Comparison of Echo State Networks with Simple Recurrent Networks and Variable-Length Markov Models on Symbolic Sequences *

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 more real-valued time series modeling tasks and performed exceptionally well. Also using ESNs for processing symbolic sequences seems to be attractive. In this work we experimentally support the claim that the state space of ESN is organized according to the Markovian architectural bias principles when processing symbolic sequences. We compare performance of ESNs with connectionist models explicitly using Markovian architectural bias property, with variable length Markov models and with recurrent neural networks trained by advanced training algorithms. Moreover we show that the number of reservoir units plays a similar role as the number of contexts in variable length Markov models.

1 Introduction

Echo state network (ESN) [1, 2] 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. Under certain conditions RNN state is a function of finite history of inputs presented to the network - the state is the "echo" of the input history. ESN training procedure is a simple adjustment of output weights to fit training data. ESNs were successfully applied in some sequence modeling tasks and performed exceptionally well [3, 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 for example 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.

Many commonly used real-world data with a time structure can be expressed as a sequence of symbols from finite alphabet - symbolic time series. Since their emergence the neural networks were applied to symbolic time series analysis. Especially popular is to use connectionist models for processing of complex language structures. Other

* This work was supported by the grants APVT-20-030204 and VG-1/4053/07

works study what kind of dynamical behavior has to be acquired by RNNs to solve particular tasks such as processing strings of context-free languages, where counting mechanism is needed [7, 8]. Some researchers realized that even in an untrained randomly initialized recurrent network considerable amount of clustering is present. This was first explained in [9] and correspondence to a class of variable length Markov models was shown in [10].

Some attempts were made to process symbolic time series using ESNs with interesting results. ESNs were trained to stochastic symbolic sequences and a short English text in [2] and ESNs were compared with other approaches including Elman's SRN trained by simple BP algorithm in [11]. Promising resulting performance was achieved, superior to the SRN. In both works results of ESNs weren't compared with RNNs trained by advanced algorithms.

2 Methods

2.1 Recurrent Neural Networks

RNNs were successfully applied in many real-life applications where processing time-dependent information was necessary. Unlike feedforward neural networks, units in RNNs are fed by activities from previous time steps through recurrent connections. In this way contextual information can be kept in units' activities, enabling RNNs to process time series.

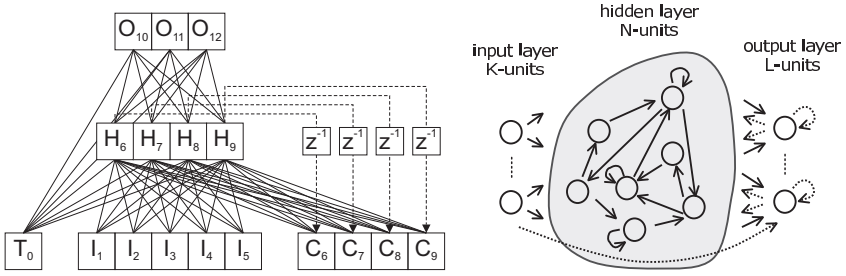

Fig. 1. (a) Elman's SRN and (b) Jaeger's ESN architectures.

Elman's simple recurrent network (SRN) proposed in [12] is probably the most widely used RNN architecture. Context layer keeps activities of hidden (recurrent) layer from previous time step. Input layer together with context layer form extended input to the hidden layer. Elman's SRN composed of 5 input, 4 hidden a 3 output units is shown in Fig. 1a.

Common algorithms usually used for RNN training are based on gradient minimization of the output error. Backpropagation through time (BPTT) [13, 14] consists of unfolding a recurrent network in time and applying the well-known backpropagation

algorithm directly. Another gradient descent approach, where estimates of derivatives needed for evaluating error gradient are calculated in every time step in forward manner, is the real-time recurrent learning (RTRL) [15, 14]. Probably the most successful training algorithms are based on the Kalman filtration (KF) [16]. The standard KF can be applied to a linear system with Gaussian noise. A nonlinear system such as RNNs with sigmoidal units can be handled by extended KF (EKF). In EKF, linearization around current working point is performed and then standard KF is applied. In case of RNNs, algorithms similar to BPTT or RTRL can be used for linearization. Methods based on the Kalman filtration outperform common gradient-based algorithms in terms of in terms of robustness, stability, final performance and convergence, but their computational requirements are usually much higher.

