Online Symbolic-Sequence Prediction with Discrete-Time Recurrent Neural Networks*

Juan Antonio Pérez-Ortiz, Jorge Calera-Rubio, and Mikel L. Forcada

Departament de Llenguatges i Sistemes Informàtics,
Universitat d'Alacant, E-03071 Alacant, Spain

Abstract. This paper studies the use of discrete-time recurrent neural networks for predicting the next symbol in a sequence. The focus is on *online* prediction, a task much harder than the classical offline grammatical inference with neural networks. The results obtained show that the performance of recurrent networks working online is acceptable when sequences come from finite-state machines or even from some chaotic sources. When predicting texts in human language, however, dynamics seem to be too complex to be correctly learned in real-time by the net. Two algorithms are considered for network training: real-time recurrent learning and the decoupled extended Kalman filter.

1 Introduction

Discrete-time recurrent neural networks (DTRNN) [5] are generally accepted as a good alternative to feedforward networks for temporal-sequence processing. Feedback enables DTRNN to develop state representations of this kind of sequences. This work analyses the use of DTRNN for predicting online the next component in a symbolic sequence.

Arithmetic compression [7] is used to evaluate the quality of the predictor. Different symbolic-sequence sources ranging from finite-state machines to texts in human language are considered in the experiments. Unlike previous works [3, 11, 13] which performed neural *offline* prediction on the kind of sequences studied here, in this paper we concentrate on *online* prediction.

2 Prediction with DTRNN

We have chosen two classical DTRNN: Elman's *simple recurrent network* (SRN) [4], and the *recurrent error propagation network* (REPN) [10]. Both architectures are applied to online prediction of symbolic sequences. The networks work in real-time trying to produce an output as correct as possible to every component of the sequence supplied at each time step; this output is considered as a prediction of the probabilities of the next symbol in the sequence.

* Work supported by the Generalitat Valenciana through grant FPI-99-14-268 and the Spanish Comisión Interministerial de Ciencia y Tecnología through grant TIC97-0941. This paper completes the results in [8].

Two online supervised training algorithms are considered: the real-time recurrent learning (RTRL) [14], and the decoupled extended Kalman filter (DEKF) [9]. Both of them update weights according to an error measure $E(t)$ whenever a new target or desired output is supplied. The RTRL algorithm performs gradient descent along the instantaneous error hypersurface. On the other hand, the DEKF computes recursively and efficiently a solution to the *least-squares method*: finding the curve of best fit for a given set of data in terms of minimizing the average distance between the data and the curve. The DEKF also needs the derivatives of $E(t)$, which may be calculated the same way as in the RTRL algorithm.

Grammatical Inference with DTRNN. Grammatical inference (GI) is a task where DTRNN have been widely used [3] and that is related to our problem, although it also has strong differences.

Even though one possible approach to GI consists of training the network to predict the next symbol in the sequence, the modus operandi is far different. In the case of GI, it can be easily shown (by writing the total quadratic prediction error as a sum over all the prefixes of all the sequences in the sample) that the ideal neural probability model obtained for a finite set of finite sequences through exhaustive *offline* training, *global* quadratic error function, and exclusive coding of outputs (see later), gives an output that approximates as much as possible the next-symbol relative frequencies observed in the finite sample, which could be used as approximate probabilities when treating unseen sequences.

Our problem is different in some respects: the processing is done *online*, a relatively-long *single* sequence is used (whereas in GI the length of the sequences is usually *small*), and as a result of online processing, the error function is *local*. This work is based on the conjecture, similar to Elman's [4], that even under these different conditions the output of the model may still be considered as an approximation to next-symbol probabilities.

Neural Online Prediction. This section shows briefly how DTRNN can be used to predict online the next symbol in a sequence. Consider we have an alphabet $\Sigma = \sigma_1, \ldots, \sigma_{|\Sigma|}$ and a temporal sequence to process $s[1], \ldots, s[t], \ldots, s[L]$. The number of neurons in the input and output layers is set equal to the size of the alphabet, $|\Sigma|$. Inputs and targets are coded by means of *exclusive coding*, that is, the symbol σ_i is coded with a unary vector with all the components but the i-th set to zero.

At time t the symbol $s[t]$ is coded in the input vector $\boldsymbol{u}[t]$ using exclusive coding. Introducing $\boldsymbol{u}[t]$ into the network, the output vector $\boldsymbol{y}[t]$ is obtained and normalized so as all its components add to one. Now, $\boldsymbol{y}_i[t]$ can be interpreted as the probability of next symbol being σ_i. After that, the next observed symbol $s[t+1]$ is exclusively coded in $\boldsymbol{d}[t]$, which is used as the target for weight updating by the online training algorithm.

We will study empirically whether the probabilities obtained this way are a good approximation to the real ones. Online learning makes this difficult because the net is simultaneously learning how much history to keep and how to predict from that history, and proving convergence to the real probabilities is not trivial; this is worth further theoretical study.

