Fuzzy Systems and Genetic Algorithms

Andrea Bonarini

Artificial Intelligence and Robotics Lab
Department of Electronics and Information
Politecnico di Milano

E-mail: bonarini@elet.polimi.it
URL:http://www.dei.polimi.it/people/bonarini

Putting together genetic algorithms and fuzzy systems

Some possibilities:

- fuzzy representation of the fitness function (good, as well, for the reinforcement function in RL)
- genetic algorithms to learn fuzzy systems
  - Learning fuzzy sets
  - Learning fuzzy rules -> Learning Fuzzy Classifier Systems

Fuzzy fitness function

Why?

As usual, because a fuzzy representation may be easier to understand and define than a formula, more robust w.r.t. noise, etc.

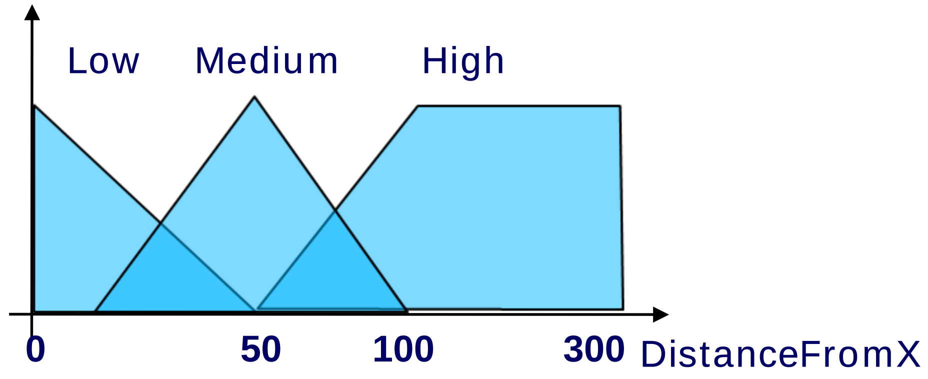
How?

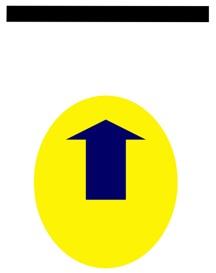
The fitness function is a relationship between characteristics of (the performance of) a solution and a number.

E.g.: for a GA that has to learn the obstacle avoidance behavior for a robot, the fitness function might be

F= (DistanceFromLeft is High) and (DistanceFromRight is High) and (DisanceFromFront is High)

Let’s see if it’s better than a numerical one (in this case)

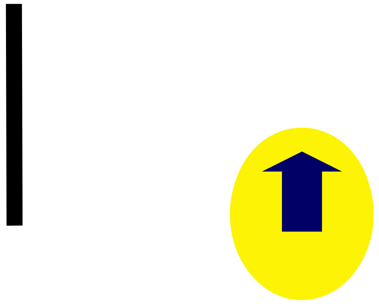




$F_f=0$

$F_n=10*300*300$
$=90000$

$F_f=$ (DistanceFromLeft is High) and
(DistanceFromRight is High) and
(DistanceFromFront is High)



$F_f=0.7$

$F_n=$ DistanceFromLeft *
DistanceFromRight *
DistanceFromFront

$F_n=80*300*300$
$=720000$

GA learning of fuzzy sets

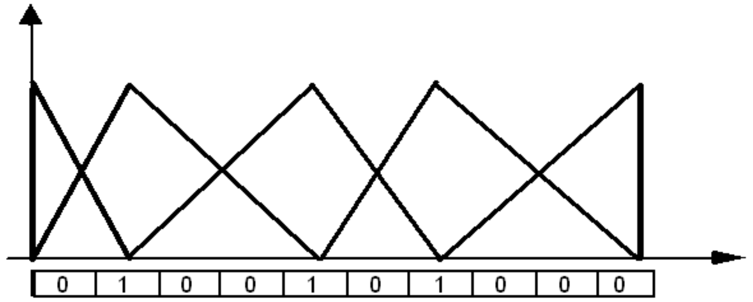
Why?

- Shape and position of fuzzy sets might be optimized
- A fitness only might be available
- The solution might have many local optima

How?

- Model for the set of fuzzy sets compatible with GA needs (e.g. string of bits),

  e.g. one bit per interval, whose value is one if the vertex of a triangular fuzzy shape is in the center of the interval, 0 if not



GA learning of fuzzy rules

Why?

