Efficient encodings of finite automata in discrete-time recurrent neural networks*

Rafael C. Carrasco, Jose Oncina, and Mikel L. Forcada

Departament de Llenguatges i Sistemes Informàtics,
Universitat d'Alacant,
E-03071 Alacant, Spain.
E-mail: {carrasco,oncina,mlf}@dlsi.ua.es

Abstract

A number of researchers have used discrete-time recurrent neural nets (DTRNN) to learn finite-state machines (FSM) from samples of input and output strings; trained DTRNN usually show FSM behaviour for strings up to a certain length, but not beyond; this is usually called *instability.* Other authors have shown that DTRNN may actually behave as FSM for strings of any length and have devised strategies to construct such DTRNN. In these strategies, m-state *deterministic* FSM are encoded and the number of state units in the DTRNN is $\Theta(m)$. This paper shows that more efficient sigmoid DTRNN encodings exist for a subclass of deterministic finite automata (DFA), namely, when the size of an equivalent nondeterministic finite automata (NFA) is smaller, because n-state NFA may directly be encoded in DTRNN with a $\Theta(n)$ units.

1 Introduction

The relationship between discrete-time recurrent neural networks (DTRNN) and finite-state machines (FSM) has very deep roots (McCulloch & Pitts 1943; Kleene 1956; Minsky 1967). These early papers show the equivalence of DTRNN with threshold linear units (TLU), having step-like transfer functions, and some classes of FSM, and use constructions which have a number of TLU which is linear with the number of states in the FSM. However, n units may be in 2^n different states, which suggests that it should be possible to encode m-state FSM in DTRNN having $O(\log m)$ units, by using a distributed encoding of FSM states. Following this suggestion, Alon *et al.* (1991) (for first-order single-layer DTRNN) and Horne & Hush (1996) (for first-order lower-triangular DTRNN) have obtained better bounds on the number of TLU, but the lower bound is never as good as $O(\log m)$.

DTRNN may also be constructed using real-valued sigmoid units. These DTRNN may be in a nondenumerably infinite number of states at any time t, and so, they may be expected to be capable of behaving as FSM. Indeed, a number of researchers have used sigmoid DTRNN to learn FSM behaviour from samples (Cleeremans *et al.* 1989; Pollack 1991; Giles *et al.* 1992; Watrous & Kuhn 1992; Maskara & Noetzel 1992; Sanfeliu & Alquézar 1994; Manolios & Fanelli 1994; Forcada & Carrasco 1995; Ñeco & Forcada 1996; Gori *et al.* 1998). DTRNN are successfully trained to behave as FSM for short input strings, but this behaviour is not stable for longer inputs. This has motivated a number of researchers (Omlin & Giles 1996; Kremer 1996; Carrasco *et al.* 1998; Kremer *et al.* 1998) to prove, by giving a constructive proof, that DTRNN may indeed stably behave as FSM for strings of any length. All of the constructions may be used to encode deterministic finite automata (DFA, a class of FSM used as language acceptors) and use $\Theta(m)$ neurons for m-state DFA. In particular, if Σ is the input alphabet, first-order constructions require $m|\Sigma| + 1$ units and second-order constructions require $m + 1$ units.

*Work supported through grant TIC97-0941 of the Spanish CICyT.

As with threshold DTRNN, one may wonder whether it would be possible to find more efficient encodings. Experimental results (see, e.g. Giles *et al.* (1992)) indeed show that second-order DTRNN may learn languages whose minimal DFA has m states with less than m units.

It is the case that some regular languages are more efficiently represented (in terms of number of states) by NFA (nondeterministic finite automata) than by DFA. Consider, for example, the family of languages L_k represented by the regular expressions $r_k = (0(0|1)^*0(0|1)^k)|(1(0|1)^*1(0|1)^k)$ (the languages of binary strings w whose $(|w|-k)$-th bit is the same as the first). It is easy to construct a NFA N_k having $k+4$ states for each L_k, that is, a number of states that is linear with k. However, it is not difficult to show that the size of the corresponding minimal DFA grows exponentially with k.

