Design strategies for weight matrices of Echo State Networks

Tobias Strauss, Welf Wustlich, Roger Labahn

University of Rostock, Germany

January 20, 2012

Introduction

A considerable problem in theory and practice of artificial recurrent neural networks (RNN) is the training of recurrent connections.

An approach to avoid these problems is the Echo State approach. The Echo State Networks (ESN) consist of three layer:

- An input layer
- a hidden layer with recurrent connections (called dynamical reservoir (DR))
- and the output layer

Echo State Networks

Echo State Networks (as well as most other neural networks) consist of neurons and links. Each neuron has a time dependent activation. We distinguish three kinds of neurons:

- input units providing external activations $\boldsymbol{u} \in \mathbb{R}^K$
- hidden units with internal activations $\boldsymbol{x} \in \mathbb{R}^N$
- output units which generate the systems output signal $\boldsymbol{y} \in \mathbb{R}^L$

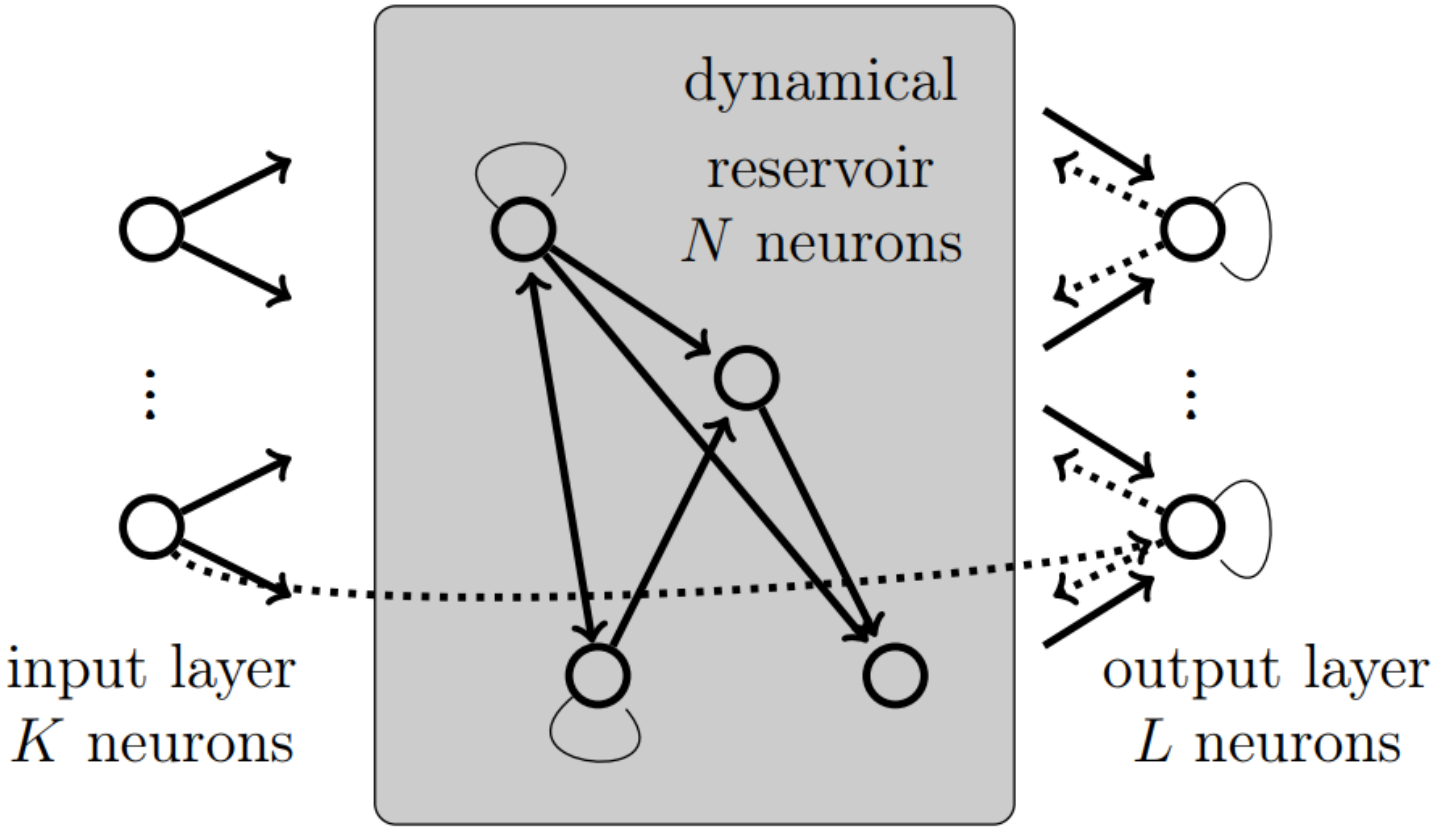
The connections between neurons are weighted. Matrices are usually denoted by W. The activations are calculated by

