Learning weighted linguistic fuzzy rules with estimation of distribution algorithms

Luis delaOssa, José A. Gámez, José M. Puerta

***Abstract*— The main feature of Estimation of Distribution Algorithms is the way they evolve by gathering the information about the best elements of each population into a probability distribution. This work studies the application of these algorithms to the learning of weighted linguistic fuzzy-rule-based systems with the wCOR method. For this purpose, we propose the use of two different probabilistic models: One which does not assume any dependence between the rule consequents and their weights, and other whose structure is fixed from these dependences.**

I. Introduction

A *Genetic Fuzzy System* (GFS) [1] or *Evolutionary Fuzzy System* (EFS) in a broader sense, is basically a fuzzy system which is induced from data in a wrapper way by using a genetic/evolutionary algorithm to guide the search during the learning phase. There are different types of fuzzy systems, the *fuzzy-rule-based systems* (FRSBs) being those that have received the greatest amount of attention from the EFSs research community. Thus, evolutionary algorithms have been used to learn or tune different components of the FRBS.

In this work, we focus on a concrete type of FRBS: the one that uses *linguistic* or *descriptive* fuzzy rules [2]. Linguistic FRBSs (LFRBSs) are especially attractive because they allow us to achieve the double goal of being useful for prediction and fully interpretable by human experts. However, the predictive capability of a LFRBS can, in general, be improved by using weights. That is, a real number $w \in [0, 1]$ which is associated to each rule and can be understood as its degree of importance. This is a simple way of increasing the precision of the system without significantly decreasing its readability.

Our work is based on the WCOR (Weighted Cooperative Rules) methodology [3], [4], which focuses on learning the rules and their weights, but does not carry out any tuning in the number or shape of the fuzzy sets associated to the linguistic labels. Concretely, we have analysed two proposals that use *Estimation of Distribution Algorithms* (EDAs) [5] as the search engine. Both of them can be viewed as the counterpart of GA described in [4], since they simultaneously evolve the rules and their weights. In the first case, the algorithms use a probabilistic model that manages each one of the these two parts independently, whereas in the second, the structure of the model explicitly elicits the relationship between rule consequents and their weights.

The paper starts with a brief description of LFRBSs (Section II) and how to learn them by using COR and WCOR methodologies (Section III). Then in section IV we describe the canonical EDA and the algorithms used in this study. Section V contains the algorithms proposed to deal with the problem of learning weighted LFRs and in Section VI we evaluate them over a series of datasets taken from a specific repository. Finally, in Section VII we present our conclusions and directions for future research.

The authors are with the Intelligent Systems and Data Mining (SIMD) research group, Department of Computer Systems, University of Castilla-La Mancha, Campus Universitario s/n, 02071, Albacete (Spain). email: {ldelaossa|jgamez|jpuerta}@info-ab.uclm.es).

II. Linguistic fuzzy rules based systems

Fuzzy Rules (FRs) [6] are based on *Fuzzy Set Theory* [7] and are grounded on the use of fuzzy predicates, X is A, where X is a problem domain variable and A is a fuzzy set[1]. The typical structure of a fuzzy rule is as follows

$$\text{If } X_1 \text{ is } v_1^{j_1} \& \ldots \& X_n \text{ is } v_n^{j_n} \text{ Then } Y \text{ is } v_y^{j_l} \qquad (1)$$

where $\{X_1, \ldots, X_n, Y\}$ are problem domain variables and $\{v_1^{j_1}, \ldots, v_n^{j_n}, v_y^{j_l}\}$ are fuzzy sets defined over the domain of their corresponding variables.

As mentioned in Section I, we focus on the so-called *Linguistic* (or Mamdani) fuzzy rules [2]. The LFRBS knowledge base is composed of two clearly differentiated components:

- A *linguistic data base* which contains the definition of the linguistic variables. That is, the domain of each input/output variable is partitioned/covered by a fixed number of fuzzy sets, each one having associated a linguistic label. For example, the domain of the variable *age* can be covered by the set of linguistic labels: {baby, child, teenager, adult, ancient}. By associating a fuzzy set to each linguistic label we get a *linguistic variable* [8].
- A *rule base* defined over the linguistic variables. In the *linguistic modeling* of a system only the linguistic labels of a variable can appear in the fuzzy predicates of the rules. That is, the fuzzy set $v_1^{j_1}$ used for variable X_1 in equation 1 cannot be chosen with total freedom, but from the set of linguistic-labels used in the definition of linguistic variable X_1.
  Because of this restriction in the designing of the fuzzy rule system, LFRs usually have a lower precision than other types of fuzzy rule systems, but on the other hand

[1] In this work we only use triangular fuzzy sets. The membership degree of a point x with respect to a triangular function defined in the interval [a,c] and maximum/middle value in b is obtained as:

$$\mu_{Triangular}(x) = \begin{cases} \frac{x-a}{b-a}, & \text{if } a \leq x \leq b \\ \frac{c-x}{c-b}, & \text{if } b \leq x \leq c \\ 0, & \text{otherwise} \end{cases}$$

as linguistic terms have a semantic meaning associated, their rules (e.g. "If car-speed is high and distance-to-next-car is short then brake-force is intense") are fully interpretable by human experts.

As mentioned previously, a numeric weight $w \in [0,1]$ can be associated to each rule in order to increase the predictive accuracy of the whole system but without significantly decreasing its readability.

