Approaches Based on Markovian Architectural Bias in Recurrent Neural Networks

Matej Makula[1], Michal Čerňanský[1], and Ľubica Beňušková[2]

[1] Faculty of Informatics and Information Technologies,
Slovak University of Technology, Ilkovičova 3,
812 19 Bratislava, Slovakia
{makula, cernans}@fiit.stuba.sk
http://www.fiit.stuba.sk

[2] Institute of Informatics, Comenius University,
Mlynská dolina, 842 48 Bratislava, Slovakia
benus@ii.fmph.uniba.sk

Abstract. Recent studies show that state-space dynamics of randomly initialized recurrent neural network (RNN) has interesting and potentially useful properties even without training. More precisely, when initializing RNN with small weights, recurrent unit activities reflect history of inputs presented to the network according to the Markovian scheme. This property of RNN is called Markovian architectural bias. Our work focuses on various techniques that make use of architectural bias. The first technique is based on the substitution of RNN output layer with prediction model, resulting in capabilities to exploit interesting state representation. The second approach, known as echo state networks (ESNs), is based on large untrained randomly interconnected hidden layer, which serves as reservoir of interesting behavior. We have investigated both approaches and their combination and performed simulations to demonstrate their usefulness.

1 Introduction

The key part of recurrent neural networks (RNNs) performance is encoded in activities of recurrent units (network state) and their variations in time (network dynamics). It is well-known fact that recurrent neural networks have universal approximation capability, although development of desired dynamics in training might be sometimes difficult or even unfeasible task. Classical gradient-based training methods face severe problems such as information latching problem [4]. This problem arises because gradient of the error function tends to vanish in time, thus it is impossible for RNNs to catch the long time dependencies in the input sequence. Several approaches have been developed in order to at least partially overcome this problem [5, 1]. But recent studies show that sometimes instead of complicated RNN weights adaptation, it might be beneficial to leave network dynamics randomly initialized.

It has been known for some time that when RNN is used to process symbolic sequences, activations of recurrent units show considerable amount of information about input sequence prior to training [2, 9, 10]. It was experimentally shown that RNNs initialized with small weights are inherently biased towards Markov models [11, 12]. This phenomenom is refered as Markovian architectural bias of RNNs. When dealing with problems, where this Markovian representation is useful, initial network dynamics can be leaved unchanged and only transformation of state to desired output has to be carried out. This can be performed either by simple neural network layer or by other advanced method, e.g. by prediction model.

Major advantage of this 'unusual' approach is the elimination of the recursive dependencies between weights in adjustment process, i.e. problem when even small 'positive' weight change in one step can have huge impact on network activities in other steps. Thus, instead of complicated training of whole network, only output layer (or prediction model) is adjusted to produce desired output from the inherent 'Markovian' dynamics.

2 Alternative Training Approaches

Typical example of 'classical' approach in recurrent network community is the first-order Elman simple recurrent network (SRN) [3], trained by various gradient-based techniques. Recently a novel approach based on architectural bias called echo state networks (ESNs) has been introduced [6].

Layer interconnections and units used in ESN are mostly equal to SRN architecture[3], but the main difference is in the size of recurrent layer and in the training process. ESNs use large randomly initialized recurrent layer, called dynamical reservoir, to obtain massive response to the input signal. In training, output weights are adjusted to use this response to 'reconstruct' desired output. Essential condition for the ESN approach is that dynamical reservoir must produce meaningful response, i.e. network state must be an 'echo' of input signal. This is achieved by rescaling network weights to small values, which is apparently equivalent to architectural bias condition, i.e. *initialization with 'small' weights*. Thus, ESNs can be viewed as 'architectural bias' based counterparts of classical RNNs.

As it was mentioned earlier, once we skip recurrent weights training, the output weights can be calculated effectively by linear regression. In ESNs it is usually done offline by calculation of pseudoinverse matrix, but also other online algorithms, such as least mean squares (LSM) or recursive least squares (RLS), can be used. It is important to notice that before training suitable memory capacity of ESN can be set up by rescaling weights in dynamical reservoir [7]. More detailed description of ESN approach can be found in [8].

