Information Theoretic Self-Organised Adaptation in Reservoirs for Temporal Memory Tasks

Sakyasingha Dasgupta[1], Florentin Wörgötter[1], and Poramate Manoonpong[1]

Bernstein Center for Computational Neuroscience, Georg-August-Universität, Göttingen, Germany
{dasgupta,worgott,poramate}@physik3.gwdg.de

Abstract. Recurrent neural networks of the Reservoir Computing (RC) type have been found useful in various time-series processing tasks with inherent non-linearity and requirements of temporal memory. Here with the aim to obtain extended temporal memory in generic delayed response tasks, we combine a generalised intrinsic plasticity mechanism with an information storage based neuron leak adaptation rule in a self-organised manner. This results in adaptation of neuron local memory in terms of leakage along with inherent homeostatic stability. Experimental results on two benchmark tasks confirm the extended performance of this system as compared to a static RC and RC with only intrinsic plasticity. Furthermore, we demonstrate the ability of the system to solve long temporal memory tasks via a simulated T-shaped maze navigation scenario.

Keywords: Recurrent neural networks, Self-adaptation, Information theory, Intrinsic plasticity

1 Introduction

Reservoir Computing (RC) is a powerful paradigm for the design, analysis and training of recurrent neural networks [3]. The RC framework has been utilized for mathematical modeling of biological neural networks as well as applications for non-linear time-series modeling, robotic applications and understanding the dynamics of memory in large recurrent networks in general. Traditionally the reservoir is randomly constructed with only the output connections trained with a regression function. Although both spiking and analog neurons have been explored previously, here we focus on the Echo-state network (ESN) type [2] using sigmoid leaky integrator neurons.

Although the generic RC shows impressive performance for many tasks, the fixed random connections and variations in parameters like spectral radius, leak-rate and number of neurons can lead to significant variations in performance. Approaches based on Intrinsic Plasticity (IP) [1] can help to improve such generic reservoirs. IP uses an information theoretic approach for information maximization at an individual neuron level in a self-organized manner. The IP performance

significantly depends on the type of transfer function, degree of sparsity required and the use of different probability distributions. However the conventional IP method is still outperformed by specific network connectivities like permutation matrices, in terms of the memory capacity performance [8].
We overcome this here, by first utilizing a new IP method [4] based on a Weibull distribution for information maximization. This is then combined with an adaptation rule for the individual neuron leak-rate based on the local information storage measure [7]. Transfer entropy is another measure for such an adaptation rule. However conventionaly this is more difficult to compute, and as it also maximizes input to output information transfer, it is difficult to combine with an IP rule. We achieve such a combination in a self-organized way to guide the individual units for both, maximizing their information and their memory based on the available input. Subsequently through two standard benchmark tasks and a simulated robot navigation task, we show that our adapted network has better performance and memory capacity as compared to static and only IP adapted reservoirs. All three tasks involve a high degree of non-linearity and requirement of adaptable temporal memory. Specifically in robotics and engineering control tasks with nonlinear dynamics and variational inputs (in the time domain), our adaptation technique can show significant performance.

2 Self-Adaptive Reservoir Framework

Here we present the description of the internal reservoir network dynamics and briefly explain (i) The local neuron memory adaptation based on information storage measure (ii) The self-organised adaptation of reservoir neurons inspired by intrinsic plasticity. This is based on maximization of the input-output mutual information of each neuron considering a Weibull distribution as the expected output distribution. Subsequently, we combine both mechanisms for a comprehensive adaptive framework.