2.2 Echo State Networks

Echo state networks represent a new powerful approach in recurrent neural network research [1, 3]. Instead of difficult learning process, ESNs are based on the property of untrained randomly initialized RNN to reflect history of seen inputs - here referred to as "echo state property". ESN can be considered as a SRN with a large and sparsely interconnected recurrent layer - "reservoir" of complex contractive dynamics. Output units are used to extract interesting features from this dynamics, thus only network's output connections are modified during learning process. A significant advantage of this approach is that computationally effective linear regression algorithms can be used for adjusting output weights.

The network includes input, hidden and output "classical" sigmoid units (Fig. 1b). The reservoir of the ESN dynamics is represented by hidden layer with partially connected hidden units. Main and essential condition for successful using of the ESNs is the "echo state" property of their state space. The network state is required to be an "echo" of the input history. If this condition is met, only network output weights adaptation is sufficient to obtain RNN with high performance. However, for large and rich reservoir of dynamics, hundreds of hidden units are needed. When $\mathbf{u}(t)$ is an input vector at time step t, activations of internal units are updated according to

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

where f is the internal 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 output units are calculated as

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

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 $x_i(t) = e_i(u(t), u(t-1), \ldots)$ [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.

2.3 Variable Length Markov Models

As pointed out in [10], the state space of RNNs initialized with small weights is organized in Markovian way prior to any training. To assess, what has been actually learnt during the training process it is always necessary to compare performance of the trained RNNs with Markov models.

Fixed order Markov model is based on the assumption that the probability of symbol occurrence depends only on the finite number of m previous symbols. In the case of the predictions task all possible substrings of length m are maintained by the model. Substrings are prediction contexts of the model and for every prediction context the table of the next symbol probabilities is associated. Hence the memory requirements grow exponentially with the model order m.

To solve some limitations of fixed order Markov models variable length Markov models (VLMMs) were proposed [17, 18]. The construction of the VLMM is a more complex task, contexts of various lengths are allowed. The probability of the context is estimated from the training sequence and rare and other unimportant contexts are not included in the model.

2.4 Models Using Architectural Bias Property

Several connectionist models directly using Markovian organization [10] of the RNN's state space were suggested. Activities of recurrent neurons in an recurrent neural network initialized with small weights are grouped in clusters [9]. The structure of clusters reflects the history of inputs presented to the network. This behavior has led to the idea described in [19] where prediction models called neural prediction machine (NPM) and fractal prediction machine (FPM) were suggested. Both use Markovian dynamics of untrained recurrent network. In FPM, activation function of recurrent units is linear and weights are set deterministically in order to create well-defined state space dynamics. In NPM, activation functions are nonlinear and weights are randomly initialized to small values as in regular RNN. Instead of using classical output layer readout mechanism, NPM and FPM use prediction model that is created by extracting clusters from the network state space. Each cluster corresponds to different prediction context with the next symbol probabilities.

More precisely, symbol presented to the network drives the network to some state (activities on hidden units). The state belongs to some cluster and the context corresponding to this cluster is used for the prediction. The context's next symbol probabilities are estimated during training process by relating the number of times that the corresponding cluster is encountered and the given next symbol is observed.

Described prediction model can be created also using activities on recurrent units of the trained RNN. In this article we will refer to this model as NPM built over the trained RNN. RNN training process is computationally demanding and should be justified. More complex dynamics than simple fixed point attractor-based one should be acquired. Hence prediction context of NPM built over the trained RNN usually do not follow Markovian architectural bias principles.

3 Experiments

3.1 Datasets

We present experiments with two symbolic sequences. The first one was created by symbolization of activations of laser in chaotic regime and chaotic nature of the original "real-world" sequence is also present in the symbolic sequence. The second dataset contains words generated by simple context free grammar. The structure and the recursion depths are fully controlled by the designer [10] in this case.

