PN-OWL: A Two Stage Algorithm to Learn Fuzzy Concept Inclusions from OWL Ontologies

Franco Alberto Cardillo
CNR-ILC
Pisa, Italy

Franca Debole
CNR-ISTI
Pisa, Italy

Umberto Straccia
CNR-ISTI
Pisa, Italy

March 14, 2023

Abstract

OWL ontologies are a quite popular way to describe structured knowledge in terms of classes, relations among classes and class instances.

In this paper, given a target class T of an OWL ontology, with a focus on ontologies with real- and boolean-valued data properties, we address the problem of learning graded fuzzy concept inclusion axioms with the aim of describing enough conditions for being an individual classified as instance of the class T.

To do so, we present PN-OWL that is a two-stage learning algorithm made of a P-stage and an N-stage. Roughly, in the P-stage the algorithm tries to cover as many positive examples as possible (increase *recall*), without compromising too much *precision*, while in the N-stage, the algorithm tries to rule out as many false positives, covered by the P-stage, as possible. PN-OWL then aggregates the fuzzy inclusion axioms learnt at the P-stage and the N-stage by combining them via aggregation functions to allow for a final decision whether an individual is instance of T or not.

We also illustrate its effectiveness by means of an experimentation. An interesting feature is that fuzzy datatypes are built automatically, the learnt fuzzy concept inclusions can be represented directly into Fuzzy OWL 2 and, thus, any Fuzzy OWL 2 reasoner can then be used to automatically determine/classify (and to which degree) whether an individual belongs to the target class T or not.

1 Introduction

OWL 2 ontologies [67] are a popular means to represent *structured* knowledge and its formal semantics is based on *Description Logics* (DLs) [6]. The basic ingredients of DLs are concept descriptions, inheritance relationships among them and instances of them.

In this work, we focus on the problem of automatically learning fuzzy $\mathcal{EL}(\mathbf{D})$ concept inclusion axioms from OWL 2 ontologies based on the terminology and instances within it. Despite an important amount of work has been carried about DLs, the application of machine learning techniques to OWL 2 ontologies is relatively less addressed compared to the *Inductive Logic Programming* (ILP) setting (see *e.g.* [69, 70] for more insights on ILP). We refer the reader to [54, 71] for an overview and to Section 5.

In this paper, the problem we address is the following: given a target class T of an OWL ontology, learn rule-like graded fuzzy $\mathcal{EL}(\mathbf{D})$ [13, 16, 86] concept inclusion axioms with the aim of describing sufficient conditions for being an individual classified as instance of the class T.

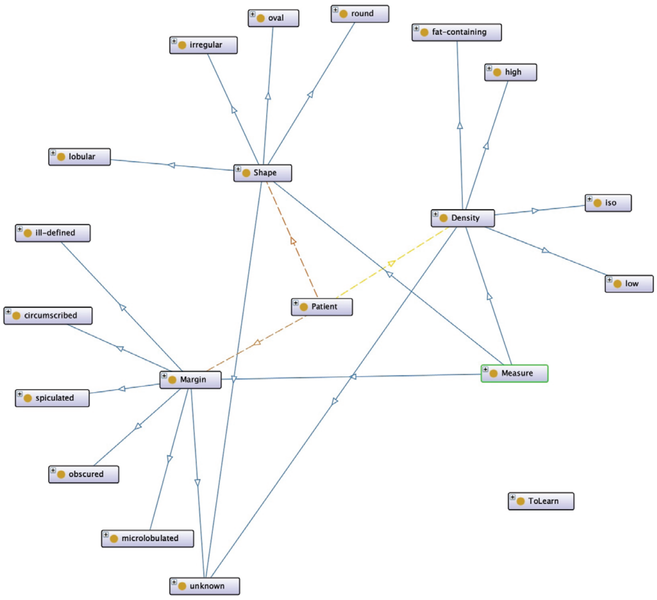

Figure 1: Excerpt of the mammographic ontology.

The following example illustrates the problem we are going to address.[1]

Example 1.1 *Consider an ontology [18, 20] that describes the meaningful entities of mammography image analysis. An excerpt of this ontology is given in Fig. 1. Now, suppose we have a set of patients that exhibit a cancer (*positive *examples) and another set which does not (*negative *examples). Now, one may ask about what characterises the patients with cancer (our target class T). Then one may learn from the ontology the following fuzzy $\mathcal{EL}(\mathbf{D})$ concept inclusion (expressed in the so-called Fuzzy OWL 2 syntax [13])*[2]

> (**implies** (**and** (**some** `hasDensity fat-containing`) (**some** `hasMargin spiculated`) (**some** `hasShape irregular`) (**some** `hasAge hasAge_high`)) `Cancer` 0.86) ,

where the fuzzy set `hasAge_high` *is defined as*

> (**define-fuzzy-concept** `hasAge_high right- (0,150,60,80)`)

In words,

> *"if the density is fat-containing, the margin is spiculated, the shape is irregular and the age is high then it is cancer, with confidence* 0.86*".*

In this work, the objective is the same as in *e.g.* [20, 53, 87] except that now we propose to rely on an adaptation of the PN-rule [2, 3, 40, 41] algorithm to the (fuzzy) OWL case. Further, like in [51, 87], we continue to support so-called *fuzzy concept descriptions* and *fuzzy concrete domains* [59, 85, 86], such as the expression (**some** `hasAge hasAge_high`) (*viz.* an aged person)

[1] See also *e.g.* [20, 51, 53, 87] for an analogous example.

[2] `http://www.umbertostraccia.it/cs/software/fuzzyDL/fuzzyDL.html`

in Example 1.1 above, which is a fuzzy concept, *i.e.* a concept for which the belonging of an individual to the class is not necessarily a binary yes/no question, but rather a matter of (truth) degree in $[0, 1]$.

For instance, in our example, the degree depends on the person's age: the higher the age the older is the person, *e.g.* modelled via a so-called *right shoulder function* (see Figure 2(d)). Here, the range of the 'attribute' `hasAge` becomes a so-called fuzzy concrete domain [85, 86].

Let us recap that the basic principle of PN-rule consists of a *P-stage* in which *positive* rules (called *P-rules*) are learnt to cover as many as possible instances of a target class, and keeping the non-positive rate at a reasonable level, and an *N-stage* in which *negative* rules (called *N-rules*) are learnt to remove most of the non-positive examples covered by the P-stage. The two rule sets are then used to build up a decision method to classify an object being instance of the target class or not [2, 3, 40, 41]. It is worth noting that what differentiates this method from all others is its second stage. It learns N-rules that essentially remove the non-positive examples (so-called false positives) collectively covered by the union of all the P-rules.

The following are the main features of our two stage algorithm, called PN-OWL:

- at the P-stage, it generates a set of fuzzy $\mathcal{EL}(\mathbf{D})$ inclusion axioms, the P-rules, that cover as many as possible instances of a target class T without compromising too much the amount on non-positives (*i.e.* , try to increase the so-called *recall*);
- at the N-stage, it generates a set of fuzzy $\mathcal{EL}(\mathbf{D})$ inclusion axioms, the N-rules, that cover as many as possible of non-positive instances of class T (*i.e.* , then try to increase the so-called *precision*);
- the fuzzy inclusion axioms are then combined (aggregated) into a new fuzzy inclusion axiom describing sufficient conditions for being an individual classified as an instance of the target class T (*i.e.* the combination aims at increasing the overall effectiveness, *e.g.* the so-called F1-measure);
- all fuzzy inclusion axioms may possibly include fuzzy concepts and fuzzy concrete domains, where each axiom has a leveraging weight (specifically, called *confidence* or *precision*);
- all generated fuzzy concept inclusion axioms can be directly encoded as *Fuzzy OWL 2* axioms [12, 13]. Therefore, a Fuzzy OWL 2 reasoner, such as *fuzzyDL* [11, 15], can then be used to automatically determine (and to which degree) whether an individual belongs to the target class T.

We will illustrate the effectiveness of PN-OWL by means of an experimentation that shows that the effectiveness of the combined approach increases w.r.t. a baseline based on the P-stage only.

In the following, we proceed as follows: in Section 2, for the sake of completeness, we recap the salient notions we will rely on this paper. Then, in Section 3 we will present our algorithm PN-OWL, which is evaluated in Section 4. In Section 5 we compare our work with closely related work appeared so far. Section 6 concludes and points to some topics of further research.

2 Background

We introduce the main notions related to *(Mathematical) Fuzzy Logics* and *Fuzzy Description Logics* we will use in this work (see also [16, 86]).

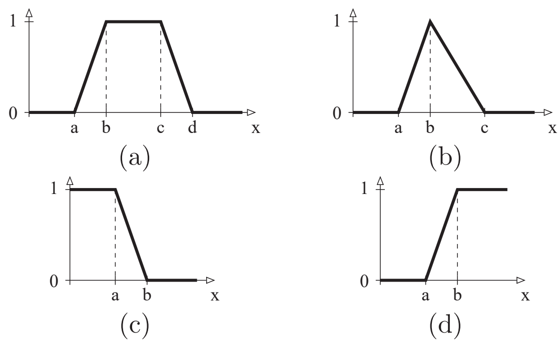

Figure 2: (a) Trapezoidal function $trz(a, b, c, d)$, (b) triangular function $tri(a, b, c)$, (c) left shoulder function $ls(a, b)$, and (d) right shoulder function $rs(a, b)$.

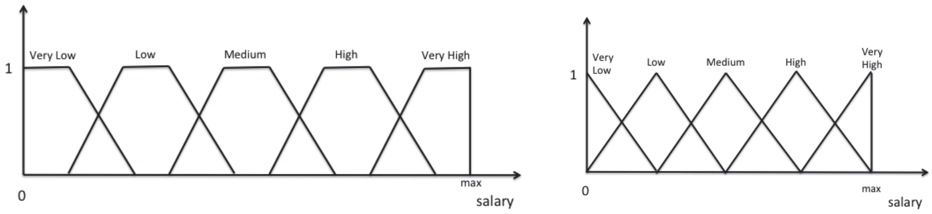

Figure 3: Uniform fuzzy sets over salaries: trapezoidal (left) or triangular (right).

Mathematical Fuzzy Logic. *Fuzzy Logic* is the logic of *fuzzy sets* [93]. A *fuzzy set* A over a countable crisp set X is a function $A \colon X \to [0, 1]$, called *fuzzy membership* function of A. A *crisp set* A is characterised by a membership function $A \colon X \to \{0, 1\}$ instead. Often, fuzzy set operations conform to $(A \cap B)(x) = \min(A(x), B(x))$, $(A \cup B)(x) = \max(A(x), B(x))$ and $\bar{A}(x) = 1 - A(x)$ ($\bar{A}$ is the set complement of A), the *cardinality* of a fuzzy set is defined as $|A| = \sum_{x \in X} A(x)$, while the *inclusion degree* between A and B is defined as $deg(A, B) = \frac{|A \cap B|}{|A|}$.

The trapezoidal, the triangular, the left-shoulder function, and the right-shoulder function are frequently used to specify membership functions of fuzzy sets (see Figure 2).

One easy and typically satisfactory method to define the membership functions is to uniformly partition the range of, *e.g.* person's age values (bounded by a minimum and maximum value), into 3, 5 or 7 fuzzy sets using triangular (or trapezoidal) functions (see Figure 3). Another popular approach may consist in using the so-called *c-means* fuzzy clustering algorithm (see, *e.g.* [8]) with 3,5 or 7 clusters, where the fuzzy membership functions are triangular functions built around the centroids of the clusters (see also [38]).

In *Mathematical Fuzzy Logic* [37], the convention prescribing that a formula ϕ is either true or false (w.r.t. an interpretation $\mathcal{I}$) is changed and is a matter of degree measured on an ordered scale that is no longer $\{0, 1\}$, but typically $[0, 1]$. This degree is called *degree of truth* of the formula ϕ in the interpretation $\mathcal{I}$. A *fuzzy formula* has the form $\langle \phi, \alpha \rangle$, where $\alpha \in (0, 1]$ and ϕ is a First-Order Logic (FOL) formula, encoding that the degree of truth of ϕ is *greater than or equal to* α. From a semantics point of view, a *fuzzy interpretation* $\mathcal{I}$ maps each atomic formula

Table 1: Truth combination functions for fuzzy logics.

	Łukasiewicz	Gödel	Product
$\alpha_1 \otimes \alpha_2$	$\max(\alpha_1 + \alpha_2 - 1, 0)$	$\min(\alpha_1, \alpha_2)$	$\alpha_1 \cdot \alpha_2$
$\alpha_1 \oplus \alpha_2$	$\min(\alpha_1 + \alpha_2, 1)$	$\max(\alpha_1, \alpha_2)$	$\alpha_1 + \alpha_2 - \alpha_1 \cdot \alpha_2$
$\alpha_1 \Rightarrow \alpha_2$	$\min(1 - \alpha_1 + \alpha_2, 1)$	$\begin{cases} 1 & \text{if } \alpha_1 \leq \alpha_2 \\ \alpha_2 & \text{otherwise} \end{cases}$	$\begin{cases} 1 & \text{if } \alpha_1 \leq \alpha_2 \\ \alpha_2/\alpha_1 & \text{otherwise} \end{cases}$
$\ominus\, \alpha$	$1 - \alpha$	$\begin{cases} 1 & \text{if } \alpha = 0 \\ 0 & \text{otherwise} \end{cases}$	$\begin{cases} 1 & \text{if } \alpha = 0 \\ 0 & \text{otherwise} \end{cases}$

into $[0, 1]$ and is then extended inductively to all FOL formulae as follows:

$$\begin{array}{rcl} \mathcal{I}(\phi \land \psi) & = & \mathcal{I}(\phi) \otimes \mathcal{I}(\psi) \ \ , \ \ \mathcal{I}(\phi \lor \psi) \ = \ \mathcal{I}(\phi) \oplus \mathcal{I}(\psi) \\ \mathcal{I}(\phi \to \psi) & = & \mathcal{I}(\phi) \Rightarrow \mathcal{I}(\psi) \ \ , \ \ \mathcal{I}(\neg\phi) \ = \ \ominus\mathcal{I}(\phi) \\ \mathcal{I}(\exists x.\phi(x)) & = & \sup_{y \in \Delta^{\mathcal{I}}} \mathcal{I}(\phi(y)) \ \ , \ \ \mathcal{I}(\forall x.\phi(x)) \ = \ \inf_{y \in \Delta^{\mathcal{I}}} \mathcal{I}(\phi(y)) \ , \end{array}$$

where $\Delta^{\mathcal{I}}$ is the (non-empty) domain of $\mathcal{I}$, and $\otimes$, $\oplus$, $\Rightarrow$, and $\ominus$ are so-called *t-norms*, *t-conorms*, *implication functions*, and *negation functions*, respectively, which extend the Boolean conjunction, disjunction, implication, and negation, respectively, to the fuzzy case.

