

---

# Evolving recurrent networks for context-free language prediction

Extended abstract for “The Workshop on Evolutionary Computation and Cognitive Science – 2000”\*

---

**Mikael Bodén**

mikael@csee.uq.edu.au  
Dept. Comp Sci & Elec Eng  
University of Queensland, Australia.

**Henrik Jacobsson**

henrikj@ida.his.se  
Dept. Comp Sci  
University of Skövde, Sweden.

**Tom Ziemke**

tom@ida.his.se  
Dept. Comp Sci  
University of Skövde, Sweden.

A series of investigations (Wiles and Elman, 1995; Tonkes and Wiles, 1998; Rodriguez et al., 1999; Bodén et al., 1999) has established that the one-step look-ahead prediction task for simple context-free languages (CFL) is difficult to learn for a Simple Recurrent Network (SRN; Elman, 1990) using a gradient-based learning algorithm. The most successful solution type to the simple CFL  $a^n b^n$  makes use of contracting oscillation for counting up  $a$ s and expanding oscillation for counting down  $b$ s. Training an SRN to such solutions involves going through at least one critical bifurcation in which the system is non-hyperbolic. In the vicinity of the bifurcation border the error gradient becomes chaotic (Bodén et al., 1999) – making gradient-based weight adaptation inappropriate. To that end, evolutionary algorithms (EAs) have shown promising results of reliably generating suitable weight sets in the same domain (Batali, 1994; Tonkes et al., 1998). This study is concerned with some simulations which evolve weight sets for both first-order and second-order recurrent networks to predict the CFL  $a^n b^n$ . Moreover, the study addresses the question if EAs are biased towards different types of solutions compared to Back-Propagation Through Time (BPTT).

Two network types were used: The SRN, which is the first-order recurrent network most commonly used for formal language recognition/prediction, and the Sequential Cascaded Network (SCN; Pollack, 1991), a second-order recurrent network. Both networks made use of the same number of processing units (1 input, 1 output and 2 state units) but due to the connection topology the SCN had slightly more weights.

All strings from the language  $a^n b^n$  ( $n \leq 10$ ) were presented to the network consecutively in random order, one letter at a time. The output was evaluated according to its ability to predict the next let-

ter in each string. A successful network manages  $a^n b^n$  for all  $n \leq 10$ . The population size was 100 of which the 20 best (according to their prediction performance) were selected in each generation. This elite was simply copied to the next generation and the remaining 80 “new” individuals were created by adding Gaussian-distributed mutations to selected members (biased by fitness) of the elite. The population of networks evolved for a maximum of 20000 generations.

The networks were evaluated quantitatively using three measures partly adapted from Tonkes et al. (1998): *Efficiency* – how many generations must be evaluated before a successful weight set is found, *Reliability* – how many of the populations find a successful weight set within  $k$  generations (here  $k = 20000$ ), and *Generalization* – how well does the successful weight set perform on longer strings of  $a^n b^n$ . For comparison a large number of networks were trained with BPTT. For various mutation settings, the reliability of the EA was considerably higher than for BPTT: around 75% on average for SRNs and 80% for SCNs compared with 50% for BPTT on SRNs and 20% for BPTT on SCNs. The efficiency was considerably lower for the EA compared with BPTT (in the order of  $10^3$  more evaluations required). The average generalization ability of the best network in the final generation was  $1 \leq n \leq 14$  on average for solutions adapted from a variety of settings.

As for qualitative differences, we examined the eigenvalues of Jacobians of the state space around the fixed points of each successful network. The eigenvalues indicate rates of contraction or expansion of system states (over time) around the fixed points and supply us with an understanding of how the task is performed. The focus is for the main fixed point when the input is held constant (either during the presentation of  $a$  or  $b$  – hereafter referred to as the A-system and the B-system). The classification criterion is based on critical borders of dynamical behaviours (bifurca-

---

\*A full paper of this abstract is under review. Please contact any of the authors for a copy

tions) effectively distinguishing between attractive and repelling behaviours, and 1- or 2-periodic or even rotating behaviours around the fixed point. For a two-dimensional state space there are two eigenvalues, each expressing the change of the system in one direction (given by the corresponding eigenvector). In the codes used below, the first two letters signify the behaviour of the smaller eigenvalue: either attracting (A) or repelling (R), and rotating (c) or, 1- or 2-periodic. The next two letters similarly indicate the behaviour of the larger eigenvalue. It should be noted that if one eigenvalue is complex (c), first, it is always accompanied by its conjugate; thus we only state the behaviour of the first eigenvalue. Second, it suggests a rotation around the fixed point.