With respect to the inference we can distinguish the following components:

- *Fuzzification/defuzzification interface*. These two interfaces are needed because the rules deal with fuzzy sets while our real problem (data) is related to numerical values. Thus, our first step is the transformation of our input numerical values in fuzzy sets. This task is carried out by producing a singleton fuzzy set $\hat{r}$ for a given number r, i.e., a fuzzy set such that r has membership degree 1 ($\mu_{\hat{r}}(r) = 1.0$) and any point $s \neq r$ has membership 0. On the other hand, the defuzzification interface takes a fuzzy set and produces a numerical output by using (in our case) the center-of-gravity of the given fuzzy set.
- *Inference engine*. Given an input $\mathbf{x} = \langle x_1, \ldots, x_n \rangle$ any rule (see eq. 1) such that $\forall_{i=1\ldots n}\ \mu_{v_i^{j_i}}(x_i) > 0$ is fired. As the fuzzy sets defining linguistic fuzzy variables usually overlap, an input usually fires several rules. When a rule is fired a fuzzy set for the target variable (Y) is obtained. In this work, the set v'_y is obtained (by using classical operators) as:

$$\mu_{v'_y}(r) = \begin{cases} \mu_{v_y^{j_l}}(r) & \text{if } \mu_{v_y^{j_l}}(r) < m \\ m & \text{if } \mu_{v_y^{j_l}}(r) \geq m \end{cases}$$

m being the matching degree of $\mathbf{x}$ to the rule: $m = min_{i=1..n}\mu_{v_i^{j_i}}(x_i)$.

If k rules are fired by a given input $\mathbf{x}$ and $v_y^1, \ldots, v_y^k$ are the obtained fuzzy sets, then we have to combine them into a single output. In this work we use the weighted FITA (*First Integrate Then Aggregate*) approach, which first defuzzifies $v_y^1, \ldots, v_y^k$ into their corresponding numerical values $r_y^1, \ldots, r_y^k$ and then aggregates them into a single value by using a weighted average and also taking into account the weight (w_i) associated to each rule:

$$\hat{y} = \frac{\sum_{i=1}^{k} r_y^i \cdot m_i \cdot w_i}{\sum_{i=1}^{k} m_i \cdot w_i},$$

m_i being the matching degree of $\mathbf{x}$ with respect to the i-th rule fired.

III. Learning weighted linguistic fuzzy rules with WCOR

Although there are different approaches to the problem of learning LFRs, we base our work on the so-called grid-based methods. These methods assume that the linguistic variables have been defined previously and focus on the rule generation process.

In this paper, we consider the easiest (and most frequently used) way of constructing the linguistic data base: (1) we use the same number of linguistic labels (l) for all the variables; and (2), a symmetrical fuzzy partition of the domain is created by using l triangular fuzzy sets (see Figure 1). Thus,

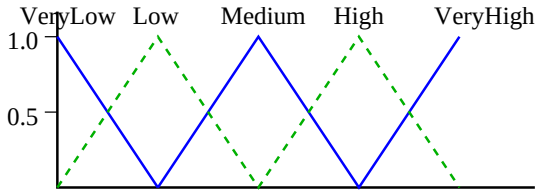


Fig. 1. Symmetrical linguistic variable with 5 labels

we have $n+1$ linguistic variables, $\{X_1, \ldots, X_n, Y\}$ in our linguistic data base, each one having l linguistic labels/terms $\{v_i^1, \ldots, v_i^l\}$. Now our goal is to learn the rule base.

Grid-based methods assume that all the variables appear in the antecedent of the rule, so they start by defining a n-dimensional grid $X_1 \times X_2 \times \cdots \times X_n$ where each cell or sub-space represent a possible antecedent. Although this gives rise to a maximum number of l^n rules, grid-based methods are guided by covering criteria of the examples in the *training set* and so, empty sub-spaces are discarded. An example $e_r = (\mathbf{x}_r, y_r) = (x_{r1}, \ldots, x_{rn}, y_r)$ belongs to a given sub-space S_i if $\forall_{i=1..n}\mu_{S_{ij}}(x_i) > 0$, S_{ij} being the linguistic label associated to variable X_j in sub-space S_i. Notice that as linguistic terms overlap (see fig. 1) the same example can belong to different sub-spaces; however most grid-based algorithms for learning LFRBSs assign an example e_r to a unique sub-space S_i (the one with the highest covering degree).

The next step is to identify the consequent for each non-empty sub-space. Given the set of examples $e_{S_i} = \{e_1, \ldots, e_m\}$ covered by sub-space S_i, Wang and Mendel (WM) algorithm [9], which is a clear representative of grid-based methods, decides the consequent for S_i as $Y = v_y^b$, such that:

$$v_y^b = \arg_k \max_{r=1,\ldots,m} \left[\left(\max_{k=1,\ldots,l} \mu_{v_y^k}(y_r) \right) \prod \mu_{S_{ij}}(x_{rj}) \right]$$

that is, WM also uses a covering criterion to (greedily) select the consequent for each sub-space.

The main advantage of the WM algorithm is that it is computationally very efficient. On the other hand, the main shortcomings of the WM algorithm result from its greedy behavior. Thus, in each sub-space it looks for the rule with the best individual performance, without considering that the interaction between all the system rules will actually define its global performance.

In [3] Casillas et al. propose the COR methodology, a grid-based method that tries to overcome the shortcomings of the WM algorithm by studying the cooperation between the different rules of the system. Thus, in the COR (*COoperative Rules*) methodology the greedy/local selection of the consequent for each sub-space is replaced by a combinatorial search in the space of all rule candidate sets. In order to state

the problem as a combinatorial one we need to define the search space and the way in wich the points or individuals of such space are evaluated. Once these components have been specified different instances of COR can be obtained by using different metaheuristics.

A. COR search space

Let $\{S_1, \ldots, S_m\}$ be the set of non-empty sub-spaces by using the covering criterion previously described. Then, our goal is to look for the consequents $\{c_1, \ldots, c_m\}$ that yield the best possible system (see below the definition of fitness function).