One usually distinguishes three different logics, namely Łukasiewicz, Gödel, and Product logics [37],[3] whose truth combination functions are reported in Table 1.

An *r-implication* is an implication function obtained as the *residuum* of a continuous t-norm $\otimes$, *i.e.* $\alpha_1 \Rightarrow \alpha_2 = \sup\{\alpha_3 \mid \alpha_1 \otimes \alpha_3 \leq \alpha_2\}$. Note also, that given an r-implication $\Rightarrow_r$, we may also define its related negation $\ominus_r \alpha$ by means of $\alpha \Rightarrow_r 0$ for every $\alpha \in [0, 1]$.

The notions of satisfiability and logical consequence are defined in the standard way, where a fuzzy interpretation $\mathcal{I}$ *satisfies* a fuzzy formula $\langle \phi, \alpha \rangle$, or $\mathcal{I}$ is a *model* of $\langle \phi, \alpha \rangle$, denoted as $\mathcal{I} \models \langle \phi, \alpha \rangle$, iff $\mathcal{I}(\phi) \geq \alpha$. Notably, from $\langle \phi, \alpha_1 \rangle$ and $\langle \phi \to \psi, \alpha_2 \rangle$ one may conclude (if $\to$ is interpreted as an r-implication) $\langle \psi, \alpha_1 \otimes \alpha_2 \rangle$ (this inference is called *fuzzy modus ponens*).

Fuzzy Description Logics basics. We recap here the fuzzy DL $\mathcal{ALC}_{@}(\mathbf{D})$, which extends the well-known fuzzy DL $\mathcal{ALC}(\mathbf{D})$ [85] with the *aggregated concept* construct [14] (indicated with the symbol @). $\mathcal{ALC}_{@}(\mathbf{D})$ is expressive enough to capture the main ingredients of fuzzy DLs we are going to consider here.

We start with the notion of *fuzzy concrete domain*, that is a tuple $\mathbf{D} = \langle \Delta^{\mathbf{D}}, \cdot^{\mathbf{D}} \rangle$ with datatype domain $\Delta^{\mathbf{D}}$ and a mapping $\cdot^{\mathbf{D}}$ that assigns to each data value an element of $\Delta^{\mathbf{D}}$, and to every 1-ary datatype predicate $\mathbf{d}$ a 1-ary fuzzy relation over $\Delta^{\mathbf{D}}$. Therefore, $\cdot^{\mathbf{D}}$ maps indeed each datatype predicate into a function from $\Delta^{\mathbf{D}}$ to $[0, 1]$. In the domain of numbers, typical datatypes predicates $\mathbf{d}$ are characterized by the well known membership functions (see also Fig. 2)

$$\begin{array}{rcl} \mathbf{d} & \to & ls(a, b) \mid rs(a, b) \mid tri(a, b, c) \mid trz(a, b, c, d) \\ & & \mid \ \geq_v \ \mid \ \leq_v \ \mid \ =_v \ , \end{array}$$

[3]Notably, a theorem states that any other continuous t-norm can be obtained as a combination of them.

where additionally $\geq_v$ (resp. $\leq_v$ and $=_v$) corresponds to the crisp set of data values that are no less than (resp. no greater than and equal to) the value v.

Aggregation Operators (AOs) are mathematical functions that are used to combine different pieces of information. There exist large number of different AOs that differ on the assumptions on the data (data types) and about the type of information that we can incorporate in the model [90]. There is no standard definition of AO. Usually, given a domain $\mathbb{D}$ (such as the reals), an AO of dimension n is a mapping $@ : \mathbb{D}^n \to \mathbb{D}$. For us, $\mathbb{D} = [0,1]$. Thus, an AO aggregates n values of n different criteria. In our scenario, such criteria will be represented by using fuzzy concepts from a fuzzy ontology and we assume to have a finite family $@_1, \ldots, @_l$ of AOs within our language.

Now, consider pairwise disjoint alphabets $\mathbf{I}, \mathbf{A}$ and $\mathbf{R}$, where $\mathbf{I}$ is the set of *individuals*, $\mathbf{A}$ is the set of *concept names* (also called *atomic concepts* or *class names*) and $\mathbf{R}$ is the set of *role names*. Each role is either an *object property* or a *datatype property*. The set of *concepts* are built from concept names A using connectives and quantification constructs over object properties R and datatype properties S, as described by the following syntactic rule ($n_i \geq 1$):

$$\begin{array}{rcl} C & \to & \top \mid \bot \mid A \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \neg C \mid C_1 \to C_2 \mid \\ & & \exists R.C \mid \forall R.C \mid \exists S.\mathbf{d} \mid \forall S.\mathbf{d} \mid \\ & & @_i(C_1, \ldots, C_{n_i}) \ . \end{array}$$

An *ABox* $\mathcal{A}$ consists of a finite set of assertion axioms. An *assertion* axiom is an expression of the form $\langle a{:}C, \alpha\rangle$ (called *concept assertion*, a is an instance of concept C to degree greater than or equal to α) or of the form $\langle (a_1, a_2){:}R, \alpha\rangle$ (called *role assertion*, (a_1, a_2) is an instance of object property R to degree greater than or equal to α), where a, a_1, a_2 are individual names, C is a concept, R is an object property and $\alpha \in (0,1]$ is a truth value. A *Terminological Box* or *TBox* $\mathcal{T}$ is a finite set of *General Concept Inclusion* (GCI) axioms, where a fuzzy GCI is of the form $\langle C_1 \sqsubseteq C_2, \alpha\rangle$ (C_1 is a sub-concept of C_2 to degree greater than or equal to α), where C_i is a concept and $\alpha \in (0,1]$. We may omit the truth degree α of an axiom; in this case $\alpha = 1$ is assumed and we call the axiom *crisp*. We also write $C_1 = C_2$ as a macro for the two GCIs $C_1 \sqsubseteq C_2$ and $C_2 \sqsubseteq C_1$. We may also call a fuzzy GCI of the form $\langle C \sqsubseteq A, \alpha\rangle$, where A is a concept name, a *rule* and C its *body*. A *Knowledge Base* (KB) is a pair $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}\rangle$, where $\mathcal{T}$ is a TBox and $\mathcal{A}$ is an ABox. With $\mathsf{I}_{\mathcal{K}}$ we denote the set of individuals occurring in $\mathcal{K}$.

Concerning the semantics, let us fix a fuzzy logic, a fuzzy concrete domain $\mathbf{D} = \langle \Delta^{\mathbf{D}}, \cdot^{\mathbf{D}}\rangle$ and aggregation operators $@_i : [0,1]^{n_i} \to [0,1]$. Now, unlike classical DLs in which an interpretation $\mathcal{I}$ maps *e.g.* a concept C into a set of individuals $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, *i.e.* $\mathcal{I}$ maps C into a function $C^{\mathcal{I}} : \Delta^{\mathcal{I}} \to \{0,1\}$ (either an individual belongs to the extension of C or does not belong to it), in fuzzy DLs, $\mathcal{I}$ maps C into a function $C^{\mathcal{I}} : \Delta^{\mathcal{I}} \to [0,1]$ and, thus, an individual belongs to the extension of C to some degree in $[0,1]$, *i.e.* $C^{\mathcal{I}}$ is a fuzzy set. Specifically, a *fuzzy interpretation* is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a nonempty (crisp) set $\Delta^{\mathcal{I}}$ (the *domain*) and of a *fuzzy interpretation function* $\cdot^{\mathcal{I}}$ that assigns: *(i)* to each atomic concept A a function $A^{\mathcal{I}} : \Delta^{\mathcal{I}} \to [0,1]$; *(ii)* to each object property R a function $R^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \to [0,1]$; *(iii)* to each datatype property S a function $S^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathbf{D}} \to [0,1]$; *(iv)* to each individual a an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ such that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ if $a \neq b$ (the so-called *Unique Name Assumption*); and *(v)* to each data value v an element $v^{\mathcal{I}} \in \Delta^{\mathbf{D}}$. Now, a fuzzy interpretation function is extended to concepts as specified below (where $x \in \Delta^{\mathcal{I}}$):

$$\begin{aligned}
&\top^{\mathcal{I}}(x) = 1 \ , \ \bot^{\mathcal{I}}(x) \ = \ 0 \ , \ (C \sqcap D)^{\mathcal{I}}(x) = C^{\mathcal{I}}(x) \otimes D^{\mathcal{I}}(x) \\
&(C \sqcup D)^{\mathcal{I}}(x) = C^{\mathcal{I}}(x) \oplus D^{\mathcal{I}}(x) \ , \ (\neg C)^{\mathcal{I}}(x) = \ominus C^{\mathcal{I}}(x) \\
&(C \rightarrow D)^{\mathcal{I}}(x) = C^{\mathcal{I}}(x) \Rightarrow D^{\mathcal{I}}(x) \ , \ (\forall R.C)^{\mathcal{I}}(x) = \inf_{y \in \Delta^{\mathcal{I}}} \{R^{\mathcal{I}}(x,y) \Rightarrow C^{\mathcal{I}}(y)\} \\
&(\exists R.C)^{\mathcal{I}}(x) = \sup_{y \in \Delta^{\mathcal{I}}} \{R^{\mathcal{I}}(x,y) \otimes C^{\mathcal{I}}(y)\} \ , \ (\forall S.\mathbf{d})^{\mathcal{I}}(x) = \inf_{y \in \Delta^{\mathbf{D}}} \{S^{\mathcal{I}}(x,y) \Rightarrow \mathbf{d}^{\mathbf{D}}(y)\} \\
&(\exists S.\mathbf{d})^{\mathcal{I}}(x) = \sup_{y \in \Delta^{\mathbf{D}}} \{S^{\mathcal{I}}(x,y) \otimes \mathbf{d}^{\mathbf{D}}(y)\} \ , \\
&(@_i(C_1, \ldots, C_{n_i}))^{\mathcal{I}}(x) = @_i({C_1}^{\mathcal{I}}(x), \ldots, {C_{n_i}}^{\mathcal{I}}(x)) \ .
\end{aligned}$$

The *satisfiability of axioms* is then defined by the following conditions: *(i)* $\mathcal{I}$ satisfies an axiom $\langle a{:}C, \alpha\rangle$ if $C^{\mathcal{I}}(a^{\mathcal{I}}) \geq \alpha$; *(ii)* $\mathcal{I}$ satisfies an axiom $\langle (a,b){:}R, \alpha\rangle$ if $R^{\mathcal{I}}(a^{\mathcal{I}}, b^{\mathcal{I}}) \geq \alpha$; *(iii)* $\mathcal{I}$ satisfies an axiom $\langle C \sqsubseteq D, \alpha\rangle$ if $(C \sqsubseteq D)^{\mathcal{I}} \geq \alpha$ with[4] $(C \sqsubseteq D)^{\mathcal{I}} = \inf_{x \in \Delta^{\mathcal{I}}}\{C^{\mathcal{I}}(x) \Rightarrow D^{\mathcal{I}}(x)\}$. $\mathcal{I}$ is a *model* of $\mathcal{K} = \langle \mathcal{A}, \mathcal{T}\rangle$ iff $\mathcal{I}$ satisfies each axiom in $\mathcal{K}$. If $\mathcal{K}$ has a model we say that $\mathcal{K}$ is *satisfiable* (or *consistent*). We say that $\mathcal{K}$ *entails* axiom τ, denoted $\mathcal{K} \models \tau$, if any model of $\mathcal{K}$ satisfies τ. The *best entailment degree* of τ of the form $C \sqsubseteq D$, $a{:}C$ or $(a,b){:}R$, denoted $bed(\mathcal{K}, \tau)$, is defined as

$$bed(\mathcal{K}, \tau) = \sup\{\alpha \mid \mathcal{K} \models \langle \tau, \alpha\rangle\} \ .$$

Remark 1 *Please note that* $bed(\mathcal{K}, a{:}C) = 1$ *(*i.e. $\mathcal{K} \models a{:}C$*) implies that* $bed(\mathcal{K}, a{:}\neg C) = 0$ *holds, and similarly,* $bed(\mathcal{K}, a{:}\neg C) = 1$ *(*i.e. $\mathcal{K} \models a{:}\neg C$*) implies that* $bed(\mathcal{K}, a{:}C) = 0$ *holds. However, in both cases the other way around does not hold. Furthermore, we may well have that both* $bed(\mathcal{K}, a{:}C) = \alpha_1 > 0$ *and* $bed(\mathcal{K}, a{:}\neg C) = \alpha_2 > 0$ *hold.*

Now, consider concept C, a rule $C \sqsubseteq A$, a KB $\mathcal{K}$ and a set of individuals I. Then the *cardinality* of C w.r.t. $\mathcal{K}$ and I, denoted $|C|^{\mathsf{I}}_{\mathcal{K}}$, is defined as

$$|C|^{\mathsf{I}}_{\mathcal{K}} = \sum_{a \in \mathsf{I}} bed(\mathcal{K}, a{:}C) \ . \tag{1}$$

The *crisp cardinality* (denoted $\lceil C\rceil^{\mathsf{I}}_{\mathcal{K}}$) is defined similarly by replacing in Eq. 1 the term $bed(\mathcal{K}, a{:}C)$ with $\lceil bed(\mathcal{K}, a{:}C)\rceil$.

