Inductive Learning of Membership Functions and Fuzzy Rules

Inductive Learning of Membership Functions and Fuzzy Rules

Michèle Sebag and Marc Schoenauer

LMS CNRS URA 317 and CMAP CNRS URA 756,
Ecole Polytechnique, 91128 Palaiseau Cedex, France
e-mail : sebag@xela.polytechnique.fr

Abstract

By means of membership functions, measurable informations (size expressed in inches, age expressed in years) are converted into linguistic qualifiers (tall, young). It is emphasized that the acception of linguistic qualifiers is context-dependant and may depend on several attributes. Therefore, the difference between membership functions and fuzzy rules vanishes. A method inspired from inductive learning is presented : it performs the extraction of membership functions and / or fuzzy rules from examples.

1. Introduction

Fuzzy logic allows to deal with fuzzy and high-level notions (e.g. *being small*) while processing low-level informations (e.g. *size is 3.4 inches*) by means of membership functions : a given size satisfies the concept of *being small* to a given extent [39]. The primary representation of the problem domain generally involves measurable real-valued informations ; in the meanwhile, experts deal with yet generalized informations. Membership functions so enable to bridge the gap between such different levels of representation.

Membership functions may be designed by the expert or by a pool of experts by means of questionnaires [41, 12] ; this method is the most widely used up to now [24]. Another possibility is to use neural networks to learn membership functions : neural networks have been used to optimize the fuzzy conclusions of control rules [5], as well as the hypotheses of control rules [36]. Genetic algorithms [18] may also be used to refine the membership functions provided by the expert, or even to learn them from scratch [10, 11].

Acquiring membership functions from the expert is tedious indeed ; moreover, our claim is that designing membership functions may be just as complex as designing fuzzy rules, for the following reasons. First of all, it is well acknowledged that membership functions are context-dependant (a *small* elephant is bigger than a *small* mouse) ; so one starts from scratch when considering a new context [6]. Second, and overall, the acception of a linguistic qualifier may depend on a number of independant features even within a given context. For instance, the acception of a *small crack* varies whether it is a circumferential or a longitudinal crack ; it also depends on the material the crack arises in : in a cement plate, the size of *small crack* may be about 1 inch ; in ceramics, a small crack is about ten times smaller. These remarks lead to model a membership function as a function of several numerical or qualitative attributes ; the difference between designing a membership function and a fuzzy rule thus vanishes.

To address this problem, we propose a method inspired from empirical inductive learning [25, 28] to automatically extract membership functions and fuzzy rules from examples. This method extends a crisp induction algorithm [33] : this extension intends to both handle real-valued attributes and learn fuzzy or real-valued conclusions.

The paper is organized as follows :
Section 2 gives a short introduction to inductive learning and presents the main algorithmic choices with respect to the characteristics of the available data. Section 3 details the proposed induction algorithm, called constrained induction ; a more detailed presentation can be found in [33, 32]. Section 4 focuses on adaptating constrained generalization to building fuzzy rules in the sense of [24] ; the turning of crisp premises and conclusions into conjunction of membership functions is discussed. Last, section 5 discusses the advantages and drawbacks of empirical inductive learning with respect to neural networks and genetic algorithms, as to build fuzzy rules.

2. Introduction to inductive learning ; basic choices

In its simplest state, the goal in inductive learning is the same as in discriminant data analysis : given examples described by a set of attributes, one attribute, called *conclusion*, is to be estimated by a function of the other attributes, called *descriptors* [25, 28]. Some of the main differences are the followings :

– The estimator built by inductive learning is a set of rules (a rule can be viewed as an hyperectangle of the description space labelled by the conclusion of the rule : *If x in interval I, and y in interval J, and ..*

then Class C_k), instead of a polynomial expression ;
- Inductive learning takes into account both the links between attributes and the partial solutions yet available, called background knowledge of the problem ;
- Inductive learning is supposed to cope with sparse examples, while data analysis requires statistically representative data.

Most attention in inductive learning is paid so far to estimating a discrete conclusion, i.e. to discrimination problems (regression problems are considered further on). There are basically two strategies for building discriminant rules : top-down algorithms evaluate a descriptor from the overall set of examples ; bottom-up algorithms consider the examples one by one.

2.1. The Top-Down Approach

The most widely used induction strategy is top down. The discriminant power of any attribute is evaluated according to a criterion inspired from the information quantity of Shannon [34]. Let x be an attribute of the problem domain, and let v_i be a value of this attribute. The quantity of information of the condition (also called *selector*) $[x = v_i]$ is evaluated by

$$I([x = v_i]) = -\sum_{j=1}^{P} p_{i,j} \; log(p_{i,j})$$

where P is the number of classes $C_1, ..C_P$ to discriminate
$p_{i,j}$ is the conditional probability of class C_j, knowing that attribute x takes value v_i.

Quantity of information of descriptor x is then given by :

$$I(x) = \sum_{i=1}^{N} \; p_i \; I([x = v_i])$$

where N is the number of values of descriptor x.
and p_i denotes the frequence of value v_i for attribute x.

The evaluation above considers a discrete attribute. When dealing with a numerical attribute x, the simplest possibility is to associate to threshold v_i the quantity of information $I(x, v_i)$ corresponding to selectors $[x < v_i]$ and $[x > v_i]$. The quantity of information of x is set to the maximum value of $I(x, v_i)$ for v_i belonging to the domain of x.

A decision tree is then iteratively built as follows (Fig. 1) : at any step, the most discriminant attribute is chosen. The set of examples is partitionned according to the possible values of this attribute[1]. The choice of the most discriminant attribute is performed again on any such subset of examples.