The first possible definition of the search space could be the cartesian product: $\{v_y^1, \ldots, v_y^l\}^m$. However COR also uses a covering criterion of the training set in order to restrict the number of possible consequents for each sub-space: given the set of examples $\{e_1, \ldots, e_r\}$ covered by a sub-space S_i, then the set of possible consequents for S_i is:

$$cons(S_i) = \left\{ v_y^k \mid k = \arg\max_{1,\ldots,l} \mu_{v_y^k}(y_r) \right\} \cup \{\aleph\}$$

where $\aleph$ is an empty consequent whose meaning is that no rule is added to the system for this sub-space.

Therefore, an individual or potential solution in COR is an array $c[]$ of integers of length m, m being the number of possible sub-spaces (rules). For a given position $1 \leq j \leq m$, $c[j]$ will be a number between 1 and l representing the index of the linguistic term chosen as consequent for antecedent S_j, or -1 ($\aleph$) that corresponds to the fact of not including any rule for sub-space S_j in the current system.

B. WCOR search space

WCOR [4] is an extension of COR methodology for the purpose of learning weighted LFRs. In WCOR, the learning process is enlarged in order to induce not only the rule but also its weight. Thus, we have to deal with a hybrid problem, viz, combinatorial plus numerical optimization. If there are m possible consequents then the search space for WCOR is:

$$(cons(S_i), [0, 1])^m$$

although for convenience it is better to place all the integers (and floats) consecutively, i.e.,

$$cons(S_1) \times cons(S_2) \times \cdots \times cons(S_m) \times [0, 1]^m.$$

Therefore, an individual of WCOR search space is an array $cw[]$ of size $2m$ where the first m positions are as in COR while the last m positions are real numbers representing the weights. Furthermore, there is a correspondence between both parts such that position $1 \leq i \leq m$ and $i + m$ are linked and represent the consequent for the rule associated with sub-space S_i and its weight (w_i).

C. Fitness function

To evaluate the goodness of a given solution $c[]$ or $cw[]$, it is decodified to its corresponding LFRBS and used to predict the output value for the examples in the training set. Then, the mean squared error (MSE) or some of its variants is used as goodness measure.

IV. Estimation of Distribution Algorithms

Estimation of Distribution Algorithms (EDAs) [5] are a family of evolutionary algorithms which have gained in importance during the last 5 years. They are based on populations as genetic algorithms (GAs) but, instead of evolving by means of genetic operators, they gather the features of the best individuals in a population into a probability distribution and use it to sample the new solutions.

EDA Approach

1) $D_0 \leftarrow$ Generate the initial population (m individuals)
2) Evaluate the population D_0
3) $k = 1$
4) Repeat
   a) $D_{tra} \leftarrow$ Select $n \leq m$ individuals from D_{k-1}
   b) Estimate/learn a new model $\mathcal{M}$ from D_{tra}
   c) $D_{aux} \leftarrow$ Sample m individuals from $\mathcal{M}$
   d) Evaluate D_{aux}
   e) $D_k \leftarrow$ Select m individuals from $D_{k-1} \cup D_{aux}$
   f) $k = k + 1$

   Until stop condition

Fig. 2. Description of EDAs operation mode

Figure 2 shows the general outline of EDA evolution process. As we can see, steps (b) and (c) replace the classical selection+crossover+mutation used in GAs. Step (b) is the key point in EDAs, because working with a joint probability distribution is not useful even in small problems, so a simpler model has to be estimated/learned. Depending on the complexity of the model considered, different models of EDAs arise. Thus, the more complex this model is the better collection of dependencies between variables it will show, but the more complex/time-consuming its estimation will be. In the literature we can find several proposals that can be grouped into: *univariate* models (no dependencies are allowed), *bivariate* models (pairwise dependencies are allowed), and *n-variate* models. In this work we focus on univariate and bivariate algorithms because they provide a good complexity-accuracy trade-off. Dealing with n-variate models allows for a great capability of modeling, but at the cost of learning a complex probabilistic model at each iteration. Concretely we use UMDA, UMDA$_g$ and MIMIC algorithms.

A. UMDA

Univariate algorithms suppose that the n-dimensional joint probability distribution is factorised as

$$P(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} P(x_i) \tag{2}$$

That is, it assumes independence among all variables and no structural learning is needed. Therefore, only marginal probabilities are required during parameter learning.

The *Univariate Marginal Distribution Algorithm* (UMDA) [10] is, perhaps, the clearest representative of these models. In the discrete case, i.e. when variables take a finite number

of states, the model in eq. 2 is used , and marginal probabilities for each variable are estimated by using the frequencies found in D_{tra} (Laplace correction is applied to smooth the resulting probabilities). In the continuous case the *gaussian* UMDA or (UMDA$_g$) [11], uses the normal distribution to model the density of each variable, and the joint density is factorized as the product of all the unidimensional and independent normal densities:

$$\begin{aligned} f(\mathbf{x};\theta) &= f_N(\mathbf{x};\mu,\mathbf{\Sigma}) = \prod_{i=1}^{n} f_N(x_i;\mu_i,\sigma_i^2) \\ &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma_i} \cdot e^{-\frac{1}{2}\left(\frac{x_i-\mu_i}{\sigma_i}\right)^2} \end{aligned} \quad (3)$$

Thus, model induction is reduced to the estimation of μ and σ^2 for each variable.

With respect to sampling, it is clear that in both cases each variable can be independently sampled.

B. MIMIC

In bivariate models the n-dimensional joint probability distribution is factorised as

$$P(x_1, x_2, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | pa(x_i)),$$

where $pa(x_i)$ is the variable which x_i is conditioned to. $pa(x_i)$ can be null, so there can be variables without parents.