The Laser dataset was obtained by quantizing activity changes of laser in chaotic regime, where relatively predictable subsequences are followed by hardly predictable events. The original real-valued time series was composed of 10000 differences between the successive activations of a real laser. The series was quantized into a symbolic sequence over four symbols corresponding to low and high positive/negative laser activity change. The first 8000 symbols are used as the training set and the remaining 2000 symbols form the test data set [20].

Deep recursion data set is composed of strings of context-free language L_G. Its generating grammar is $G = (\{R\}, \{a, b, A, B\}, P, R)$, where R is the single non-terminal symbol that is also the starting symbol, and a, b, A, B are terminal symbols. The set of production rules P is composed of three simple rules: $R \to aRb | R \to ARB | R \to e$ where e is the empty string. This language is in [7] called palindrome language. The training and testing data sets consist of 1000 randomly generated concatenated strings. No end-of-string symbol was used. Shorter strings were more frequent in the training set than the longer ones. The total length of the training set was 6156 symbols and the length of the testing set was 6190 symbols.

3.2 Performance of ESNs

In this section the predictive performance of ESNs is evaluated on the two datasets. Symbols were encoded using one-hot-encoding, i.e. all input or target activities were set to 0, except the one corresponding to given symbol, which was set to 1. Predictive performance was evaluated by means of a normalized negative log-likelihood (NNL) calculated over the test symbol sequence $S = s_1 s_2 \dots s_T$ from time step $t = 1$ to T as

$$NNL = -\frac{1}{T} \sum_{t=1}^{T} \log_{|A|} p(t), \tag{3}$$

where the base of the logarithm is the alphabet size, and the $p(t)$ is the probability of predicting symbol s_t in the time step t. For NNL error calculation the activities on output units were first adjusted to chosen minimal activity $o_{\min}$ set to 0.001 in this experiment, then the output probability $p(t)$ for NNL calculation could be evaluated:

$$\hat{o}_i(t) = \begin{cases} o_{\min} \text{ if } o_i(t) < o_{\min} \\ o_i(t) \text{ otherwise} \end{cases}, p(t) = \frac{\hat{o}_i(t)}{\sum_j \hat{o}_j(t)}, \tag{4}$$

where $o_i(t)$ is the activity of the output unit i in time t.

ESNs with hidden unit count varying from 1 to 1000 were trained using recursive least squares algorithm. Symbols were encoded using one-hot-encoding, i.e. all input or target activities were set to 0, except the one corresponding to given symbol, which was set to 1. Hidden units had sigmoidal activation function and linear activation function was used for output units. Reservoir weight matrix was rescaled to different values of spectral radius from 0.01 to 5. The probability of creating input and threshold connections was set to 1.0 in all experiments and input weights were initialized from interval $(-0.5, 0.5)$. Probability of creating recurrent weights was 1.0 for smaller reservoirs and 0.01 for larger reservoirs. It was found that this parameter has very small influence to the ESN performance (but significantly affects simulation time).

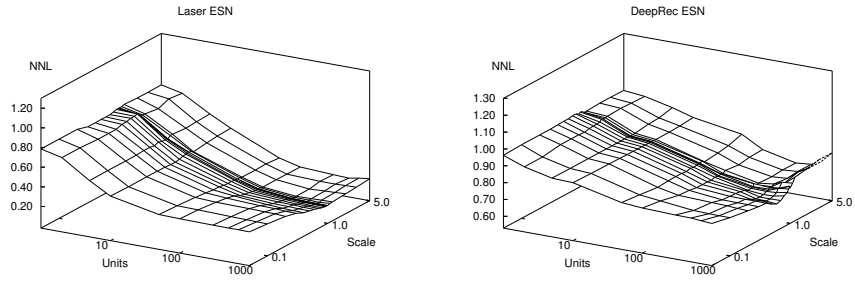

Fig. 2. Performance of ESNs with different unit counts and different values of spectral radius.

