PN-OWL: A Two-Stage Algorithm to Learn Fuzzy Concept Inclusions Rules from OWL 2 Ontologies

Franco Alberto Cardillo[a], Franca Debole[b], Umberto Straccia[b]

[a]*Istituto di Linguistica Computazionale, CNR, Pisa, Italy*
[b]*Istituto di Scienza e Tecnologie dell'Informazione, CNR, Pisa, Italy*

Abstract

Given a target class T of an OWL 2 ontology and instances of T, we address the problem of learning graded fuzzy class inclusion rules that aim to describe sufficient conditions for being an individual classified as an instance of the class T.

To do so, we present PN-OWL that is a two-stage learning algorithm consisting of a P-stage and an N-stage. In the P-stage, the algorithm learns fuzzy OWL 2 class inclusion rules (the P-rules). These rules aim to cover as many positive examples as possible, increasing *recall*, without compromising too much *precision*. In the N-stage, the algorithm learns fuzzy OWL 2 class inclusion rules (the N-rules), that try to rule out as many false positives, covered by the rules learnt at the P-stage, as possible. Roughly, the P-rules tell why an individual should be classified as an instance of T, while the N-rules tell why it should not.

PN-OWL then aggregates the P-rules and the N-rules by combining them via an aggregation function to allow for a final decision on whether an individual is an instance of T or not.

We also illustrate the effectiveness of PN-OWL through extensive experimentation.

Keywords: OWL 2 Ontologies, Machine Learning, Fuzzy Logic, Concept/Class Inclusion Rules

Email addresses: francoalberto.cardillo@ilc.cnr.it (Franco Alberto Cardillo), franca.debole@isti.cnr.it (Franca Debole), umberto.straccia@isti.cnr.it (Umberto Straccia)

1. Introduction

OWL 2 [1], a W3C standard, is nowadays a quite popular formal language allowing to represent *structured* knowledge. Its formal semantics is based on *Description Logics* (DLs) [2].

The main ingredients of DLs are concept/class descriptions, inheritance relationships among them, and instances of them. *Fuzzy DLs* [3, 4] have been proposed as the natural extension of DLs to mathematical fuzzy logic, while *Fuzzy OWL 2* [5] is a proposed fuzzy extension of OWL 2.

Despite an important amount of work that has been carried out about DLs, the application of machine learning techniques to OWL 2 ontologies is relatively less addressed, especially compared to the *Inductive Logic Programming* (ILP) setting (see *e.g.* [6] for more insights on ILP).

In this work, we focus on the problem of learning fuzzy concept/class inclusion rules from OWL 2 ontologies [7, 8]. Specifically, given a target class T of an OWL 2 ontology and instances of T (the so-called positive examples), we address the problem of learning graded fuzzy class inclusion rules that aim to describe sufficient conditions for being an individual classified as an instance of the class T.

To do so, we propose to rely on an adaptation of the PN-rule algorithm [9, 10] to the fuzzy OWL 2 case. Our algorithm, called PN-OWL, consists of a *P-stage* in which *positive* fuzzy rules (called *P-rules*) are learnt to cover as many as possible instances of the target class (increase *recall*), without compromising too much *precision*, and an *N-stage* in which *negative* fuzzy rules (called *N-rules*) are learnt to rule out as many false positives, covered by the P-stage, as possible. Essentially, the P-rules tell why an individual should be classified as an instance of T, while the N-rules tell why it should not. Then we aggregate the rules learnt at the P-stage and the N-stage by combining them via an aggregation function to allow for a final decision on whether an individual is an instance of T or not.

A feature of our PN-OWL algorithm is that it encodes all learned rules in Fuzzy OWL 2. Consequently, a Fuzzy OWL 2 reasoner can automatically assess the degree to which an individual belongs to the target class T. We will also illustrate the effectiveness of PN-OWL through extensive experimentation on various real-valued OWL 2 ontologies.

The paper is organized as follows. In Section 2, we recap shortly the main concepts and notations we will rely on in this paper. Then, in Section 3 we will present PN-OWL, which is evaluated in Section 4. In Section 5

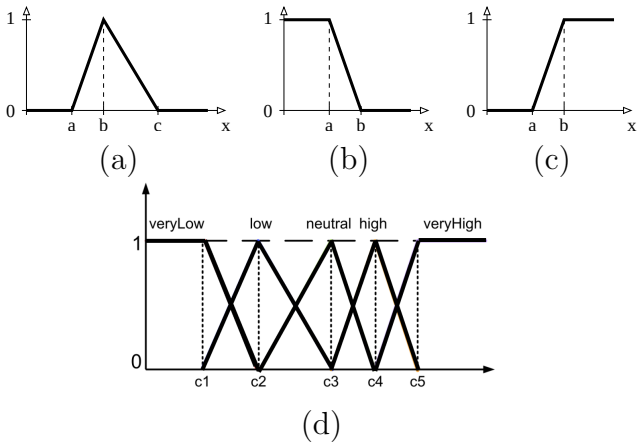

Fig. 1. (a) triangular function $tri(a, b, c)$, (b) left shoulder function $ls(a, b)$, (c) right shoulder function $rs(a, b)$, and (d) fuzzy sets over centroids.

we compare our work with closely related work appeared so far. Section 6 concludes and points to some topics of further research.

2. Background

For the sake of completeness, we introduce the main notions related to *Fuzzy Sets* and *Fuzzy Description Logics* we will use in this work (for a more in-depth presentation see *e.g.* [3, 4, 11, 12]). The informed reader may skip this section and get back when needed.

Fuzzy Sets basics. A *fuzzy set* A over a finite crisp set X is a function $A\colon X \to [0, 1]$, called *fuzzy membership* function of A. 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 $|A| = \sum_{x \in X} A(x)$. The triangular, the left-shoulder, and right-shoulder functions are frequently used to specify membership functions of fuzzy sets (see (a) - (c) in Fig. 1). One easy method to define the membership functions is to uniformly partition the values of an attribute into 3, 5, or 7 fuzzy sets. Another popular approach may consist in using the *c-means* fuzzy clustering algorithm (see, *e.g.* [13]) with 3, 5, or 7 clusters, where the fuzzy membership functions are shaped as triangular or shoulder functions built around the centroids of the clusters [14, 15] (see (d) in Fig. 1).

Fuzzy Description Logics basics. We recap the fuzzy DL $\mathcal{ALE}_{@}(\mathbf{D})$, which extends the fuzzy DL $\mathcal{ALE}(\mathbf{D})$ with the *aggregated concept* construct indicated with @ [16]. $\mathcal{ALE}_{@}(\mathbf{D})$ is expressive enough to capture the main ingredients of fuzzy DLs we are going to use here.

We start with the notion of *fuzzy concrete domain*, $\mathbf{D} = \langle \Delta^{\mathbf{D}}, \cdot^{\mathbf{D}} \rangle$, where $\Delta^{\mathbf{D}}$ is a datatype, $\cdot^{\mathbf{D}}$ is a mapping 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 $\Delta^{\mathbf{D}}$ (*i.e.* $\cdot^{\mathbf{D}}$ maps each datatype predicate to a function from $\Delta^{\mathbf{D}}$ to $[0,1]$). Typical data values one may encounter in OWL 2 ontologies are strings, numbers, dates, and booleans. For instance, in the domain of numbers, typical datatype predicates $\mathbf{d}$ are characterized by the well-known membership functions

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

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 used to combine different pieces of information. Numerous AOs differ on the assumptions of the data (data types) and the type of information we can incorporate in the model [17]. There is no standard definition of AO. To what concerns us, an AO of dimension n is a mapping $@ : [0,1]^n \rightarrow [0,1]$. We assume to have a finite family $@_1, \ldots, @_l$ of AOs.

