Mathware & Soft Computing 5 (1998) 133-140

Fuzzy Grammatical Inference using Neural Network

A. Blanco, A. Delgado and M.C. Pegalajar
Depto. de Ciencias de la Computación e Inteligencia Artificial
E.T.S.I. Informática. Universidad de Granada
Av. de Andalucía, 38, 18071 Granada (Spain)
e-mail: mcarmen@decsai.ugr.es

Abstract

We have shown a model of fuzzy neural network that is able to infer the relations associated to the transitions of a fuzzy automaton from a fuzzy examples set. Neural network is trainned by a backpropagation of error based in a smooth derivative [1]. Once network has been trainned the fuzzy relations associated to the transitions of the automaton are found encoded in the weights.

Keywords: Fuzzy Automata, Fuzzy Recurrent Neural Network, Fuzzy Grammatical Inference

1 Introduction

1.1 Grammars, Grammatical Inference and Recurrent Neural Networks

In the last years, grammatical inference using recurrent neural network has been a widely dealed with problem. In particular, it has been showed that recurrent neural network (RNN) are capable of inferring regular grammars [2, 4, 5, 10]. During the trainning, the recurrent network partitions its space of states in well defined regions that represent the correspondent states of deterministic finte automaton (DFA) which is tried to learn. However, fuzzy grammatical inference using neural network is an open problem yet. In this paper, we present a model of fuzzy recurrent neural network. by means of which it can be inferred, from sequences with membership degrees to states of automaton, the fuzzy grammar associated. The scheme of this work is the following: firstly, it will be described the architecture of the proposed fuzzy neural network; next, we will see the trainning procedure and how automaton is extracted. Finally, obtained results form the simulation of a specific example will be provided.

1.2 Fuzzy Grammars and Fuzzy Grammatical Inference

Fuzzy Regular Grammar $\mathcal{G}$ is a quadruple $\mathcal{G} =< N, T, P, S >$ where N and T are sets of non-terminal and terminal symbols, respectively . P is a set of productions in the form $A \xrightarrow{\theta} a$ or $A \xrightarrow{\theta} aB$ where $A, B \in N$, $a \in T$ and $\theta \in [0, 1]$, y S is the symbol which generates the grammar.

In a fuzzy grammar, $\mathcal{G}$, the sequences belong to the language generated by the grammar with a degree of membership. The degree of membership $\mu_G(x)$ of a sequence $x \in T^*$, is the maximum value of all the values of the possible derivations of x, where the value of a specific derivation of x is the minimum degree of membership of the production used:

$$\mu_G(x) = \mu_{G \overset{*}{\Longrightarrow} x} = \max_{S \overset{*}{\Rightarrow} x} \min[\mu_G(S \to \alpha_1), \mu_G(\alpha_1 \to \alpha_2), ..., \mu_G(\alpha_m \to x)]$$

An automaton is named fuzzy when their states are characterized by fuzzy sets and the productions of answers and following states are facilitated by the appropiated fuzzy relations. A fuzzy automaton, $\mathcal{A}$, is a fuzzy relational system defined by a quintuple $\mathcal{A} =< \mathcal{X}, \mathcal{Y}, \mathcal{Z}, \mathcal{R}, \mathcal{T} >$ where $\mathcal{X}$ is a not empty finite set of input states (input alphabet), $\mathcal{Y}$ is a not empty finite set of output states (output alphabet), $\mathcal{Z}$ is a finite set internal states, $\mathcal{R}$ is the fuzzy relation in $\mathcal{Z} \times \mathcal{Y}$ and $\mathcal{T}$ is a fuzzy relation in $\mathcal{X} \times \mathcal{Z} \times \mathcal{Z}$.

Given a fuzzy grammar $\mathcal{G}$, exists a fuzzy automaton $\mathcal{A}$ that recognizes the same language which generates the grammar $\mathcal{G}$, that is, $L(\mathcal{G}) = L(\mathcal{A})$.

