2017 International Symposium on Nonlinear Theory and Its Applications,
NOLTA2017, Cancun, Mexico, December 4-7, 2017

A Study on Synapse Update of Inactive Cells in Cortical Learning Algorithm

Takeru Aoki, Keiki Takadama, and Hiroyuki Sato

Graduate School of Informatics and Engineering, The University of Electro-Communications
1-5-1 Chofugaoka, Chofu, Tokyo 182-8585
Email: takeru-aoki@uec.ac.jp, keiki@inf.uec.ac.jp, h.sato@uec.ac.jp

Abstract—Cortical learning algorithm (CLA) is a time-series data prediction algorithm based on the behavior of human neocortex. CLA has many cells connected by synapses, receives a time-series data and predicts the data coming next while updating synapse network. The conventional CLA only update synapses of active cells contributed to the prediction during the learning, and other synapses of inactive cells are neglected and not updated. To encourage the synapse network construction and improve the prediction accuracy of CLA, in this work we propose methods to update synapses of inactive cells and verify its effectiveness on test time-series data with/without noise.

1. Introduction

The hierarchical temporal memory (HTM) is a conceptual framework for the time-series data prediction [1]. The cortical learning algorithm (CLA) modeling the human neocortex is one of the algorithms based on HTM [2]. So far, CLA has been applied to the time-series data prediction and the anomaly detection [3] and known as one of their promising approaches. Compared with RNN and LSTM [4], the CLA model has a set of columns grouping multiple neurons and neuronal dendrites and is closer to the neocortex. CLA uses many cells corresponding to neurons and synapses connecting them. CLA receives data and predicts data coming next by updating synapses' permanence values determining connection or disconnection of their synapses. Each cell has three states which are *normal*, *active*, and *predictive*. The conventional CLA updates synapse permanence values only when its cell is changed from the predictive state to the active one by the prediction success. On the other hand, the conventional CLA does not update synapse permanence values when its cell is changed from the predictive state to the normal one. Consequently, the construction of an appropriate synapse network to successfully predict the next coming data becomes insufficient, and it causes a low prediction accuracy. A solution to overcome this problem would be to update the synapse network based on all predictions and their results. There is a possibility that the construction of an appropriate synapse network can be encouraged and the prediction accuracy can be improved by updating synapse permanence values of normal (inactive) cells which transit from the predictive state to the normal one.

To improve the time-series data prediction accuracy of CLA, in this work we propose three methods to update synapse permanence values of normal inactive cells which transit from the predictive state to the normal one. To verify the effectiveness of the proposed methods, we conduct experiments to predict sine wave time-series data with/without noise and compares the prediction accuracies of the conventional CLA and CLAs with the three proposed methods. Also, to analyze the effectiveness of the proposed methods, we observe the number of state transitions among active, predictive and normal states.

2. Cortical Learning Algorithm (CLA)

2.1. Overview

CLA consists of three elements: *cell*, *column*, and *region*. A *cell* models a neuron, a *column* involves multiple cells, and the *region* involves a number of columns. Each cell has three states: *normal*, *active*, and *predictive*. Each column also has three states: *normal*, *active candidate*, and *active*. Each column has a *segment* bundling a set of synapses. Each of the column synapses is for a connection with a bit of the binary input data. On the other hand, each cell has several segments, and each synapse in them is for a connection with a cell. A synapse has a permanence value determining connection or disconnection in the range $[0, 1]$, and CLA controls it during the learning. When the permanence value is greater than a threshold, its synapse is connected. Otherwise, its synapse is disconnected.

First, CLA converts a real value input data $input(t)$ at a time t into a binary string and activates several columns in a proccess referred to as the *spatial pooling*. Next, CLA determines active cells in the active columns and predictive cells in a proccess referred to as the *temporal pooling*. The predictive value $predect(t)$ is calculated with the obtained set of predicted cells. In the following, we briefly describe the spatial pooling and temporal pooling in CLA.

2.2. Spatial Pooling

In the spatial pooling, first we pick up all columns having at least one synapse connected with a bit one in the binary input data as active candidate columns. To select a set of active columns from the active candidate ones, we rank active candidate columns in the order of the number of synapses connected to a bit one in the binary input data.

Then we activate the top AC^{max} columns from active candidate ones. Finally, we update synapse permanence values of the active columns. Concretely, we increment permanence values of synapses connected to a bit one in the binary input data by Δp_c^+. Also, we decrement permanence values of synapses connected to a bit zero by Δp_c^-. The permanence value p_c is updated by

$$p'_c = \begin{cases} p_c + \Delta p_c^+ & \text{for synapses with a bit one,} \\ p_c - \Delta p_c^- & \text{for synapses with a bit zero,} \end{cases} \tag{1}$$

where, p'_c is the updated permanence value. Synpses with permanence value p_c greater than a threshold T_c are connected, and others are disconnected.