We are now ready to define $\mathcal{ALE}_{@}(\mathbf{D})$. So, 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* r or a *datatype property* s. The set of *concepts* are built from concept names A, A_i using connectives and quantification constructs over object properties and datatype properties, as described by the following syntactic rule ($n_i \geq 1$):

$$\begin{array}{rcl} C & \rightarrow & \top \mid \ \bot \ \mid A \mid \neg A \mid C_1 \sqcap C_2 \mid \\ & & \exists r.C \mid \exists s.\mathbf{d} \mid @_i(A_1, \ldots, A_{n_i}) \ . \end{array}$$

An *ABox* $\mathcal{A}$ consists of a finite set of *assertion* axioms that are expressions of the form $\langle a{:}C, \alpha \rangle$ (called *concept assertion*, individual a is an instance of concept C to degree greater than or equal to $\alpha \in (0,1]$) 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_1, a_2 are individuals.

A *TBox* $\mathcal{T}$ is a finite set of fuzzy *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 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 class name, a *rule*, and C its *body*. A *Knowledge Base* (KB), also called *ontology*, is a pair $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$ is a TBox and $\mathcal{A}$ is an ABox. $\mathsf{I}_\mathcal{K}$ denotes the set of individuals occurring in $\mathcal{K}$.

Concerning the semantics, an interpretation $\mathcal{I}$ maps a concept 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 $A^\mathcal{I}\colon \Delta^\mathcal{I} \to [0, 1]$; *(ii)* to each object property r a function $r^\mathcal{I}\colon \Delta^\mathcal{I} \times \Delta^\mathcal{I} \to [0, 1]$; *(iii)* to each datatype property s a function $s^\mathcal{I}\colon \Delta^\mathcal{I} \times \Delta^\mathbf{D} \to [0, 1]$; *(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 is extended to concepts as specified below (where $x \in \Delta^\mathcal{I}$) [3, 4, 18]:

$$
\begin{aligned}
\top^\mathcal{I}(x) &= 1 \\
\bot^\mathcal{I}(x) &= 0 \\
(C \sqcap D)^\mathcal{I}(x) &= \min\{C^\mathcal{I}(x), D^\mathcal{I}(x)\} \\
(\neg A)^\mathcal{I}(x) &= 1 - A^\mathcal{I}(x) \\
(\exists r.C)^\mathcal{I}(x) &= \sup_{y \in \Delta^\mathcal{I}} \{\min\{r^\mathcal{I}(x, y), C^\mathcal{I}(y)\}\} \\
(\exists s.\mathbf{d})^\mathcal{I}(x) &= \sup_{y \in \Delta^\mathbf{D}} \{\min\{s^\mathcal{I}(x, y), \mathbf{d}^\mathbf{D}(y)\}\} \\
(@_i(A_1, \ldots, A_{n_i}))^\mathcal{I}(x) &= @_i({A_1}^\mathcal{I}(x), \ldots, {A_{n_i}}^\mathcal{I}(x)) \ .
\end{aligned}
$$

We will assume that an interpretation is *witnessed*, *i.e.* the supremum is attained at some point of the domain [3, 4, 11], *e.g.* $(\exists r.C)^\mathcal{I}(x) = \min\{r^\mathcal{I}(x, y), C^\mathcal{I}(y)\}$ for some $y \in \Delta^\mathbf{D}$.

The *satisfiability of axioms* is then defined by the following conditions: *(i)* $\mathcal{I}$ satisfies $\langle a{:}C, \alpha \rangle$ if $C^\mathcal{I}(a^\mathcal{I}) \geq \alpha$; *(ii)* $\mathcal{I}$ satisfies $\langle (a, b){:}r, \alpha \rangle$ if $r^\mathcal{I}(a^\mathcal{I}, b^\mathcal{I}) \geq \alpha$;

(iii) $\mathcal{I}$ satisfies $\langle C \sqsubseteq D, \alpha\rangle$ if for all $x \in \Delta^{\mathbf{D}}$, $D(x) \geq C^{\mathcal{I}}(x) \cdot \alpha^{\mathcal{I}}$. $\mathcal{I}$ 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 every model of $\mathcal{K}$ satisfies τ. The *best entailment degree* of τ of the form $C \sqsubseteq D$, $a{:}A$ or $(a,b){:}R$, denoted $bed(\mathcal{K}, \tau)$, is defined as

$$bed(\mathcal{K}, \tau) = \sup\{\alpha \mid \mathcal{K} \models \langle \tau, \alpha\rangle\} \ . \tag{1}$$

Now, consider concept C, a KB $\mathcal{K}$ and a finite 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

$$|C|^{\mathsf{I}}_{\mathcal{K}} = \sum_{a \in \mathsf{I}} bed(\mathcal{K}, a{:}C) \ . \tag{2}$$

The *crisp cardinality* (denoted $\lceil C \rceil^{\mathsf{I}}_{\mathcal{K}}$) is defined similarly by replacing in Eq. 2 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

We are going now to present PN-OWL. At first, we formally present the learning problem we will address, then present our algorithm conceptually and eventually formalize PN-OWL.

3.1. The Learning Problem

We illustrate the learning problem using an example (see also [14, 19, 20] for analogous examples).

Example 3.1. *For the sake of illustrative purposes, let us consider a structural simple OWL 2 ontology that describes the meaningful entities of mammography screening data. The target is the prediction of breast cancer [21]. An excerpt of this ontology is given in Fig. 2, while an excerpt of the patient data is given in Table 1. Patients who have breast cancer are called* positive *examples, while those who do not are called* negative *examples.*

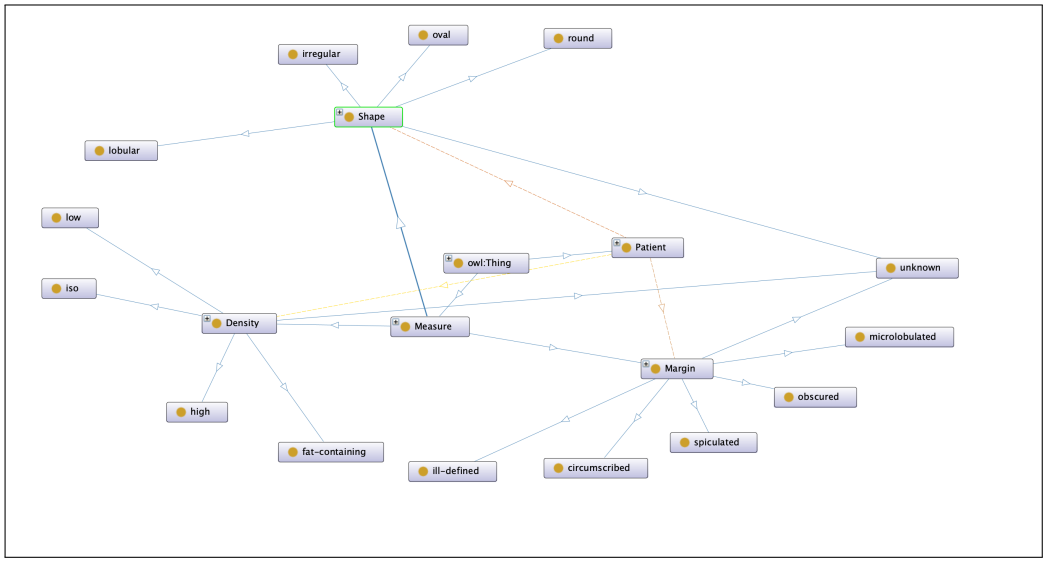

Fig. 2. Excerpt of the mammography ontology.

Table 1
Excerpt of the mammographic patient data: '-' indicates that the value of the 'attribute' is unknown.

Patient	hasDensity	hasShape	hasMargin	hasBiRads	hasAge	...
p0	low	lobular	spiculated	5	67	...
p10	high	irregular	spiculated	5	76	...
p102	-	irregular	ill-defined	4	58	...
p108	low	round	circumscribed	4	57	...
p109	-	irregular	ill-defined	5	33	...
p110	low	irregular	ill-defined	4	45	...
p111	low	irregular	ill-defined	5	71	...
...	...	...	...	...	...	...