In this paper, we show that any NFA having m states may be stably encoded in a second-order DTRNN (Giles *et al.* 1992; Watrous & Kuhn 1992; Pollack 1991) having $m+1$ states. As a result, any m'-state DFA representing a regular language for which a NFA having $m < m'$ states exists may be encoded in second-order DTRNN more efficiently than with the currently available schemes (Omlin & Giles 1996; Carrasco *et al.* 1998). Section 2 describes the second-order DTRNN architecture that will be used and defines nondeterministic finite automata. The encoding scheme is defined and justified in section 3. Finally, concluding remarks may be found in section 4.

2 Definitions

Nondeterministic finite automata: A nondeterministic finite automaton (NFA) is a five-tuple $M = (Q, \Sigma, \delta, q_I, F)$ where $Q = \{q_1, q_2, \ldots, q_{|Q|}\}$ is a finite set of *states*, $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_{|\Sigma|}\}$ is a finite *input alphabet*, $\delta : Q \times \Sigma \to 2^Q$ is the *next-state function*, such that a machine in state q_j, after reading symbol σ_k, may move to any of the states q_i in the set $\delta(q_j, \sigma_k) \subseteq Q$; $q_I \in Q$ is the *initial state* in which the machine is found before the first symbol of the input string is processed, and $F \subseteq Q$ is the set of *accepting states* (the NFA accepts those strings leading to at least one state in F).

Second-order DTRNN: The single-layer second-order DTRNN architecture used in this paper is the same used by Omlin & Giles (1996) and one of the architectures used by Carrasco *et al.* (1998). The network has n_X state units whose values at time t will be represented $x_i[t]$ $(i = 1, 2, \ldots, n_X)$, and reads n_U inputs whose values at time t are $u_i[t]$ $(i = 1, 2, \ldots, n_U)$, and a single output unit, whose value at time t is $y[t]$. In each cycle, the network computes its next state and output from the previous state and the current input as follows:

$$x_i[t] = g\left(\sum_{j=1}^{n_X}\sum_{k=1}^{n_U} W^{xxu}_{ijk} x_j[t-1] u_k[t] + W^x_i\right), \tag{1}$$

and

$$y[t] = g\left(\sum_{j=1}^{n_X}\sum_{k=1}^{n_U} W^{yxu}_{jk} x_j[t-1] u_k[t] + W^y\right). \tag{2}$$

where $g : \mathcal{R} \to [S_0, S_1]$ with $S_1 > S_0 \geq 0$ is a sigmoid function[1], W^{xxu}_{ijk} and W^{yxu}_{jk} are real weights, and W^x_i and W^y are real biases.

3 Encoding scheme

DTRNN as string acceptors: DTRNN may be used as string acceptors by choosing an appropriate encoding for inputs and an interpretation for the output. The usual choice for the input consists in taking $n_U = |\Sigma|$ and using an exclusive (one-hot) encoding for inputs, so that, when symbol σ_i is input at time t, all $u_j[t] = 0$ except for $u_i[t] = 1$. The usual interpretation for the output (which is only examined after reading the whole string) is that a high output means acceptance and a low output means rejection.

High, low, and forbidden values: Two special values, $\epsilon_0, \epsilon_1 \in [S_0, S_1]$ will be defined so that the outputs of all units will be taken to be *low* if they are in $[S_0, \epsilon_0]$, *high* if they are in $[\epsilon_1, S_1]$ and forbidden otherwise.

[1]The usual choice for $g(x)$ is the logistic function $g_L(x) = 1/(1+\exp(-x))$ which has $S_0 = 0$ and $S_1 = 1$, but the treatment in this paper applies to any g which is continuous, bounded and grows monotonously.

Encoding of NFA in DTRNN: The DTRNN will have $n_X = |Q|$ state units and will be interpreted as being in state $q_i \in Q$ at time t if $x_i[t]$ is high, and as not being in state $q_j \in Q$ if $x_j[t]$ is low. In this way, the DTRNN, even if it is a deterministic device, may be interpreted as being in more than one state of the NFA. Accordingly, the initial states $x_i[0]$ are all chosen to be equal to S_0 except for $x_I[0]$, which will be S_1. What remains is defining a set of weights such that the dynamics of states in the DTRNN mimics that of the FSM for all values of $t > 0$ (the above choice of $\mathbf{x}[0]$ ensures correct behaviour at $t = 0$).