2.3. Temporal Pooling

In the temporal pooling, first we activate all predictive cells by the previous input data at time $t-1$ in each active column. For active columns having no predictive cells, we activate all cells in their columns. Next, for the active cells changed from the predictive state, we update their synapse permanence values. We increment permanence values of synapses connected with active cells in the previous input data at the time $t-1$ by Δp_s. Also, we decrement permanence values of synapses connected with normal cells at the time $t-1$ by Δp_s. The permanence value p_s is updated by

$$p'_s = \begin{cases} p_s + \Delta p_s & \text{for synapses with an active cell,} \\ p_s - \Delta p_s & \text{for synapses with a normal cell,} \end{cases} \tag{2}$$

where, p'_s is the updated permanence value. Synpses with permanence value p_s greater than a threshold T_s are connected, others are disconnected.

Finally, to determine the predictive cells, we change the state of cells having more than Ac synapses connected to active cells to the predictive state.

2.4. Problems in the Conventional CLA

In this work, we focus on the condition to update synapse permanence values in the temporal pooling. **Fig. 1** shows the three cell states and the transit conditions from the predictive state to the active and the normal ones. In the case I, the conventional CLA updates synapse permanence values of predictive cells when their columns are active ones. In the case II, the conventional CLA does not update synapse permanence values of predictive cells when their columns are the normal state through without the active candidate one. Also, in the case III, the conventional CLA does not update synapse permanence values of predictive cells when their columns are the normal state through the active candidate one. Thus, the conventional CLA updates the synapse network only in the case I. Since the synapse network is not updated in the cases II and III, the construction of an appropriate synapse network to predict the data coming next would be inefficient, and it causes a low prediction accuracy. There is a possibility that we can improve the prediction accuracy of CLA by updating permanence values of synapses even in the two cases II and III.

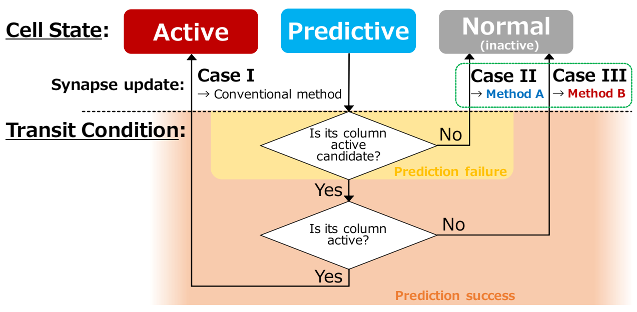


Figure 1: Transit conditions from the predictive state

3. Proposal: Synapse Update of Inactive Cells

To improve the prediction accuracy of CLA, we propose three methods to update synapses of normal inactive cells which transit from the predictive state to the normal one. As shown in **Fig. 1**, the proposed methods A and B are applied to the cases II and III, respectively. Also, the proposed method C is applied to the both cases II and III.

3.1. Three Methods

Method A updates synapse permanence values of the predictive cells for the case II. The method A weakens synapse connections since the prediction failures cause the state transition from the predictive state to the normal one in the case II. In the method A, we decrement the permanence values of synapses with active cells at the previous time step $t-1$ by Δp_s, and increment the permanence values of synapses with normal cells at the previous time step by Δp_s. The synapse permanence value is updated by

$$p'_s = \begin{cases} p_s - \Delta p_s & \text{for synapses with an active cell,} \\ p_s + \Delta p_s & \text{for synapses with a normal cell,} \end{cases} \tag{3}$$

where, p_s and p'_s are the synapse permanence values before and after the update, respectively.

Method B updates synapse permanence values of the predictive cells for the case III. The method B strengthens synapse connections since their predictions are succeed, though the active column selection from active candidate ones causes the state transition from the predictive state to the normal one in the case III. In the method B, we increment the permanence values of synapses with active cells at the previous time step $t-1$ by Δp_s, and decrement the permanence values of synapses with normal cells at the previous time step by Δp_s. The synapse permanence value is updated by

$$p'_s = \begin{cases} p_s + \Delta p_s & \text{for synapses with an active cell,} \\ p_s - \Delta p_s & \text{for synapses with a normal cell,} \end{cases} \tag{4}$$

where, p_s and p'_s are the synapse permanence values before and after the update, respectively.

Method C updates using both of the methods A and B in the case II and III.

3.2. Expected Effects

Since the conventional CLA updates synapse network only in the case I, the construction of the synapse network is inefficient. It causes an inefficient learning process. On the other hand, since the proposed methods update synapse network also in the cases II and III, we can expect to encourage the construction of the synapse network in CLA. In the proposed method A for the case II, we can expect the reduction of the prediction failures since the prediction failures are reflected the synapse network construction. In the proposed method B for the case III, we can expect the increase of the prediction success since the prediction successes are reflected the synapse network construction even active candidate columns with predictive cells do not finally become active columns by influence such as noise on the input data. Furthermore, in the proposed method C, we expect a synergistic and cooperative effect by combining the methods A and B.