Another widely used technique is to extract finite state representation from trained recurrent networks. By clusterization of recurrent units activities, com-

[3] ESN approach allows also additional input-output and output-hidden layer interconnection, but we did not use them in our experiments.

Table 1. Differences between 'classical' approaches and approaches based on 'architectural bias'

	SRN	NPM	ESN	untrained NPM
approach	classical		based on architectural bias	
output layer	neural network	prediction model	neural network	prediction model
output	symbols and real values	only symbols	symbols and real values	only symbols
hidden layer	small ≈ 10 units	small ≈ 10 units	large ≈ 10^2 units	small ≈ 10 units
network dynamics	adjusted	adjusted	random - Markovian	random - Markovian
adjustment	both network dynamics and state-output mapping	same as SRN + prediction model building	only state-output mapping	only prediction model building
training algorithm	gradient based [14], extended Kalman filter	same as SRN + model adjustment	linear regression	only model adjustment

plex network dynamics can be easily substituted with more useful finite state representation. This kind of state representation is used in simple modification of RNN called neural prediction machine (NPM)[12], where network output layer is replaced with prediction model. NPM seeks state clusters in network state space and associates them with conditional probabilities for the next symbol prediction. Conditional probabilities are determined by counting of *(state cluster, output symbol)* occurrences in training sequence. Because NPM can be also built from an untrained network, this approach can be used to exploit Markovian state representation in untrained network.

Architectures described in previous chapters are summarized in the Table. 1. The first two columns describe architectures where both input-state and state-output mapping is adjusted in training. The approaches based on architectural bias are in the third and fourth column.

3 Experiments & Results

In the following subsections we describe our experiments, data sets, and training parameters. The first experiment is comparison of SRN and ESN approach on problem of periodic sequence prediction task. The second problem is the prediction of symbolic sequence obtained by quantization of chaotic laser activity, where we compared performance of NPMs built on recurrent layer of trained and untrained recurrent network.

3.1 Periodic Sequence Prediction Task

In this experiment we studied how the desired solution is achieved by both classical SRN and novel ESN approach. Original experiment was based on problem of stable generation of periodic sequence by ESN [6]. We simplified original problem to next value prediction task and evaluated SRN and ESN with various numbers of hidden units.

Input sequence was the melody "The House of the Rising Sun" transformed to numerical values to fit interval [-0.05,0.5] (Fig. 1). Multiple melody samples were concatenated together to form the target sequence of period 48.

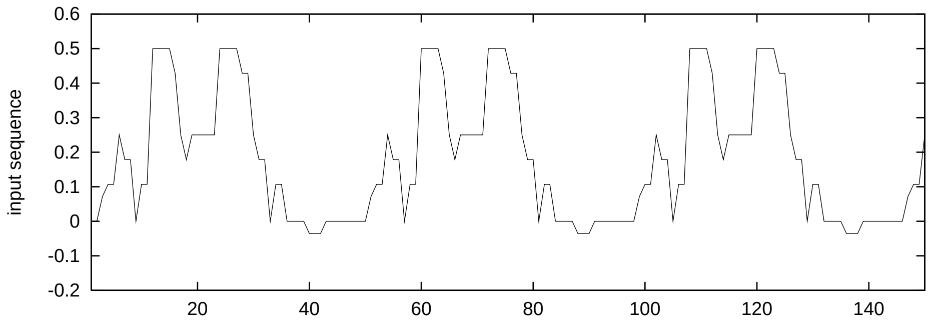

Fig. 1. Input sequence was created by concatenation of multiple samples from melody "The House of the Rising Sun"

ESN input weights were selected randomly from uniform distribution [-2, 2]. Recurrent weights were set to values +0.4, -0.4 and then scaled down to obtain $\lambda_{max} = 0.908$. The connectivity of recurrent layer was sparse with ratio 1.25 %. Since we were not concerned about stable melody generation, we did not incorporate noise into the training data. Both training and testing sequence had 1500 input symbols, where first 500 inputs were used for initialization. Output weights were calculated offline with linear regression.