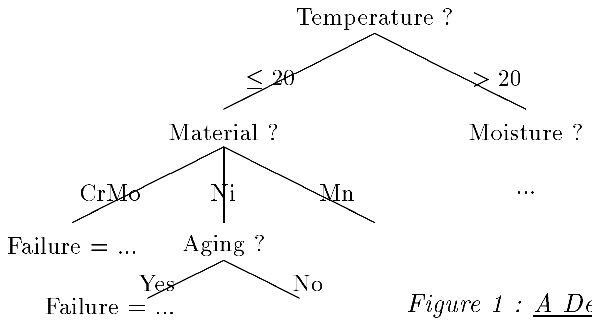


Figure 1 : *A Decision Tree*

This approach stems from the data analysis field [13]. Its good properties : polynomial complexity, low sensitivity to noisy data and its algorithmic simplicity, explain its wide use in the machine learning community [28, 29, 7, 8]. Many improvements are concerned with incremental processing of bunches of data [37], dealing with numerical attributes [17, 38], dealing with missing values, handling first order predicate logic[29, 30, 7, 8, 11],...

2.2. The Bottom-Up Approach

A major requirement of the top down approach is to deal with statistically representative data ; this requirement is not always met in the machine learning environment. Second, disjunctive concepts may be difficult to learn by top down approaches. The prototypical example of this limitation is that of learning the "XOR" concept, where classes are defined with respect to two boolean attributes : $C = (x_1 \neq x_2)$. Relevant attributes x_1 and x_2, considered as standalone, are associated a weak discriminant power. Hence other (irrelevant) attributes are likely to be first retained and relevant attributes are likely to be found in the ultimate levels of the decision tree.

The explanation is the following : when a class involves a disjunction of several sub-classes, the contributions of the sub-classes add to the global criterion that becomes difficult to interpret.

On the opposite, examples are considered one by one in the bottom-up approach. Given one example, one attemps to find rules covering (one

[1]When dealing with a numerical attribute x, the data set is then partitionned according to the threshold value maximizing quantity $I(x, v_i)$.

also says *generalizing*) this example, and covering little or none counter-examples, i.e. examples belonging to other classes than the current example. The disjunction of such rules form the *star* of the current example [25, 26]. There is no problem of learning disjunctive concepts that way : given one example, the algorithm only finds out the (conjunctive) subclasses the current example belongs to. As a counterpart, the bottom-up approach traditionnally is more sensitive to noise : rules learned from erroneous examples are meaningless and/or misleading.

In that approach, the main choice concerns the level of generality expected from the rules. Given some positive and negative examples, generalization can be viewed as a search [27] : the solution space, also called Versions Space, has a lower bound and an upper bound. The lower bound, called set S, is composed of the rules most specific still covering the positive examples ; the upper bound, called set G, is composed of the rules most general rejecting the negative examples.

Retaining only the most specific rules, i.e. performing a *maximally specific generalization* [25] maximizes the certainty of the rules ; the counterpart is that the problem domain may not be completely covered by maximally specific rules, especially if the available data are sparse. Hence many further cases will be classified as *Unknown* (no rule fired).
On the other hand, retaining the maximally general rules (*maximally discriminant generalization*) may lead to conflicting rules.

3. Constrained Generalization

The characteristics of the induction algorithm we extended to learning fuzzy rules are ruled by the tackling of disjunctive problems with sparse data. So, this algorithm is a bottom-up maximally discriminant generalization. The main difference with respect to the star algorithm of R. Michalski et coll. [25, 26], is the intermediate representation of examples we use : counter-examples are represented as constraints on the generalization of the current example.

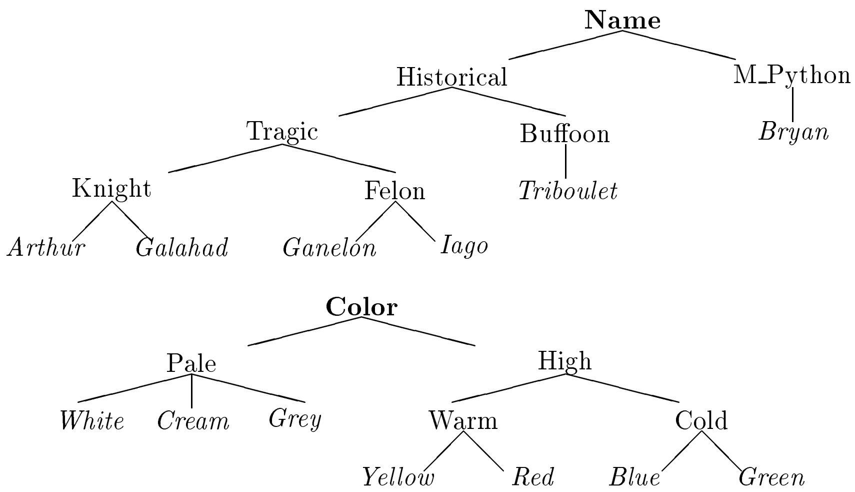
3.1. Broad Outlines

	Name	*Color*	*Nb Questions*
Ex	*Arthur*	*Blue*	*3*
Ce_1	*Ganelon*	*Grey*	*7*
Ce_2	*Iago*	*Yellow*	*4*
Ce_3	*Bryan*	*Green*	*10*
Ce_4	*Triboulet*	*Cream*	*5*

Table 1 : Positive and Negative Examples.

Let us consider a toy problem for the purpose of illustration (Table 1). The concept to learn is that of *Hero* ; attributes are the name of the person, its favorite color and the number of questions s/he asks. Attribute *Nb_Questions* is numerical ; *Name* and *Favorite Color* are tree-structured (Fig 2).

Figure 2 : <u>*Tree-Structured Attributes*</u>



Any discriminant generalization of example Ex must reject negative example Ce_1. In the case of attribute *Color*, the most general value covering $Blue$ and rejecting $Grey$ is $High$. The corresponding selector rejecting Ce_1 thus is : $[Color = High]$. The disjunction of the most general selectors covering Ex and rejecting Ce_1 is :

$$[Name = Knight] \bigvee [Color = High] \bigvee [Nb\_Questions = [0, 6]]$$

This disjunction sets a constraint upon the generalization of Ex : let the solution space be the set of conjunctions of selectors covering Ex and rejecting all negative examples ; then any term in the solution space must satisfy the logical constraint above.

