Soft Comput (2007) 11:459–466
DOI 10.1007/s00500-006-0109-z

FOCUS

Improving expressivity of inductive logic programming by learning different kinds of fuzzy rules

Mathieu Serrurier · Henri Prade

Published online: 20 June 2006

Abstract Introducing fuzzy predicates in inductive logic programming may serve two different purposes: allowing for more adaptability when learning classical rules or getting more expressivity by learning fuzzy rules. This latter concern is the topic of this paper. Indeed, introducing fuzzy predicates in the antecedent and in the consequent of rules may convey different non-classical meanings. The paper focuses on the learning of gradual and certainty rules, which have an increased expressive power and have no simple crisp counterpart. The benefit and the application domain of each kind of rules are discussed. Appropriate confidence degrees for each type of rules are introduced. These confidence degrees play a major role in the adaptation of the classical FOIL inductive logic programming algorithm to the induction of fuzzy rules for guiding the learning process. The method is illustrated on a benchmark example and a case-study database.

Keywords Inductive logic programming · Fuzzy rules

1 Introduction

Fuzzy set-based methods have been widely introduced in all areas of machine learning. First, neuro-fuzzy learning techniques have been developed for tuning fuzzy membership functions in fuzzy rules; see [13,15] for a survey. The fuzzy rules, that are produced in that way are mainly used for approximating functions in automatic control problems.

Besides, there has been an increasing interest for using fuzzy sets for the learning of propositional rules or decision trees. The main motivation for using fuzzy sets is the need for coping with imprecise or incomplete data. Different approaches [1,10,12] have been proposed for learning fuzzy decision trees by adapting the ID3 [19] or C4.5 [21] algorithms. These adaptations are made by incorporating fuzzy sets in the computation of the entropy measure that controls the algorithm. In these algorithms, fuzzy sets are provided by an expert, but some methods have been developed for performing an automatic discretization into fuzzy sets that are easily interpretable [4]. In order to induce fuzzy rules, a fuzzy adaption of the CN2 [2] algorithm has been proposed [7]. In this context, we can find some discussions on the expression of support and confidence degrees for fuzzy rules in [3,9]. However, all these methods induce propositional "if …then" rules.

Inductive logic programming (ILP) [16] provides a general framework for learning classical first-order logic rules, for which reasonably efficient algorithms have been developed (Progol [14], FOIL [20],…). There are two other approaches for using fuzzy sets in ILP. These methods are based on the FOIL algorithm too. The first one [23] extends FOIL by expressing a fuzzy version of the gain function (no longer based on a confidence degree) that guides the learning process, in order to handle fuzzy sets. Rules found are then used as classical crisp rules. A second approach [5], named FS-FOIL, is closer to the one presented in this paper, since it describes a fuzzy version of the confidence degree in the gain function to be used in FOIL. But, the definition used only applies to unary predicates, and moreover, the rules found keep a classical "if …then" reading.

A first motivation for introducing fuzzy sets in ILP is to increase the expressivity of the learned rules. Indeed fuzzy sets are well-known for providing a faithful representation for concepts such as "young", "expensive", "tall" …, whose meaning may be a matter of degree. There exist different

M. Serrurier (✉) · H. Prade
IRIT, UPS, 118 route de Narbonne,
Toulouse cedex 9, 31062, France
e-mail: serrurier@irit.fr

H. Prade
e-mail: prade@irit.fr

kinds of fuzzy rules [6], with different intended meanings, which have no simple crisp counterpart. In the following we consider the learning of two types of fuzzy rules based on multiple-valued implication connectives, which are named gradual and certainty rules respectively. Gradual rules express a positive synergy between fuzzy predicates under the form "the more X is A, the more X is B", while certainty rules rather allow for uncertain consequent, under the form " the more X is A, the more certain X is B". This article develops some of the results in [17,18] by providing a more detailed analysis of the properties of the two types of first order rules involving fuzzy predicates, and a study of their integration in the ILP framework. Thus, the use of fuzzy concepts in learning allows us to increase the expressive power for describing the data and to learn new types of rules. These new rules need a proper definition of their confidence degree. In order to learn each type of rules, we use the FOIL algorithm with the appropriate confidence degree. The methods can induce one type of rule at time. Thus, the user of the algorithm must choose the type of rule that he wants to learn.

The paper is organized as follows. First we recall some definitions about ILP and fuzzy databases in Sect. 2. Next, we present gradual and certainty rules, and their integration in the ILP machinery, in Sects. 3 and 4 respectively. Lastly, we illustrate the approach on a case-study database and a benchmark example.

2 Background

2.1 ILP

ILP is stated in the general context of first order logic. The ILP problem is described by means of two logic programs:

- $\mathcal{B}$: the background theory which is a set of ground facts and rules.
- $\mathcal{E}$: the training set which is a set of ground facts, called examples, pertaining to the predicate to be learnt. The training set describes the concept we want to learn. It is often split in $\mathcal{E}^+$ and $\mathcal{E}^-$, which correspond respectively to positive and negative examples. If only $\mathcal{E}^+$ is given, $\mathcal{E}^-$ can be deduced by using the Closed World Assumption (anything that is not stated as true is assumed to be false).

