

On Learning Context Free and Context Sensitive Languages

Mikael Bodén, Janet Wiles

M. Bodén is with the School of Information Science, Computer and Electrical Engineering, Halmstad University, PO Box 823, 30118 Halmstad, Sweden. Email: mikael.boden@ide.hh.se .

J. Wiles is with the School of Information Technology and Electrical Engineering, University of Queensland, St Lucia 4072, Australia. Email: janetw@itee.uq.edu.au

Abstract

The Long Short Term Memory is not the only neural network which learns a context sensitive language. Second-order Sequential Cascaded Networks are able to induce means from a finite fragment of a context-sensitive language for processing strings outside the training set. The dynamical behavior of the Sequential Cascaded Network is qualitatively distinct from that observed in Long Short Term Memory networks. Differences in performance and dynamics are discussed.

Keywords

Recurrent neural network, language, prediction.

I. INTRODUCTION

Gers and Schmidhuber [9] present a set of simulations with the so-called Long Short Term Memory (LSTM) network on learning and generalizing to a couple of context free and a context sensitive language. The successful result is profound for at least two reasons: First, Gold [10] showed that, under certain assumptions, no super-finite languages can be learned from positive (grammatically correct) examples only. The possibilities are thus that the network (and its environment) enforces a learning bias which enables the network to capture the language, or the predictive learning task implicitly incorporates information of what is ungrammatical (cf. [14]). Second, the network establishes the necessary means for processing embedded sentences without requiring potentially infinite memory (e.g. by using stacks). Instead the network relies on the analog nature of its state space.

Contrary to what is claimed in [9], the LSTM is not the only network architecture which has been shown able to learn and generalize to a context sensitive language. Specifically, second-order Sequential Cascaded Networks (SCNs; [12]) are able to learn to process strings well outside the training set in a manner which naturally scales with the length of strings [4]. First-order Simple Recurrent Networks (SRNs; [8]) induce similar mechanisms to SCNs but have not been observed to generalize [6].

As reported, the LSTM network exhibit impressive generalization performance by simply adding a fixed amount to a linear counter in its state space [9]. Notably, both SRNs and SCNs process these recursive languages in a qualitatively different manner compared to LSTM. The difference in dynamics is highlighted as it provides insight into the issue of generalization but also as it may have implications for application and modelling purposes.

Moreover, complementary results to Gers and Schmidhuber's [9] treatment are supplied. We focus on the SCN since it clearly demonstrates generalization beyond training data.

II. LEARNING ABILITY

Similar to [9] networks are trained using a set of strings, called S , generated from $a^n b^n c^n$ where $n \in 1, \dots, 10$. Strings from S are presented consecutively, the network is trained to predict the next letter, and n is selected randomly for each string.¹ Contrary to [9] we do not employ *start-of-string* or *end-of-string* symbols (the language is still context-sensitive). The crucial test for successfully processing a string is based on predicting the first letter of the next string.

The SCN has 3 input, and 3 output units (one for each symbol; a , b and c). Two sigmoidal state units² are sufficient. Consequently, the SCN has a small and bounded state space $[0, 1]^2$ in contrast with the LSTM which is equipped with several specialized units of which some are unbounded and some bounded.

Backpropagation through time (BPTT) is used for training the SCN. The best SCN generalizes to all strings $n \in 1, \dots, 18$ (see Table I) and the best LSTM manages all strings $n \in 1, \dots, 52$ [9]. BPTT suffers from a "vanishing gradient" and is thus prone to miss long-term dependencies [11], [1]. This may to some extent explain the low proportion of SCNs successfully learning the language (see Table I). In a related study the low success rate and observed instability during learning are partly explained by the radical shifts in dynamics employed by the network [5]. Chalup and Blair [6] trained a 3 hidden unit SRN to predict $a^n b^n c^n$ using an incremental version of hill-climbing (IHC; see Table I).

III. GENERALIZATION

A. Infinite languages

It is important to note that neither Gers and Schmidhuber [9] nor we are actually training our networks using a non-regular context sensitive language. A finite fragment of what a context sensitive grammar generates is a straightforwardly regular language –

¹The target is only the next letter of the current string and not (as in [9]) all possible letters according to the grammar. According to [9] LSTM performs similarly in both cases.

²The logistic activation function was used.

