Predictive Modeling with Echo State Networks *

Michal Čerňanský[1] and Peter Tiňo[2]

[1] Faculty of Informatics and Information Technologies, STU Bratislava, Slovakia
[2] School of Computer Science, University of Birmingham, United Kingdom
cernansky@fiit.stuba.sk, P.Tino@cs.bham.ac.uk

Abstract. A lot of attention is now being focused on connectionist models known under the name "reservoir computing". The most prominent example of these approaches is a recurrent neural network architecture called an echo state network (ESN). ESNs were successfully applied in several time series modeling tasks and according to the authors they performed exceptionally well. Multiple enhancements to standard ESN were proposed in the literature. In this paper we follow the opposite direction by suggesting several simplifications to the original ESN architecture. ESN reservoir features contractive dynamics resulting from its' initialization with small weights. Sometimes it serves just as a simple memory of inputs and provides only negligible "extra-value" over much simple methods. We experimentally support this claim and we show that many tasks modeled by ESNs can be handled with much simple approaches.

1 Introduction

Echo state network (ESN) [1] is a novel recurrent neural network (RNN) architecture based on a rich reservoir of potentially interesting behavior. The reservoir of ESN is the recurrent layer formed of a large number of sparsely interconnected units with non-trainable weights. ESN training procedure is a simple adjustment of output weights to fit training data. ESNs were successfully applied in several sequence modeling tasks and performed exceptionally well [2, 3]. Also some attempts were made to process symbolic time series using ESNs with interesting results [4].

On the other side part of the community is skeptic about ESNs being used for practical applications [5]. There are many open questions, as noted by the author of ESNs [6]. It is still unclear how to prepare the reservoir with respect to the task, what topologies should be used and how to measure the reservoir quality for example.

The key feature of ESNs is so called "echo-state" property: under certain conditions ESN state is a function of finite history of inputs presented to the network - the state is the "echo" of the input history. It simply means that ESNs are based on contractive dynamics where recurrent units reflect the history of inputs presented to the network. The most recent input has the most important influence to the current network state and this influence gradually fades out.

To study properties responsible for excellent performance of ESN architecture we propose several simplified models. The first we call "feedforward echo state network"

* This work was supported by the grants VG-1/0848/08 and VG-1/0822/08

(FF-ESN). In FF-ESN, a cascaded reservoir topology with triangular recurrent weight matrix is employed and so each FF-ESN can be unfolded into an equivalent feedforward network. By using this architecture we try to reveal that recurrent nature of the ESN reservoir is not the key architectural feature, as it usually is in recurrent multilayer perceptron trained by common algorithm such as backpropagation through time. The next simplified model is ESN-like architecture with reservoir composed of units organized in one chain, hence only elements on recurrent weight matrix subdiagonal have nonzero values. This model benefits from the nonlinear combination of network inputs and this combination is achieved by a very straightforward manner. Finally we have tested performance of linear autoregressive model represented by architecture with linear hidden units organized in chain and thus forming tapped delay line.

2 Models

2.1 Echo State Networks

Echo state networks represent a new powerful approach in recurrent neural network research [1, 3]. They belong to the class of methods known under the name "reservoir computing". Reservoir of ESN is formed of large sparsely interconnected and randomly initialized recurrent layer composed of huge number of standard sigmoid units. As a readout mechanism ESN uses standard output layer and to extract interesting features from dynamical reservoir only output connections are modified during learning process. A significant advantage of this approach over standard RNNs is that simple linear regression algorithms can be used for adjusting output weights. When $\mathbf{u}(t)$ is an input vector at time step t, activations of hidden units $\mathbf{x}(t)$ are updated according to

$$\mathbf{x}(t) = f\left(\mathbf{W^{in}} \cdot [\mathbf{u}(t), 1] + \mathbf{W} \cdot \mathbf{x}(t-1) + \mathbf{W^{back}} \cdot \mathbf{y}(t-1)\right), \tag{1}$$

where f is the hidden unit's activation function, $\mathbf{W}$, $\mathbf{W^{in}}$ and $\mathbf{W^{back}}$ are hidden-hidden, input-hidden, and output-hidden connections' matrices, respectively. Activations of utput units are calculated as

$$\mathbf{y}(t) = f\left(\mathbf{W^{out}} \cdot [\mathbf{u}(t), \mathbf{x}(t), 1]\right), \tag{2}$$

where $\mathbf{W^{out}}$ is output connections' matrix.