In the *Mutual Information Maximising Input Clustering* algorithm [12] the probabilistic model has the shape of a chain ($X_{\pi_1} \rightarrow X_{\pi_2} \rightarrow \cdots \rightarrow X_{\pi_n}$), where π is a permutation of the n variables and X_{π_i} the element of the permutation in position i. Thus, all the nodes have one parent except the chain root. Structural learning is carried out in MIMIC as follows:

1) Select as root node (X_{π_1}) the variable X_i with minimum entropy $H(X_i)$.
2) For the remaining nodes, (X_{π_i}) is that variable X_i which maximizes, $I(X_{\pi_{i-1}}, X_i)$.

Again we use Laplace correction when estimating the probabilities. In this case variables are sampled by following the chain order. Once a variable has been sampled, e.g. $X_i = a$, its child variable X_j can be sampled from the marginal distribution $P(X_j|X_i = a)$.

V. EDA-BASED MODELS TO APPROACH WCOR

In this section, we introduce the two approaches proposed to deal with the problem of learning weighted LFRBSs, but first we briefly review the main aspects of the GA proposed in [4] to instantiate the WCOR methodology. In all cases, our departure point is the representation of individuals and fitness function described in Section III.

A. GA-based WCOR learning algorithm

As we are dealing with a hybrid representation the more relevant points in the GA algorithm are the genetic operators it uses. The crossover is carried out as follows: first, two individuals c_1w_1 and c_2w_2 are selected as parents; second, the integer part (c) of both individuals are crossed giving rise to two offsprings c_1' and c_2'; third, the real part (w) of both individuals is crossed by using max-min-arithmetical crossover which produces four offsprings w_1, w_2, w_3, w_4 in the following way: if a and b are the numbers in position $j > m$ of cw_1 and cw_2 respectively, then the corresponding position in the off-springs is $a(1-\alpha) + b\alpha$, $b(1-\alpha) + a\alpha$, $min(a,b)$ and $max(a,b)$; fourth, the offspring separately obtained are combined to obtain eight complete offsprings: c_1w_1, c_1w_2, c_1w_3, c_1w_4, c_2w_1, c_2w_2, c_2w_3 and c_2w_4.

With respect to mutation, classical mutation is used in the integer part (c) while a new number in [0,1] is randomly generated when a position of the numerical part (w) is mutated.

B. UMDA-wCOR and MIMIC-wCOR learning algorithms

The ways we propose to apply EDAs to the WCOR problem arise as a direct adaptation of the GA-wCOR algorithm described in [4]. Therefore, individuals are composed of an integer and a real part.

First, we have implemented two algorithms, UMDA-wCOR and MIMIC-wCOR, whose probabilistic models are also composed of two independent parts: One of them used to learn and sample the consequents of the rules and the other one used to model the weights.

In the UMDA-wCOR algorithm (figure 3), not only the two parts are supposed to be independent, but all variables of the probabilistic model. This is equivalent to using a combination of UMDA and UMDA$_g$. The MIMIC-wCOR uses the MIMIC algorithm to model the integer part of the probabilistic model, but also assumes independence among all variables which represent the weights (figure 3).

Notice that in this way the population is the only nexus between consequents and weights. That is, as both models probabilistic models are learnt from the same set of individuals (the best half in the population) we are indirectly considering some relations between the different components (consequents and weights).

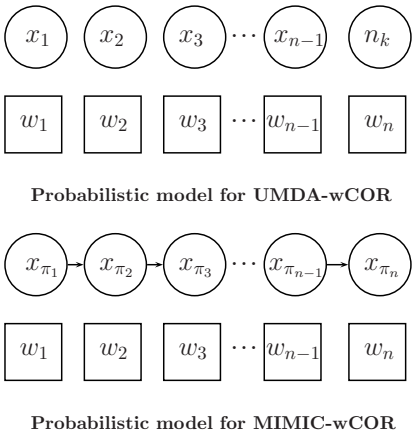


Fig. 3. Representation of the probabilistic models used in UMDA-wCOR and in MIMIC-wCOR

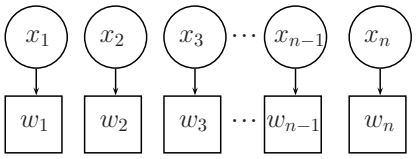
C. UMDA-cwCOR and MIMIC-cwCOR learning algorithms

Whereas in the previous algorithms the consequents and their weights are treated independently by the probabilistic model, in this section we propose two alternatives in which the relation between each rule and its weight is explicitly gathered.

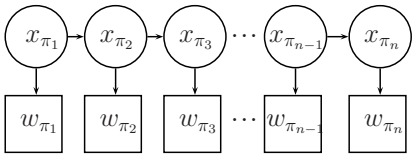
The idea is that the weigth assigned to a rule should be clearly dependent on the value selected as consequent for that rule. That is, for a given sub-space S_i we can have two possible *good* rules as "if S_i then $Y = c_i$ with $w_i = 0.2$" and "if S_i then $Y = c_j$ with $w_i = 0.9$, where clearly the weight is highly dependent on the consequent. In UMDA-wCOR and MIMIC-wCOR these dependences are missed because the parameters (mean and variance) for w_i are estimated without taking into account its associated consequent.

Here we propose to learn the weights *conditional* on the value selected for its corresponding consequent. This is easy to do in the EDAs paradigm, because we can easily express these kinds of dependences by using the probabilistic graphical model. Thus, we propose to use UMDA-cwCOR and MIMIC-cwCOR algorithms whose graphical structure is depicted in Figure 4. As we can see, again we use UMDA and MIMIC for the integer part but now we use a mixed probabilistic model instead of two separate models in order to explicitly reflect the dependence between consequents and weights.

These new models do not increase the complexity of the structural learning task (with respect to UMDA-wCOR and MIMIC-wCOR) because we simply add a link $c_i \rightarrow w_i$ for each sub-space. With respect to parameter learning in the numerical part, it is a bit more costly in space (but not in time) due to the fact of learning an unidimensional normal distribution conditioned to each element of cons(S_i) instead of a single one for each sub-space S_i.



