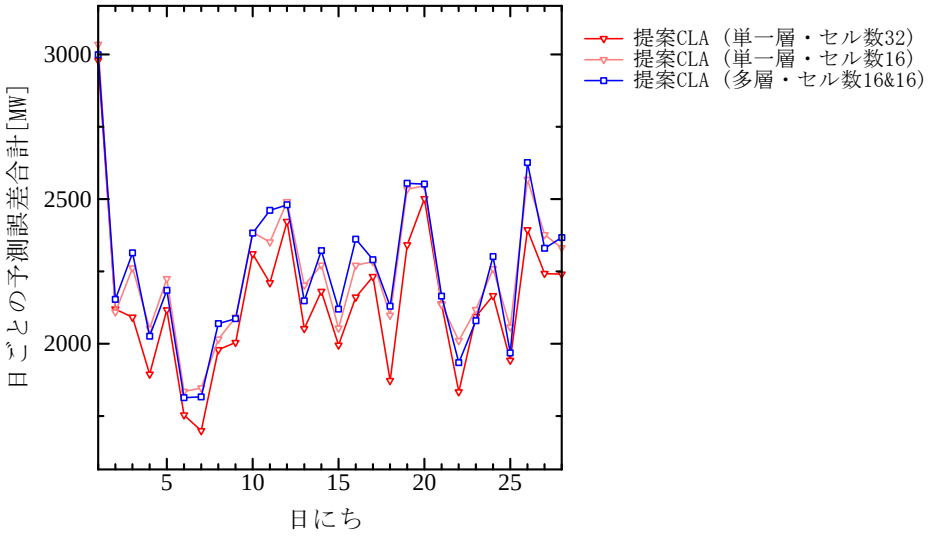
図 7.1: 実世界の電力消費予測：最後の一ヶ月の予測誤差



(a) 日ごとの予測誤差の推移：全体



(b) 日ごとの予測誤差の推移：拡大その 1



(c) 日ごとの予測誤差の推移：拡大その 2

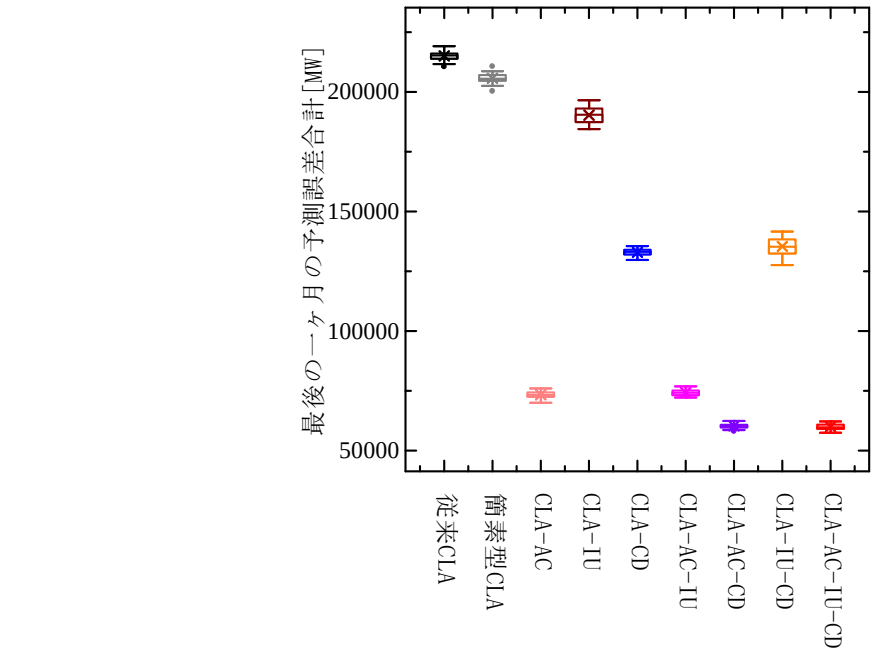
図 7.2: 実世界の電力消費予測：日ごとの予測誤差

る．次に，一部を拡大した**図 7.2 (b)** から，最適化アルゴリズムに RMSprop を用いた窓サイズ 100 の LSTM は，一週間にわたって予測誤差の推移が安定することがわかる．また，最適化アルゴリズムに SGD を用いた窓サイズ 100 の LSTM は，提案法である提案 CLA よりも予測精度が良い日にちが存在する．しかし，月曜日にあたる 4・11・25 日の予測誤差が大きい．一部の曜日での予測精度が不安定なことで，シードによる予測誤差のばらつきが大きくなり，**図 7.1** の箱ひげ図のような結果となったと考えられる．そして，さらに一部を拡大した**図 7.2 (c)** から，提案 CLA において，セル数 32 の単一層の提案 CLA が最も良い予測精度を示すことがわかる．セル数 16 の単一層の提案 CLA と，多層の提案 CLA には大きな差が存在せず，同等の予測誤差を示したといえる．

