Solving the Density Classification Task Using Cellular Automaton 184 with Memory

Christopher Stone *
Larry Bull †
Department of Computer Science
University of the West of England
Bristol, BS16 1QY, United Kingdom
**christopher3.stone@uwe.ac.uk*
†*larry.bull@uwe.ac.uk*

A type of memory based on the least mean square algorithm is explored on the density classification task, which is a well-known test problem for two-state discrete dynamical systems. In the absence of memory, there is no elementary cellular automaton that can solve this task. However, when augmented with memory, the performance of elementary cellular automaton 184 approaches that of the best-known radius three cellular automata found in the literature. It is found that rule 184 transforms spatial information about the neighborhood into temporal information that memory is able to retain and present to the rule's transition function. This causes the cell to transition to a different state compared to the case when no memory is present, which extends blocks of cells having a common state to facilitate a solution of the task.

1. Introduction

Typically, the behavior of a complex system cannot be predicted by any means but numerical simulation. For this reason, computational models are extremely important in modern biology, sociology, advanced engineering, ecology, agriculture, and urban studies.

A number of computational models have been presented with which to study and/or exploit the aggregate behavior and self-organization of a number of simple interacting components. Among these models, cellular automata (CAs) have been widely used to model complex natural and artificial systems and are well-studied in their basic form.

Typically, this study does not include memory, that is, previous state information, at the component level. However, memory is an essential feature of all living systems and a significant part of physical, chemical, and engineering systems. We are undertaking systematic studies in the use and effects of memory in the components of discrete dynamical systems with the aim of identifying new underlying prin-

https://doi.org/10.25088/ComplexSystems.18.3.329

ciples of complex systems. An overview of CAs with memory is presented in [1].

In this paper we explore the effect of a simple type of memory on the density classification task, a well-known test problem for one-dimensional CAs. We find that with memory, performance on the task for an elementary CA approaches that of the best-known CAs found by other researchers. Without memory there is no elementary CA that can solve the task [2].

This paper is organized as follows. Section 2 provides an overview of CAs and the density classification task. Section 3 describes the memory scheme used and the results of experiments undertaken on the density classification task. Section 4 gives a detailed description of the operation of the CA with memory and Section 5 provides conclusions.

2. Background

2.1 Cellular Automata

Cellular automata are a class of discrete dynamical system consisting of a spatial lattice of N homogeneous automata (cells) that are updated synchronously in discrete time steps t. At each time step $t+1$ an automaton $i \in N$ takes one of k possible states by considering its own current state σ_i and that of other spatially local automata comprising the cell's neighborhood. The state update is performed according to some transition function Φ. At start time t^0 the states of the automata are set according to some externally imposed initial configuration (IC).

The CAs considered in this paper are one-dimensional $N = 149$ with periodic boundary conditions, that is, the lattice forms a ring. The automata neighborhood η consists of a cell i and all the cells located within a radius r of that cell:

$$\eta = \{\sigma_{i-r}, \ \ldots, \sigma_{i-1}, \sigma_i, \sigma_{i+1}, \ \ldots, \sigma_{i+r}\}. \tag{1}$$

The automata are binary ($k = 2$) with $\sigma \in \{0, 1\}$ and a neighborhood of $r = 1$. That is, the next state of a cell is determined by its current state and that of the two immediately adjacent cells to its left and right, $\eta = \{\sigma_{i-1}, \sigma_i, \sigma_{i+1}\}$. Such a CA is known as an elementary CA (ECA) [3]. ECAs have a transition function of 256 possible states, a number amenable to study. For this reason, ECAs have been the subject of much research effort in the last 20 or so years.

The transition function Φ takes the form of a rule, usually represented as a binary truth table, containing all possible neighborhood configurations and a corresponding new state σ^{t+1}.

https://doi.org/10.25088/ComplexSystems.18.3.329

2.2 The Density Classification Task

The density classification task has been studied for many years. In this task, a one-dimensional binary CA is initialized with a random IC and iterated for a maximum number of steps I or until a fixed point is reached. If the IC contains more ones than zeros, the CA is deemed to have solved the task if a fixed point of all ones is reached and vice versa. The fraction of ones in a CA configuration is denoted as ρ, so the problem solution ρ_f for an IC of ρ_0 is

$$\rho_f = \begin{cases} 0 & : \quad \rho_0 < 0.5 \\ 1 & : \quad \rho_0 > 0.5. \end{cases} \tag{2}$$

The situation where the IC contains an equal number of ones and zeros ($\rho = 0.5$) is normally avoided by using an odd number of cells.

This is a difficult task for a CA because a solution requires coordinating the global state of the system while using only local communication between cells provided by the neighborhood. For this reason, the density classification task is widely used as a standard test function to explore CA behavior.

