Cellular Automata Control with Deep Reinforcement Learning

Emanuel Becerra Soto

2020-06-19

Contents

Chapter 1

CARL

1.1 Abstract

1.2 Acknowledgments

Chapter 2

Introduction

2.1 Motivations

Since antiquity the idea of building a thinking machine has always been in the minds of philosophers, artists, sciencemen, kings and commoners alike, filling us with wonder, terror and contemplation. A human creation capable of human feats, would turn us, at least in an allegorical sense, into gods.

2.2 Problem Statement

2.3 Objectives

2.3.1 Main Objectives

- Propose a novel environment for Reinforcement Learning algorithms, based on Cellular Automata, that could be used as an alternative benchmark instead of Atari games.
- Characterize the environment by solving it by state of the art methods.

2.3.2 Specific Objectives

- Select a Cellular Automaton model for the environment, in this case the forest fire cellular automaton.
- Design a RL task incorporating the CA dynamics.
- Implementation of the RL environment, following the Open AI gym API.
- Apply DQN to solve the proposed task.

Chapter 3

Cellular Automata

3.1 Introduction

Cellular Automata (sg. Cellular Automaton) are computational and mathematical systems with two essential characteristics: a discrete structure and local interaction between its parts, but its killer feature is that they are simple.

A quick *Google Scholar* search for the term "Cellular Automata", retrieves roughly 13,000 research articles since the last year (2019). Despite not being as popular (by the same metric) as other topics, like "Cancer" (~128,000) or "Deep Learning" (~104,000), they are, arguably, still popular.

The topic of CA dates back to the 1950's spanning a history of 70 years. Since then the mathematical and practical properties of CA have been studied and its applications have been explored in different branches of science including physical and social.

Perhaps the reason behind this popularity is their simplicity. CA are composed of decentralized interactions of their individual parts (cells), these cells are usually arranged in a very regular grid and are updated following the same rules for all cells. The rules only take into account the vicinity of a given cell. From this design specifications one could naively anticipate that a very homogeneous state is reached after some iterations of the CA. But this is not always the case, surprisingly for some simple configurations and rules, complex behavior emerges. This behavior is rich enough to be used to model natural systems. The structures that arises from local interactions where not designed *a priori* and their nature is capricious as they could be oscillating, chaotic, ordered, random, transient, stable. Their scale is also far from the initial size of the cell neighborhoods.

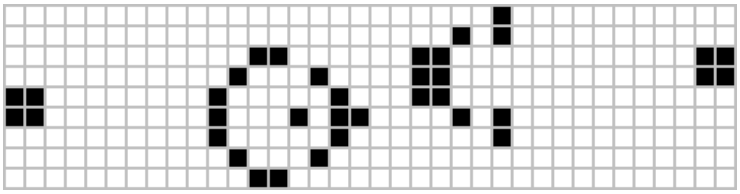

Figure 3.1: A glider gun, a famous pattern, from Conway's Cellular Automaton "Game of Life". Image taken from: *https://en.wikipedia.org/wiki/File:Game_of_life_glider_gun.svg*.

3.2 Main Characteristics

Cellular Automata are mathematical objects that are mainly characterized by (Ilachinski 2001):

1. A discrete lattice of cells: A n-dimensional arrangement of cells, usually 1-D, 2-D or 3-D.
2. Homogeneity: Cells are equivalent in the sense that they share an update function and a set of possible states.
3. Discrete states: Each cell is in one state from a finite set of possible states.
4. Local Interactions: Cell interactions are local, this is given by the update function being dependent on neighboring cells.
5. Discrete Dynamics: The system evolves in discrete time steps. At each step the update function is applied simultaneously (synchronously) to all cells.

3.3 Mathematical Definition

The following is adapted from the book Probabilistic Cellular Automata (Louis and Nardi 2018).

Cellular Automata (CA) are dynamical systems of interconnected finite-state automata (cells). The automata evolution is through discrete time steps and it is dictated by a function dependent on a neighborhood of interacting automata.

The main mathematical aspects of a CA are:

- The network G: A graph G.

  $$G = (V(G), E(G))$$

  The set of vertices $V(G)$ represents the location of the automata (cells). The set of edges $E(G)$ describes the spatial relations between automata.

- The alphabet S: Defines the states that each automata can take. In common CA settings S is a finite set. It is also called *local space* or *spin space*.
- The configuration space $S^{V(G)}$: This is the set of all possible states of the CA. A specific configuration is denoted as:

$$\sigma = \{\sigma_k \in V(G)\}$$

  σ_k is the configuration of the automaton at position k.
- The neighborhoods V_k:

$$V_k \subset V(G)$$

  The subset of nodes that can influence or interact with the automaton at $k \in V(G)$ (ordinarily it includes itself). A typical configuration for V_k is: $G = \mathbb{Z}^2$, and $V_k = \{k, k \pm e_1, k \pm e_2\}$, where (e_1, e_2) is the canonical basis of $\mathbb{Z}^2$, (north/south, east/west). This is known as the *von Neumman neighborhood*.
- The global update F:

$$F : S^{V(G)} \to S^{V(G)}$$

$$(F(\sigma))_k = f_k(\sigma_{V_k})$$

  The global update F is calculated by applying a local function f_k per automata at k. In the classical setting f_k is the same for all the automata.

