Echo State Networks and Self-Prediction

Norbert M. Mayer and Matthew Browne

GMD- Japan Research Laboratory
Collaboration Centre
2-1 Hibikino, Wakamatsu-ku
Kitakyushu, 808-0135
Japan
norbert.mayer@gmd.gr.jp

Abstract. Prediction occurs in many biological nervous systems e.g. in the cortex [7]. We introduce a method of adapting the recurrent layer dynamics of an echo-state network (ESN) without attempting to train the weights directly. Initially a network is generated that fulfils the echo state - liquid state condition. A second network is then trained to predict the next internal state of the system. In simulation, the prediction of this module is then mixed with the actual activation of the internal state neurons, to produce dynamics that a partially driven by the network model, rather than the input data. The mixture is determined by a parameter α. The target function be produced by the network was $sin^3(0.24t)$, given an input function $sin(0.24t)$. White noise was added to the input signal at 15% of the amplitude of the signal. Preliminary results indicate that self prediction may improve performance of an ESN when performing signal mappings in the presence of additive noise.

1 Introduction

Recurrent Neural Networks (RNNs) are remarkably versatile sequence processing devices. In principle, they are capable of performing a wide range of information processing tasks on temporally distributed input, by capturing the input history via recurrent connections. However, although RNNs possess very interesting theoretical properties, their efficacy for learning long connected sequences remains questionable [10]. Attempts to learn long sequences of data have not shown very good results. In particular, learning features and patterns on a very long time scale appears problematic for the basic RNN architecture [11].

Recently a new type of RNN approach, the *echo state* RNN (ESN) approach, has been published [3]. In contrast to previous approaches, the recurrent connections are are not trained. Rather, a constant, random connectivity recurrent weight matrix is used. The learning is performed only in the connections between the hidden layer and the output layer. The recurrent layer acts as a large 'reservoir of echos' of previous inputs, which may be tapped for information by the adaptive layer [1]. In a standard RNN there are many parameters to be optimized in the recurrent layer. Well-formed techniques for deciding appropriate values for these parameters are still in the development stage. The ESN approach makes

A.J. Ijspeert et al. (Eds.): BioADIT 2004, LNCS 3141, pp. 40–48, 2004.

it possible to exploit the advantages of RNNs whilst sidestepping the difficulties of training that arise in training the recurrent weight connections.

In order to qualify as an ESN, the network must satisfy the conditions outlined in [1]. The echo-state condition means that although recurrent connections exist these connections are weak enough that for sufficient long input histories the system converges to a single attractor and this attractor does not depend on the initial state of the network. Instead, the system depends on the input history. We note that *liquid state machines* [4] represent an approach that is formally equivalent; both concepts were introduced independently at roughly the same time.

Some recent research has shown that this approach may be successfully applied to many tasks. However, some problems remain to be solved. One problem is noise reduction. In the classic ESN approach learning is performed only in the output layer. The internal dynamics are highly complex and recursive function of the input. However, because the recurrent weight matrix is fixed and random, the internal dynamics are determined by a random function of the input history. The burden of demapping this function to perform the prediction task falls entirely on the linear output layer. This appears to place some limitations in the efficacy of the resulting mapping, for instance with regard to its sensitivity to additive noise. Further, a dilemma instrinsic to the ESN architecture exists in the relative weighting that should be given to the internal dynamics and the input signal. Jaeger [2] notes that this weighting is parameterized by the maximum eigenvalue of transformation matrix: as this variable approaches unity, the internal dynamics of the network become increasingly self-sustaining in the absence of an input signal. Determining the appropriate maximum eigenvalue is somewhat problematic; smaller values increase sensitivity and responsiveness of the network over short itme intervals but at the expense of memory at longer time scales. Increasing this factor has the opposite effect.