$$x(n+1) = f(Wx(n) + W^{in}u(n+1) + W^{back}y(n)) \qquad (1)$$

$$y(n+1) = f^{out}(W^{out}(u(n+1), x(n+1), y(n))) \qquad (2)$$

Echo State Networks

Figure 1: Layers of an Echo State Network.



Echo State Networks

Definition 1. Let $u(n) \in U$ and $x(n) \in A$ with compact spaces U and A. Assume that the network has no output feedback connections. Then, the network has Echo States if the state $x(n)$ is uniquely determined by any left-infinite input sequence $\bar{u}^{-\infty}$. More precisely, this means that for every input sequence $..., u(n-1), u(n) \in U^{-\mathbb{N}}$, for all state sequences $..., x(n-1), x(n)$ and $..., x'(n-1), x'(n) \in A^{-N}$, where for any $i \in Z$

$$x(i) = f(W^{in}u(i) + Wx'(i-1)) \tag{3}$$

and

$$x'(i) = f(W^{in}u(i) + Wx'(i-1)) \tag{4}$$

if holds that $x(n) = x'(n)$ for any n.

Echo State Networks

Theorem 2. Assume $f = tanh$ and a network without output feedback.

1. Let the weight matrix W satisfy $\sigma_{max} = \Lambda < 1$, where σ_{max} is it's largest singular value. Then $||x(n+1) - x'(n+1)||_2 < \Lambda||x(n) - x'(n)||_2$ for all inputs $u(n+1)$, for all states $x(n), x'(n) \in [-1, 1]^N$. This implies Echo States for all inputs $u(n+1)$, for all states $x(n), x'(n) \in [-1, 1]^N$.
2. Let the weight matrix W have a spectral radius $\rho(W) > 1$, where $\rho(W)$ is an eigenvalue of W with the largest absolute value. Then the network has an asymptotically unstable null state. This implies that it has no Echo States for any input set U containing 0 and admissible state set $A = [-1, 1]^N$.

Construction

The dynamical reservoir typically is initialized randomly and stays untrained. Herbert Jaeger (Jaeger (2002)) suggested the following procedure:

1. Randomly generate an internal weight matrix W_0
2. Normalize W_0 to a matrix W_1 with unit spectral radius by putting $W_1 = (1/|\rho(W_0)|)W_0$, where $|\rho(W_0)|$ is the spectral radius of W_0.
3. Scale W_1 to $W = \alpha W_1$, where $\alpha < 1$, such that finally W has a spectral radius of α.

Our approach is similar: Instead of Jaeger's step 1 and 2, we specify classes of matrices with spectral radius 1 from which W 0 is taken

Sparse and orthogonal reservoir matrices

Algorithm SORM:

1. Permute the rows of I to avoid small cycles. Then the singular values as well as the absolute values of the eigenvalues of the resulting matrix all are 1.
2. Choose some rotation matrix $Q(h, k, \phi)$ and multiply it from left or from right by the current weight matrix.
3. Repeat step 2 until the desired target density is reached.

CyclicSORMs

Algorithm CyclicSORM:

1. Choose a permutation π of cycle length N. The related matrix is denoted by P.
2. Choose a sparse and orthogonal transformation matrix $\boldsymbol{V}$.
3. Set $W_0 := VPV^T$.