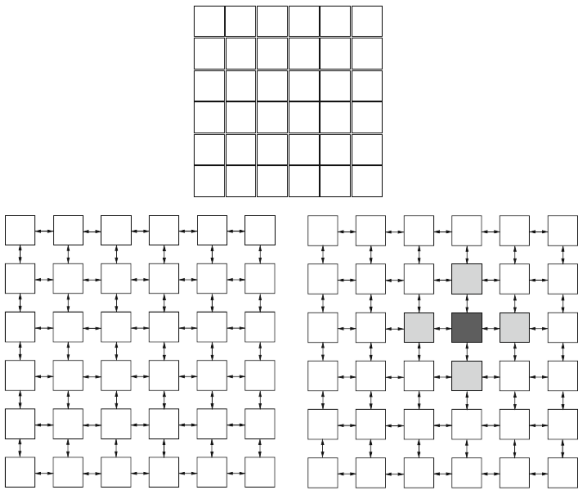

Figure 3.2: The underlying graph structure of a Cellular Automaton. A highlighted cell and its neighbors. Image adapted from (Hoekstra, Kroc, and Sloot 2010).

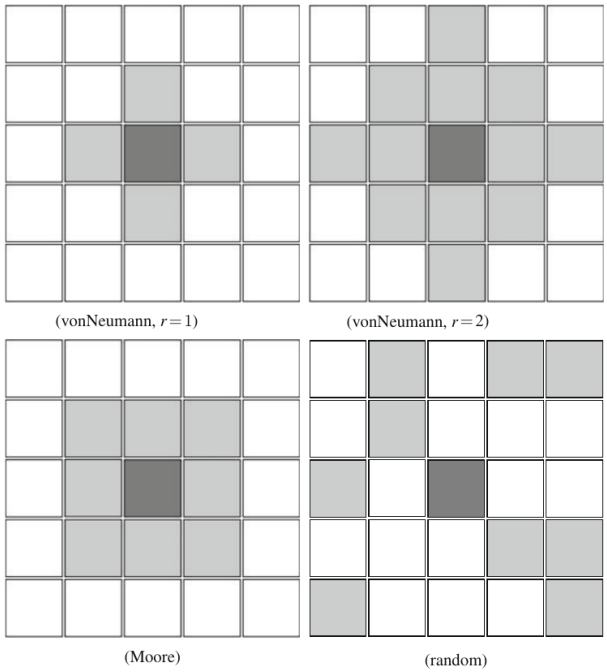

Figure 3.3: Different types of neighborhoods. Image adapted from (Hoekstra, Kroc, and Sloot 2010).

3.4 Classification

Wolfram from extensive simulations of 1-D CA grouped the general behavior of CA in four informally defined classes (Wolfram 2002). The classification is done by simulation from a variety of lattice initializations to broadly characterize the CA.

1. Class 1: A quick evolution towards a homogeneous global state is observed. All cells stop changing and all the randomness in the initial configuration disappears.
2. Class 2: The CA evolution leads to isolated patterns that could be periodic or stable.
3. Class 3: Pseudo-random or chaotic patterns emerge. Any stable structure is quickly destroyed by the surrounding noise.
4. Class 4: This is the most interesting type of behavior. Patterns that interact in a complex way emerge. These complex patterns are stable for long periods of time. Eventually the complex patterns can settle into a global state like Class 2 behavior but this can take a vast amount of time. Wolfram has conjectured that many Class 4 CA are capable of universal computation (Wolfram 2002).

This classification was inspired by the behavior observed in continuum dynamical systems. For example the homogeneous states of class 1 CA are analogous to fixed-point attracting states, or the repeating structures of class 2 CA are analogous to continuous limit cycles and class 3 chaotic patterns are analogous to strange attractors, while class 4 behavior does not have an obvious continuum analogy (Ilachinski 2001).

Other classification schemes can be found in the literature like: based in the structure of their attractors (Kůrka 1997) or by the structure of their "Garden of Eden" states (non-reachable global states) (Kari 1994).

3.5 Modeling of Complex Systems

Since the consolidation of the scientific method in the XVII century two methodologies emerged for generating and evaluating scientific knowledge. Them being the "experimental" and the "theoretical" paradigms. The experimental paradigm is concerned with observing, measuring and quantifying natural phenomena in order to test an hypothesis. Experimentation also can be made in a playful manner to collect and organize data. The theoretical paradigm seeks logical and mathematical explanations of natural phenomena. Both paradigms are complementary in the sense that predictions can be made by the theoretical paradigm and be tested using the experimental one, then if the experimental

findings support the predictions the theory is kept otherwise is rejected. In other words theory can be supported or falsified through experimentation.

A third scientific paradigm recently appeared, namely the "computational" paradigm. In this approach the study of nature is done through computer simulations. The computational paradigm works in partnership with the "experimental" and "theoretical" ones, so where observed phenomena are not easily tractable by analytical descriptions or direct experimentation is not allowed, computational simulation still permits further inquiry. Still when possible the outputs from the computations can be validated against experimental data and predictions from the theory, thus establishing helpful feedback between paradigms. The invention of the digital computer enabled the numerical solution of analytical models by means of discretization of quantities. Finally the computational paradigm can sometimes be used as a shortcut to the theoretical or experimental paradigms, for example if the problem at hand is well characterized, data obtained from a simulation could be employed as a proxy for real world data or by the contrary if the studied phenomena is poorly understood a first computational approach could be performed.