We note that both the issues mentioned above are related to the fact that the network hidden activations are coupled statically to the input / input history. In order to address these problems, one approach is to allow some degree of adaption in the recurrent connections. In doing so, a bridge is made between the 'pure' ESN and other recurrent network learning methods (see [6,8]). Our aim is to decrease the sensitivity of the ESN architecture to noise in the input signal, by creating and utilizing an internal model of the network dynamics.

We introduce a method of adapting the recurrent layer dynamics without attempting to train the weights directly. Initially a network is generated that fulfils the echo state - liquid state condition. A second network is then trained to predict the next internal state of the system. Prediction occurs in many biological nervous systems e.g. in the cortex [7]. It appears in various contexts, as to overcome signal delays, as well as to reinforce reward. In the present context, prediction is used to generate an alternative version of the original ESN, in which the hidden unit activations represents a model of the networks own behaviour, that creates a source of dynamics that is not a direct function of the input signal, but rather the network's *model* of the combined signal / echo-state

system. The hidden activations of the modified ESN are then a composite of the recursive echo history function (random, untrained), and a trained *generative model* of this function.

After an initial introduction to ESNs, we demonstrate how to perform self-prediction in echo-state networks and how self-prediction is implemented in the recurrent connectivity. Some initial comparisons with the original ESN are presented using synthetic signals

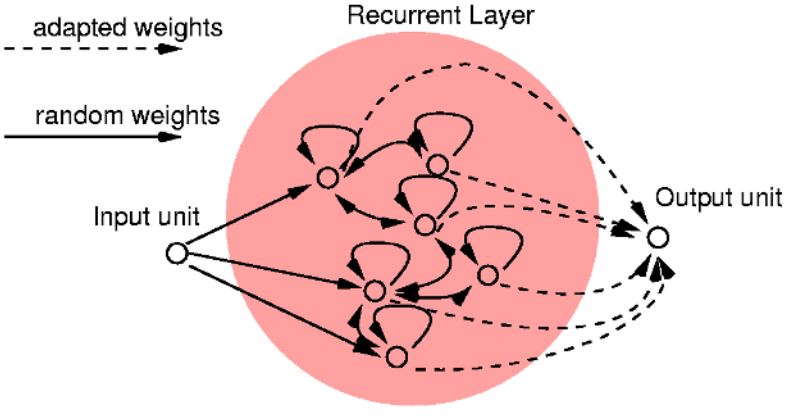

Fig. 1. ESN networks: Principle setup

2 Standard ESNs

The definitions are provided by following [1,3]. The first part consists of a summary of facts proven in [1] that are important for the following considerations. Consider a time-discrete recursive function $\mathbf{x}_{t+1} = F(\mathbf{x}_t, \mathbf{u}_t)$ where $\mathbf{x}_t$ are the internal states and $\mathbf{u}_t$ is some external input sequence, i.e. the stimulus.

The definition of the echo-state condition is the following: Assume an infinite stimulus sequence: $\bar{\mathbf{u}}^\infty = \mathbf{u}_0, \mathbf{u}_1, \ldots$ and two two random initial internal states of the system $\mathbf{x}_0$ and $\mathbf{y}_0$. To both initial states $\mathbf{x}_0$ and $\mathbf{y}_0$ the sequences $\bar{\mathbf{x}}^\infty = \mathbf{x}_0, \mathbf{x}_1, \ldots$ and $\bar{\mathbf{y}}^\infty = \mathbf{y}_0, \mathbf{y}_1, \ldots$ can be assigned.

$$\mathbf{x}_{t+1} = F(\mathbf{x}_t, \mathbf{u}_t) \tag{1}$$

$$\mathbf{y}_{t+1} = F(\mathbf{y}_t, \mathbf{u}_t) \tag{2}$$

The system then fulfills the echo-state condition if, for any time-step t there is a $\delta(\epsilon)$ for all $\epsilon > 0$. for which

$$d(\mathbf{x}_t, \mathbf{y}_t) \leq \epsilon \tag{3}$$

for all $t \geq \delta$. The ESN is designed to fulfil the echo-state condition. The ESN consists of three layers (refer to fig. 1). The first layer is the input. Here the stimulus is presented to the network. The subsequent hidden layer is a recurrently connected. Final layer is the output layer, which is trained to reproduce the target output. The network dynamics is defined for discrete timesteps t.