Eventually, we say that the *application* of rule $C \sqsubseteq A$ to individual a w.r.t. $\mathcal{K}$ is $bed(\mathcal{K}, C{:}a)$ and that rule $C \sqsubseteq A$ *applies* to individual a w.r.t. $\mathcal{K}$ if $bed(\mathcal{K}, C{:}a) > 0$.

3 PN-OWL

At first, we introduce our learning problem.

3.1 The Learning Problem

In general terms, the learning problem we are going to address is stated as follows. Consider

1. a satisfiable crisp KB $\mathcal{K}$ and its individuals $\mathsf{I}_{\mathcal{K}}$;
2. a *target concept name* T;

[4]However, note that under standard logic $\sqsubseteq$ is interpreted as $\Rightarrow_z$ and not as $\Rightarrow_{kd}$.

3. an associated classification function $f_T\colon \mathsf{I}_{\mathcal{K}} \to \{-1,0,1\}$, where for each $a \in \mathsf{I}_{\mathcal{K}}$, the values (*labels*) correspond to

$$f_T(a) = \begin{cases} 1 & a \text{ is a } \textit{positive} \text{ example w.r.t. } T \\ -1 & a \text{ is a } \textit{negative} \text{ example w.r.t. } T \\ 0 & a \text{ is an } \textit{unlabelled} \text{ example w.r.t. } T \end{cases}$$

4. the partitioning of the examples into

$$\begin{aligned} \mathcal{E}^+ &= \{(a,1) \mid a \in \mathsf{I}_{\mathcal{K}}, f_T(a) = 1\} \triangleright \text{the positive examples} \\ \mathcal{E}^- &= \{(a,-1) \mid a \in \mathsf{I}_{\mathcal{K}}, f_T(a) = -1\} \triangleright \text{the negative examples} \\ \mathcal{E}^u &= \{(a,0) \mid a \in \mathsf{I}_{\mathcal{K}}, f_T(a) = 0\} \triangleright \text{the unlabelled examples} \end{aligned}$$

   where $\mathcal{E}^+ \neq \emptyset$ is assumed. We define $\mathcal{E} = \mathcal{E}^+ \cup \mathcal{E}^- \cup \mathcal{E}^u$ as the set of all examples, and with $\overline{\mathcal{E}^+} = \mathcal{E} \setminus \mathcal{E}^+$ we denote the set of *non-positive* examples.

5. the set of individuals $\mathsf{I}_{\mathcal{S}} = \{a \mid (a,l) \in \mathcal{S}\}$, where $\mathcal{S} \subseteq \mathcal{E}$ is a set of examples. Moreover, we define

$$\begin{aligned} \mathsf{I}_{\mathcal{E}^+} &= \{a \mid (a,1) \in \mathcal{E}^+\} \triangleright \text{the positive individuals} \\ \mathsf{I}_{\mathcal{E}^-} &= \{a \mid (a,-1) \mid a \in \mathcal{E}^-\} \triangleright \text{the negative individuals} \\ \mathsf{I}_{\mathcal{E}^u} &= \{a \mid (a,0) \mid a \in \mathcal{E}^u\} \triangleright \text{the unlabelled individuals} \\ \mathsf{I}_{\overline{\mathcal{E}^+}} &= \mathsf{I}_{\mathcal{K}} \setminus \mathsf{I}_{\mathcal{E}^+} \triangleright \text{the non-positive individuals} \end{aligned}$$

6. a *hypothesis space* of classifiers $\mathcal{H} = \{h\colon \mathsf{I}_{\mathcal{K}} \to [0,1]\}$;

7. a *training set* $\mathcal{E}_{train} \subset \mathcal{E}$ of individual-label pairs, with $\mathcal{E}_{train} \cap \mathcal{E}^+ \neq \emptyset$;

8. a *test set* $\mathcal{E}_{test} = \mathcal{E} \setminus \mathcal{E}_{train}$.

We assume that the only axioms involving T in $\mathcal{K}$ are either of the form $a{:}T$ or $a{:}\neg T$. We write $\mathcal{E}(a) = 1$ if a is a positive example (*i.e.* $a \in \mathsf{I}_{\mathcal{E}^+}$), $\mathcal{E}(a) = -1$ if a is a negative example (*i.e.* $a \in \mathsf{I}_{\mathcal{E}^-}$) and $\mathcal{E}(a) = 0$ otherwise.

The general goal is to learn a classifier function $\bar{h} \in \mathcal{H}$ that is the result of *Empirical Risk Minimisation* (ERM) on $\mathcal{E}_{train}$, *i.e.*

$$\bar{h} \quad = \quad \arg\min_{h \in \mathcal{H}} R(h, \mathcal{E}_{train}) \quad = \quad \frac{1}{|\mathcal{E}_{train}|} \sum_{a \in \mathsf{I}_{\mathcal{E}_{train}}} L(h(a), \mathcal{E}_{train}(a)) \ ,$$

where L is a *loss function* such that $L(\hat{l}, l)$ measures how different the prediction $\hat{l}$ of a hypothesis is from the true label l and $R(h, \mathcal{E}_{train})$ is the *risk* associated with hypothesis h over $\mathcal{E}_{train}$, defined as the expectation of the loss function over the training set $\mathcal{E}_{train}$.

The effectiveness of the learnt classifier $\bar{h}$ is then assessed by determining $R(\bar{h}, \mathcal{E}_{test})$ on the test set $\mathcal{E}_{test}$.

In our learning setting, a hypothesis $h \in \mathcal{H}$ is a set of GCIs of the form

$$\langle C_1 \sqsubseteq P_1, \alpha_1 \rangle \ , \ldots \ , \langle C_h \sqsubseteq P_h, \alpha_h \rangle \tag{2}$$

$$@^{+}(P_1, \ldots, P_h) \sqsubseteq P \tag{3}$$

$$\langle D_1 \sqsubseteq N_1, \beta_1 \rangle \ , \ldots \ , \langle D_k \sqsubseteq N_k, \beta_k \rangle \tag{4}$$

$$@^{-}(N_1, \ldots, N_k) \sqsubseteq N \tag{5}$$

$$@(P, N) \sqsubseteq T \tag{6}$$

where each P_i, P, N_j, N are new atomic concept names not occurring in $\mathcal{K}$, and α_i, β_j are the confidence degree of the relative GCIs, $@^{+}, @^{-}, @$ are aggregation operators, and each C_i, D_j is a fuzzy $\mathcal{EL}(\mathbf{D})$ concept expression defined as (v is a boolean value)

$$\begin{array}{lcl} C & \longrightarrow & \top \mid A \mid \exists r.C \mid \exists s.\mathbf{d} \mid C_1 \sqcap C_2 \\ \mathbf{d} & \rightarrow & ls(a,b) \mid rs(a,b) \mid tri(a,b,c) \mid trz(a,b,c,d) \mid \ =_v \ . \end{array}$$

Informally, *(i)* each P_i 'rule' will tell us why an individual should be positive; *(ii)* then we aggregate the various degrees of positiveness via the aggregator operator $@^{+}$; *(iii)* on the other hand, each N_i 'rule' will tell us why an individual should be *not* positive; *(iv)* then we aggregate the various degrees of non-positiveness via the aggregator operator $@^{-}$. Typically, both $@^{+}$ and $@^{-}$ are the max operator; finally, *(v)* we use the last 'rule' to establish whether and individual is an instance of T or not (*viz.* is positive or not positive) by combining the degree of being positive or not via the $@$ operator. A simple choice for $@$ is the following and will be the one we will adopt:

> ($\star$) if the degree p of being positive is greater than the degree of being non-positive n then p, else 0.

Now, for $a \in \mathsf{I}_{\mathcal{K}}$, the *classification prediction value* $h(a)$ of a , T and $\mathcal{K}$ is defined as

$$h(a) = bed(\mathcal{K} \cup h, a{:}T) \ . \tag{7}$$

Remark 2 *Note that, as stated above, essentially a hypothesis is a sufficient condition for being an individual instance of a target concept to some degree. If $h(a) = 0$ then we say that a is not a positive instance of T, while if $h(a) > 0$ then a is a positive instance of T to degree $h(a)$. As a consequence, we will distinguish between positive and non-positive examples of T only. That is, negative examples and unlabelled examples are indistinguishable.*

Let us note that even if $\mathcal{K}$ is a crisp KB, the possible occurrence of fuzzy concrete domains in expressions of the form $\exists S.\mathbf{d}$ in a hypothesis may imply that not necessarily $h(a) \in \{0, 1\}$. A similar effect may also be induced by the aggregation operators.

Remark 3 *Clearly, the set of hypotheses by this syntax is potentially infinite due,* e.g. *to conjunction and the nesting of existential restrictions in the concept expressions. This set is made finite by imposing further restrictions on the generation process such as the maximal number of conjuncts and the maximal depth of existential nestings allowed.*

We conclude by saying that a hypothesis h *covers* (resp. θ-covers, for $\theta \in (0, 1]$) an individual $a \in \mathsf{I}_{\mathcal{K}}$ iff $h(a) > 0$ (resp. $h(a) \geq \theta$), and indicate with $Cov(h)$ (resp. $Cov_\theta(h)$) the set of covered (resp. θ-covered) individuals. Moreover, for a GCI $C \sqsubseteq T$, the *confidence degree* (also called *inclusion degree*) of $C \sqsubseteq T$ w.r.t. $\mathcal{K}$ and a set of positive individuals P, denoted $cf(C \sqsubseteq T, \mathcal{K}, P)$, is defined as

$$cf(C \sqsubseteq T, \mathcal{K}, P) = \frac{|C|^{P}_{\mathcal{K}}}{|C|^{\mathsf{I}_{\mathcal{K}}}_{\mathcal{K}}} \ , \tag{8}$$

which is the proportion of positive individuals covered by C w.r.t. the individuals covered by C. Clearly, $cf(C \sqsubseteq T, \mathcal{K}, P) \in [0, 1]$ and the closer the confidence is to 1 the 'more precise' is $C \sqsubseteq T$, in the sense the less it covers non-positive individuals. In addition, the *support* of $C \sqsubseteq T$ w.r.t. $\mathcal{K}$ and a set of individuals I, denoted $supp(C \sqsubseteq T, \mathcal{K}, \mathsf{I})$, is defined as

$$supp(C \sqsubseteq T, \mathcal{K}, \mathsf{I}) = \frac{|C|^{\mathsf{I}}_{\mathcal{K}}}{|\mathsf{I}|} \tag{9}$$

3.2 Conceptual Illustration of the Learning Method.

Before presenting our learning algorithm, we will first conceptually illustrate its principle by relying on Figure 4.

At the beginning, let us consider the sets of all individuals, the positive, the negative and the unlabelled individuals, respectively the sets $\mathsf{I}_{\mathcal{K}}, \mathsf{I}_{\mathcal{E}^+}, \mathsf{I}_{\mathcal{E}^-}$ and $\mathsf{I}_{\mathcal{E}^u}$, as depicted in Figure 4 (a).

At the first stage, the P-stage, we consider the entire training set $\mathcal{E}$ and try to maximise the covering of positive individuals, while minimising the covering of negative individuals. Specifically, let us assume that we have learnt a hypothesis h_P (a set of rules) with a covering $Cov_{\theta_P}(h_P)$, as depicted in Figure 4 (b). Here, the value θ_P acts as a confidence threshold for the learnt rules in hypothesis h_P. Note that $Cov_{\theta_P}(h_P)$ has to contain positive individuals, *i.e.* $Cov_{\theta_P}(h_P) \cap \mathsf{I}_{\mathcal{E}^+} \neq \emptyset$, but may also contain negative and unlabelled individuals. We call the individuals in $TP = Cov_{\theta_P}(h_P) \cap \mathsf{I}_{\mathcal{E}^+}$ *true positives*, while call those in $FP = Cov_{\theta_P}(h_P) \setminus \mathsf{I}_{\mathcal{E}^+}$ *false positives*, *i.e.* a false positive is an individual that is erroneously classified by h_P as an instance of the target class T, while in fact it is not (it might be an unlabelled or a negative example). This phase ends with a set of rules of the form (2) – (3).

Now, in the next stage, the N-stage, with the aim to increase the effectiveness of the classifiers, we would like to remove as many as possible false positives in FP, while avoiding removing, if possible, any of the true positives in TP. To do so, we set-up a new learning problem in which the new target class is FP, where the negatives individuals are those in TP and the positives are those in FP. Of course, the N-stage applies only if $FP \neq \emptyset$. The setup of the N-stage is depicted in Figure 4 (c). Specifically, let us assume that we have learnt now a hypothesis h_N with a covering $Cov_{\theta_N}(h_N)$, as depicted in Figure 4 (d). Note that we may have another parameter θ_N acting as a confidence threshold for the learnt rules in hypothesis h_N. This phase ends with a set of rules of the form (4) – (5).

So, in general, at the end of the two stages, the situation may be as depicted in Figure 4 (e). However, in practice one may want likely to impose that none of the initial positive individuals are covered by h_N and, thus, none of the true positives in TP will be removed by h_N.

Eventually, we aggregate the P-rules and N-rules via ($\star$). This latter step ends with the rule of the form (6). At the end of this two-stage process, we aim at to have captured most of the positive individuals of the target class, with few of the negative and unlabelled individuals (false positives).