The task of *induction* is to find, given $\mathcal{B}$ and $\mathcal{E}$, a set of formulas $\mathcal{H}$ such that :

$$\forall e \in \mathcal{E}^+, \quad \mathcal{B} \cup \mathcal{H} \models e \tag{1}$$

and

$$\forall e \in \mathcal{E}^-, \quad \mathcal{B} \cup \mathcal{H} \not\models e. \tag{2}$$

Of course, one may add two natural restrictions:

- $\mathcal{B} \not\models \mathcal{E}^+$ since, in such a case, $\mathcal{H}$ would not be necessary to explain $\mathcal{E}^+$.
- $\mathcal{B} \cup \mathcal{H} \not\models \bot$: this means that $B \cup \mathcal{H}$ is a consistent theory.

In the setting of relational databases, inductive logic programming is often restricted to Horn clauses and function-free formulas. Moreover, the set $\mathcal{E}^+$ itself satisfies the previous requirement, but it is generally not considered as an acceptable solution since it has no predictive ability. Usually, rule extraction fits with the idea of providing a compression of the information contained in $\mathcal{E}$.

In the following, first order rules are denoted by "$A(\vec{x}) \rightarrow C(\vec{x})$" where $\vec{x}$ is the vector of the n variables that appear in the clause. $A(\vec{x}) = A_0(\vec{x}) \wedge \cdots \wedge A_q(\vec{x})$ represents the antecedent of the rule. Since we do not allow for recursivity, the predicates that appear in $A(\vec{x})$ pertain only to predicates that appear in $\mathcal{B}$ and not to the predicate that appears in $\mathcal{E}$. $C(\vec{x})$ is the consequent of the rule, the predicate C pertains to the concept to be learnt. Given an attribute domain $\mathcal{D}$ and a vector $\vec{t} \in \mathcal{D}^n$ of n values of the domain, we denote, the ground substitution of the variable $\vec{x}$ with $\vec{t}$ by $C(\vec{t}) = \sigma[\vec{x}/\vec{t}]C(\vec{x})$. Then $C(\vec{t})$ is true or false in a given interpretation. An example is said to be covered by a rule if it can be deduced from the rule and $\mathcal{B}$. There are two main types of algorithms, *top down* and *bottom up* algorithms. *Top down* ones start from the most general clause and specialize it step by step. *Bottom up* procedures start from a fact and generalize it.

The FOIL [20] algorithm is one of the most efficient algorithm in ILP. One of the main difficulties in ILP is to manage the size of the hypothesis space. For instance, as soon as we allow function symbols in the target clauses, the hypothesis space becomes infinite, even if we restrict ourselves to the induction of one clause with a maximum of l literals. In case of restriction to Horn clauses without function symbol, the hypothesis space still grows up exponentially with l. As a consequence, methods performing an exhaustive search are intractable. Thus, ILP algorithm must use heuristics for guiding the learning process. FOIL is a *top down* algorithm that performs a greedy search in order to maximize a gain function. For inducing a rule, FOIL starts with the most general clause ($\top \rightarrow C(\vec{x})$) and specializes it step by step by adding literals in the antecedent. The rule is accepted when its confidence degree overcomes a fixed threshold.. The rules are induced until all examples are covered or no more rules are found that overcome the threshold. When a rule is induced, the positive examples covered by the rule are removed from E. This is the *coverage approach*. Rules with consequent C, the target predicate, are induced as described in Algorithm 1 where $\vec{x}$ is the vector of free variables of the rule and θ is the confidence degree threshold.

Algorithm 1 Learn(C,θ)

choose $A(\vec{x}) = \top$
while cf($A(\vec{x}) \rightarrow C(\vec{x})$) $< \theta$ **do**
  $A_{\text{tmp}}(\vec{x}) = A(\vec{x})$
  maxgain = 0
  for allpossible literals $L(\vec{x})$**do**
    gain = gain $(cf(A(\vec{x}) \wedge L(\vec{x}) \rightarrow C(\vec{x})),$
      $cf(A(\vec{x}) \rightarrow C(\vec{x})))$
    ifgain $\geq$ maxgain**then**
      maxgain=gain
      $A_{\text{tmp}}(\vec{x}) = A(\vec{x}) \wedge L(\vec{x})$
    end if
  end for
  $A(\vec{x}) = A_{\text{tmp}}(\vec{x})$
end while
return $A(\vec{x}) \rightarrow C(\vec{x})$

The gain function is computed by the formula :

$$\text{gain}(cf(r_1), cf(r_2)) = p * (\log_2(cf(r_1)) - \log_2(cf(r_2))),$$

where p is the number of distinct examples covered by the rule r_1. Thus, the gain is positive if and only if the new rule is more informative in the sense of Shannon's information theory (i.e. if and only if the confidence degree increases). If there are some literals to add which increase the confidence degree, the gain tends to favor the literals that offer the best compromise between the confidence degree and the number of examples covered. There exist some extensions of this algorithm. For instance, FS-FOIL [5] uses a beam search strategy and SAILP performs a stochastic search based on simulated annealing.