Echo state property means that for each internal unit x_i there exists an echo function e_i such that the current state can be written as $x_i(t) = e_i(u(t), u(t-1), \ldots)$, the network state is an "echo" of the input history [1]. The recent input presented to the network has more influence to the network state than an older input, the input influence gradually fades out. So the same input signal history $u(t), u(t-1),$ will drive the network to the same state $x_i(t)$ in time t regardless the network initial state. Echo states are crucial for successful operation of ESNs, their existence is usually ensured by rescaling recurrent weight matrix $\mathbf{W}$ to specified spectral radius λ. This can be achieved by simply multiplying all elements of a randomly generated recurrent weight matrix with λ/λ_{max}, where λ_{max} is the spectral radius of the original matrix.

2.2 Feed-Forward ESN Model

Taking into account the contractive dynamics of ESN exhibited through existence of "echo states", the network output can be seen as a complex nonlinear function of the input history with finite length since the influence of inputs fades out exponentially in time and inputs presented in earlier time steps can be ignored. By removing recurrent connections we propose modified model with the reservoir of units connected in a feed-forward manner. Units in a reservoir can be indexed and their activities depend only on activities of units with smaller indices. No cycles are present in the graph with nodes representing units and edges representing connections. FF-ESN is shown in the Fig. 1a. Technically these connections are still recurrent ones because units are fed by activities

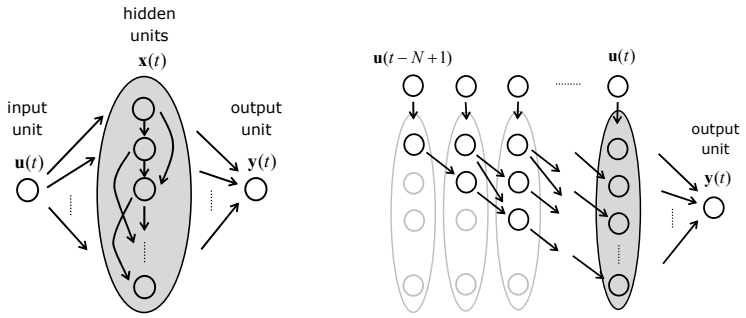

Fig. 1. (a) Modified "feed-forward" ESN architecture. (b) Feed-forwad ESN unfolded in time into regular feed-forward network.

from previous time steps. But this network can be easily transformed into regular feed-forward network by the process identical to the RNN unfolding in time when using backpropagation through time learning algorithm (see Fig. 1b).

Exactly the same training process as is commonly used in training regular ESNs can be used. The only difference is how a recurrent weight matrix is generated. Initial values for biases, recurrent and backward weights falls to the same ranges as described for regular ESNs. Recurrent weight matrix is rescaled to the required spectral radius λ and the matrix is then made lower triangular by keeping only elements below diagonal. To force the ESN to keep longer history of inputs in activities every unit i was connected to the previous one $i-1$ through the weight $w_{i,i-1}$ of chosen constant value, in our experiments we used the value of spectral radius λ.