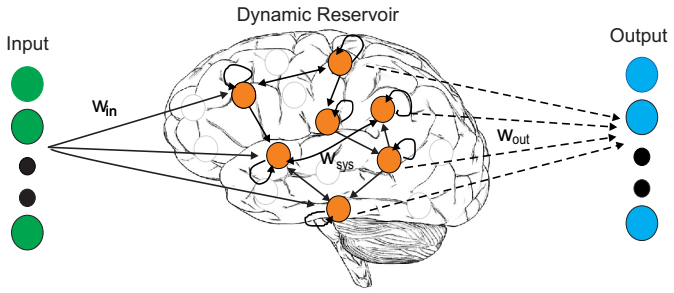

Fig. 1. The Reservoir network architecture, showing the flow of information from input to reservoir to output units. Typically only the output connections are trained. W_{in} and W_{sys} are set randomly.

2.1 Network Description

The recurrent network model is depicted in Fig. 1 as an abstract model of the mammalian neo-cortex. The firing activity of the network at discrete time t is described by the internal state activation vector $\mathbf{x}(t)$. Neural units are connected via weighted synaptic connections $\mathbf{W}$. Specifically $\mathbf{W}_{in}$ are the $K \times N$ connections from the input neurons K to the reservoir neurons N, $\mathbf{W}_{out}$ are the $N \times L$ connections from the reservoir neurons to the output neurons L and $\mathbf{W}_{sys}$ represents the weight matrix for the internal $N \times N$ connections.

The state dynamics at a particular instant of time for an individual unit is given by:

$$\boldsymbol{x}(t+1) = (\boldsymbol{I} - \boldsymbol{\Lambda})\boldsymbol{\theta}(t) + \boldsymbol{\Lambda}[\boldsymbol{W}_{sys}\boldsymbol{\theta}(t) + \boldsymbol{W}_{in}\boldsymbol{v}(t)], \tag{1}$$

$$\boldsymbol{y}(t) = \boldsymbol{W}_{out}\boldsymbol{x}(t), \tag{2}$$

$$\lambda_i = \frac{1}{T_c}(\frac{1}{1+\alpha_i}), \tag{3}$$

where $\boldsymbol{x}(t) = (x_1(t), x_2(t), ..., x_N(t))^T$ is the N dimentional vector of internal neural activation, and $\boldsymbol{v}(t) = (v_1(t), v_2(t), ..., v_K(t))^T$ is the K-dimensional time-dependent input that drives the recurrent network. $\boldsymbol{y}(t) = (y_1(t), y_2(t), .., y_L(t)$ is the output vector of the network. $\boldsymbol{\Lambda} = (\lambda_1, \lambda_2, ..., \lambda_N)^T$ is a vector of individual leak decay rates of reservoir neurons with global time constant $T_c > 0$, while $\alpha_i \in \{0, 1, 2, ..., 9\}$ is the leak control parameter. It is normally adjusted manually and kept fixed, but here it is determined by the local information storage based adaptation rule for each reservoir neuron (Section 2.2).

The ouput firing rate of neurons is given by the vector $\boldsymbol{\theta} = (\theta_1, \theta_2, ..., \theta_N)^T$.

$$\theta_i(t) = \text{fermi}(a_i x_i(t) + b_i). \tag{4}$$

A simple transformation from the fermi-dirac distribution function to the tanh transfer function can be done using:

$$\tanh(x) = 2\theta\text{fermi}(2x) - 1. \tag{5}$$

Here b_i acts as the individual neuron bias values, while a_i governs the slope of the firing rate curve. We adapt these parameters according to IP learning mechanism, presented in Section 2.3.
The output weights $\boldsymbol{W_{out}}$ are computed as the ridge regression weights of the teacher outputs $\boldsymbol{d(t)}$ on the reservoir states $\boldsymbol{x(t)}$. The objective of training is to find a set of output weights such that the summed squared error between the desired output and the actual network output $\boldsymbol{y(t)}$ are minimised. This is done by changing the weights incrementaly in the direction of the error gradient. This could also be done in a one-shot manner using the recursive least squared algorithm [11].