RingOfNeurons and ChainOfNeurons

Let us assume linear activation functions $f = id$ and $f^{out} = id$ for this section. Then the current activations can be calculated as

$$x(n) = Wx(n-1) + w^{in}u(n), \tag{5}$$

$$y(n) = w^{out}x(n). \tag{6}$$

For W and w^{in} generated by the construction of CyclicSORM, we obtain

$$x(n) = (\lambda VPV^T)x(n-1) + Ve_1u(n) \tag{7}$$

and

$$V^Tx(n) = \lambda PV^Tx(n-1) + e_1u(n). \tag{8}$$

RingOfNeurons and ChainOfNeurons

Considering $x(n)$ and $\hat{x}(n) = V^T x(n)$ as states of the same but rotated dynamical system, we can also work with the simpler version $\hat{x}$ generated by the network update

$$\hat{x}(n) = \lambda P \hat{x}(n-1) + e_1 u(n) \tag{9}$$

i.e. with the very simple reservoir matrix P together with input weights e_1.

RingOfNeurons and ChainOfNeurons

According to the permutation matrix P, the internal units of the simplified network are connected in a cyclic way. The units can be relabeled such that without loss of generality the reservoir matrix is

$$\hat{W} := \lambda \begin{pmatrix} 0 & 0 & ... & 0 & 1 \\ 1 & 0 & ... & 0 & 0 \\ 0 & ... & ... & ... & ... \\ ... & ... & ... & 0 & 0 \\ 0 & ... & 0 & 1 & 0 \end{pmatrix} \tag{10}$$

The neural net with this matrix $\hat{W}$ is called *RingOfNeurons*.

RingOfNeurons and ChainOfNeurons

The activation $x_1(n)$ is an uncontrolled superposition of the inputs $u(n - iN), (i \in N)$ which we would like to avoid by forbidding the connection from the last internal neuron to the first one ($\hat{w}_{1,N} = 0$). It remains

$$\hat{W} := \lambda \begin{pmatrix} 0 & 0 & ... & 0 & 0 \\ 1 & 0 & ... & 0 & 0 \\ 0 & ... & ... & ... & ... \\ ... & ... & ... & 0 & 0 \\ 0 & ... & 0 & 1 & 0 \end{pmatrix} \tag{11}$$

The above matrix $\hat{W}$ is nilpotent and has only zero eigenvalues. Such a dynamical reservoir acts like a FIFO-memory. After N steps the system forgets the input. We will refer to this reservoir as a ChainOfNeurons.

Experiments

We compare the error rates of ESNs equipped with one of

- StandardMat - the original randomly generated reservoirs
- SORM
- CyclicSORM
- RingOfNeurons
- ChainOfNeurons

Captions

- ρ - spectral radius, or constant connection weight along the ChainOfNeurons, respectively
- *N* - number of reservoir neurons
- *RDens* - connection density of the reservoir
- *FScale* - feedback scale: bound for weights of output-hidden connections
- *FDens* - feedback density: portion of used output-hidden connections
- *IScale* - input scale: bound for weights of input-hidden connections (see below)
- *IDens* - connection density from the input neurons into the reservoir; x - means only first input unit/vector will be used

