ESANN'2007 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 25-27 April 2007, d-side publi., ISBN 2-930307-07-2.

Several ways to solve the MSO problem

J. J. Steil - Bielefeld University - Neuroinformatics Group
P.O.-Box 10 01 31, D-33501 Bielefeld - Germany

Abstract. The so called MSO-problem, – a simple superposition of two or more sinusoidal waves –, has recently been discussed as a benchmark problem for reservoir computing and was shown to be not learnable by standard echo state regression. However, we show that are at least three simple ways to learn the MSO signal by introducing a time window on the input, by changing the network time step to match the sampling rate of the signal, and by reservoir adaptation. The latter approach is based on an universal principle to implement a sparsity constraint on the network activity patterns which improves spatio-temporal encoding in the network.

1 Introduction

Reservoir computing is a new approach to signal processing with recurrent networks by implementing a kernel-like idea for time dependent signals. In its original form introduced by Jaeger [1] for standard sigmoid neurons, the idea is to drive a fixed recurrent network by input, regard the high dimensional network state as time dependent feature vector coding for the driving input, and then analytically compute a linear output function by linear regression, see Fig. 1. Though it has been shown that difficult benchmark tasks in time series prediction can be solved with this approach, the power of this method obviously depends on the properties of the fixed recurrent network, the reservoir. As of today, there are no systematic methods to quantify the information processing capacities of nonlinear recurrent networks. Therefore approaches and networks are often compared by applying them to certain benchmark tasks. In this context, it has been argued that the so called MSO problem is difficult to solve with the standard ESN approach.

The MSO problem is the task to learn a simple superposition of sin waves

$$y(k) = \sin(0.2k) + \sin(0.311k).$$

Recently in [2], it was shown that an evolutionary evolved reservoir network can recursively predict this combination of sinusoidal signals. However, evolutionary optimization in itself is a complex technique, which dispenses with one of the main ingredients of the reservoir approach – its simplicity. A further approach has successfully solved the problem by utilizing decoupled multiple reservoirs connected by weak inhibition [3]. This approach externalizes the problem to recognize the different base frequencies and leaves the question unresolved, which kind of signals can be learned within a single reservoir. Both approaches discussed measure the performance for the period immediately following the start of recursive mode. However, for standard ESN regression, the experience shows that many networks diverge very quickly, which is possible because after recursively connecting there is an unbounded linear subloop present in the network.

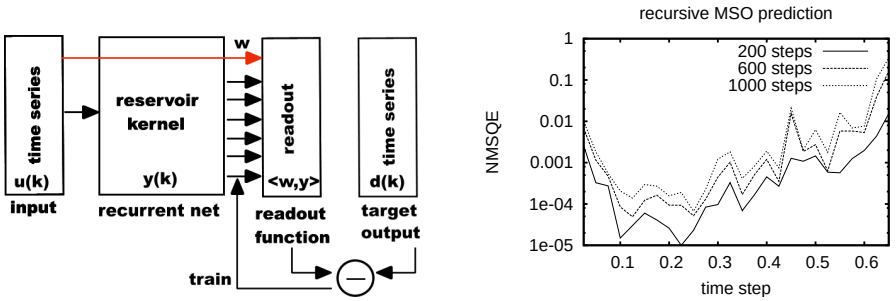


Figure 1: a) The echo state network with nonlinear reservoir and readout function $< \mathbf{w}, \mathbf{y} >$. b) NMSQE (log-scale) for recursive prediction of 1000 time steps vs. network time step Δk.

Even worse (H. Jaeger, personal communication), the networks performing well initially in most cases diverge in the long term. We show a solution to this problem in Section 6.