2.2 Neuron Memory Adaptation : Information Storage

An information theoretic measure named *local information storage* refers to the amount of information in the previous state of the neuron that is relevant in predicting its future state. It measures the amount of information stored in the current state of the neuron, which provides either positive or negative information towards its next state. Specically, the instantaneous information storage for a variable x is the local (or un-averaged) mutual information between its semi-infinite past $x_t^{(k)} = \{x_{t-k+1}, ..., x_{t-1}, x_t\}$ and its next state x_{t+1} at the time step $t+1$ calculated for finite-k estimations. Hence, the local information storage is defined for every spatio-temporal point within the network (dynamic reservoir). The local unaveraged information storage can take both positive as well as negative values, while the Average (active) information storage $A_x = \langle a_x(i,t) \rangle$ is always positive and bounded by $\log_2 N$ bits. N is the network size. The local information storage for an internal neuron state x_i is given by:

$$a_x(i, t+1) = \lim_{k \to \infty} \log_2 \left(\frac{P(x_{i,t}^{(k)}, x_{i,t+1})}{P(x_{i,t}^{(k)}) p(x_{i,t+1}} \right). \tag{6}$$

Where $a_x(i, t+1, k)$ respresents finite-k estimates. In case of neurons with a certain degree of leakage (applied after the non-linearity) as introduced in [2] for Echo-state networks (a type of RC), the leakage rate (λ) tells how much a single neuron depends on its input, as compared to the influence of its own previous activity. Since λ varies between 0 and 1, $(1-\lambda)$ can be viewed as a local neuron memory term. Where in, lower the value of λ, stronger the influence of the previous level of activation as compared to current input to the neuron, or high local memory and vice versa.
Using epochs(ϕ) with finite history length $k = 8$, the active information storage measure at each neuron adapts the leak control parameter α_i as follows :

$$\alpha_i = \begin{cases} \alpha_i + 1 & \text{if } A_x(i, \phi) - A_x(i, \phi - 1) > \epsilon \\ \alpha_i - 1 & \text{if } A_x(i, \phi) - A_x(i, \phi - 1) < \epsilon, \end{cases} \tag{7}$$

where $\epsilon = \frac{1}{2} \log_2 N$ and $0 < \alpha_i < 9$.

After each epoch, α_i and λ_i are adjusted and these values are used for the subsequent epoch. Once all the training samples are exhausted the pre-training of reservoir is completed and λ_i is fixed.

2.3 Generic Intrinsic Plasticity

Homeostatic regulation by way of intrinsic plasticity is viewed as a mechanism for the biological neuron to modify its firing activity to match the input stimulus distribution. In [6] a model of intrinsic plasticity based on changes to a neurons non-linear activation function was introduced. A gradient rule for direct minimization of the Kullback-Leibler divergence between the neurons current firing-rate distribution and maximum entropy (fixed mean), exponential output

distribution was presented. Subsequently in [1] an IP rule for the hyperbolic tangent transfer function with a Gaussian output distribution (fixed variance maximum entropy distribution) was derived. During testing the adapted reservoir dynamics, it was observed that for tasks requiring linear responses (NARMA) the Gaussian distribution performs best. However on non-linear tasks, the exponential distribution gave a better performance. In this work, with the aim to obtain sparser output codes with increased signal to noise ratio for a stable working memory task, we implement the learning rule for IP with a Weibull output distribution. The shape parameter of the Weibull distribution can be tweaked to account for various shapes of the transfer function (equation 4). With the aim for a high kurtosis number and hence more sparser output codes, we choose the shape parameter $K = 1.5$. This model was very recently introduced in [4]. However the application of this rule in the reservoir computing framework and its effect on the network performace for standard benchmark tasks had not been studied so far.