TABLE I

RESULTS FOR RECURRENT NETWORKS ON THE CSL $a^n b^n c^n$, SHOWING (FROM LEFT TO RIGHT) THE NUMBER OF HIDDEN (STATE) UNITS, THE VALUES OF n USED DURING TRAINING, THE NUMBER OF SEQUENCES USED DURING TRAINING, THE NUMBER OF FOUND SOLUTIONS/TRIALS, AND THE LARGEST ACCEPTED TEST SET.

Network	Hidden Units	Train. Set[n]	Train. Str. [10^3]	Sol./Tri.	Best Test [n]
SCN/BPTT[4]	2	1, ..., 10	max 20	23/1000	1, ..., 18
SRN/IHC[6]	3	1, ..., 12	na	na	1, ..., 12
LSTM[9]	na	1, ..., 10	avg 54	10/10	1, ..., 52

there are only ten possible strings and there is a finite automaton that will recognize each and reject all others. To test if $a^{11}b^{11}c^{11} \in S$ is to test if the network employs some other means than those expected in a finite automaton. However, networks have been observed to do so even when trained on regular languages [2]. So, to claim complete success, we need to present a proof that the network is solving all possible instances, unless we test for an infinite number of strings. Neither the LSTM nor the SCN or SRN succeed in this strict sense (admittedly, Gers and Schmidhuber test for a very large number of strings). To this end, the underlying mechanisms need to be established.

Turing machines are able to process infinitely long strings of any language since they have access to an infinite memory. Linear bounded automaton is (at most) linear in the input size and if an arbitrary long string – generated by a context sensitive grammar – is presented it can assume sufficient memory resources. Can we establish that our networks process strings in a manner which similarly scales with the size of the input?

B. Processing mechanisms

As noted by Gers and Schmidhuber, SRNs learn simple context free languages. One of the studied languages, $a^n b^n$, was also tested on SCNs [4]. SCNs are as capable of inducing mechanisms for $a^n b^n$ as SRNs. After successful training, SCNs also exhibit similar dynamics as SRNs for processing. Interestingly – in particular in the light of the clear

distinction between context free and context sensitive languages made by classical linguists – qualitatively similar dynamics is induced for $a^n b^n c^n$ [4].

For $a^n b^n$ we observe two principal types of dynamics which generalize beyond the training set [3].

- Oscillation towards an attractive fixed point while counting as and oscillation from a repelling fixed point while counting bs . The oscillation rate is set so that the a -count matches the b -count. Only one principal dimension is required to keep track of one counter.
- Spiraling towards a fixed point while counting as and spiraling counterwise from a near fixed point. The spiraling rate is set so that the a -count and b -count match. At least two dimensions are required to implement the spiral.

A majority of the successful SCNs, in a large set of simulations, employed oscillation. The SRN only employed oscillation.

For $a^n b^n c^n$ we observe only one type of dynamics which generalizes beyond training data: oscillation towards a fixed point while counting as , dual-pronged oscillation with reference to a saddle fixed point while counting bs and oscillation from a third fixed point while counting cs (for a typical example see Figure 1). The oscillation rates are set so that the a -count matches the b -count and the b -count matches the c -count. The second fixed point requires two principal dimensions: one in which the fixed point is repelling (matching the a -count) and one in which the fixed point is attractive (to match the c -count). By linearizing the system around the fixed points it is possible to characterize the dynamics in terms of eigenvalues and eigenvectors [13]. In [4] it was established, using linearization, that the dynamics for processing $a^n b^n$ and $a^n b^n c^n$ are qualitatively the same.

According to the diagrams presented in [9] state units in LSTM networks seem to count by increasing or decreasing activation (referred to as the “cell state” in [9]) monotonically. Basically, every a presented increases the activation of a cell, and every b decreases the activation of the same cell, using a matched rate. When the activation level is below a threshold the network will predict the symbol shift. For $a^n b^n c^n$ at least two cells are required but the same principle applies. This mechanism seems to rely on unbounded activation functions and an infinitely large state space. In SRNs and SCNs monotonic counters have also been found – but they do not generalize [16]. However, since bounded

sigmoidal activation functions are used on state units the activation quickly saturates.