The theoretical and computational paradigms force us to formally disclose our assumptions in the form of variables, processes and relationships among them, thus forming what is known as an abstract model or simply a model. Mathematical and computational models can be grouped according on how *state*, *space* and *time* variables are abstracted into the model (Hoekstra, Kroc, and Sloot 2010).

Table 3.1: How different models handle *state*, *space* and *time*. *C* stands for continuous and *D* for discrete. The discrete nature of CA is highlighted. Table taken from the book "Simulating complex systems by cellular automata" (Hoekstra, Kroc, and Sloot 2010).

Type of model	State	Space	Time
Partial differential equations (PDEs)	C	C	C
Integro-difference equations	C	C	D
Coupled ordinary differential equations (ODEs)	C	D	C
Interacting particle systems	D	D	C
Coupled map lattices (CMLs)	C	D	D
Systems of difference equations	C	D	D
Lattice Boltzmann equations (LBEs)	C	D	D
Cellular Automata (CA)	**D**	**D**	**D**
Lattice gas automata (LGAs)	D	D	D

Complex systems is a broadly defined term to encompass dynamical systems with more than a few interacting parts commonly in a non-linear way, in simple terms systems that have emergent properties from individual interactions. This kind of

systems are common in the the natural and social sciences. One of the classical examples is the formation of ant colonies and the organization that comes with it (Hölldobler, Wilson, and others 1994). Each ant is sensing external stimuli and acting upon them, the cumulative reactions of all the ants culminates in the building of the ant-hill, feeding it, defending it and attacking other ant colonies. Everything from following from 20 to 40 responses to a particular stimuli.

As seen by the ant-colony example, a correctly chosen set of interacting rules for a group of identical generic entities can create self-organization and/or emergent behavior (Bak 2013). However a general algorithm to find a correct set of local rules to produce a target global behavior is not know (Hoekstra, Kroc, and Sloot 2010). Regardless going in the opposite way is as easy as running a simulation using a proposed set of rules and checking if the yield the target behavior. In summary the question, What set of rules yields this behavior? Is extremely difficult, yet asking, Does this set of particular rules yields this behavior? Is rather easy.

In CA the same dichotomy arises. Going from local rules to global behavior (local to global mapping) is as easy as to just perform the computations defined by the CA but going in the other direction (global mapping to local) is extremely difficult. This issue is known as the "inverse problem". Regardless of its difficulty numerous attempts have been tried, with limited success. A common approach is the usage of optimization techniques like evolutionary algorithms or simulated annealing to find rules driving the system to desired attractor basins (Ganguly et al. 2003).

The semantics of CA made them readily available to model complex systems. In fact Ilachinski mentions that *"CA are, fundamentally the simplest mathematical representations of a much broader class of complex systems."* (Ilachinski 2001). The modeling of a complex system using CA often proceeds in a bottom-up manner. Initially essential properties of the interacting parts are abstracted and then codified into the updating rules of CA cells. Secondly the simulation is run in order to learn the *mesoscopic laws* that emerge from the individual interactions. With domain knowledge this process could be iterated, first proposing essential rules and see if they produce reasonable emergent behavior and then improving the rules from this feedback. Finally the system could be enlarged to a gigantic simulation (mainly in space) to get the *macroscopic* final behavior.

However Toffoli points out that this is not the best allocation of computational budget and recommends to use the learned *mesoscopic laws* as input for higher-level analytical or numerical models (Hoekstra, Kroc, and Sloot 2010). To give an example he imagines a CA that predicts the formation of water droplets from simple interactions of particles. Then if the computational resources are available the model could be escalated millions and millions of times to model fog, clouds, all the way up to global weather. However scaling in this way is not really necessary as once the bulk properties of a water droplet have been found they could easily be feed as numerical parameters of a higher-level model (e.g. differential equation), so in the end the CA had help us to get something

like droplets per cubic meters or temperature an so forth. In Toffoli words *"A successful CA model is the seed of its own demise!"*, nonetheless at the end of the day the CA had help us taming the complex system and clearly expressing the essential microscopic dynamics of the system.

3.6 Computing

An analogy between CA and conventional computers can be made. The initial configuration of a CA could be though as input data to be computed over by the CA rules, producing results several time steps ahead and displayed on whatever configuration reached by the lattice.

This analogy is not coincidence at all and it is further exposed by the history of CA. In the early 1950's von Neumann was trying to build a machine that not only should be self-replicating but also capable of universal computability. Von Neumann's endeavors were successful and produced the first two-dimensional automaton formally shown to be Turing-complete (Von Neumann, Burks, and others 1966). Twenty years later John Conway's "Game of Life" was introduced and later was also found to be computationally universal (Elwin, Conway, and Guy 1982)(Poundstone 2013). More recently 1-D CA "Rule 110" has been proved to be universal and is one of the simplest known systems with such property (Cook 2004).

The usual strategy to prove that a given CA is universal is to show its equivalence with other systems known to be universal. Other strategy is to directly build on the lattice all the primitive elements of computing, namely *storage* (memory), *transmission* (internal clock and wires) and *processing* (AND, OR and NOT gates) (Ilachinski 2001). Once a given system supports these computational primitives building an universal machine becomes a clerical work of assembling modules. The "Game of Life" is proven to be universal in this fashion.