We will use a weight scheme similar to that used by Omlin & Giles (1996) and Carrasco *et al.* (1998): $W^{xxu}_{ijk} = H$ if $q_i \in \delta(q_j, \sigma_k)$ and zero otherwise; $W^{yxu}_{jk} = H$ if $\delta(q_j, \sigma_k) \cap F \neq \emptyset$ and zero otherwise, all $W^x_i = -H/2$, and $W^y = -H/2$, with H a positive value to be determined in conjunction with ϵ_0 and ϵ_1 so that DTRNN states and outputs correctly represent FSM states[2].

Let us study a typical transition $\delta(q_j, \sigma_k)$; on the one hand, we want all $x_i[t]$ such that $q_i \in \delta(q_j, \sigma_k)$ to be high whenever both $x_j[t-1]$ is high and $u_k[t] = 1$ and all other $x_{i'}[t]$ to be low. Following eq. (1), the new state of unit i is given by $x_i[t] = g\left(\sum_{l \in C_{ik}} H x_l[t-1] - \frac{H}{2}\right)$, with $C_{ik} = \{l : q_i \in \delta(q_l, \sigma_k)\}$, and, obviously, $j \in C_{ik}$. To ensure $x_i[t] \geq \epsilon_1$ even in the *worst* case, when $x_j[t-1]$ contributes with the lowest possible high signal ($x_j[t-1] = \epsilon_1$) and the rest of the $x_l[t-1]$ contribute with the strongest possible low signal for all valid sigmoid functions[3], that is, $S_0 = 0$, we have

$$g\left(H\left(\epsilon_1 - \frac{1}{2}\right)\right) \geq \epsilon_1. \qquad (3)$$

To ensure a low state for the rest of state units, $i' : q_{i'} \notin \delta(q_j, \sigma_k)$, we must consider that some of the low signals at time $t-1$ may be "weak" (far from S_0 and closer to ϵ_0) and raise the value of $x_{i'}[t]$, since there may exist states q_l such that $q_{i'} \in \delta(q_l, \sigma_k)$, which prescribe nonzero weight values $W^{xxu}_{i'lk} = H$ (these are the states in $C_{i'k}$). Defining $\chi_x = \max_{j,k} |C_{jk}|$, as the size of the biggest of such sets or *maximum fan-in* of each state, the worst case occurs when χ_x low signals have the highest possible low value (ϵ_0) and contribute through a weight H to weaken (increase) the desired low signal for $x_{i'}[t]$. Using $x_{i'}[t] = g\left(\sum_{l \in C_{i'k}} H x_l[t-1] - \frac{H}{2}\right)$ the worst case translates into

$$g\left(H\left(\chi_x \epsilon_0 - \frac{1}{2}\right)\right) \leq \epsilon_0, \qquad (4)$$

an equation that may be used to guarantee that the states $\mathbf{x}_{i'}[t]$ are low.

A similar reasoning aimed at guaranteeing a correct value for the output $y[t]$ of the DTRNN yields two conditions: one identical to eq. (3), and the other one given by:

$$g\left(H\left(\chi_y \epsilon_0 - \frac{1}{2}\right)\right) \leq \epsilon_0, \qquad (5)$$

with $\chi_y = \max_k |D_k|$, the maximum fan-in of the output function and $D_k = \{l : \delta(q_l, \sigma_k) \cap F \neq \emptyset\}$.

Therefore, if one can find ϵ_0 and ϵ_1 such that $S_0 < \epsilon_0 < \epsilon_1 < S_1$ and such that conditions (3), (4), and (5) are met, then the second-order DTRNN behaves as the corresponding NFA. Conditions (4) and (5) may be ensured by a more stringent condition,

$$g\left(H\left(\chi \epsilon_0 - \frac{1}{2}\right)\right) \leq \epsilon_0, \qquad (6)$$

with $\chi = \max(\chi_x, \chi_y)$ (note that always $\chi \leq n_X$). It has to be remarked that the conditions derived here are sufficient but not necessary —due to the use of worst cases that may not occur in general, and to the merging of conditions into more stringent ones—, and thus, smaller values of H, larger values of ϵ_0 and smaller values of ϵ_1 may still be adequate for the second-order DTRNN to behave as the corresponding NFA.

If g is the logistic function $g_L(x) = 1/(1 + \exp(-x))$, such values are easily found and it is also the case that a *minimum* positive value of H can be found for each fan-in χ. Clearly, reducing H increases ϵ_0 and reduces ϵ_1.