Probabilistic model for UMDA-cwCOR



Probabilistic model for MIMIC-cwCOR

Fig. 4. Representation of the probabilistic models used in UMDA-cwCOR and in MIMIC-cwCOR

VI. Experimental evaluation

In order to carry out an experimental evaluation of the proposed schemes, we have tested them over a significant set of problems. In this section, we describe the settings as well as the results obtained.

A. Test suite

For our experiments, we have used four laboratory problems borrowed from the FMLib repository (`http://decsai.ugr.es/fmlib`) as well as three real-world problems, two of them also from FMLib. The goal is to model all of them by learning a LFRBS.

The four laboratory functions have two predictive and one output variable. The training sets are uniformly distributed in the two dimensional input space (x_1, x_2) and the test sets have been generated randomly. Next, we show the functions, the ranges of variables and the sizes of the training ($size_{tr}$) and test ($size_{ts}$) sets.

- Function F_1:

$$F_1(x_1, x_2) = x_1^2 + x_1^2$$
$$x_1, x_2 \in [-5, 5] \text{ and } F_1(\cdot) \in [0, 50]$$
$$size_{tr} = 1681,\ size_{ts} = 168$$

- Function F_2:

$$F_2(x_1, x_2) = 10 \frac{x_1 - x_1 x_2}{x_1 - 2x_1 x_2 + x_2}$$
$$x_1, x_2 \in [0, 1] \text{ and } F_2(\cdot) \in [0, 10]$$
$$size_{tr} = 674,\ size_{ts} = 67$$

- Function F_3:

$$F_3(x_1, x_2) = e^{x_1} sin(x_2)^2 + e^{x_2} sin(x_1)^2$$
$$x_1, x_2 \in [0, 1] \text{ and } F_3(\cdot) \in [0, 10]$$
$$size_{tr} = 1089,\ size_{ts} = 108$$

- Function F_4:

$$F_4(x_1, x_2) = x_1^2 + x_1^2 - cos(18x_1) - cos(18x_2)$$
$$x_1, x_2 \in [-1, 1] \text{ and } F_4(\cdot) \in [-2, 3.383]$$
$$size_{tr} = 1681,\ size_{ts} = 168$$

The two real-world problems from the FMLib repository are related to the field of engineering.

- Problem $ele1$: This consists in finding a model that relates the *total length of low voltage line* installed in a rural town to the *number of inhabitants in the town* and the *mean of the distances from the center of the town to the three furthest clients in it*. The goal is to use the model to estimate the total length of line being maintained.

Therefore, we have two predictive variables ($x_1 \in [1, 320]$ and $x_2 \in [60, 1673.33]$) and one output variable defined in $[80, 7675]$. The cardinality of the training and test sets are 396 and 99 respectively.

- Problem $ele2$: In this case, the model tries to predict the minimum maintenance costs. There are four input variables: *sum of the lengths of all streets in the town, total area of the town, area that is occupied by buildings*, and *energy supplied to the town*.

The domains for the four predictive variables are: $[0.5, 11]$, $[0.15, 8.55]$, $[1.64, 142.5]$ and $[1, 165]$. The output variable takes its value in $[64.47, 8546.03]$. For this problem, the size of the training set is 844, whereas the number of instances for the test is 212.

The problem not borrowed from FMLIB belongs to the field of farming.

- Problem $sheeps$: The frame in which the problem is defined is a genetic scheme launched in Castilla-La Mancha (Spain) with the aim of improving milk production figures in Manchego ewes. The main parameter in this scheme is

the genetic merit of an animal, which is estimated by using a standard methodology (BLUP). However, before an ewe becomes a mother and a lactation is controlled, BLUP cannot be applied and then the pedigree index (the arithmetical mean between father and mother genetic merit) is used. The data set used in this task contains two predictive variables (father and mother genetic merit) and the goal (variable) is to predict the animal genetic merit by using weighted LFRBSs instead of the pedigree index. The cardinality of the training and test sets are 1421 and 711 respectively.

B. Evaluation/fitness function

As mentioned in Section III the MSE or any of its variants is used to measure the goodness of a given solution. Concretely, we use the *Root Mean Squared Error* (RMSE) to measure the error committed by our system when used to predict the instances in the training set. Given an individual c or cw and its corresponding (weighted) LFRBS F, if $\hat{y}$ is the output generated by F for an input $\mathbf{x}$ while y is the true output, then

$$RMSE(cw) = RMSE(F) = \sqrt{\frac{1}{|\mathbf{D}|}\sum_{i=1}^{|\mathbf{D}|}(\hat{y}_i - y_i)^2}$$

where $|\mathbf{D}|$ is the number of records in the data set.

It is clear that the goal is to find the system with the smallest error; however, as in our implementation we always maximize, we have used the inverse of RMSE as fitness function, i.e., $fitness(cw) = \frac{1}{RMSE(cw)}$.

Finally, and before describing the parameter setting used for the algorithms used in our experiments, we should first out that our search algorithms have been written in Java and for the definition and evaluation of the fuzzy rule systems, we interact with FuzzyJess [13], [14] also written in Java.

C. Algorithms and settings

In the case of the COR approach we have used as search algorithms the GA [3] and the estimation of distribution algorithms UMDA and MIMIC [15], whereas for the wCOR, we have tested the algorithms (GA and EDAs) described in section V. Most parameters are common to both EDAs and GAs: The population size (*popSize*) has been fixed to 512 [2] and the population D_k is obtained from the best (*popSize*) individuals of $D_{k-1} \cup D_{aux}$, where D_{aux} is the population generated by sampling, in case of EDAs, or by application of the genetic operators, in case of the genetic algorithm.

