Encoding of Sequential Translators in Discrete-Time Recurrent Neural Nets

Ramón P. Ñeco

Departament de Ciències Experimentals i Tecnologia,
Universitat Miguel Hernández,
E-03202 Elx, Spain.
E-mail: Ramon.Neco@umh.es

Mikel L. Forcada, Rafael C. Carrasco, M. Ángeles Valdés-Muñoz

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

Abstract. In recent years, there has been a lot of interest in the use of discrete-time recurrent neural nets (DTRNN) to learn finite-state tasks, and in the computational power of DTRNN, particularly in connection with finite-state computation. This paper describes a simple strategy to devise stable encodings of sequential finite-state translators (SFST) in a second-order DTRNN with units having bounded, strictly growing, continuous sigmoid activation functions. The strategy relies on bounding criteria based on a study of the conditions under which the DTRNN is actually behaving as a SFST.

1. Introduction

The study of the relationship between DTRNN and finite-state machines (FSM) [1, 2, 3] is partly motivated by the fact that one can view DTRNN as state machines which are not finite: a new state for the hidden units is computed from the previous state and the currently available input in each cycle, and possibly an output is computed in each cycle too. Under this intuitive assumption a number of researchers set out to test whether DTRNN with sigmoid activation functions could learn FSM behavior from samples [2, 4, 5]. These works show that, indeed, DTRNN may learn FSM-like behavior from samples, but some problems persist: the correct behavior is observed for short input strings but for longer input strings, incorrect state representations may be obtained. This is often called *instability*. Some researchers have set out to define ways to program or *encode* a sigmoid-based DTRNN so that it behaves as a given FSM without stability problems [3, 6]. In this paper, we outline a simplified procedure to

Work supported by the Spanish Comisión Interministerial de Ciencia y Tecnología (CICyT) through grant TIC97-0941.

prove the stability of a devised encoding scheme for a special class of finite-state translators (which includes Mealy machines) in a second-order DTRNN architecture.

2. Definitions

Extended Mealy machines: Formally, an *extended Mealy machine* (EMM) is a six-tuple $M = (Q, \Sigma, \Gamma, \delta, \lambda, q_I)$ 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*; $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_{|\Gamma|}\}$ is a finite *output alphabet*; $\delta : Q \times \Sigma \cup \{\epsilon\} \to Q$ is the *next-state function*, where ϵ represents the empty string; $\lambda : Q \times \Sigma \cup \{\epsilon\} \to \Gamma \cup \{\epsilon\}$ is the *output function*; and $q_I \in Q$ is the *initial state*. In EMM, transitions can occur without consuming input or without producing output. For the machine to remain deterministic, states are allowed to have transitions defined either on ϵ (ϵ-transitions) or on input symbols but not on both. Loops with only ϵ-transitions are not allowed in these machines. A *Mealy machine* is a special case of EMM in which transitions consuming no input or producing no output are not allowed. EMM allow for a general class of sequential translations that are not *synchronous* or *length preserving*[1].

Discrete-time recurrent neural networks: DTRNN [8] may be viewed as *neural state machines* (NSM), and defined as follows: a DTRNN acting as a *neural Mealy machine* is a six-tuple $N = (X, U, Y, \mathbf{f}, \mathbf{h}, \mathbf{x}_0)$, where $X = [S_0, S_1]^{n_X}$ is the state space of the NSM, with S_0 and S_1 the values defining the range of outputs of the transfer functions, and n_X the number of state units; $U = \mathcal{R}^{n_U}$ defines the set of possible input vectors, with n_U the number of input lines; $Y = [S_0, S_1]^{n_Y}$ is the set of outputs, with n_Y the number of output units; $\mathbf{f} : X \times U \to X$ is the *next-state function*, a feed-forward neural network which computes a new state $\mathbf{x}[t]$ from the previous state $\mathbf{x}[t-1]$ and the input just read $\mathbf{u}[t]$; $\mathbf{h} : X \times U \to Y$ is the *output function*, that is, a feed-forward neural network which computes a new output $\mathbf{y}[t]$ from the previous state $\mathbf{x}[t-1]$ and the input just read $\mathbf{u}[t]$; and finally, $\mathbf{x}_0$ is the initial state, that is, the value that will be used for $\mathbf{x}[0]$. A commonly used second-order DTRNN architecture [2, 4] uses a next-state function whose i-th ($i = 1, \ldots, n_X$) coordinate is : $\mathbf{f}_i(\mathbf{x}[t-1], \mathbf{u}[t]) = g\left(\sum_{j=1}^{n_X} \sum_{k=1}^{n_U} W_{ijk}^{xxu} x_j[t-1] u_k[t]\right)$, where $g : \mathcal{R} \to [S_0, S_1]$ is a sigmoid function. The output function is $\mathbf{h}_i(\mathbf{x}[t-1], \mathbf{u}[t]) = g\left(\sum_{j=1}^{n_X} \sum_{k=1}^{n_U} W_{ijk}^{yxu} x_j[t-1] u_k[t]\right)$, $i = 1, \ldots, n_Y$. All weights are real numbers.