Given a Horn clause $A(\vec{x}) \rightarrow C(\vec{x})$, the confidence $cf(A(\vec{x}) \rightarrow C(\vec{x})) = P(A(\vec{x}) \wedge C(\vec{x}))/P(A(\vec{x}))$. Confidence degrees are computed in the spirit of domain probabilities [8]. ILP data are supposed to describe one interpretation under closed world assumption. We call $\mathcal{I}_{\text{ILP}}$ this interpretation. So, given a fact f:

$$\mathcal{I}_{\text{ILP}} \models f \quad \text{iff} \quad \mathcal{B} \cup \mathcal{E} \models f.$$

The domain $\mathcal{D}$ is the Herbrand domain described by $\mathcal{B}$ and $\mathcal{E}$. We take P as a uniform probability on $\mathcal{D}$. So we deduce that the confidence in a clause $A(\vec{x}) \rightarrow C(\vec{x})$ is:

$$cf(A(\vec{x}) \rightarrow C(\vec{x}))_{\mathcal{I}_{ILP}} = \frac{|\{\vec{t} \in \mathcal{D}^n \mid \mathcal{I}_{\text{ILP}} \models (A(\vec{t}) \wedge C(\vec{t})) \text{ and } C(\vec{t}) \in \mathcal{E}^+\}|}{|\{\vec{t} \in \mathcal{D}^n \mid \mathcal{I}_{\text{ILP}} \models (A(\vec{t})) \text{ and } C(\vec{t}) \in \mathcal{E}\}|}, \quad (3)$$

where $|\ |$ denotes cardinality. Testing all possible $\vec{t} \in \mathcal{D}^n$ is not tractable in practice. However, we can equivalently restrict the computation to the substitutions that map variables to constants in their specific domains. In fact, this computation is equivalent to a database query and thus, we can also use some optimization strategy such as indexing or query ordering. This makes the computation tractable although it remains costly.

Another possible definition of a confidence degree might be taken here as the proportion of the number of distinct positive examples covered by the rule w.r.t. the number of total distinct examples (positive and negative) covered by the rule. This confidence degree would represent the probability that a fact deduced from the rule is true. But this definition would not take into account the number of situations covered in the antecedent of the rule, which is not always the total number of examples covered since we are in a first-order setting.

Besides, in order to evaluate the prediction ability of a global hypothesis, i.e. a set of rules $\mathcal{H} = \{h_1, \ldots, h_r\}$, we define the accuracy of an hypothesis in the following way:

$$\text{acc}(\mathcal{H}) = \frac{tp + tn}{tp + tn + fp + fn}, \quad (4)$$

where:

- $tp = |\{e \in \mathcal{E}^+ | \mathcal{B} \cup \mathcal{H} \models e\}|$ the set of true positive examples, i.e. the set of examples that are covered by the hypothesis;
- $tn = |\{e \in \mathcal{E}^- | \mathcal{B} \cup \mathcal{H} \not\models e\}|$ the set of true negative examples, i.e. the set of counter-examples that are not covered by the hypothesis;
- $fp = |\{e \in \mathcal{E}^- | \mathcal{B} \cup \mathcal{H} \models e\}|$ the set of false positive examples, i.e. the set of counter-examples that are covered by the hypothesis;
- $fn = |\{e \in \mathcal{E}^+ | \mathcal{B} \cup \mathcal{H} \not\models e\}|$ the set of false negative examples, i.e. the set of examples that are not covered by the hypothesis.

If the number of examples in $\mathcal{E}^-$ is high with respect to the number of examples in $\mathcal{E}^+$ or if we are only interested in the deduction of positive examples, then recall and precision indices can be used:

$$\text{precision}(\mathcal{H}) = \frac{tp}{tp + fp}, \quad (5)$$

$$\text{recall}(\mathcal{H}) = \frac{tp}{tp + fn}. \quad (6)$$

Precision estimates how accurate is the hypotheses on positive examples and recall estimates what proportion of the total set of positive examples we are able to identify.

2.2 Fuzzy databases

We consider a first-order logic database $\mathcal{K}$ with fuzzy predicates (e.g., "heavy", "old" …) as a set of positive ground facts labeled by real numbers in $[0, 1]$. Thus, $\mathcal{K}$ is made of pairs of the form $(A(\vec{t}), \mu_A(\vec{t}))$ for $\vec{t} \in \mathcal{D}^n$, where $A(\vec{t})$ is a fact,

and $\mu_A(\vec{t})$ is the satisfaction degree associated with the fuzzy property A for $\vec{t}$. A fuzzy predicate can be viewed as a family of ordinary predicates whose characteristic functions are the level cut functions μ_{F_α} associated to the fuzzy set membership function μ_F, namely $\mu_{F_\alpha}(\vec{t}) = 1$ iff $\mu_F(\vec{t}) \geq \alpha$ and $\mu_{F_\alpha}(\vec{t}) = 0$ otherwise. We can also define strict level cut functions $\mu_{\bar{F}_\alpha}$, namely $\mu_{\bar{F}_\alpha}(\vec{t}) = 1$ iff $\mu_F(\vec{t}) > \alpha$ and $\mu_{\bar{F}_\alpha}(\vec{t}) = 0$ otherwise. μ_{F_1} is core of the fuzzy set, and $\mu_{\bar{F}_0}$ its support. Thus a rule "$A(\vec{x}) \rightarrow C(\vec{x})$" is naturally associated with the crisp rules "$A_\alpha(\vec{x}) \rightarrow C_\beta(\vec{x})$". Note that, if $C_\beta(\vec{x})$ holds then $C_\alpha(\vec{x})$ also holds for $\alpha \leq \beta$. If $\alpha > \beta$, one can move to the rule $A_{1-\alpha}(\vec{x}) \rightarrow C_\beta(\vec{x})$, or to the rule $A_\beta(\vec{x}) \rightarrow C_{1-\beta}(\vec{x})$, depending on whether $\alpha + \beta < 1$ or not. So we may only consider the crisp approximations "$A_\alpha(\vec{x}) \rightarrow C_\alpha(\vec{x})$", or "$A_\alpha(\vec{x}) \rightarrow C_{1-\alpha}(\vec{x})$".