3.3 The Learning Algorithm PN-OWL

We now present our two-stage learning algorithm, called PN-OWL, that we have conceptually illustrated in the section before. Essentially, at the P-stage (resp. N-stage) our algorithm invokes

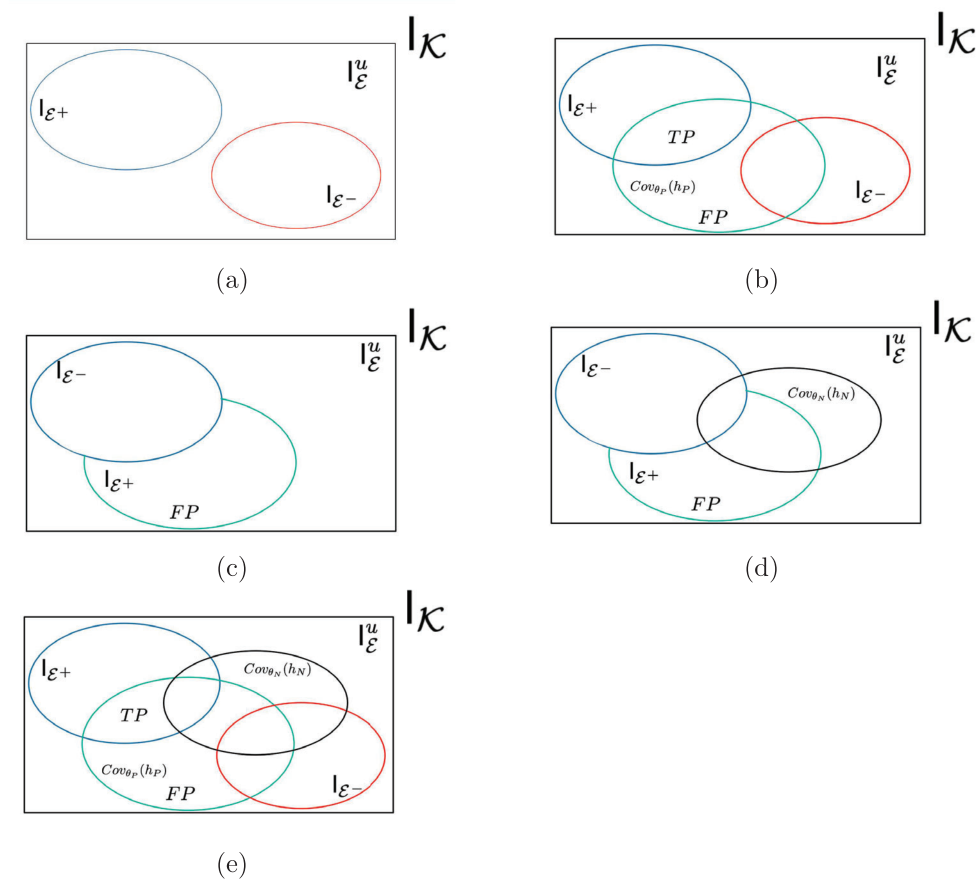

Figure 4: How PN-OWL works. (a) Original training set; (b) Coverage $Cov_{\theta_P}(h_P)$ w.r.t. learnt hypothesis h_P after the P-stage; (c) Starting dataset for N-stage: the new target class are the false positives FP of the P-stage, while the negative individuals are the initial positives; (d) Coverage $Cov_{\theta_N}(h_N)$ w.r.t. learnt hypothesis h_N after the N-stage; (e) Final scenario.

a learner, called *stage learner*, that generates a set h_P (resp. h_N) of fuzzy $\mathcal{EL}(\mathbf{D})$ candidate GCIs that has, respectively, the form

$$h_P = \{\langle C_1 \sqsubseteq T, \alpha_1\rangle, \ldots, \langle C_h \sqsubseteq T, \alpha_h\rangle\} \quad (10)$$

$$h_N = \{\langle D_1 \sqsubseteq FP, \beta_1\rangle, \ldots, \langle D_k \sqsubseteq FP, \beta_k\rangle\} \quad (11)$$

called *stage hypothesis*. In the following, we indicate with p_i the fuzzy GCI $\langle C_i \sqsubseteq T, \alpha_i\rangle$, while denote with n_j the fuzzy GCI $\langle D_j \sqsubseteq FP, \beta_j\rangle$. The rules in h_P (resp. h_N) will then be aggregated using the max aggregation operator.

The stage hypotheses are then combined into a final hypothesis for the target class T using the aggregation operator $(\star)$.

As stage learner we will use a modified version of the fuzzy Foil-$\mathcal{DL}$ [50, 51, 53] learner that will be described in Section 3.4.

Then, the PN-OWL algorithm is shown in Algorithm 1. Note that the P-stage are the steps

1-5, while the N-stage are the steps 15-19 in which at step 19 we invoke the stage learner trying to cover as many as false positives as possible. The remaining steps deal with the construction of the final classifier ensemble as per Eqs. (2)-(6).

Eventually, for an individual $a \in \mathsf{I}_{\mathcal{K}}$, the *classification prediction value of* PN-OWL for individual a is $h(a)$, where h is the returned hypothesis of PN-OWL. Moreover, we say that PN-OWL *classifies* a as instance of target class T if $h(a) > 0$.

Algorithm 1 PN-OWL

Input: KB $\mathcal{K}$, training set $\mathcal{E}$, target concept name T, confidence thresholds $\theta_P, \theta_N \in [0, 1]$, non-positive coverage percentages $\eta_P, \eta_N \in [0, 1]$
Output: Hypothesis h as by Eqs. (2)-(6).
1: // P-stage
2: $Pos \leftarrow \mathsf{I}_{\mathcal{E}+}$;
3: $Neg \leftarrow \mathsf{I}_{\mathcal{E}-}$;
4: $U \leftarrow \mathsf{I}_{\mathcal{K}} \setminus (Pos \cup Neg)$;
5: $h_P \leftarrow$ FuzzyStageLearner($\mathcal{K}$, T, Pos, Neg, U, θ_P, η_P); ▷ P-Stage hypothesis h_P, *i.e.* set of axioms $\langle C_i \sqsubseteq T, \alpha_i \rangle$
6: **if** $h_P = \emptyset$ **then return** $\emptyset$; ▷ Nothing learnt, exit
7: $Cov \leftarrow Cov_{\theta_P}(h_P)$; ▷ P-stage Coverage
8: $TP \leftarrow Cov_{\theta_P}(h_P) \cap \mathsf{I}_{\mathcal{E}+}$; ▷ True positives
9: $FP \leftarrow Cov_{\theta_P}(h_P) \setminus \mathsf{I}_{\mathcal{E}+}$; ▷ False positives
10: // Start building classifier h
11: $h \leftarrow \{\langle C_i \sqsubseteq P_i, \alpha_i \rangle, \mid \langle C_i \sqsubseteq T, \alpha_i \rangle \in h_P, P_i \text{ new } \}$; ▷ As per Eq. 2
12: **if** $FP = \emptyset$ **then** ▷ No N-stage, exit with aggregated h_P
13: $\quad h \leftarrow h \cup \{@^+(P_1, \ldots, P_h) \sqsubseteq T\}$; ▷ No need of new P in Eq. 3
14: $\quad$ **return** h;
15: // N-stage
16: $Pos \leftarrow FP$;
17: $Neg \leftarrow \mathsf{I}_{\mathcal{E}+}$;
18: $U \leftarrow \mathsf{I}_{\mathcal{K}} \setminus (Pos \cup Neg)$;
19: $h_N \leftarrow$ FuzzyStageLearner($\mathcal{K}$, FP, Pos, Neg, U, θ_N, η_N); ▷ N-Stage hypothesis h_N, *i.e.* set of axioms $\langle D_j \sqsubseteq FP, \beta_j \rangle$
20: // Build final classifier ensemble h
21: **if** $h_N = \emptyset$ **then** ▷ No learning in N-stage, return aggregated h_P
22: $\quad h \leftarrow h \cup \{@^+(P_1, \ldots, P_h) \sqsubseteq T\}$; ▷ No need of new P in Eq. 3
23: $\quad$ **return** h;
24: $h \leftarrow h \cup \{@^+(P_1, \ldots, P_h) \sqsubseteq P \mid P \text{ new } \}$; ▷ As per Eq. 3
25: $h \leftarrow h \cup \{\langle D_j \sqsubseteq N_j, \beta_j \rangle, \mid \langle D_j \sqsubseteq FP, \beta_j \rangle \in h_N, N_j \text{ new } \}$; ▷ As per Eq. 4
26: $h \leftarrow h \cup \{@^-(N_1, \ldots, N_k) \sqsubseteq N \mid N \text{ new } \}$; ▷ As per Eq. 5
27: $h \leftarrow h \cup \{@(P, N) \sqsubseteq T\}$; ▷ As per Eq. 6
28: **return** h;

3.4 The Stage Learner pnFoil-$\mathcal{DL}$

As stage learner we will use fuzzy Foil-$\mathcal{DL}$ [20, 50, 51, 53], which however will be modified to adapt to our specific setting (see Algorithm 2), which we call pnFoil-$\mathcal{DL}$. That is, the procedure invocations FuzzyStageLearner in lines 5 and 19 of the PN-OWL algorithm are indeed calls to pnFoil-$\mathcal{DL}$.

Essentially, pnFOIL-$\mathcal{DL}$ carries on inducing GCIs until as many as positive examples are covered or nothing new can be learnt. When an axiom is induced (see step 4 in Algorithm 2), the positive examples still to be covered are updated (steps 10 and 11).

In order to induce an axiom (step 4), LEARN-ONE-AXIOM is invoked (see Algorithm 3), which in general terms operates as follows:

1. start from concept $\top$;
2. apply a refinement operator to find more specific fuzzy $\mathcal{EL}(\mathbf{D})$ concept description candidates;
3. exploit a scoring function to choose the best candidate;
4. re-apply the refinement operator until a good candidate is found;
5. iterate the whole procedure until a satisfactory coverage of the positive examples is achieved.

Algorithm 2 pnFOIL-$\mathcal{DL}$

Input: KB $\mathcal{K}$, target concept name T, a set P (resp. N and U) of positive (resp. negative and unlabelled) examples, confidence threshold $\theta \in [0,1]$, non-positive coverage percentage $\eta \in [0,1]$

Output: A hypothesis, *i.e.* a set $h = \{\langle C_i \sqsubseteq T, \delta_i \rangle | 1 \leq i \leq k\}$ of fuzzy $\mathcal{EL}(\mathbf{D})$ GCIs

1: $h \leftarrow \emptyset, Pos \leftarrow P, \phi \leftarrow \top \sqsubseteq T$;
2: //Loop until no improvement
3: **while** ($Pos \neq \emptyset$) **and** ($\phi \neq$ **null**) **do**
4: $\quad \phi \leftarrow$ LEARN-ONE-AXIOM$(\mathcal{K}, T, Pos, P, N, U, \theta, \eta)$; $\triangleright$ Learn one fuzzy $\mathcal{EL}(\mathbf{D})$ GCI of the form $C \sqsubseteq T$
5: $\quad$ **if** $\phi \in h$ **then** $\triangleright$ axiom already learnt
6: $\quad\quad \phi \leftarrow$ **null**;
7: $\quad$ **if** $\phi \neq$ **null then**
8: $\quad\quad \delta \leftarrow cf(\phi, \mathcal{K}, P)$; $\triangleright$ Compute confidence of ϕ
9: $\quad\quad h \leftarrow h \cup \{\langle \phi, \delta \rangle\}$; $\triangleright$ Update hypothesis
10: $\quad\quad Pos_\phi \leftarrow Pos \cap Cov(\langle \phi, \delta \rangle)$; $\triangleright$ Positives covered by $\langle \phi, \delta \rangle$)
11: $\quad\quad Pos \leftarrow Pos \setminus Pos_\phi$; $\triangleright$ Update positives still to be covered
12: **return** h;

We now detail the steps of LEARN-ONE-AXIOM (Algorithm 3).

Computing fuzzy datatypes. For a numerical datatype s, we consider *equal width triangular partitions* of values $V_s = \{v \mid \mathcal{K} \models a{:}\exists s. =_v\}$ into a finite number of fuzzy sets (3, 5 or 7 sets), which is identical to [50, 53, 87] (see, *e.g.* Fig. 3). We additionally also consider the use of the c-means fuzzy clustering algorithm over V_s, where the fuzzy membership function is a triangular function build around the centroid of a cluster [20, 50, 53, 87].

The refinement operator. The refinement operator we employ is essentially the same as in [20, 50, 51, 57, 87]. Specifically, it takes as input a concept C and generates new, more specific concept description candidates D (*i.e.* , $\mathcal{K} \models D \sqsubseteq C$). For the sake of completeness, we recap the refinement operator here. Let $\mathcal{K}$ be a knowledge base, $\mathbf{A}_\mathcal{K}$ be the set of all atomic concepts in $\mathcal{K}$, $\mathbf{R}_\mathcal{K}$ the set of all object properties in $\mathcal{K}$, $\mathbf{S}_\mathcal{K}$ the set of all numeric datatype properties in $\mathcal{K}$, $\mathbf{B}_\mathcal{K}$ the set of all boolean datatype properties in $\mathcal{K}$ and $\mathcal{D}$ a set of (fuzzy) datatypes. The refinement operator ρ is shown in Table 2.