IV. DISCUSSION

Monotonic counters are obviously much more stable – both in terms of learning (the LSTM network was never observed to lose track of a solution) and processing (the LSTM generalized remarkably well). However, for a monotonic counter to be generalizing well it is natural to assume that an unbounded and linear state space is required. Semi-fractal counters as observed in bounded state spaces of SRNs and SCNs, on the other hand, degrades quickly with lack of precision. It needs to be emphasized that so does human memory [7]. The performance profile of SRNs on recursive languages correlates well with psycholinguistic behaviours [7]. Are LSTMs – given their remarkable generalization performance – cognitively plausible?

Siegelmann has proven that a recurrent network can be used to implement a universal Turing machine [15]. The proof relies on a fractal encoding of data in state space as opposed to digits on a tape. The fractal encoding has the advantage that apart from carrying counting information it can also incorporate contents. By fixing the alphabet in advance it is possible to adjust the length of trajectories in state space into smaller or larger fractions with respect to the particular symbol and operation (push or pop). When a sequence of digits has been encoded this way – into a single value – it is possible to uniquely (and instantaneously) identify each component. The linear monotonic counter does not obviously lend itself to this added functionality as each step must be equally long. From the analysis presented in [9], a separate counter is employed by the LSTM network for each symbol and coordinated separately. The semi-fractal counter, on the other hand, bears many similarities to Siegelmann's proposal. It remains to be seen, however, that content-carrying oscillation can be realized and automatically induced.

The two principal approaches for scaling up with an increased input size: monotonic and fractal encoding, put different requirements on the state space. To process infinitely long strings monotonic counters require that the state space is infinitely large whereas fractal counters require a state space with infinite precision. As this brief note has only scratched the surface of the possible ways of processing recursive languages, future work should establish a precise account for the dynamics of LSTMs.

REFERENCES

- [1] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, March 1994.
- [2] Alan D. Blair and Jordan B. Pollack. Analysis of dynamical recognizers. *Neural Computation*, 9(5):1127–1142, 1997.
- [3] Mikael Bodén and Alan Blair. Learning the dynamics of embedded clauses. *Applied Intelligence: Special issue on natural language processing by neural networks*, 2001. In press.
- [4] Mikael Bodén and Janet Wiles. Context-free and context-sensitive dynamics in recurrent neural networks. *Connection Science*, 12(3):197–210, 2000.
- [5] Mikael Bodén, Janet Wiles, Brad Tonkes, and Alan Blair. 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, Edinburgh, 1999. IEE.
- [6] Stephan Chalup and Alan D. Blair. Hill climbing in recurrent neural networks for learning the $a^n b^n c^n$ language. In *Proceedings of the 6th International Conference on Neural Information Processing*, pages 508–513, Perth, 1999.
- [7] Morten Christiansen and Nick Chater. Toward a connectionist model of recursion in human linguistic performance. *Cognitive Science*, 23:157–205, 1999.
- [8] Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.
- [9] Felix A. Gers and Jürgen Schmidhuber. LSTM recurrent networks learn simple context free and context sensitive languages. *IEEE Transactions on Neural Networks*, 12(6):1333–1340, 2001.
- [10] E. M. Gold. Language identification in the limit. *Information and Control*, 16:447–474, 1967.
- [11] Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. Diploma thesis, Institut für Informatik, Technische Universität München, Germany, 1991.
- [12] Jordan B. Pollack. The induction of dynamical recognizers. *Machine Learning*, 7:227, 1991.
- [13] Paul Rodriguez, Janet Wiles, and Jeffrey L. Elman. A recurrent neural network that learns to count. *Connection Science*, 11(1):5–40, 1999.
- [14] Douglas L. T. Rohde and David C. Plaut. Language acquisition in the absence of explicit negative evidence: How important is starting small? *Cognition*, 72:67–109, 1999.
- [15] Hava T. Siegelmann. *Neural Networks and Analog Computation: Beyond the Turing Limit*. Birkhäuser, 1999.
- [16] Brad Tonkes, Alan Blair, and Janet Wiles. Inductive bias in context-free language learning. In *Proceedings of the Ninth Australian Conference on Neural Networks*, pages 52–56, 1998.

