Incremental Learning for RNNs: How Does it Affect Performance and Hidden Unit Activation?

Stephan K. Chalup
School of Electrical Engineering
& Computer Science
The University of Newcastle
Callaghan, 2308, Australia
chalup@cs.newcastle.edu.au

Alan D. Blair
School of Computer Science
& Engineering
The University of New South Wales
Sydney, 2052, Australia
blair@cse.unsw.edu.au

Abstract

In this short paper we summarise our work [5] on training first-order recurrent neural networks (RNNs) on the $a^n b^n c^n$ language prediction task. We highlight the differences between incremental and non-incremental learning – with respect to success rate, generalisation performance, and characteristics of hidden unit activation.

1 Background

In 1999 a pilot study [4] demonstrated for the first time that simple recurrent networks (SRNs) [7] can learn to predict strings from subsets of the mildly context-sensitive language $\{a^n b^n c^n; n \geq 1\}$. The important aspect of this type of result [5, 23, 20, 16, 15] is that very simply structured recurrent nets can *learn* to predict fairly complex formal languages [6]. The question of representation or how to handcraft such a network is by now well understood [10, 11, 19, 18]. The training algorithm employed in our studies [4, 5] was evolutionary hill climbing [14] combined with a special version of data incremental learning [8, 3]. Later studies [1, 2] were able to obtain similar results with backpropagation through time (BPTT) and second order sequential cascaded networks, although training with BPTT of first order recurrent networks was not successful. A selection of studies of training on non-regular languages such as $a^n b^n$ and $a^n b^n c^n$ was reviewed in [22, 2]. One common characteristic of all these studies was limited generalization ability – the networks generalised only a few steps ahead with respect to the depth of the strings. It was suggested [21] that the limited generalisation ability could model human performance when processing center embedded clauses. Better generalisation, close to the performance of handcrafted nets [10], could be learned with more complicated network structures such as RAAM networks [12] or LSTM networks [9, 13, 17]. In the present paper we focus on the training of first-order recurrent networks, and emphasize the differences in results obtained with incremental and non-incremental evolutionary learning [5].

2 Task and methods

The data consisted of sequences of 30 randomly concatenated strings from the context-sensitive language $\{a^n b^n c^n; n \geq 1\}$. The network has to predict, for each symbol, the next symbol in the sequence [7]. Each symbol was encoded as a vector a = (-1,1,1), b = (1,-1,1) or c = (1,1,-1), respectively. The symbols of the sequence were fed one at a time into the three dimensional input layer (I1-I3) of the SRNSC neural network (see Figure 1). Each output unit (O1-O3) was assigned to one of the three symbols a, b, or c and the unit with the highest activation determined the predicted symbol.

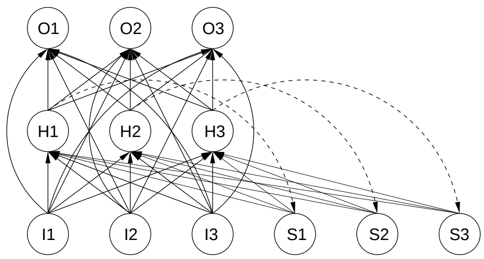

Figure 1: A SRNSC is a SRN à la Elman [7] with additional shortcut links connecting each input unit (I1-I3) to each output unit (O1-O3) [5].

The training algorithm was evolutionary hillclimbing [14] which is also known under the name (1+1)–Evolution Strategy. It was combined with *data juggling* which is a method that randomly changes the order of the strings in the training sequence after each epoch during training [5].

Data incremental learning [8, 3] for recurrent neural networks is based on the assumption that it is better to train a network on simple data initially and gradually increase the difficulty of the data as the training progresses, rather than training on the full range of data from the very beginning. In the context of the $a^n b^n c^n$ prediction task – since the strings are naturally ordered by their depth n – *data-incremental learning* was implemented by allowing only small strings at first and then increasing the maximum allowable depth once the strings of the current training set had been successfully learned. A comparison was made with a *non-incremental* approach, in which the network is trained on the full range of strings from the outset.

