Fuzzy Sets and Systems 490 (2024) 109048

Contents lists available at ScienceDirect

Fuzzy Sets and Systems

journal homepage: www.elsevier.com/locate/fss

PN-OWL: A two-stage algorithm to learn fuzzy concept inclusions 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

ARTICLE INFO

Keywords:
OWL 2 ontologies
Machine learning
Fuzzy logic
Concept/class inclusion rules

ABSTRACT

Given a target class T of an OWL 2 ontology, positive (and possibly negative) examples of T, we address the problem of learning, *viz.* inducing, from the examples, fuzzy class inclusion rules that aim to describe conditions for being an individual classified as an instance of the class T.

To do so, we present PN-OWL which is a two-stage learning algorithm consisting of a P-stage and an N-stage. In the P-stage, the algorithm learns fuzzy 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 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.

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 class inclusion rules from (non fuzzy) OWL 2 ontologies [7,8]. Specifically, given a target class T of an OWL 2 ontology and its positive (and possibly negative) examples, we address the problem of learning, *viz.* inducing, from the examples, fuzzy class inclusion rules that aim to describe conditions for being an individual classified as an instance of the class T.

* Corresponding author.
E-mail addresses: francoalberto.cardillo@ilc.cnr.it (F.A. Cardillo), franca.debole@isti.cnr.it (F. Debole), umberto.straccia@isti.cnr.it (U. Straccia).

https://doi.org/10.1016/j.fss.2024.109048
Received 7 July 2023; Received in revised form 5 June 2024; Accepted 6 June 2024

Available online 11 June 2024

To do so, we propose to rely on an adaptation of the PN-rule algorithm [9,10] to the 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 positive examples of the target class as possible (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.

We will also illustrate the effectiveness of PN-OWL by means of extensive experimentation on various OWL 2 ontologies.

The paper is organized as follows. In Section 2, we briefly recap 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 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 *Aggregation Operators*, *Fuzzy Sets*, *Fuzzy DLs* and crisp DLS we will use in this work. For a more in-depth presentation, see *e.g.*, [2–4,12,13,20].

Aggregation operators *Aggregation Operators* (AOs) are used to combine different pieces of information [20]. Numerous AOs differ on the assumptions of the data (data types) and the type of information we can incorporate in the model [20]. As far as we are concerned, we use an aggregation operator @ to decide whether an example has to be classified to be an instance of a target OWL 2 class T or not, based on the evidence p (determined by the P-rules) to be an instance of T and the evidence n (determined by the N-rules) not to be an instance of T. Therefore, for this work, an AO is a mapping $@ : [0,1]^2 \to \{0,1\}$, whose first argument is the positive evidence, while the second argument is the negative evidence. We impose the boundary conditions $@(1,0) = 1$ and $@(0,1) = 0$, and that @ is monotone non-decreasing in the first argument, while monotone non-increasing in the second argument.

Fuzzy sets A *fuzzy set* A over a finite crisp set X is a function $A : X \to [0,1]$, called *fuzzy membership* function of A [13]. 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 usual way to count the instances of a fuzzy set is to rely on the so-called *sigma-count* (see *e.g.*, [13,14]). That is, the *sigma-count* of a fuzzy set (also known as *scalar cardinality* or, for ease, *cardinality*) is defined as

$$|A| = \sum_{x \in X} A(x) \,. \tag{1}$$

The fuzzy *inclusion degree* of A in B we will rely on in this paper is defined as

$$inc(A,B) = \frac{|A \cap B|}{|A|} \tag{2}$$

if $|A| > 0$. If $|A| = 0$ then $inc(A,B)$ is left undefined. Please note that we consider the relation "is a subset of" as a *fuzzy (binary) relation* and not a crisp one.[1] That is, A is a fuzzy subset of B to some degree in $[0,1]$, which is determined by the inclusion degree of A in B.

Finally, the triangular, left-shoulder and right-shoulder (resp. *tri*, *ls* and *rs*) 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.*, [15]) 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 [16,17] (see (d) in Fig. 1). In [18] instead, aggregation operators are employed to combine (define the consensus among) fuzzy membership functions proposed by a group of experts.

Fuzzy DLs For illustrative purpose, we present here the fuzzy logic $\mathcal{ALC}(\mathbf{D})$, which is expressive enough to capture the main ingredients we are going to consider in this paper [2–4,11,12]. We recall that $\mathcal{ALC}$ is considered as a meaningful representative of the $\mathcal{AL}$ family of DL languages of which OWL 2 is an instance of (recap that the input to our learning algorithm will indeed be a crisp OWL 2 ontology).

We start with the notion of *fuzzy concrete domain*, $\mathbf{D} = \langle \Delta^{\mathbf{D}}, \Phi_{\mathbf{D}}, \cdot^{\mathbf{D}} \rangle$, where $\Delta^{\mathbf{D}}$ is a set of data values, $\Phi_{\mathbf{D}}$ is a set of 1-ary fuzzy datatype predicates, and $\cdot^{\mathbf{D}}$ is a mapping that assigns to every 1-ary fuzzy datatype predicate $\mathbf{d} \in \Phi_{\mathbf{D}}$ a fuzzy set over $\Delta^{\mathbf{D}}$ (*i.e.*, $\cdot^{\mathbf{D}}$ maps each fuzzy datatype predicate to a function from $\Delta^{\mathbf{D}}$ to $[0,1]$). In particular, $\mathbf{d}$ is a *crisp datatype predicate* if the range of $\mathbf{d}^{\mathbf{D}}$ is $\{0,1\}$ instead. A *crisp concrete domain* involves crisp datatype predicates only. Typical data values one may encounter in (crisp) OWL 2 ontologies are strings, numbers, dates, and Booleans. In the domain of numbers, typical fuzzy datatype predicates $\mathbf{d}$ are characterized by the well-known membership functions

$$\mathbf{d} \to ls(a,b) \mid rs(a,b) \mid tri(a,b,c) \mid \quad \geq_a \mid \leq_a \mid =_a \,,$$

[1] An n-ary *fuzzy relation* is a fuzzy set over X^n.