More generally, such a constraint upon the generalization of any positive term can be derived from any negative example. Generalization may be thought of as a constrained optimization problem : find the maximally general terms yet satisfying all constraints derived from negative examples. After proposing a representation of such constraints, we focus on their pruning (in 3.3) and handling in order to build crisp rules (in 3.4) or fuzzy rules (in 4).

3.2. Formalizing Constraints

In the following, the notation $x(T)$ stands for the value of attribute x in term T ; $x(T)$ thus is a qualitative value in the case of a tree-structured attribute, and an interval or a single value in the case of a numerical attribute. The set of maximally general terms covering a positive example Ex and rejecting negative examples $Ce_1, \ldots, Ce_N$ is denoted G$(Ex, Ce_1, ..Ce_N)$.

For any negative example Ce_j and any attribute x_i,
• Let $E_{i,j}$ be, if it exists, the most general value (respectively the widest interval) such that it covers (resp. includes) value $x_i(Ex)$ and does not cover (resp. include) value $x_i(Ce_j)$ if attribute x_i is qualitative (resp. numerical).
• Let $I(j)$ be the set of attributes such that value $E_{i,j}$ is defined. $I(j)$ denotes the set of discriminant attributes.
By definition, the most general selectors covering Ex and rejecting Ce_j (if we restrict to operator '='), are the $[x_i = E_{i,j}]$, for x_i in $I(j)$. Hence :

$$G(Ex, Ce_j) \;=\; \bigvee_{x_i \in I(j)} [x_i = E_{i,j}]$$

The above expression is called *constraint induced by Ce_j on the generalization of Ex*. Let O_i be the domain of attribute x_i, and let Ω denote the cross product of domains O_i : $\Omega = O_1 \times \ldots \times O_K$. A constraint may then be represented as a subset of Ω.

Definition (constraint) :
Given positive example Ex, we associate to any negative example Ce_j the subset of Ω denoted $Constraint(Ex, Ce_j)$, defined by :

$$Constraint(Ex, Ce_j) = (V_{i,j})_{i=1}^{N}$$

$$where\ V_{i,j} = \begin{cases} E_{i,j} & if\ E_{i,j}\ is\ defined \quad (\Leftrightarrow\ i \in I(j)) \\ \phi & otherwise \end{cases}$$

The empty value ϕ is assumed more specific than any value V_i in any domain O_i. The negative examples described in (3.1) give rise to the following constraints :

	Name	Color	Nb Questions
$Constraint(Ex, Ce_1)$	$Knight$	$High$	$[0, 6]$
$Constraint(Ex, Ce_2)$	$Knight$	$Cold$	$[0, 3]$
$Constraint(Ex, Ce_3)$	$Historical$	$Blue$	$[0, 9]$
$Constraint(Ex, Ce_4)$	$Tragic$	$High$	$[0, 4]$

Table 2 : Constraints derived from negative examples.

3.3. Ordering and Pruning Constraints

The partial order relations defined on domains O_i classically induce a partial order relation on their cross-product Ω, denoted $\leq_\Omega$. One has :

$$((V_i)_{i=1}^K \leq_\Omega (W_i)_{i=1}^K) \Longleftrightarrow (\forall i = 1..K,\ V_i \leq W_i)$$

where $V_i \leq W_i$ means that value V_i is covered by value W_i, if attribute x_i is tree-structured, and that interval V_i is included in interval W_i if x_i is numerical.

This order relation enables to compare constraints and thus, negative examples.

Definition (nearest-miss) :
Given positive example Ex and negative examples $Ce_1, \ldots Ce_N$*,* Ce_i *is called* nearest miss *to Ex iff* $Constraint(Ex, Ce_i)$ *is minimal with respect to the order relation* $\leq_\Omega$ *among* $Constraints(Ex, Ce_j)$*, j = 1..N.*

Negative example Ce_2 is a nearest miss to Ex ; Ce_4 is not because $Constraint(Ex, Ce_2) \leq_\Omega Constraint(Ex, Ce_4)$. *Knight* is less general than *Historical* ; *Cold* is less general than *High* ; and $[0,3] \subset [0,4]$.

Given the above definition, it is shown that learning only needs positive examples and nearest-miss negative examples.

Proposition :
Given positive term Ex and negative examples $Ce_1, \ldots Ce_N$, assume without loss of generality that nearest-misses are examples $Ce_1, \ldots Ce_L, L \leq N$. Then

$$G(Ex, Ce_1, \ldots Ce_N) = G(Ex, Ce_1, \ldots Ce_L)$$

(Proof can be found in [32], with other results about pruning attributes). Therefore, negative examples that are not nearest-misses can be pruned without any loss of information[2].

[2]This proposition extends some previous results of Smith and Rosenbloom [35] : they show that only near-misses were necessary to learning in case of a convergent data set, i.e. when data are such that the set S of maximally specific generalizations equals the set G of maximally general generalization. Of course, this condition is seldom satisfied in real-world problems.

3.4. Building crisp rules

Let Ex be an example, and let $C_1, ..C_N$ denote the constraints on the generalization of Ex derived from the counter-examples to Ex. By construction, the maximally discriminant generalizations of Ex are terms which satisfy all constraints $C_1, .., C_N$. A term, conjunction of selectors, is said to satisfy constraint C iff it contains a selector more specific than a selector in C.

Maximally discriminant terms are then obtained by a depth-first exploration of these constraints.