Fuzzy grammatical inference [3] is defined as the problem of inferring the fuzzy grammatical rules or fuzzy productions that characterize a fuzzy grammar from a set of fuzzy examples of trainning. Due to the relation between fuzzy grammar and fuzzy automaton, the problem of fuzzy grammatical inference can be focused as the obtention of a fuzzy automaton that recognizes the set of learning examples and from which, we obtain the set of productions belonging to the grammar which generates such examples.

Our problem will be centred in obtainning the fuzzy transitions associated to the fuzzy automaton (fuzzy relations that define the automaton "movement") from couples of examples (α,β) where α is a sequence of characters to process by automaton and β is a fuzzy set which represents the membership of each states of automaton, once input string is read.

2 The Fuzzy Recurrent Neural Network

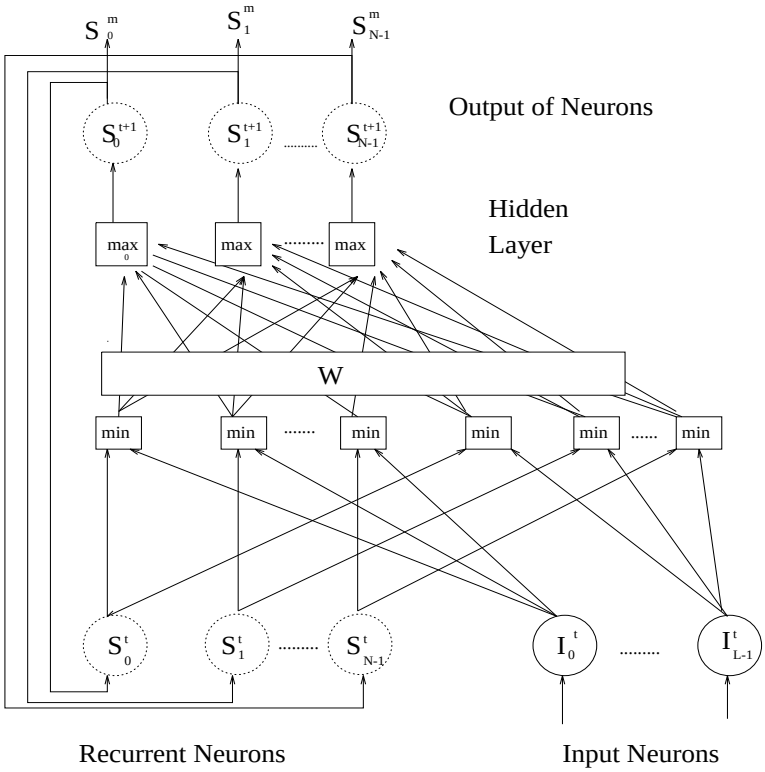

Figure 1: Max-Min Neural Network

The network we propose, Figure 1, consists of N fuzzy recurrent neurons labelled $\mathcal{S}_j$; L input neurons no-recurrent I_k and $N^2 \times L$ weights labeled W_{ijk}. This network accepts an input sequence ordered in time and its dynamic is performed in the following way:

$$\mathcal{S}_i^{t+1} = \max_{j,k} \min(W_{ijk}, S_j^t, I_k^t) \tag{1}$$

Each symbol of the input sequence is encoded in input neurons in each step of time t as follows:

a) If input character is 0, then $I_0 = 1$ and $I_1 = 0$.
b) If input character is 1, then $I_0 = 0$ and $I_1 = 1$.

Equation 1 is evaluated in each fuzzy neuron $\mathcal{S}_i$ to compute the value of following state in time $t + 1$.

Starting value, before any input was seen, is $S_0^0 = 1$ and $S_j^0 = 0$ for all $j \neq 0$.

