Representation of Fuzzy Finite State Automata in Continuous Recurrent Neural Networks *

Christian W. Omlin [a], Karvel K. Thornber [a], C. Lee Giles [a,b]
[a] NEC Research Institute, Princeton, NJ 08540
[b] UMIACS, U. of Maryland, College Park, MD 20742
Phone: (609) 951-{2691,2642,2622} FAX: (609) 951-2682
e-mail: {omlinc,karvel,giles}@research.nj.nec.com

ABSTRACT

Based on previous work on encoding deterministic finite-state automata (DFAs) in discrete-time, second-order recurrent neural networks with sigmoidal discriminant functions, we propose an algorithm that constructs an augmented recurrent neural network that encodes fuzzy finite-state automata (FFAs). Given an arbitrary FFA, we apply an algorithm which transforms the FFA into an equivalent deterministic acceptor which computes the fuzzy string membership function. The neural network can be constructed such that it recognizes strings of fuzzy regular languages with arbitrary accuracy.

1. Introduction

There has been an increased interest in combining artificial neural networks and fuzzy systems (see [2] for a collection of papers). Fuzzy logic [21] provides a mathematical foundation for approximate reasoning; fuzzy logic controllers have proven very successful in a variety of applications. The parameters of adaptive fuzzy systems have clear physical meanings which facilitates the choice of their initial values. Furthermore, rule-based information can be incorporated into fuzzy systems in a systematic way. Artificial neural networks emulate on a small scale the information processing mechanisms found in biological systems which are based on the cooperation of neurons which perform simple operations and on their ability to learn from examples. Artificial neural networks have become valuable computational tools in their own right for tasks such as pattern recognition, control, and forecasting. Fuzzy systems and multilayer perceptrons are computationally equivalent, i.e. they are both universal approximators [3, 19]. Recurrent neural networks have been shown to be computationally equivalent with Turing machines [16]; whether or not recurrent fuzzy systems are also Turing equivalent remains an open question. While the methodologies underlying fuzzy systems and neural networks are quite different, their functional forms are often similar. The development of powerful learning algorithms for neural networks has been beneficial to the field of fuzzy systems which adopted some learning algorithms; e.g. there exists a backpropagation training algorithms for fuzzy logic systems which are similar to the training algorithms for neural networks [7].

A large class of problems where the current state depends on both the current input and the previous state can be modeled by finite-state automata or their equivalent grammars. It has been shown that recurrent neural networks can represent deterministic finite-state automata (DFAs) [1, 5, 6, 11]. Thus, it is only natural to investigate whether recurrent neural networks can also represent fuzzy finite-state automata (FFAs) and thus be used to implement recognizers of fuzzy regular languages.

The purpose of this paper is to show that recurrent networks that can represent DFAs can be easily modified to accommodate FFAs. Our results show that FFAs can be encoded into recurrent networks such that a constructed network assigns membership grades to strings of arbitrary length with arbitrary accuracy. Notice that we do not claim that such a representation can be *learned*.

Fuzzy grammars have been found to be useful in a variety of applications such as in the analysis of X-rays [12], in digital circuit design [10], and in the design of intelligent human-computer interfaces [14]. The fundamentals of FFAs have been in discussed in [13] without presenting a systematic method for machine synthesis. Neural network implementations of fuzzy automata have been proposed in the literature [8, 9, 18]. A general synthesis method for synchronous fuzzy sequential circuits has been discussed in [20]. A synthesis method for a class of discrete-time neural networks with multilevel threshold neurons with applications to gray level image processing has been proposed in [15].

*Published in Proceedings of *IEEE International Conference on Neural Networks*, p. 1023, IEEE Press, 1996.