Here is the pseudo-code of the constrained generalization algorithm[3]. Array *Selector[i]* stores the active selector of constraint C_i ; i_0 is the index of the current constraint.

```
Initialize()
    For i = 1..N
        Selector[i] = 0;
    G = { } ;  T = true ;   i0 = 1.
Main()
    Initialize();
    While (0 < i0 ≤ N),
        If (constraint C_i0 is not yet satisfied by term T)
            If (Selector[i0] ≠ 0)
                Remove from T the Selector[i0]-th selector of C_i0;
            Increment Selector[i0];
            Continue :
            If (C_i0 has Selector[i0] selectors)
                T = T ⋀ the Selector[i0]-th selector of C_i0;
            Else
                Selector[i0] = 0;
                If (Backtrack()) goto Continue;
                Else stop.
        Increment i0;
    EndWhile
    G = G ⋃ T ;// Term T is a solution
    if (Backtrack()) goto Continue; // to find other solutions.
    else stop.

Backtrack()
Let j be the index of the last active constraint (with Selector[j] ≠ 0);
    While (j > 0)
```

[3]This algorithm is implemented in C^{++}.

```
        Remove from T the Selector[j]-th selector of C_j;
        Increment Selector[j];
        If (C_j has Selector[j] selectors)
            i_0 = j;
            return True;
        Else
            Selector[j] = 0;
            Decrement j;
    EndWhile
    return False;
```

This procedure allows to find all maximally discriminant generalizations of Ex. These terms are perfectly discriminant, i.e. they do not cover any counter-examples ; this property is useless and may be damaging when dealing with noisy data. In that case, a uniform forgetting rate is applied when exploring constraints ; rules admitting exceptions can then be found.

Note that many discriminant terms are found that way. Therefore, selection is necessary ; some selection criteria (parameterized by the user) are designed to cope with the defects of the data. For instance, the allowed exceptions rate is set according to the noise in the data ; the minimum number of examples a rule must cover is related to the sparseness of the data ; similarly, the redundancy rate of rules allows to some extent to fight against the noise and sparseness of the data. A detailed discussion on these heuristic criteria can be found in [33].

4. Building fuzzy rules

The main limitation of the algorithm above lies in handling and estimating real-valued attributes. More generally, inductive learning is not good in directly dealing with real-valued attributes[4] and some pre-processing is usually performed, such as discretization (see [17] among many others) or fuzzyfication [11]. It is commonly acknowledged that discretization may lead to brittleness [38]. In opposition, membership functions are able to both model high nonlinearities and yet provide the user with smooth solutions. The only problem thus lies in designing good membership functions.

So, the same problem is eventually met in inductive learning and in fuzzy logic : in both cases, logic processing relies on some "decomposition" of the work space and the decomposition step is critical for any further processing. The expert has long been in charge of this task, which, besides being tedious

[4] Only recently was some attention paid to direct learning of a real-valued conclusion, i.e. to regression [42].

may lead to brittle solutions too. Several automatic approaches have been developped to this aim ; they will be discussed in the last section.

Our purpose in this section is to propose such an automatic approach by extending the constrained generalization to learning membership functions and fuzzy rules as well. This way, induction is used at any level of the logic processing : fuzzyfication transforms low-level informations into membership functions ; then membership functions are used as high-level descriptors, and inductively used in order to form rules. Extending constrained generalization involves three stages : constrained induction is first extended to perform regression, i.e. to estimate a real-valued conclusion, and the conclusion associated to crisp premises is fuzzyfied. Second, the crisp premises associated to a fuzzy conclusion are fuzzyfied too. Last, a consensus is designed to come to a defuzzyfied prediction when several rules apply on a given example.

4.1. Handling real-valued attributes

Constrained generalization is based on the only definitions of counter-example and discriminant attribute. These notions are unambiguous when dealing with discrete attributes. When dealing with real-valued attributes, two precision thresholds denoted η and ϵ are required from the expert. Given these thresholds, let us give some definitions :

- Let $\mathcal{F}$ denote the conclusion, i.e. the attribute to estimate, and let Ex and Ex' be two examples. If $\mathcal{F}$ is qualitative, Ex and Ex' are counter-examples to each other iff $\mathcal{F}(Ex) \neq \mathcal{F}(Ex')$, as before. If $\mathcal{F}$ is numerical, Ex and Ex' are counter-examples to each other iff

  $$|\mathcal{F}(Ex) - \mathcal{F}(Ex')| > \eta.(Max(\mathcal{F}) - Min(\mathcal{F}))$$

  where interval [Min($\mathcal{F}$), Max($\mathcal{F}$)] stands for the domain of $\mathcal{F}$ [5].

- Let x denote a descriptor, i.e. an attribute other than the conclusion. If x is qualitative, the selector involving x and discriminating Ex and Ex' is defined as in 3.2. Otherwise, x discriminates Ex and Ex' iff

  $$|x(Ex) - x(Ex')| > \epsilon.(Max(x) - Min(x))$$

  where interval [Min(x), Max(x)] stands for the domain of x. In this case, the crisp selector involving x, generalizing Ex and discriminating Ex' is given by $[x = V]$ where V denotes the largest interval including $x(Ex)$ and not overlapping interval $[x(Ex')-\frac{\epsilon}{2}, x(Ex')+\frac{\epsilon}{2}]$. Ideally, every attribute x should be associated a threshold ϵ. However,

[5] If the domain is not known in advance, its bounds are computed from the data set.

the number of parameters to be tuned by the expert is not to be too much increased. Given example Ex, another possibility is adjusting threshold ϵ so that the wanted percentage of examples is discriminated from Ex by x, i.e. falls outside interval $[x(Ex) - \frac{\epsilon}{2}, x(Ex) + \frac{\epsilon}{2}]$.

Same problem as before is considered for the purpose of illustration, except that the conclusion to learn is now the length of the quest (in days):

	Name	*Color*	*Nb Questions*	*Length*
Ex_1	*Arthur*	*Blue*	*3*	*3652*
Ex_2	*Ganelon*	*Grey*	*7*	*2756*
Ex_3	*Iago*	*Yellow*	*4*	*3999*
Ex_4	*Bryan*	*Green*	*10*	*1001*
Ex_5	*Triboulet*	*Cream*	*5*	*-1*

Table 3 : Examples in a regression problem.

Precision threshold η is set to 10^{-1}. Domain of *quest length*, according to observations, is [-1, 3999] ; so *Ganelon, Bryan* and *Triboulet* are counter-examples to *Arthur*, *Iago* is not.