Delayline

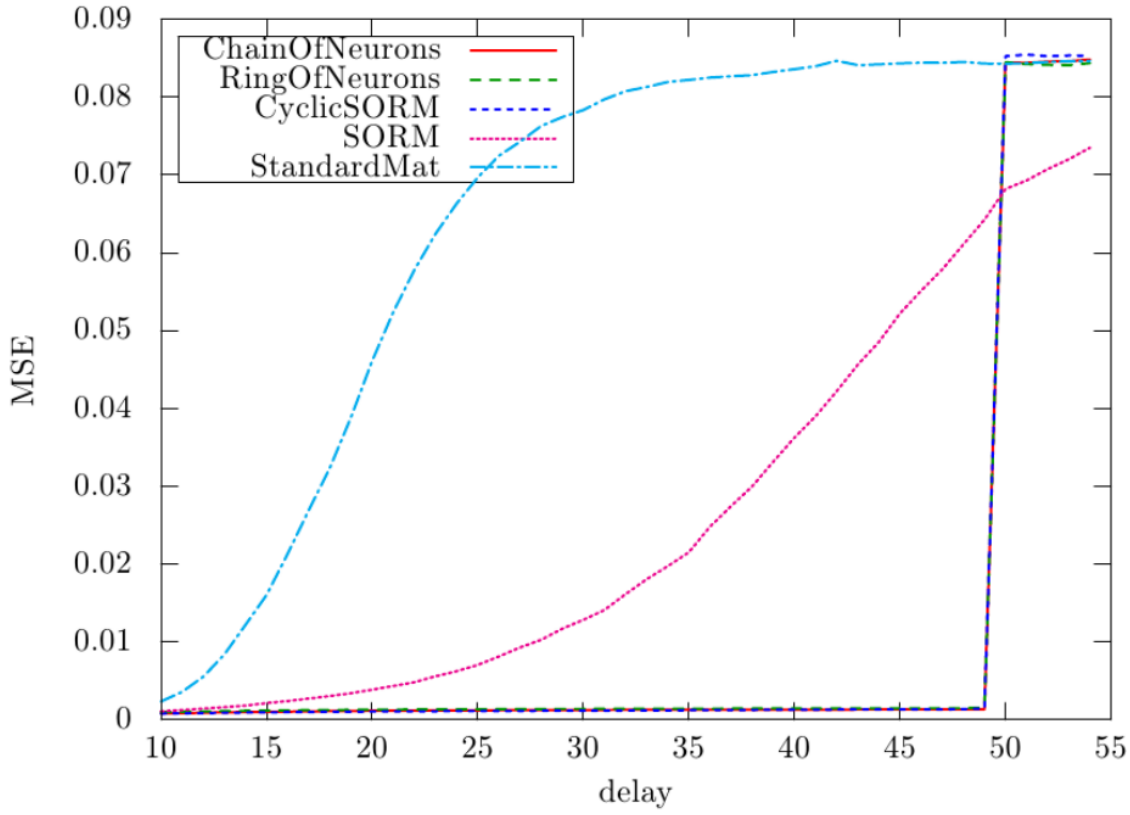
The neural net gets random inputs (uniformly distributed over [-0.5, 0.5]), and the task is to reproduce those inputs after a given time delay at the output of the neural net.

	ρ	*IScale*	*IDens*
ChainOfNeurons	0.95	1.0	$\times$
RingOfNeurons	0.95	1.0	$\times$
CyclicSORM	0.95	1.0	$\times$
SORM	0.95	1.0	0.1
StandardMat	0.99	1.0	0.1

Table 1: Best parameters for the delayline experiment optimized by hand

Delayline

Figure 2: Delayline experiment executed with different reservoir types and averaged over 50 different initializations



Pattern detection

The task is to distinguish between a known pattern (six consecutive values $p_1, ..., p_6 \in [-0.5, 0.5]$) and a random sequence. The input is random sequence merged with this pattern. The intervals between two pattern are of random length.

- Two data sets with 10000 elements + 200 for washout. One is for training, the other is for validation.
- The NN contains 1 input, 1 output and 50 hidden neurons.
- Four dimensions of grid parameters search - bias scaling value, input connection density, scaling value for the input and reservoir weights.
- We say the network has detected a pattern if the output neuron has an activation greater than a certain threshold.