The ability of a particular ECA to solve the density classification task depends on the IC. Intuitively, ICs containing many ones or zeros are closer in Hamming distance to one of the solution fixed points, making it easier for a CA to iterate to the correct fixed point compared to an IC containing a more or less equal mix of ones and zeros. For this reason, performance of a CA on the density classification task is estimated by sampling many ICs generated from a known distribution. Performance is then the fractional number of times the CA achieves the correct fixed point. It has been proven [4] that no binary CA exists that solves the density classification task for all possible ICs. Thus, a binary CA can only solve the problem for specific ICs or to a particular degree over multiple ICs.

Generating ICs using an equal probability of each cell being in the one or zero state creates a binomial distribution. There are more ways to create ICs with $\rho \approx 0.5$ than the extreme values of $\rho = 0$ and $\rho = 1$. Mean performance with ICs created using a binomial distribution is thus typically lower than that obtained when ICs are created using a distribution where values of ρ are sampled uniformly in the range [0, 1].

The stochastic variation occurring when sampling ICs is such that, depending on available computational resources, a rule is evaluated using a minimum of 100 random ICs. For the exhaustive search data presented in Section 3.2, we use 10^4 ICs for each datum. Other researchers interested in determining the best possible performance obtainable from a CA on the density classification task commonly use 10^5 or more trials each of a maximum of $I = 300$ iterations [5].

https://doi.org/10.25088/ComplexSystems.18.3.329

3. The Effects of Memory

3.1 Memory

A simple way of implementing memory is for the CA transition function to consider the neighborhood η of a cell i supplemented with the state σ_i^{t-1} of the cell on the previous cycle:

$$\sigma_i^{t+1} = \Phi\left(\sigma_j^t \in \eta_i, \sigma_i^{t-1}\right). \tag{3}$$

However, this means that the size of the transition function must increase to incorporate the extra state, doubling the size of the state space. Moreover, such a memory mechanism makes it difficult to compare directly the effects of adding memory to a CA, since the transition function is different with and without memory.

To overcome these limitations, memory must be included in a CA without affecting the transition function Φ. An approach to solving the density classification task with memory was made in [6, 7] by means of a memory implementation considering the majority of the last three state values ($\tau = 3$):

$$\begin{aligned} s_i^t &= \text{majority}\left(\sigma_i^{t-2}, \sigma_i^{t-1}, \sigma_i^t\right) \\ \sigma_i^{t+1} &= \Phi\left(s_j^t \in \eta_i\right). \end{aligned} \tag{4}$$

In the simulation of a series of 10^5 different initial densities uniformly sampled over [0, 1], 114 densities were misclassified under rule 184 with $N = 400$, $I = 1500$, and $\tau = 3$. All of the misclassifications occurred with initial densities close to the watershed $\rho_0 = 0.5$.

Here we implement a form of memory using the well-known least mean square (LMS) algorithm [8] with learning rate β. This provides an exponentially weighted moving average memory with no transition function overhead. For this type of memory:

$$\begin{aligned} m_i^0 &= 0.5 \\ m_i^{t+1} &= m_i^t + \beta\left(\sigma_i^t - m_i^t\right) \\ s_i^t &= \begin{cases} 0 & : \; m_i^{t+1} \le 0.5 \\ 1 & : \; m_i^{t+1} > 0.5 \end{cases} \\ \sigma_i^{t+1} &= \Phi\left(s_j^t \in \eta_i\right). \end{aligned} \tag{5}$$

The learning rate β controls the amount of memory with $\beta = 0$ providing infinite memory and $\beta \ge 0.5$ corresponding to no memory. With this memory arrangement, the transition function is the same as the case with no memory. However, the neighborhood presented to the transition function may be different between the two cases

https://doi.org/10.25088/ComplexSystems.18.3.329

because of the action of memory. This aspect will be extensively discussed in the following sections.

3.2 The Density Classification Task with Memory

The density classification task is usually undertaken using a binary CA with radius $r = 3$. This is because smaller neighborhoods do not support the behavior necessary to provide sufficient performance on the task [2]. In particular, with $r = 1$ performance measured over 10^4 trials of ICs does not exceed much more than 0.5 for any of the 256 possible rules (Figure 1). In these runs each trial starts from a random IC created using an unbiased (binomial) distribution. The CA is then iterated for up to 300 configurations or until a fixed point is reached.

In contrast, CAs with radius $r = 3$ routinely achieve performances of around 0.65 or above [2, 9]. At the time of writing, the best $r = 3$ CA found so far using machine learning techniques achieves a performance of 0.8616 averaged over 10^5 trials, each of 300 iterations [5].