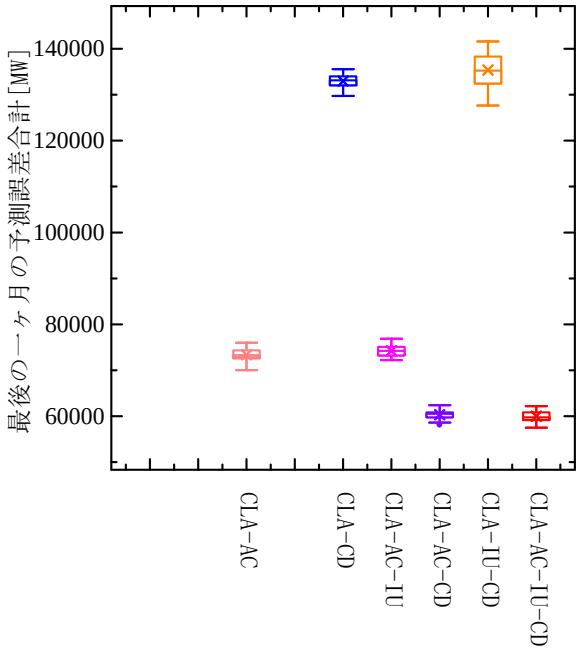
以上の結果から，三つの方法論を組み合わせることで，提案する CLA が良好な予測精度を達成することを示した．

7.2.2 CLA における各提案法の効果

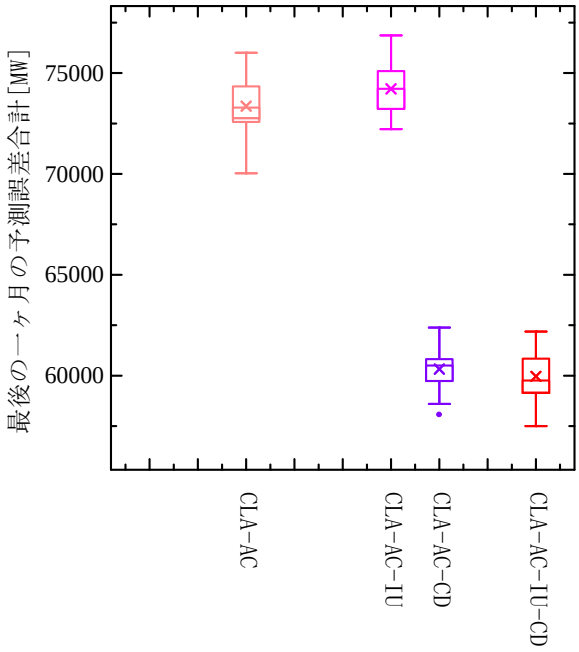
CLA における各提案法の効果を確認する．**図 7.3** に 2019 年 2 月にあたる最後の 28 日間での予測誤差合計を箱ひげ図として出力した結果を示す．まず，全体の結果を示した**図 7.3 (a)** から，今回の実験で基準とした簡素型 CLA は，従来 CLA よりも予測誤差が低いことがわかる．また，各提案法をそれぞれ単独で導入した CLA-AC，CLA-IU，CLA-CD は全て基準の方法である簡素型 CLA よりも予測誤差が低くなることがわかる．これは，その効果の大小には差があるものの，各提案法が実問題においても予測精度を改善できることを示している．次に，**図 7.3 (a)** の一部を拡大した**図 7.3 (b)** から，CLA-AC-CD は CLA-AC および CLA-CD よりも予測誤差が小さく，組み合わせることでより予測精度を改善できることがわかる．一方，CLA-AC-IU および CLA-IU-CD は，それぞれ CLA-AC および CLA-CD と比べると予測誤差が増大することがわかる．CLA-AC-IU は，AC により用意した計算領域を有効活用できることで，IU でのシナプスの更新機会が増えるため，そのシナプス更新をより正確にする仕組みや，構築されたシナプスネットワークを出力に素早く反映できるような仕組みが必要だと考えられる．また，実問題などにおいてシナプス更新の正確度には限界があるため，より内部状態にロバストなデコーダが必要だと考えられる．また，CLA-IU-CD は，セルが不足した状況では IU におけるシナプス更新の正確度が落ちるため，まずセルの不足を解消する必要があると考えられ



(a) 予測誤差合計の箱ひげ図：全体



(b) 予測誤差合計の箱ひげ図：拡大その 1



(c) 予測誤差合計の箱ひげ図：拡大その 2

図 7.3: 実世界の電力消費予測における各提案法の効果

る．もしくは，そのシナプス更新をより正確する仕組みや，さらに内部状態にロバストなデコーダへと改良する必要があると考えられる．そして，**図 7.3 (b)** をさらに拡大した**図 7.3 (c)** から，三つの提案法を適用した CLA-AC-IU-CD が最も良い予測精度を達成することがわかる．前述の提案法を組み合わせた際に生じる問題を，三つの提案法を組み合わせることで解決し，相乗効果が得られたと考えられる．

以上の結果から，三つの提案法がそれぞれ予測誤差を低減し，組み合わせることでより良好な予測精度を実現することを示した．

7.3 結言

電力消費量を予測する問題において，本論文で提案した四つの方法を組み合わせることで，より低い予測誤差を達成できることを示した．第 3 章で提案したカラムのシナプスの適応配置と，第 4 章で提案した不活性セルのシナプス更新，第 6 章で提案したカラムに基づくデコーダの三つの提案法を組み合わせることで，予測精度を改善できることを示した．また，それに加えて第 5 章で提案した予測器の多層化を導入することで，予測誤差のばらつきを抑制することを示した．