Equation of the dynamics of the hidden layer are

$$\mathbf{x}_{\mathsf{lin},t+1} = \mathbf{W}\mathbf{x}_t + \mathbf{w}^{in}\mathbf{u}_t \tag{4}$$

$$\mathbf{x}_{t+1} = \mathsf{tanh}\left(\mathbf{x}_{\mathsf{lin},t+1}\right) \tag{5}$$

$$\mathbf{y}_t = \mathbf{w}^{out}\mathbf{x}_t \tag{6}$$

where the vectors $\mathbf{u}_t, \mathbf{x}_t, \mathbf{y}_t$, are the input and the neurons of the hidden layer and output layer respectively, and $\mathbf{w}^{in}, \mathbf{W}, \mathbf{w}^{out}$ are the matrices of the respective synaptic weight factors. Note that for the output neurons linear connectivity is used.[1]

As mentioned above learning is restricted to the connectivity between the hidden and the output layer $\mathbf{w}^{out}$. All other connections are chosen random. In order to fulfil the echo-state condition the connectivity matrix $\mathbf{W}$ of the weights of the hidden layer should meet the following requirements:

1. Necessary is that the real part biggest eigenvalue of $\mathbf{W}$ is smaller than 1.
2. Sufficient is that the biggest singular value of $\mathbf{W}$ is smaller than one.

For all matrices $\mathbf{w}^{out}$ that fulfil requirement 1, but do not fulfil requirement 2 no general rule is known whether or not the network meets the echo-state condition. All real valued matrices can easily transformed to matrices that fulfil the first or both requirements by multiplying and appropriate scalar prefactor to the matrix.

The convergence of an ESN goes as:

$$d(\mathbf{x}_t, \mathbf{y}_t) \leq a\lambda^t, \tag{7}$$

where $\lambda < 1$ and a are constants that are determined by the properties of the connectivity matrix $\mathbf{W}$. Thus, the convergence of the echo-state network is exponential. That means also the forgetting of the initial state $\mathbf{x}_0$ also becomes exponential.

The learning in the output layer $\mathbf{w}^{out}$ is done by solving the (for sufficiently long time sequences where the solution is over defined) system of linear equations:

$$\mathbf{w}^{out}\mathbf{x}_t = \mathbf{u}_t^{\mathsf{teach}} \tag{8}$$

for all times t of the teaching period, where there is one equation for each time step t. With a sufficient amount of timesteps the $\mathbf{w}^{out}$ can be calculated by solving these equations.

[1] The nonlinear output layer definition $\mathbf{y}_t = \mathsf{tanh}(\mathbf{w}^{\mathsf{out}}\mathbf{x}_t)$ is the common alternative to Eqn. (6).

In the offline version of the ESN, Eqn. (8) can be solved by inverting the matrix of which row vectors consist of $\mathbf{x}_t$. This can be done using singular value decomposition (SVD). In addition, online learning can be done with a recursive least squares (RLS) learning algorithm [3].

3 Using Self-Prediction

In the following, not only the hidden-to-output connections $\mathbf{u}_t^{\mathsf{teach}}$ must be learned but also the linear response of the neurons $\mathbf{x}_{\mathsf{lin},t+1}$ Eqn. 4) in the *next* time step. Thus, the teaching signal is

$$\{\mathbf{u}_t^{\mathsf{teach}}\ ,\ \mathbf{x}_{\mathsf{lin},t+1}\}. \tag{9}$$

As in Eqn. (8), the output weights can be calculated by solving a system of linear equations

$$\mathbf{w}_1^{out}\mathbf{x}_t = \mathbf{u}_t^{\mathsf{teach}} \tag{10}$$

$$\mathbf{w}_2^{out}\mathbf{x}_t = \mathbf{x}_{\mathsf{lin},t+1} \tag{11}$$

for all times t of the teaching period (except the initial transient period).