2.3 Tapped Delay Line Models

Further simplification of ESN architecture resulted in the model with hidden unites organized into a tapped delay line. Hence the only existing recurrent weights are $w_{i,i-1}$ connecting every hidden unit to its predecessor, excluding the first hidden unit. The first tapped delay line model variant labeled TDL-I (tapped delay line with inputs) also allows input and threshold connections for hidden units. TDL-I uses hidden units with

nonlinear activation function and was constructed in order to prove our supposition that very simple nonlinear combination of inputs can be responsible for stunning performance of ESNs on some tasks. The second variant is even greater simplification. Its a TDL with linear hidden units and only the first hidden unit is connected to the input. All connections are set to 1.0. It supposes that the process being modeled can be handled with auto-regressive model. Both models are represented in Fig. 2

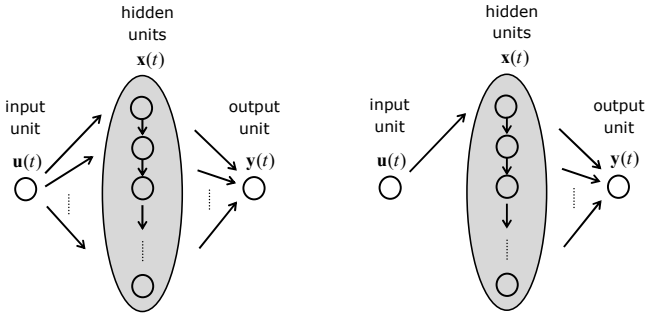

Fig. 2. (a) Architecture with reservoir orginized as TDL with nonlinear units connected to input. (b) Architecture with reservoir formed of linear units organized to TDL.

3 Method

We trained ENS, FF-ESN, TDL-I and TDL models on several time series modeling tasks. We provide mean square error (MSE) results for several datasets taken from published works. For each model and each dataset we have calculated means and standard deviations over 10 simulation runs with the best meta-parameters we were able to find (only number of reservoir units was chosen to match the published experiments).

We used both standard training as proposed in [1] and training using recursive least squares [3] for each of four architectures and each dataset. Since we used output units with linear activation function for all architectures target values were directly used in the least squares fit. All time series are single-dimensional hence all architectures had only one input and one output unit.

The standard training consists of first running the network on the train set and collecting activities of input and hidden units. Initial "washout" activities are thrown away and remaining values forming matrix $\mathbf{X}$ are used for least squares fitting. Output weights are found by solving $\mathbf{X}\mathbf{w} = \hat{\mathbf{y}}$ for $\mathbf{w}$ using singular value decomposition. $\mathbf{w}$ stands for the vecor of output unit weights. It is formed by concatenating output unit threshold, input-output weights and hidden-output weights. Matrix $\mathbf{X}$ has rows corresponding to time steps. The first value in each row is the constant 1.0 corresponding to the output unit threshold followed first by input values and then hidden activities collected from the corresponding time step. Vector $\hat{\mathbf{y}}$ is the vector of target values from the training set.

Recursive least squares training starts after initial "washout" steps. Output weights are updated every time step according to the following equations:

$$\mathbf{k}(t) = \frac{\mathbf{P}(t-1)\mathbf{v}(\mathbf{t})}{\mathbf{v}^T(t)\mathbf{P}(t-1)\mathbf{v}(t) + \gamma}, \tag{3}$$

$$\mathbf{P}(t) = \gamma^{-1}\Big(\mathbf{P}(t-1) - \mathbf{k}(t)\mathbf{v}^T(t)\mathbf{P}(t-1)\Big), \tag{4}$$

$$\mathbf{w}(t) = \mathbf{w}(t-1) + \mathbf{k}(t)[\hat{y}(t) - y(t)], \tag{5}$$

where $\mathbf{k}$ stands for the innovation vector calculated in every time step. $\hat{y}$ and y correspond to the desired and calculated output unit activities. $\mathbf{w}$ is the vector of output weights (threshold, weights from input and hidden units). $\mathbf{P}$ is error covariance matrix initialized with large diagonal values and updated in every time step. Forgetting parameter γ is usually set to the value smaller or equal to 1.0. $\mathbf{v}$ is the vector of activities of input and hidden units form actual time step. The first value of $\mathbf{v}$ is constant 1.0 corresponding to the threshold of the output unit.