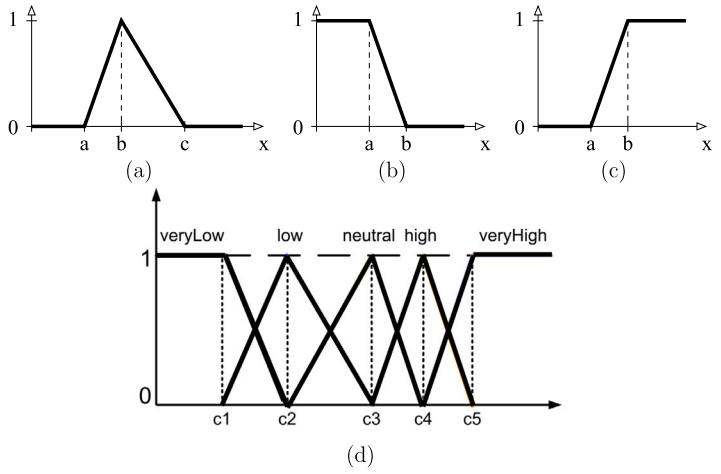

Fig. 1. (a) triangular function $tri(a,b,c)$, (b) left shoulder function $ls(a,b)$, (c) right shoulder function $rs(a,b)$, and (d) fuzzy sets over centroids.

where additionally $\geq_a$ (resp. $\leq_a$ and $=_a$) corresponds to the crisp set of data values that are no less than (resp. no greater than and equal to) the value a.[2]

We are now ready to define fuzzy $\mathcal{ALC}(\mathbf{D})$. 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* is built from concept names A, A_i using connectives and quantification constructs over object properties, datatype properties and is described by the following syntactic rule ($n \geq 1$):

$$C \to \top \mid \bot \mid A \mid \neg C \mid C_1 \sqcap C_2 \mid \exists r.C \mid \exists s.\mathbf{d} \ .$$

A *fuzzy assertion* axiom is an expression of the form $\langle a{:}C, \alpha\rangle$ (called *fuzzy 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 *fuzzy role assertion*, –the pair (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 *fuzzy General Concept Inclusion* (fuzzy GCI) axiom is of the form $\langle C_1 \sqsubseteq C_2, \alpha\rangle$ (C_1 is a subsumed by concept C_2 to degree greater than or equal to α), where C_i is a fuzzy $\mathcal{ALC}(\mathbf{D})$ concept and $\alpha \in (0,1]$.

A fuzzy GCI of the form $\langle C \sqsubseteq A, \alpha\rangle$, where A is a class name, may also be called a *fuzzy rule* with *body* C. A *fuzzy Knowledge Base* (KB), also called *fuzzy ontology*, is a finite set $\mathcal{K}$ of fuzzy axioms. With $\mathtt{I}_{\mathcal{K}}$ we denote the set of individuals occurring in $\mathcal{K}$.

Concerning the semantics, as our objective is to learn fuzzy rules from *crisp* (OWL) ontologies/KBs, we introduce here the notion of *strong fuzzy interpretation*. Essentially, a strong fuzzy interpretation is as a usual fuzzy interpretation (see, *e.g.*, [3,4]), except that now atomic concepts and roles are mapped into $\{0,1\}$ instead of $[0,1]$. Specifically, a *strong fuzzy interpretation*, or simply *interpretation* if clear from context, is a pair $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consisting of a nonempty (crisp) set $\Delta^{\mathcal{I}}$ (the *domain*, disjoint from $\Delta^{\mathbf{D}}$) and of a *strong interpretation function* (or simply *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 $s^{\mathcal{I}} : \Delta^{\mathcal{I}} \times \Delta^{\mathbf{D}} \to \{0,1\}$; and *(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*).

An interpretation is extended to concepts as specified below (where $x \in \Delta^{\mathcal{I}}$) [3,4,21]:

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

We will assume that an interpretation is *witnessed*, *i.e.*, the supremum is attained at some point of the domain [3,4,12], *e.g.*, $(\exists r.C)^{\mathcal{I}}(x) = \min\{r^{\mathcal{I}}(x,y), C^{\mathcal{I}}(y)\}$ for some $y \in \Delta^{\mathcal{I}}$.

The following simple property holds.

[2] That is, $\geq_a, \leq_a$ and $=_a$ are crisp datatype predicates.

Proposition 1. *Let C be a fuzzy $\mathcal{ALC}(\mathbf{D})$ concept and let $\mathcal{I}$ be a strong fuzzy interpretation. If the fuzzy datatype predicates ls, rs and tri do not occur in C, then $C^{\mathcal{I}}(x) \in \{0,1\}$, for all $x \in \Delta^{\mathcal{I}}$.*

Therefore, if $C^{\mathcal{I}}(x) \notin \{0,1\}$, then one of ls, rs and tri have to occur in C. It is easily verified that Proposition 1 can be extended to the DL $\mathcal{SROIQ}$ (see, *e.g.*, [1,4,67], –the DL behind OWL 2) in place of $\mathcal{ALC}$.

The *satisfiability of fuzzy 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 $x \in \Delta^{\mathbf{D}}$, $D^{\mathcal{I}}(x) \geq C^{\mathcal{I}}(x) \cdot \alpha$. $\mathcal{I}$ is a *model* of a fuzzy KB $\mathcal{K}$ iff $\mathcal{I}$ satisfies each fuzzy axiom in $\mathcal{K}$. If $\mathcal{K}$ has a model we say that $\mathcal{K}$ is *satisfiable* (or *consistent*).

Remark 1. Concerning the semantics of GCIs, based on our experimental observations, using the product to integrate body degree with the fuzzy rule inclusion degree of learnt fuzzy GCIs yields more effective results than employing alternatives such as the minimum or Łukasiewicz t-norms. We anticipate that in our learning setting, *i.e.*, we learn fuzzy rules from *crisp* ontologies (defined later on), the above semantics of fuzzy GCIs do not affect the decidability of the reasoning problems used within the PN-OWL algorithm (see Proposition 2 later on and Appendix B), under the assumption that a finite truth space, such as $L_n = \{0, \frac{1}{n}, \frac{2}{n}, \ldots, \frac{(n-1)}{n}, 1\}$ (for integer $n > 1$), is assumed in place of $[0,1]$ (see also *e.g.*, [4,62–64]). Note that this is also the assumption under which *e.g.*, the *fuzzyDL* [59] reasoner operates.[3]

We say that a fuzzy KB $\mathcal{K}$ *entails* a fuzzy axiom τ, denoted $\mathcal{K} \models \tau$, if every model of $\mathcal{K}$ satisfies τ. The *best entailment degree* of an expression γ of the form $C \sqsubseteq D$, $a{:}A$ or $(a,b){:}r$, denoted $bed(\mathcal{K}, \gamma)$, is defined as

$$bed(\mathcal{K}, \gamma) = \sup\{\alpha \mid \mathcal{K} \models \langle \gamma, \alpha\rangle\} \ . \tag{3}$$