2 Recurrent reservoir dynamics

In the following, we consider the recurrent reservoir dynamics

$$\mathbf{x}(k+\Delta k) = \mathbf{x}(k) + \Delta k \left[-\mathbf{x}(k) + \mathsf{W}_{res}\mathbf{f}(\mathbf{x}(k)) + \mathsf{W}_u\mathbf{u}(k)\right], \tag{1}$$

where $x_i, i = 1, \ldots, N$ are the neural activations, $\mathsf{W}_{res} \in \mathbb{R}^{N\times N}$ is the reservoir weight matrix, $\mathsf{W}_u \in \mathbb{R}^{N\times R}$ the input weight matrix, and k is the discretized time with time step Δk. Let $\mathbf{u}(k) = (u_1(k), \ldots, u_R(k))^T$ the R-dimensional vector of inputs. Throughout the paper we assume that $\mathbf{y} = \mathbf{f}(\mathbf{x})$ is the vector of neuron activities obtained by applying parameterized Fermi functions component wise to the vector $\mathbf{x}$ as

$$y_i = f_i(x_i, a_i, b_i) = 1/\left(1 + \exp^{(-a_i x_i - b_i)}\right). \tag{2}$$

In the following sections, we follow the standard procedure to train an echo state network. All networks are generated with 10 internal recurrent connections per node on average, i.e. 10% connectivity for a 100 neuron network, and 50% input connectivity in W_u. All weights are initialized with uniform distribution from the interval $[-0.02, +0.02]$. Networks subject to standard ESN regression are rescaled to have 0.8 spectral radius, networks using reservoir adaptation are not rescaled, i.e. have initially very small weight matrix eigenvalues. We first iterate the network driven by input for 1000 steps, then record the network states $\mathbf{y}(k)$ for the next 1000 steps together with the target output $d(k)$, which in this case is a one-step-ahead prediction of the MSO input signal. Then we perform a linear regression for the output weights $\mathbf{w}_{out}$ such that the network output is $\hat{d}(k) =<$

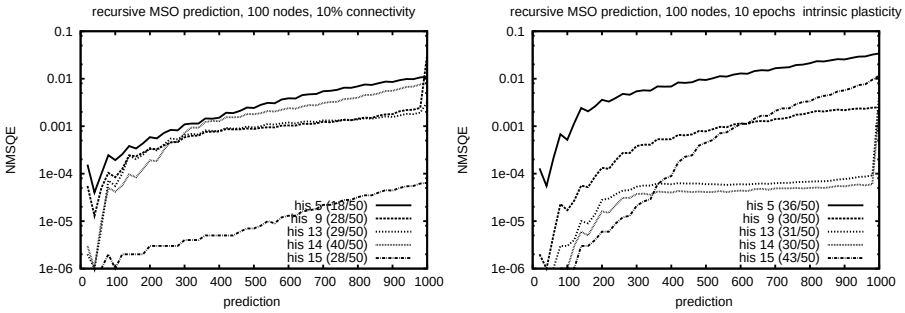


Figure 2: Mean NMSQE for 50 runs of recursive prediction. Errors are computed every 20 steps of recursive prediction after ESN regression for 100 neuron reservoir network. Left: direct regression, right: regression after 10 epochs (= 100000 steps) of intrinsic plasticity training. Number of converged runs in brackets.

$\mathbf{w}_{out}, \mathbf{y}(k) >$. Finally, we reconnect the output to the input $u_1(k+1) \leftarrow \hat{d}(k)$ to let the network run recursively on the predicted target. Performance is measured as NMSQE for T test time steps by computing $\left(1/T \sum_{k=1}^{T} e(k)^2\right) / var(d)$, where $e(k) = \hat{d}(k) - d(k)$ is the error with respect to the target signal $d(k)$ and $var(d)$ is the variance of the target.

3 The MSO-problem and the time step

For continuous signals with discrete sampling, it is often crucial to adjust the network time-step Δk to the sampling rate of the problem. Fig. 1 shows that for the MSO problem time steps between 0.05 and 0.5 allow the network to succeed in recursive prediction very easily. Thus in principle the MSO problem can be solved straightforward and in a very simple manner within the standard ESN framework and there is no need for advanced techniques. However, setting the time step equal to one makes the problem harder and it remains interesting to investigate whether the network could internally develop a dynamics providing the necessary coding even under these circumstances.