4 Results

4.1 Mackey-Glass Chaotic Time Series

Mackey-Glass (MG) system is a chaotic time series where ESNs showed excellent performance exceeding other approaches by several orders [1, 3]. MG system is defined by the differential equations $\partial x/\partial t = \big(0.2(x-\tau)\big)/\big(1 + x(t-\tau)^{10} - 0.1x(t)\big)$ and we used $\tau = 17$ as in [3]. We have generated 10000 values using MG system. Than values were shifted by adding -1 and tanh function was used to squash values into appropriate interval as in [3]. The first 5000 values were used for the training set and remaining 5000 values were used for testing. Initial 1000 from the training set were used only for forward propagation and corresponding hidden units activities were discarded from the adaptation process.

Models with 1000 hidden units were trained for the next value prediction task. Input weights and thresholds for ESN, FF-ESN and TDL-I were initialized from $(-0.2, 0.2)$ interval. Recurrent weights for ESN and FF-ESN were created with 1.0% probability and then recurrent weight matrix was rescaled to spectral radius $\lambda = 0.70$ for ESN and 0.65 for FF-ESN. "Backbone" recurrent connections $w_{i,i-1}$ were set to 0.65 for both FF-ESN and TDL-I model. RLS forgetting parameter γ was set to 1.0. Resulting MSE means together with standard deviations in parenthesis are shown in the Tab. 1. The best performance was achieved using ESN, with FF-ESN and TDL-I models inferior performance was obtained, but within the same order of magnitude. This is in slight contrast from our previous findings, where FF-ESNs performed comparably well on MG generation task [7]. FF-ESN architecture is more sensitive to the network initialization parameters and more thorough parameter searching process could result in some performance improvement. Quite surprising is a very good prediction performance of TDL-I model. Although this model is extremely simple and has far less connections than ESN or FF-ESN, its resulting performance was comparable. Simple linear TDL

	One Step Least Square Fit	Recursive Least Squares
ESN	9.82×10^{-16} (9.02×10^{-17})	3.30×10^{-9} (1.95×10^{-10})
FF-ESN	1.84×10^{-15} (2.13×10^{-16})	1.77×10^{-9} (1.62×10^{-10})
TDL-I	1.88×10^{-15} (2.49×10^{-16})	3.74×10^{-9} (1.19×10^{-10})
TDL	1.18×10^{-8}	1.32×10^{-8}

Table 1. The next value prediction results for Mackey-Glass time series.

model is not able to achieve comparable performance to other models that can exploit nonlinearity through hidden units' activation functions when combining inputs. RLS results for all models are severely inferior to results obtained by standard training, since numerical accuracy is a key factor to achieve results of 10^{-15} order. The required precision is lost when doing recursive updates. On the other hand it's hard to imagine real-life application where results of similar precision would be achievable.

4.2 Nonlinear Communication Channel

We used the same nonlinear channel model as in [3]. First the a sequence $d(n)$ of symbols transmitted through the channel was generated by randomly choosing values from $\{-3, -1, 1, 3\}$. Then $d(n)$ values were used to form $q(n)$ sequence by $q(n) = 0.08d(n+2) - 0.12d(n+1) + d(n) + 0.18d(n-1) - 0.1d(n-2) + 0.09d(n-3) - -0.05d(n-4) + 0.04d(n-5) + 0.03d(n-6) + 0.01d(n-7)$ what represents linear filter equation. Nonlinear transformation is then applied to $q(n)$ sequence to produce row corrupted signal $u(n)$ by $u(n) = q(n) + 0.0036q(n)^2 - 0.11q(n^3) + v(n)$ where $v(n)$ represents the zero mean Gaussian noise. In our experiments no noise was added. The task was to predict value $d(n-2)$ when $u(n)$ was presented to the network. We also shifted $u(n)$ signal by adding value of 30 like in [3].