Domain of *Nb of Questions* is known to be [0, 10] ; with a precision threshold $\epsilon = 10^{-1}$, this attribute discriminates *Arthur* from *Ganelon, Bryan* and *Triboulet.* The corresponding constraints become accordingly[6]:

	Name	Color	Nb Questions
$Constraint(Ex_1, Ex_2)$	$Knight$	$High$	$[0, 6.5[$
$Constraint(Ex_1, Ex_4)$	$Historical$	$Blue$	$[0, 9.5[$
$Constraint(Ex_1, Ex_5)$	$Tragic$	$High$	$[0, 4.5[$

Table 4 : Constraints derived from negative examples.

What is changed compared to the similar array of crisp constraints (in 3.2) is that the definition of counter-examples and selectors involving numerical attributes is more tunable and more capable (via thresholds) to deal with fuzzy and/or numerical informations. Let us think of replacing the crisp attribute *Nb of Questions* by attribute *Average Difficulty of Questions* !

Given these new constraints, the algorithm described in 3.4 still applies and derives discriminant (crisp) premises. The point is now to associate a conclusion to such discriminant premises.

[6]Let us assume one may ask half a question !

4.2. Designing a fuzzy conclusion

Let T denote some conjunctive (crisp) premises generalizing example Ex, i.e. covering no counter-examples to Ex according to set η. When dealing with a qualitative conclusion, the conclusion associated to premises T merely is the conclusion of example Ex, i.e. the conclusion of (a majority of) examples covered by T. In what follows, the notation $Ex \prec T$ stands for Ex is covered by T.

So the design of a conclusion is worth considering only when dealing with a numerical conclusion.
Four strategies are possible.

(i) Associate to premises T the conclusion of example Ex itself :

$$T \Rightarrow [\mathcal{F} = \mathcal{F}(Ex)]$$

However, this is a rough solution as T can cover examples with conclusion somewhat different from that of Ex.

(ii) Associate to T the averaged conclusion of examples covered by T :

$$Let\ \mathcal{F}_T = Mean_{Ex \prec T}\{\mathcal{F}(Ex)\}\ ;\ then\ T \Rightarrow [\mathcal{F} = \mathcal{F}_T]$$

Though better, this solution does not account for the dispersion of the solutions around the average value $\mathcal{F}_T$.

(iii) Associate to premises T the conditional probability distribution of $\mathcal{F}$ knowing T. This would be the best thing to do ; however, machine learning does never assume nor does it require the data to be statistically representative of the problem domain.

(iv) Use membership functions over the domain of $\mathcal{F}$, as we cannot afford a probabilistic modelization of uncertainty.

The membership function on $\mathcal{F}$ domain associated to premises T, denoted $\mathcal{D}_{\mathcal{F},T}$, is designed as follows. First of all, this function is approximated by a triangle ; the choice of a triangle is done because aggregating two triangles is far easier than aggregating two bell-shaped curves[7].

The top of this triangle corresponds to the mean value $\mathcal{F}_T$; its height is 1. Then the upper mean and lower mean of $\mathcal{F}$ with respect to T, respectively noted $U_{\mathcal{F},T}$ and $L_{\mathcal{F},T}$ are defined by :
$U_{\mathcal{F},T} = Mean\{\mathcal{F}(Ex), Ex \prec T, \mathcal{F}(Ex) > \mathcal{F}_T\}$

[7]Another possibility would be to learn rules bounding $\mathcal{F}$: for instance, given a threshold v, set counter-examples to examples Ex with $\mathcal{F}(Ex) < v$; this would lead to learn premises with conclusion of the form $[\mathcal{F} > v]$. In that case, monotonic membership functions should be used [4].

$L_{\mathcal{F},T} = Mean\{\mathcal{F}(Ex), Ex \prec T, \mathcal{F}(Ex) < \mathcal{F}_T\}$[8]
The middle height of the triangle is reached for values $U_{\mathcal{F},T}$ and $L_{\mathcal{F},T}$:

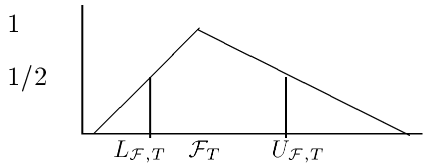


Figure 3 : *Concluding to a membership function*

4.3. Fuzzyfying premises

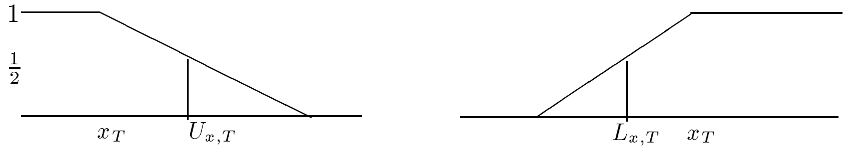
Let us consider a numerical selector involved in premises T : $[x = V]$, where V is an interval of the domain of x. This selector rejects examples with values falling outside V for attribute x ; V being chosen because a significant number of counter-examples fall outside V. However, all regions in V are not ensured to reject such counter-examples with same certainty : typically, the center of V is "safer" than the border.

This fact is modelized by replacing selector $[x = V]$ by a membership function over the domain of x denoted $\mathcal{D}_{x,T}$. Two cases are distinguished whether V contains a boundary of the domain of x.

If V does not contain any boundary of the domain of x, selector $[x = V]$ is replaced by a triangular membership function designed as above :
Let x_T denote the average value of x taken on all examples satisfying premises T. The upper mean and lower mean of x with respect to T, respectively noted $U_{x,T}$ and $L_{x,T}$ are defined by :
$U_{x,T} = Mean\{x(Ex), Ex \prec T, x(Ex) > x_T\}$
$L_{x,T} = Mean\{x(Ex), Ex \prec T, x(Ex) < x_T\}$[9]
The top of the triangle $\mathcal{D}_{x,T}$ corresponds to the mean value x_T ; its height is 1. The middle height of the triangle is reached for values $U_{x,T}$ and $L_{x,T}$.