On the other hand possessing the same power as a conventional digital computer plays an important role on our mathematical ability to make predictions on the behavior of CA, because all universal computers require resources in the same order of magnitude to process a particular algorithm, thus in general a computational shortcut to the simulation of any universal CA does not exist (Toffoli 1977). This implies that even if an analytical expression for exactly capturing the evolution of a universal CA is obtained, evaluating such expression would take asymptotically the same time as just running the CA and observing its own evolution. Thus remarkably the most efficient way to characterize a universal CA is through its own simulation (Ilachinski 2001).

Furthermore locality being one of the main ingredients of CA imposes an almost independent update on each cell that is only influenced by its neighbors. As the execution of the program by the CA is being carried out individually by its cells the computations are being executed in a fully parallel manner. Consequently

simulations on CA allow for efficient parallel implementations of any real world system that can be codified into the CA formalism. For example CA based machines CAMs (CA Machines) have been proposed by Toffoli and others (Toffoli 1984). A hardware implementation of a CAM was developed at MIT (Margolus 1995) that for the modeling of complex systems it could achieve a performance of several orders of magnitude higher that a traditional computer at a comparable cost.

3.6.1 Generalized Cellular Automata

Generalizations to the classical attributes of CA can be conceived (Ilachinski 2001) enabling extensions like:

- Asynchronous CA: Allows asynchronous updating of the CA.
- Coupled-map Lattices: Allows real valued cell states. These systems are simpler than partial differential equations but more complex that standard CA.
- Probabilistic CA: The rules are allowed to be stochastic, assigning probabilities to cell state transitions.
- Non-homogeneous CA: Updating functions are allowed to vary from cell to cell. A simple example is a CA with two rules distributed randomly throughout the lattice. On the other extreme case, simulations have been performed with random assignment of all Boolean functions with small number of inputs (Kauffman 1984).
- Boolean Networks: A type of non-homogeneous CA with variable inputs per node (Alonso-Sanz 2011).
- Mobile CA: In this model some cells can move through the lattice. The mobile parts of the CA can be thought as robots or agents and their movement is dictated from an internal state that reflects the features of the local environment.
- Structurally Dynamic CA: Considers the possibility of evolving cell arrangement. In standard CA the lattice is only a substrate for the ongoing computation, but what happens when this passivity is removed.

3.7 History

Precursor ideas about Cellular Automata (CA) can be traced back to 1946 cybernetics models of excitable media of Wiener and Rosenbluth (Weiner and Rosenblunth 1946), however their usual agreed upon inception was when in 1948 John von Neumann following a suggestion from mathematician Stanislaw Ulam

introduced CA to study self replicating systems, particularly biological ones (Von Neumann and others 1951)(Von Neumann, Burks, and others 1966).

Von Neumann's basic idea was to build a lattice in $\mathbb{Z}^2$ capable of copying itself, to another location in $\mathbb{Z}^2$. The solution, in spite of being elaborate and involving 29 different cell states, was modular and intuitive. Since then more constructions capable of the same feat have been found with a lesser number of states (Codd 1968).

In the 1960's theoretical studies of CA were made, especially as instances of dynamical systems and their relation to the field of symbolic dynamics. A notable result from the epoch is the Curtis-Hedlund-Lyndon theorem (Hedlund 1969), which characterizes translation-invariant CA transformations.

In 1969 Konrad Zuse published the book *Calculating Space* (Zuse 1970) with the thesis that the universe is fundamentally discrete as a result of the computations of a CA-like machinery. Likewise during 1960's computer scientist Alvy Ray Smith demonstrated that 1-D CA are capable of universal computation and showed equivalences between Moore and von Neumann neighborhoods, reducing the first to the second (Smith III 1971).

A key moment came with the invention of 2-D CA Game of Life. Pure mathematician J.H. Conway created "Life" as a solitaire and simulation type game. To play "Life" a checkerboard was needed, then counters or chips were put on top of some squares. This represented an initial alive population of organisms and the initial configuration would evolve following reproduction and dying rules. The rules were tweaked by Conway to produce unpredictable and mesmerizing patterns. The game was made popular when was published as recreational mathematics by Martin Gardner in 1970 (Gardner 1970). Despite its name and interesting properties "Life" has little biological meaning and should be only interpreted as a metaphor (Ermentrout and Edelstein-Keshet 1993).

During the 80s the notoriety of CA was boosted to the current status as CA became quintessential examples of complex systems. The focus of research was shifted towards CA as modeling tools. Is in this decade that the first CA conference was held at MIT (Ilachinski 2001) and that a seminal review article of Stephen Wolfram was published (Wolfram 1983).

Since then applications have been coming in a variety of domains. In the biological sciences models of excitable media, developmental biology, ecology, shell pattern formation and immunology, to name a few, have been proposed (Ermentrout and Edelstein-Keshet 1993). CA can be applied in image processing for noise removal and border detection (Popovici and Popovici 2002). For physical systems fluid and gas dynamics are well suited for CA modeling (Margolus 1984). Also they have been proposed as a discrete approach to expressing physical laws (Vichniac 1984).

Table 3.2: Key events in the history of Cellular Automata. Table adapted from the book Cellular Automata A Discrete Universe (Ilachinski 2001).