Now, consider a concept C, a fuzzy KB $\mathcal{K}$ and a finite set of individuals I. Various "scoring" functions we will use in this paper are based on counting the individuals of $\mathcal{K}$ that are *known* to be instances of an induced candidate concept (*viz.* to know whether they are 'positive' or 'negative', see later on). To do so, we adapt the notion of the sigma-count of a fuzzy set (*viz.* cardinality) we have seen previously (*cf.* Eq. (1)) to fuzzy $\mathcal{ALC}(\mathbf{D})$ in the usual way [16,22–24]. That is, the *sigma-count* (or simply the *cardinality*), of C w.r.t. $\mathcal{K}$ and I, denoted $|C|^{\mathrm{I}}_{\mathcal{K}}$, is defined as

$$|C|^{\mathrm{I}}_{\mathcal{K}} = \sum_{a \in \mathrm{I}} bed(\mathcal{K}, a{:}C) \ . \tag{4}$$

Lastly, similarly to fuzzy sets, we consider the concept subsumption relation as a binary fuzzy relation among two concepts. Specifically, the notion of inclusion degree among fuzzy sets (see Eq. (2)) is extended naturally to fuzzy $\mathcal{ALC}(\mathbf{D})$ as follows: the *inclusion degree* of C in D w.r.t. $\mathcal{K}$ and a set of individuals I, denoted $inc(C, D, \mathcal{K}, \mathrm{I})$, is defined as

$$inc(C, D, \mathcal{K}, \mathrm{I}) = \frac{|C \sqcap D|^{\mathrm{I}}_{\mathcal{K}}}{|C|^{\mathrm{I}_{\mathcal{K}}}_{\mathcal{K}}} \tag{5}$$

if $|C|^{\mathrm{I}}_{\mathcal{K}} > 0$. If $|C|^{\mathrm{I}}_{\mathcal{K}} = 0$ then $inc(C, D, \mathcal{K}, \mathrm{I})$ is left undefined in our setting.

As already anticipated, the objective of this work is to learn fuzzy rules from a *crisp* ontology/KB. To avoid any confusion, we define here explicitly the notion of *crisp* ontology/KB.

Crisp KBs/ontologies To start with, from a syntax point of view, let $\mathcal{ALC}(\mathbf{D})$ concepts be as fuzzy $\mathcal{ALC}(\mathbf{D})$ concepts, except that now the fuzzy datatype predicates ls, rs and tri are disallowed. Now, a *crisp* KB, also called *crisp ontology*, or simply *ontology*, is a finite set of *assertion axioms* and *GCI axioms*. An *assertion axiom* is of the form $a{:}C$, (called *concept assertion*, –individual a is an instance of concept C) or of the form $(a_1, a_2){:}r$ (called *role assertion*, –the pair (a_1, a_2) is an instance of object property r), where a_1, a_2 are individuals, and C is an $\mathcal{ALC}(\mathbf{D})$ concept. A *General Concept Inclusion* (GCI) axiom, is of the form $C_1 \sqsubseteq C_2$ (C_1 is a subsumed by concept C_2), where C_i is an $\mathcal{ALC}(\mathbf{D})$ concept.

From a semantics point of view, a strong fuzzy interpretation $\mathcal{I}$ *satisfies* an $\mathcal{ALC}(D)$ axiom γ of the form $C \sqsubseteq D$, $a{:}A$ or $(a,b){:}r$, if $\mathcal{I}$ satisfies $\langle \gamma, 1\rangle$. $\mathcal{I}$ is a *model* of a crisp KB $\mathcal{K}$ iff $\mathcal{I}$ satisfies each axiom in $\mathcal{K}$. $\mathcal{K}$ is *satisfiable* if it has a model.

So, obviously, by Proposition 1, over $\mathcal{ALC}(\mathbf{D})$, a strong fuzzy interpretation is exactly the same as a 'classical' DL interpretation [2].

3. PN-OWL

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

In what follows, we will assume that the input KB $\mathcal{K}$ of PN-OWL *is a satisfiable crisp KB.*

[3] Specifically, the truth space may be the finite set of floating-point numbers in $[0,1]$ a computer's floating-point arithmetic is able to deal with.

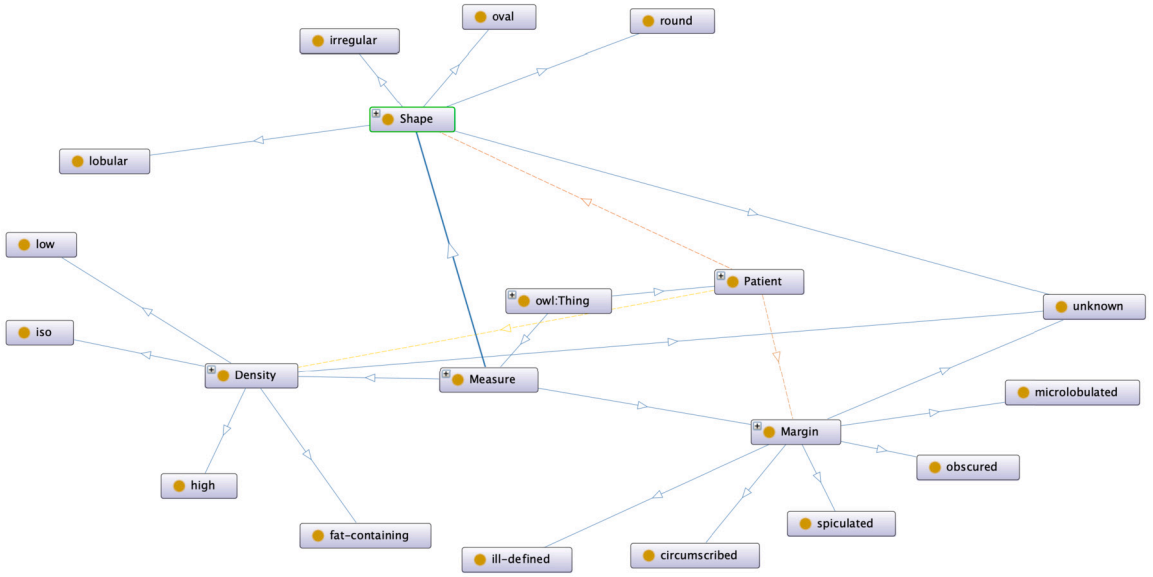

Fig. 2. Excerpt of the mammography ontology.

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	...
...	...	...	...	...	...	...

3.1. The learning problem

We illustrate the learning problem using an example (see also [16,22,23] for analogous examples).

Example 3.1. For the sake of illustrative purposes, let us consider a structurally simple OWL 2 ontology that describes the meaningful entities of mammography screening data. The target is the prediction of breast cancer [25]. An excerpt of the ontology $\mathcal{K}$ 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.