2.1 Trainning procedure

The goal of trainning the network is to fit the weights in order to the set of input examples produce the set of wished ouputs. The error for each fuzzy neuron is computed by minimizing the square of the difference between the wished output T_i, and the obtained output by network $\mathcal{S}_i^f$ for each patterns used in the learning. $\mathcal{S}_i^f$ will be, hence, the value of each neuron, once the presentation of an example sequence has finished. We will compute the error by using the following expression:

$$E_i = \frac{1}{2}(T_i - \mathcal{S}_i^f)^2 \tag{2}$$

where

$$\mathcal{S}_i^f = \max_{j,k} \min(W_{ijk}, S_j^{f-1}, I_k^{f-1}) \tag{3}$$

being T_i the degree of membership to state i of automaton $\mathcal{A}$ and $\mathcal{S}_i^f$ the degree of membership to state j that network gives as result .

To fit weights, we use a trainning algorithm that updates the weights at the end of the presentation of each examples sequence. We will update the weights if error is greater than ε, that it will be chosen depending on the accuracy that is wished in the results. So, updated rules of the weights will be

$$\nabla W_{ilm} = -\alpha \frac{\partial E_i}{\partial W_{ilm}} = \alpha(T_i - S_i^f).\frac{\partial \mathcal{S}_i^f}{\partial W_{ilm}} \tag{4}$$

$$\frac{\partial \mathcal{S}_i^f}{\partial W_{imn}} = \frac{\partial \max_{j,k} \min(W_{ijk}, \mathcal{S}_j^{f-1}, I_k^{f-1})}{\partial \min(W_{imn}, \mathcal{S}_m^{f-1}, I_n^{f-1})} . \frac{\partial \min(W_{imn}, \mathcal{S}_m^{f-1}, I_n^{f-1})}{\partial W_{imn}} \tag{5}$$

Let be $P1$:

$$\frac{\partial \max \min_{j,k}(W_{ijk}, \mathcal{S}_j^{f-1}, I_k^{f-1})}{\partial \min(W_{imn}, \mathcal{S}_m^{f-1}, I_n^{f-1})} = \tag{6}$$

$$\frac{\partial \max_{j \neq mk \neq n}[\min(W_{imn}, \mathcal{S}_m^{f-1}, I_n^{f-1}), \min(W_{ijk}, S_j^{f-1}, I_k^{f-1})]}{\min(W_{imn}, \mathcal{S}_m^{f-1}, I_n^{f-1})}$$

and P2:

$$P2 = \frac{\partial \min(W_{imn}, \mathcal{S}_m^{f-1}, I_n^{f-1})}{\partial W_{imn}} \tag{7}$$

The derivative of functions $min(y, p)$ and $max(y, p)$, exists for $y < p$ and $y > p$, but it is not defined in $y = p$

$$\frac{\partial \min(y,p)}{\partial y} = \left\{ \begin{array}{l} 1 \text{ if } y < p \\ 0 \text{ if } y > p \end{array} \right\} ; \frac{\partial \max(y,p)}{\partial y} = \left\{ \begin{array}{l} 1 \text{ if } y > p \\ 0 \text{ if } y < p \end{array} \right\} \tag{8}$$

As it is usual [7], we will assign in any case for $y = p$ the value to the right or the left of the correspondent derivative

$$\frac{\partial \min(y,p)}{\partial y} = \left\{ \begin{array}{l} 1 \text{ if } y \leq p \\ 0 \text{ if } y > p \end{array} \right\} ; \frac{\partial \max(y,p)}{\partial y} = \left\{ \begin{array}{l} 1 \text{ if } y \geq p \\ 0 \text{ if } y < p \end{array} \right\} \tag{9}$$

However, we have changed this "crisp" behaviour by a "fuzzy" one which is capable to capture the real mening of $(y \leq p)$ or $(y \geq p)$ in an imprecise contex. We will use as a measure of the degree of inclusion of p in y the Gödel's implication. So, it remains :

$$\frac{\partial \min(y,p)}{\partial y} = \left\{ \begin{array}{l} 1 \text{ if } y \leq p \\ y \text{ if } y > p \end{array} \right\} ; \frac{\partial \max(y,p)}{\partial y} = \left\{ \begin{array}{l} 1 \text{ if } y \geq p \\ p \text{ if } y < p \end{array} \right\} \tag{10}$$