A CA may be supplemented with the exponentially weighted moving average memory described in Section 3 without changing the rule used by the CA to transition to the next state. Figure 2 shows the performance of the 256 ECA rules with memory. The learning rate β of the memory was sampled in the range $[0, 0.5]$ in steps of 0.01. Values of β greater than 0.5 provided performance equal to that of the CA with no memory and for greater clarity are not shown. Due to the total number of trials involved in producing this map, the performance of each combination of CA rule and learning rate was sampled over only 10^4 trials. Even so, Figure 2 shows clearly that two rules are able to provide reasonable performance on the density classification task over a range of learning rates. For reference, these rules are 184 and 226 using the standard ECA nomenclature [3] and they achieve performance of up to approximately 0.65 with $\beta \in [0.45, 0.49]$.

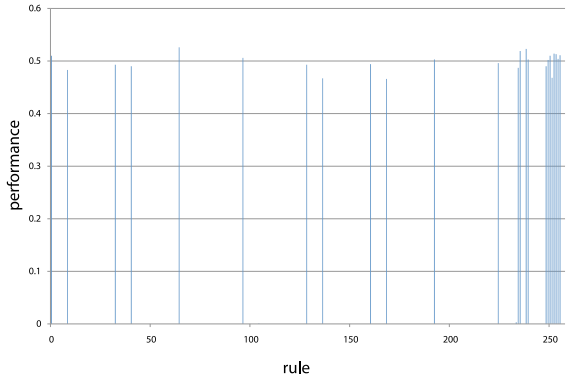

Figure 1. Performance of the 256 ECA rules on the density classification task averaged over 10^4 trials of ICs created using an unbiased distribution. $N = 149$, $I = 300$.

https://doi.org/10.25088/ComplexSystems.18.3.329

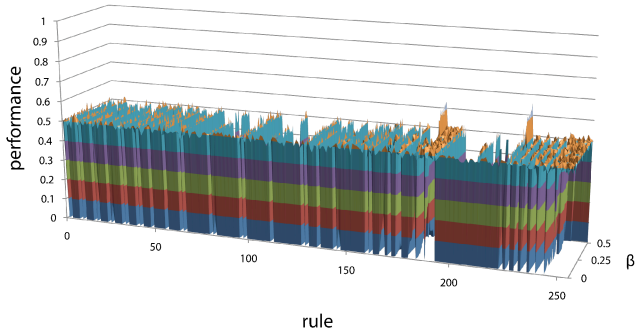

Figure 2. Performance of the 256 ECA rules with memory on the density classification task averaged over 10^4 trials of ICs created using an unbiased distribution. $N = 149$, $I = 300$.

Although the performance of ECAs 184 and 226 with memory is respectable, it is not competitive with that obtained from the $r = 3$ CA results appearing in the literature. One of the differences between $r = 1$ and $r = 3$ CAs is that the latter support faster communication across cells. This is because it is possible for a signal in the form of a spatial pattern to move three cells per CA iteration in the case of an $r = 3$ CA compared to one cell for $r = 1$. As this speed is slower for the ECAs, it is to be expected that they might take longer to reach a fixed point than would be the case for the $r = 3$ CAs and therefore may be handicapped by the limit of $I = 300$ iterations per trial. To determine if this was the case, we produced a further performance map of the ECAs where the maximum number of iterations allowed was 900, this being three times the previous limit. Results are shown in Figure 3.

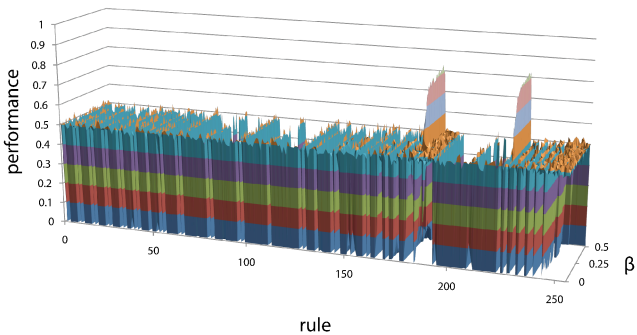

Figure 3. Performance of the 256 ECA rules with memory on the density classification task averaged over 10^4 trials of ICs created using an unbiased distribution. $N = 149$, $I = 900$.

https://doi.org/10.25088/ComplexSystems.18.3.329

When allowed more iterations to reach a fixed point, rules 184 and 226 were able to provide performance exceeding 0.8 on the density classification task. Moreover, the shape of the performance curves seen in Figure 2 as the learning rate varies is now much flatter, suggesting that the task is solvable over a wider range of possible learning rates. Figure 4 shows that, on average, it takes longer to reach a fixed point with lower numerical values of β and that even with higher values of β, the mean number of iterations to reach a fixed point is roughly 220, which is close to the standard maximum of 300 iterations. This is the reason for the poorer performance when the maximum number of iterations is limited to 300 and with low learning rates.