A neural EMM is a neural Mealy machine that has two additional output units: $y_U[t]$, the "advance input" control signal for cycle $t+1$, and $y_Y[t]$, the "output length" control signal. In each cycle, if the previous $y_U[t]$ was high, input is advanced and the symbol just read is presented to the network; if $y_U[t]$ was low, a special symbol is presented without advancing the input. If $y_Y[t]$ is high then the output is taken from the output units; if $y_Y[t]$ is low, the net has

[1] EMM perform the same transductions as deterministic *letter transducers*[7]

no output in cycle t. The values of the control units are computed in the same way as the other outputs.

When DTRNN are used to learn FSM behavior from samples [2, 4, 5], a common choice is to use continuous, real-valued activation functions for neurons in the DTRNN; this allows the construction of gradient-based learning algorithms. We will consider a general class of sigmoid functions in which any element g has the following four properties: (1) *strictly growing* ($g'(x) > 0\ \forall x \in \mathcal{R}$), (2) *continuous*, (3) *positive* ($g(x) \geq 0\ \forall x \in \mathcal{R}$), and (4) *bounded* (between S_0 and S_1, with $S_1 > S_0$): $g : \mathcal{R} \to [S_0, S_1]$. Note that as a consequence of these properties, g has an inverse[2]. In this paper, the output range of all units, $[S_0, S_1]$ will be partitioned in three subintervals, *high*: $[\epsilon_1, S_1]$, *low*: $[S_0, \epsilon_0]$, and *forbidden*: $]\epsilon_0, \epsilon_1[$, with $S_0 < \epsilon_0 < \epsilon_1 < S_1$.

3. Conditions for DTRNN to behave as EMM

Casey [9] found that a DTRNN robustly performing a finite-state computation must organize its state space in mutually disjoint, closed sets with nonempty interiors corresponding to the states of the FSM. These states are the sets of points such that if the DTRNN is initialized with any of them, it will produce the same output as the FSM initialized in the corresponding state. Following this formulation, a DTRNN N behaves as an EMM M when the following six conditions are met:

Partition of state space: Each state $q_i \in Q$ is assigned a nonempty region $X_i \subseteq X$ such that the DTRNN N is said to be in state q_i at time t when $\mathbf{x}[t] \in X_i$; these regions must be disjoint. Note that there may be points of X that are not assigned to any state. In our encoding the state q_i is assigned a region $X_i(\epsilon_0, \epsilon_1) = \{\mathbf{x} \in X | x_i \in [\epsilon_1, S_1], x_j \in [S_0, \epsilon_0], i \neq j\}$ (the regions are disjoint because $\epsilon_0 < \epsilon_1$).

Representation of input symbols: Each possible input symbol $\sigma_k \in \Sigma$ is assigned a different vector $\mathbf{u}_k \in U$. In our encoding, the j-th coordinate of $\mathbf{u}_k$, u_{jk} equals δ_{jk} (*exclusive* or *unary* encoding of inputs).

Interpretation of output: Each possible output symbol $\gamma_m \in \Gamma$ is assigned a nonempty region $Y_m \subseteq Y$ such that the DTRNN N is said to output symbol γ_m at time t when $\mathbf{y}[t] \in Y_m$; these regions must be disjoint. In our encoding scheme, they are defined by $Y_m(\epsilon_0, \epsilon_1) = \{\mathbf{y} \in Y | y_m \in [\epsilon_1, S_1], y_n \in [S_0, \epsilon_0], n \neq m\}$, for each output symbol γ_m.

Correctness of the initial state: The initial state of the DTRNN N, belongs to the region assigned to the initial state q_I: $\mathbf{x}_0 \in X_I$. In our encoding the initial state is $\mathbf{x}_0 = \{S_0 + \delta_{iI}(S_1 - S_0)\}_{i=1}^{n_X}$, correct independently of ϵ_0 and ϵ_1.