By taking into account previous equations, we obtain the following results:

$$P1 = \left\{ \begin{array}{ll} 1 & \text{si } \min(W_{imn}, \mathcal{S}_m^{f-1}, I_n^{f-1}) \geq MAX \\ \min(W_{imn}, \mathcal{S}_m^{f-1}, I_n^{f-1}) & \text{si } \min(W_{imn}, \mathcal{S}_m^{f-1}, I_n^{f-1}) < MAX \end{array} \right\} \tag{11}$$

$$P2 = \left\{ \begin{array}{c} 1 \text{ si } W_{imn} \leq \{S_m^{f-1}, I_n^{f-1}\} \\ S_m^{f-1} \text{ si } S_m^{f-1} \leq \{W_{imn}, I_n^{f-1}\} \\ I_n^{f-1} \text{ si } I_n^{f-1} \leq \{W_{imn}, S_m^{f-1}\} \end{array} \right\} \tag{12}$$

By combining the values of $P1$ and $P2$:

$$\frac{\partial S_i^f}{\partial W_{imn}} = \left\{ \begin{array}{l} \text{if } MIN = W_{imn} \left\{ \begin{array}{l} W_{imn} \geq MAX \rightarrow R1 = 1 \\ W_{imn} < MAX \rightarrow R2 = W_{imn} \end{array} \right. \\ \text{if } MIN = S_m^{f-1} \left\{ \begin{array}{l} S_m^{f-1} \geq MAX \rightarrow R3 = S_m^{f-1} \\ S_m^{f-1} < MAX \rightarrow R4 == S_m^{f-1} * S_m^{f-1} \end{array} \right. \\ \text{if } MIN = I_n^{f-1} \left\{ \begin{array}{l} I_n^{f-1} \geq MAX \rightarrow R5 = I_n^{f-1} \\ I_n^{f-1} < MAX \rightarrow R6 = I_n^{f-1} * I_n^{f-1} \end{array} \right. \end{array} \right.$$

where

$$MAX = \max_{j \neq m, k \neq n} [\min(W_{ijk}, S_j^{f-1}, I_k^{f-1})]$$

$$MIN = \min(W_{imn}, \mathcal{S}_m^{f-1}, I_n^{f-1})$$

Once this increment is calculated, we use a momentum term, taking into account in the calculation of current increment, the inmediately previous increment:

$$\nabla W_{ijk} = \mu.\nabla W_{ijk}^{old} + (1-\mu).\nabla W_{ijk} \tag{13}$$

Usually, we have taken $\mu = 0.5$.

2.2 Showing of Trainning Examples

Initially, weights are ramdomly chosen in interval [0,1]. The trainning data, as previously referred, consist of pairs (α, β), where α is an input sequence $x_1x_2...x_l$, $x_i \in \{0,1\}$ y $\beta = (\mu_1(x), \mu_2(x), ..., \mu_N(x))$, being N the number of automaton states and the number of fuzzy neurons of the neural network. Starting value for fuzzy neurons $\mathcal{S}_i^0$ is zero except for neuron $\mathcal{S}_0^0$ that it will be equal to 1. Network starts trainning with only 20 examples: The remaining trainning examples is separated in other set that network will study once it has learned the first 20 examples or done a specific number of epoques, understanding by epoque the period that network spends in processing only one time all of sequences of trainning. When the network has learned its set of trainning pattern or exceeded the maximum number of epoques , then it studies the remaining set of examples that it has not seen yet, acting in the following way: if one sequence of this set is not recognized ($E > \epsilon$), then it is included in the trainning set and, in other case, processes another one. The number of sequences to introduce is limited to 10 in order to avoid the overspecialization of the network. It is termed cycle to the set of actions of processing the sequences of trainning set and, subsequently, studying the set of sequences that it is not seen and including if it is not recognized. Usually, we use 20 cycles as maximum to train the network, whenever a sequence of trainning set was processed and its error was greater than ϵ we will pass to fit the weights by using previously given equations.

2.3 Extraction of Fuzzy Automaton

Once network has learned the trainning examples, the learned fuzzy automaton can be extracted by studying the weights of conections of the fuzzy recurrent neurons, that is, the weight W_{ijk} represents the degree of membership of the transition of state j to state i of automaton, having read the input symbol k. Therefore, we can build the fuzzy relation $\mathcal{T}[0]$ y $\mathcal{T}[1]$ as a function of input k, $k \in \{0,1\}$:

$$\mathcal{T}[0] = \begin{pmatrix} W_{000} & W_{100} & . & W_{N00} \\ W_{010} & W_{110} & . & W_{N10} \\ . & . & . & . \\ W_{0N0} & W_{1N0} & . & W_{NN0} \end{pmatrix} \quad \mathcal{T}[1] = \begin{pmatrix} W_{001} & W_{101} & . & W_{N01} \\ W_{011} & W_{111} & . & W_{N11} \\ . & . & . & . \\ W_{0N1} & W_{1N1} & . & W_{NN1} \end{pmatrix}$$

3 Simulation

We have chosen as example the automaton that works as expressed by the following fuzzy relations:

$$\mathcal{T}[0] = \begin{pmatrix} 0.8 & 0.2 & 0.1 & 0.2 & 0.1 \\ 0.3 & 0.1 & 0.9 & 0.2 & 0.1 \\ 0 & 0.1 & 0.2 & 0.85 & 0.1 \\ 0.1 & 0 & 0.9 & 0.2 & 0.1 \\ 0.1 & 0.1 & 0.2 & 0.1 & 0.7 \end{pmatrix} \mathcal{T}[1] = \begin{pmatrix} 0.1 & 0.9 & 0.2 & 0.1 & 0.2 \\ 0.8 & 0 & 0.1 & 0.2 & 0 \\ 0.2 & 0.1 & 0 & 0.1 & 0.9 \\ 0.1 & 0.8 & 0.1 & 0 & 0.1 \\ 0.1 & 0.1 & 0 & 0.2 & 0.8 \end{pmatrix}$$

As it can be observed in matrixs, the automaton has 5 states, being final states the ones 0, 1 and 3. Initial state will be 0 and it will be represented by $I_0 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \end{pmatrix}$.

The network used to learn this automaton has been a neural network with 5 fuzzy recurrent neurons and two input neurons. The parameters used to learn have been a learning ratio, $\alpha = 0.2$, an answering tolerance $\epsilon = 0.0005$ and momentum term, $\mu = 0.5$. In this simulation were needed 3 cycles of 500 epoques to obtain a learning results of 100% of examples. The obtained results once network was trainned were

$W_{000} = 0.8000$	$W_{001} = 0.1000$	$W_{010} = 0.3000$	$W_{011} = 0.7999$
$W_{020} = 0.0019$	$W_{021} = 0.2000$	$W_{030} = 0.2002$	$W_{031} = 0.0126$
$W_{040} = 0.0007$	$W_{041} = 0.0323$	$W_{100} = 0.2000$	$W_{101} = 0.9000$
$W_{110} = 0.0887$	$W_{111} = 0.0198$	$W_{120} = 0.0609$	$W_{121} = 0.0999$
$W_{130} = 0.1080$	$W_{131} = 0.8695$	$W_{140} = 0.0886$	$W_{141} = 0.0999$
$W_{200} = 0.1000$	$W_{201} = 0.2000$	$W_{210} = 0.9014$	$W_{211} = 0.0000$
$W_{220} = 0.2000$	$W_{221} = 0.0000$	$W_{230} = 0.2000$	$W_{231} = 0.0001$
$W_{240} = 0.1464$	$W_{241} = 0.0000$	$W_{300} = 0.1210$	$W_{301} = 0.1260$
$W_{310} = 0.1197$	$W_{311} = 0.1696$	$W_{320} = 0.1214$	$W_{321} = 0.2026$
$W_{330} = 0.2516$	$W_{331} = 0.2374$	$W_{340} = 0.1259$	$W_{341} = 0.1998$
$W_{400} = 0.0000$	$W_{401} = 0.1999$	$W_{410} = 0.0005$	$W_{411} = 0.0000$
$W_{420} = 0.0000$	$W_{421} = 0.8744$	$W_{430} = 0.2512$	$W_{431} = 0.0000$
$W_{440} = 0.6782$	$W_{441} = 0.8281$		

So, for instance, the meaning of weight W_{231} is that the automaton has a transition with degree of membership 0.0001 from state 3 to state 2, reading the input symbol 1. Therefore, we can extract the fuzzy automaton associated to the trainning examples by building the fuzzy matrixs with the semantic of previous examples. In this way, the following fuzzy relations are obtained:

$$\mathcal{T}[0] = \begin{pmatrix} 0.8000 & 0.2000 & 0.1000 & 0.1210 & 0.0000 \\ 0.3000 & 0.0887 & 0.9014 & 0.1197 & 0.0005 \\ 0.0019 & 0.0609 & 0.2000 & 0.1214 & 0.0000 \\ 0.2002 & 0.1080 & 0.2000 & 0.2516 & 0.2512 \\ 0.0007 & 0.0886 & 0.1464 & 0.1259 & 0.6782 \end{pmatrix}$$

$$\mathcal{T}[1] = \begin{pmatrix} 0.1000 & 0.9000 & 0.2000 & 0.1260 & 0.1999 \\ 0.7999 & 0.0198 & 0.0000 & 0.1696 & 0.0000 \\ 0.2000 & 0.0999 & 0.0000 & 0.2026 & 0.8744 \\ 0.0126 & 0.8695 & 0.0001 & 0.2374 & 0.0000 \\ 0.0323 & 0.0999 & 0.0000 & 0.1998 & 0.8281 \end{pmatrix}$$

4 Conclusions

We have proposed a fuzzy recurrent neural network which is capable of inferring the associated automaton from sequences of examples of such fuzzy language. We simulate the behaviour of the network in the learning phase for a specific example, showing that the network learns to infer the automaton associated to the training examples, computing in their weights the relations which characterizing the behaviour of the automaton to learn.

References

[1] Blanco A., Delgado M., Requena I., Identification of fuzzy relational equations by fuzzy neural networks, Fuzzy Sets and Systems, Vol.71, p.215-226, (1995).

[2] Cleeremans, A., Servan-Schreiber, D., and McClelland, J. 1989. Finite state automata and simple recurrent networks. Neural Computation. 1(3), 372-381.

[3] Fu, K. S., Syntactic Pattern Recognition and Applications, Prentice Hall, Englewood Cliffs, N.J. (1982).

[4] Giles, C.L., and Omlim, C.W. 1992. Inserting rules into recurrent neural network Proc 1992 IEEE Workshop Neural Networks Signal Process, pp. 13-22. Copenhagen, Denmark.

[5] Giles, C.L., Miller, C.B., Chen, D.,Chen, H.H., Sun, G.Z., and Lee, Y.C. 1992. Learning and extracting finite state automata with second-order recurrent neural networks. Neural Computation 4, 393-405.

[6] Harrison, M.H., Introduction to Formal Language Theory, Addison-Wesley, Reading, Mass., (1978).

[7] Ikoma N., Pedrycz W. and Hirota K.1993. Estimation of fuzzy relational matrix by using probabilistic descent method. Fuzzy Set Systems 57, 335-349.

[8] Watrous, R.L., and Kuhn, G.M. 1992. Induction of finite-state languages using second-order recurrent networks. Neuronal Computation 4, 406-414.

[9] Zeng, Z., Goodman, R., and Smyth, P. 1993. Learning finite state machines with self-clustering recurrent networks. Neural Computation 5(6), 976-990.

[10] Zeng, Z., Goodman, R., and Smyth, P., 1994. Discrete recurrent neural networks for gramatical inference. IEEE Transact. Neural Networks 5(2), 320-330.