第 8 章

結論

8.1　得られた知見

本研究では，CLA において予測器の構成とその取り扱いに関する方法論を構築した．

具体的には，まず様々なデータに対応するため，入力データにあわせてシナプスネットワークを適応的に構築する方法を提案した．次に，予測精度を向上するため，シナプスネットワークの構築を促進する方法，予測器を重ねることで予測の補正を実現し，予測器の表現力を高める方法を提案した．そして，予測速度を向上するため，既存のシナプスネットワークを活用して追加の学習を回避する方法を提案した．時系列データを予測する実験において，従来の CLA よりも時系列予測性能が改善することを示した．また，比較手法として用いた LSTM よりも予測誤差が小さくなることを示した．

第 3 章は，カラムのシナプスに着目した．従来の CLA は，カラムのシナプスの初期設定に起因して，予測精度が悪くなることを示した．また，従来の CLA は，カラムのシナプスの配置を固定するため，偏った値が入力されると，入力データの内部表現に利用されないカラムが生じる．これに対し，予測精度を高めるカラムのシナプスの初期設定を提案した．また，入力データの傾向にあわせて適応的にカラムのシナプスを配置する CLA-AC を提案した．偏りのある入力データを用いた実験結果から，提案法は，入力データにあわせてカラムのシナプスを配置することで，予測誤差を低減することを示した．オンラインでシナプスを適応配置することで，入力データの分布とシナプスの分布を一致させることができ，カラムを有効活用できることを示した．

第 4 章は，セルのシナプスに着目した．従来の CLA は，予測が成功したときのみにシナプスを更新する．具体的には，セルが予測状態から活性状態に遷移したとき，そのセルを予測状態にしたシナプスの永続値を強化する．本研究では，セルのシナプスネットワークの構築を促進するため，不活性セルのシナプスを更新する CLA-IU を提案した．具体的には，セルが予測状態から通常状態に遷移したときにも，永続値を更新する仕組みを導入し，時系列予測精度が改善されることを明らかにした．予測状態から活性状態にならずに通常状態になったセルにおいて，予測の成功を意味するセルと，予測の失敗を意味するセルのケースを分類し，それぞれに適した永続値の更新法を適用することで，予測に失敗するセルを減少させ，予測に成功するセルを増加させる方法が明らかとなった．

第 5 章は，CLA における予測器を積み重ね，相互に連携させることに着目した．CLA の予測器を重ねて，CLA のデータの表現力を高める CLA-DL を提案し，予測精度を改善することを示した．階層化された予測器において，第二層の予測器が第一層の予測器における予測を補正することで，単一層では達成できない予測精度を達成できることを示した．

第 6 章は，カラムと入力データの関係に着目した．従来の CLA は，予測器の内部表現を入力データと同じ形式へデコードするために，学習を伴う手段を利用していた．学習を回避して，予測器の内部表現を入力データと同じ形式にデコードするために，本研究では，カラムに基づくデコーダを提案した．提案法は，空間プーリングによって構築された，入力データを予測器の内部表現にエンコードするためのカラムのシナプスネットワークを活用し，予測器の内部表現を入力データの形式にデコードする．提案するカラムに基づくデコーダを用いる CLA-CD は，構築済みのシナプスを活用するため，追加の学習を回避できるだけでなく，予測誤差の収束を早めることができることを明らかにした．

第 7 章は，第 3 章から第 6 章で提案した CLA-AC，-IU，-DL，-CD を統合した方法について実験，議論した．提案 CLA-AC，-IU，-CD を統合することで予測精度を改善することを示した．さらに，CLA-DL の統合によって予測精度のばらつきを抑制できることを明らかにした．

8.2 今後の課題と展望

8.2.1 短期的な課題

本研究の短期的な課題を以下に述べる．

8.2.1.1 動的なカラムのシナプスの適応配置

第 3 章では，カラムのシナプスを入力データにあわせて適応的に配置することで，予測精度が改善することを示した．一方，この提案法では時点間隔 w をパラメータとして設定する必要があり，これが提案法の性能を大きく左右すると考えられる．w が小さすぎる場合，入力のデータ分布を正しく認識することができず，無駄にシナプスの再配置が繰り返され，予測精度が悪化する．また，w が大きすぎる場合，カラムのシナプスが再配置された後に十分に学習することができず，予測精度が収束しないことが考えられる．適当な w は入力によって異なるため，未知の入力に対して事前に設定することは困難である．これに対して，w を動的に決定することで，シナプスの再配置のタイミングを自動で決定する方法を検討する．具体的には，予測器の内部表現の安定性を見ることで，学習済みの入力であるかどうかを判断することができ，適当な w を設定できると考えられる．

8.2.1.2 CLA-CD における層数と予測器のサイズの検討

第 5 章において，予測器を重ねる CLA-CD が予測を補正することで低い予測誤差を達成することを示した．本論文では，2 層の CLA での結果を示したが，理論上はさらに層数を増やすことも可能である．また，第一層と第二層における予測器のサイズは同一に設定したが，より適当な予測器のサイズがある可能性もある．今後，実問題も含めた様々な時系列データにおいて，それぞれ適当な層数および予測器のサイズについて検討する．これにより，入力の複雑度に対してどのように予測器の構成を設定すべきかを明らかにすることができる．

