A comparative study on feedforward and recurrent neural networks in time series prediction using gradient descent learning

Manfred Hallas, Georg Dorffner
Austrian Research Institute for Artificial Intelligence
Schottengasse 3, A-1010 Vienna, Austria
and
Dept.of Medical Cybernetics and Artificial Intelligence, Univ. of Vienna
email: georg@ai.univie.ac.at

Abstract

This paper reports about a comparative study on several linear and nonlinear feedforward and recurrent neural networks trained on artificially created time series. This has lead to interesting empirical results about the capabilities of these network models trained with a gradient descent learning procedure. Several of the time series were generated by some of the neural network models, in order to test whether they could learn to predict a time series which they could theoretically perfectly model. The results show that recurrent networks do not seem to be able to do so under the given conditions. They also show that a simple feedforward network (a nonlinear autoregressive model) significantly performs best for most of the nonlinear time series. These empirical results can be taken as valuable hints with respect to the practical application of neural networks in prediction tasks.

1 Introduction

Feedforward and recurrent multilayer perceptrons (e.g. of the so-called Jordan and Elman type – see [Bengio, 1995] or [Dorffner, 1996] for an overview) are popular neural networks for complex time series processing tasks, such as forecasting in financial applications [Weigend *et al.*, 1991; Refenes *et al.*, 1994; Trippi & Turban, 1993]. They are applied due to their strength in handling non-linear functional dependencies between past time series values and the estimate of the value to be forecast. Training is usually done by minimizing the summed squared error criterion. While efficient optimization methods like conjugent gradient or quasi-Newton methods are well-known, simple gradient descent procedures ("backpropagation learning") are still the most widely used learning rules.

Relatively little is known about the practical applicabilities and limits of the above-mentioned neural network types for a given time series (a notable exception is a similar study by [Horne & Giles, 1995], who, incidentally, also focus on gradient descent learning). Therefore, few authors justify their use of a particular network other than by presenting comparative results. Among the important questions, still largely unanswered in practical terms, are the following:

- When does a nonlinear method (in particular, a non- or semi-parametric method like neural networks) actually achieve better results than classical linear methods (e.g. linear ARMA or state-space models)?
- When should recurrent networks be applied, as opposed to feedforward neural networks with time window input?
- Can gradient descent learning exploit the theoretical capacity of feedforward or recurrent multilayer perceptrons in forecasting?
- Can a neural network trained by gradient descent achieve optimal performance on a time series produced by the network itself as the generator?

Especially the last question, which – when succesfully answered – should shed some light on the capabilities and limits of the models, lead us to design the comparative study reported here. We were interested in exploring whether different neural network types could optimally identify a model behind a time series if the network could theoretically implement that model perfectly. By also letting each network type identify the model behind the time series produced by all the other networks, and that of additional artificial series, one obtains empirical results about the capabilities of the networks faced with non- linear time series under controled conditions. Thus, we expected partial answers to the other questions raised above, as well.

2 Methods and data

2.1 Neural networks

The following eight networks, including 2 linear ones (roughly corresponding to classical autoregressive models) were compared. To leave as many parameters constant as possible, all networks were sought to have approximately the same number of degrees-of-freedom (weights), namely between 220 and 250.

- SLAR: a single layer perceptron with an input window of size 220 (identical to a linear AR(220) autoregressive model)

- LAR: a two-layer perceptron with an input window of size 20 and a hidden layer of 10 linear hidden units (a multilayer version of a AR(20) model)
- NAR: a two-layer perceptron with an input window of size 20 and 10 nonlinear hidden units with sigmoid activation functions (a general nonlinear NAR(20) autoregressive model)
- JORDAN: a Jordan-type recurrent perceptron (based on the one proposed in [Jordan, 1986]) with an input of size 1, a sigmoidal hidden layer of size 55, and a feedback from the output unit to an extra context layer with a self-recurrent loop.
- JORDAN2: a Jordan-type recurrent perceptron with an input of size 1, two sigmoidal hidden layers of size 13 each, and a feedback from the output unit to an extra state layer with a self- recurrent loop.
- ELMAN: an Elman-type recurrent perceptron (based on the one proposed in [Elman, 1990]) with an input of size 1, a sigmoidal hidden layer of size 14, and a feedback from the hidden layer to an additional context layer of size 14 with self-recurrent loops.
- ELMAN2: an extended Elman-type recurrent perceptron with an input of size 1, two sigmoidal hidden layers of size 8 each, a feedback from the output to an additional state layer of size 1 (with feedforward connections to the output layer), a feedback from the second hidden layer to an additional context layer of size 8 (feedforward connections to the second hidden layer), and a feedback from the first hidden layer to an additional context layer of size 8 (feedforward connections to the first hiddenlayer). All state and context layers had self-recurrent loops.
- MRN: a multi-recurrent network [Ulbricht, 1994], which is a mixed Jordan- and Elman-type recurrent perceptron, with an input of size 1, a sigmoidal hidden layer of size 10, a feedback from the output to a first state layer with weight 0.75, and a self-recurrent loop with weight 0.25, an additional feedback from the output to a second state layer with weight 0.25 and a self-recurrent loop with weight 0.75, a feedback from the hidden layer to a first context layer of size 10 with weights 0.75 and self-recurrent loops with weights 0.25, and a feedback from the hidden layer to a second context layer of size 10 with weights 0.25 and self-recurrent loops with weights 0.75. (For a justification of this type of network, see [Ulbricht, 1994].)