Table 2: Downward Refinement Operator.

$$\rho(C) = \begin{cases} \mathbf{A}_{\mathcal{K}} \cup \{\exists r.\top \mid r \in \mathbf{R}_{\mathcal{K}}\} \cup \{\exists s.d \mid s \in \mathbf{S}_{\mathcal{K}}, d \in \mathcal{D}\} \cup & \\ \quad \{\exists s. =_b, \mid s \in \mathbf{B}_{\mathcal{K}}, b \in \{\mathbf{true}, \mathbf{false}\}\} & \text{if} \quad C = \top \\ \{A' \mid A' \in \mathbf{A}_{\mathcal{K}}, \mathcal{K} \models A' \sqsubseteq A\} \cup & \\ \quad \{A \sqcap A'' \mid A'' \in \rho(\top)\} & \text{if} \quad C = A \\ \{\exists r.D' \mid D' \in \rho(D)\} \cup \{(\exists r.D) \sqcap D'' \mid D'' \in \rho(\top)\} & \text{if} \quad C = \exists r.D, r \in \mathbf{R}_{\mathcal{K}} \\ \{(\exists s.d) \sqcap D \mid D \in \rho(\top)\} & \text{if} \quad C = \exists s.d, s \in \mathbf{S}_{\mathcal{K}}, d \in \mathcal{D} \\ \{(\exists s. =_b) \sqcap D \mid D \in \rho(\top)\} & \text{if} \quad C = \exists s. =_b, s \in \mathbf{B}_{\mathcal{K}}, \\ & \quad b \in \{\mathbf{true}, \mathbf{false}\} \\ \{C_1 \sqcap ... \sqcap C'_i \sqcap ... \sqcap C_n \mid i = 1, ..., n, C'_i \in \rho(C_i)\} & \text{if} \quad C = C_1 \sqcap ... \sqcap C_n \end{cases}$$

The scoring function. The scoring function we use to assign a score to each candidate hypothesis is essentially a *gain* function, like to the one employed in [20, 50, 51, 57, 87], and it implements an information-theoretic criterion for selecting the best candidate at each refinement step. Specifically, given a fuzzy $\mathcal{EL}(\mathbf{D})$ GCI ϕ of the form $C \sqsubseteq T$ chosen at the previous step, a KB $\mathcal{K}$, a set of positive examples Pos still to be covered and a candidate fuzzy $\mathcal{EL}(\mathbf{D})$ GCI ϕ' of the form $C' \sqsubseteq T$, then

$$gain(\phi', \phi, \mathcal{K}, Pos) = p * (log_2(cf(\phi', \mathcal{K}, Pos)) - log_2(cf(\phi, \mathcal{K}, Pos))) \ , \tag{12}$$

where $p = |C' \sqcap C|^{Pos}_{\mathcal{K}}$ is the fuzzy cardinality of positive examples in Pos covered by ϕ that are still covered by ϕ'.

Please note that in Eq. 12, the confidence degrees are calculated w.r.t. the positive examples still to be covered (Pos). In this way, Learn-One-Axiom is somewhat guided towards positives not yet covered so far by pnFoil-$\mathcal{DL}$. Note also that the gain is positive if the confidence degree increases.

Stop criterion. Learn-One-Axiom stops when the confidence degree is above a given threshold $\theta \in [0, 1]$ and the non-positive coverage percentage is below $\eta \in [0, 1]$, or no GCI can be learnt anymore.

The Learn-One-Axiom algorithm. The Learn-One-Axiom algorithm just like defined in Algorithm 3: steps 1 - 3 are simple initialisation steps. Please note here that NP are the non-positives in accordance with Remark 2, which states that we will distinguish among positives and non-positives only (*cf.* also step. 18, where the non-positive coverage percentage is used). Steps 5-21 are the main loop from which we may exit in case the stopping criterion is satisfied, in step 8 we determine all new refinements, which then are scored in steps 10-15 in order to determine the one with the best gain. At the end of the algorithm, once we exit from the main loop, the best found GCI is returned (step 22).

Remark 4 *pn*Foil-$\mathcal{DL}$ *also allows to use a backtracking mechanism (step 19), which, for ease of presentation, we omit to include. The mechanism is the same as for the p*Foil-$\mathcal{DL}$*-learnOneAxiom described in [87, Algorithm 3]. Essentially, a stack of top-k refinements is maintained, ranked in decreasing order of the confidence degree from which we pop the next best refinement (if the stack is not empty) in case no improvement has occurred.* C_{best} *becomes the popped-up refinement.*

Algorithm 3 LEARN-ONE-AXIOM

Input: KB $\mathcal{K}$, target concept name T, set Pos of positive examples still to be covered, training sets P, N, U of positive, negative and unlabelled examples, respectively, confidence threshold $\theta \in [0,1]$, non-positive coverage percentage $\eta \in [0,1]$

Output: A fuzzy $\mathcal{EL}(\mathbf{D})$ GCI of the form $C \sqsubseteq T$

$NP \leftarrow N \cup U$; ▷ Note: NP are the non-positives
$C \leftarrow \top$; ▷ Start from $\top$
$\phi \leftarrow C \sqsubseteq T$;
//Loop until no improvement
while $C \neq$ **null do**
  $C_{best} \leftarrow C$;
  $maxgain \leftarrow 0$;
  $\mathcal{C} \leftarrow \rho(C)$; ▷ Compute all refinements of C
  // Compute the score of the refinements and select the best one
  for all $C' \in \mathcal{C}$ **do**
    $\phi' \leftarrow C' \sqsubseteq T$;
    $gain \leftarrow gain(\phi', \phi, \mathcal{K}, Pos)$;
    if $(gain > maxgain)$ **then**
      $maxgain \leftarrow gain$;
      $C_{best} \leftarrow C'$;
  if $C_{best} = C$ **then** ▷ No improvement
    //Stop if confidence degree above threshold or non-positive coverage below threshold
    if $(cf(C_{best} \sqsubseteq T, \mathcal{K}, P) \geq \theta)$ **and** $supp(C_{best} \sqsubseteq T, \mathcal{K}, NP) \leq \eta)$ **then break**;
    // Manage backtrack here, if foreseen
  $C \leftarrow C_{best}$;
  $\phi \leftarrow C \sqsubseteq T$;
return ϕ;

4 Evaluation

We have implemented the algorithm within the *FuzzyDL-Learner*[5] system and have evaluated it over a set of (crisp) OWL ontologies.

Datasets. Several OWL ontologies from different domains have been selected as illustrated in Table 3. In it, we report the DL the ontology refers to, the number of concept/class names, object properties, datatype properties and individuals in the ontology. For each ontology $\mathcal{K}$ we indicate also the number $|\mathcal{E}^+|$ of positive examples. All others are non-positive and we set $\mathcal{E}^- = \overline{\mathcal{E}^+} = \mathsf{I}_\mathcal{K} \setminus \mathsf{I}_{\mathcal{E}^+}$. The ontologies `Iris`, `Wine`, `Wine Quality` and `Yeast` are built from the well-known *UC Irvine Machine Learning Repository* (UCIMLR) [27] and have been transformed from the CSV format, provided by that repository, into OWL ontologies according to the procedure described in [20]. In the `Wine Quality` ontology, the `quality` attribute has been removed as the positive examples (the `GoodRedWines`) are those having "quality" greater than or equal to 7.

All other ontologies, except `malware`, belong to the well-known SML-Bench dataset [91].[6] The `malware` ontology has been described in [88, 89].

For completeness, in Appendix A, a succinct description of what the ontologies are about is provided.

[5]Data and implementation `http://www.umbertostraccia.it/cs/software/FuzzyDL-Learner/`.
[6]See also, `https://github.com/SmartDataAnalytics/SML-Bench`

Table 3: Facts about the ontologies of the evaluation.

ontology	DL	class.	obj. prop.	data. prop.	ind.	target T	pos
NTN	$\mathcal{SHOIN}(\mathcal{D})$	51	29	9	723	ToLearn_Woman	46
Lymphography	$\mathcal{ALC}$	50	0	0	148	ToLearn	81
Mammographic	$\mathcal{ALC}(\mathcal{D})$	20	3	2	975	ToLearn	445
Malware	$\mathcal{ALH}(\mathcal{D})$	192	6	10	5669	malware	500
Iris	$\mathcal{ALEHF}(\mathcal{D})$	4	0	5	150	Iris-versicolor Iris-virginica	50 50
Wine	$\mathcal{ALEHF}(\mathcal{D})$	3	0	13	178	1 2 3	59 71 48
Wine Quality	$\mathcal{ALEHF}(\mathcal{D})$	7	0	11	6497	GoodRedWine	217
Yeast	$\mathcal{ALEHF}(\mathcal{D})$	11	0	8	1462	CYT	444

Remark 5 *While evaluating ontology-based learning algorithms is untypical on numerical datatype properties,*[7] *we believe it is interesting to do so as an important ingredient of our algorithm is the use of fuzzy concrete datatype properties to improve the human understandability of the classification decision process.*

Remark 6 *We leave it for future work to look at* e.g. *methods to learn from the training data a threshold $0 \leq \tau_p \leq 1$ such that h predicts individual a to be a positive example if $h(a) > \tau_p$. However, in this paper, we will always have $\tau_p = 0$.*

More generally, unlike we do now, if we would like to distinguish the negative examples from the unlabelled ones, we may well learn a classifier h^- for negative examples and then define a decision method that predicts an individual a to be a positive (resp. negative) example based on the prediction value $h(a)$ (resp. $h^-(a)$) of a being a positive *(resp. negative) example. That is, depending on the pair $\langle h(a), h^-(a)\rangle$, one may then define e decision criteria whether a is a positive or negative example, or just leave the prediction as* unknown *if there is not enough evidence of being one of the two.*

Measures. We considered the following effectiveness measures (see also [87, 20]), which, for the sake of completeness, we recap here. Specifically, consider a learnt classifier h and let us assume to have added it to the KB $\mathcal{K}$. In our setting, we always have the condition that if the classifier prediction value $h(a)$ of an individual a is non-zero then the learner classifies a as an instance of T, *i.e.* h predicts a to be a positive example iff $h(a) > 0$.

In line with what we have said above, as all individuals are either positive or non-positive, we will consider the following measures, all of which are based on crisp cardinality (see also Eq. 1).

True Positives: denoted TP, is defined as the number of instances of T that are positive

$$TP = \lceil T \rceil_{\mathcal{K}}^{\mathsf{I}_{\mathcal{E}^+}} \tag{13}$$

False Positives: denoted FP, is defined as the number of instances of T that are not positive

$$FP = \lceil T \rceil_{\mathcal{K}}^{\mathsf{I}_{\overline{\mathcal{E}^+}}} \tag{14}$$

Precision/Confidence: denoted P, is defined as the fraction of true positives w.r.t. the covered examples of h

$$P = \frac{TP}{\lceil T \rceil_{\mathcal{K}}^{\mathsf{I}_{\mathcal{E}}}} \tag{15}$$

[7] To the best of our knowledge, we are unaware of any evaluation of ontology-based methods on those data sets.

Recall: denoted R, is defined as fraction of true positives w.r.t. all positives

$$R = \frac{TP}{|\mathsf{I}_{\mathcal{E}^+}|} , \tag{16}$$

$F1$-score: denoted $F1$, is defined as

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} . \tag{17}$$

For each parameter configuration, a stratified k-fold cross validation design[8] was adopted (specifically, $k = 5$) to determine the macro average of the above described performance measures. In all tests, we have that $\mathsf{I}_{\mathcal{E}} = \mathsf{I}_{\mathcal{K}}$ and that, of course, there is at least one positive example in each fold. For each fold, during the training phase, we remove all assertions involving test examples from the ontology, and, thus, restrict the training phase to training examples only.

All configuration parameters for the best runs are available from the downloadable data, which we do not report here. Some of the salient parameters, used within our algorithm, are reported in Table 4.

Table 4: Some salient parameters of the PN-OWL algorithm.

θ_P	confidence threshold for positive rules of P-stage
θ_N	confidence threshold for negative rules of N-stage
η_P	non-positive coverage percentage threshold for positive rules of P-stage
η_N	non-positive coverage percentage threshold for negative rules of N-stage
c_P	maximal number of conjuncts for positive rules of P-stage
c_N	maximal number of conjuncts for negative rules of P-stage
d_P	maximal role depth for positive rules of P-stage
d_N	maximal role depth for negative rules of P-stage

A typical parameter setup is as follows, but may vary depending on the ontology and may be subject of a search for the optimal setting.

P-stage. $c_P = 5, d_P = 1, \theta_P = 0.1, \eta_P = 1.0$

N-stage. $c_N = 10, d_N = 1, \theta_N = 0.3, \eta_N = 0.2$

Let us briefly comment them. During the P-stage, we would like to increase recall, that is the percentage of covered positives w.r.t. all positives. To this end, we choose a low positive rule confidence threshold θ_P and high non-positive coverage percentage threshold η_P. In the N-stage however, we want to be more precise in removing the false positives in order to avoid removing true positives of the P-Stage. Therefore, we increase the confidence threshold θ_N, lower the non-positive coverage percentage threshold η_N and increase the number of maximal conjuncts c_N. The maximal role depth is determined manually a priori by inspecting the ontology.

For $@^+, @^-$ (resp. $@$) we used max (resp. $(\star)$), and for concept conjunction $\sqcap$ (resp. GCI operator $\sqsubseteq$) we used the t-norm min (resp. the Łukasiewicz implication). These could well be another set of parameters to be optimised. However, the parameter space is already quite large,

[8]Stratification means here that each fold contains roughly the same proportions of positive and non-positive instances of the target class.

so we fixed these logical operators as specified.[9] Concerning other parameter settings, we also varied the number of fuzzy sets $(3, 5$ or $7)$. For c-means, we fixed the hyper-parameter to the default $m = 2$, the threshold to $\epsilon = 0.05$ and the number of maximum iterations to 100.

As baseline, we consider Fuzzy FOIL-$\mathcal{DL}$ [50, 51, 53, 20], with best parameter setup as specified in [20]. Essentially, Fuzzy FOIL-$\mathcal{DL}$, is as PN-OWL, except that it stops after the P-stage and, thus, is as PN-OWL in which the negative set of rules h_N is by definition empty (*cf.* lines 21-23 of PN-OWL algorithm). This allows us to appreciate the added value (if any) in terms of effectiveness of the N-stage phase.

The results are reported in Table 5. For the UCIMLR datasets, in case of multiple targets, the average of the measures has been considered.