When allowed a maximum of $I = 900$ iterations and using $\beta = 0.48$, rules 184 and 226 with memory provide mean performance of 0.82 averaged over 10^5 trials. This is competitive with that obtained from $r = 3$ rules over 300 iterations. The learning rate of $\beta = 0.48$ is used throughout this paper as it is a typical value resulting in good performance on the density classification task.

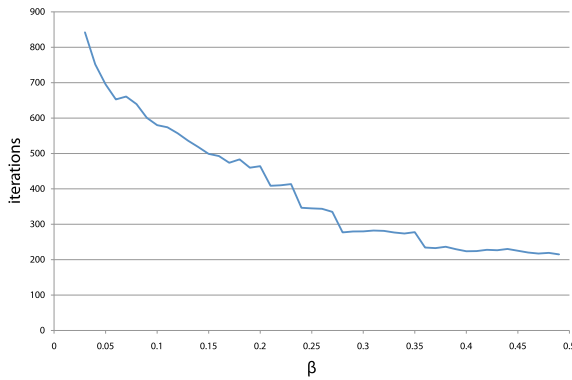

Figure 4. Mean number of iterations to reach a fixed point against learning rate. Rules 184 and 226 with memory on the density classification task averaged over 10^4 trials of ICs created using an unbiased distribution. $N = 149$, $I = 900$.

3.3 Memory and the Neighborhood

Rules 184 and 226 belong to the same equivalence class. If the neighborhood is reflected around the center cell such that the orientation of cells $i - 1$ and $i + 1$ are reversed, the two rules are identical. Alternatively, one rule may be transformed into the other by inverting the states in the transition rule so that a 0 becomes a 1 and vice versa. These ECAs may thus be viewed as left- and right-handed versions of the same transition rule or as a rule operating on complementary state definitions. To focus the discussion, we will generally ignore the exis-

https://doi.org/10.25088/ComplexSystems.18.3.329

tence of rule 226 as a solution and concentrate only on rule 184. However, all results and conclusions for rule 184 apply equally to rule 226.

The performance described in Section 3.2 was obtained with a transition function Φ operating on the memory function s_j^t of all cells $j \in \eta_i$ in the neighborhood, including the center cell i that is being updated. To determine whether this architecture was strictly necessary for successful operation on this task, we tested variations of this scheme whereby: (*i*) the transition function used only the memory s_i^t of the center cell i and used the states σ_{i-1}^t, σ_{i+1}^t of the left and right neighbors as is the case for a CA without memory, and (*ii*) the opposite scheme where the transition function considered only the memory of the left and right neighbors and not that of the center cell. These memory schemes are detailed in equations (6) and (7), respectively:

$$\sigma_i^{t+1} = \Phi\left(\sigma_{i-1}^t, s_i^t, \sigma_{i+1}^t\right) \tag{6}$$

$$\sigma_i^{t+1} = \Phi\left(s_{i-1}^t, \sigma_i^t, s_{i+1}^t\right). \tag{7}$$

To minimize variation due to the effects explained in Section 3.2 caused by the finite number of iterations allowed, all trials were run with a maximum of $I = 1500$ iterations. Results for the memory scheme in equation (6) were no better than those obtained without memory for the same number of iterations, suggesting that memory was needed for one or more of the left and right neighbors. In contrast, results (not shown) obtained for the memory scheme in equation (7) matched those obtained from the original memory scheme described in equation (5). These results suggest that it is the memory in the left and right neighbors that supports performance and that use of memory in the center cell is not necessary.

As rules 184 and 226 are handed, it is possible that, for a given rule, memory is needed in only the left or right neighbor and not both. To assess this possibility, we implemented memory according to equation (8) whereby only the memory state of the left neighbor was considered (results are shown in Figure 5):

$$\sigma_i^{t+1} = \Phi\left(s_{i-1}^t, \sigma_i^t, \sigma_{i+1}^t\right). \tag{8}$$

Perhaps surprisingly, when memory is only used with the left neighbor, neither rule 184 nor rule 226 operates successfully on the density classification task. It would thus appear that memory is needed on both left and right neighbors for rules 184 and 226, despite the handedness of these rules.