2. Finite State Automata and Recurrent Neural Networks

Recurrent neural networks have been shown to be at least computationally equivalent to Turing machines [16]. Their computational power and training ability make them useful tools for modeling nonlinear dynamical systems. DFAs can be represented in many discrete-time, recurrent network architectures [1, 5, 6]. We choose for convenience networks with second-order weights W_{ijk}. The continuous network dynamics are described by the following equations:

$$S_i^{t+1} = h(a_i(t)) = \frac{1}{1+e^{-a_i(t)}}, a_i(t) = b_i + \sum_{j,k} W_{ijk} S_j^t I_k^t,$$

where b_i is the bias associated with hidden recurrent state neurons S_i; I_k denotes input neurons; h is the nonlinearity; and a_i is the activation of the ith neuron. The product $S_j^t I_k^t$ in the DFA directly corresponds to the state transition $\delta(q_j, a_k) = q_i$. After a string has been processed, the output of a designated neuron S_0 decides whether the network accepts or rejects a string. The network accepts a given string if the value of the output neuron S_0^t at the end of the string is greater than some preset value such as 0.5; otherwise, the network rejects the string. For the remainder of this paper, we assume a one-hot encoding for input symbols a_k, i.e. $I_k^t \in \{0, 1\}$.

We have recently proven that DFAs can be encoded in discrete-time, second-order recurrent neural networks with sigmoidal discriminant functions such that the DFA and constructed network accept the same regular language [11]. The desired finite-state dynamics are encoded into a network by programming a small subset of all available weights to values $+H$ and $-H$ leading to a nearly orthonormal internal DFA state representation:

Theorem 2.1 *For any given DFA M with n states and m input symbols, there exists a sparse recurrent neural network with $n+1$ sigmoidal state neurons and m input neurons can be constructed from M such that the internal state representation remains stable. Furthermore, the constructed network has at most $3mn$ second-order weights with alphabet $\Sigma_w = \{-H, 0, +H\}$, $n+1$ biases with alphabet $\Sigma_b = \{-H/2\}$, and maximum fan-out $3m$.*

The above theorem can be be proven by showing that there exists a lower bound on the magnitude of H which guarantees stable state dynamics for strings of arbitrary length. The number of weights and the maximum fan-out follow directly from the DFA encoding algorithm.

Since deterministic and fuzzy finite state automata share a common underlying structure expressed in terms of state transitions, we will be able to use the result on the stability of the network dynamics for DFAs to implement fuzzy finite-state automata.

3. Fuzzy Finite State Automata

We begin by defining the class of fuzzy automata for which we develop a synthesis method for recurrent neural networks:

Definition 3.1 *A fuzzy regular grammar $\widetilde{G}$ is a quadruple $\widetilde{G} =< S, N, T, P >$ where S is the start symbol, N and T are non-terminal and terminal symbols, respectively, and P are productions of the form $A \xrightarrow{\theta} a$ or $A \xrightarrow{\theta} aB$ where $A, B \epsilon N$, $a \epsilon T$ and $0 \leq \theta \leq 1$.*

Definition 3.2 *Given a regular fuzzy grammar $\widetilde{G}$, the membership grade $\mu_G(x)$ of a string $x \in T$ in the regular language $L(\widetilde{G})$ is the maximum value of any derivation of x, where the value of a specific derivation of x is equal to the minimum weight of the productions used:*

$$\mu_G(x) = \mu_G(S \overset{*}{\Rightarrow} x) = \max_{S \overset{*}{\Rightarrow} x} \min[\mu_G(S \rightarrow \alpha_1), \mu_G(\alpha_1 \rightarrow \alpha_2), \ldots, \mu_G(\alpha_m \rightarrow x)]$$

This is akin to the definition of stochastic regular languages where the min-and max-operators are replaced by the product- and sum-operators, respectively.

Definition 3.3 *A fuzzy finite state automaton (FFA) $\widetilde{M}$ is a 6-tuple $\widetilde{M} =< \Sigma, Q, Z, R, \delta, \omega >$ where Σ is the input alphabet, Q is a set of fuzzy states, Z is a finite output alphabet, R is the fuzzy initial state, $\delta : \Sigma \times Q \times [0, 1] \rightarrow Q$ is the fuzzy transition map and $\omega : Q \rightarrow Z$ is the output map.*