All weights in SRN were selected randomly from uniform distribution [-0.1,0.1]. Training was performed by extended Kalman filter method (EKF). Error covariance matrix was initialized $\mathbf{P} = 1000 \cdot \mathbf{I}$, measurement noise matrix was set to $\mathbf{R} = 100 \cdot \mathbf{I}$ and output noise matrix was set to $\mathbf{Q} = 10^{-4} \cdot \mathbf{I}$, where $\mathbf{I}$ is the identity matrix. Weights derivatives were calculated by back-propagation through time (BPTT) method [13] with window size 30. Both training and testing sequence had 10000 input symbols, where first 500 inputs were used for initialization.

We use bipolar (*tanh*) activation function for both ESN and SRN units. All values presented in figure Fig. 2 are the averages of mean square error (MSE) with standard deviations from 10 runs of ESN and SRN network training.

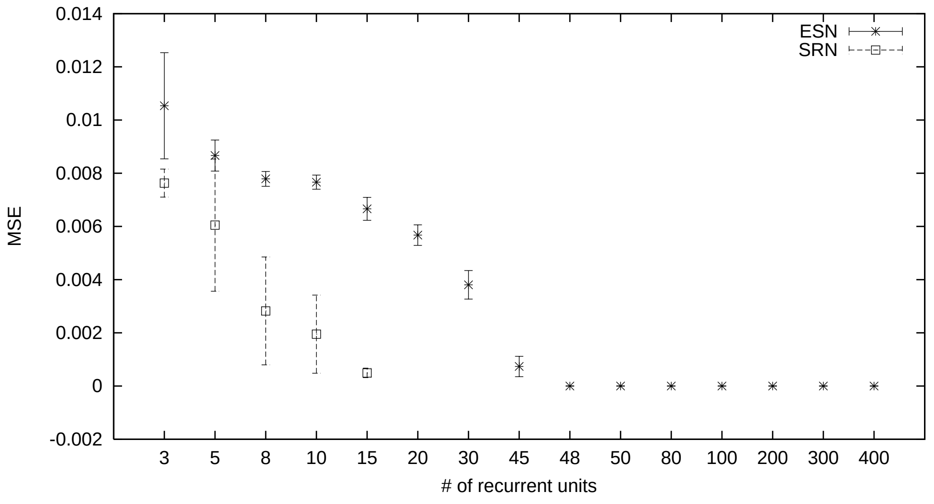

Fig. 2. Mean square error (MSE) for SRN and ESN after training. MSE was calculated for networks with different number of hidden units. We can see that SRN with the same number of hidden units outperforms ESN, although MSE for larger ESNs is almost zero (MSE $\approx 10^{-30}$ - limited by computer decimal number precision)

3.2 Chaotic Sequence Prediction Task

In this experiment we focus on the second approach based on architectural bias, i.e. NPM built on untrained RNN. Performance of NPMs built on untrained network was compared with performance of NPMs built on recurrent layer of trained SRN. We have used networks with various numbers of hidden units.

As a training sequence for this experiment we chose a long symbolic sequence of quantized activity changes of laser in chaotic regime. Data set includes various levels of memory structures, i.e. relatively predictable subsequences followed by global, harder to predict events, that require a deeper memory (Fig 3).

Whole sequence[4], i.e. 10 000 differences Δ_t between two subsequent activations, was quantized into a symbolic stream of four symbols from the alphabet $\{a, b, c, d\}$, corresponding to low / high and positive / negative activity changes. Entire sequence was divided into the training sequence (the first 8000 symbols) and the test sequence (the last 2000 symbols). Symbols were encoded using one-hot-encoding.

As activation function we use unipolar sigmoid $1/(1 + e^x)$. Training of SRN was performed by EKF in 10 epochs. First 50 symbols were used for initialization. Other SRN parameters were the same as in the previous experiment. For NPMs built form untrained network underlying dynamics was produced by dynamical reservoir of ESN initialized identically with the previous experiment.