In one first investigation the quality of the resulting prediction giving a periodic input signal was tested. Online learning as outlined in [3] was compared with the offline learning In the case of offline learning the error of the self-prediction is by several orders of magnitude better than the error from RLS.

The self-prediction was implemented into the modified ESN in the following way. The update rule of the recurrent layer is now (cf. Eqns. (4-5))

$$\mathbf{x}_{\mathsf{lin},t+1} = \left((1-\alpha)\mathbf{W} + \alpha\mathbf{w}_2^{out}\right)\mathbf{x}_t + (1-\alpha)\mathbf{w}^{in}\mathbf{u}_t \tag{12}$$

$$\mathbf{x}_{t+1} = \mathsf{tanh}\left(\mathbf{x}_{\mathsf{lin},t+1}\right) \tag{13}$$

where α is constant parameter. If the quality of the prediction is sufficient, $\mathbf{w}_2^{out}\mathbf{x}_t$ and $\mathbf{W}\mathbf{x}_t + \mathbf{w}^{in}\mathbf{u}_t$ should be near to each other. In these situations the dynamics of the network remains almost unaffected by the value of α. In situations where prediction is poor, α is expected to determine the degree to which the dynamics are modulated by the internal model of the system, versus the actual observed data.

If α is equal to 1, the network becomes independent from the input. Of course, the present context, this makes only sense in the case of periodic or at least quasi-periodic output signal. In this case the network becomes an autonomous pattern generator. A value of α being 0 results in the modified network reverting to the original ESN. Other values of α result in some mixture of the two. The dynamics of the system are identical in both the initial network and the network after implementing the prediction. From a formal point of view merely the weights of the network are changed. Thus one can write Eqn. (12) as:

$$\mathbf{x}_{\mathsf{lin},t+1} = \mathbf{W}_{\mathsf{new}}\mathbf{x}_t + \mathbf{w}_{\mathsf{new}}^{in}\mathbf{u}_t \tag{14}$$

where $\mathbf{W}_{\mathsf{new}} = (1-\alpha)\mathbf{W} + \alpha\mathbf{w}_2^{out}$ and $\mathbf{w}_{\mathsf{new}}^{in} = (1-\alpha)\mathbf{w}^{in}$. In this way the equation is formal equvalent to the equation for the ESN (cf. Eqn. (4)), except that $\mathbf{W}$ is replaced by $\mathbf{W}_{\mathsf{new}}$ and $\mathbf{w}^{in}$ by $\mathbf{w}_{\mathsf{new}}^{in}$ respectively. For sufficiently high values of α the echo-state condition is no longer fulfilled, in particular if α is equal to 1.

4 Simulation Details

The aim of the analysis was to compare the performance of the self-predicting network with that of the original ESN. By adjusting the α parameter, we are able to test the effect of adding self-prediction to the normal echo state dynamics. An α parameter of zero corresponds to 'pure' echo state dynamics, an α parameter of 1 corresponds to dynamics governed completely by the self-prediction model. Of interest is determining the value of α that corresponds to best network performance in the presence of noisy input.

We used one input neuron, 80 neurons in the hidden layer and one output neuron. For all simulations presented here the connectivity matrices $\mathbf{w}^{in}$, $\mathbf{W}$, $\mathbf{w}^{out}$ were initialized with equally distributed random values from -0.5 to 0.5 and fully connected. The matrix $\mathbf{W}$ was a random orthonormal matrix, that was scaled to 0.8. Both RLS and offline verison of the network were tested. However, due to the poor prediction quality of the RLS prediction only the offline version produced usable results.