For EDAs we have used an standard setting, i.e., they estimate the model at the k-th generation from the best 50% individuals of the population D_{k-1}.

In the case of genetic algorithms, the way that offspring is generated differs from GA-COR to GA-wCOR. In both cases the crossing individuals are selected by ranking but, in the first case, $popSize/2$ couples of individuals are crossed by using the typical one-point crossover, whereas in the second case, since each crossover produces eight individuals, only $popSize/8$ couples are selected.

With respect to the stopping condition, each algorithm can evolve up to a maximum of 250 generations. However, it stops if there is no improvement in the average fitness from one generation to another.

In all the experiments, we have considered symmetric partitions of the real domain with triangular fuzzy sets, with both 5 and 7 labels, to represent each variable.

D. Results and analysis

Each one of the algorithms has been run 20 times. Average RMSE of the obtained models (training \ test), as well as the average number of rules and of evaluations carried out, are shown for the cases of 5 (table I) and 7 (table II) labels. Since the criteria being optimized by the search algorithms is the RMSE over the training set, we have marked this result in bold.

As can be seen, results (training \ test) are better in almost all cases when using 7 labels, although the obtained models have more rules and so are less comprehensible. At this point, it is worth noticing that the number of rules has not been taken into account in the fitness function, and no optimization of the resulting system has been carried out.

Focusing on the search algorithms, which are the goal of our study, it is clear that those which use weights are more precise modeling the training dataset than those which does not. Particularly, it can be seen that cwCOR-based algorithms outperform the rest in most cases, using either 5 or 7 labels. In the first case, UMDA-cwCOR seems to be the outstanding algorithm since it achieves the lower error in 5 of the 7 problems. When using 7 labels for variable, this algorithm is the most accurate only on three occasions, whereas on another three is MIMIC-cwCOR.

However, in many cases differences among results yielded by the algorithms are very small. Therefore, we have determined which of them does not have a significant difference with the best one for each problem by means of Mann-Whitney unpaired tests. Table III shows the results of the comparisons. We have marked with $\circ$ that algorithm which offers the lowest error for the training prediction. Afterwards, we have marked those which does not present a significant difference ($p-value > 0.05$) with $\bullet$. While this study can be viewed as an analysis evaluation of the search algorithms, it is also necessary to study the errors over the test set in order to evaluate the predictive capability of the obtained LFRBSs. In this case, the reference algorithm has been marked with $\square$ whereas the algorithms which show no statistical difference have been marked with $\blacksquare$ (Table III).

From the statistical analysis, in the case of training, UMDA-cwCOR and MIMIC-cwCOR are the algorithms of choice, being in the outstanding group in 5 out of the 7 problems. With respect to test errors the results vary when using 5 or 7 linguistic labels. Thus, when using 7 labels the same conclusions apply, but in the case of 5 labels there is not a clearly dominant approach, and in any case algorithm UMDA-wCOR seems to be as competitive as the cwCOR approach.

When evaluating the test datasets with the models that use 5 labels, the tendency is not so defined, since both UMDA-cwCOR and MIMIC-cwCOR algorithms stop being

[2]Results for population 256 can be seen in http://info-ab.uclm.es/SIMD/CEC06

Results using 5 labels for each variable.

Problem	GA-COR	GA-wCOR	UMDA-COR	UMDA-wCOR	UMDA-cwCOR	MIMIC-COR	MIMIC-wCOR	MIMIC-cwCOR
f1	2.6741 \ 2.3314 25 – 40059.5	2.4846 \ 2.2187 25 – 95945.85	2.6886 \ 2.3686 25 – 12590.9	2.2053 \ 1.9348 25 – 116623.25	**2.1549** \ 1.8537 25 – 117163.4	2.6886 \ 2.3737 25 – 7953.55	2.3599 \ 2.0922 25 – 54660.65	2.1673 \ 1.8581 25 – 122720.25
f2	0.6047 \ 0.5844 23 – 15834.2	0.58 \ 0.5689 22.85 – 104918.1	0.6047 \ 0.5844 23 – 4018.35	**0.5464** \ 0.528 23.7 – 69000.1	0.5472 \ 0.5285 24.55 – 68375.85	0.6047 \ 0.5844 23 – 4562.4	0.5682 \ 0.5461 23 – 47360.9	0.5473 \ 0.5286 24.45 – 71760.65
f3	525.0568 \ 510.7837 5 – 37769.7	**497.2208** \ 460.0705 5 – 116766.1	635.9083 \ 623.3604 21.9 – 5514.6	581.3778 \ 571.0836 22.2 – 82078.85	577.0631 \ 580.3826 22.9 – 68863.75	575.6649 \ 563.0907 12.95 – 26277.85	550.1683 \ 525.0203 12.65 – 104021.15	519.6225 \ 503.2405 17.6 – 81057.9
f4	1.0049 \ 1.056 19.85 – 51178	1.0025 \ 1.0495 20.45 – 106018	1.0056 \ 1.0547 20.8 – 10481.15	0.9916 \ 1.0356 23.8 – 121084.4	**0.9906** \ 1.0099 24.9 – 97288.4	1.0057 \ 1.0545 20.9 – 13642.95	1.0021 \ 1.05 20.95 – 60043.2	0.9913 \ 1.0209 24.95 – 97450.35
ele1	591.7298 \ 586.5343 15 – 11728.55	575.0993 \ 610.8644 14.8 – 100844.25	591.9458 \ 587.1284 15.55 – 5842.6	572.2036 \ 608.7981 16.8 – 108828.15	**571.6744** \ 613.3346 17.3 – 84309.7	591.9986 \ 586.7362 15.45 – 6861.3	577.1068 \ 601.3017 15.85 – 38269.5	571.7257 \ 615.8812 17.5 – 71144
ele2	413.1216 \ 422.6315 47.3 – 115912.35	422.2865 \ 428.777 46.1 – 104963.05	404.4527 \ 417.615 50.45 – 21028.45	371.3274 \ 371.7103 56.1 – 126559.95	**370.2879** \ 376.2354 60.7 – 127806.5	404.2662 \ 415.929 49.6 – 23100.65	389.618 \ 393.5926 50.15 – 99549.8	370.3089 \ 374.8992 61.9 – 128176.9
sheeps	7.7329 \ 7.0855 15 – 13584	7.343 \ 6.6741 15.85 – 102100.6	7.7329 \ 7.0855 15 – 9328.45	7.2977 \ 6.6475 18 – 54579.95	**7.2878** \ 6.6457 18 – 63533.15	7.7329 \ 7.0855 15 – 7842.8	7.3998 \ 6.7304 16.1 – 62385.85	7.2888 \ 6.6424 17.95 – 57466.7