[4] Taken from http://www-psych.stanford.edu/˜andreas/Time-Series/SantaFe.html

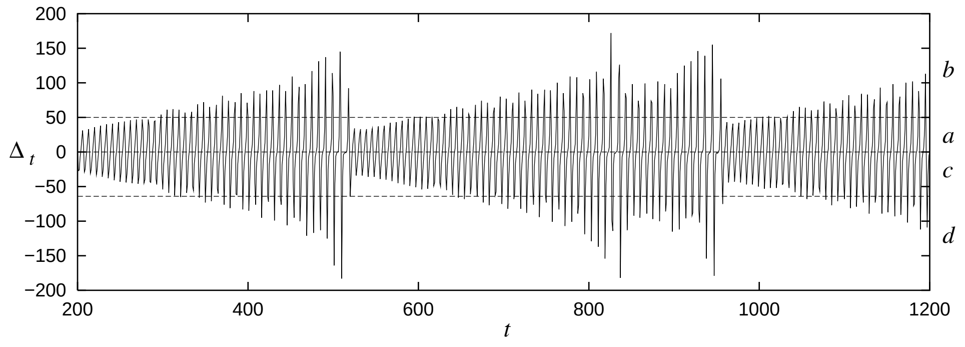

Fig. 3. Example of 1000 differences Δ_t of the laser activations. Dotted horizontal lines corresponds to 10% and 90% sample quantiles. Letters on the right vertical axis indicate quantization into four symbols

Finally, state space of both trained and untrained network was clustered by K-means algorithm with 250 clusters and NPM was built. Performance of NPMs was evaluated by means of the normalized negative log-likelihood (NNL) (Fig. 4). NNL can be viewed as inverse compression ratio, i.e. NNL=0.5 means that sequence can be compressed to half of its original size. Ideal NNL value is 0, NNL = 1 corresponds to random guessing.

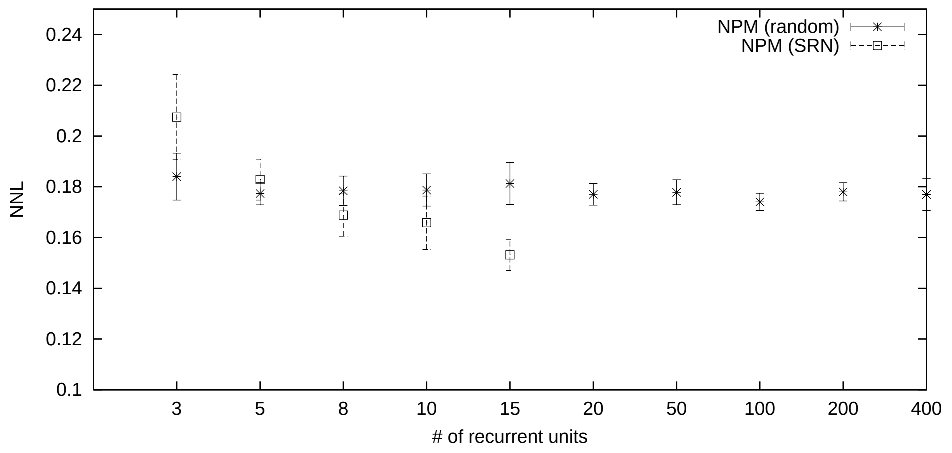

Fig. 4. Normalized negatibe log-likelihood (NNL) for NPMs build from trained SRN and untrained ESN reservoir

4 Discussion

First task is a straightforward demonstration of difference between the classical and novel approach. Once a periodic signal is presented to the untrained RNN, its state moves on periodic orbit in the state space. Classical approach performs training by reshaping state trajectory and simultaneous adaptation of state-output mapping. When sufficient number of recurrent units is used and network has enough state space dimensions to examine, desired dynamics emerges and the next value can be successfully predicted (Fig. 2, for SRN with 15 hidden units MSE $\approx 1.7 \cdot 10^{-4}$).