8.2.1.3　様々な性質を持つデータにおける CLA の性能検証

本研究では，人工の時系列データと，実問題である電力消費量において時系列予測性能が改善することを示した．しかし，本研究で用いたデータは時点間隔によっては定常的なデータに限りなく近づくデータである．例えば，電力消費量のデータは，月ごとの時点間隔で見た場合，平均と分散が変化するため，非定常なデータといえる．一方，年ごとの時点間隔で見た場合，季節ごとの変化などが緩和され，平均と分散の変化は小さくなり，定常なデータに近づく．非定常性が時点間隔によらず強いデータにおいて，CLA がどのような性能を示すか検証する必要がある．CLA はパターン認識器の側面を持つため，新しいパターンが常に現れるような入力値に対して予測が不可能となる．これに対して，学習器内部で入力値の差分を取るなど，入力データを多角的に表現することで過去のパターンと一致する確率を高め，予測を可能にする方法が考えられる．

また，データの線形性および非線形性という側面でも，第 3 章においてその性質が途中で大きく変化する入力を用いたが，徐々に変化する入力については検証していない．そして，本研究で用いた時系列データは全て連続値のデータであり，離散値のデータでの性能は検証していない．これらのデータで CLA の性能を検証することで，CLA の適用可能範囲をより明確化することができると考えられる．

8.2.1.4　学習器のサイズによる CLA の予測精度の検証

第 6 章および第 7 章において，入力によって予測精度が最善となる CLA のセル数が変化することを示した．これは，一般的にセル数が多いほど，内部表現可能なデータの複雑度は増すが，セル数が増えることで初期の学習に遅れが生じる傾向があるためだと考えられる．この点について，より異なるデータ，および異なるセル数で検証する必要がある．また，同様に学習器のサイズのパラメータとして，カラム数 n_c が設定されている．このパラメータの増減による予測精度への影響もあわせて検証することで，データによってどのようなサイズの学習器が適当なのか，を確認することができると考えられる．

8.2.2　長期的な課題

本研究について，実用性の向上と実応用を考慮した長期的な課題を以下に述べる．

8.2.2.1 問題の拡張

本研究では，実世界の時系列データとして電力消費量を予測する問題を用いた．一方，より現実的な問題を考えた場合，電力消費量だけでなく，気温や湿度などのデータも存在すると考えられる．このような複数データの関係性を予測に活用することができれば，より予測精度を改善することができると考えられる．また，そのような複数の時系列データを統合して抽象的に予測する仕組みは，人間の脳における予測に近いと考えられ，予測知能の加速および発展に貢献できると考えられる．

本論文で提案した手法を複数の時系列データの予測に用いるには，複数のデータを同じ予測器に同時に入力する方法が考えられる．しかし，同じ予測器に複数のデータを直接入力した場合，予測が混同して精度が悪化する可能性がある．そこで，CLA-CD の第一層においてデータごとに予測器を分離し，第二層でそれらを統合する仕組みを検討する．これにより，単一のデータによる予測と複数のデータによる予測を同時に実現することができ，その双方の予測からより高い予測精度を実現できる可能性がある．また，複数のデータの関連性が低い場合，それらのデータによる予測表現は不安定となるため，単一のデータによる予測を重視することも考えられる．

8.2.2.2 予測の因果関係を示すことによる意思決定支援

本研究では，時系列予測性能の向上を目的として方法論を構築した．しかし，実際に企業などに技術を提供する際，人間が納得して予測結果を意思決定に利用できる方法論が重要となる．そのため，予測の解釈性をもたらすことが必要となるが，CLA は予測結果のみを出力し，内部表現も離散的な集合であるため，その予測結果を説明できない．

そこで，予測結果だけでなくその根拠を出力する手法の検討が必要である．例えば，進化ルール学習を用いた方法が考えられる．進化ルール学習は，離散データから人間にわかりやすい IF-THEN ルールを抽出する手段として長けている．CLA の内部表現は離散的な集合であるため，進化ルール学習との親和性も高いと考えられる．電力消費量の予測において，ただその予測値を出力するだけでなく，学習からの根拠として気温の上昇なども出力することができるため，より納得しやすい予測結果を出力することができると考えられる．また，進化ルール学習が機能する入力データ長には限界があるため，CLA の内部

表現をそのまま入力するのではなく，それを圧縮するような方法論も必要となる．

以上のように，今後，より実用性の向上と実応用を考慮した時系列予測の研究に取り組むことで，予測性能の向上だけでなく，この技術を活用する方々に貢献することで社会的役割を果たすとともに，予測知能の成熟および発展に貢献していきたいと考えている．

謝辞

本研究の全般的な進行および博士論文の執筆にあたり，丁寧なご指導を賜りました佐藤寛之准教授に心からの感謝とともに厚く御礼申し上げます．また，本研究について多くの議論とそれによる多くの助言を頂きました髙玉圭樹教授に感謝申し上げます．また，貴重なお時間を頂戴し，本論文の審査をして下さる西野哲朗教授，庄野逸教授，高橋裕樹准教授に感謝申し上げます．そして，日頃からご指摘とご助力を頂きました佐藤研究室，髙玉研究室の皆様に感謝の気持ちと御礼を申し上げたく，謝辞に代えさせていただきます．

参考文献

[1] J. L. Elman: Finding Structure in Time, *Cognitive Science*, Vol. 14, Issue 2, pp. 179–211, 1990.

[2] J. T. Connor, R. D. Martin, and L. E. Atlas: Recurrent Neural Networks and Robust Time Series Prediction, *IEEE Trans. on Neural Networks*, Vol. 5, No. 2, pp. 240–254, 1994.

[3] S. Hochreiter and J. Schmidhuber: Long Short-Term Memory, *Neural Computation*, Vol. 9, No. 8, pp. 1735–1780, 1997.

