Progress report: cellular automata rule identification

Witold Bołt <witold.bolt@hope.art.pl>

September 17, 2012

1 Problem definition

In this research we develop an automated method of identifying a cellular automata rule from spatio-temporal images (i.e. observations of system evolution in time). Such problem in general is of great importance – we hope to develop a method that could be used to analyze real-world observations and build models based on that.

To examine different methods of constructing such models we start with a simplified version of the problem. Firstly a randomly chosen, one dimensional, binary CA rule R is chosen. Only rules with symmetric neighborhoods with positive radius are considered. Then a set S of selected initial conditions (states of 1D finite lattice) is formed by randomly filling 0's and 1's. After that each space state $s \in S$ is evolved according to R. Exactly T evolutions (iterations) are calculated for each $s \in S$, so for each s we have a spatio-temporal image $i(s)$. This sets our evaluation criteria - we will try to discover the rule R by means of evolutionary computing, relying only on samples $i(s)$.

We shall use following notation:

1. S - set of starting conditions,
2. $s \in S$ - selected starting condition,
3. R - cellular automata 1D rule,
4. $rad(R)$ - radius of the neighborhood of rule R,
5. $i(s)$ - spatio-temporal image of starting condition s containing samples for time $t = 0, 1, \ldots, T$,

6. $|s|$ - count of cells in 1D space of starting state s,

7. $i_t(s)$ - state in time t of evolution $i(s)$,

8. $i_t(s)[j]$ - is value of j-th cell in time t in image $i(s)$,

9. we will use rule symbol R as a function symbol, so that we will write: $R(i_t(s)) = i_{t+1}(s)$. Following this notation applying rule R on state $i(s)$ multiple times (t-times) is denoted with: $R^{(t)}(i(s)) = i_t(s)$.

2 Genetic algorithm

We start by randomly selecting N rules r_i^0, $i = 1, \ldots, N$, with radiuses $0 < \mathcal{R}_1 \leq rad(r_i^0) \leq \mathcal{R}_2$, where $\mathcal{R}_1 \leq rad(R) \leq \mathcal{R}_2$.

Then we evaluate all those rules according to fitness function. For some rule r the fitness function is given by:

$$fit(r) = \frac{1}{|S|} \sum_{s \in S} 1 - \frac{\sum_{t=1}^{T-1} 2^{-t} \left(\sum_{j=1}^{|s|} |r^{(t)}(i_0(s))[j] - i_t[j]| \right)}{|s|(1 - 2^{-T})}.$$

Note that for any rule r we have $0 \leq fit(r) \leq 1$, and $fit(r) = 1$ if $r = R$ or if r is good enough to fully simulate R on given image set.

So the goal of genetic algorithm is to maximize fit function, by selecting best matches, reproducing and manipulating them.

2.1 Selection

We select rules randomly, with weighted chances based on the fitness (i.e. rules with higher fitness have bigger chances of being picked). For the population of N individuals we select $N/2$ pairs and feed them to crossover procedure, and then mutate each of the effecting organisms.

TODO: describe selection in details.

2.2 Cross-over

Crossover between rules r_1 and r_2 can only happen when $rad(r_1) = rad(r_2)$. It it's the case, than we use one-point crossover. We randomly select a point in rules LUT and cross-over entries in LUT starting from this point.

In case where $rad(r_1) \neq rad(r_2)$ then we randomly pick one of the rules, and up-scale or down-scale the other one, to match radiuses.

Up-scaling is always possible and its "lossless", since class of rules with bigger radius is richer than those with lower one. Down-scaling on the other hand looses some information of the rule definition.

TODO: add more details on up-scaling and down-scaling implementation.

2.3 Mutation

We have 3 types of mutations: LUT entry level mutation, random up-scaling and down-scaling.

First kind of mutation simply switches entries in LUT with some small probability p_m.

The additional type of mutations down-scale or up-scale the radius of a rule at random with some small probabilities p_{down}, p_{up}, to give more diversity to the population and to enable searches in different radiuses.

2.4 Sucess criteria

The evolution of genetic algorithm continues with iterative applying of fitness recalculation, selection, cross-over and mutation. In each iteration we additionally keep the best individual from previous iteration without any modifications.

We end the algorithm when there is a rule with $fit(r) = 1$ or when a fixed number of iterations elapsed.

3 Results

TODO: charts, tables, samples..