The TBox of the mammography KB contains GCIs such as

$$\texttt{Margin} \sqsubseteq \texttt{Measure}$$
$$\texttt{ill-defined} \sqsubseteq \texttt{Margin}$$
$$\texttt{spiculated} \sqsubseteq \texttt{Margin}$$

while, *e.g.*, patient `p102` can be encoded as the concept assertion

$$\texttt{p102} : \texttt{Patient} \sqcap \exists \texttt{hasShape.irregular} \sqcap \exists \texttt{hasMargin.ill-defined} \sqcap \exists \texttt{hasAge.} =_{58} \sqcap \exists \texttt{hasBiRads.} =_{4} . \tag{6}$$

Examples of positive and negative examples are, respectively

$$\texttt{p0, p10, p109, p111} \quad \triangleright \textit{ positive examples}$$
$$\texttt{p102, p108, p110} \quad \triangleright \textit{ negative examples} .$$

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

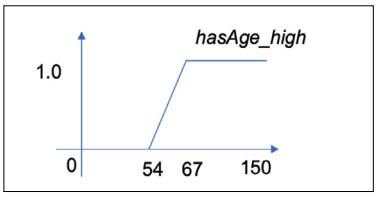

Fig. 3. Fuzzy set `hasAge_high`.

Assuming that our target class is called `Cancer`, then one may learn from the ontology, for instance by using Fuzzy FOIL-$\mathcal{DL}$ [16, 22,24], the following fuzzy concept/class inclusion rule:

$$\begin{aligned}\langle &\exists \texttt{hasMargin.ill-defined} \sqcap \\ &\exists \texttt{hasShape.irregular} \sqcap \\ &\exists \texttt{hasAge.hasAge\_high} \sqsubseteq \texttt{Cancer}, 0.853\rangle\end{aligned} \tag{7}$$

where `hasAge_high` is the fuzzy set $rs(54, 67)$ (see Fig. 3), which has automatically been built via the c-means clustering algorithm over the values of the `hasAge` attribute. The informal reading of rule (7) is as follows:

"if in a patient's mammography, there is a region whose margin is ill-defined, whose shape is irregular, and the patient's age is high, then the mammography is about breast cancer."

Now, let `BC` be the body of (7), *i.e.*,

$$\begin{aligned}&\exists \texttt{hasMargin.ill-defined} \sqcap \\ &\quad \exists \texttt{hasShape.irregular} \sqcap \exists \texttt{hasAge.hasAge\_high}\,.\end{aligned} \tag{8}$$

Then, the value 0.853 is the inclusion degree of `BC` in `Cancer` w.r.t. $\bar{\mathcal{K}} = \mathcal{K} \cup \{\texttt{BC} \sqsubseteq \mathit{Cancer}\}$, and the set of positive examples P and, thus, is determined as

$$0.853 = \frac{|\texttt{BC}|^{P}_{\mathcal{K}}}{|\texttt{BC}|^{\mathrm{I}_{\mathcal{K}}}_{\mathcal{K}}}\,. \tag{9}$$

Furthermore, it can be shown that

$$\mathit{bed}(\mathcal{K}, \texttt{p102:}\ \exists \texttt{hasAge.hasAge\_high}) = 0.308$$

as $\texttt{hasAge\_high}(58) = 0.308$ and, thus,

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

Moreover, if we add the learnt rule $\langle \texttt{BC} \sqsubseteq \texttt{Cancer}, 0.853\rangle$ to $\mathcal{K}$, *i.e.*, the rule (7), then we will infer that

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

That is, it is true to degree no less than 0.262 that `p102` (a negative example) has breast cancer. Similarly, we have

$$\mathit{bed}(\mathcal{K}, \texttt{p111:BC}) = 1.0$$

and, thus,

$$\mathit{bed}(\mathcal{K} \cup \{\langle \texttt{BC} \sqsubseteq \texttt{Cancer}, 0.853\rangle\}, \texttt{p111:Cancer}) = 0.853\,,$$

i.e., patient `p111` (a positive example) has breast cancer to degree no less than 0.853.

Formally, the learning problem we address is stated in general terms as follows (see also [16,42]).

Given:

1. a satisfiable crisp KB $\mathcal{K}$ and its individuals $\mathrm{I}_{\mathcal{K}}$;
2. a *target concept name* T with an associated unknown classification function $f_T : \mathbf{I} \to \{0, 1\}$, where for each $a \in \mathbf{I}$, the possible values (*labels*) correspond, respectively, to $\mathcal{K} \nvDash a{:}T$ (a is a *negative* example of T) and $\mathcal{K} \vDash a{:}T$ (a is a *positive* example of T);
3. a *hypothesis space* of classifiers $\mathcal{H} = \{h_T : \mathrm{I}_{\mathcal{K}} \to \{0, 1\}\}$;

4. a *training set* $\mathcal{E} = \mathcal{E}^+ \cup \mathcal{E}^- \subset \mathtt{I}_\mathcal{K}$ of individuals, where

$$\mathcal{E}^+ = \{a \mid a \in \mathtt{I}_\mathcal{K}, f_T(a) = 1\}$$
$$\mathcal{E}^- = \{a \mid a \in \mathtt{I}_\mathcal{K}, f_T(a) = 0\} ,$$

   on which f_T is known. We write $\mathcal{E}(a) = 1$ if a is a positive example, $\mathcal{E}(a) = 0$ if a is a negative example and assume that there is at least one positive example in $\mathcal{E}$;
5. a *test set* $\bar{\mathcal{E}} \subset \mathtt{I}_\mathcal{K}$ of individuals with $\mathcal{E} \cap \bar{\mathcal{E}} = \emptyset$. We assume that $\bar{\mathcal{E}}$ contains at least one positive example. The conditions and notation used for the test set mirror those of the training set, with $\mathcal{E}$ replaced by $\bar{\mathcal{E}}$.

Learn:

a classifier $\bar{h} \in \mathcal{H}$ that approximates best the classification function according to the principle of *Empirical Risk Minimisation* (ERM) [68] on $\mathcal{E}$: *i.e.*, choose a classifier $\bar{h}$ such as:

$$\bar{h} = \operatorname*{argmin}_{h \in \mathcal{H}} R(h, \mathcal{E})$$

where

$$R(h, \mathcal{E}) = \frac{1}{|\mathcal{E}|} \sum_{a \in \mathcal{E}} L(h(a), \mathcal{E}(a)) ,$$

with L being a *loss/error function* $L : \{0,1\}^2 \to \mathbb{R}$ such that $L(\hat{l}, l)$ measures how different the prediction $\hat{l}$ of a hypothesis is from the true outcome l and $R(h, \mathcal{E})$ is the *risk* associated with hypothesis h over $\mathcal{E}$, defined as the expectation of the loss function over $\mathcal{E}$. The loss function we adopt in our paper is the well-known 0-1 *loss function*, defined as

$$L(\hat{l}, l) = \begin{cases} 1 & \text{if } \hat{l} \neq l \\ 0 & \text{otherwise} , \end{cases}$$

and, thus, we try to minimise the number of misclassified training examples. The generalization ability of the learnt classifier $\bar{h}$ (or its effectiveness) is then assessed by means of some well-known effectiveness measures on the test set $\bar{\mathcal{E}}$.