There are two basic ways SRNs implement the task: (1) By first oscillating towards a fixed point (often using A2A2) while  $a$  is presented and then, after a shift to the part of the state space predicting  $b$ , oscillating from a fixed point (often using R2A2), with a starting point determined by the shift, until the state reaches the part of the state space which predicts  $a$  again. (2) By monotonic state-changes for each presented  $a$  relative to a fixed point (typically A1A1) and then monotonic state-changes in the opposite direction for each presented  $b$  relative a fixed point (typically A1A1). The SCN also employs a third group of behaviours namely entangled spiralling: By rotating towards a fixed point (typically Ac) while  $a$  is presented and while  $b$  is presented rotating from another fixed point (typically Rc, close to the first fixed point) in the reversed direction, until the prediction of  $a$  is triggered.

The weight sets adapted by the EA are more varied than those adapted by BPTT. With BPTT almost all SRNs made use of A2A2 for counting up  $as$ . With the EA a majority of the SRNs employed A2A2 but also A2A1 and A1A1 were used, exploiting the attracting direction to implement solution (2) above. For counting down the  $bs$ , BPTT adapted SRNs to use R2A2 exclusively. The EA managed to employ R2A1 and A1A1 as well for the same task. For SCNs BPTT predominantly used R2A2 and R2A1 for counting down  $bs$ , whereas the EA found A2A1, A2R1 and A1A1 fixed points as well. However, the entangled spiralling counting behaviour, which require that both the A-system and the B-system implement rotation, was only found with BPTT. The few Ac fixed points found by the EA only implemented slight rotation (similar to A2A2 oscillation) around the main fixed point of the A-system.

Notably, the EA makes use of more fixed points in

the solutions. For SRNs adapted by BPTT only one fixed point was found in each system. With the EA 6% additional fixed points were found each with its own behaviour. For SCNs adapted with BPTT 57% additional fixed points were used. With the EA the additional fixed points increased to 67%.

It has been demonstrated in this study that EAs can, in contrast to BPTT, reliably adapt small recurrent networks to predict a simple CFL. The efficiency by which this is done is considerably lower than with BPTT. However, the EA is not restricted to follow an error gradient and attempts, sometimes successfully, to solve the problem in ways which are seemingly hidden by error gradients. The more varied repertoire of dynamical behaviours and more available fixed points are two indicators of this that have been put forward.

## References

- Batali, J. (1994). Innate biases and critical periods: Combining evolution and learning in the acquisition of syntax. In *Proceedings of the Alife IV Workshop*, Cambridge, MA.
- Bodén, M., Wiles, J., Tonkes, B., and Blair, A. (1999). Learning to predict a context-free language: Analysis of dynamics in recurrent hidden units. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 359–364.
- Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14:179–211.
- Pollack, J. B. (1991). The induction of dynamical recognizers. *Machine Learning*, 7:227.
- Rodriguez, P., Wiles, J., and Elman, J. L. (1999). A recurrent neural network that learns to count. *Connection Science*, 11(1):5–40.
- Tonkes, B., Blair, A., and Wiles, J. (1998). Inductive bias in context-free language learning. In *Proceedings of the Ninth Australian Conference on Neural Networks*, pages 52–56.
- Tonkes, B. and Wiles, J. (1998). Learning a context-free task with a recurrent neural network: An analysis of stability. In *Proceedings of the Fourth Biennial Conference of the Australasian Cognitive Science Society*. In press.
- Wiles, J. and Elman, J. L. (1995). Learning to count without a counter: A case study of dynamics and activation landscapes in recurrent networks. In *Proceedings of the Seventeenth Annual Meeting of the Cognitive Science Society*, pages 482–487. Lawrence Erlbaum.