https://doi.org/10.25088/ComplexSystems.18.3.329

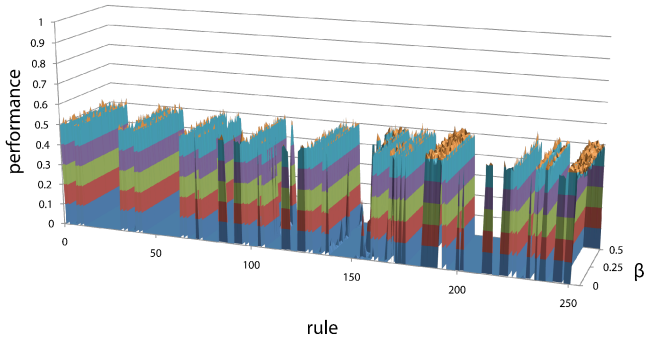

Figure 5. Performance of the 256 ECA rules with asymmetric memory (equation (8)) on the density classification task averaged over 10^4 trials of ICs created using an unbiased distribution. $N = 149$, $I = 1500$.

4. The Operation of Memory

4.1 Rule 184

The truth table for rule 184 is shown in Table 1. The operation of the rule is easily summarized as: "if the center cell is at state zero, shift the state of the left neighbor into the center cell, else shift the state of the right neighbor into the center cell" (equation (9)). This operation implements a switch or multiplexer, but can also be interpreted as being annihilating particles traveling in opposite directions [10-12]:

$$\sigma_i^{t+1} = \begin{cases} \sigma_{i-1}^t & : \quad \sigma_i^t = 0 \\ \sigma_{i+1}^t & : \quad \sigma_i^t = 1. \end{cases} \tag{9}$$

σ_{i-1}^t	σ_i^t	σ_{i+1}^t	σ_i^{t+1}
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Table 1. Operation of rule 184. The first three columns are the neighborhood and the rightmost column is the state of the center cell that results from applying the transition function on the neighborhood.

https://doi.org/10.25088/ComplexSystems.18.3.329

Rule 184 is conservative, that is, the number of one states in the lattice is invariant across CA configurations. The rule can only alter the distribution of density across the lattice, but it cannot alter the overall density of the lattice. Rule 184 therefore cannot solve the density classification task as it is formulated in Section 2.2. However, it can be used to solve a modified version of the problem where the final configuration consists of one or more blocks of consecutive ones or zeros depending on the initial density [13]. Other authors have shown that it is possible to use rule 184 followed by a different rule to solve the density classification task in its standard formulation [12, 14, 15].

Figure 6(a) demonstrates the operation of rule 184 on a sample lattice. In this example, the IC consists of five black cells (state one) and six white cells (state zero). Much of the final configuration in Figure 6(a) is made up of alternating black and white cells that is the periodic background state characteristic of rule 184. However, note the presence of two consecutive white cells representing the solution to the modified version of the task. The final configuration for a solution to the formulated version of the density classification task for this IC should be a lattice consisting of only white cells. It is clear that this has not been achieved, even for this simple example.

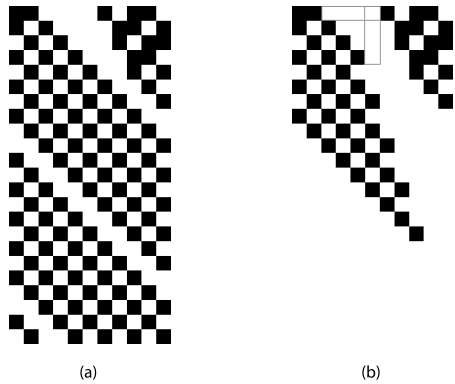

Figure 6. (a) Rule 184 without memory fails to solve the density classification task as formulated. (b) Rule 184 with memory solves the density classification task using the same IC, the transformation of spatial information into temporal information that is used to solve the task is highlighted with dotted rectangles. $\beta = 0.48$.

4.2 Memory State

The memory system used here stores its condition as a real value, which is thresholded and discretized to generate the binary state used as input to the transition function. The results in Section 2 show that supplementing rule 184 with such memory allows it to solve the density classification task. To gain some intuition about how this is

https://doi.org/10.25088/ComplexSystems.18.3.329

achieved we consider the internal condition of the memory m_i^t (equation (5)) as representing its long-term value and see how the application of specific temporal sequences of cell states σ_i^{t+n} affects the memory state s_i^{t+n} used for the transition function n iterations later.

Table 2 shows the truth table for the memory state s_i^{t+4} for all possible combinations of four successive cell states σ_i^{t+n}, $n \in [0, 3]$ applied to a range of long-term memory conditions m_i^t with $\beta = 0.48$.