- The structure of a fuzzy rule set has to be otimized
- The only available information is a fitness function
- The solution might have local optima

How?

- Models for the rules compatible with he GA needs
- Eventual models also for the fuzzy sets

What is a Fuzzy LCS?

A fuzzy LCS is a Learning Classifier System where the rules are fuzzy rules, i.e., antecedents and consequents are represented by fuzzy sets.

The main issue is that by learning a small number of parameters, we have a complete, general mapping from real numbers to real numbers.

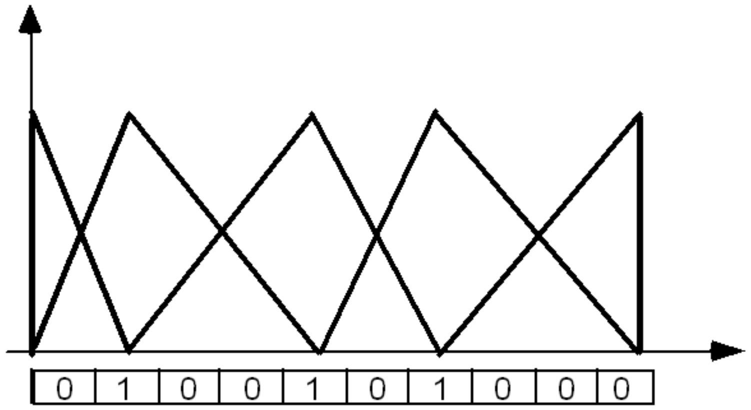
There are many approaches aimed at learning MFs and rules, or only the rule rule structure, with given MFs

Learning both rules and MFs

Nomura (1992)

Populations of bit strings corresponding to the characteristic points of the MFs

This represents number and shape of the MFs. The rulebase is complete.



Learning only the rule structure

[Valenzuela-Rendon, 1991][Thrift, 1991][Bonarini, 1991][Bonarini, 2001]

- Both with Michigan and Pitts representation (and others)
- Rules are strings of numbers corresponding to MFs

  E.g.: 132#:12

ELF

Evolutionary Learning of Fuzzy rules [Bonarini, Bonacina, Matteucci, 2001]

- Michigan-style representation
- Cover detector to generate rules
- Niche evolution: fuzzy rules with the same antecedent compete to propose the best actions, rules of different niches collaborate in the fuzzy frame
- Dynamical adaptation to niche performance: as the niche is performing well, the selection pressure is rised in the niche, so the number of rules accepted in the niche are reduced

Fuzzyfication of reinforcement distribution

A general way to apply reinforcement distribution algorithms to fuzzy rules models: consider that the innovative part has to be weighted by the contribution a rule has given (its triggering value).

For instance, considering that a pair (s,a) corresponds to a rule, here is a sort of fuzzy Q-learning:

$$Q(rule_k) \leftarrow (1-\alpha)Q(rule_k) + \mu_k\alpha\ [r + \gamma\ max_{rule_j \in R(s)}\ Q(rule_j)]$$

where R(s) is the set of rules applicable from the reached state s, and μ_k is the weight of activation of *rule*$_k$ that lead to the reinforcement r, and state s.

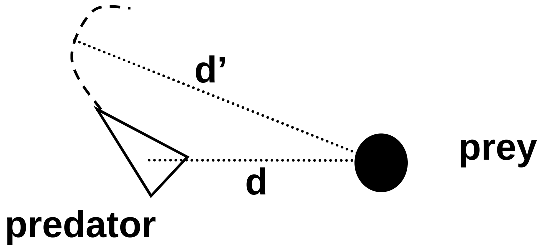
Analogously, all the other reinforcement distribution algorithms can be fuzzyfied.

Problems with biased reinforcement functions

In some cases, the reinforcement function is biased, so that some niches are penalized w.r.t. others.

When this happens, general rules, triggering in different niches, may tend to worsen the estimate of the niche.

For instance, if we have a prey-predator system, and the predator has a limitation in the steering angle, and the movement in direction of the prey is reinforced...



$$r = (d-d')$$

...when the prey is on the back, even the best action the predator can select is punished.

A possible solution for biased reinforcement function

Instead of distributing the “innovation” contribution as it is, normalize it by the estimate of the potentiality of the niche, e.g., the value of its best rule.