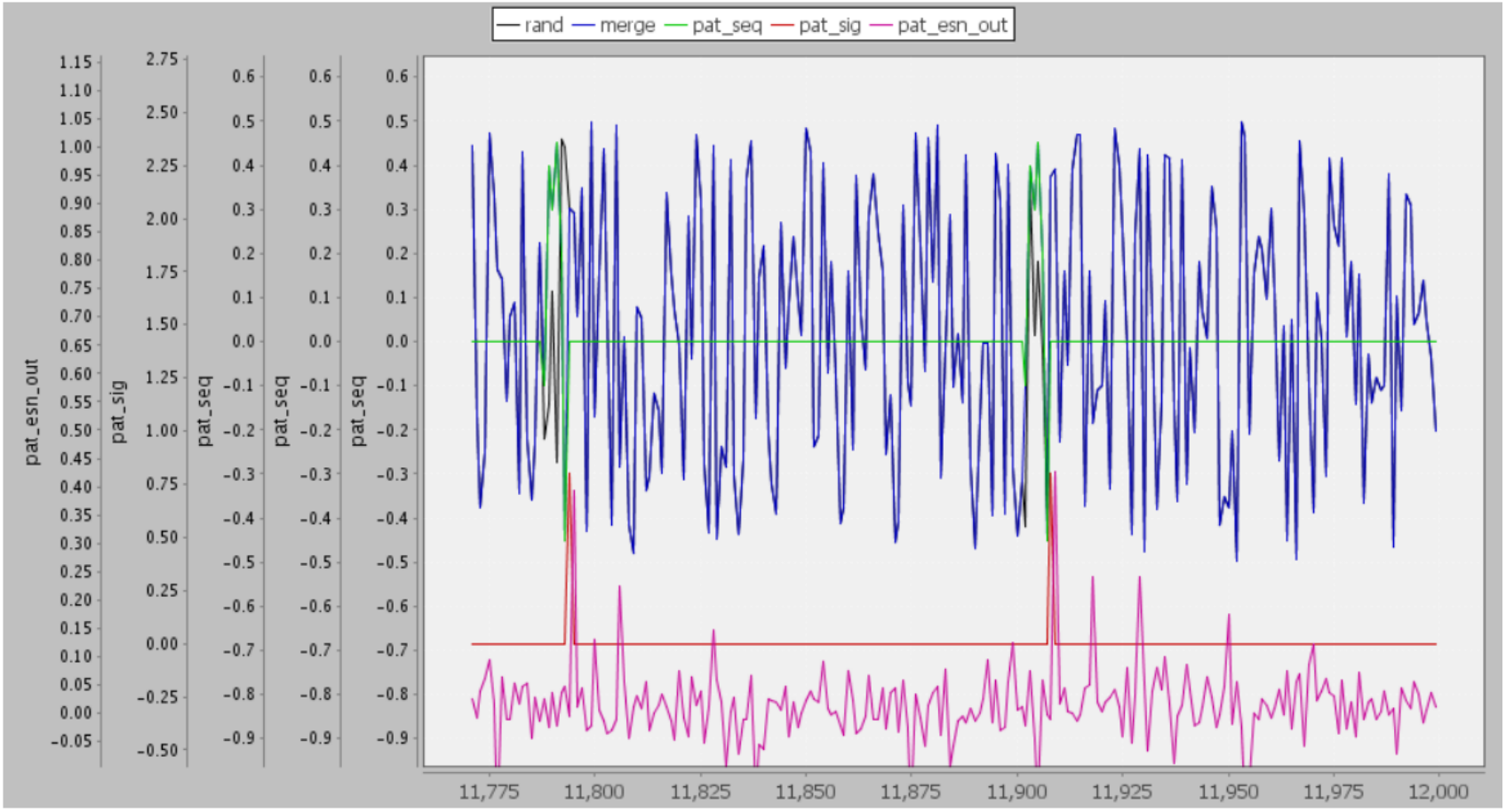
Pattern detection

	ρ	*IScale*	*IDens*
ChainOfNeurons	0.1	0.8	1/7
RingOfNeurons	0.1	1.0	1/9
CyclicSORM	0.15	0.9	1/9
SORM	0.1	1.0	1.0
StandardMat	0.05	1.0	1.0

Table 2: Best parameter setup for the pattern detection experiment found by gridsearch.