3 Summary of results

In the case of the $a^n b^n c^n$ prediction task with SRNSCs we obtained strong indications [5] that incremental learning finds solutions (for stage 8) faster and with a higher success rate (58%) than non-incremental learning (success rate 25%). However, only 30% of the successful incrementally trained networks were able to generalise to higher stages while 60% of the successful non-incrementally trained networks showed evidence of generalization.

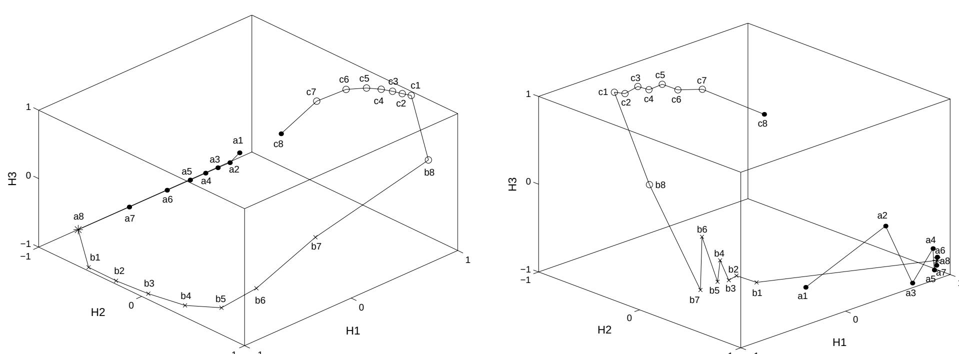

Figure 2: Solution after incremental learning on stages 3-8 (left) and with non-incremental learning directly on stage 8 (right) starting from the same initial conditions.

In addition, we noticed a qualitative difference in the hidden unit activity of the resulting solution networks – namely, incremental training was more likely to produce solutions with monotonic trajectories, while the non-incremental solutions had trajectories which oscillated within the symbol clusters (see Figure 2).

These different network behaviors can be distinguished empirically by counting the number of positive and negative self-weights. The higher level of generalization for the non-incrementally trained networks concurs with earlier work for the $a^n b^n$ task [20] where it was noted that oscillating solutions are more likely to generalize than monotonic ones.

Animated graphics showing the development of the hidden unit dynamics during evolution are at `http://www.cs.newcastle.edu.au/~chalup/anbncn.html`. The movies show that in most cases the qualitative characteristics of the networks' hidden unit activity did not change much during training.

Our hypothesis is that the incrementally trained networks find it easier to learn a monotonic solution when presented with the short strings in the initial training set, but are then "locked in" to this behavior and find it increasingly difficult to accommodate longer strings within this pattern of monotonic dynamics. In contrast, the non-incrementally trained networks take longer to find a solution initially – but generally settle on an oscillating solution, which can more easily generalize to longer strings.

References

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

[2] Bodén, M., and Wiles, J. (2002). On learning context-free and context-sensitive languages. *IEEE Transactions on Neural Networks* **13**(2), pp. 491–493.

[3] Chalup, S. K. (2002). Incremental learning in biological and machine learning systems. *International Journal of Neural Systems* **12**(6), pp. 447–465.

[4] Chalup, S. K. and Blair, A. D. (1999). Hill climbing in recurrent neural networks for learning the $a^n b^n c^n$ language. *Proceedings, 6th International Conference on Neural Information Processing (ICONIP'99)*, pp. 508–513.

[5] Chalup, S. K. and Blair, A. D. (2003). Incremental training of first order recurrent neural networks to predict a context-sensitive language. *Neural Networks* **16**(7), pp. 955–972.

[6] Doya, K. (2003). Recurrent Networks: Learning Algorithms. In: Arbib, M. A. (editor). *The Handbook of Brain Theory and Neural Networks, Second Edition*, MIT Press, pp. 955–960.