[4] J. Hawkins and S. Blakeslee: *On Intelligence*, Times Books, 2004.

[5] S. Ahmad and J. Hawkins: *Properties of Sparse Distributed Representations and their Application to Hierarchical Temporal Memory*, Numenta, pp. 1–18, 2015.

[6] J. Hawkins, A. Subutai, and D. Dubinsky: *Hierarchical temporal memory including HTM cortical learning algorithms*, Technical report, Numenta, Inc, 2010.

[7] J. Hawkins and A. Subutai: Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex, *Frontiers in Neural Circuits*, Vol. 10, pp. 1–13, 2016.

[8] NuPIC: https://github.com/numenta/nupic (2020/2/27 access)

[9] Y. Cui, S. Ahmad, and J. Hawkins: Continuous Online Sequence Learning with an Unsupervised Neural Network Model, *Neural Computation*, Vol. 28, Issue. 11, pp. 2474–2504, 2016.

[10] New York ISO (Independent System Operator): http://mis.nyiso.com/public/ (2020/2/27 access)

[11] L. R. Rabiner and B. H. Juang: An Introduction to Hidden Markov Models,

IEEE Acoustics, Speech & Signal Processing Magazine, Vol. 3, No. 1, pp. 4–16, 1986.

[12] L. R. Rabiner: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, *In Proc. of the IEEE*, Vol. 77, No. 2, pp. 257–285, 1989.

[13] A. Gupta and B. Dhingra: Stock Market Prediction Using Hidden Markov Models, *In Proc. of Students Conf. on Engineering and Systems*, pp. 1–4, 2012.

[14] A. Krogh, B. Larsson, G. Von Heijne, and E. L. L. Sonnhammer: Predicting Transmembrane Protein Topology with a Hidden Markov Model: Application to Complete Genomes, *Journal of Molecular Biology*, Vol. 305, Issue 3, pp. 567–580, 2001.

[15] P. Dagum, A. Galper, and E. Horvitz: Dynamic Network Models for Forecasting, *In Proc. of the 8th Int'l Conf. on uncertainty in artificial intelligence (UAI1992)*, pp. 41–48, 1992.

[16] K. Murphy, Dynamic Bayesian Networks: *Representation, Inference and Learning*, Ph. D. thesis, University of California Berkeley, Computer Science Division, 2002.

[17] A. Metallinou, S. Lee, and S. Narayanan: Decision Level Combination of Multiple Modalities for Recognition and Analysis of Emotional Expression, *In Proc. of the 2010 IEEE Int'l Conf. on Acoustics, Speech and Signal Processing (ICASSP 2010)*, pp. 2462–2465, 2010.

[18] Y. Zhang and Q. Ji: Active and Dynamic Information Fusion for Facial Expression Understanding from Image Sequences, *IEEE Trans. on Pattern Analysis and Machine Intelligence* , Vol. 27, No. 5, pp. 1–16, 2005.

[19] M. Zou and S. D. Conzen: A New Dynamic Bayesian Network (DBN) Approach for Identifying Gene Regulatory Networks from Time Course Microarray Data, *Bioinformatics*, Vol. 21, No. 1, pp. 71–79, 2005.

[20] J. Pearl: *Bayesian networks: A model of self-activated memory for evidential reasoning*, Technical Report. CSD-850021, R-43, 1985.

[21] T. S. Furey, N. Cristianini, N. Duffy, D. W. Bednarski, M. Schummer, and D.

Haussler: Support Vector Machine Classification and Validation of Cancer Tissue Samples Using Microarray Expression Data, *Bioinformatics*, Vol. 16, Issue 10, pp. 906–914, 2000.

[22] M. Wöllmer, F. Eyben, B. Schuller, E. D. -Cowie, and R. Cowie: Data-Driven Clustering in Emotional Space for Affect Recognition Using Discriminatively Trained LSTM Networks, *In Proc. of InterSpeech*, pp. 1595–1598, 2009.

[23] J. H. Min and Y. -C. Lee: Bankruptcy Prediction Using Support Vector Machine With Optimal Choice of Kernel Function Parameters, *Expert Systems with Applications*, Vol. 28, Issue 4, pp. 603–614, 2005.

[24] S. Hua and Z. Sun: Support Vector Machine Approach for Protein Subcellular Localization Prediction, *Bioinformatics*, Vol. 17, No. 8, pp. 721–728, 2001.

[25] T. Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur: Recurrent Neural Network Based Language Model, *In Proc. of InterSpeech*, pp. 1045–1048, 2010.

[26] C. -M. Kuan and T. Liu: Forecasting Exchange Rates Using Feedforward and Recurrent Neural Networks, *Journal of Applied Econometrics*, Vol. 10, Issue 4, pp. 347–364, 1995.

[27] B. A. Pearlmutter: Learning State Space Trajectories in Recurrent Neural Networks, *Neural Computation*, Vol. 1, Issue 2, pp. 263–269, 1989.

[28] R. J. Williams: Complexity of Exact Gradient Computation Algorithms for Recurrent Neural Networks, Technical Report, Boston: Northeastern University, College of Computer Science, 1989.

[29] S. E. Fahlman: The Recurrent Cascade-correlation Architecture, *In Proc. of Advances in Neural Information Processing Systems 3 (NIPS 1990)*, pp. 190–196, 1991.