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

$$\langle C_1 \sqsubseteq P, \alpha_1 \rangle , \dots , \langle C_h \sqsubseteq P, \alpha_h \rangle \tag{10}$$

$$\langle D_1 \sqsubseteq N, \beta_1 \rangle , \dots , \langle D_k \sqsubseteq N, \beta_k \rangle \tag{11}$$

together with a decision operator of the form ($a \in \mathtt{I}_\mathcal{K}$)

$$h_T(a) = @(p_a, n_a) , \tag{12}$$

where

1. P and N are new class names not occurring in $\mathcal{K}$;
2. α_i, β_j are the inclusion degrees of the relative rules;
3. h_P is the set of P-rules as per Eq. (10);
4. h_N is the set of N-rules as per Eq. (11);
5. $p_a = bed(\mathcal{K} \cup h_P, a{:}P)$;
6. $n_a = bed(\mathcal{K} \cup h_N, a{:}N)$;
7. $@$ is an aggregation operator; and
8. each C_i, D_j is a fuzzy $\mathcal{EL}(\mathbf{D})$ concept expression [26] defined as (b is a Boolean value)

$$\begin{array}{lcl} C & \longrightarrow & \top \mid A \mid \exists r.C \mid \exists s.\mathbf{d} \mid C_1 \sqcap C_2 \\ \mathbf{d} & \to & ls(a,b) \mid rs(a,b) \mid tri(a,b,c) \mid =_b . \end{array} \tag{13}$$

Informally, *(i)* each P-rule will tell us why an individual should be positive; *(ii)* on the other hand, each N-rule will tell us why an individual should be *non-positive*, *i.e.*, should be negative. Finally, *(iii)* we use the aggregation operator in Eq. (12) to establish whether an individual is an instance of T or not (*viz.* is positive or negative) by combining the degree of being positive or negative via the $@$ operator. A simple choice for $@$ is the following and is the one we will adopt:

$$@(p, n) = \begin{cases} 1 & \text{if } p > n \\ 0 & \text{otherwise} \end{cases} \tag{$\star$}$$

For $a \in \mathtt{I}_\mathcal{K}$, $h_T(a)$ is called the *classification prediction value* of a.

Remark 2. In experimental works, it is commonplace that you try out various options for all parameters involved. We explored alternatives to (⋆), but have not yet found a more effective solution.

Remark 3. Please note that we do not learn rules involving concepts of the form $\exists s.=_a$ (for integer/real value a) as the search space would be too large.

Example 3.2. Consider Example 3.1. As we have seen, a P-rule example following Eq. (10) is rule r^+ defined as

$$\langle \exists \texttt{hasMargin.ill-defined} \sqcap \exists \texttt{hasShape.irregular} \sqcap \exists \texttt{hasAge.hasAge\_high} \sqsubseteq \texttt{P}, 0.853 \rangle \tag{14}$$

while an example of N-rule following Eq. (11) is rule r^- defined as

$$\langle \exists \texttt{hasDensity.low} \sqcap \exists \texttt{hasAge.hasAge\_medium} \sqcap \exists \texttt{hasBiRads.hasBiRads\_medium} \sqsubseteq \texttt{N}, 1.0 \rangle \ , \tag{15}$$

where hasAge_medium is defined as $tri(40, 54, 67)$ and hasBiRads_medium is defined as $tri(3, 4, 5)$. Now, one may verify that for $h_P = \{r^+\}$ and $h_N = \{r^-\}$

$$bed(\mathcal{K} \cup h_P, \texttt{p102:P}) = 0.262$$
$$bed(\mathcal{K} \cup h_N, \texttt{p102:N}) = 0.692$$
$$bed(\mathcal{K} \cup h_P, \texttt{p111:P}) = 0.853$$
$$bed(\mathcal{K} \cup h_N, \texttt{p111:N}) = 0.0 \ .$$

Therefore, by applying (⋆), we would correctly classify patient p102 as not having breast cancer, while correctly identifying that p111 has it.

Remark 4. The set of rules determined by this syntax can be potentially infinite because of conjunctions and the nesting of existential restrictions in the concept expressions. However, as we will discuss later, we will enforce additional restrictions, like limiting the maximum number of conjuncts and setting a cap on the depth of existential nestings.

We conclude by saying that a hypothesis h_T *covers* an individual $a \in \mathtt{I}_{\mathcal{K}}$ iff $h_T(a) = 1$, and indicates with $Cov(h_T)$ the set of covered individuals. Analogously, we say that a set h_P (resp. h_N) of P-rules (resp. N-rules) θ-covers, for $\theta \in (0, 1]$ an individual $a \in \mathtt{I}_{\mathcal{K}}$ iff $p_a \geq \theta$ (resp. $n_a \geq \theta$), and indicate with $Cov_\theta(h_P)$ (resp. $Cov_\theta(h_N)$) the set of covered individuals by h_P (resp. h_N). Moreover, for a P-rule or N-rule $C \sqsubseteq X$ ($X \in \{P, N\}$) that covers at least one example (*i.e.*, $|C|^{\mathtt{I}_{\mathcal{K}}}_{\mathcal{K}} > 0$), the *confidence degree* (also called the *precision*) of $C \sqsubseteq X$ w.r.t. $\mathcal{K}$ and a non-empty set of positive examples P, denoted $cf(C \sqsubseteq X, \mathcal{K}, P)$, is defined as the inclusion degree of C in X w.r.t. $\bar{\mathcal{K}} = \mathcal{K} \cup \{C \sqsubseteq X\}$ and P, *i.e.*,[4]

$$cf(C \sqsubseteq X, \mathcal{K}, P) = inc(C, X, \bar{\mathcal{K}}, P) = \frac{|C \sqcap X|^P_{\bar{\mathcal{K}}}}{|C|^{\mathtt{I}_{\bar{\mathcal{K}}}}_{\bar{\mathcal{K}}}} = \frac{|C|^P_{\mathcal{K}}}{|C|^{\mathtt{I}_{\mathcal{K}}}_{\mathcal{K}}} \ . \tag{16}$$

Clearly, $cf(C \sqsubseteq X, \mathcal{K}, P) \in [0, 1]$ and the closer the confidence is to 1, the 'more precise' is rule $C \sqsubseteq X$; that is, the fewer non-positive individuals are covered by it. In addition, the *support* of $C \sqsubseteq X$ w.r.t. $\mathcal{K}$ and a non-empty set of individuals $\mathtt{I}$, denoted $supp(C \sqsubseteq X, \mathcal{K}, \mathtt{I})$, is defined as

$$supp(C \sqsubseteq X, \mathcal{K}, \mathtt{I}) = \frac{|C \sqcap X|^{\mathtt{I}}_{\bar{\mathcal{K}}}}{|\mathtt{I}|} = \frac{|C|^{\mathtt{I}}_{\mathcal{K}}}{|\mathtt{I}|} \ , \tag{17}$$

where again $\bar{K}B = \mathcal{K} \cup \{C \sqsubseteq X\}$.