The set $\mathsf{I}_{\mathcal{E}^+}$ *of positive and the set* $\mathsf{I}_{\mathcal{E}^-}$ *of negative examples/individuals contain*

$$\{\mathtt{p0}, \mathtt{p10}, \mathtt{p109}, \mathtt{p111}\} \subseteq \mathsf{I}_{\mathcal{E}^+} \tag{3}$$

$$\{\mathtt{p102}, \mathtt{p108}, \mathtt{p110}\} \subseteq \mathsf{I}_{\mathcal{E}^-} \,. \tag{4}$$

Now, our goal is to determine what characterizes the patients with cancer (our target class T*). That is, we would like to induce OWL 2 class expressions whose instances include as many positive examples as possible and as few*

negative examples as possible.

Now, assuming that all positive individuals are instances of the class `ToLearn`*, while all others are instances of the class* `NotToLearn` *(see Fig. 2), then one may learn from the ontology the following fuzzy concept/class inclusion rule, expressed in fuzzyDL syntax [22]:*

```
(implies (and (some hasMargin ill-defined)          | if the margin is ill-defined,
              (some hasShape irregular)             | the shape is irregular,
              (some hasAge hasAge_high))            | and the age is high
ToLearn 0.853)                                      | then it is breast cancer
                                                    | with confidence 0.853
```

where `hasAge_high` *is the fuzzy set (see Fig. 3)*

```
(define-fuzzy-concept hasAge_high right-shoulder (0,150,54,67)) .
```

For instance, as we will see later on, by referring to the patient data in Table 1, by this rule patient `p111` *will correctly be classified, while patient* `p102` *will incorrectly be classified as having breast cancer. The confidence value of* 0.853 *expresses the percentage of instances correctly classified among all those that satisfy the antecedent of this rule. This metric is also referred to as the* precision *of the rule.*

The left-hand expression in the learnt rule is expressed in DL syntax as the concept

$$\begin{array}{l}\exists \texttt{hasMargin.ill-defined} \sqcap \\ \exists \texttt{hasShape.irregular} \sqcap \\ \exists \texttt{hasAge.hasAge_high}\end{array} \tag{5}$$

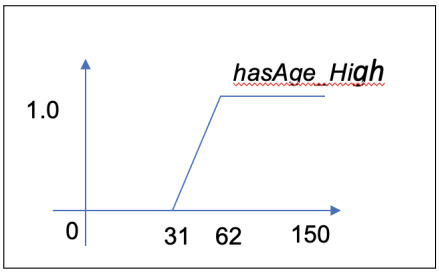

Fig. 3. Fuzzy set `hasAge_high`.

and the learnt rule can be expressed as

$$\begin{array}{l}\langle \exists \texttt{hasMargin.ill-defined} \sqcap \\ \exists \texttt{hasShape.irregular} \sqcap \\ \exists \texttt{hasAge.hasAge_high} \sqsubseteq \texttt{ToLearn}, 0.853\rangle\end{array} \tag{6}$$

The TBox of the mammography KB contains GCIs such as

$$\begin{array}{rcl}\texttt{Measure} & \sqsubseteq & \texttt{Margin} \\ \texttt{Margin} & \sqsubseteq & \texttt{ill-defined} \\ \texttt{Margin} & \sqsubseteq & \texttt{spiculated}\end{array}$$

The patient p102 *can be encoded as the concept assertion*

$$\begin{array}{l}\texttt{p102} : \texttt{Patient} \sqcap \exists \texttt{hasShape.irregular} \sqcap \\ \qquad \exists \texttt{hasMargin.ill-defined} \sqcap \\ \qquad \exists \texttt{hasAge}.\, =_{58} \sqcap \exists \texttt{hasBiRads}.\, =_{4} .\end{array} \tag{7}$$

Furthermore, let BC *be the concept in Eq. 5 and* $\mathcal{K}$ *be the mammography KB, then it can be shown that*

$$bed(\mathcal{K}, \texttt{p102}\!:\, \exists \texttt{hasAge.hasAge_high}) \;=\; 0.308$$

as $\texttt{hasAge_high}(58) = 0.308$ *and, thus,*

$$bed(\mathcal{K}, \texttt{p102}\!:\!\texttt{BC}) \;=\; 0.308 \; .$$

Moreover, if we add the learnt rule $\langle \texttt{BC} \sqsubseteq \texttt{ToLearn}, 0.853\rangle$ *to* $\mathcal{K}$, i.e. *the rule in Eq. 6, then we will infer that*

$$bed(\mathcal{K} \cup \{\langle \texttt{BC} \sqsubseteq \texttt{ToLearn}, 0.853\rangle\}, \texttt{p102}\!:\!\texttt{ToLearn}) \;=\; 0.308 \cdot 0.853 = 0.262 \; .$$

That is, we classify patient p102 *as having breast cancer to degree* 0.262. *Similarly, one may verify that*

$$\begin{array}{rcl} bed(\mathcal{K} \cup \{\langle \texttt{BC} \sqsubseteq \texttt{ToLearn}, 0.853\rangle\}, \texttt{p109}\!:\!\texttt{ToLearn}) & = & 0.0 \\ bed(\mathcal{K} \cup \{\langle \texttt{BC} \sqsubseteq \texttt{ToLearn}, 0.853\rangle\}, \texttt{p110}\!:\!\texttt{ToLearn}) & = & 0.0 \\ bed(\mathcal{K} \cup \{\langle \texttt{BC} \sqsubseteq \texttt{ToLearn}, 0.853\rangle\}, \texttt{p111}\!:\!\texttt{ToLearn}) & = & 0.853 \; . \end{array}$$

It can easily also be verified that

$$\begin{array}{rcl} |\texttt{BC}|^{\mathcal{I}_\mathcal{K}}_{\mathcal{K}} & \geq & 0.262 + 0.0 + 0.0 + 0.853 = 1.115 \\ |\texttt{BC} \sqcap \texttt{ToLearn}|^{\mathcal{I}_\mathcal{K}}_{\mathcal{K}} & \geq & 0.853 \end{array}$$

and that $|\mathtt{BC} \sqcap \mathtt{ToLearn}|_{\mathcal{K}}^{\mathsf{I}_{\mathcal{K}}} \leq |\mathtt{BC}|_{\mathcal{K}}^{\mathsf{I}_{\mathcal{K}}}$.

Eventually, the quantity

$$\frac{|\mathtt{BC} \sqcap \mathtt{ToLearn}|_{\mathcal{K}}^{\mathsf{I}_{\mathcal{K}}}}{|\mathtt{BC}|_{\mathcal{K}}^{\mathsf{I}_{\mathcal{K}}}} \quad = \quad 0.853$$

is the confidence/precision of the rule $\mathtt{BC} \sqsubseteq \mathtt{ToLearn}$.

□

Our learning problem is formalised as follows. Considering

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

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

4. the partitioning of the examples into

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

   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 $\mathcal{E}' = \mathcal{E} \setminus \mathcal{E}^+$ 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

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

   For each positive individual $a \in \mathsf{I}_{\mathcal{E}^+}$ there is an assertion $a{:}T$ in $\mathcal{K}$, while for each negative individual $a \in \mathsf{I}_{\mathcal{E}^-}$ there is an assertion $a{:}\mathtt{Not}T$ in $\mathcal{K}$, where $\mathtt{Not}T$ is a concept name not occurring anywhere else in $\mathcal{K}$;

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* $\mathcal{E}_{test} = \mathcal{E} \setminus \mathcal{E}_{train}$, with with $\mathcal{E}_{test} \cap \mathcal{E}^+ \neq \emptyset$.