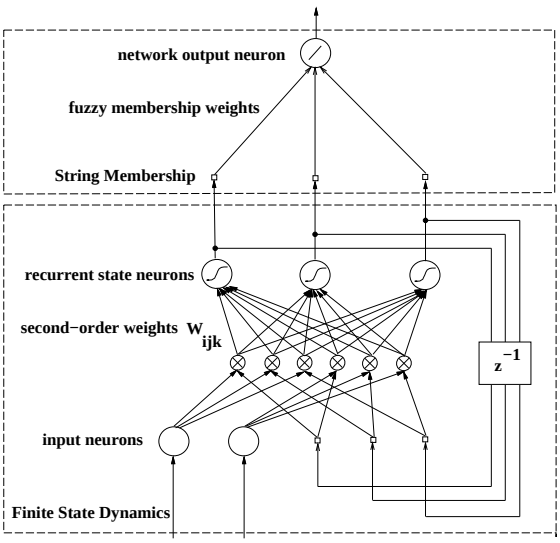

Fig. 1: **Recurrent Network Architecture for Fuzzy Finite State Automata:** The architecture consists of two parts: Recurrent state neurons with second-order weights implement the finite-state dynamics of the deterministic acceptor. The state neurons are connected via weights to a linear output neuron. The value of the weights is equal to the labels μ_i of states q_i of the deterministic acceptor.

In this paper, we consider a restricted type of fuzzy automaton whose initial state is not fuzzy, and ω is a function from F to Z, where F is a non fuzzy subset of states, called *final states*. Any fuzzy automaton as described in definition 3.3 is equivalent to a restricted fuzzy automaton [4]. [1]

There exists a correspondence between FFAs and fuzzy regular grammars [4]:

Theorem 3.1 *For a given fuzzy grammar $\widetilde{G}$, there exists a fuzzy automaton $\widetilde{M}$ such that $L(\widetilde{G}) = L(\widetilde{M})$.*

Our goal is to use only continuous (sigmoidal and linear) discriminant functions for the neural network implementation of FFAs. The following results greatly simplifies the encoding of FFAs in recurrent networks with continuous discriminant functions:

Theorem 3.2 *Given a regular fuzzy grammar $\widetilde{G}$, there exists a deterministic finite state automaton M with output alphabet $Z \subseteq \{\theta : \theta \text{ is a production weight}\} \cup \{0\}$ which computes the membership function $\mu : \Sigma^* \rightarrow [0,1]$ of the language $L(\widetilde{G})$.*

The constructive proof can be found in [17]. An immediate consequence of this theorem is the following corollary:

Corollary 3.1 *Given a regular fuzzy grammar $\widetilde{G}$, there exist an equivalent unambiguous grammar G in which productions have the form $A \stackrel{1.0}{\rightarrow} aB$ or $A \stackrel{\theta}{\rightarrow} a$.*

Thus, all states q_i in the deterministic acceptor M are assigned a label $0 \leq \mu_i \leq 1$ such that $\mu_G(x) = \mu_i$ if $\delta^*(x, R, \cdot) = q_i$.

4. Network Architecture for Fuzzy Automata

Theorem 3.2 enables us to transform any FFA into a deterministic automaton which computes the same membership function $\mu : \Sigma^* \rightarrow [0,1]$. We just need to demonstrate how to implement the computation of μ with continuous discriminant functions. For that purpose, we augment the network architecture used for encoding DFAs with additional weights which connect the recurrent state neurons to a linear output neuron. The recurrent neurons shown in figure 1 implement the desired finite state dynamics (see also theorem 2.1). The weights connecting the recurrent state neurons with the linear output neuron are just the memberships assigned to the DFA states after the transformation of a FFA into an equivalent DFA.

[1] Notice that FFAs reduce to DFAs when $Z = \{0,1\}$, and the weights of all transitions are set to 1. This leaves open the possibility of non-deterministic finite-state automata (NDFAs). However, for each NDFA, there exists a DFA which accepts the same language: $L(M_{NDFA}) = L(M_{DFA})$. Thus, states q_i in DFAs with $\omega(q_i) = 1$ and $\omega(q_i) = 0$ are labeled accepting and rejecting states, respectively.

