# Software documentation RCCA-system

Magnus Gundersen Autumn 2016

Software documentation of the program that is used for CA and RC systems

## System overview

The system is used for examining reservoir computing systems, together with cellular automata.

### Prerequisites

The system is implemented in Python.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

## Modules of the system



### Project

Contains all specific tasks that is accomplished in the project. This means that the tasks must be explicitly described, and all parameters that are up to the user should be set here.

Proposed: Each problem (like 5-bit problem) must be described as a RCCA-problem.

This package allows the following functionality:

#### execute\_feedforward\_majority\_task()

Runs the majority task where a bit string is presented to the system, and the system is asked whether there is a majority of 0’s or 1’s.

### Reservoir

Package that contains the possible reservoirs that can be used in the system.

#### CA

Elementary CA is implemented. The CA must be initialized with a rule, that creates a scheme for updating each iteration. The CA is executed by running the run\_simulation() method. This will execute a cellular automation with the given rule, for a given number of iterations. This facilitates the following functionality of the system:

* The CA is the same for all experiments.
* The execution of a CA with R>1 may be done with both concatenating the input-bit string before and after the execution of the CA reservoir itself.
* TODO: Currently it does not facilitate how parallel reservoirs might be implemented, and how information may flow between reservoirs with different size, rules and perhaps also I and R.

### RC-CA-system

Package that contains the specifics of using a CA as a reservoir for reservoir computing. This level of the software is needed because it initializes the encoder with the chosen values, initializes a CA-reservoir and initializes a classifier.

The RC-CA-system facilitates the following functionality:

* Keep all technical configurations, and data of what is available, on one place.
* All initializations are kept to one location in the code
* TODO: allow for parallel reservoirs with the functionality described in RC\_framework
* TOOD: allow rc-ca systems to be saved (pickle?)

### RC-framework

Package that contains the generics of reservoir computing. It is not specific to actually using cellular automata.

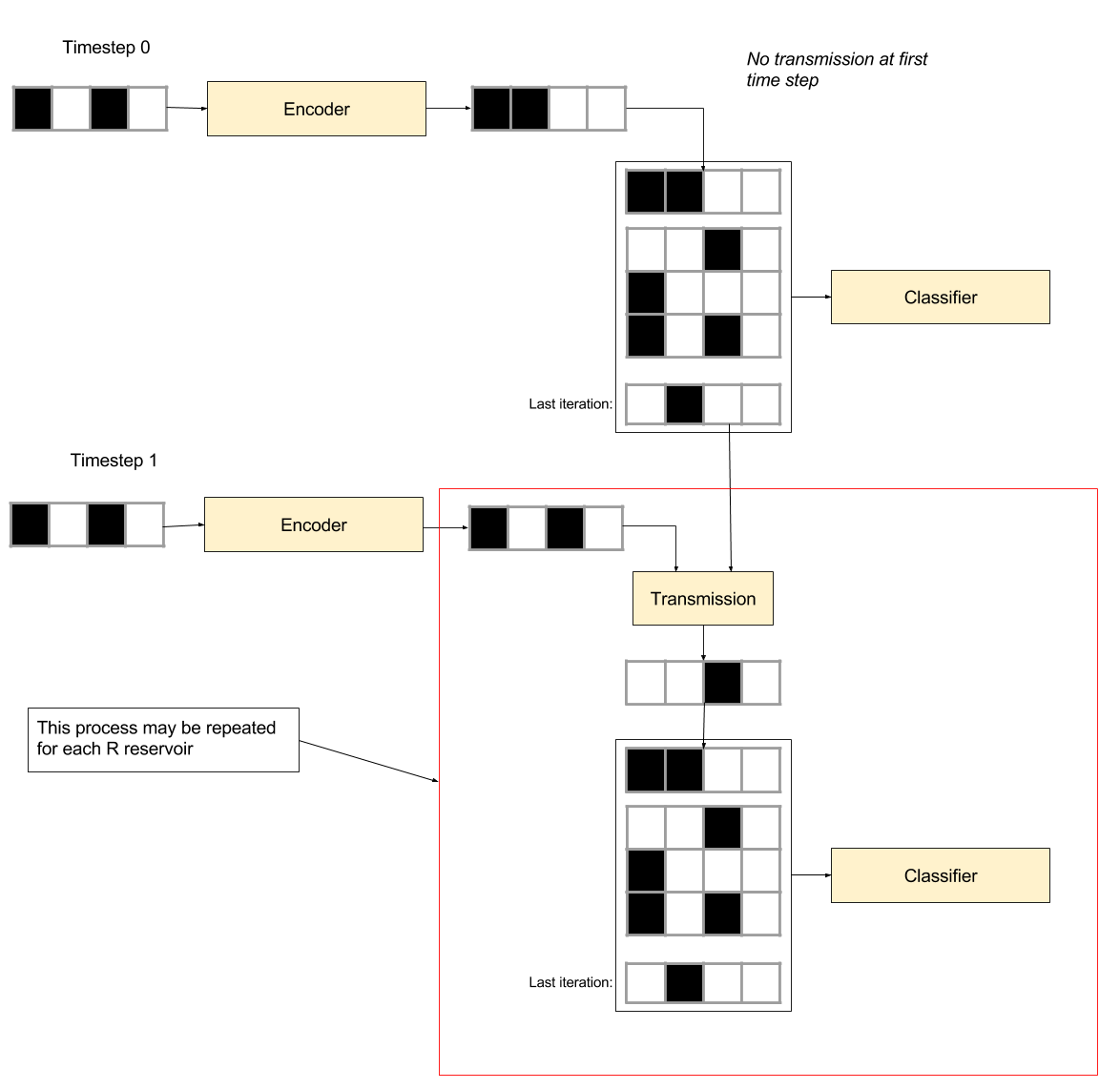
The components (reservoir, classifier etc.) that may be used by this framework is described in the “rc\_interface” module.

The frameworks facilitate the following functionality of the system:

* Train a classifier to learn the outputs of the reservoir.
* TODO: Able to present the RC with timeseries data. This should be incorporated with the non-temporal data trainer.
* Using an encoder with an R-parameter with R>1. This means that the RC-framework must take the possibility of numerous reservoir per input, thus making the flattened vector larger.
* Using encoder to have “concat” or “separate” reservoirs for each value of R.
* TODO: Giving a reasonable representation of the execution of the training to the GUI/visualizer.
* TODO: using parallel reservoirs.
  + With different rules
  + With different connectivity to the other reservoirs

fit\_to\_training\_data(training set, iterations, trans\_scheme)

This method is used to fit the RC-system to any data.



The transmission lets data from previous time steps propagate to the next ones. This is consistent with the echo state property that is needed of reservoirs.

The classifiers in the figure above is the same classifier.

## Specific interactions

### Non-temporal classification procedure

The system is able to be trained to classify non-temporal problems.

### Temporal system classification

A temporal system classifier must be trained with a time series training set. This training set must contain inputs to the system at each time step, and the correct output. As a temporal system, these inputs must be dependent on earlier inputs.

**Example of training-set:**

Training set:

[

[

(0100, 001),

(0100, 001),

(1000, 001)

],

[

(0100, 001),

(0001, 001),

(1000, 001)

],

[

(0100, 001),

(0100, 001),

(1000, 001)

]

]

All of these examples would be fed to the RCCA-system, and the output of the system would be used to

## Miscellaneous