The probability distribution of the two-parameter Weibull random variable θ is given as follows:

$$f_{weib}(\theta;\beta,k) = \begin{cases} \frac{k}{\beta}(\frac{\theta}{\beta})^{k-1} exp - \frac{\theta}{\beta}^{k} & \text{if } \theta \geq 0 \\ 0 & \text{if } \theta < 0 \end{cases} \tag{8}$$

The parameters $k > 0$ and $\beta > 0$ control the shape and scale of the distribution respectively. Between $k = 1$ and $k = 2$, the weibull distribution interpolates between the exponential distribution and Rayleigh distribution. Specifically for $k = 5$, we obtain an almost normal distribution. Due to this generalization capability it serves best to model the actual firing rate distribution and also account for different types of neuron non-linearities. The neuron firing rate parameters a and b of equation (4) are calculated by minimising the Kullbeck-Leibler divergence between the real output distribution f_θ and the desired distribution f_{weib} with fixed mean firing rate $\beta = 0.2$. Here the Kullbeck-Leibler divergence is given by:

$$\begin{aligned} D_{KL}(f_\theta, f_{weib}) &= \int f_\theta(\theta) log\Big(\tfrac{f_\theta(\theta)}{f_{weib}(\theta)}\Big) \mathrm{d}\theta \\ &= -H(\theta) + \tfrac{1}{\beta^k} E(\theta^k) - (k-1)E(log(\theta)) - log(\tfrac{k}{\beta^k}) \end{aligned} \tag{9}$$

Where, $f_\theta(\theta) = \frac{f_x(x)}{\frac{\mathrm{d}\theta}{\mathrm{d}x}}$. for a single neuron with input x and output θ

Differentiating D_{KL} with respect to a and b (not shown here) we get the resulting online stochastic gradient descent rule for calculating a and b with the learning rate η and time step h as:

$$\Delta a = \eta\Big[\frac{1}{a} + kx(h) - (k+1)x(h)\theta(h) - \frac{k}{\beta^k}x(h)\theta(h)^k(1-\theta(h))\Big], \tag{10}$$

$$\Delta b = \eta\Big[-\theta(h) + k(1-\theta(h)) - \frac{k}{\beta^k}\theta(h)^k(1-\theta(h))\Big]. \tag{11}$$

IP tries to optimize the neurons information content with respect to the incoming input signal. In contrast the neural local memory adaptation rule tries to modulate the leakage based on a quantification of the extent of influence, the past activity of a neuron has on it's activity in the next time step. We therefore combine IP learning with the neuron memory adaptation rule in series. The leakage adaptation is carried out after the intrinsic adaptation of the neuron non-linearity. This combination leads to a single self-adaptive framework that controls the memory of each neuron based on the input to the entire network.

3 Experiments

To show the performance of our self-adapted RC, we test it on two benchmark tests, namely the NARMA-30 time series modeling task, and a delayed 3-bit parity task. We choose these two tasks taking into consideration the inherent non-linearity and requirements of extended temporal memory. Finally we use a simulated delayed response task of robot navigation though a T-junction. This task shows the potential application of our network for solving real robotic tasks with long delay period between memory storage and retreival.

3.1 Experimental Setup

For all experiments the internal reservoir network was constructed using N=400 leaky integrator neurons initialised with a 10% sparse connectivity. Internal reservoir weights $\boldsymbol{W}_{sys}$ were drawn from the uniform distribution over [-1,1] and were subsequently scaled to a spectral radius of 0.95. Input weights and output feedback weights(if present) can be randomnly generated in general. The firing rate parameters are initialised as $a = 1$ and $b = 0$ with the learning rate for the stochastic gradient descent algorithim fixed at $\eta = 0.0008$. Weibull IP and leak adaptation were carried out in 10 epochs in order to determine the optimal parameters a, b and λ for each neuron. Performance evaluation was done after the neuron leak and transfer function parameters have been fixed. For the delayed 3-bit parity memory task the setup consisted of a single input neuron, the internal reservoir and 800 output units. The simulated robotic task was performed using 6 input neurons (number of sensors) and 2 output neurons (number of actuators) with the internal reservoir size fixed to the original value.