Year	Researcher	Discovery
1936	Turing	Formalized the concept of computability.
1948	von Neumann	Introduced self-reproducing automata.
1950	Ulam	Realistic models for complex extended systems.
1966	Burks	Extended von Neumann's work.
1967	von Bertalanffy, et al	Applied System Theory to human systems.
1969	Zuse	Introduced the concept of "computing spaces".
1970	Conway	Introduced the CA "Game of Life".
1977	Toffoli	Applied CA to modeling physical laws.
1983	Wolfram	Authored a seminal review article about CA.
1984	Cowan, et al	The Santa Fe Institute is founded.
1987	Toffoli, Wolfram	First CA conference held at MIT.
1992	Varela, et al	First European conference on artificial life.

3.8 Forest Fire Models

Forest fire models are a type of Probabilistic Cellular Automata (PCA). They try to capture the dynamics and general patterns of tree clusters that emerge from an evolving forest subject to perturbations.

They trace their origins to statistical physics and are closely related with percolation phenomena and dissipative structures. However they have been proved a valuable tool for ecological and natural hazard sciences as simple but powerful modeling tools (Zinck, Johst, and Grimm 2010).

Forest fire models help to tackle questions like: Will the tree population eventually dies out?, What is the general shape of tree clusters?, What is the shape of the boundary between the forest and the fire?

At first glance forest fire models seem similar to epidemiological cellular automata models though they place emphasis on finite population and the persistence of a pathology over time in contrast to the infinite forest population and the emphasis on the spatial extension of the fire.

They broadly have the following characteristics (Louis and Nardi 2018):

1. Cells of at least three types:

- Non-burning tree
- Burning tree
- No tree (empty)

2. A rule for fire initiation:

- A starting configuration with fire cells, usually randomly chosen fire positions.
- Accident simulation, like with a small probability self-ignition of a tree cell.
- Space-time distributed ignition instances (e.g. Poisson distributed).

3. A rule for fire propagation. It involves a stochastic rule for fire spreading between neighborhoods that can be based on actual terrain conditions.

3.9 The Drossel and Schwabl forest fire model

The forest fire model that will be used through this document is the Drossel and Schwabl model (DSM) (Drossel and Schwabl 1992).

DSM was born from research on statistical physics about phase transitions and self-organized criticality (Bak, Chen, and Tang 1990)(Drossel and Schwabl 1992). For this reason the CA was not intended as a modeling tool for real wildfires and was only a metaphor.

Nevertheless data from real wildfires was compared against DSM predictions with the, no so surprising, observation that the model did not perfectly match real world datasets, as it was build with the only concern of generating fire sizes following a power law. DSM was overestimating the frequency of large fires (Millington, Perry, and Malamud 2006). Even though its origins and pitfalls DMS is still valuable, as it has a strong advantage against other wildfire models due to its simplicity and analytical tractability (Zinck, Johst, and Grimm 2010). Likewise it provides a set of starting assumptions that can be augmented to the required complexity along with the usually seen trade-off between increasing a model's predictive capabilities and its generalizing power.

Consequently success have been achieved using this simple model, for example fire shape patterns have been obtained that closely resemble actual wildfires (Zinck and Grimm 2008)(Zinck, Johst, and Grimm 2010).

The Drossel and Schwabl model consist of a lattice in $\mathbb{Z}^2$ populated with three type of cells: 0 (no tree), 1 (burning tree), and 2 (non-burning tree). All cells are synchronously updated according to the following rules: For each state $\sigma_k(n)$ at site k and time step n.

- A burning tree is consumed at next time step:

$$\sigma_k(n) = 1 \mapsto \sigma_k(n) = 0 \quad \text{With probability 1}$$

- A new tree grows from an empty position k, dictated by parameter $p \in [0, 1]$:

$$\sigma_k(n) = 0 \mapsto \sigma_k(n) = 2 \quad \text{With probability } p$$

- The fire is propagated through the vicinity or a lightning event occurs, which ignites a tree and is tuned by $f \in [0,1]$:

$$\sigma_k(n) = 2 \mapsto \sigma_k(n) = 1$$

$$\text{With probability} \begin{cases} 1, & \text{if at least one neighboring tree is burning} \\ f, & \text{if no neighboring tree is burning} \end{cases}$$

Chapter 4

Reinforcement Learning

4.1 Machine Learning

Machine Learning is a subfield of Computer Science and Artificial Intelligence (AI). It aims at creating entities capable of learning from examples. What distinguishes it from other AI approaches is the special emphasis on avoiding explicit programming for the task at hand, instead relying only on examples (data) to solve it and from there generalizing to previously unseen cases. So at the heart of this approach lies data. In the classical machine learning setting the algorithm is provided with a set of questions and answers. Then after a period of computations over the pairs questions-answers, the algorithm is presented with new questions, some of them never seen before, that must be answered well enough.

The formalization of this setting leads to an approach known as Supervised Learning, where the questions-answers pairs are codified into mathematical objects, usually numeric vectors for questions and real numbers or categories for answers and the proposed solution comes in the form of a function that maps questions to answers. The name supervised comes from the fact that the algorithm is provided with the answers to the questions, so in a figurative way it is being supervised by a teacher.