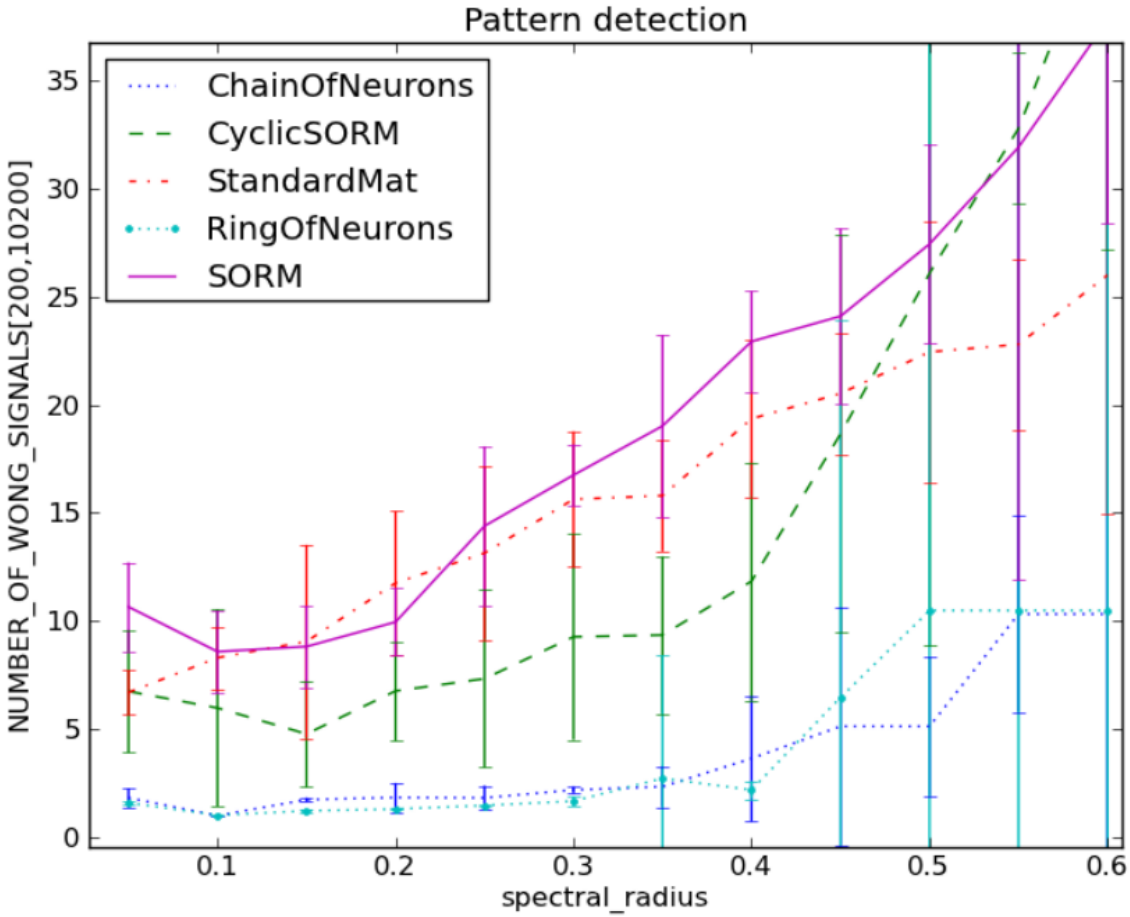
Pattern detection

Figure 3: Pattern detection: time series of the random sequence (black), the merged sequence (blue), the pattern (green), the training signal (red) and the network output (pink)



Pattern detection

Figure 4: Results of pattern detection experiments averaged over 60 different initialization. The error bars denote the variance.



Mackey-Glass

The task is to do a 1-step ahead prediction of Mackey-Glass time series. Mackey-Glass is a nonlinear chaotic time series given by differential equation

$$\dot{y}(t) = \alpha y(t-\tau)/(1+y(t-\tau)^{\beta}) - \gamma y(t) \qquad (12)$$

where parameters are set to $\alpha = 0.2$, $\beta = 10$, $\gamma = 0.1$. For current experiment $\tau = 17$. The system is designed

- with 400 reservoir neurons
- one output neuron aiming to predict the next step
- input is organized by feeding back the output of the system