4. Experimental Setup

4.1. Test Input Data

In this work, we employ two test input data. The first one is the sine wave given by

$$input_1(t) = \sin\left(\frac{(t-1)\cdot\pi}{50}\right), \tag{5}$$

where, t is the time. One cycle of the sine wave needs a period $t = 100$. The second input data is given by

$$input_2(t,\delta) = input_1(t) + noise(\delta), \tag{6}$$

where, $noise(\delta)$ is an uniform random number in the range $[-\delta, +\delta]$. For the $input_2(t,\delta)$, we aims to predict $input_1(t)$ excluding $noise(\delta)$. In this work, $\delta = 0.01$ is employed.

4.2. Parameters

We use 2048 columns for the region, and each column has 32 cells. Total input time in each experiment is 10^4. For the spatial pooling, the segment generation radius for each column is set to 16, and the synapse generation ratio in the radius is set to 0.8. The maximum number of active columns for each input at the time t is set to $AC^{max} = 40$. The increment and decrement permanence values are $\Delta p_c^+ = 0.05$ and $\Delta p_c^- = 0.025225$, respectively. Also, the threshold on the border between the connection and the disconnection of each synapse is set to $T_c = 0.1$. For the temporal pooling, the initial number of synapses in each segment is 20, the initial synapse permanence value is 0.21, the increment and decrement permanence values are $\Delta p_s = 0.1$, and the threshold on the border between the connection and the disconnection of each synapse is set to $T_s = 0.5$. Also, to determine predictive cells, the number of synapses connected with active cells is set to $Ac = 15$.

4.3. Evaluation Metrics

The main metric to evaluate the prediction accuracy is the prediction error. The prediction error is the difference between the prediction value $predict(t)$ and the actual input data $input(t+1)$. In this work, the sum of prediction errors in the every $100t$ input data is calculated by

$$e(T) = \sum_{t=100T-99}^{100T} |predict(t) - input(t+1)|. \tag{7}$$

$e(T)$ is calculated in the range $T = [1, 100]$.

As additional evaluation metrics, we also observe the number of state transitions to cases II and III shown in **Fig. 1**. In the case II, since predictive cells transit to the normal ones due to the prediction failure, the smaller number of the cell transition, the better. On the other hand, in the case of III, predictive cells transit to the normal one since CLA does not finally activate their columns even they are active candidate ones indicating the prediction successes. Therefore, the larger number of state transitions, the better.

5. Experimental Results and Discussion

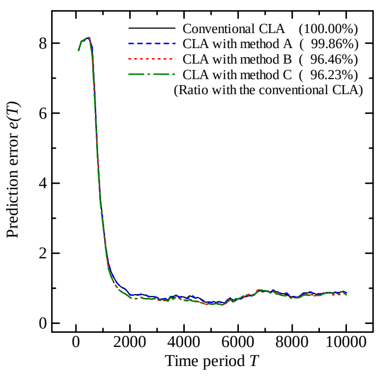
5.1. Time-Series Data Prediction without Noise

Fig. 2 shows the results on $input_1(t)$ without noise. **Fig. 2 (a)** shows the results of the prediction errors on $input_1(t)$ without noise. For the four CLAs, the total prediction error ratios are shown with the graph legends. First, we can see that the three proposed methods achieve lower prediction errors than the conventional CLA. Also, the proposed method C achieves the lowest prediction error among the four CLAs. This result reveals that the proposed synapse updates for the cases II and III improve the prediction accuracy.

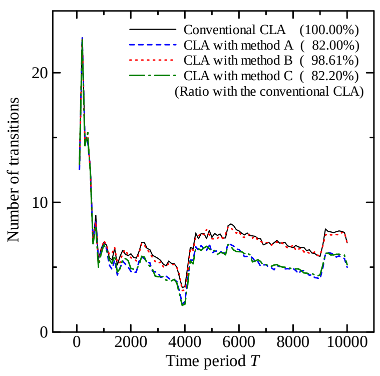
Next, **Fig. 2 (b)** shows the number of cell state transitions to the case II. Since cell state transitions to the case II indicates prediction failures, the lower number of the transitions, the better result. From the result, we can see that the three proposed methods show a lower number of transitions than the conventional CLA. Also, the method A achieves a lower number of transitions than the method B. This result reveals that the method A weakening synapse connections causing the prediction failures contributes to decreasing the prediction failures.