Example 3.2. *Consider Example 3.1. It can be verified that the mammography KB is satisfiable. The target class T is the class* `ToLearn` *whose instances are the patients having breast cancer,* i.e. *the set of positive individuals $\mathsf{I}_{\mathcal{E}^+}$ are instances of* `ToLearn` *and contains* $\{\mathtt{p0}, \mathtt{p10}, \mathtt{p109}, \mathtt{p111}\}$*, while the set of negative individuals is $\mathsf{I}_{\mathcal{E}^-} = \mathsf{I}_{\mathcal{K}} \backslash \mathsf{I}_{\mathcal{E}^+}$,* i.e. *the non-positive individuals, and contains* $\{\mathtt{p102}, \mathtt{p108}, \mathtt{p110}\}$*, which are also instances of the class* `NotToLearn`*.* □

We write

$$\begin{aligned} \mathcal{E}(a) &= 1 \quad \text{if } a \in \mathsf{I}_{\mathcal{E}^+} \\ \mathcal{E}(a) &= 0 \quad \text{if } a \in \mathsf{I}_{\mathcal{E}^u} \\ \mathcal{E}(a) &= -1 \quad \text{if } a \in \mathsf{I}_{\mathcal{E}^-} \ . \end{aligned}$$

The general goal is to learn a classifier function $\bar{h} \in \mathcal{H}$ that is the result of *Empirical Risk Minimisation* (ERM) on $\mathcal{E}_{train}$ [14]:

$$\begin{aligned} \bar{h} &= \arg\min_{h \in \mathcal{H}} R(h, \mathcal{E}_{train}) \\ &= \frac{1}{|\mathcal{E}_{train}|} \sum_{a \in \mathsf{I}_{\mathcal{E}_{train}}} L(h(a), \mathcal{E}_{train}(a)) \ , \end{aligned} \tag{8}$$

where L is a *loss function* that measures how different the prediction of a hypothesis is from the true label 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 $R(\bar{h}, \mathcal{E}_{test})$ on the test set $\mathcal{E}_{test}$ [14].

In our learning setting, a hypothesis $h \in \mathcal{H}$ is a set of fuzzy GCIs/rules

of the form

$$\langle C_1 \sqsubseteq P_1, \alpha_1 \rangle \ , \ldots \ , \langle C_h \sqsubseteq P_h, \alpha_h \rangle \tag{9}$$

$$@^+(P_1, \ldots, P_h) \sqsubseteq P \tag{10}$$

$$\langle D_1 \sqsubseteq N_1, \beta_1 \rangle \ , \ldots \ , \langle D_k \sqsubseteq N_k, \beta_k \rangle \tag{11}$$

$$@^-(N_1, \ldots, N_k) \sqsubseteq N \tag{12}$$

$$@(P, N) \sqsubseteq T \ , \tag{13}$$

where each P_i, P, N_j, N are new class 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 [23] 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 or not an individual is an instance of T (*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 is the one we will adopt:

> *($\star$) if the degree p of being positive is greater than the degree n of being non-positive 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

$$h(a) = bed(\mathcal{K} \cup h, a{:}T) \ . \tag{14}$$

Remark 1. *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. □

Example 3.3. *Consider Example 3.2. As we have seen, a P-rule example following Eq. 9 is rule p_1 defined as*

$$\langle \exists \texttt{hasMargin.ill-defined} \sqcap \exists \texttt{hasShape.irregular} \sqcap \exists \texttt{hasAge.hasAge_high} \sqsubseteq \texttt{P}_1, 0.853 \rangle \tag{15}$$

while an example of N-rule following Eq. 11 is rule n_1 defined as

$$\langle \exists \texttt{hasDensity.low} \sqcap \exists \texttt{hasAge.hasAge_medium} \sqcap \exists \texttt{hasBiRads.hasBiRads_medium} \sqsubseteq \texttt{N}_1, 1.0 \rangle \ , \tag{16}$$

where `hasAge_medium` *is defined as $tri(40, 54, 67)$ and* `hasBiRads_medium` *is defined as $tri(3, 4, 5)$. Now, by adding rules p_1, n_1 to the KB one may verify that*

$$\begin{aligned} bed(\mathcal{K} \cup \{p_1, n_1\}, \texttt{p102}{:}\texttt{P}_1) &= 0.262 \\ bed(\mathcal{K} \cup \{p_1, n_1\}, \texttt{p102}{:}\texttt{N}_1) &= 0.692 \\ bed(\mathcal{K} \cup \{p_1, n_1\}, \texttt{p111}{:}\texttt{P}_1) &= 0.853 \\ bed(\mathcal{K} \cup \{p_1, n_1\}, \texttt{p111}{:}\texttt{N}_1) &= 0.0 \ . \end{aligned}$$

Therefore, by applying ($\star$) we would correctly classify patient `p102` *as not having breast cancer, while correctly* `p111` *does so.* □

Remark 2. *The set of hypotheses defined by this syntax is potentially infinite due to conjunction and the nesting of existential restrictions in the concept expressions. As we will see later on, we will impose further restrictions such as the maximum number of conjuncts and the maximum 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 rule $C \sqsubseteq T$, the *confidence degree* (also called the *precision*) 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

$$cf(C \sqsubseteq T, \mathcal{K}, P) = \frac{|C|^P_{\mathcal{K}}}{|C|^{\mathsf{I}_{\mathcal{K}}}_{\mathcal{K}}} \ , \tag{17}$$

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 $C \sqsubseteq T$, *i.e.* the less non-positive individuals are covered by it. 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

$$supp(C \sqsubseteq T, \mathcal{K}, \mathsf{I}) = \frac{|C|^{\mathsf{I}}_{\mathcal{K}}}{|\mathsf{I}|} \ . \tag{18}$$

3.2. The Learning Method

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

In the beginning, 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 $\mathsf{I}_{\mathcal{E}^u}$ (Fig. 4 (a)).

At the first stage, the P-stage, we consider the entire training set $\mathcal{E}_{train} = \mathcal{E}$ and try to maximize the covering of positive individuals while minimizing the covering of non-positive 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 Fig. 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 as in Eqs. 9-10. For instance, by referring to Example 3.3, one may learn at the P-stage the rule p_1 covering individuals `p102` and `p111`. The latter is a true positive, while the former is a false positive.

In the next stage, the N-stage, to increase the effectiveness of the classifiers, we try to remove as many 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: the new target class is FP, where the negative

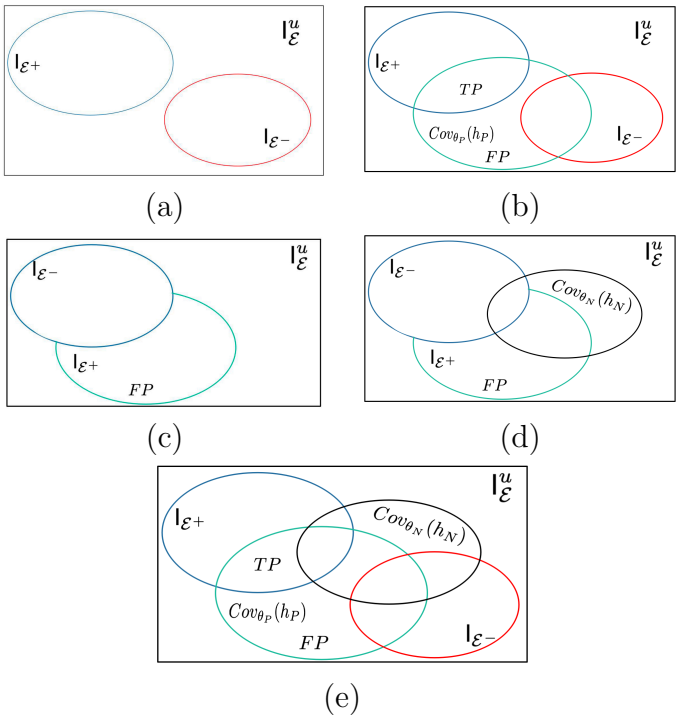

Fig. 4. How PN-OWL works: (a) Original training set; (b) Coverage $Cov_{\theta_P}(h_P)$ w.r.t. learnt hypothesis h_P after the P-stage; (c) Starting dataset for N-stage: the new target class is the false positives FP of the P-stage, while the negative individuals are the initial positives; (d) Coverage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

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 Fig. 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 Fig. 4 (d). Note that we may have another parameter θ_N as a confidence threshold for the learnt rules in hypothesis h_N. This phase ends with a set of rules of the form as in Eqs. 11-12. So, in general, at the end of the two stages, the situation may be as depicted in Fig. 4 (e). For instance, continuing referring to Example 3.3, one may learn at the N-stage the rule n_1 covering individual `p102` only.