TABLE I

AVERAGE ERRORS IN TRAINING AND TEST, AVERAGE NUMBER OF RULES AND NUMBER OF EVALUATIONS FOR 20 EXECUTIONS

Results using 7 labels for each variable.

Problem	GA-COR	GA-wCOR	UMDA-COR	UMDA-wCOR	UMDA-cwCOR	MIMIC-COR	MIMIC-wCOR	MIMIC-cwCOR
f1	1.7902 \ 1.6909 48.7 – 108601.7	2.2314 \ 2.1565 47.4 – 104844.2	1.7018 \ 1.5728 49 – 18589.35	1.4553 \ 1.3988 49 – 128235.75	**1.4348** \ 1.3629 49 – 128066.7	1.6964 \ 1.5666 49 – 20033.4	1.6538 \ 1.5832 49 – 86102.55	1.4542 \ 1.3838 49 – 128144.7
f2	0.3868 \ 0.3579 45.8 – 109960.9	0.4377 \ 0.406 43.6 – 113125.2	0.383 \ 0.355 47 – 8516.45	**0.3433** \ 0.3188 48.55 – 128182.75	0.3442 \ 0.3207 48.7 – 128087.9	0.383 \ 0.355 47 – 10277.1	0.3643 \ 0.3384 47.05 – 64264.1	0.3443 \ 0.3206 48.7 – 128269.75
f3	461.1832 \ 447.7107 22.15 – 118280.05	465.7829 \ 445.4247 28.3 – 112122.25	471.2536 \ 472.6312 32.3 – 29063.8	435.8191 \ 435.8938 43.1 – 124196.2	432.7979 \ 434.2939 44.55 – 126631.5	459.944 \ 459.9902 25.95 – 52160.35	473.5072 \ 457.5737 44.15 – 70634.7	**405.2092** \ 407.3548 38.45 – 123925
f4	1.0067 \ 1.033 46.9 – 119632.7	1.0126 \ 1.0414 45.2 – 114516.5	1.001 \ 1.0091 45.4 – 24444.1	0.9649 \ 0.9724 48.2 – 127826	**0.9491** \ 0.962 49 – 127920.9	1.0013 \ 1.0116 45.55 – 43971.7	0.9973 \ 1.0056 48.9 – 92130.1	0.9547 \ 0.9612 49 – 127108.55
ele1	570.8154 \ 635.3562 25.6 – 34415.85	561.3602 \ 670.1182 25.55 – 109533.85	571.4037 \ 650.8432 25.4 – 9726.4	549.1604 \ 672.3695 27.15 – 94108.2	544.4554 \ 694.6266 27.55 – 98417.8	571.3502 \ 652.6484 25.35 – 11113.75	557.6415 \ 677.268 26.05 – 58143.3	**543.4306** \ 695.1677 27.5 – 108671.15
ele2	317.3053 \ 337.453 83.2 – 110309.1	333.1077 \ 355.9825 84.45 – 109866	283.7599 \ 298.0921 86.8 – 28181.35	258.8464 \ 269.0333 94.05 – 127754.25	254.2318 \ 265.3457 98.35 – 128183.5	283.097 \ 304.6788 85.85 – 33045.75	279.8552 \ 291.6212 90.85 – 107415.7	**253.723** \ 265.5097 98.2 – 127885.5
sheeps	7.2211 \ 6.9497 30.1 – 80290.85	7.2001 \ 6.9302 31.2 – 114364.65	7.2215 \ 6.9587 30.15 – 16486.85	7.0778 \ 6.8455 33.6 – 115882.4	**7.067** \ 6.8234 34.35 – 113290.45	7.229 \ 6.9513 30.15 – 18021.55	7.1574 \ 6.9095 31.75 – 72315.6	7.068 \ 6.8306 34.65 – 116548.6

TABLE II

AVERAGE ERRORS IN TRAINING AND TEST, AVERAGE NUMBER OF RULES AND NUMBER OF EVALUATIONS FOR 20 EXECUTIONS

Tests for 5 labels for each variable

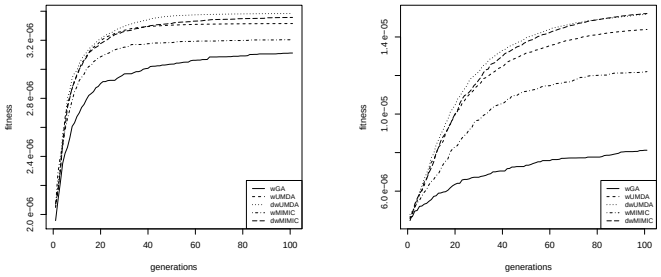
Problem	GA	wGA	UMDA	wUMDA	cwUMDA	MIMIC	wMIMIC	cwMIMIC
f1					○ □			● ■
f2				○ □				
f3		○ □						
f4				●	○ □			●
ele1	□		■	●	○	■		●
ele2				□	○ ■			● ■
sheeps		■		■	○ ■			● □

Tests for 7 labels for each variable

Problem	GA	wGA	UMDA	wUMDA	cwUMDA	MIMIC	wMIMIC	cwMIMIC
f1				●	○ □			● ■
f2				○ □	■			■
f3								○ □
f4				■	○ ■			□
ele1	□		■		●	■		○
ele2					● □			○ ■
sheeps					○ □			● ■

TABLE III

STATISTICAL TESTS FOR TRAINING AND TEST RESULTS



ele1 (7 labels) ele2(5 labels)

Fig. 5. Fitness vs Generation for the wCOR algorithms

in the outstanding groups in some cases and the difference in performance with UMDA-wCOR is not so clear. In this point it is worth noticing that no mechanism for overfitting prevention has been used during the search process.