Finally, **Fig. 2 (c)** shows the number of cell transitions to the case III. Since cell state transitions to the case III indicates the prediction success even its column is not finally activated, the higher number of the transitions, the better result. From the results, we can see that the methods B and C achieves a higher number of the transitions than the conventional CLA and the method A. These result reveals that the method B strengthing synapse connections with cells in the active candidate columns contributes to increasing the prediction successes.

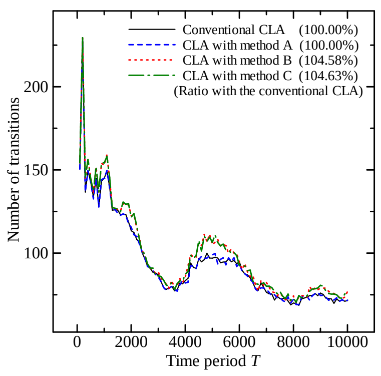
Thus, the method A works to reduce the prediction failures, the method B works to increase the prediction suc-



(a) Prediction errors

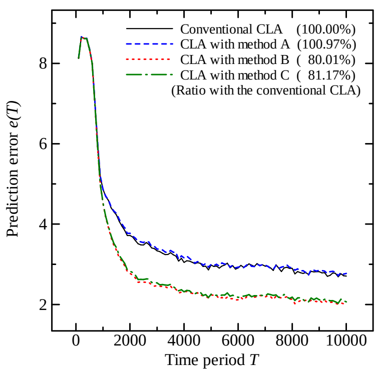


(b) Transitions to the case II

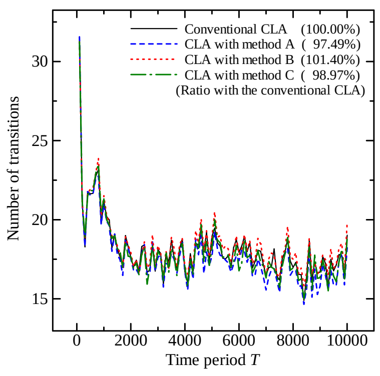


(c) Transitions to the case III

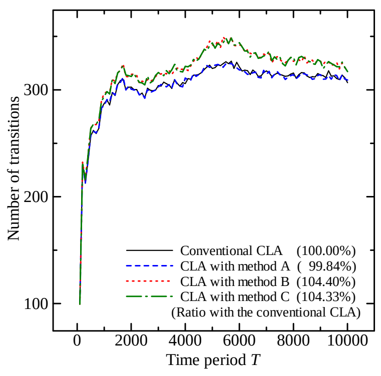
Figure 2: Result on $input_1(t)$ without the noise



(a) Prediction errors



(b) Transitions to the case II



(c) Transitions to the case III

Figure 3: Result on $input_2(t, \delta)$ with the noise $\delta = 0.01$

cesses, and the method C combining them achieves the lowest prediction error for the $input_1(t)$ without noise.

5.2. Prediction of Time-Series Data with Noise

Fig. 3 shows the results on $input_2(t)$ with noise. From the results in **Fig. 3 (a)**, we can see that the proposed method B significantly contributes to reducing the prediction errors. From the results in **Fig. 3 (b)**, the method A slightly reduce the number of the transitions to the case II and the prediction failures. However, the effect cannot be seen in the results of the prediction error on the time-series data prediction with noise. On the other hand, from the results in **Fig. 3 (c)**, we can see that the method B significantly improves the number of transitions to the case III and the prediction successes. This effect has a large impact on the reduction of the prediction errors shown in **Fig. 3 (a)**.

Thus, also in the time-series data with noise, the method A works to reduce the prediction failures, the method B works to increase the prediction successes. Also, the method B significantly contributes to reducing the prediction errors especially in the data prediction with noise.

6. Conclusions

To improve the prediction accuracy of CLA, in this work we proposed methods updating synapse permanence values of normal inactive cells transiting from the predictive state to the normal one. Experimental results showed that the proposed method A reduce the prediction failures by weakening the synapse connections of the predictive cells in the normal columns. Also, we showed that the proposed method B increase the prediction successes by strengthening the synapse connections of the predictive cells in active candidate columns which are not finally activated. For the time-series data without noise, the both methods contribute to improving the prediction accuracy. For the data with noise, the method B significantly improve the prediction accuracy of CLA.

As future work, we will address to the adaptive control of the performance increment and decrement values.

References

[1] J. Hawkins and S. Ahmad, “Why Neurons Have Thousands of Synapses, a Theory of Sequence Memory in Neocortex”, *Frontiers in Neural Circuits*, 10:23, 2016.

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

[3] A. Lavin and S. Ahmad, “Evaluating Real-time Anomaly Detection Algorithms – the Numenta Anomaly Benchmark,” Proc. in the 14th Int’l Conf. on Machine Learning and Applications (IEEE ICMLA’15), pp. 1–8, 2015.

[4] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory”, *Neural Computation*, Vol. 9 (8), pp. 1735–1780, 1997.