Eventually, we aggregate the P-rules and N-rules via ($\star$). This latter step ends with the rule of the form as in Eq. 13. Referring to Example 3.3, this would allow us to correctly remove patient `p102` from those classified as

having breast cancer.

3.3. The Learning Algorithm PN-OWL

We now formalize PN-OWL, which we have conceptually illustrated in the section before. Essentially, at the P-stage (resp. N-stage) our algorithm invokes 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

$$h_P = \{\langle C_1 \sqsubseteq T, \alpha_1\rangle, \ldots, \langle C_h \sqsubseteq T, \alpha_h\rangle\} \quad (19)$$

$$h_N = \{\langle D_1 \sqsubseteq FP, \beta_1\rangle, \ldots, \langle D_k \sqsubseteq FP, \beta_k\rangle\} \quad (20)$$

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 aggregation operator ($\star$).

As stage learner, we will use fuzzy FOIL-$\mathcal{DL}$ [14, 19, 24]. For completeness of the presentation, we will recap succinctly fuzzy FOIL-$\mathcal{DL}$ in the appendix.

The PN-OWL algorithm is shown in Algorithm 1. Note that the P-stage corresponds to steps 1-5, while the N-stage corresponds to steps 15-19 in which at step 19 we invoke the stage learner trying to cover as many false positives as possible. The remaining steps deal with the construction of the final classifier ensemble as per Eqs. 9-13.

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 hypothesis returned by PN-OWL. Moreover, we say that PN-OWL *classifies* a as instance of target class T if $h(a) > 0$.

4. Evaluation

We have implemented the algorithm within the *FuzzyDL-Learner* system [25].

Datasets. Several OWL 2 ontologies from different domains have been selected as illustrated in Table 2, which reports 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 other individuals are non-positive and we set $\mathcal{E}^- = \mathcal{E}' = \mathsf{I}_\mathcal{K} \setminus \mathsf{I}_{\mathcal{E}^+}$.

Algorithm 1 PN-OWL

Input: KB $\mathcal{K}$, training set $\mathcal{E}$, target concept name T, set of parameters (see Table 3)
Output: Hypothesis h as by Eq. 9-13.

// P-stage
$Pos \leftarrow \mathsf{I}_{\mathcal{E}^+}$;
$Neg \leftarrow \mathsf{I}_{\mathcal{E}^-}$;
$U \leftarrow \mathsf{I}_{\mathcal{K}} \setminus (Pos \cup Neg)$;
$h_P \leftarrow$ FuzzyStageLearner($\mathcal{K}$, T, Pos, Neg, U, θ_P, η_P); ▷ P-Stage hypothesis h_P, *i.e.* set of axioms of the form $\langle C_i \sqsubseteq T, \alpha_i \rangle$
if $h_P = \emptyset$ **then return** $\emptyset$; ▷ Nothing learnt, exit
$Cov \leftarrow Cov_{\theta_P}(h_P)$; ▷ P-stage Coverage
$TP \leftarrow Cov_{\theta_P}(h_P) \cap \mathsf{I}_{\mathcal{E}^+}$; ▷ True positives
$FP \leftarrow Cov_{\theta_P}(h_P) \setminus \mathsf{I}_{\mathcal{E}^+}$; ▷ False positives
// Start building classifier h
$h \leftarrow \{\langle C_i \sqsubseteq P_i, \alpha_i \rangle, \mid \langle C_i \sqsubseteq T, \alpha_i \rangle \in h_P, P_i \text{ new } \}$; ▷ As per Eq. 9
if $FP = \emptyset$ **then** ▷ No N-stage, exit with aggregated h_P
    $h \leftarrow h \cup \{@^+(P_1, \ldots, P_h) \sqsubseteq T\}$; ▷ No need of new P in Eq. 10
    return h;
// N-stage
$Pos \leftarrow FP$;
$Neg \leftarrow \mathsf{I}_{\mathcal{E}^+}$;
$U \leftarrow \mathsf{I}_{\mathcal{K}} \setminus (Pos \cup Neg)$;
$h_N \leftarrow$ FuzzyStageLearner($\mathcal{K}$, FP, Pos, Neg, U, θ_N, η_N); ▷ N-Stage hypothesis h_N, *i.e.* set of axioms of the form $\langle D_j \sqsubseteq FP, \beta_j \rangle$
// Build final classifier ensemble h
if $h_N = \emptyset$ **then** ▷ No learning in N-stage, return aggregated h_P
    $h \leftarrow h \cup \{@^+(P_1, \ldots, P_h) \sqsubseteq T\}$; ▷ No need of new P in Eq. 10
    return h;
$h \leftarrow h \cup \{@^+(P_1, \ldots, P_h) \sqsubseteq P \mid P \text{ new } \}$; ▷ As per Eq. 10
$h \leftarrow h \cup \{\langle D_j \sqsubseteq N_j, \beta_j \rangle, \mid \langle D_j \sqsubseteq FP, \beta_j \rangle \in h_N, N_j \text{ new } \}$; ▷ As per Eq. 11
$h \leftarrow h \cup \{@^-(N_1, \ldots, N_k) \sqsubseteq N \mid N \text{ new } \}$; ▷ As per Eq. 12
$h \leftarrow h \cup \{@(P, N) \sqsubseteq T\}$; ▷ As per Eq. 13
return h;

The ontologies `Iris`, `Wine`, `Wine Quality` and `Yeast` are built from the well-known *UC Irvine Machine Learning Repository* (UCIMLR) [26] and have been transformed from the CSV format, provided by that repository, into OWL 2 ontologies according to the procedure described in [14].

All other ontologies, except `malware`, belong to the well-known SML-Bench dataset [21]. The `malware` ontology has been described in [27].

Remark 3. *Note that all but* `Lymphography` *have datatype properties. While evaluating ontology-based learning algorithms is untypical on numerical datatype properties, 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,* i.e. *the explainability of the classification decision*

Table 2
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

process.

Measures. We considered the following well-known effectiveness measures (see also [14]), which we recap here for the sake of completeness. 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 if $h(a) > 0$. As all individuals are either positive or non-positive, we will consider the following measures, all of which are based on crisp cardinality.

The *True Positives*, denoted TP, is defined as the number of instances of T that are positive, *i.e.* $TP = \lceil T \rceil_{\mathcal{K}}^{\mathsf{I}_{\mathcal{E}^+}}$. The *False Positives*, denoted FP, is defined as the number of instances of T that are not positive, *i.e.* $FP = \lceil T \rceil_{\mathcal{K}}^{\mathsf{I}_{\mathcal{E}'}}$. *Precision*, or *Confidence*, denoted P, is defined as the fraction of true positives w.r.t. the covered examples of h, *i.e.* $P = \frac{TP}{\lceil T \rceil_{\mathcal{K}}^{\mathsf{I}_{\mathcal{E}}}}$. *Recall*, denoted R, is defined as fraction of true positives w.r.t. all positives, *i.e.* $R = \frac{TP}{|\mathsf{I}_{\mathcal{E}^+}|}$ and $F1$-*score*, denoted $F1$, is defined as $F1 = 2 \cdot \frac{P \cdot R}{P+R}$.

Parameters. Some of the salient parameters used within our algorithm are reported in Table 3 of which the maximal role depth (usually, $d_P = d_N$) is determined manually a priori by inspecting the ontology. For the other

Table 3
Some salient parameters of the PN-OWL algorithm.

param	**description**
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

parameters, we generated automatically via random grid search various parameter configurations $\langle \theta_P, \theta_N, \eta_P, \eta_N, c_P, c_N, d_P, d_N \rangle$ by varying confidence thresholds, non-positive coverage percentage, and maximal number of conjuncts.