Except for MRN, all feedback connections were one-to-one connections with weights equal to 1, and self-recurrent loops possessed a weight equal to 0.6.

Compared to traditional approaches to time series analyses, these networks are rather large, especially the linear AR models. On one hand we considered the number of degrees-of-freedom as an important parameter to be held constant, mainly because vis a vis the number of training data a fair comparison appears to be possible. On the other hand this means that the nonlinear neural networks are rather on the non-paramteric side of model estimation and could therefore theoretically exploit their full potential of nonlinear approximation.

2.2 Training sequences

The following training sequences were artificially created:

- Sin-AM: an amplitude-modulated sine wave
- Chaos: a chaotic time series using the logistic equation $x(t+1) = ax(t)(1-x(t))$ with $x_0 = 0.2027$ and $a = 4$
- QLAR: a quasi-linear time series generated by the NAR network in its quasi-linear range (using small intial weights)
- NAR: a non-linear time series generated by the NAR network (with random weight initialization)
- three more non-linear time series generated by th JORDAN2, ELMAN2 and MRN networks, respectively, with random weight initialization.

From each model, ten non-overlapping time series of length 2000 were created.

2.3 Training of the networks

Each of the eight network model was trained on each of the seven time series in 20 different runs, with a learning rate equal to 0.2 (except for the two linear networks, where it was set to 0.01). The 20 runs consisted of two different weight initializations and ten different training sets. Prior to these runs, ten different runs with different weight initializations were performed on each network. The initialization that yielded the best result, and the initialization that was closest to average performance were selected as the two initializations in the actual 20-time cross-evaluation runs. The training sets consisted of the first 1000 data points in each of the non-overlapping time series created (thus outnumbering the number of weights by about a factor of 4). Training was halted by an early-stopping procedure by using a random 10 % of the training patterns used as a validation set (or, alternatively, after 10000 epochs). All weight initializations were in the range of [-1,1], except for the linear networks, where it was [-0.1,0.1].

After training, 500 of the remaining 1000 data points of each time series were used for testing prediction performance with a time-lag of 1. The mean squared error over those 500 points was recorded. For the recurrent networks, 100 data points were used to build up the context and state layers before evaluating each prediction.

3 Results

Tables 1 through 3 summarize the results of the 20 runs for each network and each time series. Means and standard deviations of the mean squared error are depicted. The best results for each sequence are

SLAR	μ_{MSE}	**8.4e-06**	**9.91e-06**
	σ_{MSE}	4.12e-06	7.88e-06
LAR	μ_{MSE}	0.000151	**9.1e-06**
	σ_{MSE}	3.56e-06	6.1e-07
NAR	μ_{MSE}	**2.65e-05**	**9.29e-06**
	σ_{MSE}	4.05e-06	1.64e-06
JORDAN	μ_{MSE}	0.000186	0.0128
	σ_{MSE}	2.06e-05	0.00337
JORDAN2	μ_{MSE}	0.000155	0.0073
	σ_{MSE}	1.96e-05	0.00137
ELMAN	μ_{MSE}	0.00172	0.0811
	σ_{MSE}	0.00564	0.157
ELMAN2	μ_{MSE}	0.00181	0.0263
	σ_{MSE}	0.00268	0.0159
MRN	μ_{MSE}	0.00466	0.0163
	σ_{MSE}	0.0115	0.0233

Table 1: Results from 20 training runs on two time series (amplitude modulated sine wave and quasi-linear AR process; mean μ and standard deviation σ of the MSE are depicted)