Cell States				Initial Internal Memory Condition m_i^t										
σ_i^0	σ_i^1	σ_i^2	σ_i^3	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	1	1	1	1	1	1	1	1	1	1	1
0	1	1	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	1	1	1	1	1	1	1	1	1	1	1
1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	0	1	1	1	1	1	1	1	1	1	1	1	1
1	**1**	**1**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 2. Truth table showing the memory state s_i^{t+4} resulting from four successive cell states σ_i^{t+n}, $n \in [0, 3]$ with various initial internal memory conditions m_i^t and $\beta = 0.48$. The resulting memory state varies only in the presence of temporal cell states 0001 and 1110 (bold numbers).

The table shows that memory state s_i^{t+4} resulting from a sequence of four successive cell states does not depend on the long-term memory condition m_i^t except in the case of temporal patterns 0001 and 1110. Except for these two cases with certain values of m_i^t, the memory state s_i^{t+4} is identical to σ_i^3 and memory has no effect. Similar re-

https://doi.org/10.25088/ComplexSystems.18.3.329

sults were obtained for other temporal sequence lengths (not shown). The success of rule 184 with memory therefore must be attributable to the changed behavior seen with temporal sequences of the form $0^* 1$ and $1^* 0$, that is, those where a one state occurs after a series of zero states or vice versa. Once again, we will ignore symmetries and consider only input sequences of the form $0^* 1$, noting that all results apply equally to the temporal pattern $1^* 0$.

It is apparent from Table 2 that the long-term condition of memory affects whether an input pattern of the form $0^* 1$ is able to affect the resulting memory state compared to the case with no memory. But just how many repeated zeros are necessary for this to happen?

For a given long-term memory condition m_i^t, the memory condition m_i^{t+n} resulting from a sequence $\sigma_i^t \ldots \sigma_i^{t+n-1}$ of n zeros is given by

$$m_i^{t+n} = (1-\beta)^n \, m_i^t. \tag{10}$$

If state $\sigma_i^{t+n} = 1$ is appended to the sequence

$$m_i^{t+n+1} = m_i^{t+n} + \beta\left(1 - m_i^{t+n}\right). \tag{11}$$

Without memory, a temporal sequence of the form $0^* 1$ will always result in state $\sigma_i^{t+n} = 1$. For memory to create the opposite state $\sigma_i^{t+n} = 0$, m_i^{t+n+1} must threshold to zero. So,

$$\begin{aligned} m_i^{t+n} + \beta\left(1 - m_i^{t+n}\right) &\leq 0.5 \\ m_i^{t+n} &\leq 1 - \frac{1}{2\,(1-\beta)} \\ n &\leq \log_{1-\beta}\left(\frac{1-\frac{1}{2\,(1-\beta)}}{m_i^t}\right). \end{aligned} \tag{12}$$

For $\beta = 0.48$ there must be a sequence of up to five consecutive zeros ($n \leq 4.98$) for memory to threshold to the opposite state to that occurring with no memory. Figure 7 shows the surface resulting from equation (12) for various values of long-term memory condition m^t and learning rate β. Given that a low value of m_i^t means that the cell already has a history of zeros, by cross referencing Figure 3 with Figure 7 it is evident that typically three or four zeros are needed to allow memory to create the opposite state compared to that obtained with no memory. This action enables subsequent solution of the density classification task.

https://doi.org/10.25088/ComplexSystems.18.3.329

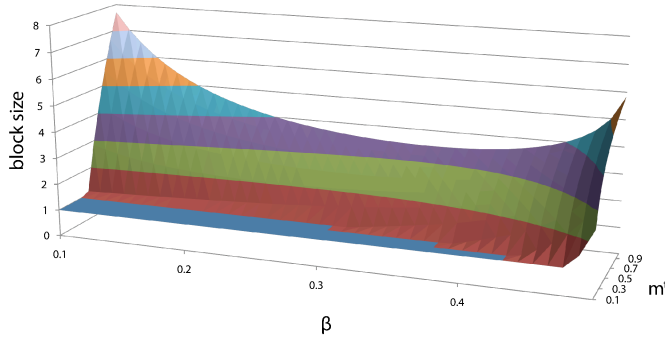

Figure 7. Number of consecutive zeros (block size) required to modify the state of a cell's neighborhood for various values of long-term memory condition m^t and learning rate β.

4.3 Interaction with Rule 184

Section 4.2 showed that the succession of consecutive zeros that is input into memory is critical to successful performance. These zeros are a result of the shifting action of rule 184. Recall that with a center cell of state zero, rule 184 shifts the state of the left neighbor into the center cell. This operation is repeated for successive consecutive zeros over multiple CA iterations, transforming spatial information into temporal information. Similarly, with a center cell of state one, the state of the right neighbor is taken. This symmetry explains the finding in Section 3 that memory is needed on both left and right neighbors for successful operation.