[30] J. Schmidhuber: A Fixed Size Storage $O(n^3)$ Time Complexity Learning Algorithm for Fully Recurrent Continually Running Networks, *Neural Computation*, Vol. 4, Issue 2, pp. 243–248, 1992.

[31] B. A. Pearlmutter: Gradient Calculations for Dynamic Recurrent Neural Net-

works: A Survey, *IEEE Trans. on Neural Networks*, Vol. 6, Issue 5, pp. 1212–1228, 1995.

[32] K. J. Lang, A. H. Waibel, and G. E. Hinton: A Time-delay Neural Network Architecture for Isolated Word Recognition, *Neural Network*, Vol. 3, Issue 1, pp. 23–43, 1990.

[33] B. De Vries and J. C. Principe: A Theory for Neural Networks With Time Delays, *In Proc. of Advances in Neural Information Processing Systems 3 (NIPS 1990)*, pp. 162–168, 1991.

[34] T. A. Plate: Holographic Recurrent Networks, *In Proc. on Advances in Neural Information Processing Systems 5 (NIPS 1992)*, pp. 34–41, 1993.

[35] T. Lin, B. G. Horne, P. Tino, and C. L. Giles: Learning Long-Term Dependencies in NARX Recurrent Neural Networks, *IEEE Trans. on Neural Networks*, Vol. 7, No. 6, pp. 1329–1338, 1996.

[36] N. Liang, G. Huang, P. Saratchandran and N. Sundararajan: A Fast and Accurate Online Sequential Learning Algorithm for Feedforward Networks, *in IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411–1423, 2006.

[37] H. Jaeger and H. Haas: Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, *Science*, 304, pp. 78–80, 2004.

[38] Y. Fan, Y. Qian, F. Xie, and F. K. Soong: TTS Synthesis with Bidirectional LSTM based Recurrent Neural Networks, *In Proc. of InterSpeech*, pp. 1964–1968, 2014.

[39] M. Sundermeyer, R. Schlüter, and H. Ney: LSTM Neural Networks for Language Modeling, *In Proc. of InterSpeech*, pp. 194–197, 2012.

[40] A. Graves and J. Schmidhuber: Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures, *Neural Network*, Vol. 18, Issue 5-6, pp. 602–610, 2005.

[41] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard and L. D. Jackel: Backpropagation Applied to Handwritten Zip Code Recognition, *Neural Computation*, Vol. 1, Issue 4, pp. 541–551, 1989.

[42] J. Bradbury, S. Merity, C. Xiong, and R. Socher: *Quasi-Recurrent Neural Networks*, in CoRR, arXiv:1611.01576v2, 2016.

[43] A. Oord, N. Kalchbrenner, and K. Kavukcuoglu: *Pixel Recurrent Neural Networks*, in CoRR, arXiv:1601.06759v3, 2016.

[44] A. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu: *WaveNet: A Generative Model for Raw Audio*, in CoRR, arXiv:1609.03499v2, 2016.

[45] D. George, W. Lehrach, K. Kansky, M. L. -Gredilla, C. Laan, B. Marthi, X. Lou, Z. Meng, Y. Liu, H. Wang, A. Lavin, and D. S. Phoenix: A Generative Vision Model That Trains With High Data Efficiency and Breaks Text-based CAPTCHAs. *Science*, Vol. 358, Issue 6368, 2017.

[46] R. P. N. Rao and D. H. Ballard: Predictive Coding in the Visual Cortex: A Functional Interpretation of Some Extra-classical Receptive-field Effects, *Nature Neuroscience*, Vol. 2, No. 1, pp. 79–87, 1999.

[47] W. Lotter, G. Kreiman, and D. Cox: *Deep Predictive Coding Networks for Video Prediction and Unsupervised Learning*, in CoRR, arXiv:1605.08104v5, 2016

[48] H. Jaeger: *The "Echo State" Approach to Analysing and Training Recurrent Neural Networks – With An Erratum Note*, Technical Report 154, German National Research Center for Information Technology, 2001.

[49] W. Maass, T. Natschläger, and H. Markram: Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations, *Neural Computation*, Vol. 14, Issue 11, pp. 2531–2560, 2002.

[50] I. Sutskever, O. Vinyals, and Q. V. Le: Sequence to Sequence Learning with Neural Networks, Advances in neural information processing systems, pp. 3104–3112, 2014.

[51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin: Attention Is All You Need, Advances in neural information processing systems, pp. 5998–6008, 2017.

[52] J. Devlin, M. - W. Chang, K. Lee, and K. Toutanova: BERT: Pre-training of

Deep Bidirectional Transformers for Language Understanding, arXiv preprint arXiv:1810.04805, 2018.

[53] N. Wu, B. Green, X. Ben, and S. O'Banion: Deep Transformer Models for Time Series Forecasting: The Influenza Prevalence Case, arXiv preprint arXiv:2001.08317, 2020.

[54] K.Fukushima: Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position, *Biological Cybernetics*, Vol. 36, Issue 4, pp. 193–202, 1982.

[55] D. George: *How the Brain Might Work: A Hierarchical and Temporal Model for Learning and Recognition*, Ph. D. dissertation, Stanford University, 2008.

[56] Y. Ichisugi: A Cerebral Cortex Model that Self-Organizes Conditional Probability Tables and Executes Belief Propagation, *2007 Int'l Joint Conf. on Neural Networks*, pp. 178–183, 2007.