[7] Elman, J. L. (1990). Finding structure in time. *Cognitive Science* **14**, pp. 179–211.

[8] Elman, J. L. (1993). Learning and development in neural networks: The importance of starting small. *Cognition* **48**, pp. 71–99.

[9] Gers, F. A. and Schmidhuber, J. (2001). LSTM recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions On Neural Networks* **12**(6), pp. 1333–1340.

[10] Hölldobler, S., Kalinke, Y., and Lehmann, H. (1997). Designing a counter: Another case study of dynamics and activation landscapes in recurrent networks. *KI-97: Advances in Artificial Intelligence – Proceedings of the 21st Annual German Conference on Artificial Intelligence*, LNAI **1303**, Springer-Verlag, pp. 313–324.

[11] Kalinke, Y. and Lehmann, H. (1998). Computation in Recurrent Neural Networks: From Counters to Iterated Function Systems. *Advanced Topics in Artificial Intelligence – 11th Australian Joint Conference on Artificial Intelligence, AI'98*, LNAI **1502**, Springer-Verlag, pp. 179–190.

[12] Melnik, O., Levy, S., and Pollack, J. B. (2000). RAAM for infinite context-free languages. *International Joint Conference on Neural Networks (IJCNN'2000)*, Vol. **5**, IEEE Press.

[13] Pérez-Ortiz, J. A., Gers, F. A., Eck, D., and Schmidhuber, J. (2002). Kalman filters improve LSTM network performance in problems unsolvable by traditional recurrent nets. *Neural Networks* **16**(2), pp. 241–250.

[14] Pollack, J. B. and Blair, A. (1998). Co-evolution in the successful learning of backgammon strategy. *Machine Learning* **32**, pp. 225–240.

[15] Rodriguez, P. (2001). Simple recurrents networks learn context-free and context-sensitive languages by counting. *Neural Computation* **13**(9), pp. 2093–2118.

[16] Rodriguez, P., Wiles, J., and Elman, J. L. (1999). A recurrent neural network that learns to count. *Connection Science* **11**(1), pp. 5–40.

[17] Schmidhuber, J., Gers, F. A., and Eck, D. (2002). Learning nonregular languages: A comparison of simple recurrent networks and LSTM. *Neural Computation* **14**(9), pp. 2039–2041.

[18] Siegelmann, H. T. (1999). *Neural Networks and Analog Computation—Beyond the Turing Limit*. Progress in Theoretical Computer Science. Birkhäuser Boston.

[19] Steijvers, M. and Grünwald, P. (1996). A recurrent network that performs a context-sensitive prediction task. *Proceedings of the 18th Annual Conference of the Cognitive Science Society*. Morgan Kauffman, pp. 335–339.

[20] Tonkes, B., Blair, A., and Wiles, J. (1998). Inductive bias in context-free language learning. *Proceedings of the Ninth Australian Conference on Neural Networks (ACNN'98), Brisbane*. University of Queensland, pp. 52–56.

[21] Tonkes, B. and Wiles, J. (1999). Learning a context-free task with a recurrent neural network: An analysis of stability. In: Heath, R., Hayes, B., Heathcote, A., Hooker, C. (eds.), *Dynamical Cognitive Science: Proceedings of the Fourth Biennial Conference of the Australasian Cognitive Science Society (OzCogSci97)*.

[22] Wiles, J., Blair, A., and Boden, M. (2001). Representation beyond finite states: Alternatives to push-down automata. In: Kolen, J. F. and Kremer, S. C. (eds.). *A Field Guide to Dynamical Recurrent Networks*. IEEE Press, pp. 129-142.

[23] Wiles, J. and Elman, J. L. (1995). Learning to count without a counter: A case study of dynamics and activation landscapes in recurrent networks. *Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society*. MIT Press, pp. 482–487.