Figure 6(b) shows an example of rule 184 with memory solving the density classification task and highlights this spatiotemporal transformation. In this case, a block of four consecutive white cells occurring in the IC is transformed by the rule into a temporal sequence of four zeros across successive CA configurations. At the time of the IC, $m_i^t = m_i^0 = 0.5$. Under these internal memory conditions $n \leq 3.92$ (equation (12)) for $\beta = 0.48$, so four zeros are all that is necessary to allow this to occur. This is seen in Figure 6(b) as the highlighted block of four white cells. As already discussed, Figure 6(a) shows the same example without memory, where the system dynamics are controlled by a periodic attractor resulting in the alternating background pattern characteristic of rule 184.

The neighboring cells to the immediate left and right of the center cell can be considered to be sensors detecting the environment. The multiplexing action of rule 184 samples this environment and takes in a copy of the state of the environment to the cell's left or right, depending on the value of the center cell. The center cell's memory is able to retain aspects of the information passing through the center cell. Due to the action of rule 184 transforming spatial information

https://doi.org/10.25088/ComplexSystems.18.3.329

into temporal information, the temporal history of the center cell provides information about the spatial state of the environment outside the immediate $r = 1$ sensory area of the cell's neighborhood. As demonstrated in Section 4, for $\beta = 0.48$ memory provides information up to five time steps back and hence up to five cells away from the center cell, albeit that this information is delayed in time. In this way, memory extends the sensory capabilities of the CA and provides information that is otherwise inaccessible to a standard $r = 1$ CA.

In the example in Figure 6(b), a block of four zero-value (white) cells provides the seed for memory to create a cell state that enables white cells to take over the lattice. Modification of cell state by the action of memory has already been discussed in Section 4.2, but there is a further effect that allows the density classification task to be solved. Without memory, a spatial pattern of the form $0^* 1$ results in the center cell having state one after the pattern has been consumed and the center cell will transition to the state of the right neighbor. In other words, it will sense the environment to its right. However, with memory, if the above conditions are satisfied, the center cell will have state zero at the next iteration. This in turn affects the subsequent evolution of the lattice because this state means that at the next iteration the center cell will transition to the state of the left neighbor once again. This provides a mechanism for growing contiguous blocks of cells with state zero. Once one of these blocks starts growing, the memory conditions m_i^t rapidly become polarized in cells where the block is located and it becomes increasingly difficult for cells with state one to disrupt the growth of the block.

5. Conclusion

In its standard formulation the density classification task cannot be solved by an $r = 1$ CA. However, we have shown that by augmenting a well-known ECA, rule 184 (or rule 226), with a simple form of memory, performance approaching that of the best-known $r = 3$ CAs is possible. This performance is achieved using only local communication, that is, with the neighbors immediately adjacent to a cell. Such communication topology is important for parallel computing architectures embedded in traditional silicon hardware devices or for future nanoscale devices.

Investigation of rule 184 with the chosen memory scheme revealed three key differences compared to operation of the CA without memory. (*i*) Rule 184 allows spatial information encoded in the lattice to be transformed into temporal information. In the standard CA this offers no benefit, but memory is able to retain aspects of this information that are useful in solving the problem. (*ii*) The memory scheme investigated can detect the existence of a block of cells having

https://doi.org/10.25088/ComplexSystems.18.3.329

a common state. When this occurs, information is signaled to the transition function as a changed state. (*iii*) Thus, the cell is able to transition to a different state compared to the case with no memory and increase the size of the block of cells having a common state.

Although the discussion has focused on blocks of zeros, symmetries in both the rule and memory mean that blocks of ones are similarly affected. Globally, this allows the ECA to break its usual conservativeness and allow lattice density to change, a requirement for solving this task.

Other aspects of the operation of the memory scheme are also interesting. The memory scheme is quite robust on this task and successful operation is possible over a wide range of learning rates. It is very specific in operation and presents the same neighborhood to a cell as for the case with no memory, except under the specific circumstances when a block is detected. Furthermore, the transition function and memory are symbiotic, each providing mechanisms and information that is used by the other to solve the problem. These aspects warrant further study to see if they are general phenomena exhibited by other successful natural and artificial systems with memory.

Following the success of solving the density classification task with rule 184 and memory, we have extended this work to see how memory affects other emergent systems [16].

Acknowledgments

This work was supported under EPSRC grant number EP/E049281/1.

References

[1] R. Alonso-Sanz, "Cellular Automata with Memory," *Encyclopedia of Complexity and System Science* (R. A. Meyers, ed.), New York: Springer-Verlag, 2009.