Deviation from this classical setting leads to the other broad categories of machine learning. Unsupervised Learning algorithms are provided with just data (questions and not answers) and aim to uncover some structure in such data. Semisupervised Learning is halfway between Unsupervised and Supervised methods as only some answers are given, thus the algorithm must first find some structure on the questions (Unsupervised) to extend the answers to similar questions and then perform Supervised Learning. Finally, Reinforcement Learning algorithms get data and answers, but answers have only a grade on

how favorable they are. The data and grades are obtained through interaction with an environment and the task here is to find answers that maximize the obtained grades.

4.2 Reinforcement Learning

The contents of this section have mostly been adapted from Sutton and Barto's Introduction to Reinforcement Learning book (Sutton and Barto 2018).

Reinforcement Learning (RL) aims to develop algorithms that can satisfactorily inform an agent which decisions to make in order to achieve a goal. The world in which the agent acts is called the environment. The decisions made by the agent affects the environment and hopefully drives it closer to the goal. A signal of how well the agent is performing is received at each decision step. The signal is called reward and it is an abstraction to represent great or poor sequences of actions. Plenty of reward means a successful agent closer to its objective.

The Reinforcement Learning Problem could be formalized with Markov Decision Processes (MDPs). The MDP framework models the interaction between an agent and an environment. The agent takes actions and the environment responds with observations and a reward signal, this process is repeated until termination or forever.

So at each time step $t = 0, 1, 2, 3, ..., T$ the agent using the information of state $S_t \in S$ of the environment must choose an action $A_t \in A(S_t)$ from the available ones, then in the next time step $t + 1$ the environment transits to a new state $S_{t+1} \in S$ and returns a reward $R_t \in R \subset \mathbb{R}$. The dynamics between the agent and environment produces a trajectory of states, actions and rewards indexed by time.

$$S_0, A_0, R_1, S_1, A_1, R_2, S_2, A_2, R_3, ...$$

Of special importance is the case of a finite MDP, in which the cardinality of states, actions and rewards (S, A and R) is finite. In a finite MDP the random variables S_t, R_t have well defined discrete probability distributions that depend only on the previous action $A_t - 1$ and state $S_t - 1$. This is known as the Markov property:

$$p(s', r|s, a) \doteq Pr(S_t = s',\ R_t = r \mid S_{t-1} = s,\ A_{t-1} = a)$$

This $p : S \times R \times S \times A \to [0, 1]$ function is known as the *dynamics of the MDP*. The "|" (bar) symbol instead of a "," (comma) in the function arguments is just a reminder that p is calculating a conditional probability given s and a.

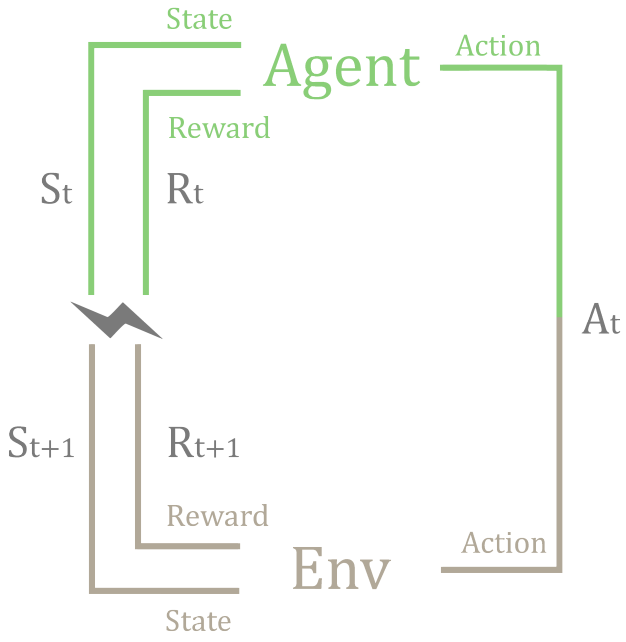

Figure 4.1: Agent-Environment system formalized as a MDP. Figure adapted from (Sutton and Barto 2018).

Chapter 5

Experiments

5.1 Materials and Methods

5.1.1 Environment

A general tool for creating RL tasks was design (*Forest Fire Environment Maker (FFEM)*). The tool generates environments based on the forest fire CA, specifically the Drossel and Schwabl model (DSM) (Drossel and Schwabl 1992). We developed the tool from an initial idea of an "helicopter" flying over the symbolic wild fire, represented by the DMS, trying to extinguish the fire by means of allowing the "helicopter" to change fire cells to empty cells. During the implementation the environment was naturally generalized. The basic idea is to have an agent on top of the CA lattice. The agent has its own state and parameters. Particularly it has an specified position at one of the nodes of the CA, where it can act by changing the cell on that position to another of its cell states. The agent navigates the CA changing cells states. When the agent reads its next instruction it moves to a new cell (or stays in place) and then affects, if applicable, the destination. After some agent movements the CA is updated and the full process is repeated.

The ability of allowing an agent to change the CA configurations express the attempt to drive the CA to a desired global state. In the RL framework this is done by giving a reward to the agent at each emitted action. This was done by mere counting all cell states indicating unwanted behavior and then multiplying by some weight (usually negative). This was generalized into getting the frequency of any cell type and then multiplying by a weight to generate a score per cell type. Then all the scores were combined into a global score. In summary the used reward function was the dot product of the cell-states frequencies by some weights, with the semantics that weights for desirable states are greater than the unwanted ones.