The way in which ESN produces desired solution is different. Initial network dynamics behave similarly with SRN, i.e. network state jumps between 48 different clusters in the state space. In training, output weights are used to fit desired output from network state. When dimension of state space is small, i.e. small number of recurrent units, network output cannot be precisely reconstructed. Once output weights have enough state space dimension to explore, linear combination of state vector components can produce desired periodic output signal with almost zero precision (MSE $\approx 10^{-30}$). It might seem that great number of units in ESN recurrent layer is not necessary, but in original experiment this additional dimensions were used by ESN to stably reproduce original periodic sequence [6].

Results of our second experiment confirm that the prediction of chaotic laser sequence is rather difficult task due to long time dependency problem. Training of recurrent network by extended Kalman filter slightly improve state representation, which is clearly visible when sufficient number of hidden units is used (Fig. 4). However, NPM built on hidden layer of untrained network can provide comparable results. Interestingly, in this case larger number of hidden units did not help to improve NPM performance.

5 Conclusion

In this paper we presented approaches based on architectural bias. These approaches leave randomly initialized network dynamics unchanged and adjust only state to output transformation. On selected experiments we evaluate their performance and compared obtained results with classical approaches.

Results show that SRN can produce desired solution with smaller number of recurrent units, although it requires much more computationally expensive training. On the other hand, nature of approaches based on architectural bias allows to increase the number of recurrent units without significant increasing of training complexicity. ESN can use this high dimensional state space to fit desired output accurately. In the second experiment simple prediction model - NPM, built on untrained network with 5 recurrent units, produced little worse results than NPM built on much larger network trained by computationally demanding EKF technique.

Acknowledgments

This work was supported by the VEGA grant 1/9046/02.

References

1. Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.
2. M. H. Christiansen and N. Chater. Toward a connectionist model of recursion in human linguistic performance. *Cognitive Science*, 23:417–437, 1999.
3. J. L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.
4. S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In J. Kolen and S. Kremer, editors, *Field Guide to Dynamic Recurrent Networks*, pages 237–243. Wiley-IEEE Press, 2001.
5. S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
6. H. Jaeger. The "echo state" approach to analysing and training recurrent neural networks. Technical Report GMD Report 148, German National Research Center for Information Technology, 2001.
7. H. Jaeger. Short term memory in echo state networks. Technical Report GMD Report 152, German National Research Center for Information Technology, 2001.
8. H. Jaeger. Tutorial on training recurrent neural networks. Available on: `http://www.ais.fraunhofer.de/INDY/herbert/ESNTutorial/`, 2002. release September / October, 2002.
9. J. F. Kolen. The origin of clusters in recurrent neural network state space. In *Proceedings from the Sixteenth Annual Conference of the Cognitive Science Society*, pages 508–513. Hillsdale, NJ: Lawrence Erlbaum Associates, 1994.
10. J. F. Kolen. Recurrent networks: state machines or iterated function systems? In D. S. Touretzky J. L. Elman M. C. Mozer, P. Smolensky and A.S. Weigend, editors, *Proceedings of the 1993 Connectionist Models Summer School*, pages 203–210. Erlbaum Associates, Hillsdale, NJ, 1994.
11. P. Tiňo, M. Čerňanský, and L. Beňušková. Markovian architectural bias of recurrent neural networks. *accepted to IEEE Transactions on Neural Networks*.
12. P. Tiňo and M. Čerňanský and L. Beňušková. Markovian architectural bias of recurrent neural networks. In P. Sinčák et al., editor, *Intelligent Technologies – Theory and applications*, pages 17–23. IOS Press, 2002.
13. P. J. Werbos. Backpropagation through time; what it does and how to do it. *Proceedings of the IEEE*, 78:1550–1560, 1990.
14. R. J. Williams and D. Zipser. Gradient-based learning algorithms for recurrent networks and their computational complexity. In Y. Chauvin and D. E. Rumelhart, editors, *Back-propagation: Theory, Architectures and Applications*, pages 433–486. Lawrence Erlbaum Publishers, Hillsdale, N.J., 1995.