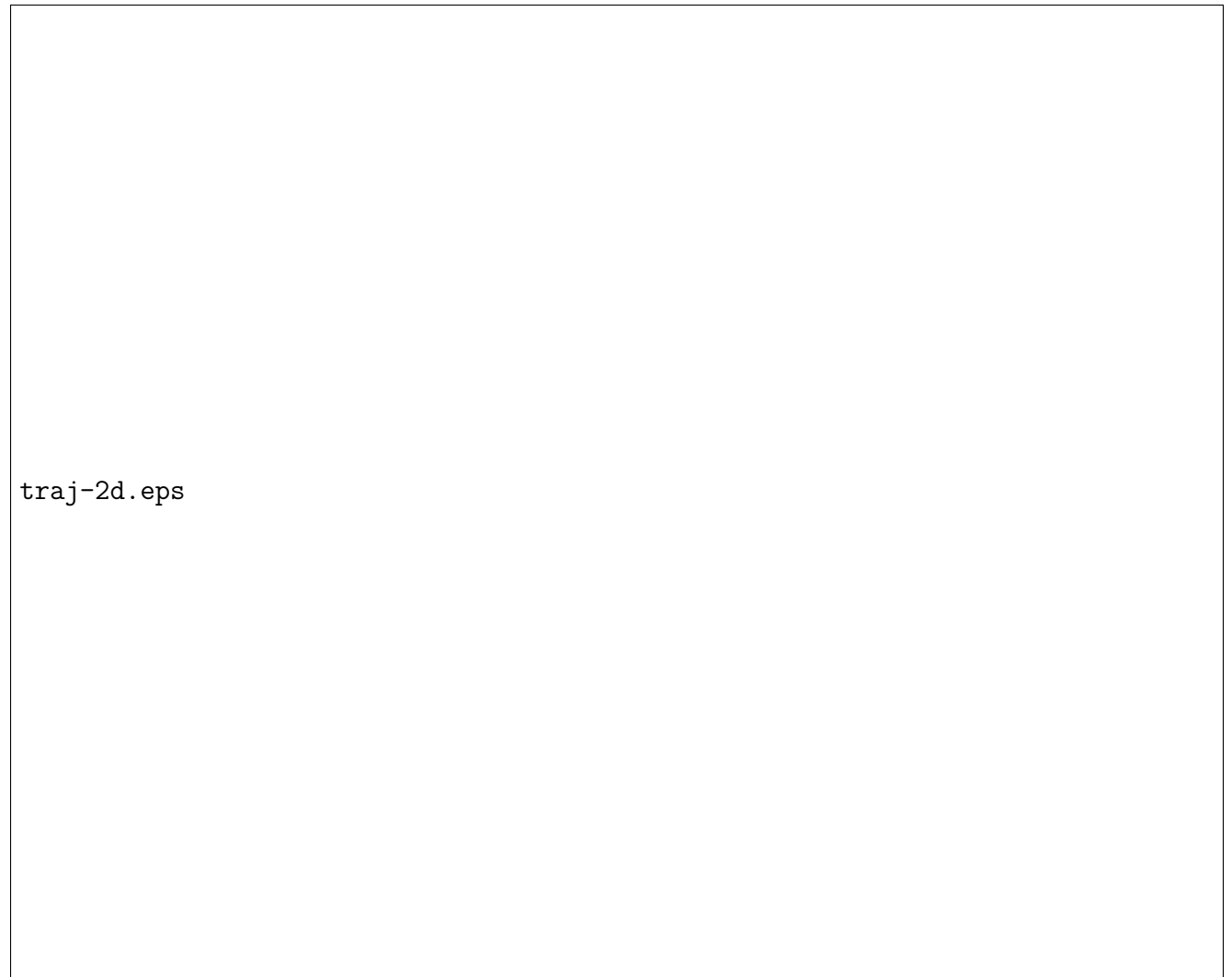


Fig. 1. The activation trajectory in the state space of an SCN, generated when presenting $a^8b^8c^8$, is shown as a solid line. The decision boundaries in output space (0.5) are shown as dotted lines. When the first a is presented the state quickly aligns with the fixed point of the a -system (1.00, 0.56) and starts oscillating towards it. When the first b is presented the state is attracted to the b fixed point (0.28, 0.45) but the attraction is only in one principal dimension and as oscillation continues the state is repelled from the same fixed point in the other principal dimension. The repel rate matches the a count to signal when the first c is expected. The attract rate of the b fixed point is used for determining when the final c has been presented. The c fixed point (0.36, 0.89) is repelling by oscillation. The activation crosses the decision boundaries one time step ahead of each symbol shift due to a temporal delay of the SCN.