SLAR	μ_{MSE}	0.856	0.0913
	σ_{MSE}	1.03	0.0734
LAR	μ_{MSE}	0.127	0.0256
	σ_{MSE}	0.00412	0.00628
NAR	μ_{MSE}	0.00201	**0.000996**
	σ_{MSE}	0.000259	0.000816
JORDAN	μ_{MSE}	**0.000277**	0.0766
	σ_{MSE}	4.99e-05	0.00629
JORDAN2	μ_{MSE}	**6.37e-05**	0.071
	σ_{MSE}	2.35e-05	0.00583
ELMAN	μ_{MSE}	0.0514	0.0768
	σ_{MSE}	0.084	0.045
ELMAN2	μ_{MSE}	0.0156	0.0821
	σ_{MSE}	0.0299	0.003
MRN	μ_{MSE}	0.00211	0.0841
	σ_{MSE}	0.00266	0.0647

Table 2: Results from 20 training runs on two time series (chaotic logisitic equation and nonlinear AR process; mean μ and standard deviation σ of the MSE are depicted)

highlighted (bold face script). Each of these results was significantly better than the remaining network results ($p < 0.001$ in a performed t-test).

4 Discussion

Several important conclusions can be drawn from this comparative study:

- Whenever a time series can be sufficently described by a linear model (such is the case for the SinAM and the quasi-linear NAR series), among the nonlinear networks only the feedforward NAR model comes close to optimal performance, while all recurrent neural networks result in significantly sub-optimal predictions. Therefore, one can conclude that those recurrent networks do not easily contain the linear case and thus should only be applied when there is sufficient evidence for a particular nonlinear process.
- Except for the feedforward NAR model, none of the nonlinear neural networks was able to optimally estimate its own time series, even though they could theoretically perfectly implement the underlying model. This points to additional limits of the recurrent networks.
- In most nonlinear cases, the simple feedforward NAR model yielded significantly better results than the other nonlinear neural networks. This stands in interesting contrast to the findings in [Horne & Giles, 1995], where recurrent networks consistently outperformed feedforward networks in the tasks of finite state machine induction and nonlinear system identification.
- In the case of the chaotic time series, the variants of Jordan-type networks yielded the best results (especially JORDAN2). This points to a certain effectiveness of this kind of network in temporal processes whose underlying model has a first order iterative generator. This is somewhat surprising, since the feedfoward NAR network apparently more directly implements such a first-order process but nevertheless achieves significantly worse results.

Net		JORDAN2	ELMAN2	MRN
SLAR	μ_{MSE}	0.0644	0.181	0.661
	σ_{MSE}	0.0626	0.195	0.805
LAR	μ_{MSE}	0.00761	0.00803	0.0107
	σ_{MSE}	0.000852	0.00201	0.00192
NAR	μ_{MSE}	**5.08e-05**	**0.00508**	**0.00037**
	σ_{MSE}	3.24e-05	0.00101	0.000115
JORDAN	μ_{MSE}	0.048	0.0134	0.00516
	σ_{MSE}	0.0568	0.0114	0.000373
JORDAN2	μ_{MSE}	0.0158	0.0113	0.00494
	σ_{MSE}	0.032	0.00917	0.000588
ELMAN	μ_{MSE}	0.0288	0.0413	0.00454
	σ_{MSE}	0.0674	0.07	0.00201
ELMAN2	μ_{MSE}	0.00747	0.0238	0.00308
	σ_{MSE}	0.0155	0.0211	0.00248
MRN	μ_{MSE}	0.0326	0.0875	0.00386
	σ_{MSE}	0.0637	0.169	0.00185

Table 3: Results from 20 training runs on the remaining three time series (generated by the networks JORDAN2, ELMAN2, and MRN; mean μ and standard deviation σ of the MSE are depicted)

Of course, all results have to be seen against the background of using gradient descent as a learning procedure. Thus, all perspicuous limits are obviously limits of this kind of learning (compare [Horne & Giles, 1995], p.700).

One can conclude from these results that for unknown underlying nonlinear characteristics of a time series, the feedforward NAR model appears to be most likely to lead to satisfying results. In [Dorffner, 1996] it has been shown that Jordan-type neural networks resemble a version of a nonlinear autoregressive moving average (NARMA) model (see also [Connor *et al.*, 1992]). In our case, both Jordan-type networks implement NARMA(1,1) models (with the extension of self-recurrent loops at the state layer). Thus it is no surprise that they have difficulties in modeling sequences of higher order, such as the NAR sequence. Output feedback, even with self-recurrent loops, does not replace a time window.