In the other case, selector $[x = V]$ is replaced by a monotonic membership function, taking value 1 from the average value x_T to the corresponding boundary of the domain of x :

[8]Value $U_{\mathcal{F},T}$ (respectively $L_{\mathcal{F},T}$) is not defined if set $\{Ex, Ex \prec T, \mathcal{F}(Ex) > \mathcal{F}_T\}$ (resp. $\{Ex, Ex \prec T, \mathcal{F}(Ex) < \mathcal{F}_T\}$) is empty. In this case, $U_{\mathcal{F},T}$ ($L_{\mathcal{F},T}$) is set to $\mathcal{F}_T + \eta^2$ (resp. $\mathcal{F}_T - \eta^2$).
[9]Value $U_{x,T}$ (respectively $L_{x,T}$) is not defined if set $\{Ex, Ex \prec T, x(Ex) > x_T\}$ (resp. $\{Ex, Ex \prec T, x(Ex) < x_T\}$) is empty. In this case, $U_{x,T}$ ($L_{x,T}$) is set to $x_T + \epsilon^2$ (resp. $x_T - \epsilon^2$).



(a) V contains the lower bound of x. (b) V contains the upper bound of x.

Figure 4 : Membership function in premises

Fuzzy premises $\mathcal{T}$ are then defined by the conjunction of distributions $\mathcal{D}_{x,T}$, for x attribute involved in premises T. Function $\mathcal{D}_{x,T}$ associated to a qualitative attribute x just is the characteristic function of selector $[x = V]$.

4.4. Firing fuzzy rules and reaching a consensus

Now, given the fuzzy rules learned by the extended constrained induction, the final goal is to use them on further examples.

Let $\mathcal{T}$ denote the fuzzy premises defined as the conjunction of membership functions $\mathcal{D}_{x,T}$ for attribute x involved in crisp discriminant premises T. Then, example Ex matches premises $\mathcal{T}$ to some extent, computed as the minimum value of $\mathcal{D}_{x,T}(x(Ex))$, for x involved in T.

However, example Ex may fire several premises to some non null extent. The trouble arises when these premises are associated to different conclusions. Two kinds of consensus are performed, whether learning concerns a qualitative (crisp) conclusion, or a numerical (fuzzy) one.

4.4.1. Firing rules with crisp conclusions

The usual strategy in case of conflicting conclusions [15, 14], is to perform a majority vote among the conclusions of the fired rules : each fired rule votes for its conclusion, the resulting conclusion being the majoritary one. However, rules are not equally relevant : some rules are more important than others. Rules are so weighted to account for their relevance, and a weighted majority vote is then performed (each rule votes for its conclusion with its weight). The weight of a rule is usually set to the number of training examples covered by this rule. The adaptation of this strategy to the case of fuzzy rules is straightforward : the weight of a rule just is set to the sum of the degrees of match between the training examples and the rule.

4.4.2. Firing rules with fuzzy conclusions

The above procedure does not apply when an example fires rules with numerical conclusions : the consensus is expected to be more an interpolation than a vote. Several interpretations of an AND between fuzzy terms (here, the fuzzy conclusions of the fired rules) are encountered in the literature.

- The conjunctive attitude, which applies when aggregating *consistent* fuzzy terms [9], leads to an eventual membership function which is the minimum of all membership functions. This solution is not adapted when dealing with possibly inconsistent rules : the minimum of all membership functions often is identically zero.
- On the opposite, when knowledge is "not completely reliable, a cautious attitude may be to conclude that the disjunction of the various conclusions is true" [16]. In that case, the operator *maximum* is used. But if redundant rules are considered, the support of this maximum membership functions may be very large.
- Trade-off attitude consists of taking a weighted sum of functions $\mathcal{D}_{\mathcal{F},T}$. This last solution is retained[10]; the weight of a rule is set to the sum of the degrees of match between the training examples and the rule.

4.4.3. Defuzzyfication

When a decision is to be made (e.g. for control applications), the last step is to convert the membership function resulting from the consensus into a real value. A state of the art of the defuzzyfication methods can be found in [4]. The main possibilities are (a) taking the mean of maxima (M.O.M) of the resulting membership functions ; and (b) taking the center of average (C.O.A) such that it equally divides the area under the membership function. The latter solution is chosen ; M.O.M. method experimentally shows too rough.

5. State of the art

The proposed method is to be situated with respect to other existing methods of building membership functions and fuzzy rules. Methods are distinguished, whether they rely on experts solely, on neural networks, or on genetic algorithms.

5.1. Knowledge Acquisition

The traditional approach by knowledge acquisition from the expert, was the only tractable one in the pioneers's times : fortunately, it was successful enough to support the promises of fuzzy logic. Notice that relying on the expert often is another name for the errors and trials method. For instance, the right level of granularity of a numerical attribute (*Large, Medium, Small* or *Very Large, Large, Medium, Small* and *Very Small*) has

[10]To still obtain a membership function, the weighted sum of functions $\mathcal{D}_{\mathcal{F},T}$ is divided by the maximum rule weight.

to be experimentally decided [4] ; as far as we know, there is no a priori criterion as to the right zooming of fuzzy processing.

To put it short, the expert touch is always required (at least, obviously, to have a well-posed problem) and may provide a number of valuable insights. However, letting him/her in charge of the overall design of a fuzzy system is both a burden for the expert and frustrating from the engineer point of view, as no insurance can be given as to the optimality of the final process.

As a transition toward using neural networks, a knowledge acquisition model is the Fuzzy Cognitive Maps of Kosko [21–23]. This model only requires the expert to state the existence of relationships between the different input and output of the problems. The links so created provide the architecture of an artificial neural network, which can be weighted and optimized by a number of techniques.

5.2. Artificial Neural Networks

More generally, a number of methods have been proposed to use neural networks to learn a fuzzy knowledge base ; several international conferences witness the cross-fertilization of fuzzy logic and neural networks [1–3].

The point is that neural networks and fuzzy logic share three major characteristics : they are able to model highly non-linear systems, to provide the user with smooth output, and, to some extent, to face unforeseen situations. The last point is to be contrasted with the possibilities of crisp logic and expert systems, where unforeseen situations are not handled at all. This is possible because of the interpolation abilities of both fuzzy logic and neural networks. Moreover, neural networks can be used both to learn the knowledge base of a fuzzy system by using some backward propagation, and to perform fuzzy inferences by feed-forward propagation of the input.