As can be seen from the plots in Fig. 2 results are very similar for wide range of spectral radii. More units in the reservoir results in better prediction. To better asses the importance of reservoir parameterization several intervals for reservoir weights' values were tested starting from $(-0.01, 0.01)$ and ending by $(-1.0, 1.0)$. Also several probabilities of recurrent weight existence were tested from 0.01 to 1.00. Of course no spectral radius rescaling was done in this type of experiments. Various probabilities and intervals for reservoir weights did not influence the resulting performance a lot, hence no figures are shown in the paper. For small weight range and low probability the information stored in the reservoir faded too quickly so the differentiation between points corresponding to long contexts was not possible. This effect was more prominent for the Laser dataset where storing long contexts is necessary to achieve good prediction and hence resulting performance of ESN with weight range of $(-0.1, 0.1)$ and probability 0.01 are worse for higher unit count in the reservoir. Also high probability and wide interval are not appropriate. In this case ESN units are working in the saturated part of its working range very closed to 0.0 and 1.0. Differentiating between states is difficult and hence for example for weight range of $(-1.0, 1.0)$ and probability of 1.0 and higher unit count such as 300 unsatisfactory performance is achieved. For higher values of unit count the performance is worse since the saturation is higher, not because of the overtraining. But for wide range of combinations of these parameters very similar results were obtained. This observation is in accordance with the principles of Marko-

vian architectural bias. Fractal organization of the recurrent neural network state space is scale free and as long as the state space dynamics remains contractive the clusters reflecting the history of the symbols presented to the network are still present.

3.3 Recurrent Neural Networks

In the experiments of this section we show how classical RNNs represented by Elman's SRN perform on the two datasets. Gradient descent approaches such as backpropagation through time or real-time recurrent learning algorithms are widely used by researchers working with symbolic sequences. In some cases even simple backpropagation algorithm is used to RNN adaptation [11, 21]. On the other hand, techniques based on the Kalman filtration used for recurrent neural network training on real-valued time series have already shown their potential.

We provide results for standard gradient descent training techniques represented by simple backpropagation and backpropagation through time algorithms and for extended Kalman filter adopted for RNN training with derivatives calculated by BPTT-like algorithm. 10 training epochs (one epoch – one presentation of the training set) for EKF were sufficient for reaching the steady state, no significant NNL improvement has occurred after 10 epochs in any experiment. 100 training epochs for BP and BPTT were done. We improved training by using scheduled learning rate. We used linearly decreasing learning rate in predefined intervals. But no improvements made the training as stable and fast as the EKF training (taking into account the number of epochs). Although it may seem that further training (beyond 100 epochs) may result in better performance, most of BPTT runs started to diverge in higher epochs.

For NNL calculation the value $p(t)$ is obtained by normalizing activities of output units and choosing normalized output activity corresponding to the symbol s_t. NNL performance was evaluated on the test dataset every 1000 training steps.

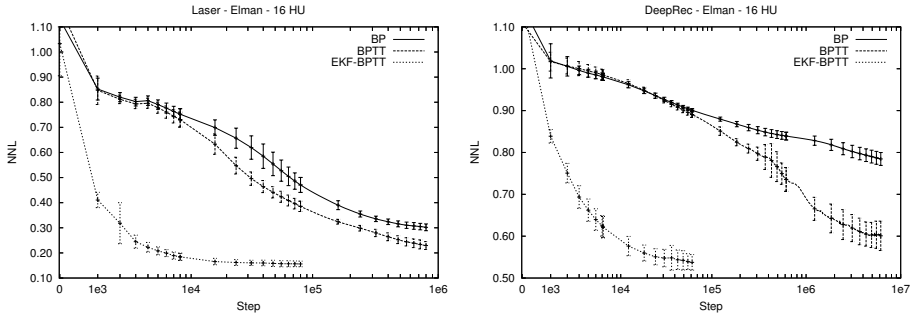

Fig. 3. Performance of Elman's SRN with 16 hidden units trained by BP, BPTT and EKF-BPTT training algorithms.