In this article, the databases $\mathcal{B}$ and $\mathcal{E}$ are thus fuzzy databases. The resolution process used is the classical one, the membership degrees are managed separately. Fuzzy databases can be issued from an expert who gives the membership degrees or by a fuzzy discretization of a crisp database that contains numerical predicates. In this latter case, we transform quantitative information into qualitative information. For instance, the ground fact "price(x_1, y_1)" (i.e. the price of the element x_1 is y_1), can be transformed into the fuzzy expression "(price$(x_1, \mathbf{low}), \alpha)$" where *low* corresponds to a fuzzy set and α is the membership degree of y_1 to this fuzzy set. Notice that here what is referred to as fuzzy predicates, are in fact predicates that contain labels of fuzzy instantiations. Indeed, in our example, we could also think of transforming "price(x_1, y_1)" into "(low_price$(x_1), \alpha)$" which is equivalent to the previous writing. However, in the following, we use the first transformation because it is simpler when a predicate involves more than one variable to which a fuzzy property may apply.

For instance, let us consider a database that describes houses in a town. This database will be used for providing examples of gradual and certainty rules. First we have some fuzzy relational predicates expressions such as (distance $(x, y, \mathbf{close}), \alpha)$, which means that the house x is close to the house y with a membership degree α. This kind of fuzzy predicates expression can be computed from a fuzzy partition of the distance universe. Another example of fuzzy predicate is in this context (know$(x, y), \alpha')$, which means that the owner of house x knows the owner of house y at a degree α' (from 0 for "unknown" to 1 for "friends"). In the example, houses are also described by means of some nearly propositional fuzzy predicate expressions such as price(x, **expensive**) or size(x, **small**).

3 Gradual rules

One reason for using fuzzy rules is that it allows us to represent imprecise and gradual properties. Rules of this kind that handle fuzzy predicates can have meanings different from classical rules and may not have any simple crisp counterpart. We first consider gradual rules [6].

The meaning of a gradual rule "$A(\vec{x}) \rightarrow C(\vec{x})$" is "the more $\vec{t}$ is A, the more $\vec{t}$ is C", understood as the constraint $\forall \vec{t}, \mu(A(\vec{t})) \leq \mu(C(\vec{t}))$. It corresponds to the following set of crisp rules:

$$\forall \vec{x}, \forall \alpha \quad A_\alpha(\vec{x}) \rightarrow C_\alpha(\vec{x}).$$

According to the database described in the previous section, an example of gradual rule is :

$$\text{size}(x, \mathbf{large}) \rightarrow \text{price}(x, \mathbf{expensive}),$$

i.e. "the larger the house, the more expensive it is", which describes the fact that price grows with size. Thus, a gradual rule is equivalent to a set of ordinary rules with nested predicates and summarizes it into a unique fuzzy rule. Gradual rules are defined using the Gödel implication $I_{\text{Gödel}}$, or the core of that implication I_{RS}, also called Rescher–Gaines implication in the literature, namely,

$$I_{\text{RS}}(\mu_A(\vec{t}), \mu_C(\vec{t})) = \begin{cases} 1 & \text{if } \mu_A(\vec{t}) \leq \mu_C(\vec{t}) \\ 0, & \text{otherwise.} \end{cases}$$

However, Rescher–Gaines implication is very restrictive, since here covering a positive example, while violating the gradual requirement will be equivalent to covering a negative example. Thus, we prefer the Gödel's implication-based definition of a gradual rule :

$$I_{\text{G"odel}}(\mu_A(\vec{t}), \mu_C(\vec{t})) = \begin{cases} 1 & \text{if } \mu_A(\vec{t}) \leq \mu_C(\vec{t}) \\ \mu_C(\vec{t}), & \text{otherwise.} \end{cases}$$

Indeed, this definition makes a difference between covering a positive example while violating the gradual requirement, and covering a negative example. However, any residuated implication defined from a left-continuous triangular norm [11] could be used. Gödel is chosen since it is associated with minimum operation and because of its qualitative nature.

According to Eq. (3), the confidence degree is computed as follows:

$$cf_{\text{grad}}(A(\vec{x}) \rightarrow C(\vec{x})) = \frac{\sum_{\vec{t} \in \mathcal{D}^n \mid \mathcal{I}_{\text{ILP}} \models (A(\vec{t}) \wedge C(\vec{t}))} \text{ and } C(\vec{t}) \in \mathcal{E}^+ \; I_{\text{G"odel}}(\mu_A(\vec{t}), \mu_C(\vec{t}))}{|\{\vec{t} \in \mathcal{D}^n \mid \mathcal{I}_{\text{ILP}} \models A(\vec{t}) \text{ and } C(\vec{t}) \in \mathcal{E}\}|} \tag{7}$$

When the valuation of the antecedent of the rule is a conjunction of ground literals, the satisfaction degree of this conjunction is the minimum of the degrees of each literal. Similarly, fuzzy predicates involving multiple fuzzy instantiations are associated with the minimum of the membership degrees of the example to the fuzzy sets used as instantiations. Gradual

rules describe a correlated evolution of membership degrees of the antecedent with the membership degree of the consequent of the rule. This kind of rule has no simple crisp counterpart. If we use it for deduction from a new example, we can thus deduce a lower bound value for the membership degree of the example to the concept. Since we have only information about membership degrees on positive examples, the component *tn* in the definition of accuracy (4) is not relevant, so we will only focus on precision and recall computed as follows:

$$\text{precision}_{\text{grad}}(\mathcal{H}) = \frac{tp_{\text{grad}}}{tp_{\text{grad}} + fp_{\text{grad}}}, \tag{8}$$

$$\text{recall}_{\text{grad}}(\mathcal{H}) = \frac{tp_{\text{grad}}}{tp_{\text{grad}} + fn_{\text{grad}}}, \tag{9}$$

where:

- $tp_{\text{grad}} = \sum_{e \in \mathcal{E}^+, \mathcal{B} \wedge \mathcal{H} \models e} \min(\{I_{\text{Gödel}}(\mu_{A_i}(\vec{t}), \mu_C(\vec{t})), \vec{t} \in \mathcal{D}^n, \mathcal{B} \wedge h_i \models e\})$
- $fp_{grad} = \sum_{e \in \mathcal{E}^-, \mathcal{B} \wedge \mathcal{H} \models e} max(\{I_{\text{Gödel}}(\mu_{A_i}(\vec{t}), \mu_C(\vec{t})), \vec{t} \in \mathcal{D}^n, h_i \in \mathcal{H}, \mathcal{B} \wedge h_i \models e\}) = fp$ as defined in Sect. 2.1 because we consider that $\mu_C(\vec{t}) = 1$ for all counter-examples, and then $I_{\text{Gödel}}(\mu_{A_i}(\vec{t}), \mu_C(\vec{t})) = 1$ if $\mu_{A_i}(\vec{t}) = 1$.
- $fn_{\text{grad}} = fn$ as defined in Sect. 2.1.

Note that gradual rules are useful only when the fuzzy sets used for the symbolic labeling of data have a sufficiently restrictive core and a sufficiently large support. Indeed, if the core and the support of the fuzzy sets were too close, few examples would have a membership degree in $(0, 1)$, and then gradual rules would tend to be equivalent to crisp ones. So, in this context, we can find fuzzy rules of each type with a good confidence degree.

4 Certainty rules

The meaning of a fuzzy rule "$A(\vec{x}) \rightarrow C(\vec{x})$" understood as a certainty rule is then "the more $\vec{t}$ is A, the more certain $\vec{t}$ is C". An example of a *certainty rule*, according to the database described in Sect. 2.2, is:

$$\text{distance}(x, y, \textbf{close}) \rightarrow \text{know}(x, y).$$

This rule means "the closer the houses, the more we are sure that the owners know each other". The certainty that the owners of close houses know each other can decrease when the distance between the houses grows.

This makes sense only if the antecedent is fuzzy. When A is not fuzzy, the rule means that if the antecedent holds for $\vec{t}$, then it is completely certain that $\vec{t}$ is restricted by C (possibly in a fuzzy way), otherwise nothing can be said. The implication known as Kleene-Dienes implication, which has a certainty meaning is [6]:

$$I_{\text{KD}}(\mu_A(\vec{t}), \mu_C(\vec{t})) = \max(1 - \mu_A(\vec{t}), \mu_C(\vec{t})).$$

For this kind of rule, we must focus on the confidence degree of the rule for the high level cut of the implication, corresponding to more certain conclusions. Note that here, certainty is a matter of necessity measure [6], while confidence is a matter of counting. The confidence degree for an α-cut is:

$$cf_{\text{cert}}(A(\vec{x}) \rightarrow C(\vec{x}))_\alpha = \frac{|\{\vec{t} \in \mathcal{D}^n \mid \mathcal{I}_{\text{ILP}} \models A(\vec{t}) \text{ and } I_{\text{KD}}(\mu_A(\vec{t}), \mu_C(\vec{t})) \geq \alpha \text{ and } C(\vec{t}) \in \mathcal{E}^+\}|}{|\{\vec{t} \in \mathcal{D}^n \mid \mathcal{I}_{\text{ILP}} \models A(\vec{t}) \text{ and } \mu_A(\vec{t}) \geq \alpha \text{ and } C(\vec{t}) \in \mathcal{E}\}|} \tag{10}$$

For favoring the confidence degree for the high level cuts of the implication, we use a Choquet integral as in [3] :

$$cf_{\text{cert}}(A(\vec{x}) \rightarrow C(\vec{x})) = \sum_{\alpha_i} (\alpha_i - \alpha_{i+1}) * cf_{\text{cert}}(A(\vec{x}) \rightarrow C(\vec{x}))_{\alpha_i},$$

where $\alpha_1 = 1 \geq \cdots \geq \alpha_n = 0$ are decreasing by order. The idea is to be more demanding for high α-level cuts, i.e., the smaller α, the greater the tolerance for exceptions in the corresponding crisp rule, in agreement with the intended meaning of the fuzzy rule. In the extreme case where only the evaluation acknowledges the fact, that it is more and more easy to cover examples when the consequent of the rule is fuzzy, we enlarge the scope of the consequent. In the general case, it is expected that the majority of the examples are in the core of the fuzzy set of the rule and that the confidence degree increases when α ranges between the core and the support. If only the antecedent of the rule is fuzzy, the confidence degree of the rule needs to be high for the high α-cuts of the antecedent of the rule, which correspond to crisp rules with a more narrow focus. Definition of the precision and the recall are the same than the gradual ones with :

- $tp_{\text{cert}} = \sum_{e \in \mathcal{E}^+, \mathcal{B} \wedge \mathcal{H} \models e} \min(\{I_{\text{KD}}(\mu_{A_i}(\vec{t}), \mu_C(\vec{t})), \vec{t} \in \mathcal{D}^n, \mathcal{B} \wedge h_i \models e\})$
- $fp_{\text{cert}} = \sum_{e \in \mathcal{E}^-, \mathcal{B} \wedge \mathcal{H} \models e} \max(\{I_{\text{KD}}(\mu_{A_i}(\vec{t}), \mu_C(\vec{t})), \vec{t} \in \mathcal{D}^n, h_i \in H, \mathcal{B} \wedge h_i \models e\}) = fp$ because we consider that $\mu_C(\vec{t}) = 1$ for all counter-examples, and then $I_{\text{KD}}(\mu_{A_i}(\vec{t}), \mu_C(\vec{t})) = 1$ if $\mu_A(\vec{t}) = 1$.
- $fn_{\text{cert}} = fn$.