The particular environment that was used for this thesis is one that has the semantics of a "helicopter" trying to extinguish a forest fire. This is represented by an agent influencing the dynamics of a DSM and is as follows: The first component is a DSM with a lattice of size 3x3 with parameters $p = 0.33$ (tree growth probability) and $f = 0.066$ (tree spontaneous burning probability). The used boundary conditions were set to invariant using an extra exogenous cell type that it is not involved in any CA dynamics, we just called them "rocks". The starting lattice configuration was made via random sampling by cell with probability of choosing "tree" of 0.75 and "fire" of 0.25. The second component, the agent ("helicopter") has a starting potion in the middle of the lattice (*2nd row and column*), from there it can move in any direction of a Moore's neighborhood, thus the agent can choose from 9 actions (*"Left-Up", "Up", "Right-Up", "Right", "Stay", "Left", "Left-Down", "Down", "Right-Down"*), then after arriving to its new destination the helicopter extinguishes the fire, represented by changing, if applicable, a "fire" cell to "empty". After some movements of the "helicopter" the forest fire CA is updated following the dynamics dictated by DSM. The specific sequence of agent movements and CA updates is: *move, update, move, move, update, move, move, update, move, move, ...*. In accordance to the RL paradigm, at each performed action the environment must respond with a reward signal, in our setting this was $+1$ per "tree" cell and -1 per "fire" cell, in other words a balance between trees and fires. Finally the RL task was continuous.

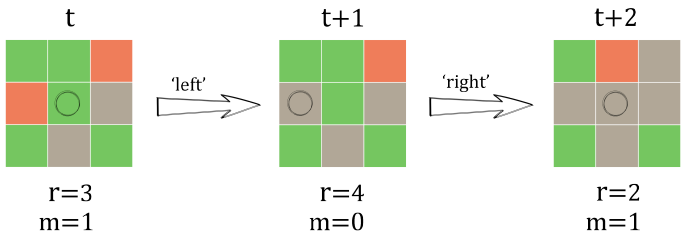

Figure 5.1: Illustration of two transitions in the proposed environment. On time t the helicopter is at 2nd row and column, then it moves to the left, on arriving to its new position it changes the "fire" cell to "empty". Then, at the next time step, it returns to its original position, however the CA was updated before its arrival, anyway the "fire" cell at the middle (Not shown) is promptly eliminated and replaced by an "empty" cell. The parameter m is an internal state of the environment, when it reaches 0 the CA is updated at the next step, before the action takes place. The current reward r is $+1$ per "tree" and -1 per "fire".

5.1.2 Code Availability

The source code for this thesis is openly available at:

1. https://github.com/elbecerrasoto/gym-forest-fire for the RL environments.
2. https://github.com/elbecerrasoto/CARL for the DQN implementations.

5.1.3 Deep Q Networks

We tried to solve the proposed environment using model-free, value-function approximation RL. This approach is justified by the lack of general analytical models for CA dynamics (Ilachinski 2001) and the combinatorial explosion of states. For illustration in our small 3x3 environment the combinatorics are 3^9 grid states multiplied by 3^2 "helicopter" positions and 2 internal states for the agent-environment synchronization, giving a grand total of $2 \times 3^2 \times 3^9 = 3.54294 \times 10^5$, a still manageable quantity by tabular RL standards, however merely scaling to a 16x16 grid leads to an untenable $\approx 10^{124}$ states.

Deep Q Networks (DQN) (Mnih et al. 2013)(Mnih et al. 2015) meets the previously stated conditions and is well suited for tasks with a discrete set of actions.

5.1.3.1 Preprocessing

5.1.3.2 Model Architecture

5.1.3.3 Training Details

5.1.4 Evaluation Procedure

5.2 Results

5.3 Disscusion

Table 5.1: Hyperparameters of 3x3 forest forest environments runs. The mean and standard deviation of reward per step were obtained from a 100,000 steps per model simulation.

Run	Return	Mean	SD	Exploration	Unrolling	Architecture	LR	Batch Size
c	640,658	6.41	2.25	heuristic	2	A3	0.0001	16
a	638,094	6.38	2.30	linear	2	A1	0.0003	32
d	615,091	6.15	2.47	heuristic	3	A3	0.0003	32
i	591,021	5.91	2.53	linear	2	A3	0.0003	32
g	507,313	5.07	3.28	linear	10	A3	0.0003	32
H	503,521	5.04	2.99					
b	327,597	3.28	3.63	linear	1	A2	0.0001	256
R	319,833	3.20	3.44					
h	294,965	2.95	3.56	linear	1	A3	0.0001	16
f	293,722	2.94	3.53	heuristic	1	A3	0.0003	32
e	293,600	2.94	3.54	heuristic	1	A2	0.0001	16

Chapter 6

Conclusion

Alonso-Sanz, Ramon. 2011. *Discrete Systems with Memory*. Vol. 75. World Scientific.

Bak, Per. 2013. *How Nature Works: The Science of Self-Organized Criticality*. Springer Science & Business Media.

Bak, Per, Kan Chen, and Chao Tang. 1990. "A Forest-Fire Model and Some Thoughts on Turbulence." *Physics Letters A* 147 (5-6): 297–300.