These values are easily obtained by turning condition (6) into an equation, solving for H, and taking $\partial H / \partial \epsilon_0 = 0$. The resulting equation may be iteratively solved for ϵ_0 using

$$\epsilon_0[t+1] = \left(\frac{g_L^{-1}(\epsilon_0[t])}{\epsilon_0[t] - \frac{1}{2\chi}} - \frac{1}{1 - \epsilon_0[t]}\right) \qquad (7)$$

[2]This encoding uses a finite weight alphabet, and thus, it may not be expected to optimal, as discussed by Alon *et al.* (1991).

[3]It has to be noted that this worst case may indeed occur when C_{ik} contains only j.

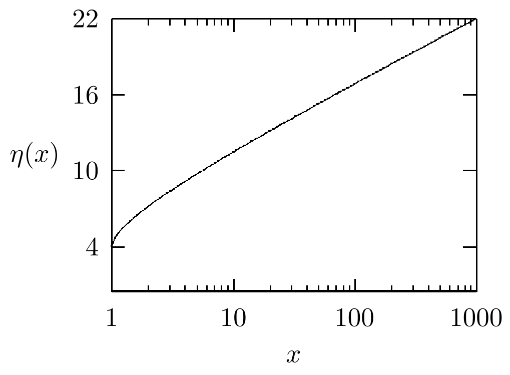

Figure 1: The function η (see text)

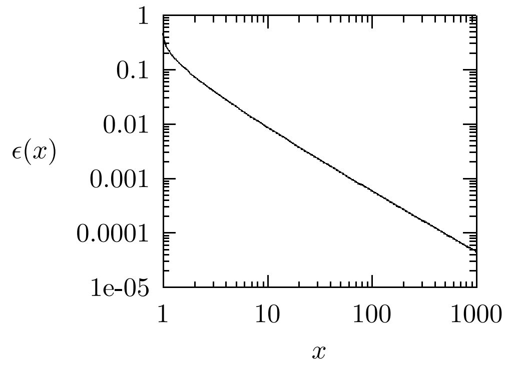

Figure 2: The function ϵ (see text)

(a few iterations starting with $\epsilon_0[0] = 1/4\chi$ are enough unless χ is very close to 1, where $H = 4^+$ $\epsilon_0 = 0.5^-$ and $\epsilon_1 = 0.5^+$). The resulting ϵ_0 is substituted in eq. (6) and (3) to get the corresponding values of H and ϵ_1. The minimum H is a function $H = \eta(\chi)$ of χ, shown in figure 1, that grows slower than $\log(\chi)$ (the ratio $\eta(\chi)/\log(\chi)$ is a decreasing function of χ). On the other hand, ϵ_0 is a decreasing function $\epsilon(\chi)$ of χ, shown in figure 2, which vanishes slightly faster than $1/\chi$.

It should be noted that the forbidden interval $]\epsilon_0, \epsilon_1[$ is rather wide, except in the case $\chi = 1$, where it is infinitesimally small.

4 Concluding remarks

We have constructively shown that nondeterministic finite automata may be directly encoded in second-order DTRNN, so that the network behaves exactly like the corresponding NFA for strings of any length. For some regular languages, it is the case that the number of states of a NFA accepting them is smaller than the number of states of the corresponding minimal DFA; in this case, an acceptor for the language may be encoded in a smaller DTRNN using the present approach. It is also the case that for some languages it is very easy to write a NFA; the current method allows to encode it directly without having to convert it first to a DFA (although it is perfectly useful for DFA too). However, no general method to devise an efficient NFA encoding for a given language exists yet.

The apparently contradictory fact that we seem to be encoding nondeterministic automata in a deterministic neural device deserves further comment: what the DTRNN simulates is a partially distributed encoding of an equivalent deterministic automaton, an encoding in which the states of a NFA may however be recognized. It would be interesting to study for how large a fraction of all regular languages there exist NFA encodings which are more efficient than the corresponding DFA encodings; this study would have to be performed with a methodology similar to those described by Alon *et al.* (1991) and Horne & Hush (1996). If that fraction does not vanish, it would imply that n-state DFA may be generally encoded in DTRNN having less than $O(n)$ units[4].

References

Alon, N., A. K. Dewdney, & T. J. Ott. 1991. Efficient simulation of finite automata by neural nets. *Journal of the Association of Computing Machinery* 38(2):495–514.