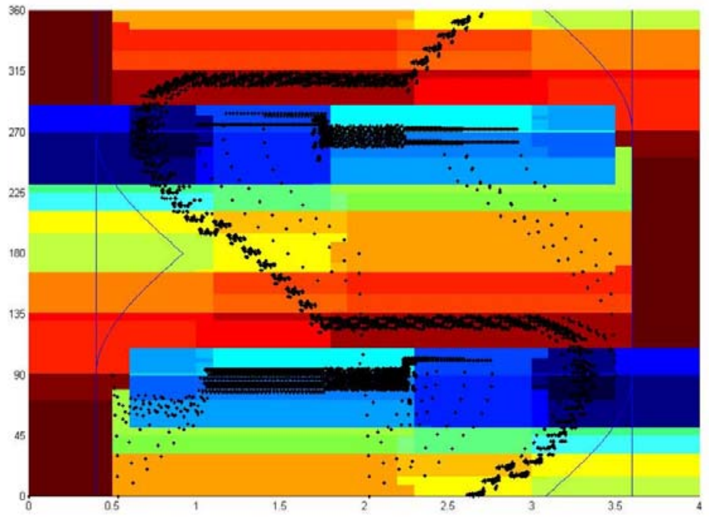
In this way, the value of a “good” general rule may be updated using a value coherent with that of the rules belonging to the same niche. E.g., if in a niche, the best reinforcement obtained is -0.3, the values are updated by considering this value as 1: worst values will be scaled accordingly, eventual better values, will upgrade the estimate. Thus, the values of rules in the niche will exploit their full range.

Reinforcement aliasing

Another problem concerns reinforcement aliasing.

In a given state, described by a set of fuzzy values, a rule may receive different reinforcement, according to the reinforcement function definition. This generates incoherence and reduces the learning abilities.

A possible solution is to partition the state in niches coherent with reinforcement

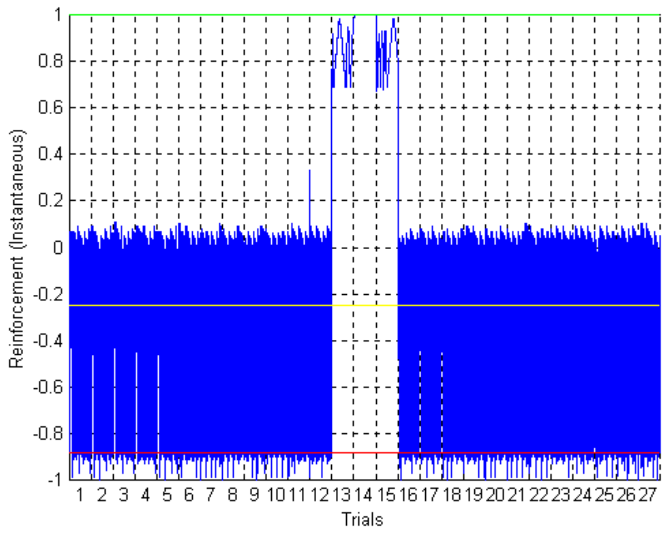


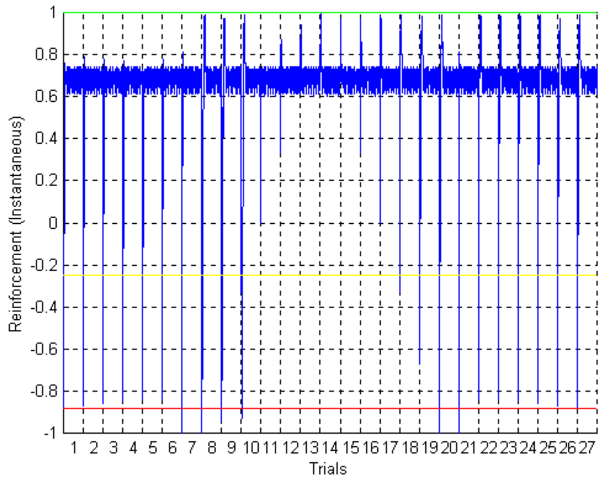
Some results

Reinforcement obtained in different trials, starting from different positions, with different partitions of the state space

Incoherence in reinforcement

in almost all the trials





ɜrent reinforcement

Results obtained by ELF