We present mean and standard deviations of 10 simulations for Elman's SRN with 16 hidden units in Fig. 3. Unsatisfactory simulations with significantly low performance

were thrown away. This was usually the case of BP and BPTT algorithms that seems to be much more influenced by initial weight setting and are sensitive to get stuck in local minima or to diverge in later training phase. Generally for all architectures, NNL performances of RNNs trained by EKF are better. It seems to be possible to train RNN by BPTT to have similar performance as the networks trained by EKF, but it usually required much more overhead (i.e. choosing only few from many simulations, more than one thousand of training epochs, extensive experimenting with learning and momentum rates). Also EKF approach to training RNNs on symbolic sequences shows higher robustness and better resulting performance. BP algorithm is too week to give satisfactory results. NNL performances are significantly worse in comparing with algorithms that take into account the recurrent nature of network architectures.

Extended Kalman filter shows much faster convergence in terms of number of epochs and resulting NNLs are better. Standard deviation of results obtained by BPTT algorithm are high revealing BPTT's sensitivity to initial weight setting and to get stuck to local minimum. Although computationally more difficult, extended Kalman filter approach to training recurrent networks on symbolic sequences shows higher robustness and better resulting performance.

3.4 Markov Models and Methods Explicitly Using Architectural Bias Property

To assess what has been actually learnt by the recurrent network it is interesting to compare the network performance with Markov models and models directly using architectural bias of RNNs. Fractal prediction machines were trained for the next symbol prediction task on the two datasets. Also neural prediction machines built over the untrained SRN and also SRN trained by EKF-BPTT with 16 hidden units are tested and results are compared with VLMMs and ESNs. Prediction contexts for all prediction machines (FPMs and NPMs) were identified using K-means clustering with cluster count varying from 1 to 1000.

10 simulation were performed and mean and standard deviation are shown in plots. Neural prediction machines uses dynamics of different networks from previous experiments for each simulation. For fractal prediction machines internal dynamics is deterministic. Initial clusters are set randomly by K-menas clustering hence slightly different results are obtained for each simulation also for FPMs. VLMMs were constructed with the number of context smoothly varying from 1 context (corresponding to the empty string) to 1000 contexts. Results are shown in Fig. 4.

The first observation is that the ESNs have the same performance as other models using architectural bias properties and that the number of hidden units plays very similar role as the number of contexts of FPMs and NPMs built over untrained SRN. For Laser dataset incrementing the number of units resulted in prediction improvement. For Deep recursion dataset and higher units count (unit counts above 300) ESN model is overtrained exactly as other models. ESN uses linear readout mechanism and the more dimensional state space we have the better hyper-plane can be found with respect to the desired output.

Training can improve the state space organization so better NPM models can be extracted form the recurrent part of the SRN. For Laser dataset the improvement is present for models with small number of context. For higher values of context count

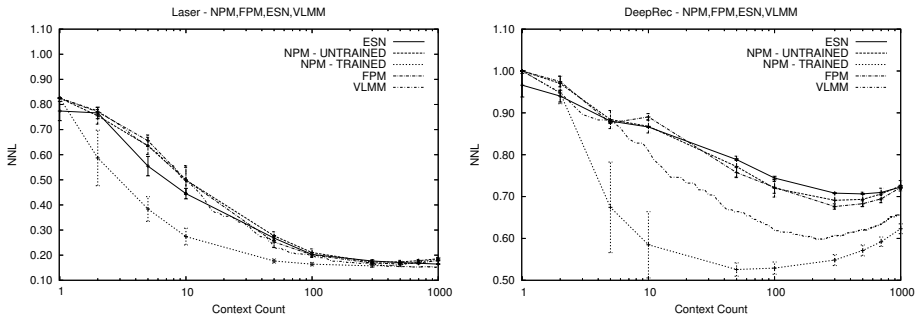

Fig. 4. Performance of ESNs compared to FPMs, NPMs and VLMMs.

the performance of the NPMs created over the trained SRN is the same as for other models. But the carefully performed training process using advanced training algorithm significantly improves the performance of NPMs built over the trained SRN for the Deep recursion dataset.