Correctness of the next state function: For any state q_j and symbol σ_k of M, the transitions performed by the DTRNN N from any point in the region X_j assigned to state q_j when symbol σ_k is presented to the network must lead to

[2]The commonly used *logistic function*: $g_L(x) = 1/(1 + \exp(-x))$, $g : \mathcal{R} \to [0, 1]$ has these properties.

points that belong to the region X_i assigned to $q_i = \delta(q_j, \sigma_k)$; in our encoding, this may be expressed as $\mathbf{f}_k(X_j(\epsilon_0, \epsilon_1)) \subseteq X_i(\epsilon_0, \epsilon_1)\ \forall q_j, \sigma_k : \delta(q_j, \sigma_k) = q_i$ where $\mathbf{f}_k(A) = \{\mathbf{f}(\mathbf{x}, \mathbf{u}_k) : \mathbf{x} \in A\}$.

Correctness of output: For any state q_j and symbol σ_k of M, the output produced by the DTRNN N from any point in the region X_j assigned to state q_j when symbol σ_k is presented to the network belongs to the region Y_m assigned to $\gamma_m = \lambda(q_j, \sigma_k)$; in our encoding scheme, this may be expressed as $\mathbf{h}_k(X_j(\epsilon_0, \epsilon_1)) \subseteq Y_m(\epsilon_0, \epsilon_1)\ \forall q_j, \sigma_k : \lambda(q_j, \sigma_k) = \gamma_m$, where $\mathbf{h}_k(A) = \{\mathbf{h}(\mathbf{x}, \mathbf{u}_k) : \mathbf{x} \in A\}$. If in cycle t, M has no output (its output symbol is ϵ) then y_Y has to be low in that cycle, and high otherwise. If in the next cycle M is in a state with only ϵ-transitions, then y_U has to be low, and high otherwise.

4. Encoding of EMM in DTRNN

To encode extended Mealy machines in the second-order DTRNN described in secion 2., we will use $n_X = |Q|$, $n_U = |\Sigma|$, and $n_Y = |\Gamma|$, and two special outputs for the control units. Now we have to define the weights that define the next-state and output functions. The weights depend on a single adjustable parameter $H > 0$.

The values of the weights for the next-state, output and control functions are defined as:

$$W^{xxu}_{ijk} = \begin{cases} H & \text{if } \delta(q_j, \sigma_k) = q_i \\ -H & \text{otherwise} \end{cases} \qquad W^{yxu}_{ijk} = \begin{cases} H & \text{if } \lambda(q_j, \sigma_k) = \gamma_i \\ -H & \text{otherwise} \end{cases}$$

$$W^{Uxu}_{jk} = \begin{cases} H & \text{if } \delta(q_j, \sigma_k) \notin Q_\epsilon \\ -H & \text{otherwise} \end{cases} \qquad W^{Yxu}_{jk} = \begin{cases} H & \text{if } \lambda(q_j, \sigma_k) \neq \epsilon \\ -H & \text{otherwise} \end{cases}$$

where Q_ϵ is the set of states having only ϵ-transitions. The weights W^{Uxu}_{jk} and W^{Yxu}_{jk} are used to compute the outputs of the control units y_U and y_Y. Using this weight scheme, it may be shown [10] that the DTRNN constructed behaves exactly as the corresponding extended Mealy machine (the next state and output function are correct) when $\epsilon_0 + \epsilon_1 = 1$ and $g\left(-H + NH\epsilon_0\right) \leq \epsilon_0$. These conditions are derived from a worst-case analysis and the general conditions in the previous section, and therefore they are sufficient but not necessary, and so, it may be the case that smaller values of H and larger values of ϵ_0 may still yield a DTRNN that has the corresponding behavior. The preceding inequality may be easily solved for ϵ_0 by taking $dH/d\epsilon_0 = 0$, and then the resulting equation may be iteratively solved to produce the values shown in table 1.

5. Experimental Results

We have performed experiments to estimate the minimum value of H needed to ensure correct encodings of randomly-generated extended Mealy machines.

N	2	3	4	5	6	7	10	30
H	2^+	3.113	3.589	3.922	4.181	4.392	4.863	6.224
ϵ_0	0.5^-	0.1220	0.0753	0.0538	0.0416	0.0336	0.0210	0.0054

Table 1: Values of weights and limiting values for encoding a EMM on a (second-order) DTRNN. Note that $\epsilon_1 = 1 - \epsilon_0$. The notation 2^+ (resp. 0.5^-) is used to represent a value arbitrarily close to 2 (resp. 0.5) from the right (resp. left).