[57] N. Leadholm, M. Lewis and S. Ahmad: Grid Cell Path Integration For Movement-Based Visual Object Recognition, The 32nd British Machine Vision Conference (BMVC 2021), 12 pages, 2021.

[58] M. Lewis, S. Purdy, S. Ahmad and J. Hawkins: Locations in the Neocortex: A Theory of Sensorimotor Object Recognition Using Cortical Grid Cells, *Frontiers in Neural Circuits*, Vol. 13, 18 pages, 2019.

[59] J. Hawkins, M. Lewis, M. Klukas, S. Purdy and S. Ahmad: A Framework for Intelligence and Cortical Function Based on Grid Cells in the Neocortex, *Frontiers in Neural Circuits*, Vol. 12, 14 pages, 2019.

[60] M. A. Abbas: Improving Deep Learning Performance Using Random Forest HTM Cortical Learning Algorithm, 2018 First International Workshop on Deep and Representation Learning (IWDRL), pp. 13–18, 2018.

[61] M. Ahmed: A Robot Obstacle Avoidance Method based on Random Forest HTM Cortical Learning Algorithm, Webology, Vol. 17, No. 2, pp. 788-803, 2020.

[62] A. M. Zyarah and D. Kudithipudi: Neuromemrisitive Architecture of HTM with On-Device Learning and Neurogenesis, *ACM Journal on Emerging Technologies*

in Computing Systems, Vol. 15, No. 3, Article 24, 24 pages, 2019.

[63] Y. Cui, S. Ahmad, and J. Hawkins: The HTM Spatial Pooler–A Neocortical Algorithm for Online Sparse Distributed Coding, *Frontiers in Computational Neuroscience*, Vol. 11, 15 pages, 2017.

[64] D. P. Kingma and J. Ba: Adam: A Method for Stochastic Optimization, in CoRR, arXiv:1412.6980, 2014.

[65] K. Hornik: Approximation Capabilities of Multilayer Feedforward Networks, *Neural Network*, Vol. 4, No. 2, pp. 251–257, 1991.

[66] Adam Filion: Data Analytics with MATLAB Webinar Files (https://www.mathworks.com/matlabcentral/ fileexchange/49063-data-analytics-with-matlab-webinar-files), MATLAB Central File Exchange (2020/2/27 access)

[67] T. Aoki, K. Takadama, and H. Sato: Study on Simple Cortical Learning Algorithm and Prediction Accuracy Improvement, *SSI 2017, The Society of Instrument and Control Engineers*, pp. 135–140, 2017. (Japanese)

[68] Keras: https://github.com/keras-team/keras (2020/2/27 access)

[69] S. Suzugamine, T. Aoki, K. Takadama, and H. Sato: Self-Structured Cortical Learning Algorithm by Dynamically Adjusting Columns and Cells, *Journal of Advanced Computational Intelligence and Intelligent Informatics (JACIII)*, Volume 24, Issue 2, pp. 185–198, 2020.

[70] Electricity Load Diagrams 2011-2014 Data Set, UCI Machine Learning Repository, https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014 (2020/7/31 access)

関連発表

本論文に関連する発表論文

査読付き学術雑誌論文

[1] Takeru Aoki, Keiki Takadama, and Hiroyuki Sato, “Adaptive Synapse Arrangement in Cortical Learning Algorithm,” Journal of Advanced Computational Intelligence and Intelligent Informatics (JACIII), Vol. 25, No. 4, pp. 450-466, 2021. (第 3 章に関連)

査読付き国際会議発表論文

[1] Takeru Aoki, Keiki Takadama, and Hiroyuki Sato, “A Study on Synapse Update of Inactive Cells in Cortical Learning Algorithm,” The 2017 International Symposium on Nonlinear Theory and Its Applications (NOLTA2017), pp. 391-394, 2017. (第 4 章に関連)

[2] Takeru Aoki, Keiki Takadama and Hiroyuki Sato, “Column-Based Decoder of Internal Prediction Representation in Cortical Learning Algorithms,” Joint 11th International Conference on Soft Computing and Intelligent Systems and 21st International Symposium on Advanced Intelligent Systems (SCIS&ISIS2020), pp. 1-7, 2020. (第 6 章に関連)

[3] Takeru Aoki, Keiki Takadama, and Hiroyuki Sato, “A preliminary study on a multi-layered cortical learning algorithm,” The 7th UEC Seminar in ASEAN, 2020 and The 2nd ASEAN-UEC Workshop on Energy and AI, 2020. (第 5 章

に関連)

[4] Takeru Aoki, Keiki Takadama, and Hiroyuki Sato, “Double-layered Cortical Learning Algorithm for Time-series Data Prediction,”in Proc. of 13th EAI International Conference on Bio-inspired Information and Communications Technologies (BICT2021), 2021. (第 5 章に関連)

国内口頭発表

[1] 青木 健，高玉 圭樹，佐藤 寛之,”不活性セルのシナプス更新による大脳新皮質アルゴリズムの予測精度向上に関する一検討,”第 44 回 知能システムシンポジウム，計測自動制御学会，in CD-ROM，SY0004/17/A3-1，6 pages，2017. (第 4 章に関連)

[2] 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 2017，pp. 135-140, 2017. (第 3 章に関連)

[3] 青木 健，高玉 圭樹，佐藤 寛之,”大脳新皮質学習におけるカラムに基づく予測表現デコーダに関する検討, ” 計測自動制御学会 システム・情報部門 学術講演会 2020，pp. 130-135, 2020. (第 6 章に関連)