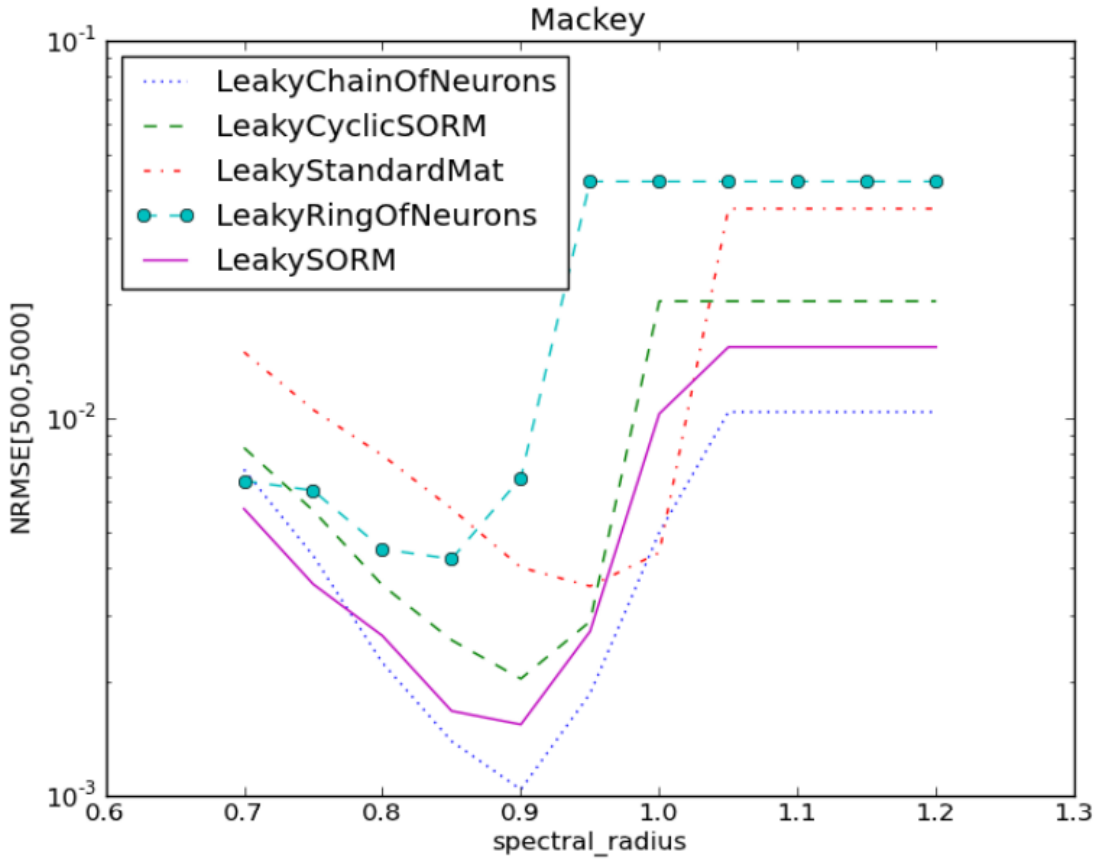
Mackey-Glass

	ρ	*FScale*	*FDens*
ChainOfNeurons	0.85	1.1	1.0
RingOfNeurons	0.95	1.1	1.0
CyclicSORM	1.0	1.2	1.0
SORM	1.05	1.2	1.0
StandardMat	1.1	0.9	1.0

Table 3: Best parameters for the Mackey-Glass experiment

Mackey-Glass

Figure 5: Results of the Mackey-Glass experiment executed with different reservoir types and averaged over 144 different initializations



NARMA

The task is to predict signal for one future step. In this experiment signal is 10th order NARMA

$$t(n+1) = 0.3t(n) + 0.05t(n)\left(\sum_{i=0}^{9} t(n-i)\right) + 1.5u(n-9)u(n) + 0.1 \tag{13}$$

- NN has 200 hidden, 1 input and 1 output neuron
- The data sets contain 2200 elements each, where first 200 steps are for washout