Models were trained on 4000 values and then the performance was evaluated on the next 6000 values. Initial 100 values from the training set were not used in adaptation process. All models had 47 hidden units (as in [3]), input weights for ESN, FF-ESN and TDL-I models were generated from $(-0.025, 0.025)$ interval. Recurrent weight were generated with 20% probability and the recurrent weight matrix was then rescaled to spectral radius $\lambda = 0.5$. The same value was used for FF-ESN and TDL-I backbone connections. RLS forgetting parameter γ was set to 1.0. Results are given in the Tab. 2. Results for ESN, FF-ESN and TDL-I models are very similar for both standard

	One Step Least Square Fit	Recursive Least Squares
ESN	0.022 (0.0016)	0.028 (0.010)
FF-ESN	0.022 (0.0017)	0.025 (0.0039)
TDL-I	0.018 (0.0024)	0.026 (0.0027)
TDL	0.052	0.053

Table 2. The next value prediction results for nonlinear communication channel.

training and RLS training, although slight degradation of performance can be observed

for RLS. Simple architecture of TDL-I model is sufficient to obtain good results. Although performance of linear TDL model is much inferior, it is still comparable to other models.

4.3 NARMA System

The 10th order nonlinear autoregressive moving average (NARMA) system from [8] defined as $d(n+1) = 0.3d(n)+0.05d(n)\left[\sum_{i=0}^{9} d(n-i)\right]+1.5u(n-9)u(n)+0.1$ was used. Values for input sequence $u(n)$ were generated uniformly from interval $[0, 0.5]$ and the task was to predict $d(n)$. Training set was composed of 2200 values and first 200 values were used as initial washout steps. Next 2000 values were used to train models. Testing was performed on the next 2000 values. No noise was added to the sequence.

We trained models with hidden layer formed of 100 units. Input weight were initialized from $(-0.1, 0.1)$ interval. Recurrent weights were created with 5% probability and recurrent weight matrix was rescaled to have spectral radius $\lambda = 0.95$ (for ESN and FF-ESN model). Backbone connections (for FF-ESN and TDL-I models) were set to the same value. RLS forgetting factor was set to $\gamma = 1.0$. Results are shown in Tab. 3. Results reveal only slight differences between standard and RLS training. Both

	One Step Least Square Fit	**Recursive Least Squares**
ESN	0.00091 (0.00017)	0.00084 (0.000084)
FF-ESN	0.00090 (0.00011)	0.00096 (0.00027)
TDL-I	0.00132 (0.00021)	0.00141 (0.00023)
TDL	0.00189	0.00189

Table 3. Prediction results for NARMA system.

ESN and FF-ESN models achieved similar performance. Significantly worse but yet comparable performance was achieved with TDL-I and TDL models.

4.4 Predictive Modeling Problem

The next task is called predictive modeling taken form [9]. Time series of 10000 values was generated using $sin(n + sin(n))$ for $n = 1, 2, \ldots$, the first 1000 values initial transient activities are thrown away, training is performed on the next 4000 values and predictive performance is evaluated on the next 5000 values.

Models with 600 units were tested, small $\lambda = 0.35$ value was used to rescale recurrent weight matrix for ESN and FF-ESN models. The same value was also used for backbone connections of FF-ESN and TDL-I models. Recurrent weights for ESN and FF-ESN were created with 10% probability. Input weights for ESN, FF-ESN and TDL-I were created from $(-0.25, 0.25)$ interval. Hidden units had no threshold connections. Results are shown in Tab. 4. We were able to train models using nonlinear activation function to the same level of performance. Linear TDL showed significantly poorer performance. ESN model in [9] was trained with suboptimal parameters, scaling

	One Step Least Square Fit	Recursive Least Squares
ESN	$0.00046\ (5.20 \times 10^{-5})$	$0.018\ (0.0019)$
FF-ESN	$0.00055\ (6.22 \times 10^{-5})$	$0.0074\ (0.00066)$
TDL-I	$0.00058\ (6.52 \times 10^{-5})$	$0.015\ (0.00081)$
TDL	0.056	0.056

Table 4. The next value prediction results for predictive modeling problem.

recurrent weights to relatively small spectral radius resulted into significant prediction improvement. We were not able to train models to the comparable level of performance with RLS, much better results were obtained using standard one-step linear regression.