Dynamic System Modelling with 30th Order NARMA : Its dynamics is given by:

$$y(t+1) = 0.2y(t) + 0.004y(t)\sum_{i=0}^{29} y(t-i) + 1.5v(t-29)v(t) + 0.001 \qquad (12)$$

Here $y(t)$ is the output of the system at time 't'. $v(t)$ is the input to the system at time 't', and is uniformly drawn from the interval [0,0.5]. The system

was trained to output $y(t)$ based on $v(t)$. The task in general is quite complex considering that the current system output depends on both the input and the previous outputs. Consequently we use feedback connections ($\boldsymbol{W}_{back}$) from the output neurons to the internal neurons with Equation (1) modified to: $\boldsymbol{x}(t+1) = (\boldsymbol{I} - \boldsymbol{\Lambda})\boldsymbol{\theta}(t) + \boldsymbol{\Lambda}[\boldsymbol{W}_{sys}\boldsymbol{\theta}(t) + \boldsymbol{W}_{in}\boldsymbol{v}(t) + \boldsymbol{W}_{back}\boldsymbol{y}(t)]$, and Equation (2) modified to: $\boldsymbol{y}(t+1) = \boldsymbol{W}_{out}[\boldsymbol{x}(t), \boldsymbol{y}(t)]$. The task requires extended temporal memory. The training, validation and tesing were carried out using 1000, 2000 and 3000 time steps respectively. Five fold cross-validation was used with the training set. Here the first 50 steps was used to warm up the reservoir and was not considered for the training error measure. Performance is evaluated using the normalised root mean squared error:

$$NRMSE = \sqrt{\frac{\langle (d(t) - y(t))^2 \rangle}{\langle (d(t) - < y(t) >)^2 \rangle}} \tag{13}$$

Delayed 3-bit Parity Task : The delayed 3-bit parity task functions over input sequences τ time steps long. The input consists of a temporal signal $v(t)$ drawn uniformly from the interval [-0.5,0.5]. The desired output signal was calculated as the PARITY $(v(t-\tau), v(t-\tau-1), v(t-\tau-2)$ for increasing time delays of $0 \leq \tau \leq 800$. Since the parity function (XOR) is not linearly seperable, this task is quite complex and requires long memory capabilities. We evaluated the memory capacity of the network as the amount of variance of the delayed input signal recoverable from the optimally trained output units summed over all delays. For a given input signal delayed by k time steps, the net memory capcity is given by:

$$MC = \sum_k MC_k = \sum_k \frac{cov^2(y(t-k), d(t))}{var(y(t))var(d(t))} \tag{14}$$

where cov and var denote covariance and variance operations, respectively.

Simulated Robot T-maze Navigation : In order to demonstrate the temporal memory capacity of our system, we employ a four wheeled mobile robot (NIMM4 Fig. 2(a)) to solve a delayed response task. The primary task of the robot was to drive from the starting position untill the T-junction. Then it should either take a left or right turn depending on whether a spherical ball(Yellow) appeared to its right or left before the T-junction. Hence, here the robot has to learn both the reactive behaviarol task of turning at the T-junction as well as remembing the event of a colored ball being shown before. Therefore conventional methods like landmarks to identify the T-junction is not needed. Moreover, to demonstrate the generalization capability of the system to longer time delays, we divided the task into two mazes of different lengths. Maze B requires a longer temporal memory as compared to maze A. NIMM4 consists of four infrared sensors (two on the left and right respectively), a relative distance sensor, a relative angle of deviation sensor and four actuators to control the desired turning and

speed. The experiment consists of data-set aquisition, training of our adapted RC and offline testing. During the first phase using the simulator, we manually controlled the robot movement through the maze using simple keyboard instructions and recorded the sensor and actuator values. We recorded 80 examples in total with different initial starting positions. 40% of these were used for trainng and 60% for offline testing. After the first phase, the self-adapted RC was trained using imitation learning on the collected data with the actuator values from manual control as desired output. Finally we peformed offline testing using the remaining set of recorded data. Simulations were carried out using the C++ based LPZRobot simulator.