Codd, EF. 1968. "Cellular Automata, Academic Press." *New York*.

Cook, Matthew. 2004. "Universality in Elementary Cellular Automata." *Complex Systems* 15 (1): 1–40.

Drossel, Barbara, and Franz Schwabl. 1992. "Self-Organized Critical Forest-Fire Model." *Physical Review Letters* 69 (11): 1629.

Elwin, R Berkelamp, John H Conway, and Richard K Guy. 1982. "Winning Ways for Your Mathematical Plays." Academic Press.

Ermentrout, G Bard, and Leah Edelstein-Keshet. 1993. "Cellular Automata Approaches to Biological Modeling." *Journal of Theoretical Biology* 160 (1): 97–133.

Ganguly, Niloy, Biplab K Sikdar, Andreas Deutsch, Geoffrey Canright, and P Pal Chaudhuri. 2003. "A Survey on Cellular Automata."

Gardner, Martin. 1970. "Mathematical Games." *Scientific American* 222 (6): 132–40.

Hedlund, Gustav A. 1969. "Endomorphisms and Automorphisms of the Shift Dynamical System." *Mathematical Systems Theory* 3 (4): 320–75.

Hoekstra, Alfons G, Jiri Kroc, and Peter MA Sloot. 2010. *Simulating Complex Systems by Cellular Automata*. Springer.

Hölldobler, Bert, Edward O Wilson, and others. 1994. *Journey to the Ants: A Story of Scientific Exploration.* Harvard University Press.

Ilachinski, Andrew. 2001. *Cellular Automata: A Discrete Universe.* World Scientific Publishing Company.

Kari, Jarkko. 1994. "Reversibility and Surjectivity Problems of Cellular Automata." *Journal of Computer and System Sciences* 48 (1): 149–82.

Kauffman, Stuart A. 1984. "Emergent Properties in Random Complex Automata." *Physica D: Nonlinear Phenomena* 10 (1-2): 145–56.

Kůrka, Petr. 1997. "Languages, Equicontinuity and Attractors in Cellular Automata." *Ergodic Theory and Dynamical Systems* 17 (2): 417–33.

Louis, Pierre-Yves, and Francesca R Nardi. 2018. *Probabilistic Cellular Automata.* Springer.

Margolus, Norman. 1984. "Physics-Like Models of Computation." *Physica D: Nonlinear Phenomena* 10 (1-2): 81–95.

———. 1995. "CAM-8: A Computer Architecture Based on Cellular Automata." *arXiv Preprint Comp-Gas/9509001.*

Millington, James DA, George LW Perry, and Bruce D Malamud. 2006. "Models, Data and Mechanisms: Quantifying Wildfire Regimes." *Geological Society, London, Special Publications* 261 (1): 155–67.

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. "Playing Atari with Deep Reinforcement Learning." *arXiv Preprint arXiv:1312.5602.*

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, et al. 2015. "Human-Level Control Through Deep Reinforcement Learning." *Nature* 518 (7540): 529–33.

Popovici, Adriana, and Dan Popovici. 2002. "Cellular Automata in Image Processing." In *Fifteenth International Symposium on Mathematical Theory of Networks and Systems*, 1:1–6. Citeseer.

Poundstone, William. 2013. *The Recursive Universe: Cosmic Complexity and the Limits of Scientific Knowledge.* Courier Corporation.

Smith III, Alvy Ray. 1971. "Simple Computation-Universal Cellular Spaces." *Journal of the ACM (JACM)* 18 (3): 339–53.

Sutton, Richard S, and Andrew G Barto. 2018. *Reinforcement Learning: An Introduction.* MIT press.

Toffoli, Tommaso. 1977. "Cellular Automata Machines."

———. 1984. "CAM: A High-Performance Cellular-Automaton Machine." *Physica D: Nonlinear Phenomena* 10 (1-2): 195–204.

Vichniac, Gérard Y. 1984. "Simulating Physics with Cellular Automata." *Physica D: Nonlinear Phenomena* 10 (1-2): 96–116.

Von Neumann, John, Arthur W Burks, and others. 1966. "Theory of Self-Reproducing Automata." *IEEE Transactions on Neural Networks* 5 (1): 3–14.

Von Neumann, John, and others. 1951. "The General and Logical Theory of Automata." *1951*, 1–41.

Weiner, N, and A Rosenblunth. 1946. "The Mathematical Formulation of the Problem of Conduction of Impulses in a Network of Connected Excitable Elements Specifically in Cardiac Muscle."

Wolfram, Stephen. 1983. "Statistical Mechanics of Cellular Automata." *Reviews of Modern Physics* 55 (3): 601.

———. 2002. *A New Kind of Science*. Vol. 5. Wolfram media Champaign, IL.

Zinck, RD, and V Grimm. 2008. "More Realistic Than Anticipated: A Classical Forest-Fire Model from Statistical Physics Captures Real Fire Shapes." *The Open Ecology Journal* 1 (1).

Zinck, Richard D, Karin Johst, and Volker Grimm. 2010. "Wildfire, Landscape Diversity and the Drossel–Schwabl Model." *Ecological Modelling* 221 (1): 98–105.

Zuse, Konrad. 1970. *Calculating Space*. Massachusetts Institute of Technology, Project MAC Cambridge, MA.