4 The MSO-problem and the time history window

In the next experiment, we supply an increasingly long time history window as input to the network, i.e. $u_1(k) = d(k), u_2(k) = d(k-1), \ldots, u_R(k) = d(k-R+1)$. When recursively connecting after training, the predicted $d(k+1)$ is feed back to the first input u_r and the rest of the history is shifted by one time step. Thereby the network runs fully autonomous in recursive mode after R time steps. In confirmation of the earlier results of Jaeger, for small history windows $R < 5$ the networks almost always ($> 95\%$) diverge in recursive mode and if they

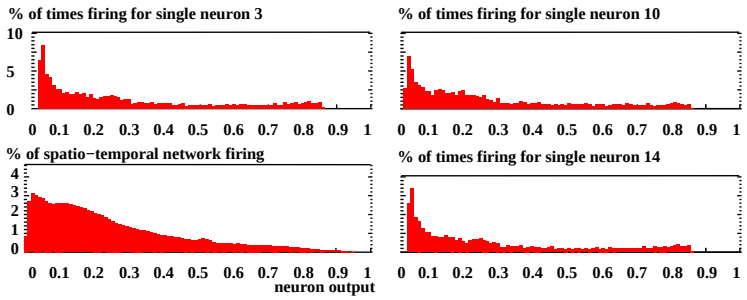


Figure 3: Output distribution of randomly selected single neurons after the IP learning and spatio-temporal average of all neurons in the network histogramed over one epoch of 4000 steps.

do not diverge, in most cases oscillate without implementing the MSO dynamics. But for history length R larger than five the network sometimes diverges, but if it does not, it is able to learn the MSO task. Fig. 2 shows mean NMSQE for up to 1000 steps of recursive prediction for different length of the history window. We conducted 50 runs for every history length and give the number of runs not diverging within the prediction period together with the mean prediction.

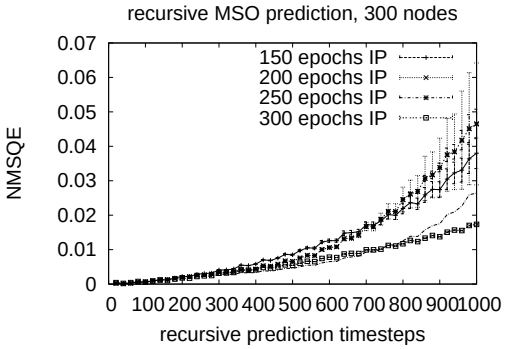
5 The MSO-problem and intrinsic plasticity

Our third approach to solve the MSO problem is based on adapting the reservoir to the input signal to first optimize the temporal coding in the network before computing the readout function. For adaptation we use a learning rule developed in [4], which has been introduced to reservoir computing in [5]. It is motivated from the capability of biological neurons to change their excitability according to the stimulus distribution the neuron is exposed to. It has been argued that the underlying principle may be that exponential output distributions maximize information transmission under the constraints imposed by maintaining a constant mean firing rate. In [4], an online adaption rule to achieve this goal of maximizing information transmission has been developed. It adjusts the parameters a, b of the Fermi function in order to minimize the Kullback-Leibler divergence of the actual output distribution of the neuron with respect to an exponential distribution with desired mean activity level μ. It adapts parameters a, b for the neuron with learning rate ζ in time step k as ([4]):

$$\Delta b(k) = \zeta \left(1 - \left(2 + \frac{1}{\mu} \right) y(k) + \frac{1}{\mu} y(k)^2 \right), \tag{3}$$

$$\Delta a(k) = \zeta \left(\frac{1}{a} + x(k) - \left(2 + \frac{1}{\mu} \right) x(k) y(k) + \frac{1}{\mu} x(k) y(k)^2 \right). \tag{4}$$

Fig. 3 illustrates the effect of IP learning by displaying a randomly chosen



#res	#stable	#MSO
50	16/50	10/50
100	41/50	10/50
150	50/50	44/50
200	50/50	43/50
250	50/50	42/50
300	50/50	46/50

Figure 4: Left: Recursive prediction errors computed every 20 steps after pre-adapting 300 neuron reservoir networks with intrinsic plasticity. Means and variances are given over 20 network initializations, no short term divergence occurs. Right: Long term stability and convergence to the desired MSO attractor. The table displays #res = number of nodes in the reservoir, #stable = number of networks converging to an attractor, #MSO = number of networks converging to the MSO attractor according to the measure given in the text.