4.5 Multiple Superimposed Ocillator

Multiple superimposed oscillators time series was used for the next value prediction problem in [9, 10]. Dataset values are created as $sin(0.2n) + sin(0.311n)$ for $n = 1, 2, \ldots$. Sequence of 1000 values was generated, the first 100 values were used for initial transient steps. Training was performed on the following 600 values and predictive performance was evaluated on the next 300 values.

We provide results for models with 400 hidden units. Input weight were initialized from $(-0.2, 0.2)$ interval. Recurrent weights were created with 10% probability and recurrent weight matrix was rescaled to have spectral radius $\lambda = 0.5$ for ESN for FF-ESN models. Also backbone connections for FF-ESN and TDL-I models were set to 0.5. RLS forgetting factor was set to $\gamma = 1.0$. Results are shown in Tab. 5. Results

	One Step Least Square Fit	Recursive Least Squares
ESN	$1.30 \times 10^{-26}\ (1.11 \times 10^{-26})$	$5.64 \times 10^{-12}\ (2.27 \times 10^{-12})$
FF-ESN	$6.30 \times 10^{-27}\ (4.79 \times 10^{-27})$	$1.34 \times 10^{-12}\ (3.14 \times 10^{-13})$
TDL-I	$7.45 \times 10^{-27}\ (3.59 \times 10^{-26})$	$5.35 \times 10^{-12}\ (1.37 \times 10^{-12})$
TDL	1.04×10^{-28}	1.90×10^{-24}

Table 5. The next value prediction results for multiple superimposed oscillator problem.

reveal that MSO problem can be modeled by simple linear autoregressive model and hence linear TDL outperforms other models exploiting nonlinearity by several orders of magnitude. All models were trained to significantly better performance than in [9] since they are equipped with linear output units. Because of smaller λ values hidden units operate in linear part of their activation functions and models achieve better results than if higher λ values were used.

4.6 IPIX Radar

IPIX radar data are taken from [9] where the dataset of 2000 values was used as noisy nonlinear real-life prediction task. The first 200 values were used as initial transient

inputs, the training was performed using the next 800 values and predictive performance was evaluated on the remaining 1000 values.

Models of 80 units were tested. Input weight were initialized from $(-0.2, 0.2)$ interval. Recurrent weights were created with 10% probability and recurrent weight matrix was rescaled to have spectral radius $\lambda = 0.9$ (for ESN and FF-ESN model). Backbone connections (for FF-ESN and TDL-I models) were set to the same value. RLS forgetting factor was set to $\gamma = 1.0$. Results are shown in Tab. 6. We have achieved slightly

	One Step Least Square Fit	**Recursive Least Squares**
ESN	0.00079 (0.000043)	0.00071 (0.000039)
FF-ESN	0.00074 (0.000052)	0.00070 (0.000019)
TDL-I	0.00075 (0.000051)	0.00073 (0.000016)
TDL	0.00084	0.00079

Table 6. The next value prediction results for IPIX Radar data.

better accuracy as in [9] for ESN, FF-ESN and TDL-I models. Also there was no difference between standard training and RLS training performance. Linear TDL model achieved slightly worse results, but still comparable to results of other models.

5 Conclusion

Several ESN model simplifications were suggested in this work and multiple simulations of different tasks taken from literature were performed. Summary of results is shown in Tab. 7. FF-ESN architecture with lower-triangular recurrent weight matrix is only a minor modification to ESN and on most datasets achieved comparable results to standard ESN. Surprisingly this is also the case of TDL-I model with hidden units organized into tapped delay line. This model is based on more simple and more straightforward way of combining inputs presented to the network and performed very well in comparing with standard ESN on multiple tasks. Experiments with TDL model with linear hidden units forming tapped delay line revealed that some tasks used in the community can be simply handled as linear autoregressive models. One should be aware that for some tasks ESN reservoir can serve just as simple memory of inputs. We encourage researchers to compare results obtained using ESN models with standard techniques such as autoregressive moving average model. Using appropriate parameters for ESN reservoir preparation can result into important performance improvement. For most of the tasks from [9] we were able to find better parameters and results differ significantly. Nevertheless searching for good parameters is difficult and time demanding process and more appropriate parameters may still remain undiscovered.