Significantly better results were achieved by VLMMs on Deep recursion dataset than with ESN or methods based on Markovian architectural bias properties. The reason is in a way how a VLMM tree is constructed. VLMM is built incrementally and the context importance is influenced by Kullback-Leibler divergence between the next symbol distributions of the context and its parent context, context being extended by symbol concatenation. No such mechanism that would take into account the next symbol distribution of the context exists in models based on Markovian architectural bias. Prediction contexts correspond to the clusters that are identified by quantizing the state space. Clustering is based on vectors occurrences (or probabilities) and the distances between vectors. To prove this idea experiments with modified VLMM were performed. Node importance was given by its probability and its length and this type of VLMMs achieved results almost identical to methods based on Markovian architectural bias properties.

4 Conclusion

Extensive simulation using ESNs were made and ESNs were compared with the carefully trained SRNs, with other connectionist models using Markovian architectural bias property and with VLMMs. Multiple parameters for ESN reservoir initialization were tested and the resulting performance wasn't significantly affected. Correspondence between the number of units in ESN reservoir and the context count of FPM, NPM models and Markov models was shown. According to our results ESNs are not able to beat Markov barrier when processing symbolic time series. Carefully trained RNNs or VLMMs can achieve better results on certain datasets. On the other side computational expensive training process may not be justified on other datasets and models such as ESNs can perform just as well as thoroughly trained RNNs.

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.: Short term memory in echo state networks. Technical Report GMD 152, German National Research Center for Information Technology (2001)
3. 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
4. Jaeger, H., Haas, H.: Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. Science **304**(5667) (2004) 78–80
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. Rodriguez, P.: Simple recurrent networks learn contex-free and contex-sensitive languages by counting. Neural Computation **13** (2001) 2093–2118
8. Bodén, M., Wiles, J.: On learning context free and context sensitive languages. IEEE Transactions on Neural Networks **13**(2) (2002) 491–493
9. Kolen, J.: The origin of clusters in recurrent neural network state space. In: Proceedings from the Sixteenth Annual Conference of the Cognitive Science Society, Hillsdale, NJ: Lawrence Erlbaum Associates (1994) 508–513
10. Tiňo, P., Čerňanský, M., Beňušková, Ľ.: Markovian architectural bias of recurrent neural networks. IEEE Transactions on Neural Networks **15**(1) (2004) 6–15
11. Frank, S.L.: Learn more by training less: Systematicity in sentence processing by recurrent networks. Connection Science, in press (2006)
12. Elman, J.L.: Finding structure in time. Cognitive Science **14**(2) (1990) 179–211
13. Werbos, P.: Backpropagation through time; what it does and how to do it. Proceedings of the IEEE **78** (1990) 1550–1560
14. Williams, R.J., Zipser, D.: Gradient-based learning algorithms for recurrent networks and their computational complexity. In Chauvin, Y., Rumelhart, D.E., eds.: Back-propagation: Theory, Architectures and Applications. Lawrence Erlbaum Publishers, Hillsdale, N.J. (1995) 433–486
15. Williams, R.J., Zipser, D.: A learning algorithm for continually running fully recurrent neural networks. Neural Computation **1** (1989) 270–280
16. Williams, R.J.: Training recurrent networks using the extended Kalman filter. In: Proceedings of International Joint Conference on Neural Networks IJCNN 1992, Baltimore. Volume 4. (1992) 241–246
17. Ron, D., Singer, Y., Tishby, N.: The power of amnesia. Machine Learning **25** (1996) 117–149
18. Machler, M., Bühlmann, P.: Variable length Markov chains: methodology, computing and software. Journal of Computational and Graphical Statistics **13** (2004) 435–455
19. Tiňo, P., Dorffner, G.: Recurrent neural networks with iterated function systems dynamics. In: International ICSC/IFAC Symposium on Neural Computation. (1998)
20. Tiňo, P., Dorffner, G.: Predicting the future of discrete sequences from fractal representations of the past. Machine Learning **45**(2) (2001) 187–218
21. Farkaš, I., Crocker, M.: Recurrent networks and natural language: exploiting self-organization. In: Proceedings of the 28th Cognitive Science Conference, Vancouver, Canada. (2006) 1275–1280