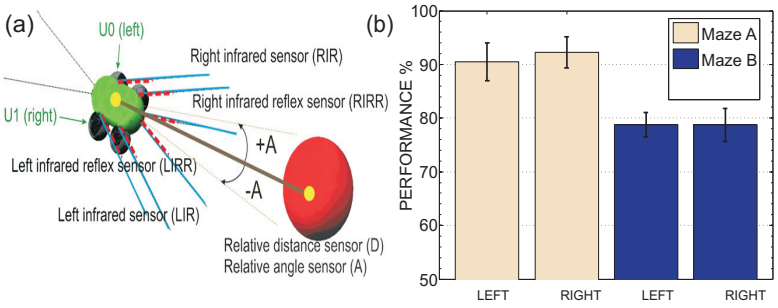

Fig. 2. (a) Model of the simulated robot NIMM4 showing the sensors (LIR,RIR,LIRR,RIRR) and actuators (U0,U1). The red ball in front of the robot represents its goal. (b) Performance of the robot in the two maze tasks (Maze B longer than Maze A) measured in terms of percentage of correct times the robot took the proper trajectory (left/right turn at T-junction). 5% noise is considered on sensors.

3.2 Experimental Results

In Table 1. we summarize the results of our self-adaptive RC with information maximization based on a Weibull distribuition as compared to the performace obtained by a static RC and RC with Gaussian distribution based intrinsic plasticity [1]. Our network outperforms the other two models, both in terms of lowest normalised root mean squared error (0.362) for the 30th order NARMA task, as well as an extended average memory capacity of 47.173 for the delayed 3-bit parity task. Non-normal networks (e.g. a simple delay like network) have been shown to theoretically allow extensive memory [9] which is not possible for arbitary recurrent networks. However our self-adaptive RC network shows considerable increase in the memory capacity, which was previously shown to improve only in case of specifically selected network connections (permutation matrices).

Figure 3(left,centre) shows screenshots of the robot performing the maze navigation task and successfully making the correct turn at the T-junction. The

Table 1. Normalised root mean squared error (NRMSE) and average memory capacity performance for the NARMA-30 and 3-bit parity tasks, comparing the basic RC (ESN) model, the RC model with a intrinsic plasticity method using Gaussian probability density and our self-adapted RC (SRC) network using Weibull probability density.

Dataset	Measure	ESN	RC with IP(GAUSS)	SRC with IP(Weib)
NARMA-30	NRMSE	0.484	0.453	**0.362**
	Std. Dev.	0.043	0.067	0.037
3-bit Parity	Memory Cap.(MC)	30.362	32.271	**47.173**
	Std. Dev.	1.793	1.282	1.831

turn depends on the prior input appearing while driving along the corridor. Our adapted RC is able to successfully learn this task and use only the sensor data to drive along the corridor and negotiate the correct turn. The offline testing results in the form of the percentage of correct turns from the total test set for both mazes are shown in Fig. 2(b). In case of the shorter maze A (smaller memory retention period) we achieve average performance of 92.25% with a standard deviation of 2.88. A good generalization capability for the longer maze B is also observed with the average performace of 78.75% with a standard deviation of 3.11, both for right turn. This is quite high as compared to previous results obtained by [5] for a similar task with static Echo-state network. Furthemore in Fig.3 (right) one can see that our adapted reservoir network clearly outperforms a static RC for the same task. Note that if additional sensors were availabe to the robot, this could further improve the performace, owing to the availability of additional information.