Many improvements were suggested for ESNs in the literature: decoupled echo state networks [9], refined version of the training algorithm [3], working with enhanced states [8], using leaky-integrator units [1] etc. In this work we have tried to simplify ESN architecture to better understand where the performance gains originate. It seems that for some of tasks taken from other papers, no complex reservoir architecture is required

One Step Least Squares Fit	ESN	FF-ESN	TDL-I	TDL
Mackey Glass Chaotic Time Series	9.82×10^{-16}	1.84×10^{-15}	1.88×10^{-15}	1.18×10^{-8}
Nonlinear Communication Channel	0.022	0.022	0.018	0.052
NARMA System	0.00091	0.00090	0.00132	0.00189
Predictive Modeling Problem	0.00047	0.00055	0.00058	0.056
Multiple Superimposed Oscillators	1.30×10^{-26}	6.30×10^{-27}	7.47×10^{-27}	1.04×10^{-28}
IPIX Radar	0.00079	0.00074	0.00075	0.00084
Recursive Least Squares	**ESN**	**FF-ESN**	**TDL-I**	**TDL**
Mackey Glass Chaotic Time Series	3.30×10^{-9}	1.77×10^{-9}	3.74×10^{-9}	1.32×10^{-8}
Nonlinear Communication Channel	0.028	0.025	0.026	0.053
NARMA System	0.00084	0.00096	0.00141	0.00189
Predictive Modeling Problem	0.018	0.0074	0.015	0.056
Multiple Superimposed Oscillators	5.64×10^{-12}	1.34×10^{-12}	5.35×10^{-12}	1.90×10^{-34}
IPIX Radar	0.00071	0.00070	0.00073	0.00079

Table 7. Summary of MSE results.

to achieve comparable results. Interestingly enough, iterative RLS training was not as accurate as standard one-step training algorithm. Extremely high precision observed when using ESNs on some tasks is achieved by using one step linear regression for training.

References

1. Jaeger, H.: The "echo state" approach to analysing and training recurrent neural networks. Technical Report GMD 148, German National Research Center for Information Technology (2001)
2. Jaeger, H.: Adaptive nonlinear system identification with echo state networks. In Becker, S., Thrun, S., Obermayer, K., eds.: Advances in Neural Information Processing Systems 15, MIT Press, Cambridge, MA (2003) 593–600
3. Jaeger, H., Haas, H.: Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science **304**(5667) (2004) 78–80
4. Frank, S.L.: Learn more by training less: Systematicity in sentence processing by recurrent networks. Connection Science, in press (2006)
5. Prokhorov, D.: Echo state networks: Appeal and challenges. In: Proceedings of International Joint Conference on Neural Networks IJCNN 2005, Montreal, Canada. (2005) 1463–1466
6. Jaeger, H.: Reservoir riddles: Suggestions for echo state network research. In: Proceedings of International Joint Conference on Neural Networks IJCNN 2005, Montreal, Canada. (2005) 1460–1462
7. Čerňanský, M., Makula, M.: Feed-forward echo state networks. In: Proceedings of International Joint Conference on Neural Networks IJCNN 2005, Montreal, Canada. (2005) 1479–1482
8. Jaeger, H.: Adaptive nonlinear system identification with echo state networks. In: Proceedings of Neural Information Processing Systems NIPS 2002, Vancouver, Canada. (2002)
9. Xue, Y., Yang, L., Haykin, S.: Decoupled echo state network with lateral inhibition. IEEE Transactions on Neural Network, in press (2007)
10. Wierstra, D., Gomez, F.J., Schmidhuber, J.: Modeling systems with internal state using evolino. In: GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation, New York, NY, USA, ACM (2005) 1795–1802