This kind of rules is of interest for learning a concept in a broad sense. For instance, if we consider the concept "activity" of the well-known ILP benchmark database "mutagenesis",[1] active molecules have a positive activity level and non-active molecules have a negative one. Then, the fuzzy sets used will have disjoint parts around 0. A gradual rule may inform on the level of activity of the molecule, while a certainty rule will focus on determining if the molecule is somewhat active or not. This kind of rules can be also used for learning non-fuzzy concepts, when looking for rules that remain reasonably reliable when their scope is somewhat enlarged.

5 Experiments

In all reported experiments, the FOIL algorithm is used. Different rules are found by using the appropriate definition of the confidence degree in the gain function. The number of positive examples covered does not depend on the kind of rules we want to learn. As an illustration of our approach, we first explore a database that can be found on the PRETI platform (http://www.irit.fr/PRETI/). This database describes houses to let for vacations. There are more than 600 houses described in terms of about 25 attributes. A lot of these attributes are about distances between the house and another place (sea, fishing place, swimming pool, …) or about prices at different periods in a year (June, weekend, scholar vacations …). Clearly, these attributes may have a fuzzy interpretation. So, from our database, we build a fuzzy database by merely changing price, number of room, and distance information into fuzzy information such as "cheap", "expensive", "small_capacity", "high_capacity" (these two latter expressions refer to the size of the house), "far", "not_too_far", "not_far" together with a membership degree. The fuzzy sets are supposed here to be obtained from the intended meaning of the words of user's vocabulary. This is why the fuzzy sets are not necessarily providing a good coverage of the referential. In this approach, fuzzy sets are not tuned for improving the rules, since they represent the meaning of the user's words. For instance, "cheap" and "expensive" are described in Fig. 1. This allows us to describe fuzzy predicate expressions of the form $(\text{price}(x, \text{august}, \textbf{cheap}), \alpha)$ which means that house x is cheap in August. We can learn different concepts from this database. For instance, the algorithm found the following gradual rules:

$\text{price}(A, B, \textbf{expensive}), \text{washingmachine}(A) \rightarrow \text{comfort}(A, \textbf{medium})\text{area}(A, \text{NARBONNAIS}), \text{capacity}(A, \textbf{high}) \rightarrow \text{dist\_to}(A, \textit{sea}, \textbf{close})\text{comfort}(A, \textbf{weak}), \text{capacity}(A, \textbf{small}) \rightarrow \text{price}(A, \text{September}, \textbf{cheap})$.

[1] http://www.web.comlab.ox.ac.uk/oucl/research/areas/machlearn/mutagenesis.html

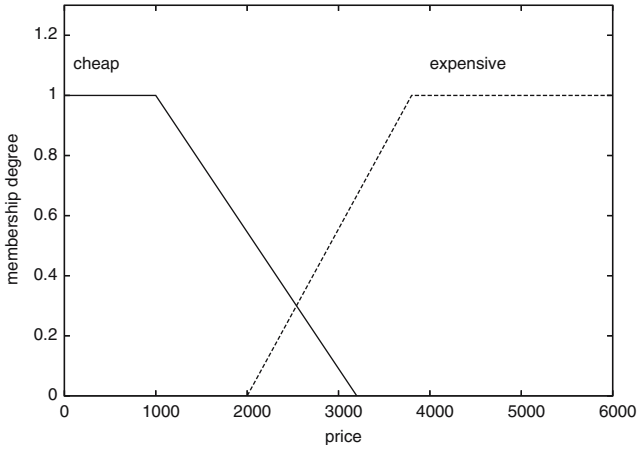

Fig. 1 Fuzzy sets representing the price of a vacation

The first rule can be read as " if there exists a month when the house is expensive and if the house has a washing machine, the more expensive the house is in this month, the more it is a medium comfort house". The meaning of the second rule is "if the house is located in the Narbonnais, the higher the capacity, the closer to the sea the house is". When using FOIL with the definition of the confidence degree of a rule viewed as certainty rules, the following rules are, for instance, obtained :

$\text{area}(A, \text{LAURAGAIS}), \text{price}(A, C, \textbf{expensive}) \rightarrow \text{comfort}(A, \textbf{medium})\text{comfort}(A, \textbf{high}) \rightarrow \text{dist\_to}(A, \text{sea}, \textbf{close})$
$\text{area}(A, \text{LIMOUXIN}) \rightarrow \text{price}(A, \text{September}, \textbf{cheap})$
$\text{price}(A, B, \textbf{expensive}), \text{phone}(A), \text{capacity}(A, \textbf{small}) \rightarrow \text{television}(A)$.