3 Measure of Prediction Quality

Mean squared error may be a way of measuring the quality of a predictor. But in the common case where we do not know the real next-symbol probabilities, the error may only be based on the difference between the predicted symbol (usually the one with the highest probability) and the symbol observed at next time step. An alternative error measure which considers the whole vector of predicted probabilities is necessary: arithmetic compression is a possible solution.

An *arithmetic compressor* [7] derives its performance from a correct model of next-symbol probabilities. The important property that makes it appropiate for evaluating the quality of a predictor (neural or not) is that the better this probability model, the larger the *compression ratio* (CR) obtained. Besides that, this model can be adaptive, which is the case when the processing is online and probabilities change dynamically at every time step.

With the object of comparising, in addition to DTRNN, *n-grams* are used as an alternative probability model for arithmetic compression. In an n-gram model, next-symbol probability depends only on the $n-1$ symbols preceding it. Very good results are obtained when various models of different orders n are maintained simultaneously and the probability estimation is computed by mixing them [7]. The predictor considered in our experiments uses $n \in [0,4]$.

4 Results

This section presents the results of the experiments developed with some symbolic sequences. The graphs illustrate the CR versus the number of state neurons, denoted with n_X.[1] Results are shown for both the RTRL and the DEKF training algorithms (using a quadratic prediction error), and SRN and REPN architectures. Note that $n_X = 0$ means that there is no recurrence at all (in the case of SRN, only output biases are adjusted).

RTRL was used with learning rate 0.9 and momentum 0.4 (chosen after preliminary experiments). In the case of the DEKF, some experiments were carried out in order to determine correct values for the tunable parameters of the algorithm; the values proposed by Haykin [5, p. 771] proved to be correct.

The results are the average of 7 experiments (the variance was very small in all cases so it is omitted). The initial values for the weights were taken randomly from a uniform distribution in $[-0.2, 0.2]$. We consider three different kinds of symbolic sequences: generated by finite-state machines, chaotic, and texts in human language. A discussion of one particular sequence of each group and the results obtained follows.

[1] The code used for the arithmetic coder was written by Nelson [7] and is freely available at `http://dogma.net/markn/articles/arith`.

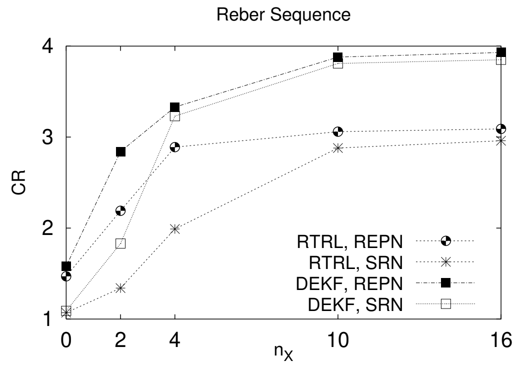

Fig. 1. Compression ratios for the continual embedded Reber sequence. The $[0,4]$-gram model gives a CR of 4.23.

Finite-State Sequences. The experiments consider the continual embedded Reber sequence [12]. From the outset, good results are expected since DTRNN are capable of learning *offline* regular languages [3] and even emulating finite-state machines [2]. On the other hand, languages derived from this automaton have been proved [12] to be hard to learn with common recurrent architectures due to the existence of long-term dependencies [1] (see also [6]).

The graph in Fig. 1 illustrates the CR for this sequence (20000 symbols long, $|\Sigma| = 7$). The CR with the $[0,4]$-gram model is 4.23. As can be seen, the number of state neurons n_X affects the CR attained, although for values of $n_X \geq 10$ the influence is not very significant. Both architectures, REPN and SRN, give comparable results. The DEKF training algorithm gives CR near those of the $[0,4]$-gram model, whereas those of RTRL are lower.

Chaotic Sequences. We show the results for a symbolic sequence composed of the activation measures of a chaotic laser [13]. Figure 2 shows the CR for this sequence (length 10000, $|\Sigma| = 4$). The $[0,4]$-gram model gives a CR of 2.73.

When using RTRL, differences between REPN and SRN are bigger than before. Again, the DEKF surpasses the results of RTRL. Anyway, RTRL and the REPN give CR (for $n_X > 4$) similar to those of the $[0,4]$-gram model, and the DEKF (independently of the architecture) attains much higher ones (near 4).

Human Language Texts. Finally, we considere an essay in English about Polish film director Krzysztof Kieslowski. Figure 3 illustrates the results for this sequence in human language (length 62648, $|\Sigma| = 27$). The CR with the $[0,4]$-gram model is 1.85.