[4] 青木健，高玉圭樹，佐藤寛之,” 大脳新皮質学習における多層化に関する検討, ” 計測自動制御学会 第 48 回 知能システムシンポジウム，5 pages, 2021. (第 5 章に関連)

[5] 青木健，高玉圭樹，佐藤寛之,”大脳新皮質学習の階層化における抑制フィードバックの検討, ” 計測自動制御学会 システム・情報部門 学術講演会 2020，6 pages, 2021. (第 5 章に関連)

その他の論文

査読付き学術雑誌論文

[1] Sotetsu Suzugamine, Takeru Aoki, Keiki Takadama, and Hiroyuki Sato, “Self-Structured Cortical Learning Algorithm by Dynamically Adjusting Columns and Cells,” Journal of Advanced Computational Intelligence and Intelligent Informatics (JACIII), Volume 24, Issue 2, pp. 185-198, 2020.

査読付き国際会議発表論文

[1] Sotetsu Suzugamine, Takeru Aoki, Keiki Takadama, and Hiroyuki Sato, “A Study on a Cortical Learning Algorithm Dynamically Adjusting Columns and Cells,” Proc. of Joint 10th International Conference on Soft Computing and Intelligent Systems and 19th International Symposium on Advanced Intelligent Systems (SCIS&ISIS2018), pp.478-484, 2018.

[2] Akihiko Nagashima, Takeru Aoki, Keiki Takadama, and Hiroyuki Sato, “ A Study on Multivariate CLA Complementing Missing Time-series Data,” The 7th UEC Seminar in ASEAN, 2020 and The 2nd ASEAN-UEC Workshop on Energy and AI, 2020.

[3] Kazushi Fujino, Takeru Aoki, Keiki Takadama, and Hiroyuki Sato, “Cortical Learning Based Action-decision Under Uncertain Environment,” The 3rd ASEAN-UEC Workshop on Informatics and Engineering for SDGs, 2pages, 2021.

[4] Yuki Goto, Takeru Aoki, Keiki Takadama, and Hiroyuki Sato, “A Forgetting Mechanism in Cortical Learning Algorithm for Time-series Forecast,” The 3rd ASEAN-UEC Workshop on Informatics and Engineering for SDGs, 2pages, 2021.

国内口頭発表

[1] 鈴ヶ嶺聡哲，青木健，高玉圭樹，佐藤寛之, “大脳新皮質学習におけるカラムとセルの動的構成に関する検討, ” 第 117 回数理モデル化と問題解決 (MPS) 研究会報告, 情報処理学会，Vol. 2018-MPS-117, No. 24, pp. 1-2, 2018.

[2] 鈴ヶ嶺聡哲，青木健，高玉圭樹，佐藤寛之, “自己構成型の大脳新皮質学習アルゴリズムに関する検討, ” 計測自動制御学会（SICE)，システム・情報部門，第 13 回コンピューテーショナル・インテリジェンス研究会, pp.51-58,2018.

[3] 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，in USB memory, 2018.

[4] 鈴ヶ嶺聡哲，青木健，高玉圭樹，佐藤寛之, “大脳新皮質学習におけるカラムとセルの自己構成法の動作解析, ” 計測自動制御学会 システム・情報部門 学術講演会 2018，in USB memory, 2018.

[5] 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適応型シナプス配置法の検討, ” 計測自動制御学会 第 46 回 知能システムシンポジウム，in CD-ROM, 6 pages, 2019.

[6] 鈴ヶ嶺聡哲，青木健，高玉圭樹，佐藤寛之, “自己構成型大脳新皮質学習における時間遅れシナプスの検討, ” 計測自動制御学会 第 46 回 知能システムシンポジウム，in CD-ROM, 6 pages, 2019.

[7] 長島晶彦，青木健，高玉圭樹，佐藤寛之, “大脳新皮質学習における異なる時系列データの複合予測に関する基礎検討, ”計測自動制御学会 (SICE)，システム・情報部門，第 15 回コンピューテーショナル・インテリジェンス研究会, pp. 5-12, 2019.

[8] 長島晶彦，青木健，高玉圭樹，佐藤寛之,”多変量大脳新皮質学習によるデータの連続欠損に対する予測持続に関する検討, ” 計測自動制御学会 システム・情報部門学術講演会 2020，pp. 124-129, 2020.

[9] 藤野和志，青木健，高玉圭樹，佐藤寛之, “時系列予測に基づく行動決定のための大脳新皮質学習に関する基礎検討,” 第 17 回コンピューテーショナル・インテリジェンス研究会，システム・情報部門，計測自動制御学会，pp. 63-67, 2021.

受賞歴

[1] SSI 優秀論文賞，計測自動制御学会 システム・情報部門 学術講演会 2017.

[2] 平成 29 年度 電気通信大学 学生表彰, 2018.

[3] Best Paper Award, The Joint 10th International Conference on Soft Computing and Intelligent Systems and 19th International Symposium on Advanced Intelligent Systems (SCIS&ISIS2018), 2018.

[4] 平成 30 年度 目黒会賞, 2019.

[5] Young Researcher Encouragement Award, The 2nd ASEAN-UEC Workshop on Energy and AI, 2020.

[6] JACIII 優秀論文賞 2021（JACIII Best Paper Award 2021）, 2021.