neuron's output distribution after application of the IP rule to a network of 200 neurons driven by the MSO signal. The network is iterated in epochs of 4300 steps while the IP learning rule is applied in each step with target average activity $\mu = 0.2$ and learning rate $\zeta = 0.001$. We use identical inputs from the MSO time series and zero initial states for each epoch. In epoch 500 we record the network outputs $\mathbf{y}(k)$ after a relaxation phase of 300 steps, and plot the histogram of activities of a randomly chosen neuron in Fig. 3 in 100 bins of width 0.01. It is clearly visible that the neuron qualitatively approximates an exponential output characteristic as well as the overall network output firing pattern shown in Fig. 3, bottom. This activation pattern implies a sparse coding, because most of the neuron have small output for most of the time. We now persue the idea to pre-optimize the reservoir with IP adaptation and to apply a standard echo state regression to the adapted reservoir.

Fig. 2 shows that application of IP pre-training can improve the performance on an short time-scale of only 10000 training points (10 epochs of 1000 points). However, still a considerable number of networks approximate the target network not well enough and diverge in recursive mode. On the other hand this is not surprising, because the statistical IP adaptation can exhibit its full power only on a longer time scale. Fig. 4 shows the NMSQE for up to 1000 steps of recursive prediction for different lengths of long term IP training. The effect of sparse coding here is that the network even with a single input can reliably learn the MSO task, the better, the longer we pre-train it with the IP rule. Further, there occur no instabilities in recursive mode.

6 Long term stability

All our experiments confirm that the recursively connected networks tend to diverge when running for long times (we tested up to 10^7 steps). The reason seems to be that initially the network trajectory stays close to the MSO oscillator, while later slight drift inside the system tends to amplify. However, a combination of intrinsic plasticity, time step 0.3, and the addition of noise in the frequency domain helps to solve this problem, i.e. we train with $\sin(r_1 0.2k) + \sin(r_2 0.2k)$, where $r_{1,2}$ are uniformly drawn from $\pm 10^{-7}$. Then, we pre-train with IP for $1.2 * 10^6$ steps and apply the regression. Every 8192 time steps we compute the power spectrum of the discrete Fourier transform and compute the angle to the spectrum of the MSO signal as similarity measure. We regard the network as converged, if the variation in this measure is below a small threshold for 10 consecutive times ($\approx$ 82000 steps) and consider the network as implementing the MSO attractor if the angle is below $0.01 rad$ in the high dimensional space of the frequency spectra vectors. Fig.3 shows that with this training networks converge and most of the time even to the desired attractor.

7 Conclusion

While measuring the encoding quality for a given input signal remains an open and important problem, we believe that the MSO problem is not the best benchmark to show what is difficult for echo state regression. The three approaches we showed to solve the MSO problem complement the more complex methods to tackle this problem introduced earlier. We conclude that the MSO problem is too simple to serve as hard benchmark. On the other hand, the present investigation shows that there exists a rich variety of modifications to the standard ESN network approach, which can significantly change the performance. The most instructive result from the present work is that intrinsic plasticity learning is capable to change the reservoir dynamics based on a very general and unspecific statistical unsupervised learning principle, which is used to optimize coding in the reservoir in a signal specific way. It can also be used to stabilize the long term dynamics to converge to the desired attractor.

References

[1] H. Jaeger. Adaptive nonlinear system identification with echo state networks. In *NIPS*, pages 593–600, 2002.

[2] J. Schmidhuber D. Wierstra, F. Gomez. Modeling systems with internal state using evolino. In *Proc. GECCO*, pages 1795–1802, 2005.

[3] Xue and S. Haykin. Decoupled ESNs with lateral inhibition. NIPS Workshop on ESN, 2006. To appear in Neural Networks special issue on ESN and Liquid states.

[4] J. Triesch. A gradient rule for the plasticity of a neuron's intrinsic excitability. In *Proc. ICANN*, number 3695 in LNCS, pages 65–79, 2005.

[5] J. J. Steil. Online reservoir adaptation by intrinsic plasticity for backpropagation-decorrelation and echo state learning. *Neural Networks*, 2007.