For each machine, we looked experimentally for this value by searching for $H(L)$, the minimum value of H needed to ensure correct behavior[3] of the network for strings of length L. In the experiments, we randomly generated ten different EMM for each automaton size ($|Q| = 3, 4, 7, 10$) and computed $H(L)$ for $L = 1, 2, \ldots, 20$ to a precision of 0.01 for each FSM[4]. In Figure 1 we show $H(L)$ for some EMM. For each EMM size, the highest $H(L)$ obtained is shown, normalized with respect to the theoretical value of H (H_{th}) given in Table 1. We show two sets of experiments: the first one using the theoretical values of ϵ_0 and the second one using $\epsilon_0 = 0.5$ (no forbidden interval). As may be seen, $H(L)$ increases with L, is smaller than the theoretical upper bound $H_{\rm th}$ in Table 1, and seems to stabilize around a value, $H_{\rm exp}$, that is smaller than $H_{\rm th}$. This can be explained by the fact that we performed a worst-case analysis. This experimental value $H_{\rm exp}$ is smaller when we take $\epsilon_0 = 0.5$, that is, using no forbidden interval. The results confirm that the sufficient conditions derived can be used to encode sequential transducers.

6. Concluding Remarks

In this paper we have described a simple strategy to encode a class of sequential finite-state translators in DTRNN, and shown that this encoding ensures that the network has a stable behavior for arbitrarily long input strings. This simple approach applies to DTRNN having continuous, positive, strictly growing and bounded activation functions.

References

[1] Cleeremans, A., Servan-Schreiber, D., McClelland, J.L. (1989) *Neural Computation* 1(3):372–381.

[2] Giles, C.L., Miller, C.B., Chen, D. *et al.* (1992) *Neural Computation* 4(3):393–405.

[3] We say that the DTRNN is behaving correctly for strings of a given length when the DTRNN produces correct outputs for all strings up to length L. Note that if the network is behaving correctly for strings of length L, then it is behaving correctly for strings of any length $L' < L$.

[4] The case $|Q| = 2$ is special because an $H > 0$ gives correct behavior in our experiments.

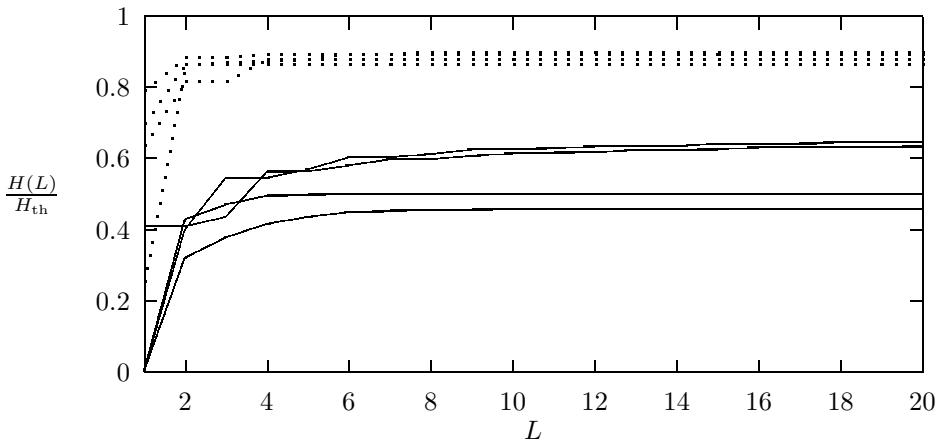

Figure 1: Normalized weight values for randomly-generated EMM of different sizes. Dotted lines: $|Q| = 3, 4, 7, 10$ and theoretical ϵ_0. Continuous lines: $|Q| = 3, 4, 7, 10$ and $\epsilon_0 = 0.5$.

[3] Kremer, S.C (1996) *A Theory of Grammatical Induction in the Connectionist Paradigm*, Edmonton, Alberta; Department of Computer Science, University of Alberta, PhD dissertation.

[4] Pollack, J.B. (1991) *Machine Learning* 7:227–252.

[5] Ñeco, R.P., Forcada, M.L. (1996) Beyond Mealy machines: Learning translators with recurrent neural networks, in *Proceedings of the World Conference on Neural Networks '96*, San Diego, California, USA.

[6] Omlin, C.W., Giles, C.L. (1996) *Neural Computation* 8:675–696.

[7] Roche, E., Schabes, Y. (1997) *Finite-State Language Processing*, Cambridge, Mass.: MIT Press; p. 17.

[8] Haykin, S. (1994) *Neural Networks, A Comprehensive Foundation*, New York, NY: Macmillan.

[9] Casey, M. (1996) *Neural Computation* 8(6):1135–1178.

[10] Carrasco, R.C., Forcada, M.L., Valdés, M.A., Ñeco, R.P. (1998) Stable encoding of finite-state machines in discrete-time recurrent neural nets with sigmoid units, submitted to *Neural Computation*.