Discarding the COR algorithms, it seems that GA-wCOR and MIMIC-wCOR are clearly worse than the other algorithms which use weights. In order to see whether the poor results are due to premature convergence, we have plotted in Figure 5 the evolution (average over 10 runs) of the best fitness through generations[3] for problems ele1 (7 labels) and ele2 (5 labels). As can be seen, the poor results of the two algorithms are not due to premature convergence. In fact UMDA-cwCOR even converges fastest.

The results obtained lead us to believe that using conditional weights with the wCOR approach is the best option. cwUMDA, despite its simplicity, is in most cases the best or in the group of the best algorithms. Moreover, the graphs show that, although it gets the best result, it nevertheless converges faster.

VII. CONCLUDING REMARKS

In this work, we have made a first approach to the use of EDAs as search algorithm when learning weighted LFRBSs through the wCOR methodology. The use of probabilistic models as main element of the evolution process allows a more precise estimation of weights, which translates to more accurate models. We think that the more remarkable points of the paper are: (1) the way in which one can take advantage of the graphical language used by EDAs in order to incorporate domain knowledge, and (2) the experimental analysis of the wCOR methodology carried out, which uses 7 different problems while previous analysis where based on a single problem.

As future work, we plan to extend our research in different directions: (1) once the applicability of EDAs to the wCOR methodology has been shown, we think that some mechanism for overfitting prevention should be incorporated to the search process; (2) less rules means more interpretable systems, so we plan to study how to reduce the number of rules in the discovered systems but without to degrade its performance. Perhaps, the problem can be approached as a multiobjective one by considering number of rules and error as fitness measures; and (3) more complex models of EDAs or different ways of using domain knowledge are also topics that worth to be studied.

[3]Graphics of convergence for all problems with 5 and 7 labels can be consulted in http://www.info-ab.uclm.es/SIMD/CEC06

ACKNOWLEDGMENT

This work has been partially supported by the Consejería de Educación y Ciencia (JCCM) under project PBI-05-022.

REFERENCES

[1] O. Cordón, F. Herrera, F. Hoffmann, and L. Magdalena, *Genetic fuzzy systems: Evolutionary tuning and learning of fuzzy knowledge bases*, World Scientific, 2001.

[2] E.H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, vol. 7, pp. 1–13, 1975.

[3] J. Casillas, O. Cordón, and F. Herrera, "COR: A methodology to improve ad hoc data-driven linguistic rule learning methods by inducing cooperation among rules," *IEEE Trans. on Systems, Man, and Cybernetics - Part B: Cybernetics*, vol. 32, pp. 526–537, 2002.

[4] R. Alcalá, J. Casillas, O. Cordón, and F. Herrera, "Improving simple linguistic fuzzy models by means of the weighted COR methodology," in *Advances in Artificial Intelligence - IBERAMIA 2002 (F.J. Garijo, J.C. Riquelme, M. Toro, Eds.)*. 2002, pp. 294–302, Springer Velarg.

[5] P. Larrañaga and J.A. Lozano, *Estimation of distribution algorithms. A new tool for evolutionary computation*, Kluwer, 2001.

[6] L.A. Zadeh, "Outline of a new approach to the analysis of complex systems and decision processes," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 3, no. 1, pp. 28–44, 1973.

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

[8] L.A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning," *Information Science*, vol. 8, pp. 199–249, 1975.

[9] L.X. Wang and J.M. Mendel, "Generating fuzzy rules by learning from examples," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 22, no. 6, pp. 1414–1427, 1992.

[10] H. Mühlenbein, "The equation for response to selection and its use for prediction," *Evolutionary Computation*, vol. 5, pp. 303–346, 1998.

[11] P. Larrañaga, R. Etxeberria, J.A. Lozano, and J.M. Peña, "Optimization by learning and simulation of Bayesian and Gaussian networks," Tech. Rep. EHU-KZAA-IK-4-99, University of the Basque Country, 1999.

[12] J.S. de Bonet, C. L. Isbell, and P. Viola, "MIMIC: Finding optima by estimating probability densities," *Advances in Neural Information Processing Systems*, vol. Vol. 9, 1997.

[13] R. Orchand, "Fuzzy reasoning in Jess: The FuzzyJ toolkit and FuzzyJess," in *Proccedings of the 3rd International Conference on Enterprise Information Systems*, 2001, pp. 533–542.

[14] IRG-NCR, "FuzzyJ toolKit & Fuzzy Jess URL," `http://www.iit.nrc.ca/IR_public/fuzzy/fuzzyJToolkit2.html`, 2005.

[15] J. Flores, J.A. Gámez and and J.M. Puerta, "Learning linguistic fuzzy rules by using Estimation of Distribution Algorithm as the search engine in the COR methodology," in *Towards a new evolutionary computation. Advances in Estimation of Distribution Algorithms*. Studies in Fuzziness and Soft Computing, vol. 192, 2005, pp. 259-280, Springer.