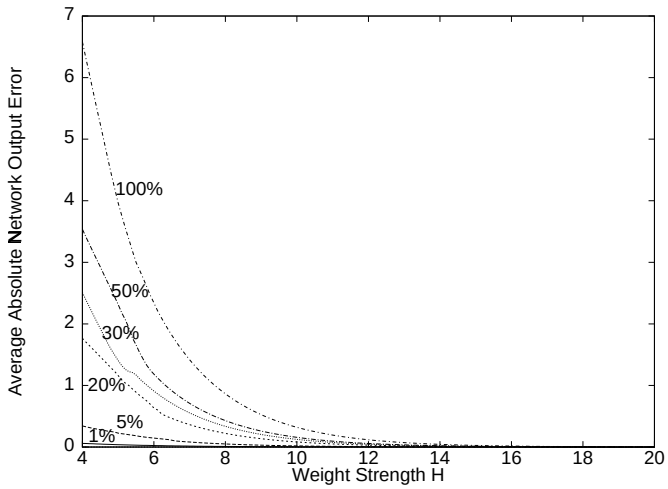

Fig. 2: **Network Performance**: The graphs show average absolute error of the network output when tested on 100 randomly generated strings of length 100 as a function of the weight strength H used to encode the finite state dynamics of randomly generated DFAs with 100 states. The percentages of DFA states with $\mu_i > 0$ were 1%, 5%, 20%, 30%, 50% and 100% respectively, of all DFA states.

Let μ_i denote the graded memberships assigned to DFA states q_i. In the worst case, the network computes for a given string the fuzzy membership function

$$\mu_{RNN} = \mu_i \, \phi^+ + (n_{acc} - 1) \, \phi^-$$

where n_{acc} is the number of DFA states with $\mu_i > 0$.

Since ϕ^- and ϕ^+ converge toward 0 and 1, respectively for increasing values of H, μ_{RNN} converges toward μ_i. Notice that $|\mu_{RNN} - \mu_i|$ can be made arbitrarily small by increasing H.

5. Simulation Results

We randomly generated deterministic acceptors for fuzzy regular languages over the alphabet $\{0, 1\}$ with 100 states as follows: For each DFA state, we randomly generated a transition for each of the two input symbols to another state. Each accepting DFA state q_i was assigned a membership $0 < \mu_i < 1$; for all non-accepting states q_j, we set $\mu_j = 0$. We encoded these acceptors into recurrent networks with 100 recurrent state neurons, two input neurons (one for each of the two input symbols 0 and 1), and one linear output neuron. We measured their performance on 100 randomly generated strings of length 100 whose membership was determined from their deterministic acceptors. The graphs in figure 2 show the average absolute error of the network output as a function of the weight strength H used to encode the finite state dynamics for DFAs where 1%, 5%, 20%, 30%, 50% and 100% of all states had labels $0 < \mu_i < 1$. We observe that the error exponentially decreases with increasing weight strength H, i.e. the average output error can be made arbitrarily small. The value of H for which the dynamics of all six DFAs remains stable for strings of arbitrary length is approximately $H \approx 9.8$.

6. Conclusions

We have proposed a method for representing fuzzy finite state automata (FFAs) in recurrent neural networks with continuous discriminant functions. Based on a previous result on encoding stable representations of finite state dynamics in recurrent networks, we have shown how FFAs can be encoded in recurrent networks that compute string membership functions with arbitrary accuracy. The method uses an algorithm which transforms FFAs into equivalent DFAs which compute fuzzy string membership. A membership label μ_i with $0 < \mu_i \leq 1$ is associated with each accepting DFA state; nonaccepting DFA states have label $\mu_i = 0$. The membership of a string is equal to the membership label of the last visited DFA state.

A recurrent network is constructed from the original architecture used for DFA encodings by connecting the recurrent state neurons to a linear output neuron. The weights of these connections are set to the value of the membership labels of the DFA states. The accuracy of the computation of the string membership function depends on the network size, the number of DFA states which membership label $\mu_i > 0$, and the weight strength H used to encode the finite state dynamics in the recurrent network. The larger H is chosen,

the more accurate the network computes membership functions.

An interesting question is whether representations of FFAs can be *learned* through training on example strings and how weighted production rules are represented in trained networks. Such insight may lead to a more direct encoding of FFAs in recurrent networks without the additional step of transforming FFAs into equivalent DFAs which compute the same string membership functions, i.e. a *fuzzy representation* of states and outputs. This may lead to smaller analog VLSI implementations of finite state controllers.