An analysis of various neural architectures used to model fuzzy processing with respect to criteria such that trainability, theoretical predictability, understandability, accuracy, and compatibility with composition rules, can be found in [19].

5.3. Genetic Algorithms

An other attempt for the automatic acquisition of membership functions relies on both inductive learning and genetic algorithms [11] : system ML-Smart+ of Botta et coll, is the last version of well-known concept learner ML-Smart [7, 8] that builds decision trees.

Membership functions are used in order to improve the handling of real-valued descriptors - one weakness of decision trees. Membership functions may be provided by the user or learned quasi from scratch by the system.

The steps of the system are the followings :

- The user provides the granularity of a descriptor (the number of linguistic qualifier to be used) and the approximative zones where the change is to be found.
- A decision tree is built up on the basis on the quantity of information and on other statistical criteria. The initial membership functions are evaluated on the basis of these criteria and possibly retained in the decision tree.
- A refinement step consists of using genetic algorithms to improve the thresholds involved in the retained membership functions.

An interesting point is that a given membership function can appear at several nodes of the decision tree ; and the parameters related to the occurences of this membership function are independantly optimized. In other words, ML-Smart+ considers and optimizes the occurences of a given membership function as if they were different membership functions. This confirms the conjecture that a membership function should be conceived as depending on several descriptors. However, the increase in the number of membership functions to consider should prevent from iterating the process.

5.4. Discussion

A preliminary remark is that the different approaches have been designed to handle different problems. Some proeminent criteria to judge the usability of an approach on a given problem are the followings :

- Is there some available explicit knowledge ? Neural networks can sometimes encode available knowledge, through the network architecture in general ; apart from this, they are not good at using knowledge. Similarly, genetic algorithms can use available knowledge through the design of problem-dependent operators and representation. In opposition, available knowledge is explicitly handled in inductive learning. Furthermore, the knowledge encapsulated by inductive learning is directly explicit.
- Are the data statistically representative ? Neural networks and decision trees traditionnally deal with statistically representative examples. Similarly, if membership functions are to be optimized by GA, a statistically representative set of data must be available. In the meanwhile, bottom-up induction may deal with sparse data, provided that any phenomenon in the domain space is represented[11].

[11] The difference can be illustrated as follows ; in a failure detection problem, a statistically representative set of examples should include, say, 90% examples in the normal-

– Is the concept to learn disjunctive ? Learning disjunctive concepts is difficult for decision trees, as emphasized in 2. It is difficult too for neural networks ; as a matter of fact, imagine a disjunction of rules encoded within a multi-layers neural network. An error in one of these rules will cause all the rules to be corrected by the effects of back-propagation. In the meanwhile, learning disjunctive concepts presents no difficulty for bottom-up learning.
– What is the type (numerical, qualitative, both) of data ? Neural networks are good at using numerical information ; handling qualitative information may result in significant increase of the size of the problem. Top down learning is good at using qualitative information, but sophisticated heuristiques are necessary to handle numerical information. Our claim is that bottom-up learning, embedded into the fuzzy sets formalism, accurately handles both numerical and qualitative informations.

6. Conclusion

This paper is an attempt to bridge the gap between inductive learning and fuzzy logic. Fuzzy logic brings a significant help in offering a formalism to deal with real-valued descriptors in a both robust and smooth way. On the other hand, induction provides the way of instanciating these templates according to the interdependance of the descriptors in the problem at hand.

The hybrid fuzzy induction proposed achieves the building of a set of fuzzy rules, capable to perform discrimination as well as regression. This system thus offers many opportunities of technical applications. It has been validated on real-world problems, concerning the prediction of the elastic limit of new materials from a data base about trials on composite materials [31].

Further researches are concerned with replacing the estimating frame (the crisp associated induction delivers rules such as : *If so and so,* $\mathcal{F}$ *belongs to this interval*) by a bounding frame (*If so and so,* $\mathcal{F}$ *is greater than* ...). The motivations are to improve the behavior of the fuzzy estimator around the boundaries of the domain of $\mathcal{F}$, and to use this estimator for optimization (where to search the optima of $\mathcal{F}$).

Another perspective is to use learning as to the aggregation process, when several fuzzy rules are fired by an example. Last, we would like to study a fuzzy constraint-based estimation : the idea is to directly use the

state class ; a representative set of examples may include as many examples of the normal state class, as in any class of failures.

constraints to perform prediction, escaping the effective expression of the fuzzy rules and dealing only with the disjunction of membership functions.

Acknowledgement

We thank Bernadette Bouchon-Meunier, from Laforia Paris-VI, for several discussions and comments about this work.

References

[1] *International Conference on Fuzzy Logic and Neural Networks*, Iizuka, Japan, 1988.

[2] *International Conference on Fuzzy Logic and Neural Networks*, Iizuka, Japan, 1990.

[3] *International Conference on Fuzzy Logic and Neural Networks*, Iizuka, Japan, 1992.

[4] H.R. Berenji, *Fuzzy Logic Controllers*, in An Introduction to Fuzzy Logic Applications in Intelligent Systems, R Yager and L. Zadeh Eds, Kluwer Academic Publishers, 1992, pp 69-96.

[5] H.R. Berenji, *An Architecture for Designing Fuzzy Controllers Using Neural Networks*, 2^{nd} Joint Technology Workshop on Neural Networks and Fuzzy Logic, Houston Texas, April 1990.

[6] F. Bergadano, S. Matwin, R. Michalski, J. Zhang, *Learning Two-Tiered Descriptions of Flexible Concepts: The Poseidon System*, Machine Learning Journal, Kluwer Academic Publishers, Jan 1992, pp 5-43.

[7] F. Bergadano, A Giordana, L. Saitta, *Automated Concept Acquisition in Noisy Environments*, IEEE Trans on Pattern Analysis and Machine Intelligence, PAMI-10, pp 555-578, 1988.

[8] F. Bergadano, A Giordana, *A Knowledge Intensive Approach to Concept Induction*, ICML 1988, pp 305-317.