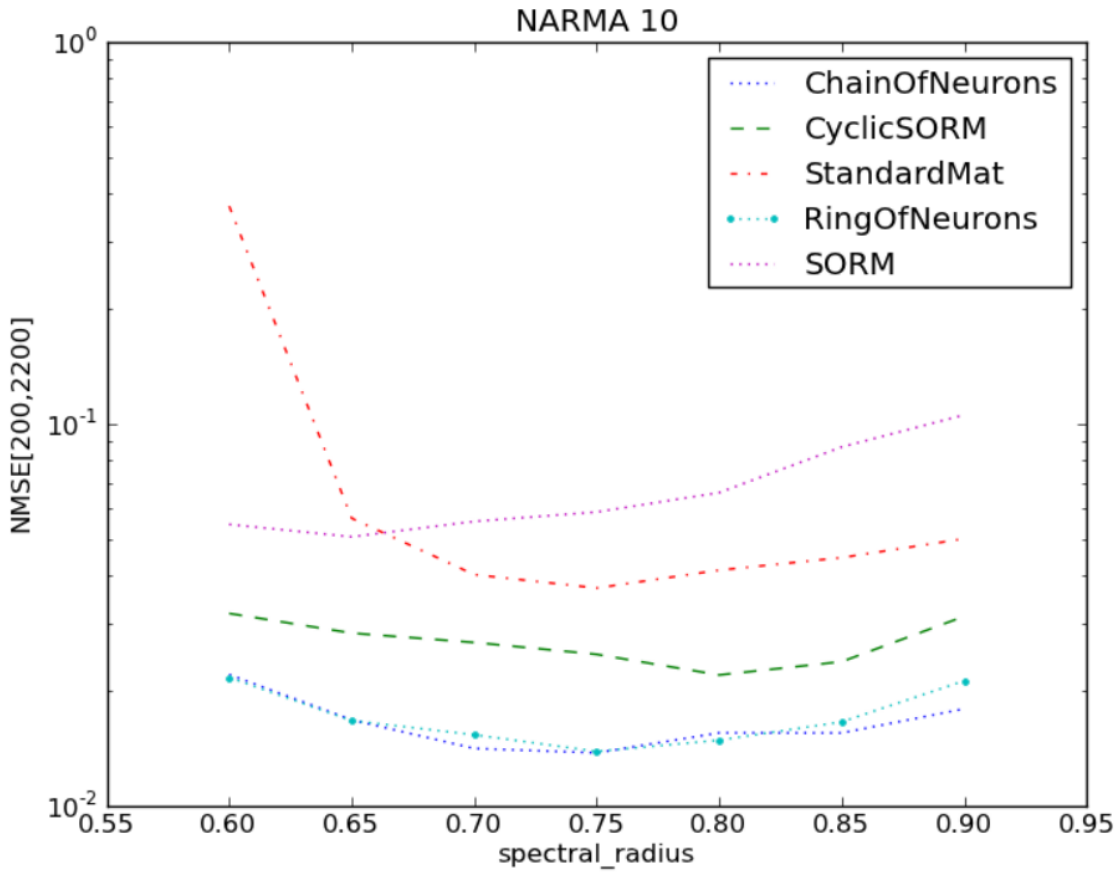
NARMA

	ρ	*IScale*	*IDens*
ChainOfNeurons	0.75	0.95	1/9
RingOfNeurons	0.75	0.95	1/9
CyclicSORM	0.8	0.75	1/9
SORM	0.65	0.45	1/5
StandardMat	0.75	0.2	1/8

Table 4: Best parameters for the NARMA experiment

NARMA

Figure 6: Results of the NARMA experiment executed with different reservoir types and averaged over 50 different initializations.



Conclusions

Different approaches where compared at 4 academic test scenarios:

- short term memory test
- pattern detection
- discretization of the Mackey-Glass differential equation
- NARMA

Surprisingly simple ChainOfNeurons was shown to be the most robust in the linear case and performed very well in experiments. The ChainOfNeurons is not even recurrent system anymore. It seems that the reservoir does not need internal dynamics for many tasks.