One problem with training fully recurrent networks with sigmoidal discriminant functions to behave like FFAs is the instability of learning algorithms based on gradient descent, i.e. it can become very difficult to train sigmoidal neurons to target values which are outside of the saturated regions of the discriminant function. This suggests the use of continuous multilevel threshold neurons [15] which also have the potential for stable internal DFA state representations. Whether training such networks is feasible remains an open question.

References

[1] R. Alquezar and A. Sanfeliu, "An algebraic framework to represent finite state machines in single-layer recurrent neural networks," *Neural Computation*, vol. 7, no. 5, p. 931, 1995.

[2] J. Bezdek, ed., *IEEE Transactions on Neural Networks – Special Issue on Fuzzy Logic and Neural Networks*, vol. 3. IEEE Neural Networks Council, 1992.

[3] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, pp. 303–314, 1989.

[4] D. Dubois and H. Prade, *Fuzzy sets and systems: theory and applications*, vol. 144 of *Mathematics in Science and Engineering*, pp. 220–226. Academic Press, 1980.

[5] P. Frasconi, M. Gori, M. Maggini, and G. Soda, "Representation of finite state automata in recurrent radial basis function networks," *Machine Learning*, 1995. In press.

[6] C. Giles, C. Miller, D. Chen, H. Chen, G. Sun, and Y. Lee, "Learning and extracting finite state automata with second-order recurrent neural networks," *Neural Computation*, vol. 4, no. 3, p. 380, 1992.

[7] V. Gorrini and H. Bersini, "Recurrent fuzzy systems," in *Proceedings of the Third IEEE Conference on Fuzzy Systems*, vol. I, pp. 193–198, 1994.

[8] J. Grantner and M. Patyra, "Synthesis and analysis of fuzzy logic finite state machine models," in *Proceedings of the Third IEEE Conference on Fuzzy Systems*, vol. I, pp. 205–210, 1994.

[9] S. Lee and E. Lee, "Fuzzy neural networks," *Mathematical Biosciences*, vol. 23, pp. 151–177, 1975.

[10] S. Mensch and H. Lipp, "Fuzzy specification of finite state machines," in *Proceedings of the European Design Automation Conference*, pp. 622–626, 1990.

[11] C. Omlin and C. Giles, "Stable encoding of large finite-state automata in recurrent neural networks with sigmoid discriminants," *Neural Computation*, 1996. Accepted for publication.

[12] A. Pathak and S. Pal, "Fuzzy grammars in syntactic recognition of skeletal maturity from x-rays," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, no. 5, pp. 657–667, 1986.

[13] E. Santos, "Maximin automata," *Information and Control*, vol. 13, pp. 363–377, 1968.

[14] H. Senay, "Fuzzy command grammars for intelligent interface design," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 5, pp. 1124–1131, 1992.

[15] J. Si and A. Michel, "Analysis and synthesis of a class of discrete-time neural networks with multilevel threshold neurons," *IEEE Transactions on Neural Networks*, vol. 6, no. 1, p. 105, 1995.

[16] H. Siegelmann and E. Sontag, "On the computational power of neural nets," *Journal of Computer and System Sciences*, vol. 50, no. 1, pp. 132–150, 1995.

[17] M. Thomason and P. Marinos, "Deterministic acceptors of regular fuzzy languages," *IEEE Transactions on Systems, Man, and Cybernetics*, no. 3, pp. 228–230, 1974.

[18] F. Unal and E. Khan, "A fuzzy finite state machine implementation based on a neural fuzzy system," in *Proceedings of the Third International Conference on Fuzzy Systems*, vol. 3, pp. 1749–1754, 1994.

[19] L.-X. Wang, "Fuzzy systems are universal approximators," in *Proceedings of the First International Conference on Fuzzy Systems*, pp. 1163–1170, 1992.

[20] T. Watanabe, M. Matsumoto, and M. Enokida, "Synthesis of synchronous fuzzy sequential circuits," in *Proceedings of the Third IFSA World Congress*, pp. 288–291, 1989.

[21] L. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, pp. 338–353, 1965.