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 set of all individuals $\mathtt{I}_{\mathcal{K}}$, the set $\mathcal{E}^+$ (resp. $\mathcal{E}^-$) of positive (resp. negative) examples (Fig. 4 (a)).

In the first stage, the P-stage, we try to maximize the covering of positive examples while minimizing the covering of negative examples. 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 examples, but may also contain negative examples. We call the individuals in $TP = Cov_{\theta_P}(h_P) \cap \mathcal{E}^+$ *true positives*, while call those in $FP = Cov_{\theta_P}(h_P) \setminus \mathcal{E}^+$ *false positives*, *i.e.*, a false positive is an individual that would be predicted by h_P as positive, while

[4] If $|C|^{\mathtt{I}_{\mathcal{K}}}_{\mathcal{K}} = 0$, the confidence is undefined. Of course, we are not interested in rules that do not cover examples.

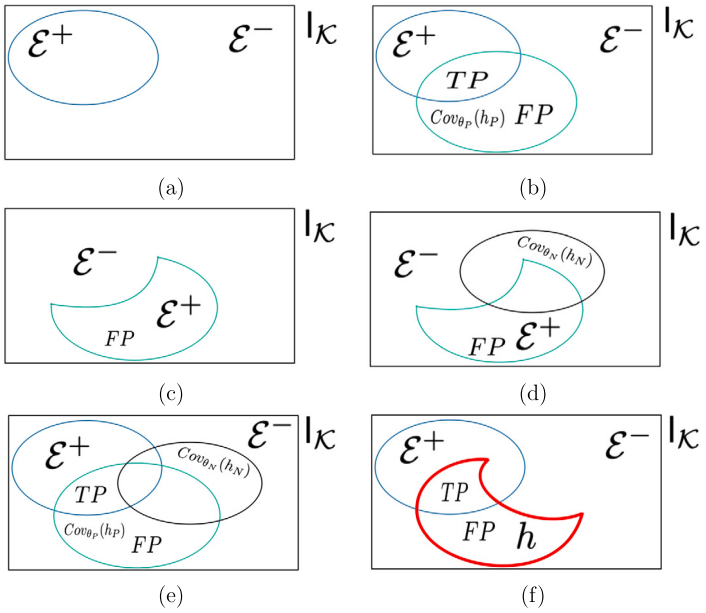

Fig. 4. How PN-OWL works: (a) Original training set; (b) Coverage $Cov_{\theta_P}(h_P)$ w.r.t. learnt hypothesis h_P after the P-stage; (c) Starting dataset for N-stage: the new target class is the set of false positives FP of the P-stage; (d) Coverage $Cov_{\theta_N}(h_N)$ w.r.t. learnt hypothesis h_N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

actually it is not. This phase ends with a set of rules of the form as in Eq. (10). For instance, by referring to Example 3.2, one may learn at the P-stage the rule r^+ covering individuals such as p102 and p111. The latter is a true positive, while the former is a false positive.

In the subsequent stage, the N-stage, we would like to improve the effectiveness of the classifier by trying to remove as many false positives in FP as possible, while avoiding to remove, if possible, any of the true positives in TP. To accomplish this, we define a new learning problem: the new target class is N, where the positive examples are now those in FP and, consequently, all others are negative examples. Of course, the N-stage applies only if $FP \neq \emptyset$. The setup of the N-stage is depicted in Fig. 4 (c) and 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 Eq. (11). So, in general, at the end of the two stages, the situation may be as depicted in Fig. 4 (e), showing all sets. For instance, continuing by referring to Example 3.2, one may learn at the N-stage the rule r^- covering individual p102 only.

Finally, we aggregate the evidences provided by the P-rules and N-rules using ($\star$) to make our final classification prediction according to Eq. (12). Fig. 4 (f) illustrates the covering of the final hypothesis h.

Referring to Example 3.2, 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 previous section. 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

$$h_P = \{\langle C_1 \sqsubseteq P, \alpha_1\rangle, \ldots, \langle C_h \sqsubseteq P, \alpha_h\rangle\} \tag{18}$$

$$h_N = \{\langle D_1 \sqsubseteq N, \beta_1\rangle, \ldots, \langle D_k \sqsubseteq N, \beta_k\rangle\} \tag{19}$$

called *stage hypothesis*. In the following, we indicate with p_i the fuzzy GCI $\langle C_i \sqsubseteq P, \alpha_i\rangle$, while denote with n_j the fuzzy GCI $\langle D_j \sqsubseteq N, \beta_j\rangle$.

As the final classification prediction is determined via Eq. (12), it suffices to describe the algorithm that returns the set $h_P \cup h_N$ of P- and N-rules, which is what the PN-OWL algorithm does.

As stage learner, we will use Fuzzy FOIL-$\mathcal{DL}$ [16,22,24]. For the sake of completeness, we will give a brief summary of Fuzzy FOIL-$\mathcal{DL}$ in Appendix A.

The PN-OWL algorithm is shown in Algorithm 1. Note that the P-stage corresponds to steps 1-4, while the N-stage corresponds to steps 13-16. At step 16 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 $h_P \cup h_N$ as per Eqs. (18)-(19).

Finally, the *classification prediction value of* PN-OWL for individual $a \in \mathrm{I}_\mathcal{K}$ is $h_T(a)$, using Eq. (12) and ($\star$) as aggregation operator.

Table 2
Facts about the ontologies of the evaluation.

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

Algorithm 1 PN-OWL.

Input: A satisfiable crisp KB $\mathcal{K}$, a set $\mathcal{E}$ of training examples, set of parameters (see Table 3)
Output: Stage hypotheses $h_P \cup h_N$ as by Eqs. (18)-(19).