Example 4.1 *We provide here examples of learnt rules (in Fuzzy OWL syntax) via* PN-OWL *applied to the* `Mammographic` *ontology. The first one is one of the learnt rules during the P-stage, while the second one is one of the learnt rules during the N-Stage. In the latter case,* `FALSEP_ToLearn` *denotes the class of false positives covered by rules learnt during the P-stage. The number associated to a rule is its confidence/precision. We also report the specification of some learnt fuzzy sets via fuzzy c-means.*

```
(implies (and (some hasDensity low)
              (some hasShape irregular)
              (some hasAge hasAge_veryHigh)
              (some hasBiRads hasBiRads_high))
   ToLearn 0.965068)

(implies (and (some hasDensity low)
              (hasMargin some microlobulated)
              (hasShape some oval)
              (hasBiRads some hasBiRads_medium))
   FALSEP_ToLearn 0.75)

(define-fuzzy-concept hasBiRads_medium    triangular(1,6,2.780,3.997,5.022))
(define-fuzzy-concept hasBiRads_high      right-shoulder(1,6,3.997,5.022))
(define-fuzzy-concept hasAge_veryHigh     right-shoulder(1,6,62.793,71.882))
```

[9] A run with fixed parameters, *e.g.* on the `malware` ontology, may already take up to 4 days of computation time.

Table 5: Results table. The measures are the macro average over the 5 folds w.r.t. the test set.

Dataset	Algorithm	Precision	Recall	F1	% Improvement
`NTN`	Fuzzy DL-FOIL	0.661	0.513	0.548	**80.47%**
	PN-OWL	**1.000**	**0.980**	**0.989**	
`Lymphography`	Fuzzy DL-FOIL	**0.861**	**0.851**	**0.855**	**-2.57%**
	PN-OWL	0.836	0.841	0.833	
`Mammographic`	Fuzzy DL-FOIL	0.737	0.692	0.710	**11.27%**
	PN-OWL	**0.746**	**0.831**	**0.790**	
`Malware`	Fuzzy DL-FOIL	0.623	**0.830**	0.704	**5.06%**
	PN-OWL	**0.701**	0.818	**0.740**	
`Iris`	Fuzzy DL-FOIL	0.886	0.910	0.890	**4.16%**
	PN-OWL	**0.949**	0.910	**0.927**	
`Wine`	Fuzzy DL-FOIL	0.884	**0.971**	0.895	**0.98%**
	PN-OWL	**0.933**	0.904	**0.914**	
`Wine Quality`	Fuzzy DL-FOIL	0.227	**0.917**	0.363	**27.93%**
	PN-OWL	**0.365**	0.659	**0.464**	
`YEAST`	Fuzzy DL-FOIL	0.427	0.746	0.540	**4.37%**
	PN-OWL	**0.432**	**0.815**	**0.564**	

Discussion. In Table 5, the last column reports the improvement of PN-OWL relative to the measure $F1$ (see Eq. 17), over our baseline Fuzzy FOIL-$\mathcal{DL}$. Overall, PN-OWL performs better than Fuzzy FOIL-$\mathcal{DL}$ (with the exception of `Lymphography`) and in some cases the improvement is particularly high, such as for `NTN, Mammographic` and `Wine Quality`.

Essentially, for PN-OWL we were able to find a better compromise between precision and recall than for FOIL-$\mathcal{DL}$. In particular, we were able to increase precision confirming our conjecture that indeed the N-stage is able to remove the false positives.

Concerning `Lymphography`, we were unable to replicate the results of Fuzzy FOIL-$\mathcal{DL}$ in [20], for which we get now an F1 measure of 0.805 in place of 0.855. The difference lies in few miss-classified examples. We also noted that in this case PN-OWL achieves $F1 = 1.0$ during the training phase, which may suggest an over-fitting problem.

Last but not least, let us mention that PN-OWL (so does Fuzzy FOIL-$\mathcal{DL}$) does definitely not yet behave well on the `Wine Quality` and `Yeast` datasets, which will be the subject of further investigation.

The overall lesson learnt with PN-OWL is that indeed the N-stage may provide a non negligible contribution to improve effectiveness of the classification process, provided one may find the appropriate balance among precision and recall. Unfortunately, searching the parameter space of PN-OWL for an optimum is quite time consuming and a brute-force approach may likely not be feasible (at least not with our computational resources at hand). In fact, we proceeded one run per time, and by analysing the results tried to figure out whether and how to change some of the parameters in Table 4 to increase recall and/or precision. On the other-hand, optimising FOIL-$\mathcal{DL}$ is much easier as it has half of the parameters of PN-OWL.

5 Related Work

Concept inclusion axiom learning in DLs is essentially inspired by statistical relational learning, where classification rules are (possibly weighted) Horn clause theories (see *e.g.* [69, 70]), and various methods have been proposed in the DL context so far (see *e.g.* [54, 24, 71]). The general idea consists in the exploration of the search space of potential concept descriptions that cover the available training examples using so-called refinement operators (see, *e.g.* [7, 22, 45, 46, 47, 48, 49]). The goal is then to learn a concept description of the underlying DL language covering (possibly) all the provided positive examples and (possibly) not covering any of the provided negative examples. The fuzzy case (see [50, 53, 87, 20]) is a natural extension relying on fuzzy DLs [10, 86] and fuzzy ILP (see *e.g.* [82]) instead.

As already mentioned, our two-stage algorithm is conceptually inspired by PN-rule [2, 3, 40, 41] consisting of a P-stage in which positive rules (called P-rules) are learnt to cover as many as possible instances of a target class and an N-stage in which negative rules (called N-rules) are learnt to remove most of the non-positive examples covered by the P-stage. The two rule sets are then used to build up a decision method to classify an object being instance of the target class or not [2, 3, 40, 41]. It is worth noting that what differentiates this method from all others is its second stage. The main differences of PN-OWL w.r.t. PN-rule are: *(i)* PN-rule operates with *tabular data* only, *i.e.* the data consists of attribute value pairs (A, v), while we are in the context of OWL ontologies.[10]; PN-rules are of the form $cond \rightarrow T$, where the condition $cond$ is of the form $(A \in [l, h])$ or $(A \notin [l, h])$ for continuous attribute A.[11], while we have, conjunction of conditions in the rule body and each condition may be fuzzy, besides being either a class name or a restriction on attributes (attributes may be also nested); and *(iii)* PN-rule considers a completely different rule scoring and combination strategy than we use in PN-OWL. The latter can be represented in Fuzzy OWL 2 [12, 13], while for the former we conjecture it cannot: so, we left this option out as a fuzzy DL reasoner would not be able to reason with those types of rules.

Other closely related works are [30, 28, 36, 35, 50, 53, 87]. In fact, [30, 28, 36, 78] can be seen as an adaption to the DL case of the the well-known FOIL-algorithm, while [50, 53] that stem essentially from [51, 52, 55, 56, 57, 58], propose *fuzzy* FOIL-like algorithms instead, and are inspired by fuzzy ILP variants such as [26, 82, 84].[12] Let us note that [50, 56] consider the weaker hypothesis representation language DL-Lite [5], while here we rely on an aggregation of fuzzy $\mathcal{EL}(\mathbf{D})$ inclusion axioms. Fuzzy $\mathcal{EL}(\mathbf{D})$ has also been considered in [87], which however differs from [50, 53] by the fact that a (fuzzy) probabilistic ensemble evaluation of the fuzzy concept description candidates has been considered.[13] Let us recap that, to our opinion, fuzzy $\mathcal{EL}(\mathbf{D})$ concept expressions are appealing as they can straightforwardly be translated into natural language and, thus, contribute to the explainability aspect of the induced classifier.

Discrete boosting has been considered in [35] that also shows how to derive a weak learner (called WDLF) from conventional learners using some sort of random downward refinement operator covering at least a positive example and yielding a minimal score fixed with a threshold. Related to this work is [20] that deals with fuzziness in the hypothesis language and a real-valued variant of AdaBoost and differentiates from the previous one by using a descent-like gradient algorithm to search for the best alternative. Notably, this also deviates from 'fuzzy' rule learning AdaBoost variants, such as [25, 66, 68, 81, 92] in which the weak learner is required to generate the whole rules' search space beforehand the selection of the best current alternative. Such an

[10]Tabular data can easily be mapped into OWL ontologies as illustrated in [20].
[11]If A is categorical then obviously *cond* is either of the form $A = v$ or $A \neq v$.
[12]See, *e.g.* [23], for an overview on fuzzy rule learning methods.
[13]Also, to the best of our knowledge, concrete datatypes were not addressed in the evaluation.

approach is essentially unfeasible in the OWL case due to the size of the search space.

In [39] a method is described that can learn fuzzy OWL DL concept equivalence axioms from FuzzyOWL 2 ontologies, by interfacing with the *fuzzyDL* reasoner [15]. The candidate concept expressions are provided by the underlying DL-Learner [44, 18, 19] system. However, it has been tested only on a toy ontology so far. Moreover, let us mention [42] that is based on an ad-hoc translation of fuzzy Łukasiewicz $\mathcal{ALC}$ DL constructs into fuzzy *Logic Programming* (fuzzy LP) and uses a conventional ILP method to learn rules. Unfortunately, the method is not sound as it has been shown that the mapping from fuzzy DLs to LP is incomplete [64] and entailment in Łukasiewicz $\mathcal{ALC}$ is undecidable [21]. To be more precise, undecidability holds already for $\mathcal{EL}$ under the infinitely valued Łukasiewicz semantics [17].[14]

While it is not our aim to provide an extensive overview about learning w.r.t. ontologies literature, we nevertheless recap here that there are also alternative methods to what we present here, but are related only to the extent that they deal with concept description induction in the context of DLs. So, *e.g.* , the series of works [32, 33, 75, 74, 76, 72, 80, 77, 79] are inspired on *Decision Trees/Random Forests*, [9, 29, 31, 34] consider *Kernel Methods* for inducing concept descriptions, while [60, 62, 61, 63, 94] consider essentially a *Naive Bayes* approach. Last but not least, [43] is inspired on *Genetic Programming* to induce concept expressions, while [65] is based on the *Reinforcement Learning* framework. Eventually, [73] proposes to use decision trees to learn so-called *disjointness axioms*, *i.e.* expressions of the form $C \sqcap D \sqsubseteq \perp$, declaring that class C and D are disjoint.

6 Conclusions & Future Work

In this work, we addressed the problem of automatically learning fuzzy concept inclusion axioms from OWL 2 ontologies to describe sufficient condition of being an individual classified as instance of target class T. That is, given a target class T of an OWL ontology, we have addressed the problem of inducing fuzzy concept inclusion axioms that describe sufficient conditions for being an individual instance of T. Specifically, we have presented a two-stage algorithm, called PN-OWL that is inspired on the PN-rule [2, 3, 40, 41] and adapted to the context of OWL. The main features of our algorithm are essentially the fact that *(i)* at the P-stage, it generates a set of fuzzy inclusion axioms, the P-rules, that cover as many as possible instances of the target class T without compromising too much the amount on non-positives; *(ii)* at the N-stage, it generates a set of fuzzy inclusion axioms, the N-rules, that cover as many as possible of non-positive instances of class T of the P-stage; *(iii)* the fuzzy inclusion axioms are then combined (aggregated) into a new fuzzy inclusion axiom describing sufficient conditions for being an individual classified as an instance of the target class T. Additionally, all fuzzy inclusion axioms may possibly include fuzzy concepts and fuzzy concrete domains, where each axiom has a leveraging weight (specifically, called confidence or precision), and all generated fuzzy concept inclusion axioms can directly be encoded as *Fuzzy OWL 2* axioms.

We have also conducted an extensive evaluation, comparing it with fuzzy Foil-$\mathcal{DL}$. Our evaluation shows that, PN-OWL performs generally better than fuzzy Foil-$\mathcal{DL}$ in terms of effectiveness, though finding an optimal parameter configuration is much more time consuming than for Foil-$\mathcal{DL}$ as PN-OWL has double as many parameters than fuzzy Foil-$\mathcal{DL}$.

Concerning future work, besides investigating about other learning methods, and future work listed here and there in the paper, we envisage various aspects worth to be investigated in more detail: *(i)* it is still unclear how the construction of fuzzy sets may impact effectiveness. So far, we did not notice a clear winner between the uniform clustering and c-means clustering algo-

[14] We recall that $\mathcal{EL}$ is a strict sub-logic of $\mathcal{ALC}$.

rithms used to build fuzzy datatypes. This is somewhat surprising. We would like to investigate that in more detail by considering various alternatives as well [1] and/or considering clustering methods based on the aggregation of data properties, *i.e.* multi-dimensional clustering versus uni-dimensional clustering; *(ii)* moreover, we would like to cover more OWL datatypes than those considered here so far (numerical and boolean) such as strings, dates, etc., possibly in combination with some classical machine learning methods (see, *e.g.* [83]); *(iii)* we would like to investigate the computational aspect: so far, for some ontologies, a learning run may take even a week (w.r.t. our available resources). Here, we would like to investigate both parallelization methods as well as to investigate about the impact, in terms of effectiveness, of efficient, logically sound, but not necessarily complete, reasoning algorithms; *(iv)* in principle, our two-stage algorithm PN-OWL is parametric w.r.t. the learner to be used during both the P-stage and the N-stage (*cf.* lines 5 and 19 of the PN-OWL algorithm): here we would like to investigate how to plug in another alternative such as Fuzzy OWL-Boost [20] and to verify its effectiveness; *(v)* we would like to asses also the impact of other alternative scoring functions to information gain (*cf.* Eq. 12) within our setting, inclusive various alternative choices of t-norms and r-implications; and (*vi*) we are looking for combining our Fuzzy DL-Learning with sub-symbolic learning methods, such as *e.g.* Neural Networks, an activity that is already on-going.

Moreover, we really would like to consider extending the hypothesis language $\mathcal{EL}(\mathbf{D})$ with so-called *threshold concepts* [11] of the form $C[\geq d]$ (resp. $C[\leq d]$), where $d \in [0,1]$ and C is either a class name or an existential restriction, with the intended meaning "$C[\geq d]$ (resp. $C[\leq d]$) is the fuzzy set of individuals that are instances of C to degree greater (resp. smaller) than or equal to d." This would provide us a more fine grained hypothesis language in which a threshold may be defined for each conjunct of a rule rather than via a rule confidence threshold as it is now. A Fuzzy OWL 2 example of such a rule may be, by referring to the `Wine Quality` ontology and target wine `1`

```
(implies (and (some alcohol alcohol_VH)[<= 0.786]
              (some sulphates sulphates_H)[>= 0.289]
              (some pH pH_L)[<= 0.106])
              1)
```

with intended meaning "if, for an individual (wine) a, the alcohol level of being very high is smaller than or equal to 0.786, the sulphates level of being high is greater than or equal to 0.289 and the pH level of being low is smaller than or equal 0.106 then classify a to some extend (*e.g.* the minimum of the degrees of being a an instance of a conjunct) as instance of the target class `1`.

Acknowledgment

This research was partially supported by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under (GA No 952215). This work has also been partially supported by the H2020 DeepHealth Project (GA No. 825111). This paper is also supported by the FAIR (Future Artificial Intelligence Research) project funded by the NextGenerationEU program within the PNRR-PE-AI scheme (M4C2, investment 1.3, line on Artificial Intelligence). Eventually, this work has also been partially supported by the H2020 STARWARS Project (GA No. 101086252), type of action HORIZON TMA MSCA Staff Exchanges.

We wish to thank Centro Servizi CNR of the ICT-SAC Department of the National Research Council for the precious computing services and resources they made available. We wish to address a special thanks to Ing. Giorgio Bartoccioni (ICT-SAC) for his technical support.

References

[1] *Special issue on Fuzzy Clustering, Fuzzy Sets and Systems*, volume 389. Elsevier, 2020.

[2] Ramesh C. Agarwal and Mahesh V. Joshi. Pnrule: A new framework for learning classifier models in data mining (a case-study in network intrusion detection). Technical report rc 21719, IBM Research Report, 2000.

[3] Ramesh C. Agarwal and Mahesh V. Joshi. Pnrule: A new framework for learning classifier models in data mining (a case-study in network intrusion detection). In *Proceedings of the 2001 SIAM International Conference on Data Mining (SDM-01)*, pages 1–17, 2001.

[4] H. Anderson and Phil Roth. Ember: An open dataset for training static pe malware machine learning models. *ArXiv*, abs/1804.04637, 2018.

[5] Alessandro Artale, Diego Calvanese, Roman Kontchakov, and Michael Zakharyaschev. The DL-Lite family and relations. *Journal of Artificial Intelligence Research*, 36:1–69, 2009.

[6] Franz Baader, Rafael Peñaloza, and Boontawee Suntisrivaraporn. Pinpointing in the description logic $\mathcal{EL}^+$. In *Proceedings of the 30th Annual German Conference on Advances in Artificial Intelligence (KI-07)*, number 4667 in Lecture Notes in Computer Science, pages 52–67, Berlin, Heidelberg, 2007. Springer-Verlag.

[7] Liviu Badea and Shan-Hwei Nienhuys-Cheng. A refinement operator for description logics. In *Inductive Logic Programming, 10th International Conference, ILP-00*, volume 1866 of *Lecture Notes in Computer Science*, pages 40–59. Springer, 2000.

[8] James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Springer Verlag, 1981.

[9] Stephan Bloehdorn and York Sure. Kernel methods for mining instance data in ontologies. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007.*, volume 4825 of *Lecture Notes in Computer Science*, pages 58–71. Springer Verlag, 2007.

[10] Fernando Bobillo, Marco Cerami, Francesc Esteva, Àngel García-Cerdaña, Rafael Peñaloza, and Umberto Straccia. Fuzzy description logics in the framework of mathematical fuzzy logic. In Carles Noguera Petr Cintula, Christian Fermüller, editor, *Handbook of Mathematical Fuzzy Logic, Volume 3*, volume 58 of *Studies in Logic, Mathematical Logic and Foundations*, chapter 16, pages 1105–1181. College Publications, 2015.

[11] Fernando Bobillo and Umberto Straccia. fuzzyDL: An expressive fuzzy description logic reasoner. In *2008 International Conference on Fuzzy Systems (FUZZ-08)*, pages 923–930. IEEE Computer Society, 2008.

[12] Fernando Bobillo and Umberto Straccia. Representing fuzzy ontologies in owl 2. In *Proceedings of the 19th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2010)*, pages 2695–2700. IEEE Press, July 2010.

[13] Fernando Bobillo and Umberto Straccia. Fuzzy ontology representation using OWL 2. *International Journal of Approximate Reasoning*, 52:1073–1094, 2011.

[14] Fernando Bobillo and Umberto Straccia. Aggregation operators for fuzzy ontologies. *Applied Soft Computing*, 13(9):3816–3830, 2013.

[15] Fernando Bobillo and Umberto Straccia. The fuzzy ontology reasoner *fuzzyDL*. *Knowledge-Based Systems*, 95:12 – 34, 2016.

[16] Fernando Bobillo and Umberto Straccia. Reasoning within fuzzy owl 2 el revisited. *Fuzzy Sets and Systems*, 351:1–40, 2018.

[17] Stefan Borgwardt, Marco Cerami, and Rafael Peñaloza. The complexity of fuzzy $\mathcal{EL}$ under the Lukasiewicz t-norm. *International Journal of Approximate Reasoning*, 91:179–201, 2017.

[18] Lorenz Bühmann, Jens Lehmann, and Patrick Westphal. Dl-learner - A framework for inductive learning on the semantic web. *Journal of Web Semantics*, 39:15–24, 2016.

[19] Lorenz Bühmann, Jens Lehmann, Patrick Westphal, and Simon Bin. Dl-learner structured machine learning on semantic web data. In *Companion of the The Web Conference 2018 on The Web Conference 2018, WWW 2018, Lyon , France, April 23-27, 2018*, pages 467–471. ACM, 2018.

[20] Franco Alberto Cardillo and Umberto Straccia. Fuzzy owl-boost: Learning fuzzy concept inclusions via real-valued boosting. *Fuzzy Sets and Systems*, 438:164–186, 2022.

[21] Marco Cerami and Umberto Straccia. On the (un)decidability of fuzzy description logics under lukasiewicz t-norm. *Information Sciences*, 227:1–21, 2013.

[22] Mahsa Chitsaz, Kewen Wang, Michael Blumenstein, and Guilin Qi. Concept learning for $\mathcal{EL}^{++}$; by refinement and reinforcement. In *Proceedings of the 12th Pacific Rim international conference on Trends in Artificial Intelligence*, PRICAI'12, pages 15–26, Berlin, Heidelberg, 2012. Springer-Verlag.

[23] Marcos E. Cintra, Maria Carolina Monard, and Heloisa de Arruda Camargo. On rule learning methods: A comparative analysis of classic and fuzzy approaches. In *Soft Computing: State of the Art Theory and Novel Applications*, volume 291, pages 89–104. Springer Verlag, 2013.

[24] Claudia d'Amato. Machine learning for the semantic web: Lessons learnt and next research directions. *Semantic Web*, 11(1):195–203, 2020.

[25] María José del Jesús, Frank Hoffmann, Luis Junco Navascués, and Luciano Sánchez. Induction of fuzzy-rule-based classifiers with evolutionary boosting algorithms. *IEEE Transactions on Fuzzy Systems*, 12(3):296–308, 2004.

[26] Mario Drobics, Ulrich Bodenhofer, and Erich-Peter Klement. Fs-foil: an inductive learning method for extracting interpretable fuzzy descriptions. *International Journal of Approximate Reasoning*, 32(2-3):131–152, 2003.

[27] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[28] Nicola Fanizzi. Concept induction in description logics using information-theoretic heuristics. *Int. J. Semantic Web Inf. Syst.*, 7(2):23–44, 2011.

[29] Nicola Fanizzi and Claudia d'Amato. A declarative kernel for $\mathcal{ALC}$ concept descriptions. In *Foundations of Intelligent Systems, 16th International Symposium, ISMIS 2006, Bari, Italy, September 27-29, 2006, Proceedings*, volume 4203 of *Lecture Notes in Computer Science*, pages 322–331. Springer Verlag, 2006.

[30] Nicola Fanizzi, Claudia d'Amato, and Floriana Esposito. DL-FOIL concept learning in description logics. In Filip Zelezný and Nada Lavrač, editors, *Inductive Logic Programming*, volume 5194 of *Lecture Notes in Computer Science*, pages 107–121. Springer, 2008.

[31] Nicola Fanizzi, Claudia d'Amato, and Floriana Esposito. Induction of classifiers through non-parametric methods for approximate classification and retrieval with ontologies. *Int. J. Semantic Computing*, 2(3):403–423, 2008.

[32] Nicola Fanizzi, Claudia d'Amato, and Floriana Esposito. Induction of concepts in web ontologies through terminological decision trees. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part I*, volume 6321 of *Lecture Notes in Computer Science*, pages 442–457. Springer Verlag, 2010.

[33] Nicola Fanizzi, Claudia d'Amato, and Floriana Esposito. Towards the induction of terminological decision trees. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 1423–1427, New York, NY, USA, 2010. ACM.

[34] Nicola Fanizzi, Claudia d'Amato, and Floriana Esposito. Induction of robust classifiers for web ontologies through kernel machines. *J. Web Sem.*, 11:1–13, 2012.

[35] Nicola Fanizzi, Giuseppe Rizzo, and Claudia d'Amato. Boosting DL concept learners. In *The Semantic Web - 16th International Conference, ESWC-19*, volume 11503 of *Lecture Notes in Computer Science*, pages 68–83, 2019.

[36] Nicola Fanizzi, Giuseppe Rizzo, Claudia d'Amato, and Floriana Esposito. Dlfoil: Class expression learning revisited. In *Knowledge Engineering and Knowledge Management - 21st International Conference, EKAW-18*, volume 11313 of *Lecture Notes in Computer Science*, pages 98–113, 2018.

[37] Petr Hájek. *Metamathematics of Fuzzy Logic*. Kluwer, 1998.

[38] Ignacio Huitzil, Umberto Straccia, Natalia Díaz-Rodríguez, and Fernando Bobillo. Datil: Learning fuzzy ontology datatypes. In *Proceedings of the 17th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2018), Part II*, volume 854 of *Communications in Computer and Information Science*, pages 100–112. Springer, June 2018.

[39] Josué Iglesias and Jens Lehmann. Towards integrating fuzzy logic capabilities into an ontology-based inductive logic programming framework. In *Proceedings of the 11th International Conference on Intelligent Systems Design and Applications (ISDA 2011)*, pages 1323–1328, 2011.

[40] Mahesh V. Joshi, Ramesh C. Agarwal, and Vipin Kumar. Mining needle in a haystack: Classifying rare classes via two-phase rule induction. In Sharad Mehrotra and Timos K. Sellis, editors, *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data, Santa Barbara, CA, USA, May 21-24, 2001*, pages 91–102. ACM, 2001.

[41] Mahesh V. Joshi, Ramesh C. Agarwal, and Vipin Kumar. Predicting rare classes: can boosting make any weak learner strong? In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, pages 297–306. ACM, 2002.

[42] S. Konstantopoulos and A. Charalambidis. Formulating description logic learning as an inductive logic programming task. In *Proceedings of the 19th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2010)*, pages 1–7. IEEE Press, 2010.

[43] Jens Lehmann. Hybrid learning of ontology classes. In *Machine Learning and Data Mining in Pattern Recognition, 5th International Conference, MLDM 2007, Leipzig, Germany, July 18-20, 2007, Proceedings*, volume 4571 of *Lecture Notes in Computer Science*, pages 883–898. Springer Verlag, 2007.

[44] Jens Lehmann. DL-Learner: Learning concepts in description logics. *Journal of Machine Learning Research*, 10:2639–2642, 2009.

[45] Jens Lehmann and Christoph Haase. Ideal downward refinement in the EL\mathcal{EL} description logic. In *Inductive Logic Programming, 19th International Conference, ILP 2009, Leuven, Belgium, July 02-04, 2009. Revised Papers*, volume 5989 of *Lecture Notes in Computer Science*, pages 73–87. Springer, 2009.

[46] Jens Lehmann and Pascal Hitzler. A refinement operator based learning algorithm for the ALC description logic. In *Inductive Logic Programming, 17th International Conference, ILP 2007, Corvallis, OR, USA, June 19-21, 2007, Revised Selected Papers*, volume 4894 of *Lecture Notes in Computer Science*, pages 147–160. Springer Verlag, 2007.

[47] Jens Lehmann and Pascal Hitzler. Foundations of Refinement Operators for Description Logics. In H. Blockeel, J. Ramon, J. W. Shavlik, and P. Tadepalli, editors, *Inductive Logic Programming*, volume 4894 of *Lecture Notes in Artificial Intelligence*, pages 161–174. Springer, 2008.

[48] Jens Lehmann and Pascal Hitzler. Concept learning in description logics using refinement operators. *Machine Learning*, 78(1-2):203–250, 2010.

[49] Francesca A. Lisi and Donato Malerba. Ideal refinement of descriptions in al-log. In *Inductive Logic Programming: 13th International Conference, ILP-03*, volume 2835 of *Lecture Notes in Computer Science*, pages 215–232, 2003.

[50] Francesca A. Lisi and Umberto Straccia. A logic-based computational method for the automated induction of fuzzy ontology axioms. *Fundamenta Informaticae*, 124(4):503–519, 2013.

[51] Francesca A. Lisi and Umberto Straccia. A system for learning GCI axioms in fuzzy description logics. In *Proceedings of the 26th International Workshop on Description Logics (DL-13)*, volume 1014 of *CEUR Workshop Proceedings*, pages 760–778. CEUR-WS.org, 2013.

[52] Francesca A. Lisi and Umberto Straccia. Can ilp deal with incomplete and vague structured knowledge? In Stephen H. Muggleton and Hiroaki Watanabe, editors, *Latest Advances in Inductive Logic Programming*, chapter 21, pages 199–206. World Scientific, 2014.

[53] Francesca A. Lisi and Umberto Straccia. Learning in description logics with fuzzy concrete domains. *Fundamenta Informaticae*, 140(3-4):373–391, 2015.

[54] Francesca Alessandra Lisi. Logics in machine learning and data mining: Achievements and open issues. In *Proceedings of the 34th Italian Conference on Computational Logic, Trieste, Italy, June 19-21, 2019.*, volume 2396 of *CEUR Workshop Proceedings*, pages 82–88. CEUR-WS.org, 2019.

[55] Francesca Alessandra Lisi and Umberto Straccia. An inductive logic programming approach to learning inclusion axioms in fuzzy description logics. In *26th Italian Conference on Computational Logic (CILC-11)*, volume 810, pages 57–71. CEUR Electronic Workshop Proceedings, 2011.

[56] Francesca Alessandra Lisi and Umberto Straccia. Towards learning fuzzy dl inclusion axioms. In *9th International Workshop on Fuzzy Logic and Applications (WILF-11)*, volume 6857 of *Lecture Notes in Computer Science*, pages 58–66, Berlin, 2011. Springer Verlag.

[57] Francesca Alessandra Lisi and Umberto Straccia. Dealing with incompleteness and vagueness in inductive logic programming. In *28th Italian Conference on Computational Logic (CILC-13)*, volume 1068, pages 179–193. CEUR Electronic Workshop Proceedings, 2013.

[58] Francesca Alessandra Lisi and Umberto Straccia. A foil-like method for learning under incompleteness and vagueness. In *23rd International Conference on Inductive Logic Programming*, volume 8812 of *Lecture Notes in Artificial Intelligence*, pages 123–139, Berlin, 2014. Springer Verlag. Revised Selected Papers.

[59] Thomas Lukasiewicz and Umberto Straccia. Managing uncertainty and vagueness in description logics for the semantic web. *Journal of Web Semantics*, 6:291–308, 2008.

[60] Pasquale Minervini, Claudia d'Amato, and Nicola Fanizzi. Learning terminological naive bayesian classifiers under different assumptions on missing knowledge. In *Proceedings of the 7th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW-11*, volume 778 of *CEUR Workshop Proceedings*, pages 63–74. CEUR-WS.org, 2011.

[61] Pasquale Minervini, Claudia d'Amato, and Nicola Fanizzi. Learning probabilistic description logic concepts: Under different assumptions on missing knowledge. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 378–383, New York, NY, USA, 2012. ACM.

[62] Pasquale Minervini, Claudia d'Amato, and Nicola Fanizzi. Learning terminological bayesian classifiers - A comparison of alternative approaches to dealing with unknown concept-memberships. In *Proceedings of the 9th Italian Convention on Computational Logic, Rome, Italy, June 6-7, 2012*, volume 857 of *CEUR Workshop Proceedings*, pages 191–205, 2012.

[63] Pasquale Minervini, Claudia d'Amato, Nicola Fanizzi, and Floriana Esposito. Learning probabilistic description logic concepts under alternative assumptions on incompleteness. In *Uncertainty Reasoning for the Semantic Web III - ISWC International Workshops, URSW 2011-2013, Revised Selected Papers*, volume 8816 of *Lecture Notes in Computer Science*, pages 184–201, 2014.

[64] Boris Motik and Riccardo Rosati. A faithful integration of description logics with logic programming. In *Proceedings of the 20th international joint conference on Artifical intelligence*, pages 477–482, San Francisco, CA, USA, 2007. Morgan Kaufmann Publishers Inc.

[65] Matthias Nickles and Achim Rettinger. Interactive relational reinforcement learning of concept semantics. *Machine Learning*, 94(2):169–204, 2014.

[66] José Otero and Luciano Sánchez. Induction of descriptive fuzzy classifiers with the logitboost algorithm. *Soft Computing*, 10(9):825–835, 2006.

[67] OWL 2 Web Ontology Language Document Overview. *https://www.w3.org/TR/owl2-overview/*. W3C, 2009.

[68] Ana M. Palacios, Luciano Sánchez, and Inés Couso. Using the AdaBoost algorithm for extracting fuzzy rules from low quality data: Some preliminary results. In *FUZZ-IEEE 2011, IEEE International Conference on Fuzzy Systems, Taipei, Taiwan, 27-30 June, 2011, Proceedings*, pages 1263–1270. IEEE, 2011.

[69] Luc De Raedt. *Logical and relational learning*. Cognitive Technologies. Springer, 2008.

[70] Luc De Raedt and Kristian Kersting. Statistical relational learning. In *Encyclopedia of Machine Learning and Data Mining*, pages 1177–1187. Springer, 2017.

[71] Achim Rettinger, Uta Lösch, Volker Tresp, Claudia d'Amato, and Nicola Fanizzi. Mining the semantic web - statistical learning for next generation knowledge bases. *Data Minining and Knowledge Discovery*, 24(3):613–662, 2012.

[72] Giuseppe Rizzo, Claudia d'Amato, and Nicola Fanizzi. On the effectiveness of evidence-based terminological decision trees. In *Foundations of Intelligent Systems - 22nd International Symposium, ISMIS-15*, volume 9384 of *Lecture Notes in Computer Science*, pages 139–149, 2015.

[73] Giuseppe Rizzo, Claudia d'Amato, and Nicola Fanizzi. An unsupervised approach to disjointness learning based on terminological cluster trees. *Semantic Web*, 12(3):423–447, 2021.

[74] Giuseppe Rizzo, Claudia d'Amato, Nicola Fanizzi, and Floriana Esposito. Tackling the class-imbalance learning problem in semantic web knowledge bases. In *Knowledge Engineering and Knowledge Management - 19th International Conference, EKAW-14*, volume 8876 of *Lecture Notes in Computer Science*, pages 453–468, 2014.

[75] Giuseppe Rizzo, Claudia d'Amato, Nicola Fanizzi, and Floriana Esposito. Towards evidence-based terminological decision trees. In *Information Processing and Management of Uncertainty in Knowledge-Based Systems - 15th International Conference, IPMU 2014, Montpellier, France, July 15-19, 2014, Proceedings, Part I*, volume 442 of *Communications in Computer and Information Science*, pages 36–45. Springer, 2014.

[76] Giuseppe Rizzo, Claudia d'Amato, Nicola Fanizzi, and Floriana Esposito. Inductive classification through evidence-based models and their ensembles. In *The Semantic Web. Latest Advances and New Domains - 12th European Semantic Web Conference, ESWC-15*, volume 9088 of *Lecture Notes in Computer Science*, pages 418–433, 2015.

[77] Giuseppe Rizzo, Claudia d'Amato, Nicola Fanizzi, and Floriana Esposito. Tree-based models for inductive classification on the web of data. *Journal of Web Semantics*, 45:1–22, 2017.

[78] Giuseppe Rizzo, Nicola Fanizzi, and Claudia d'Amato. Class expression induction as concept space exploration: From dl-foil to dl-focl. *Future Generation Computing Systems*, 108:256–272, 2020.

[79] Giuseppe Rizzo, Nicola Fanizzi, Claudia d'Amato, and Floriana Esposito. Approximate classification with web ontologies through evidential terminological trees and forests. *International Journal of Approximate Reasoning*, 92:340–362, 2018.

[80] Giuseppe Rizzo, Nicola Fanizzi, Jens Lehmann, and Lorenz Bühmann. Integrating new refinement operators in terminological decision trees learning. In *Knowledge Engineering and Knowledge Management - 20th International Conference, EKAW-16*, volume 10024 of *Lecture Notes in Computer Science*, pages 511–526, 2016.

[81] Luciano Sánchez and José Otero. Boosting fuzzy rules in classification problems under single-winner inference. *International Journal of Intelligent Systems*, 22(9):1021–1034, 2007.

[82] Mathieu Serrurier and Henri Prade. Improving expressivity of inductive logic programming by learning different kinds of fuzzy rules. *Soft Computing*, 11(5):459–466, 2007.

[83] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms.* Cambridge University Press, 2014.

[84] D. Shibata, N. Inuzuka, S. Kato, T. Matsui, and H. Itoh. An induction algorithm based on fuzzy logic programming. In N. Zhong and L. Zhou, editors, *Methodologies for Knowledge Discovery and Data Mining, Third Pacific-Asia Conference, PAKDD-99, Beijing, China, April 26-28, 1999, Proceedings*, volume 1574 of *Lecture Notes in Computer Science*, pages 268–273. Springer, 1999.

[85] Umberto Straccia. Description logics with fuzzy concrete domains. In Fahiem Bachus and Tommi Jaakkola, editors, *21st Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 559–567, Edinburgh, Scotland, 2005. AUAI Press.

[86] Umberto Straccia. *Foundations of Fuzzy Logic and Semantic Web Languages.* CRC Studies in Informatics Series. Chapman & Hall, 2013.

[87] Umberto Straccia and Matteo Mucci. pFOIL-DL: Learning (fuzzy) $\mathcal{EL}$ concept descriptions from crisp OWL data using a probabilistic ensemble estimation. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing (SAC-15)*, pages 345–352, Salamanca, Spain, 2015. ACM.

[88] Peter Svec, Stefan Balogh, and Martin Homola. Experimental evaluation of description logic concept learning algorithms for static malware detection. In Paolo Mori, Gabriele Lenzini, and Steven Furnell, editors, *Proceedings of the 7th International Conference on Information Systems Security and Privacy, ICISSP 2021, Online Streaming, February 11-13, 2021*, pages 792–799. SCITEPRESS, 2021.

[89] Peter Svec, Stefan Balogh, Martin Homola, and Ján Kluka. Knowledge-based dataset for training PE malware detection models. *CoRR*, abs/2301.00153, 2023.

[90] Vicenç Torra and Yasuo Narukawa. *Information Fusion and Aggregation Operators.* Cognitive Technologies. Springer Verlag, 2007.

[91] Patrick Westphal, Lorenz Bühmann, Simon Bin, Hajira Jabeen, and Jens Lehmann. SML-bench - A benchmarking framework for structured machine learning. *Semantic Web*, 10(2):231–245, 2019.

[92] Hong yang Zhu, Yi Ding, Hong Gao, and Wei Liu. Fuzzy prediction in classification of AdaBoost algorithm. In *International Conference on Oriental Thinking and Fuzzy Logic*, volume 443 of *Advances in Intelligent Systems and Computing*, pages 129–136. Springer, 2016.

[93] L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.

[94] Man Zhu, Zhiqiang Gao, Jeff Z. Pan, Yuting Zhao, Ying Xu, and Zhibin Quan. Tbox learning from incomplete data by inference in belnet+. *Knoweledge-Based Systems*, 75:30–40, 2015.

A Brief Description of the Datasets

Find below a brief description about the OWL ontologies in Table 3 used in our experiments.

SemanticBible (NTN). *New Testament Names* (NTN) is an ontology describing each named thing in the New Testament, about 600 names in all. Each named thing (an entity) is categorized according to its class, including God, Jesus, individual men and women, groups of people, and locations. These entities are related to each other by properties that interconnect the entities into a web of information.[15] The target is to learn sufficient conditions to be a woman.

Lymphography. This ontology is about lymphography patient data and the target is the prediction of a diagnosis class based on the lymphography patient data [91].

[15] `http://semanticbible.com/ntn/ntn-overview.html`

Mammographic. This ontology is about mammography screening data and the target is the prediction of breast cancer severity based on the screening data [91].

Malware. This ontology is the description of a PE Malware Ontology that offers a reusable semantic schema for Portable Executable (PE,Windows binary format) malware files [88, 89]. The ontology is inspired by the structure of the data in the EMBER dataset,[16] which is intended for static malware analysis [4].

The following datasets have been taken from the well-known *UC Irvine Machine Learning Repository* [27].

Iris. The data set contains 3 classes of 50 instances each, where each class refers to a type of iris plant. The attributes are: sepal length in cm, sepal width in cm, petal length in cm and petal width in cm. The target classes are: Iris Setosa, Iris Versicolour and Iris Virginica.

Wine. These data are the results of a chemical analysis of wines grown in the same region in Italy but derived from three different cultivars. The analysis determined the quantities of 13 constituents found in each of the three types of wines. The attributes are alcohol, malic acid, ash, alcalinity of ash, magnesium, total phenols, flavonoids, nonflavonoid phenols, proanthocyanins, color intensity, hue, OD280/OD315 of diluted wines and proline. The target classes are the three wines $1, 2$ and 3.

Wine Quality. The data set is related to red and white variants of the Portuguese "Vinho Verde" wine. The goal is to model wine quality based on physicochemical tests. Due to privacy and logistic issues, only physicochemical (inputs) and sensory (the output) variables are available (e.g. there is no data about grape types, wine brand, wine selling price, etc.). The attributes are: fixed acidity, volatile acidity, citric acid, residual sugar, chlorides, free sulfur dioxide, total sulfur dioxide, density, pH, sulphates, alcohol and quality (score between 0 and 10). The target is to describe good red wines, which are defined as red wines having quality score greater than or equal to 7. The quality attribute has been removed from the ontology during training and tests.

Yeast. The data set is about the prediction of the cellular localization sites of proteins (10 target classes) The set of attributes is: Sequence Name (accession number for the SWISS-PROT database), mcg (McGeoch's method for signal sequence recognition); gvh (von Heijne's method for signal sequence recognition); alm (score of the ALOM membrane spanning region prediction program); mit (Score of discriminant analysis of the amino acid content of the N-terminal region, 20 residues long, of mitochondrial and non-mitochondrial proteins); erl (presence of "HDEL" substring, thought to act as a signal for retention in the endoplasmic reticulum lumen, binary attribute); pox (peroxisomal targeting signal in the C-terminus); vac (score of discriminant analysis of the amino acid content of vacuolar and extracellular proteins); and nuc (score of discriminant analysis of nuclear localization signals of nuclear and non-nuclear proteins).

[16] https://github.com/elastic/ember