For each parameter configuration, a stratified k-fold cross-validation design (i.e. each fold contains the same proportions of positive and non-positive instances of T) was adopted (specifically, $k = 5$) to determine the macro average of the above-described effectiveness measures.

In all tests, we have that $\mathsf{I}_{\mathcal{E}} = \mathsf{I}_{\mathcal{K}}$ and that 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 experimental data is available from the downloadable resource [1]. The resource contains the executable code, the ontologies, and each ontology, the parameter configuration that produced the best test macro average F1-score together with learnt rules and effectiveness measures for each fold.

It is worth noting that a typical parameter configuration is as follows, but may vary depending on the ontology.

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

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

[1] http://www.umbertostraccia.it/cs/software/FuzzyDL-Learner/download/pnowl.experiments.zip

Let us briefly comment on it. During the P-stage, we would like to increase recall, that is the percentage of covered positives w.r.t. all positives, and then use the N-stage to remove the false positive. In this way, we ultimately increase recall and precision and, thus, the F1-score. So, 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 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.

We recall that for $@^+, @^-$ (resp. $@$) we used max (resp. $(\star)$). Also, for concept conjunction $\sqcap$ we used the t-norm min. These could well be another set of parameters: however, the parameter space is already quite large, so we fixed these logical operators as specified. We will investigate the impact of other choices for these parameters in future work.

Concerning some 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 from the experience we gained in previous experiments [14, 19] and taking into account that clustering is done for each single data property.

Remark 4. *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* $h(a) > \tau_p$. *In this paper, we will always have* $\tau_p = 0$. *More generally, if we would like to distinguish the negative examples from the unlabelled ones, we may try to 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.* □

As baseline, we consider the learner Fuzzy FOIL-$\mathcal{DL}$ [19, 24], with best parameter setup as specified in [14]. 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 the effectiveness of the N-stage phase.

Table 4
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	**0.810**	**0.803**	**0.805**	**3.48%**
	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**	

The results are reported in Table 4. For the UCIMLR datasets, in the case of multiple targets, the average of the measures has been considered.

Discussion. In Table 4, the last column reports the improvement of PN-OWL relative to the measure $F1$ over our baseline Fuzzy FOIL-$\mathcal{DL}$. Overall, despite the algorithm is conceptually simple, PN-OWL outperforms Fuzzy FOIL-$\mathcal{DL}$ 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 via the N-stage, after increasing recall via the P-stage confirming our conjecture that indeed the N-stage can remove the false positives.

The overall lesson learnt with PN-OWL is that indeed the N-stage may provide a non-negligible contribution to improve the effectiveness of the classification process provided one may find the appropriate balance between 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) thus, we used a random grid search. On the other hand, optimizing FOIL-$\mathcal{DL}$ is much easier as it has half of the parameters of PN-OWL.

5. Related Work

The problem of learning GCIs, addressed in works such as [8, 28, 29, 30, 31, 32, 33], is typically inspired by statistical relational learning, where classification rules are (possibly weighted) Horn clause theories [6]. The goal is to learn a concept description of the underlying DL language covering (possibly) all positive examples and (possibly) not covering any of the provided negative examples. The fuzzy case (see [19, 20, 24]) is a natural extension in which one relies on fuzzy DLs [3, 4] and fuzzy ILP (see *e.g.* [34]): while, in the former works the authors propose *fuzzy* FOIL-like algorithms that are inspired by fuzzy ILP variants such as [35, 36, 37], in other works such as e.g. [38, 39] they adapt the well-known FOIL-algorithm to the DLs case. As already mentioned, our two-stage algorithm follows conceptually PN-rule [9, 10], and consists of a P-stage trying to cover as many as possible instances of a target class and an N-stage trying to remove most of the non-positive examples covered by the P-stage. It is worth noting that what differentiates this method from all others is (typically) its second stage. Moreover, the main differences of PN-OWL w.r.t. PN-rule are:

- 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 2 ontologies [2];
- 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, [3] while we have the conjunction of conditions in the rule body and each condition may be fuzzy, each rule is graded, and each conjunction may be either a class name or a restriction on attributes, where attributes may be also nested;
- 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 [5], while for the former we conjecture it cannot: so, we

[2] Tabular data can easily be mapped into OWL 2 ontologies as illustrated in; [14].
[3] If A is categorical then obviously $cond$ is either of the forms $A = v$ or $A \neq v$.

left this option out as a fuzzy DL reasoner would not be able to reason with those types of rules.

Discrete boosting has been considered in [40], while [14] is a real-valued fuzzy 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 [41, 42, 43, 44, 45].

The works [46, 47] consider *Kernel Methods* for inducing concept descriptions, [48, 49, 50] are inspired on *Decision Trees/Random Forests*, while [51, 52] consider essentially a *Naive Bayes* approach. Last but not least, [53] is inspired by *Genetic Programming* to induce concept expressions, while [54] is based on the *Reinforcement Learning* framework. Eventually, [55] proposes to use decision trees to learn so-called *disjointness axioms*, *i.e.* expressions of the form $C \sqcap D \sqsubseteq \bot$, declaring that class C and D are disjoint.

As pointed out in Remark 3, typically ontology-based learning algorithms as those listed above do not rely on numerical datatype properties (except [14, 19, 20]), we believe it is interesting to do so, not only to increase possibly the effectiveness but also to improve the human understandability, *i.e.* the explainability of the classification decision process (*cf.* Example 3.3).

6. Conclusions & Future Work

We have presented a two-stage algorithm, called PN-OWL that is inspired on the PN-rule [9, 10] 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.

All generated fuzzy concept inclusion axioms can be encoded as *Fuzzy OWL 2* axioms, allowing a fuzzy DL reasoner to reason with them.

We have also conducted an extensive evaluation, comparing it with fuzzy FOIL-$\mathcal{DL}$. Our evaluation shows that, despite being our algorithm conceptually simple, PN-OWL outperforms fuzzy FOIL-$\mathcal{DL}$ in terms of effective-

ness, though finding an optimal parameter configuration is much more time-consuming.

Concerning future work, besides investigating other learning methods, and future work mentioned in various places in the paper, we envisage various aspects worth to be investigated in more detail: *(i)* 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 investigate the impact, in terms of effectiveness, of efficient, logically sound, but not necessarily complete, reasoning algorithms; *(ii)* 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 [14] and to verify its effectiveness; and *(iii)* we would like to asses also the impact of other alternative scoring functions to information gain (*cf.* Eq. A.1) within our setting, inclusive various choices of t-norms and aggregation functions.