1: // P-stage
2: $Pos \leftarrow \mathcal{E}^+$;
3: $Neg \leftarrow \mathcal{E}^-$;
4: $h_P \leftarrow$ FuzzyStageLearner($\mathcal{K}$, P, Pos, Neg, θ_P, η_P); //P-Stage hypothesis h_P, *i.e.*, set of axioms of the form $\langle C_i \sqsubseteq P, \alpha_i \rangle$, where P is a new concept name
5: **if** $h_P = \emptyset$ **then return** $\emptyset$; //Nothing learnt, exit
6: $Cov \leftarrow Cov_{\theta_P}(h_P)$; //P-stage Coverage
7: $TP \leftarrow Cov_{\theta_P}(h_P) \cap \mathcal{E}^+$; //True positives
8: $FP \leftarrow Cov_{\theta_P}(h_P) \cap \mathcal{E}^-$; //False positives
9: // Start building classifier h
10: $h \leftarrow h_P$; //As per Eq. (18)
11: **if** $FP = \emptyset$ **then** //No N-stage, exit
12: **return** h;
13: // N-stage
14: $Pos \leftarrow FP$;
15: $Neg \leftarrow \mathtt{I}_{\mathcal{K}} \setminus Pos$;
16: $h_N \leftarrow$ FuzzyStageLearner($\mathcal{K}$, N, Pos, Neg, θ_N, η_N); //N-Stage hypothesis h_N, *i.e.*, set of axioms of the form $\langle D_j \sqsubseteq N, \beta_j \rangle$, where N is a new concept name
17: // Build final classifier ensemble h
18: **if** $h_N = \emptyset$ **then** //No learning in N-stage, exit
19: **return** h;
20: $h \leftarrow h \cup h_N$; //As per Eq. (19)
21: **return** h;

Concerning the computational aspects of the algorithm, the following proposition can be shown (for more details, see Appendix B)

Proposition 2. *Given an OWL 2 ontology $\mathcal{K}$, a set $\mathcal{E}$ of training examples, the parameters in Table 3, an individual $a \in \mathtt{I}_{\mathcal{K}}$ and assume that a finite truth space semantics, such as $L_n = \{0, \frac{1}{n}, \frac{2}{n}, \ldots, \frac{(n-1)}{n}, 1\}$ (for integer $n > 1$), is adopted. Then, the classification prediction value of an individual a can be computed in finite time.*

4. Evaluation

We have implemented the algorithm within the *FuzzyDL-Learner* system [27].

Datasets Several OWL 2 ontologies from different domains have been selected as shown in Table 2, which reports for each ontology the (crisp) DL to which it refers, the number of concept/class names, object properties, datatype properties, and individuals in the ontology. For each ontology we also report the number of positive examples (all other individuals are considered negative).

The ontologies Iris, Wine, Wine Quality and Yeast are built from the well-known *UC Irvine Machine Learning Repository* (UCIMLR) [28] and have been transformed into OWL 2 ontologies according to the procedure described in [16]. While the malware ontology has been described in [29], all other ontologies belong to the well-known SML-Bench dataset [25].

Remark 5. Note that all but Lymphography have datatype properties. While it is atypical to evaluate ontology-based learning algorithms on numerical datatype properties, we believe it is interesting to do so because an important part of our algorithm is the use of fuzzy concrete datatype properties to improve human understandability, *i.e.*, the explainability of the classification decision process.

Measures We considered the following known measures of effectiveness, which we summarise here for the sake of completeness and replicability of the experiments. Specifically, consider a learnt classifier h_T and assume that all P-rules and N-rules have been

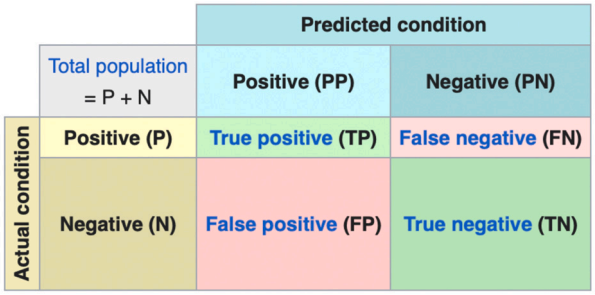

Fig. 5. Confusion matrix.

added to the input ontology $\mathcal{K}$. As an individual a is either classified as positive ($h_T(a) = 1$) or negative ($h_T(a) = 0$), we will consider the following measures, based on the well-known *confusion matrix* in Fig. 5.

The *True Positive* (TP) is defined as the number of predicted positive examples that are actually positive, *i.e.*,

$$TP = |Cov(h_T) \cap \mathcal{E}^+| \ . \tag{20}$$

The *False Positive* (FP) is defined as the number of predicted positive examples that are actually negative, *i.e.*,

$$FP = |Cov(h_T) \cap \mathcal{E}^-| \tag{21}$$

The *True Negative* (TN) is defined as the number of predicted negative examples that are actually negative, *i.e.*,

$$TN = |(\mathrm{I}_{\mathcal{K}} \setminus Cov(h_T)) \cap \mathcal{E}^-| \ . \tag{22}$$

The *False Negative* (FN) is defined as the number of predicted negative examples that are actually positive, *i.e.*,

$$FN = |(\mathrm{I}_{\mathcal{K}} \setminus Cov(h_T)) \cap \mathcal{E}^+| \ . \tag{23}$$

Of course,

$$|\mathrm{I}_{\mathcal{K}}| = TP + TN + FP + FN$$

$$|Cov(h_T)| = TP + FP$$

hold and the number of correctly classified examples is $TP + TN$, while the number of misclassified examples is $FP + FN$.

Precision, denoted $Prec$, is defined as the fraction of true positives w.r.t. the covered examples, *i.e.*,

$$Prec = \frac{TP}{|Cov(h_T)|} \ . \tag{24}$$

Recall, denoted Rec, is defined as fraction of true positives w.r.t. actual positives, *i.e.*,

$$Rec = \frac{TP}{|\mathcal{E}^+|} \ . \tag{25}$$

The $F1$-*score*, denoted $F1$, is defined as

$$F1 = 2 \cdot \frac{Prec \cdot Rec}{Prec + Rec} \ . \tag{26}$$

Accuracy, denoted Acc, is defined as

$$Acc = \frac{TP + TN}{|\mathrm{I}_{\mathcal{K}}|} \ , \tag{27}$$

while the *Misclassification Rate*, denoted MR, is defined as

$$MR = \frac{FP + FN}{|\mathrm{I}_{\mathcal{K}}|} = 1 - Acc \ . \tag{28}$$

The ranges of $Prec$, Rec, $F1$, Acc, and MR are in $[0, 1]$. Higher values of $Prec$, Rec, $F1$, and Acc indicate better results. Conversely, a lower MR value signifies a better result.

Parameters Table 3 presents some key parameters used in our algorithm. In the experimentation, the maximal role depth has been set to an ontology-dependent threshold (usually, $d_P = d_N$) determined (manually) a priori through an inspection of the ontology. For the other parameters, we tested several configurations performing a random grid search over a large parameter-space $\langle \theta_P, \theta_N, \eta_P, \eta_N, c_P, c_N \rangle$ by varying confidence thresholds, non-positive coverage percentage, and maximal number of conjuncts.

