Prior Knowledge in Evolutionary Fuzzy Recurrent Controllers Design

Javier Apolloni, Carlos Kavka and Patricia Roggero

LIDIC
Departamento de Informática
Universidad Nacional de San Luis
Ejército de los Andes 950
D5700HHW - San Luis - Argentina
Tel: 02652-420823 Fax: 02652-430224
e-mail: {javierma,ckavka,proggero}@unsl.edu.ar

Abstract

A fuzzy controller is usually designed by formulating the knowledge of a human expert into a set of linguistic variables and fuzzy rules. As it is well known, the use of prior knowledge can dramatically improve the performance and quality of the fuzzy system design process. In previous works we have introduced the RFV model, a representation for recurrent fuzzy controllers based on Voronoi diagrams that can represent fuzzy systems with synergistic rules, fulfilling the completeness property and providing a simple way to introduce prior knowledge. In this work we present our current approach in the study of the inclusion of prior knowledge in the context of the RFV model.

1 Introduction

The development of controllers by using fuzzy logic techniques has been subject of fundamental research with many successful applications produced during last years [1]. The main reason is that fuzzy logic controllers (FLC) provide satisfactory performance in face of uncertainty and imprecision [5], while keeping an equivalence in knowledge representation with other methods like neural networks and automata [4]. An FLC represents a non linear model as the combination of a set of local linear models, where each one represents the dynamics of a complex system in a single local region [3]. Each local model is specified by a fuzzy rule, which defines the local region in which the rule applies through the membership functions used in the antecedent, while the consequent defines the output of the model.

One of the advantages of fuzzy systems is the possibility to use prior knowledge, a well known concept that specifies the information about the desired form of a solution which is additional to the information provided by the training data [2]. The correct use of prior knowledge during model design leads to better models even in the presence of deficient and incomplete data sets [10].

Most non linear problems in control require the processing of temporal sequences, or in other words, in these problems the output depends on the current input and previous values of inputs and/or outputs. Fuzzy recurrent models can deal with this kind of problems. In previous works, we have introduced the Recurrent Fuzzy Voronoi (RFV) model, a recurrent model that supports the learning

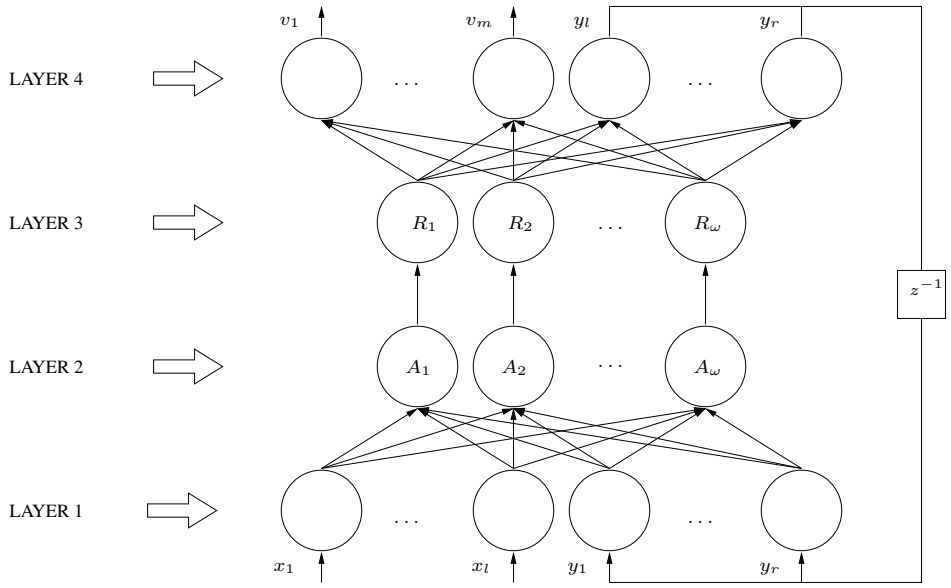


Figure 1: The structure of the RFV model

of temporal sequences providing a clear semantic interpretation of recurrent units and the possibility of prior knowledge insertion.

This paper is organized as follows. Section 2 describes the structure of the RFV model. In section 3, the design of fuzzy controllers by using genetic algorithms is introduced and section 4 presents experimental results with the RFV model. Section 5 details current experiments and future works in the area.

2 The RFV model

A schematic diagram of the model is shown in figure 1, which is organized in four layers and consists of l input variables, r internal variables, m output variables and ω rules. Units in layer 1 are called input units. There are two types of input units: external inputs and internal units that are used also as standard inputs in rule definition. Units in layer 2 are called partition units. They act as multidimensional fuzzy membership functions defined in terms of Voronoi diagrams. Units in layer 3 are called rule units. Each fuzzy rule in the fuzzy system has a corresponding rule unit. There is a one to one correspondence with units in layer 2. Units in layer 4 are called output units. They compute the outputs as a weighted linear combination of input units, generating both the external outputs and the values of the internal units to be made available as inputs in the next time step.

The rules defined in the RFV controller are standard TS (Takagi Sugeno) type fuzzy rules [1], with their own inputs and outputs. The complete system is recurrent because some outputs are connected to inputs, but each rule by itself is a standard TS type fuzzy rule. This fact contributes to provide a clear interpretation of the rules and make easy to define the prior rules for the RFV controller. This approach contrasts with other models like RFNN, RSONFIN, DFNN or the TRFN [7], where the rules themselves include backward connections.

3 RFV design with evolutionary algorithms

Evolutionary algorithms are selected as the optimization tool for RFV controller design, since they have been very successful on problems where training data or gradient information is very difficult or

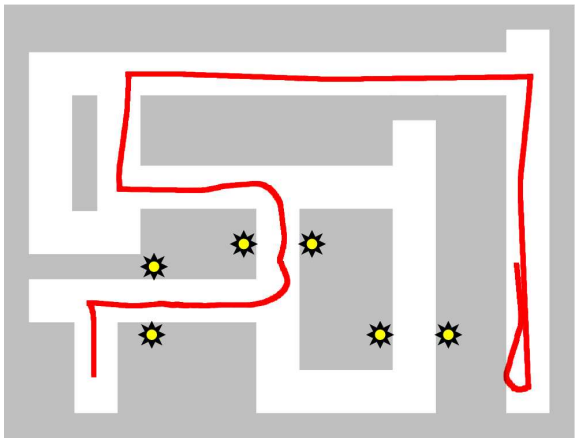


Figure 2: The performance of the best controller when evaluated in a scenario not used during evolution.

costly to obtain, like most control problems. A floating point coding scheme is selected, where each individual (or chromosome) represents all free parameters of the RFV controller as a variable length vector of floating point values. An individual I with ω rules is defined as the vector:

$$Ind = R_1 : \ldots : R_\omega \tag{1}$$

where each sub-vector $R_i (1 \leq i \leq \omega)$ is the floating point vector defined by the real valued parameters associated to the rule R_i. The evolutionary algorithm is described in details in [8]. The crossover operator is based on geometrical exchange of Voronoi sites between the parents with respect to a random hyperplane. The mutation operator can either modify the parameters of a particular rule by some standard Gaussian mutation, or add or delete a Voronoi site, i.e. a rule.

4 Evolutionary robotics

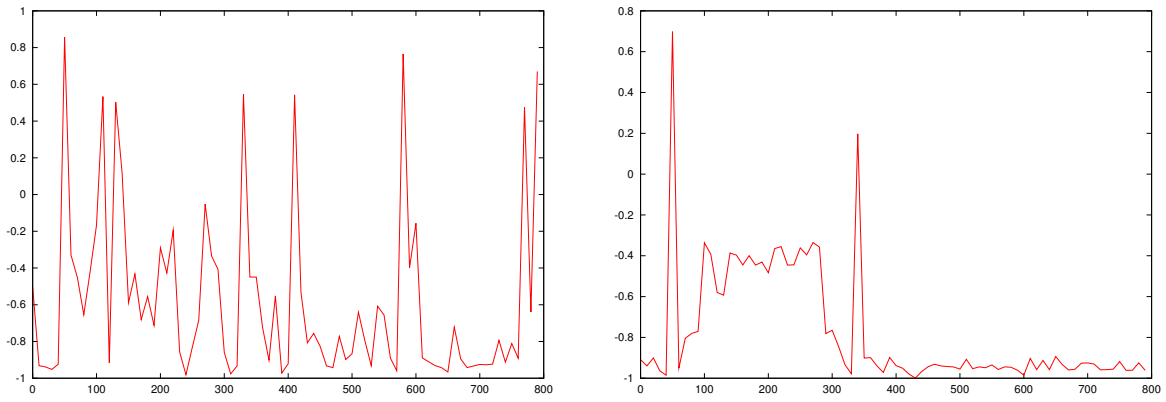
As a test base for experiments, a simulated Khepera robot was used for experimentation. A Khepera robot has 8 sensors that can be used to measure proximity of objects and ambient light levels, and two independent motors to control the speed and direction of the robot. The problem consists in driving the robot while avoiding collisions, starting from a fixed initial position, to a target position that depends on light based signals that are set to on or off status in the trajectory. The presence of an illuminated signal (on status) indicates to the robot that it has to turn left in the next intersection, and its absence (or off status) that it has to turn right. The controller needs internal memory, since the signal is not present in the intersection, but in a previous (and maybe distant) point in the trajectory. The controller has to learn also when to *forget* light signals that affected the behavior in previous intersections and have not to be considered in other point of the trajectory. The fitness of a RFV controller is computed as mentioned in [6].

The controllers are defined with five inputs, two outputs and one internal variable. The inputs are, respectively, the average of the two left sensors, the two front sensors, the two right sensors, the two back sensors and an average of ambient light as measured by all sensors. The outputs correspond to the speed of the two motors. Note that the presence of an internal variable forces the rules to be defined with six inputs and three outputs (see figure 1).

The experiments were performed with and without prior knowledge. The rules defined beforehand and inserted (as explained in [9]) are shown in table 1. The semantic of the rules is defined by considering the internal variable y_1 as a flag that indicates if a light signal was seen before. The first

Rule	*site*	v_1	v_2	y_1
	L,C,R,B,G,y_1	$a_0^1,a_1^1,a_2^1,a_3^1,a_4^1,a_5^1,a_6^1$	$a_0^2,a_1^2,a_2^2,a_3^2,a_4^2,a_5^2,a_6^2$	$a_0^3,a_1^3,a_2^3,a_3^3,a_4^3,a_5^3,a_6^3$
R_1	0,0,0,0,0,0	1,0,0,0,0,0,0	1,0,0,0,0,0,0	0,0,0,0,0,0,0
R_2	0,0,0,0,0,1	1,0,0,0,0,0,0	1,0,0,0,0,0,0	1,0,0,0,0,0,0
R_3	0,0,0,0,1,0	1,0,0,0,0,0,0	1,0,0,0,0,0,0	1,0,0,0,0,0,0
R_4	0,0,0,0,1,1	1,0,0,0,0,0,0	1,0,0,0,0,0,0	1,0,0,0,0,0,0
R_5	0,1,0,0,0,0	1,0,0,0,0,0,0	0,0,0,0,0,0,0	0,0,0,0,0,0,0
R_6	0,1,0,0,0,1	0,0,0,0,0,0,0	1,0,0,0,0,0,0	0,0,0,0,0,0,0

Table 1: A Priori rules. The value of the site corresponds to the center of the Voronoi region defined by the rule. It is specified by the normalized values of the left (L), center (C), right (R), back (B) and light (G) sensors, and the internal variable y_1. The values a_j^i correspond to the parameters used to define the approximators.



(a) (b)

Figure 3: Value of the internal unit of the best controller obtained through evolution (a) without and (b) with prior knowledge when evaluated on the test scenario of figure 2.

four rules correspond to the situation where there are no obstacles near the robot (all distance sensor values are equal to 0). The output produced in all cases for the motors is maximum forward speed (note the constant term of the approximator is 1 for both motor outputs v_1 and v_2). The value of the internal variable y_1 is set to 1 when light is present (rules 3 and 4) and to the previous value (can be 0 or 1) if no light is measured (rules 1 and 2). Rule 5 produces a turn to the right (left motor at maximum speed) if no light was detected before ($y_1 = 0$) and rule 6 a turn to the left (right motor at maximum speed) if light was detected ($y_1 = 1$). In both cases, the flag (internal variable y_1) is reset to 0. It is important to note that the a priori knowledge is defined by specifying rules that determine the expected behavior of the controller in specific points in the input domain, without specifying the area of application of the rules.

The most important point is that a definite semantic interpretation of the internal unit is provided with the a priori rules: the internal unit behavior indicates if light was or not detected before the intersection. There is no guarantee that a clear semantic is provided with the approach without prior knowledge. Figure 3 shows the value of the internal unit of the best controllers plotted when evaluated on the test scenario from figure 2. The value of the internal unit for the controller evolved with prior knowledge represents the expected semantics, with two peaks on the areas where light signals were detected. This behavior was observed in most controllers obtained after evolution.

5 Current and Future Developments

The previous experiment showed that it is possible to define internal units with a clear semantic. One of the advantages of this approach is that an RFV system can be used as a component of a combined fuzzy system. A hierarchically combined fuzzy system can be defined in terms of a set of independent fuzzy systems, by using outputs from some of them as inputs of others. Considering RFV systems, the value of the internal units with clear semantic can also be used as inputs of other fuzzy systems together with the external inputs defined by human experts. Usually, a hierarchically combined fuzzy system is developed more easily and the number of rules is smaller.

We are currently performing an analysis of the convergence of the algorithm by using more internal units. Experiments are being carried by considering a larger number of internal units with a semantic defined by the problem (as in section 4) and also free internal units with no prior semantics. The objective of free internal units is to provide more freedom degrees to the algorithm in order to enhance its search capabilities. We are currently performing the experiments in evolutionary robotics, but experiments in other control areas are also foreseen.

References

[1] R. Babuška. Fuzzy modeling: Principles, methods and applications. In C. Bonivento, C. Fantuzzi, and R. Rovatti, editors, *Fuzzy Logic Control: Advances in Methodology*, pages 187–220. World Scientific, 1998.

[2] Christopher Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

[3] Gang Feng. An approach to adaptive control of fuzzy dynamic systems. *IEEE Transactions on Fuzzy Systems*, 10(2):268–275, April 2002.

[4] Lee Giles, Christian Omlin, and Karvel Thornber. Equivalence in knwoledge representation: Automata, recurrent neural networks and dymical fuzzy systems. *Proc. of the IEEE*, 87(9):1623–1640, September 1999.

[5] Hani A. Hagras. A hierarchical type-2 fuzzy logic control architecture for autonomous mobile robots. *IEEE Transactions on Fuzzy Systems*, 12(4):524–539, 2004.

[6] Carlos Kavka, Patricia Roggero, and Javier Apolloni. Evolución de controladores difusos recurrentes basados en diagramas de voronoi. In *Proceedings of the X CACIC*, 2004.

[7] Carlos Kavka, Patricia Roggero, and Marc Schoenauer. Evolution of voronoi based fuzzy recurrent controllers. *Accepted for publication in GECCO*, 2005.

[8] Carlos Kavka and Marc Schoenauer. Voronoi diagrams based function identification. *GECCO, Lecture Notes in Computer Science*, 2723:1089–1100, 2003.

[9] Carlos Kavka and Marc Schoenauer. Evolution of voronoi based fuzzy controllers. *PPSN, Lecture Notes in Computer Science*, 3242:541–550, 2004.

[10] Biing-Tsair Tien and G. Van Straten. The incorporation of qualitative information into t-s fuzzy model. In *Proceedings of the Annual Meeting of the North American Fuzzy Information Processing Society*, pages 148–153. NAFIP, 1997.