The first rule means "if a house is located in LAURAGAIS and if there exists a month when the house is expensive, the more the house is expensive in this month, the more we are certain that the comfort of the house is medium". Note that this rule is close to the first gradual rule above. It may indicate a strong relation between the expensive price of a house and its (at least) medium comfort. The second rule means " The higher the comfort of the house, the more certain the house is close to the sea". The third rule can be read as "if the house is located in the LIMOUXIN then we are certain that the house is cheap in September"; note that $\text{area}(A, \textit{LIMOUXIN})$ is not fuzzy. The last rule has a classical meaning since the consequent is not fuzzy.

In order to check the effectiveness of our algorithm, we applied our approach to the database "mutagenesis", which is a classical benchmark for ILP. This database describes molecules. The concept we want to learn is the activity level of a molecule. The database contains 188 instances described with seven predicates. Moreover, four predicates, including the concept to be learned, contain numerical attributes. The molecule is considered as "active" if and only if the activity

level is positive. Otherwise, the molecule is "inactive". For this experiment, we discretize all numerical attributes. The non-target attributes are discretized into a partition with three fuzzy terms corresponding to the fuzzy set : low, medium and high (see Fig. 3 for the discretization of the lumo attributes for instance). Since we have no expert for making the discretization manually, the fuzzy sets are here constructed automatically by choosing the three triangular fuzzy sets which have the same support size and which split the domain in three equal parts. A molecule will be considered as completely active if its activity level is greater than three and be considered less and less active when its activity level tends to 0 (see Fig 2). The molecules that have an activity level less than 0 will be considered as negative examples. A crisp database is obtained by making a discretization of the numerical attributes with the crisp sets which correspond to the 0.5-cut of the fuzzy sets (except for the concept to be learnt for which positive examples are those which have an activity level greater than 0). For the comparison, we first run FOIL on the crisp database issued from the fuzzy one. Next we use the fuzzy version of FOIL for independently inducing the two types of fuzzy rules. For each case, we compute the precision, the recall, and the accuracy according to the type of rules. Results for the whole database are shown in the following table.

Type of rules	Precision	Recall	Accuracy
Classical	0.86	0.94	0.86
Gradual	0.83	0.93	0.84
Certainty	0.91	0.91	0.88

We can observe that the fuzzy versions of FOIL (corresponding to the two types of rules) have performance that are close to the classical version. However, if we use fuzzy sets that are trapezoidal, the recall of gradual and certainty rules falls respectively to 0.26 and 0.25. This is due to the fact that, when membership degrees to the antecedent of the rules are close to 1, the constraint on membership degrees to the consequent may prevent to cover examples that have a low membership degree (a frequent situation in this base). This has a negative impact on the confidence of the rules. Thus, finding rules that have a confidence degree greater than the threshold may become very tricky. This illustrates the fact that gradual and certainty rule may not be adequate for machine learning (where one has to cover all the examples with a minimal set of rules) task for some fuzzy database. Nevertheless, the precision of the hypotheses found is very good. This illustrates that although sometimes gradual (or certainty) rules are not sufficient for explaining all the examples in the training set, they can provide some valuable additional information that cannot be extracted with a classical method.

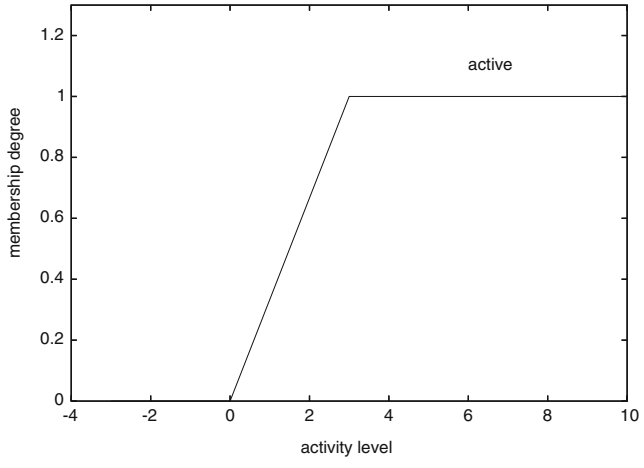

Fig. 2 Fuzzy sets representing the lumo properties of a molecule

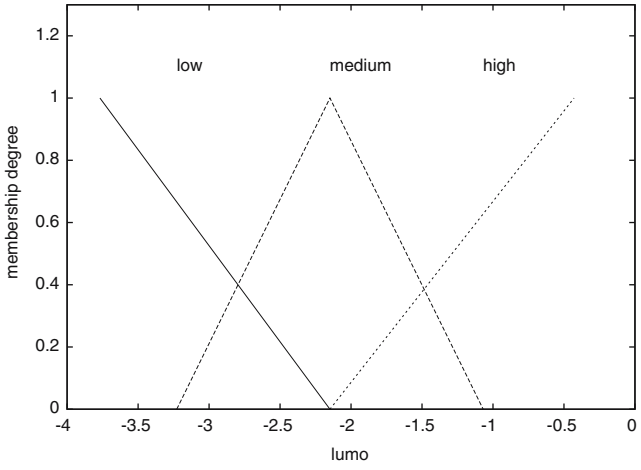

Fig. 3 Fuzzy set representing the activity of a molecule

Here are two examples of rules found by the algorithm:

Gradual rule

ind1(A, '*true*'), atm ($A, B, C, 22,$ **low**) $\rightarrow$ act (A, **active**)

Certainty rule

ind1(A, '*false*'), lumo (A, **low**), logp (A, **low**) $\rightarrow$ act (A, **active**).