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

parameter	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 N-stage
d_P	maximal role depth for positive rules of P-stage
d_N	maximal role depth for negative rules of N-stage

Considering the limited size of the datasets and to prevent results that are biased by a particular training-test split, we performed a stratified k-fold cross-validation for each parameter configuration[5] and calculated the average of Prec, Rec, F1-score and MR over the k test sets (for multi-target datasets we report the macro-average over the targets) to provide a *global* estimate of the model performance. A brief description of the method is given below:

Partitioning the dataset. The examples are partitioned in k disjoint subsets $flds = \{fld_1, \ldots, fld_k\}$, called folds. In a stratified design, each fold fld_i contains roughly the same ratio of positive to negative examples as the full set of examples.
Training and testing. The model is trained and tested k times using different training and test sets: at iteration i, the fold fld_i is used as test set and $flds \backslash fld_i$ as training set.
Effectiveness evaluation. After all iterations are complete, the effectiveness metrics are averaged over the k different test sets.

In our implementation, for each fold, we remove all assertions that involve test examples from the ontology before the training stage. In this way, the model is trained without any information about test individuals.

All experimental data is available in the downloadable resource.[6] The resource includes the executable code, the ontologies and, for each ontology, the parameter configuration that yielded the highest test macro average F1-score. Additionally, it provides the learnt rules, the fuzzy datatype predicates and effectiveness measures for each fold.

It is worth noting that a typical configuration of the best parameters found by a grid search is as follows, but it may vary depending on the ontology in question:

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 on it. During the P-stage, one 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 positives. In this way, one ultimately increases recall and precision and, thus, the F1-score. So, in the P-stage, a low confidence threshold θ_P and a high non-positive coverage percentage η_P has been determined by a grid search through the parameter space. In the N-stage however, one may want to be more precise in removing the false positives while avoiding to remove the true positives from the P-Stage. Therefore, there is an increase of the confidence threshold θ_N, a decrease of the non-positive coverage percentage threshold η_N, and an increase of the number of maximal conjuncts c_N.

We recall that we used ($\star$) as aggregator @. 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.

For the settings of other parameters not reported in Table 3, we tested different numbers of fuzzy sets for each data property (i.e., 3, 5, or 7). For c-means, we kept the parameter m (degree of fuzziness of the solution) to its default value ($m = 2$), the threshold ϵ to 0.05, and set the maximum number of iterations to 100. These choices were made according to the experience gained in previous experiments [16,22] and taking into account that clustering is done for each single data property.

As baseline, we consider the learner Fuzzy FOIL-$\mathcal{DL}$ [22,24], with best parameter setup as specified in [16]. 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 18-19 of PN-OWL algorithm). This allows us to appreciate the added value (if any) in terms of the effectiveness of the N-stage phase.

Discussion The results of the experiments carried out are presented in the Table 4, where the column %ΔF1 (resp. %ΔMR) reports the percentage of increase (resp. decrease) for PN-OWL of the $F1$ (resp. MR) measure over our baseline Fuzzy FOIL-$\mathcal{DL}$. Overall, despite its conceptual simplicity, PN-OWL outperforms Fuzzy FOIL-$\mathcal{DL}$.[7] The improvement is particularly high in some cases, such as for `NTN`, `Mammographic` and `Wine Quality`.

[5] For a brief overview, see *e.g.*, https://en.wikipedia.org/wiki/Cross-validation_(statistics).
[6] http://www.umbertostraccia.it/cs/software/FuzzyDL-Learner/download/pnowl.experiments.zip.
[7] For `YEAST`, the MR values are essentially equal.

Table 4
Results table. Metrics are averaged over the test sets of the five folds. For the multi-target datasets `Iris` and `Wine`, the values are macro-averaged over the targets. The ↑/↓ arrows mean that a higher/lower value of the metric is better.

Dataset	Algorithm	Prec ↑	Rec ↑	F1 ↑	MR ↓	%ΔF1	%ΔMR
NTN	Fuzzy FOIL-DL	0.661	0.513	0.548	0.053	+**80.47%**	**-98.11%**
	PN-OWL	**1.000**	**0.980**	**0.989**	**0.001**		
Lymphography	Fuzzy FOIL-DL	0.810	0.803	0.805	0.210	+**3.48%**	**-12.38%**
	PN-OWL	**0.836**	**0.841**	**0.833**	**0.184**		
Mammographic	Fuzzy FOIL-DL	0.737	0.692	0.710	0.256	+**10.56%**	**-19.14%**
	PN-OWL	**0.746**	**0.831**	**0.785**	**0.207**		
Malware	Fuzzy FOIL-DL	0.623	**0.830**	0.704	0.060	+**4.97%**	**-18.12%**
	PN-OWL	**0.701**	0.818	**0.739**	**0.049**		
Iris	Fuzzy FOIL-DL	0.886	0.910	0.890	0.077	+**4.16%**	**-41.18%**
	PN-OWL	**0.949**	0.910	**0.927**	**0.045**		
Wine	Fuzzy FOIL-DL	0.884	**0.971**	0.895	0.091	+**2.12%**	**-37.50%**
	PN-OWL	**0.933**	0.904	**0.914**	**0.057**		
Wine Quality	Fuzzy FOIL-DL	0.227	**0.917**	0.363	0.109	+**27.93%**	**-53.21%**
	PN-OWL	**0.365**	0.659	**0.464**	**0.051**		
YEAST	Fuzzy FOIL-DL	0.427	0.746	0.540	**0.382**	+**4.37%**	+**0.26%**
	PN-OWL	**0.432**	**0.815**	**0.564**	0.383		

Essentially, for PN-OWL the grid search over the parameter space was able to find a better compromise between precision and recall than for Fuzzy FOIL-$\mathcal{DL}$. In particular, we were able to increase precision through the N-stage, after increasing recall in the P-stage. This confirms our conjecture that the N-stage is indeed effective in removing false positives. This is further confirmed by the MR measure.

Remark 6. For `Lymphography`, we were unable to replicate the results of Fuzzy FOIL-$\mathcal{DL}$ in [16], for which we get now an F1 measure of 0.805 in place of 0.855. The difference lies in a few misclassified examples.[8] Furthermore, as the `Wine Quality` dataset changed slightly with respect to the one used in [16], we re-run the experiments for Fuzzy FOIL-$\mathcal{DL}$.

The overall lesson learnt with PN-OWL is that, provided one may find the appropriate balance between precision and recall, the N-stage may indeed provide a non-negligible contribution to improve the effectiveness of the classification process. