Learning Graph Cellular Automata

Daniele Grattarola, Lorenzo Livi, Cesare Alippi
Neural Information Processing Systems 2021
Cellular automata

(a) Transition rule

(b) Evolution of CA
Cellular automata

(a) Rule 73
(b) Rule 90
(c) Rule 110
(d) Rule 184
Cellular automata

(a) Rule 73  
(b) Rule 90  
(c) Rule 110  
(d) Rule 184  
(e) Conway’s Game of life
A Graph CA is the generalisation of typical CA:

- Cells arranged in a graph;
- State space is any vector space;
- Transition rule $\tau$ as a function of neighbours $\mathcal{N}(i)$;
The transition rule has the form:

$$\tau(s_i) : \{s_i\} \cup \{s_j, e_{ji} \mid j \in \mathcal{N}(i)\} \rightarrow s'_i,$$

where $e_{ji}$ encodes type, distance, direction, or unique ID of neighbour.
Problem: how to design a useful rule?
• Wulff and Hertz [1]: learning 1D and 2D CA using “∑-∏ networks with weight sharing”;

Learning GCA

- Wulff and Hertz [1]: learning 1D and 2D CA using “∑-Π networks with weight sharing”;
- Elmenreich and Fehérvári [2], Nichele, Ose, Risi, et al. [3]: neuroevolution to learn rules with target behaviour;

Learning GCA

- **Wulff and Hertz [1]**: learning 1D and 2D CA using “$\sum - \prod$ networks with weight sharing”;
- **Elmenreich and Fehérvári [2], Nichele, Ose, Risi, et al. [3]**: neuroevolution to learn rules with target behaviour;
- **Gilpin [4]**: universal CNN architecture for $M$-state 2D CA;

Learning GCA

- **Wulff and Hertz** [1]: learning 1D and 2D CA using “$\sum - \prod$ networks with weight sharing”;
- **Elmenreich and Fehérvári** [2], **Nichele, Ose, Risi, et al.** [3]: neuroevolution to learn rules with target behaviour;
- **Gilpin** [4]: universal CNN architecture for $M$-state 2D CA;
- **Mordvintsev, Randazzo, Niklasson, et al.** [5]: learning to *grow* a given configuration (inspired by flatworms).

---

GCA transition rules are message-passing functions:

$$s'_i = \gamma \left( s_i, \sum_{j \in \mathcal{N}(i)} \phi(s_i, s_j, e_{ji}) \right).$$

**Graph Neural Cellular Automata**: GCA with GNN transition rule.
Extend the results of Gilpin [4] to implement any $M$-state GCA:

- MLP for one-hot encoding states;
- Message-passing for pattern matching;

Binary GCA on Voronoi tessellation (equiv. Delaunay triangulation).

Simplest extension to GCA.
Outer-totalistic rule only depends on the density $\rho_i$ of alive neighbours:

$$
\tau(s_i) = \begin{cases} 
    s_i, & \text{if } \rho_i \leq \kappa \\
    1-s_i, & \text{if } \rho_i > \kappa.
\end{cases}
$$

(a) Example transition
GNCA on Voronoi tessellation

(a) Loss.

(b) Accuracy.

(c) Entropy.
Continuous-state GCA with dynamic graph based on the Boids algorithm [6]:

(a) Separation  
(b) Alignment  
(c) Cohesion

Loss goes to $10^{-6}$ almost immediately, but approximation is not good:

Use sample entropy and correlation dimension to evaluate how good the learned rule is.

(a) Loss.

(a) SampEn and CD during training.
(a) Examples of flocks from the true system and the GNCA.
GNCA that converge to a fixed target

**Goal:** design a rule with a desired behaviour (converging to target).
GNCA that converge to a fixed target

Apply rule for $t$ steps starting from $S$. 

$t \in [10, 20]$
GNCA that converge to a fixed target

Compute loss w.r.t. target state $\hat{S}$. 

$t \in [10, 20]$
GNCA that converge to a fixed target

\[ \tau^t(S) \to \hat{S} \]

Use a cache to ensure stable attractor and adequate state space exploration [5].

GNCA that converge to a fixed target

(a) Logo, GNCA trained with $t = 20$, example of convergence.

(a) Logo, GNCA trained with $t = 20$, example of convergence.
GNCA that converge to a fixed target

(a) Grid, GNCA trained with $t = 10$, example of periodic behaviour.

(a) Bunny, GNCA trained with $t = 20$, example of periodic behaviour.
Future research

- Predict global properties (e.g. graph classification);
  - Something similar was done by Gori, Monfardini, and Scarselli [7];

Future research

- Predict global properties (e.g. graph classification);
  - Something similar was done by Gori, Monfardini, and Scarselli [7];
- Decentralised control (e.g. IoT, autonomous vehicles);

Future research

- Predict global properties (e.g. graph classification);
  - Something similar was done by Gori, Monfardini, and Scarselli [7];
- Decentralised control (e.g. IoT, autonomous vehicles);
- Epidemiological networks;

Future research

• Predict global properties (e.g. graph classification);
  • Something similar was done by Gori, Monfardini, and Scarselli [7];
• Decentralised control (e.g. IoT, autonomous vehicles);
• Epidemiological networks;
• Modelling biological systems (e.g. neurons).

Learning Graph Cellular Automata
Daniele Grattarola, Lorenzo Livi, Cesare Alippi

github.com/danielegrattarola/GNCA

Summary:
- Learn GCA rules with GNNs;
- Universal architecture for $M$-state GCA;
- Enable design of GCA from high-level specification;
- Solve tasks through emergent computation.

Get in touch: daniele.grattarola@usi.ch


