Learning Nonregular Languages: A Comparison of Simple Recurrent Networks and LSTM

J. Schmidhuber, F. Gers, D. Eck
e-mail: {juergen,felix,doug}@idsia.ch web: www.idsia.ch
Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA)
Galleria 2, 6928 Manno-Lugano, Switzerland

January 6, 2003

Abstract

In response to Rodriguez' recent article (2001) we compare the performance of simple recurrent nets and *"Long Short-Term Memory"* (LSTM) recurrent nets on context-free and context-sensitive languages.

Rodriguez (2001) examined the learning ability of simple recurrent nets (SRNs) (Elman, 1990) on simple context sensitive and context free languages (CFLs and CSLs). He trained his SRN on short training sequences of length not much greater than 10. He found that the SRN does not generalize well on significantly larger test sets, and frequently is unable to store the training set. Similar results were recently reported by Bodén and Wiles (2000) who studied the simple context sensitive language (CSL) $a^n b^n c^n$. They trained

SRNs on sequences defined by $n = 1, 2, \ldots, 10$ and found that SRNs fail to reliably generalize to $n > 13$. SCNs (sequential cascaded networks) did only moderately better.

We applied *"Long Short-Term Memory"* (LSTM) recurrent nets (Hochreiter and Schmidhuber, 1997) to similar problems (Gers and Schmidhuber, 2001a,b). Many of our training set sizes were comparable to those of Rodriguez and Bodén & Wiles. We found that LSTM almost always stores the training set and generalizes well on much larger test sets.

For instance, we trained LSTM on strings from the context-sensitive language $a^n b^n c^n$ with $n <= 40$. From this training data LSTM readily generalized up to string size 1500, that is, it learned to accept legal test strings such as $a^{500} b^{500} c^{500}$ while rejecting slightly different illegal strings such as $a^{500} b^{499} c^{500}$. In other experiments LSTM was able to generalize well having seen only very limited training data. For example, from the training strings with $n = 50, 51$ LSTM generalized to $43 <= n <= 57$.

Similarly, LSTM trained on the CFL $a^n b^n$ for $1 <= n <= 30$ generalized in the best case up to $n = 1000$ and on average up to $n = 408$. We refer readers to Gers and Schmidhuber (2001b) for an analysis of this and similar tasks, as well as for LSTM equations.

True, even LSTM did not learn these languages in the strict sense — we do not have a proof that it will generalize to arbitrary n. But it is obvious that LSTM exhibits excellent generalization performance, while SRNs do not. Because LSTM was designed to solve hard problems and not specifically to address human cognition, it remains to be seen how well LSTM will model behavioral data.

LSTM also successfully learned CSLs that require *embedded* counters, e.g., the deterministic palindrome language $a^m b^n B^n A^m$ discussed in Rodriguez's paper (traditional RNNs did not even learn the training set). We refer to reference Gers and Schmidhuber (2001b) for details. Testing more complex CFLs is also a topic of future research.

How does LSTM solve problems like $a^n b^n c^n$? Counters of potentially unlimited size are automatically and naturally implemented by linear units, the "Constant Error Carousels" (CECs) of standard LSTM, originally designed to overcome error decay problems plaguing previous RNNs (Hochreiter, 1991; Hochreiter et al., 2001). Each linear CEC is surrounded by a cloud of nonlinear units responsible for controlling the flow of information in and out of the CEC. Whereas standard RNNs have a hard time implementing precise linear counters with their squashing functions — they need finely tuned weights to countermand the nonlinearities — LSTM can simply use the CECs for counting and so is free to focus its robust and precise weight adaptation process on the nonlinear aspects of sequence processing.

In the case of the CFL $a^n b^n$, LSTM uses one CEC to count up for a's and down for b's while using nonlinear gates to protect the CEC from irrelevant signals and to generate the appropriate accept or reject response. In the case of the CSL $a^n b^n c^n$ LSTM uses one CEC to count up for a's and down for b's and a second CEC to count up for b's and down on c's. SRN/SCN counting mechanisms are analyzed in Rodriguez (2001) and Bodén and Wiles (2000).

The CEC-based counters not only deal much better with the well-know long time lag problem (Hochreiter, 1991; Hochreiter et al., 2001) but also the problem of installing easily accessible linear counters whose increments and decrements implement natural push and pop operations. This explains in a nutshell why LSTM outperforms other recurrent nets not only on regular (Hochreiter and Schmidhuber, 1997) but also on context-free and context-sensitive languages (Gers and Schmidhuber, 2001b).

References

Bodén, M. and Wiles, J. (2000). Context-free and context-sensitive dynamics in recurrent neural networks. *Connection Science*, 12(3/4):197–210.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, 14:179–211.

Gers, F. A. and Schmidhuber, J. (2001a). Long Short-Term Memory learns context free and context sensitive languages. In V. Kurkova et.al., editor, *Proceedings of the ICANNGA 2001 Conference*, volume 1, pages 134–137, Wien, NY. Springer.

Gers, F. A. and Schmidhuber, J. (2001b). LSTM recurrent networks learn simple context free and context sensitive languages. *IEEE Transactions on Neural Networks*, 12(6):1333–1340.

Hochreiter, S. (1991). Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München. See http://ni.cs.tu-berlin.de/∼hochreit/papers/-hochreiter.dipl.ps.gz.

Hochreiter, S., Bengio, Y., Frasconi, P., and Schmidhuber, J. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. In Kremer, S. C. and Kolen, J. F., editors, *A Field Guide to Dynamical Recurrent Neural Networks*. IEEE Press.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Rodriguez, P. (2001). Simple recurrent networks learn context-free and context-sensitive languages by counting. *Neural Computation*, 13(9):2093–2118.