[2] J. P. Crutchfield, M. Mitchell, and R. Das, "The Evolutionary Design of Collective Computation in Cellular Automata," *Evolutionary Dynamics—Exploring the Interplay of Selection, Neutrality, Accident, and Function* (J. P. Crutchfield and P. K. Schuster, eds.), New York: Oxford University Press, 2003 pp. 361-411.

[3] S. Wolfram, *A New Kind of Science*, Champaign, IL: Wolfram Media, Inc., 2002.

[4] M. Land and R. K. Belew, "No Perfect Two-State Cellular Automata for Density Classification Exists," *Physical Review Letters*, **74**(25), 1995 pp. 5148-5150.

[5] P. P. B. de Oliveira, J. C. Bortot, and G. M. B. Oliveira, "The Best Known Class of Dynamically Equivalent Cellular Automata Rules for Density Classification," *Neurocomputing*, **70**(1-3), 2006 pp. 35-43. doi.10.1016/j.neucom.2006.07.003.

https://doi.org/10.25088/ComplexSystems.18.3.329

[6] R. Alonso-Sanz and L. Bull, "Elementary Coupled Cellular Automata with Memory," *Automata 2008: Theory and Applications of Cellular Automata* (A. Adamatzky, R. Alonso-Sanz, A. Lawniczak, G. J. Martinez, K. Morita, and T. Worsch, eds.), Frome, UK: Luniver Press, 2008 pp. 72-99.

[7] R. Alonso-Sanz and L. Bull, "One-Dimensional Coupled Cellular Automata with Memory: Initial Investigations," *Journal of Cellular Automata*, in press.

[8] B. Widrow and M. E. Hoff, "Adaptive Switching Circuits," *IRE WESCON Convention Record*, **4**, 1960 pp. 96-104.

[9] H. Juillé and J. B. Pollack, "Coevolving the 'Ideal' Trainer: Application to the Discovery of Cellular Automata Rules," in *Genetic Programming 1998, Proceedings of the Third Annual Programming Conference (GP'98), Madison, WI* (J. R. Koza, W. Banzhaf, K. Chellapilla, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. L. Riolo, eds.), San Francisco: Morgan Kaufmann, 1998 pp. 519-527.

[10] V. Belitsky and P. A. Ferrari, "Ballistic Annihilation and Deterministic Surface Growth," *Journal of Statistical Physics*, **80**(3-4), 1995 pp. 517-543. doi.10.107/BF02178546.

[11] V. Belitsky and P. A. Ferrari, "Invariant Measures and Convergence Properties for Cellular Automaton 184 and Related Processes," *Journal of Statistical Physics*, **118**(3-4), 2005 pp. 589-623.

[12] H. Fukś, "Solution of the Density Classification Problem with Two Cellular Automata Rules," *Physical Review E*, **55**(3), 1997 pp. 2081-2084.

[13] M. S. Capcarrère, M. Sipper, and M. Tomassini, "Two-State, $r = 1$ Cellular Automaton that Classifies Density," *Physical Review Letters*, 77(24), 1996 pp. 4969-4971.

[14] H. Kanoh and Y. Wu, "Evolutionary Design of Rule Changing Cellular Automata," in *Knowledge-Based Intelligent Information and Engineering Systems, Proceedings of the Seventh International Conference (Part 1) (KES'03), Oxford, UK, Lecture Notes in Computer Science*, **2773**, Berlin: Springer, 2003 pp. 258-264. doi.10.1007/b12002.

[15] C. L. M. Martins and P. P. B de Oliveira, "Evolving Sequential Combinations of Elementary Cellular Automata Rules," in *Advances in Artificial Life, Proceedings of the Eighth European Conference (ECAL'05), Canterbury, UK* (M. S. Capcarrère, A. A. Freitas, P. J. Bentley, C. G. Johnson, and J. Timmis, eds.), *Lecture Notes in Artificial Intelligence*, **3630**, Berlin: Springer, 2005 pp. 461-470. doi.10.1007/11553090_ 47.

[16] C. Stone, R. Toth, B. De Lacy Costello, L. Bull, and A. Adamatzky, "Coevolving Cellular Automata with Memory for Chemical Computing: Boolean Logic Gates in the BZ Reaction," in *Proceedings of the Tenth International Conference on Parallel Problem Solving from Nature (PPSN X), Dortmund, Germany* (G. Rudolph, T. Jansen, S. Lucas, C. Poloni, and N. Beume, eds.), *Lecture Notes in Computer Science*, **5199**, Berlin: Springer, 2008 pp. 579-588. doi.10.1007/978-3-540-87700-4_ 58.

https://doi.org/10.25088/ComplexSystems.18.3.329