A training period of 2000 iterations was used. The target function be produced by the network was $sin^3(0.24t)$, given an input function $sin(0.24t)$. White noise was added to the input signal at 15% of the amplitude of the signal. ESNs require a brief initialization period in order to stabilize the recurrent dynamics. The first 150 iterations were used to initialize the network and were not included in the optimization procedure. At beginning of both the transient and the test period the state vector $\mathbf{x}_0$ was initialized with weak noise.

5 Results and Discussion

Figure 2 displays the output of the trained network with respect to to the target signal for various levels of α. From figure 2, it can be seen that the unmodified ESN is most sensitive to noise in the input signal. As expected, the case of $\alpha = 1.0$ results in output that well reflects the the structure of the input signal, since the network dynamics have been optimized to reflect the dynamics of the target signal. However, since this network receives no input it can no longer synchronize with the target signal. Best performance was observed for quite high values of α, corresponding to a network with dynamics that are a mixture of both echo states (that are a function of the input history) and self prediction states (that are a function of network's model of the input history).

Figure 1 displays the mean square error relative to signal amplitude for varying parameters of α. As α increases the performance of the network up to approximately a value of 0.97, at which point the performance decreases sharply.

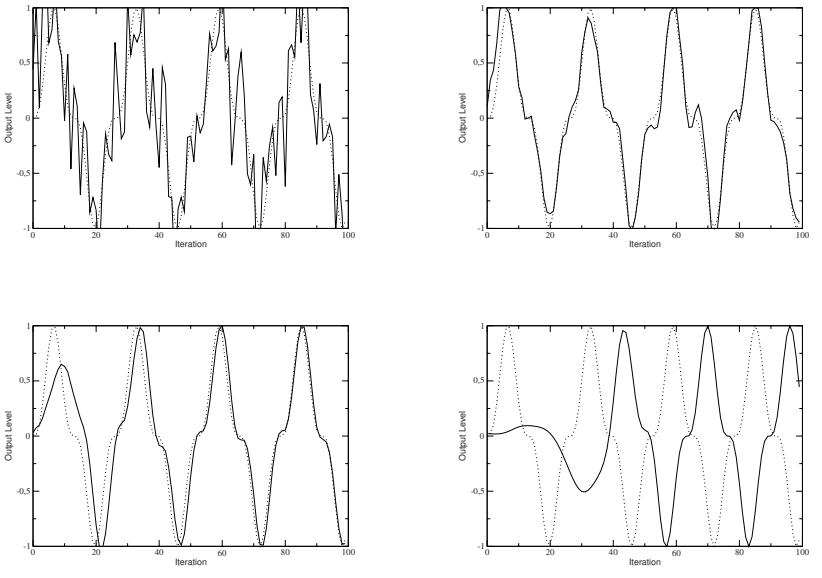

Fig. 2. Effect of varying α for the modified ESN operating on input data with 0.015% relative error. Solid line shows network output, dashed line shows the target output signal synchronous to the network input. The trained pattern is $sin^3(0.24t)$, the input signal is $sin(0.24t)$. Top left $\alpha = 0$ (standard ESN), top right $\alpha = 0.8$, bottom left $\alpha = 0.97$, bottom right $\alpha = 1.0$.

Thus, network with dynamics primarily governed by the network model, with minimal correction provided by the input signal.

The present study represents an initial investigation of the efficacy of improving the performance of ESNs by adding a self prediction module to normal echo state dynamics. Preliminary results indicate that the inclusion of self prediction may improve the performance of an ESN performing a function mapping task in the presence of additive noise. We emphasize that the present results should be taken as conclusive. However, the inclusion of self prediction modules may represent an interesting direction for extending the basic ESN architecture.