Eventually, we would like to consider extending the hypothesis language $\mathcal{EL}(\mathbf{D})$ with so-called *threshold concepts* [56] 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 a degree greater (resp. smaller) than or equal to d". On our ongoing implementation, as an example of a rule, by referring to the `Wine Quality` ontology and target wine number `1`, we generated an expression of the form

```
(implies (and (some alcohol alcohol_VH)[<= 0.786]
              (some sulfates sulfates_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 sulfates 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 an instance of the target class* `1`*"*. This would provide a more fine-grained hypothesis language in which a threshold may be defined for each conjunct of a rule rather than via rule confidence.

Algorithm 2 Fuzzy FOIL-$\mathcal{DL}$

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

Appendix A. The Stage Learner Fuzzy Foil-$\mathcal{DL}$

The procedure invocations FUZZYSTAGELEARNER at steps 5 and 19 in the Algorithm 1 are calls to fuzzy FOIL-$\mathcal{DL}$ [14, 19, 24], which we briefly recap in this section. Essentially, fuzzy FOIL-$\mathcal{DL}$ carries on inducing GCIs until as many positive examples are covered or nothing new can be learnt, 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 candidates;
3. score them 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.

As shown on Algorithm 2, when a rule is induced (step 3), the positive examples still to be covered are updated (step 9 and 10). To induce an axiom (step 3), LEARN-ONE-AXIOM is invoked (see Algorithm 3),

Refinement operator. The refinement operator is the same as in [14, 20, 24, 57]. It takes as input a concept C and generates a more specific concept

Table A.5
Downward Refinement Operator.

$$\rho(C) = \begin{cases} \mathbf{A}_{\mathcal{K}} \cup \{\exists r.\top \mid r \in \mathbf{R}_{\mathcal{K}}\} \cup \\ \quad \{\exists s.d \mid s \in \mathbf{S}_{\mathcal{K}}, d \in \mathcal{D}\} \cup \\ \quad \{\exists s. =_b, \mid s \in \mathbf{B}_{\mathcal{K}}, b \in \{\mathbf{t}, \mathbf{f}\}\} & \text{if } C = \top \\ \{A' \mid A' \in \mathbf{A}_{\mathcal{K}}, \mathcal{K} \models A' \sqsubseteq A\} \cup \\ \quad \{A \sqcap A'' \mid A'' \in \rho(\top)\} & \text{if } C = A \\ \{\exists r.D' \mid D' \in \rho(D)\} \cup \\ \quad \{(\exists r.D) \sqcap D'' \mid D'' \in \rho(\top)\} & \text{if } C = \exists r.D, r \in \mathbf{R}_{\mathcal{K}} \\ \{(\exists s.d) \sqcap D \mid D \in \rho(\top)\} & \text{if } C = \exists s.d, s \in \mathbf{S}_{\mathcal{K}}, \\ & \quad d \in \mathcal{D} \\ \{(\exists s. =_b) \sqcap D \mid D \in \rho(\top)\} & \text{if } C = \exists s. =_b, s \in \mathbf{B}_{\mathcal{K}}, \\ & \quad b \in \{\mathbf{t}, \mathbf{f}\} \\ \{C_1 \sqcap ... C'_i ... \sqcap C_n \mid C'_i \in \rho(C_i)\} & \text{if } C = C_1 \sqcap ... \sqcap C_n \end{cases}$$

description candidates D [4] (see Table A.5).

The GCI scoring function. The scoring function used in fuzzy FOIL-$\mathcal{DL}$ is the *Information Gain* function [14, 20, 24, 57]. 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

$$gain(\phi', \phi, \mathcal{K}, Pos) = p * (log_2(cf(\phi', \mathcal{K}, Pos)) - log_2(cf(\phi, \mathcal{K}, Pos))) \tag{A.1}$$

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

[4] $\mathbf{A}_{\mathcal{K}}$ is the set of all atomic concepts in $\mathcal{K}$, $\mathbf{R}_{\mathcal{K}}$ is the set of all object properties in $\mathcal{K}$, $\mathbf{S}_{\mathcal{K}}$ is the set of all numeric datatype properties in $\mathcal{K}$, $\mathbf{B}_{\mathcal{K}}$ is the set of all boolean datatype properties in $\mathcal{K}$ and $\mathcal{D}$ a set of (fuzzy) datatypes.

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 $C \sqsubseteq T$

1: $NP \leftarrow N \cup U$; ▷ Note: NP are the non-positives
2: $C \leftarrow \top$; ▷ Start from $\top$
3: $\phi \leftarrow C \sqsubseteq T$;
4: **while** $C \neq$ **null do**
▷ Loop until no improvement
5: $C_{best} \leftarrow C$;
6: $maxgain \leftarrow 0$;
7: $\mathcal{C} \leftarrow \rho(C)$; ▷ Compute all refinements of C
▷ Compute the score of the refinements and select the best one
8: **for all** $C' \in \mathcal{C}$ **do**
9: $\phi' \leftarrow C' \sqsubseteq T$;
10: $gain \leftarrow gain(\phi', \phi, \mathcal{K}, Pos)$;
11: **if** $(gain > maxgain)$ **then**
12: $maxgain \leftarrow gain$;
13: $C_{best} \leftarrow C'$;
14: **if** $C_{best} = C$ **then** ▷ No improvement
▷ Stop if the confidence degree is above the threshold or the non-positive coverage is below the threshold
15: **if** $(cf(C_{best} \sqsubseteq T, \mathcal{K}, P) \geq \theta)$ **and** $supp(C_{best} \sqsubseteq T, \mathcal{K}, NP) \leq \eta)$ **then break**;
16: $C \leftarrow C_{best}$;
17: $\phi \leftarrow C \sqsubseteq T$;
18: **return** ϕ;

The LEARN-ONE-AXIOM *algorithm.* The LEARN-ONE-AXIOM algorithm (see, *e.g.* [14, 20, 24]) is illustrated in Algorithm 3: lines 1 - 3 are simple initialisation steps. Please note here that NP are the non-positives (see Remark 1), which states that we will distinguish among positives and non-positives only (*cf.* also line 15, where the non-positive coverage percentage is used). Steps 4-18 are the main loop from which we may exit if the stopping criterion is satisfied, in step 7 we compute all new refinements, which are then scored in steps 8-13 and update the best one. At the end of the algorithm, once we exit from the main loop, the best-found GCI is returned (step 19). The LEARN-ONE-AXIOM algorithm 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.

References

[1] OWL 2 Web Ontology Language Document Overview, https://www.w3.org/TR/owl2-overview/ (2012).

[2] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider, The

Description Logic Handbook: Theory, Implementation and Applications, 2nd Edition, Cambridge University Press, 2007.

[3] F. Bobillo, M. Cerami, F. Esteva, À. García-Cerdaña, R. Peñaloza, U. Straccia, Fuzzy description logics, in: Handbook of Mathematical Fuzzy Logic, Volume 3, 2015, Ch. XVI, pp. 1105–1181.

[4] U. Straccia, Foundations of Fuzzy Logic and Semantic Web Languages, Chapman & Hall/CRC, 2013.

[5] F. Bobillo, U. Straccia, Fuzzy ontology representation using OWL 2, International Journal of Approximate Reasoning 52 (2011) 1073–1094.

[6] L. D. Raedt, K. Kersting, Statistical relational learning, in: Encyclopedia of Machine Learning and Data Mining, 2017, pp. 1177–1187.

[7] F. A. Lisi, Logics in machine learning and data mining: Achievements and open issues, in: Proc. of the 34th Italian Conference on Computational Logic, Trieste, Italy, June 19-21, 2019., Vol. 2396 of CEUR Workshop Proc., 2019, pp. 82–88.

[8] A. Rettinger, U. Lösch, V. Tresp, C. d'Amato, N. Fanizzi, Mining the semantic web - statistical learning for next generation knowledge bases, Data Minining and Knowledge Discovery 24 (3) (2012) 613–662.

[9] R. C. Agarwal, M. V. Joshi, PNrule: A new framework for learning classifier models in data mining (a case-study in network intrusion detection), in: Proc. of the 2001 SIAM International Conference on Data Mining (SDM), 2001, pp. 1–17.

[10] R. C. A. Mahesh V. Joshi, V. Kumar, Predicting rare classes: can boosting make any weak learner strong?, in: Proc. of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2002, pp. 297–306.