As can be seen, the networks are not capable of developing a useful state representation of sequence dynamics. The influence of n_X on the prediction

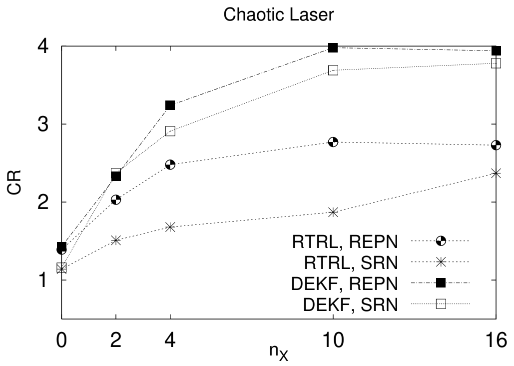

Fig. 2. Compression ratios for the chaotic laser sequence. The $[0, 4]$-gram model gives a CR of 2.73

with the REPN and both training algorithms is not noticeable, although DEKF consistently gives better results. The SRN partially overcomes this limitation, although the CR are lower than those of the REPN. The results of the DEKF are once more slightly better than the results of RTRL, but they are still far lower than those achieved with the $[0, 4]$-gram model.

5 Concluding Remarks

The DEKF training algorithm outperforms RTRL when predicting online the symbols in a sequence. It has to be pointed out, however, that the DEKF has greater complexity because it has to compute the same number of derivatives plus some matrix operations (including an inversion) at every time step.

In the case of the regular sequence the results of the DEKF are comparable to those obtained with n-grams. When processing the chaotic sequence, even the results obtained with RTRL are similar to those of an $[0, 4]$-gram probability model, and the DEKF clearly outperforms it.

When dealing with sequences in human language, however, DEKF results are much worse than $[0, 4]$-gram results. Prediction over texts in human language seems to be a difficult task for RTRL and the DEKF. It has to be noted, anyway, that the number of free parameters used in a $[0, 4]$-gram model is far greater than the number of parameters in the recurrent neural networks used in this paper.

A more detailed and theoretical study has to be carried out in order to analyse how a DTRNN approximates symbol probabilities when working online. We have found that acceptable approximations are possible for regular or simple chaotic sequences and that the problem is far more arduous with more complex sequences. Besides that, it would be very interesting to analyse the evolution of the network state while processing the different kind of sequences used in this paper.

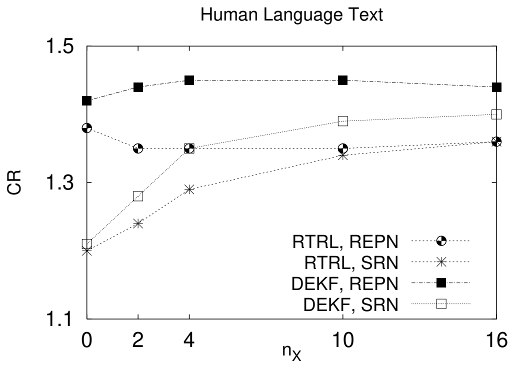

Fig. 3. Compression ratios for the natural language sequence. The $[0, 4]$-gram model gives a CR of 1.85

References

1. Bengio, Y., P. Simard and P. Frasconi (1994), "Learning long-term dependencies with gradient descent is difficult", *IEEE Transactions on Neural Networks*, **5**(2).
2. Carrasco, R. C. et al. (2000), "Stable-encoding of finite-state machines in discrete-time recurrent neural nets with sigmoid units", *Neural Computation*, **12**(9).
3. Cleeremans, A., D. Servan-Schreiber and J. L. McClelland (1989), "Finite state automata and simple recurrent networks", *Neural Computation*, **1**(3), 372–381.
4. Elman, J. L. (1990), "Finding structure in time", *Cognitive Science*, **14**, 179–211.
5. Haykin, S. (1999), *Neural networks: a comprehensive foundation*, Chapter 15, Prentice-Hall, New Jersey, 2nd edition.
6. Hochreiter, J. (1991), *Untersuchungen zu dynamischen neuronalen Netzen*, Diploma thesis, Institut für Informatik, Technische Universität München.
7. Nelson, M. (1991), "Arithmetic coding + statistical modeling = data compression", *Dr. Dobb's Journal*, Feb. 1991.
8. Pérez-Ortiz, J. A., J. Calera-Rubio, M. L. Forcada (2001), "Online text prediction with recurrent neural networks", *Neural Processing Letters*, in press.
9. Puskorius, G. V. and L. A. Feldkamp (1991), "Decoupled extended Kalman filter training of feedforward layered networks", in *International Joint Conference on Neural Networks*, volume 1, pp. 771–777.
10. Robinson, A. J. and F. Fallside (1991), "A recurrent error propagation speech recognition system", *Computer Speech and Language*, **5**, 259–274.
11. Schmidhuber, J. and S. Heil (1996), "Sequential neural text compression", *IEEE Transactions on Neural Networks*, **7**(1), pp. 142-146.
12. Smith, A. W. and D. Zipser (1989), "Learning sequential structures with the real-time recurrent learning algorithm", *International Journal of Neural Systems*, **1**(2).
13. Tiňo, P., M. Köteles (1999), "Extracting finite-state representations from recurrent neural networks trained on chaotic symbolic sequences", *IEEE Transactions on Neural Networks*, **10**(2), pp. 284–302.
14. Williams, R. J. and R. A. Zipser (1989), "A learning algorithm for continually training recurrent neural networks", *Neural Computation*, **1**, 270–280.