In principle, self prediction in recurrent neural networks is proven paradigm for learning and modeling signals. In the present approach, a self-prediction module, which is optimized using standard feed-forward methods, is used as a proxy for the recurrent weights, avoiding difficulties associated with other methods of training hidden weights in a recurrent network. As a next step, the architecture must be tested with non-stationary signals. A related problem is that the present prediction is related to a single pattern. Preliminary investigations indicate that the present architecture, which utilizes linear prediction, may be inherently un-

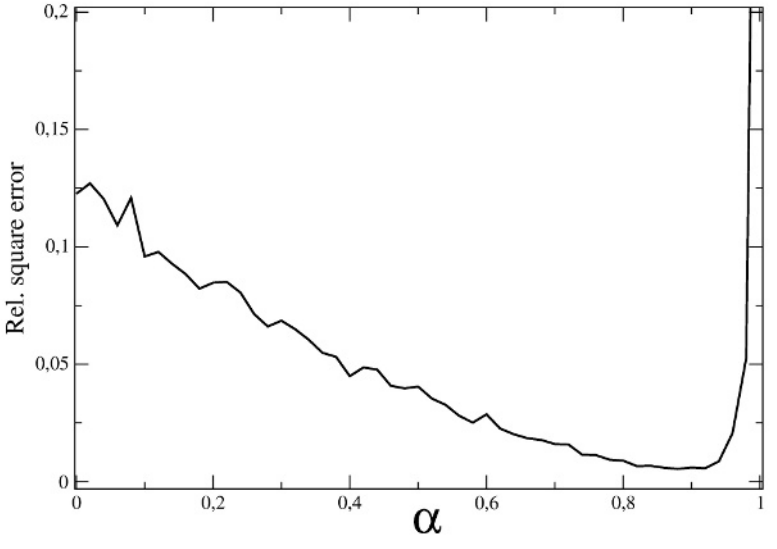

Fig. 3. Network error to a noisy input signal: The network error depends on the value of α for high values of α the network error increases again due to the lacking synchronization of the output signal to the input signal.

stable when attempting to store several multiple patterns. Modifications of the current architecture have shown promising results in simultaneously learning multiple functions, with changes to the learning method, and the connectivity matrix leading to improved results.

Finally it is to mention that Jaeger [1] suggested a different setup for pattern generator using ESN networks. The capabilities of Jaeger's pattern generator are equivalent to the functionality of our system in the case of α being 1.0. However, something like a tuning of the α-parameter is not possible there.

Acknowledgments. Thanks to H. Jaeger, J. M. Herrmann, S. Shiry-Ghidary and J. Schmidhuber for very helpful hints and discussions. Also thanks to T. Christaller for his support.

References

1. H. Jaeger, The 'echo state' approach to analysing and training recurrent neural networks. GMD Report 148, GMD German National Research Institute for Computer Science (2001).
http://www.gmd.de/People/Herbert.Jaeger/Publications.html

2. H. Jaeger, Short term memory in echo state networks. GMD Report 152, GMD German National Research Insitute for Computer Science (2001). http://www.gmd.de/People/Herbert.Jaeger/Publications.html
3. H. Jaeger, Adaptive nonlinear system identification with echo state networks. Proc. of NIPS 2002, AA14 (2003).
4. W. Maas, T. Natschläger, and H. Markram, Real-time computing without stable states: A new framework on for neural computation based on perturbations. NeuroCOLT Technical Report NC-TR-01-113 (2001).
5. B. Farhang-Boroujeny, Adaptive Filters, Wiley , 1999.
6. PJ. Werbos, Backpropagation through time: what it does and how to do it. Proceedings of the IEEE 78(10), 1550-1560 (1990).
7. SA. Huettel,PB. Mack, G. McCarthy, Perceiving patterns in random series: dynamic processing of sequence in the prefrontal cortex, Nature Neurosci. 5:5,pp 485-490 (2002).
8. F.A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: continual prediction with LSTM. Neural Computation, 12(10):2451-2471 (2000)
9. J. Elman. Finding Structure in Time. Cognitive Science, 14(2):179-211 (1990).
10. Y. Bengio. Neural Networks for Speech and Sequence Recognition. International Thomson Publishing Inc., 1996.
11. W. Kadous. Temporal Classification: Extending the Classification Paradigm to Multivariate Time Series Dissertation, School of Computer Science and Engineering, University of New South Wales, 2002.