6 Conclusion

In this paper we have considered two different kinds of first order fuzzy rules. These two types encode distinct information (gradual synergy between antecedents and consequents, and rules with exceptions) and both of them are thus worth inducing. Fuzzy rules can be induced by FOIL using the appropriate definition of the confidence degree corresponding to the type of rule to be learnt. We have shown that fuzzy predicates can be introduced in ILP for improving the expressivity of the rules that are learnt. This expressive rules have no simple crisp counterpart.

Gradual rules and certainty rules provide new kinds of summaries of features present in databases. Since these summaries involve membership degrees, a new definition of accuracy is needed. This kind of rules are natural when the database contains predicates whose meaning is typically fuzzy. Finding out these more expressive rules seems especially appropriate for data mining, since we are then interested in finding noticeable rules rather than covering all the examples, which contrasts with the goal of automatic classification.

In order to cover more examples in the database, we may try to learn both gradual and certainty rules in the same time. In this spirit, one approach has been proposed in [22] in order to learn rules together with their underlying implication connective (encoded by a matrix).

References

1. Bouchon-Meunier B, Marsala C (1999) Learning fuzzy decision rules. In: Bezdek JC, Dubois D, Prade H (eds) Fuzzy sets in approximate reasoning and information systems,The handbooks of fuzzy sets series. Kluwer, p 279–304
2. Clark P, Niblett T (1989) The cn2 induction algorithm. Mach Learn 3:261–283
3. Delgado M, Sanchez D, Vila MA (2000) Fuzzy cardinality based evaluation of quantified sentences. Int J Approx Reaso 23:23–66
4. Drobics M (2004) Choosing the best predicates for data-driven fuzzy modeling. In: IEEE international conference on fuzzy systems p 245–249
5. Drobics M, Bodenhofer U, Klement EP (2003) FS-FOIL: an inductive learning method for extracting interpretable fuzzy descriptions. Int J Approx Reason 32:131–152
6. Dubois D, Prade H (1996) What are fuzzy rules and how to use them. Fuzzy Sets Syst 84:169–189
7. Fertig CS, Freitas AA, Arruda LVR, Kaestner C (1999) A fuzzy beam-search rule induction algorithm. In Zytkow J, Rauch J, (eds) In: Proceedings of PKDD-99, 3 Conference on principles and pratice of knowledge discovery in databases, Vol 1704 in LNAI. Springer, Berlin Heidelberg New York, p 341–347
8. Halpern J (1990) An analysis of first-order logics of probability. Artif Intell 46:310–355
9. Hüllermeier E (2001) Implication-based fuzzy association rules. In: De Raedt L, Siebes A (eds) In: Proceedings of PKDD-01, 5th Conferance on principles and pratice of knowledge discovery in databases, Vol 2168 in LNAI. Springer, Berlin Heidelberg New York, p 241–252
10. Janikow CZ (1998) Fuzzy decision trees: issues and methods. IEEE Trans Syst Man Cybern Part B: Cybern 28(1):1–14
11. Klement EP, Mesiar R, Pap E (2000) Triangular norms. Kluwer, Dordrecht
12. Marsala C (1998) Application of fuzzy rule induction to data mining. In:Proceedings of the third international conference on flexible query answering systems. Springer Berlin Heidelberg New York, p 260–271
13. Mitra S, Hayashi Y (2000) Neuro-fuzzy rule generation: survey in soft computing framework. IEEE Trans Neural Netw 11:748–768
14. Muggleton SH (1995) Inverse entailment and Progol. New Gener Comput 13:245–286
15. Nauck D, Kruse R (1999) Neuro-fuzzy methods in fuzzy rule generation. In:Bezdek JC, Dubois D, Prade H, (eds), Fuzzy sets in approximate reasoning and information systems, The handbooks of fuzzy sets series, Kluwer p 305–334
16. Nienhuys-Cheng SH, de Wolf R (1997) Foundations of inductive logic programming. LNAI, Vol 1228. Springer, Berlin Heidelberg New York
17. Prade H, Richard G, Serrurier M (2003) Enriching relational learning with fuzzy predicates. In: Proceedings of 7th European conference on principle and practice of knowledge discovery in databases (PKDD 2003), LNAI, Vol 2838 Springer, Berlin Heidelberg New York, p 399–410.
18. Prade H, Richard G, Serrurier M (2003) On the induction of different kinds of first-order fuzzy rules. In: Proceedings of 7th European conference on symbolic and qualitative approaches to reasoning with uncertainty (ECSQARU-03), LNAI,Vol 2711 Springer, Berlin Heidelberg New York, p 370–381
19. Quinlan JR (1986) Induction of decision trees. Mach Learning 1:81–106,
20. Quinlan JR (1990) Learning logical definitions from relations. Mach Learn 5:239–266
21. Quinlan JR (1993) C4.5 Program for machine learning. Morgan Kaufmann
22. Serrurier M, Prade H, Sudkamp T, Dubois D, Richard G (2004) Learning first order fuzzy rules with their implication operator. In:Proceedings of the 10th international conference on information processing and management of uncertainty in knowledge-based systems (IPMU'04), p 831–838
23. Shibata D, Inuzuka N, Katoand S, Matsui T, Itoh H (1999) Methodologies for knowledge discovery and data mining. In: Zhong N, Zhou L (eds) In: Proceedings of the third pacific-Asia conference on knowledge discovery and data mining (PAKDD-99), LNAI, Vol 1574. Springer, Berlin Heidelberg New York, pp 268–273