[9] P.P. Bonissone, *Summarizing and Propagating Uncertain Information with Triangulars Norms*, Inter. J. of Approximate Reasoning, Vol 1, 1987, pp 71-101.

[10] M. Botta, A Giordana, L. Saitta, *Learning in Uncertain Environments*, in An Introduction to Fuzzy Logic Applications in Intelligent Systems, R Yager and L. Zadeh Eds, Kluwer Academic Publishers, 1992, pp 281-296.

[11] M. Botta, A Giordana, *SMART+ : A Multi-Strategy Learning Tool*, IJCAI-93, pp 937-943.

[12] Bouchon Meunier B, Yager R, *Entropy of similarity relations in questionnaires and decisions trees*, Proc. FUZZ IEE, San Francisco 1993.

[13] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, *Classification and Regression by trees*, Belmont California, Wadsworth 1984.

[14] B. Cestnik, I. Bratko, I. Kononenko, *ASSISTANT 86: A knowledge elicitation tool for sophisticated users*, in Progress in machine learning, Proc. EWSL 1987, I. Bratko N. Lavrac Eds, Sigma Press

[15] P. Clark T. Niblett, *Induction in noisy domains*, in Progress in Machine Learning, Proc. EWSL 1987, I. Bratko N. Lavrac Eds, Sigma Press.

[16] D. Dubois, H. Prade, *Coping with uncertain knowledge : in defense of possibility and evidence theory*, Rapport L.S.I. 269, nov 1987.

[17] Fayyad U.M., Irani B.K., *Multi-Interval Discretization of Continuous Valued Attributes for Classification Learning*, IJCAI-93, pp 1022-1027.

[18] Goldberg D.E., Genetic algorithms in search, optimization and machine learning, Addison Wesley 1989.

[19] Y. Hayashi, J.M. Keller, *Comparison of Network-Based Inference Mechanisms for Fuzzy Logic*, Proc. of 2^{nd} International Symposium on Uncertainty Modeling and Analysis, B. Ayyub Ed, IEEE Computer Society Press, Maryland, 1993, pp 334-338.

[20] A. Jovanovic, *Use of inquiry in acquisition of engineering knowledge*, in Expert Systems in Structural Safety Assessment, A. Jovanovic & al. Eds, Springer-Verlag, 1989.

[21] B. Kosko, Neural Networks and Fuzzy Systems, Prentice Hall 1992.

[22] B. Kosko, *Fuzzy Associative Memories*, in Fuzzy Expert Systems, A. Kandel Ed, Addison Wesley, 1987.

[23] B. Kosko, *Fuzzy Cognitive Maps*, International J. of Man-Machine Studies, 24, pp 65-75, 1986.

[24] E.H. Mamdani, S. Assilian, *An experiment in linguistic synthesis with a fuzzy logic controller*, Int. Journal of Man-Machine Studies, 1975, 7-1, pp 1-13.

[25] Michalski R.S. *A theory and methodology for inductive learning* in Machine Learning: An Artificial Intelligence Approach, I, R.S. Michalski, J.G. Carbonnell, T.M. Mitchell Eds, Springer Verlag, (1983), p 83-134.

[26] Michalski R.S. I. Mozetic, J. Hong, N. Lavrac *The AQ15 inductive learning system: an overview and experiment* Proceedings of IMAL, 1986 Orsay.

[27] T.M. Mitchell, *Generalization as Search*, Artificial Intelligence Vol 18, pp 203-226, 1982.

[28] R. Quinlan, *Induction of decision trees* in Machine Learning, 1, 1986, pp 81-106.

[29] R. Quinlan, *Learning logical definitions from relations*, in Machine Learning, 5, 1990, pp 239-266.

[30] R. Quinlan, R.M. Cameron-Jones, *FOIL ; A Midterm Report*, ECML 93, P.B. Bradzil Ed, Springer-Verlag, 1993, pp 3-20.

[31] M. Sebag, G. Regnier, *Apprentissage à partir de bases de données d'essai.* European Conference on New Advances in Computational Mechanics, Gien 1993.

[32] M. Sebag, *Using Constraints to Building Version Spaces*, ECML-1994, L. de Raedt F. Bergadano Eds, Springer Verlag.

[33] M. Sebag M. Schoenauer *Incremental Learning of Rules and Meta-Rules* 7^{th} International Conference on Machine Learning, R. Porter B. Mooney Eds, Morgan Kaufmann, Austin Texas 1990.

[34] C.E. Shannon, W. Weaver, The mathematical theory of coomunication,

University of Illinois Press, Urbana, 1949.

[35] B. Smith, P. Rosenbloom, *Incremental non-backtracking focussing : A polynomially- bounded generalization algorithm for version space*, Proc. National Conference on Artificial Intelligence, 1990.

[36] Takagi Hayashi, *Artificial Neural Network - driven Fuzzy Reasoning*, Int. J. of Approximate Reasoning, 1992.

[37] *ID5 : An incremental ID3*, in Proceedings of 5^{th} ICML, Morgan Kaufmann, Las Altos 1988.

[38] Van de Merckt T, *Decision Trees in Numerical Attribute Spaces*, IJCAI-93, pp 1016-1021.

[39] L.A. Zadeh, *Fuzzy sets as a basis for a theory of possibility* Fuzzy Sets & Systems, Vol 1, 1978, pp 3-28.

[40] R. Zhao, R. Govind, *Defuzzyfication of Fuzzy Intervals*, B. M. Ayyub Ed., Proc. of 1^{st} International Symposium on Uncertainty Modeling and Analysis, IEEE Computer Society Press, Maryland, 1990.

[41] H.J. Zimmerman, <u>Fuzzy set theory and its applications</u>, 3^{rd} edition, Kluwer Academic Publishers Group, 1988.

[42] Weiss S.M., Indurkhya N, *Rule-Based Regression*, IJCAI-93, pp 1072-1077.

[43] P.H. Winston, *Learning Structural Descriptions from Examples* in <u>The Psychology of Computer Vision</u>, P.H. Winston Ed, Mc Graw Hill, New York, 1975, pp 157-209.