In the case of the Elman-type networks one at first sight would expect better performance in modeling higher-order sequences. (Note that in literature, hidden layer feedback is frequently depicted as incorporating potentially unlimited memory about the past.) In [Dorffner, 1996] it was shown that an Elman-type network is a slight modification (but not a general nonlinear extension) of the classical state-space model of time series analysis. Thus it inherently bears limits with respect to handling general nonlinear time series models. Given the generally poor results of the ELMAN network this seems to be confirmed. The ELMAN2 network comes more closely to a nonlinear state-space model given its additional hidden layer (by which it implements a nonlinear functional dependency between the state layer and the output). However, even though better in some cases, the performance is equally poor pointing to a deeper reason for the network's limitations.

One can also conclude that with respect to recurrent neural networks, more efficient learning procedures are needed. This has been pointed out before by [Bengio, 1995]. Backpropagation in time, second-order or global optimization procedures like simulated annealing or genetic learning might lead to a more optimal exploitation of the networks' capabilities.

Another comment is in place. All generated time series used in this study are deterministic ones with no noise added, whereas real-world problem almost exclusively deal with stochastic series. Therefore one must be careful in transferring our results to real forecasting applications.

5 Conclusion and future work

The results from the comparative study point to serious limitations of recurrent neural networks applied to nonlinear prediction tasks, when gradient descent learning is applied. An interesting exception are Jordan-type networks applied to series generated by a first-order iterative process, even if it exhibits chaotic behavior. Although a study of this kind leaves many questions open (and partly stands in contrast to earlier studies [Horne & Giles, 1995]), it can be used as guidelines for future work.

Our ultimate goal is to identify optimal neural networks to model real-world financial time series. For that, the next step is to compare the characteristics of such series (through so-called stylized facts) with the characteristics of the series generated here under controled conditions. Additionally, a major emphasis will be put on adding different noise models to create truly stochastic time series. Together with theoretical analyses, we envision such empirical studies to lead to further insights into the complex behavior of sequential processes.

Acknowledgments

This work was done as part of the project SFB 10 "Adaptive Models in Economics and Management Science", funded by the Austrian Fund for Scientific Research, FWF. We thank Kurt Hornik, Helmut Strasser, Manfred Deistler, and Adrian Trapletti for helpful comments. The Austrian Research Institute for Artificial Intelligence is supported by the Austrian Federal Ministry of Science and Transport.

References

[Bengio, 1995] Bengio Y.: *Neural Networks for Speech and Sequence Recognition*, Thomson, London, 1995.

[Connor *et al.*, 1992] Connor J., Atlas L.E., Martin D.R.: Recurrent Networks and NARMA Modeling, in Moody J.E., et al.(eds.), *Neural Information Processing Systems 4*, Morgan Kaufmann, San Mateo, CA, pp.301-308, 1992.

[Dorffner, 1996] Dorffner G.: Neural networks for time series processing, Neural Network World, 6(4)447-468, 1996.

[Elman, 1990] Elman J.L.: Finding Structure in Time, Cognitive Science, 2(14)179-212, 1990.

[Horne & Giles, 1995] Horne B.G., Giles C.L.: An Experimental Comparison of Recurrent Neural Networks, in Tesauro G., et al.(eds.), Advances in Neural Information Processing System 7, MIT Press, Cambridge, MA, pp.697-704, 1995.

[Jordan, 1986] Jordan M.I.: Serial Order: A Parallel Distributed Processing Approach, ICS- UCSD, Report No. 8604, 1986.

[Refenes *et al.*, 1994] Refenes A.N., Zapranis A., Francis G.: Stock Performance Modeling Using Neural Networks: A Comparative Study with Regression Models, *Neural Networks*, 7(2), 375-388, 1994.

[Trippi & Turban, 1993] Trippi R.R., Turban E.(eds.): *Neural Networks in Finance and Investing*, Probus, Chicago, 1993.

[Ulbricht, 1994] Ulbricht C.: Multi-Recurrent Networks for Traffic Forecasting, in *Proceedings of the Twelfth National Conference on Artificial Intelligence*, AAAI Press/MIT Press, Cambridge, MA, pp.883-888, 1994.

[Weigend *et al.*, 1991]
Weigend A.S., Rumelhart D.E., Huberman B.A.: Generalization by Weight- Elimination with Application to Forecasting, in Lippmann R.P., et al.(eds.), *Advances in Neural Information Processing 3*, Morgan Kaufmann, San Mateo, CA, 1991.