[11] P. Hájek, Making fuzzy description logics more general, Fuzzy Sets and Systems 154 (1) (2005) 1–15.

[12] G. J. Klir, B. Yuan, Fuzzy sets and fuzzy logic: theory and applications, Journal of Chemical Information and Computer Sciences (1996) 619–619.

[13] J. C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Springer New York, NY, 1981.

[14] F. A. Cardillo, U. Straccia, Fuzzy OWL-BOOST: Learning fuzzy concept inclusions via real-valued boosting, Fuzzy Sets and Systems 438 (2022) 164–186.

[15] I. Huitzil, U. Straccia, N. Díaz-Rodríguez, F. Bobillo, Datil: Learning fuzzy ontology datatypes, in: Information Processing and Management of Uncertainty in Knowledge-Based Systems. Theory and Foundations, 2018, pp. 100–112.

[16] F. Bobillo, U. Straccia, Aggregation operators for fuzzy ontologies, Applied Soft Computing 13 (9) (2013) 3816–3830.

[17] V. Torra, Y. Narukawa, Modeling Decisions: Information Fusion and Aggregation Operators, Springer Berlin, Heidelberg, 2007.

[18] U. Straccia, Reasoning within fuzzy description logics, Journal of Artificial Intelligence Research 14 (2001) 137–166.

[19] F. A. Lisi, U. Straccia, Learning in description logics with fuzzy concrete domains, Fundamenta Informaticae 140 (3-4) (2015) 373–391.

[20] U. Straccia, M. Mucci, pFOIL-DL: Learning (fuzzy) $\mathcal{EL}$ concept descriptions from crisp OWL data using a probabilistic ensemble estimation, in: Proc. of the 30th Annual ACM Symposium on Applied Computing (SAC-15), 2015, pp. 345–352.

[21] P. Westphal, L. Bühmann, S. Bin, H. Jabeen, J. Lehmann, SML-bench - A benchmarking framework for structured machine learning, Semantic Web 10 (2) (2019) 231–245.

[22] F. Bobillo, U. Straccia, The fuzzy ontology reasoner fuzzyDL (2016).
URL https://www.umbertostraccia.it/cs/software/fuzzyDL/fuzzyDL.html

[23] F. Bobillo, U. Straccia, Reasoning within fuzzy OWL 2 EL revisited, Fuzzy Sets and Systems 351 (2018) 1–40.

[24] F. A. Lisi, U. Straccia, A logic-based computational method for the automated induction of fuzzy ontology axioms, Fundamenta Informaticae 124 (4) (2013) 503–519.

[25] F. Bobillo, U. Straccia, Fuzzydl-learner (2016).
URL http://www.umbertostraccia.it/cs/software/FuzzyDL-Learner/

[26] D. Dua, C. Graff, UCI Machine Learning Repository (2017).
URL http://archive.ics.uci.edu/ml

[27] P. Svec, S. Balogh, M. Homola, Experimental evaluation of description logic concept learning algorithms for static malware detection, in: Proc. of the 7th International Conference on Information Systems Security and Privacy, 2021, pp. 792–799.

[28] L. Badea, S. Nienhuys-Cheng, A refinement operator for description logics, in: Inductive Logic Programming, 2000, pp. 40–59.

[29] M. Chitsaz, K. Wang, M. Blumenstein, G. Qi, Concept learning for $\mathcal{EL}^{++}$; by refinement and reinforcement, in: PRICAI 2012: Trends in Artificial Intelligence, 2012, pp. 15–26.

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

[31] J. Lehmann, P. Hitzler, Foundations of Refinement Operators for Description Logics, in: Inductive Logic Programming, 2008, pp. 161–174.

[32] J. Lehmann, P. Hitzler, Concept learning in description logics using refinement operators, Machine Learning 78 (2010) 203–250.

[33] F. A. Lisi, D. Malerba, Ideal refinement of descriptions in AL-Log, in: Inductive Logic Programming, 2003, pp. 215–232.

[34] M. Serrurier, H. Prade, Improving expressivity of inductive logic programming by learning different kinds of fuzzy rules, Soft Computing 11 (2007) 459–466.

[35] M. E. Cintra, M. C. Monard, H. 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, 2013, pp. 89–104.

[36] M. Drobics, U. Bodenhofer, E.-P. Klement, FS-FOIL: an inductive learning method for extracting interpretable fuzzy descriptions, International Journal of Approximate Reasoning 32 (2-3) (2003) 131–152.

[37] D. Shibata, N. Inuzuka, S. Kato, T. Matsui, H. Itoh, An induction algorithm based on fuzzy logic programming, in: Methodologies for Knowledge Discovery and Data Mining, 1999, pp. 268–274.

[38] N. Fanizzi, G. Rizzo, C. d'Amato, F. Esposito, DLFoil: Class expression learning revisited, in: Knowledge Engineering and Knowledge Management, 2018, pp. 98–113.

[39] G. Rizzo, N. Fanizzi, C. d'Amato, Class expression induction as concept space exploration: From DL-Foil to DL-Focl, Future Generation Computing Systems 108 (2020) 256–272.

[40] N. Fanizzi, G. Rizzo, C. d'Amato, Boosting DL concept learners, in: The Semantic Web, 2019, pp. 68–83.

[41] M. J. del Jesús, F. Hoffmann, L. J. Navascués, L. Sánchez, Induction of fuzzy-rule-based classifiers with evolutionary boosting algorithms, IEEE Trans. Fuzzy Syst. 12 (3) (2004) 296–308.

[42] J. Otero, L. Sánchez, Induction of descriptive fuzzy classifiers with the logitboost algorithm, Soft Computing 10 (9) (2006) 825–835.

[43] A. M. Palacios, L. Sánchez, I. Couso, Using the AdaBoost algorithm for extracting fuzzy rules from low quality data: Some preliminary results, in: IEEE International Conference on Fuzzy Systems, 2011, pp. 1263–1270.

[44] L. Sánchez, J. Otero, Boosting fuzzy rules in classification problems under single-winner inference, International Journal of Intelligent Systems 22 (9) (2007) 1021–1034.

[45] H.-y. Zhu, Y. Ding, H. Gao, W. Liu, Fuzzy prediction in classification of AdaBoost algorithm, in: International Conference on Oriental Thinking and Fuzzy Logic, Vol. 443 of Advances in Intelligent Systems and Computing, 2016, pp. 129–136.

[46] S. Bloehdorn, Y. Sure, Kernel methods for mining instance data in ontologies, in: The Semantic Web, Vol. 4825, 2007, pp. 58–71.

[47] N. Fanizzi, C. d'Amato, F. Esposito, Induction of robust classifiers for web ontologies through kernel machines, Journal of Web Semantics 11 (2012) 1–13.

[48] F. Nicola, C. d'Amato, E. Floriana, Towards the induction of terminological decision trees, in: Proc. of the 2010 ACM Symposium on Applied Computing, 2010, pp. 1423–1427.

[49] G. Rizzo, C. d'Amato, N. Fanizzi, F. Esposito, Tree-based models for inductive classification on the web of data, Journal of Web Semantics 45 (2017) 1–22.

[50] G. Rizzo, N. Fanizzi, C. d'Amato, F. Esposito, Approximate classification with web ontologies through evidential terminological trees and forests, International Journal of Approximate Reasoning 92 (2018) 340–362.

[51] P. Minervini, C. d'Amato, N. Fanizzi, Learning probabilistic description logic concepts: Under different assumptions on missing knowledge, in: Proc. of the 27th Annual ACM Symposium on Applied Computing, 2012, pp. 378–383.

[52] M. Zhu, Z. Gao, J. Z. Pan, Y. Zhao, Y. Xu, Z. Quan, Tbox learning from incomplete data by inference in BelNet+, Knoweledge-Based Systems 75 (2015) 30–40.

[53] J. Lehmann, Hybrid learning of ontology classes, in: Machine Learning and Data Mining in Pattern Recognition, 2007, pp. 883–898.

[54] M. Nickles, A. Rettinger, Interactive relational reinforcement learning of concept semantics, Machine Learning 94 (2) (2014) 169–204.

[55] G. Rizzo, C. d'Amato, N. Fanizzi, An unsupervised approach to disjointness learning based on terminological cluster trees, Semantic Web 12 (3) (2021) 423–447.

[56] U. S. Fernando Bobillo, The fuzzy ontology reasoner *fuzzyDL*, Knowledge-Based Systems 95 (2016) 12 – 34.

[57] F. A. Lisi, U. Straccia, Dealing with incompleteness and vagueness in inductive logic programming, in: 28th Italian Conference on Computational Logic (CILC-13), Vol. 1068, 2013, pp. 179–193.