Carrasco, Rafael C., Mikel L. Forcada, M. Ángeles Valdés-Muñoz, & Ramón P. Ñeco. 1998. Stable encoding of finite-state machines in discrete-time recurrent neural nets with sigmoid units. Technical report, Departament de Llenguatges i Sistemes In-

[4]After the acceptance of this paper we have become aware of a paper (Šíma 1997) which proves that any DTRNN using threshold units may be converted into a topologically equivalent sigmoid DTRNN which shows the same behaviour for any time t; this implies that the bounds on the number of units found by Horne & Hush (1996) and Alon *et al.* (1991) are also applicable sigmoid DTRNN.

formàtics, Universitat d'Alacant, Alacant, Spain. Submitted to Neural Computation.

Cleeremans, A., D. Servan-Schreiber, & J. L. McClelland. 1989. Finite state automata and simple recurrent networks. *Neural Computation* 1(3):372–381.

Forcada, M. L., & R. C. Carrasco. 1995. Learning the initial state of a second-order recurrent neural network during regular-language inference. *Neural Computation* 7(5):923–930.

Giles, C. L., C. B. Miller, D. Chen, H. H. Chen, G. Z. Sun, & Y. C. Lee. 1992. Learning and extracted finite state automata with second-order recurrent neural networks. *Neural Computation* 4(3):393–405.

Gori, Marco, Marco Maggini, E. Martinelli, & G. Soda. 1998. Inductive inference from noisy examples using the hybrid finite state filter. *IEEE Transactions on Neural Networks* 9(3):571–575.

Horne, B. G., & D. R. Hush. 1996. Bounds on the complexity of recurrent neural network implementations of finite state machines. *Neural Networks* 9(2):243–252.

Kleene, S.C. 1956. Representation of events in nerve nets and finite automata. In *Automata Studies*, ed. by C.E. Shannon & J. McCarthy, 3–42. Princeton, N.J.: Princeton University Press.

Kremer, Stefan C., 1996. *A Theory of Grammatical Induction in the Connectionist Paradigm*. Edmonton, Alberta: Department of Computer Science, University of Alberta dissertation.

——, Ramón P. Ñeco, & Mikel L. Forcada. 1998. Constrained second-order recurrent networks for finite-state automata induction. In *Proceedings of the 8th International Conference on Artificial Neural Networks ICANN'98*, ed. by L. Niklasson, M. Bodén, & T. Ziemke, volume 2, 529–534, London. Springer.

Manolios, P., & R. Fanelli. 1994. First order recurrent neural networks and deterministic finite state automata. *Neural Computation* 6(6):1154–1172.

Maskara, Arun, & Andrew Noetzel. 1992. Forcing simple recurrent neural networks to encode context. In *Proceedings of the 1992 Long Island Conference on Artificial Intelligence and Computer Graphics*.

McCulloch, W. S., & W. H. Pitts. 1943. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* 5:115–133.

Minsky, M.L. 1967. *Computation: Finite and Infinite Machines*. Englewood Cliffs, NJ: Prentice-Hall, Inc. Ch: Neural Networks. Automata Made up of Parts.

Ñeco, R. P., & M. L. Forcada. 1996. Beyond Mealy machines: Learning translators with recurrent neural networks. In *Proceedings of the World Conference on Neural Networks '96*, 408–411, San Diego, California.

Omlin, C. W., & C. L. Giles. 1996. Constructing deterministic finite-state automata in recurrent neural networks. *Journal of the ACM* 43(6):937–972.

Pollack, Jordan B. 1991. The induction of dynamical recognizers. *Machine Learning* 7:227–252.

Sanfeliu, A., & R. Alquézar. 1994. Active grammatical inference: a new learning methodology. In *Shape and Structure in Pattern Recognition*, ed. by Dov Dori & A. Bruckstein, Singapore. World Scientific. Proceedings of the IAPR International Workshop on Structural and Syntactic Pattern Recognition SSPR'94 (Nahariya, Israel).

Šíma, Jiří. 1997. Analog stable simulation of discrete neural networks. *Neural Network World* 7:679–686.

Watrous, R. L., & G. M. Kuhn. 1992. Induction of finite-state languages using second-order recurrent networks. *Neural Computation* 4(3):406–414.