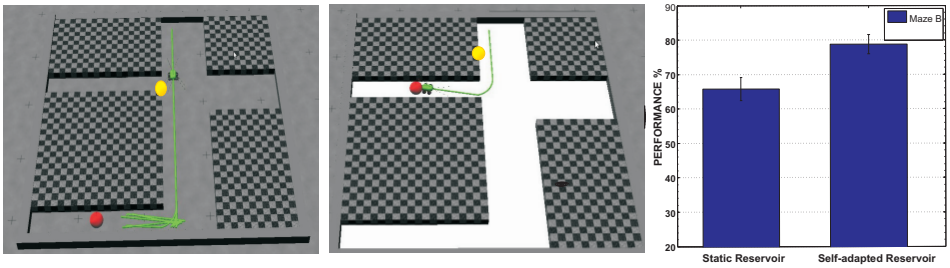

Fig. 3. (left) Screenshots of the robot successfully navigating through the large maze B. Yellow ball is cue to turn right at the T junction, Red ball marks its goal. (centre) Screenshot of the robot navigating through small maze A. (right) Performance on the maze B task after 80 trials for static reservoir vs our self-adapted reservoir. Our network outperforms by 10%. 5% noise is considered on sensors.

4 Conclusion

We have presented and evaluated an adaptation rule for the reservoir computing network that successfully combines intrinsic plasticity using a Weibull output distribution with a neuron leak adjustment rule based on local information storage measure. The neuron leak rate governs the degree of influence of local memory, while intrinsic plasticity ensure information maximization at each neuron output. The evaluated performance on the two benchmark tasks and the robotic simulation demonstrates a reduction in performance error along with an increased memory capacity, of our adapted network as compared to basic RC setups. Future works include using online testing of our network on more complex navigation scenarios. This will require longer working memory and fast adaptation of reservoir time scale. We will also apply this network to an actual hexapod robot AMOSII [10] for enabling memory guided behaviour. Moreover, possible hierarchical arrangement of such adapted reservoirs for both short-term and long term memory (different time scales) within a single framework will be evaluated. **Acknowledgements**: *This research was supported by the Emmy Noether Program (DFG, MA4464/3-1) and the Bernstein Center for Computational Neuroscience II Göttingen (01GQ1005A, project D1).*

References

1. Schrauwen, B.,wardermann, M., Verstraeten, D., Steil, J.J., Stroobandt, D.: Improving Reservoirs using Intrinsic Plasticity. Neurocomputing. 71, 1159–1171 (2008)
2. Jaeger, H., Lukosevicius, M., Popovici, D., Siewert, U.: Optimization and Applications of Echo State Networks with Leaky-integrator Neurons. Neural Networks. 20, 335–352 (2007)
3. Lukoeviius, M., Jaeger, H.: Reservoir Computing Approaches to Recurrent Neural Network Training. Computer Science Review. 3, 127–149 (2009)
4. Li, C.: A model of Neuronal Intrinsic Plasticity. IEEE Trans. on Autonomous Mental Development. 3, 277–284 (2011)
5. Antonelo, E., Schrauwen, B., Stroobandt, D.: Mobile Robot Control in the Road Sign Problem using Reservoir Computing Networks. Proceedings of the IEEE Int. Conf. on Robotics and Automation (ICRA). (2008)
6. Triesch, J.: Synergies between Intrinsic and Synaptic Plasticity Mechanisms. Neural Computation. 4, 885-909 (2007)
7. Lizier, T.J., Pritam, M., Prokopenko, M.: Information Dynamics in Small-world Boolean Networks. Artificial Life. 17, 293–314 (2011)
8. Boedecker, J., Obst, O., Mayer, M.N., Asada, M.: Initialization and Self-Organized Optimization of Recurrent Neural Network Connectivity. HFSP Journal. 5, 340–349 (2009)
9. Ganguli, S., Dongsung, H., Sompolinsky, H.: Memory Traces in Dynamical Systems. PNAS. 105, 18970–18975 (2008)
10. Steingrube1, S., Timme, M., Wörgötter, F., Manoonpong, P.: Self-Organized Adaptation of a Simple Neural Circuit Enables Complex Robot Behaviour. Nature Physics. 6, 224–230 (2010)
11. Jaeger, H.: Adaptive Nonlinear System Identification with Echo State Networks. In Advances in Neural Information Processing Systems. 15, (2003)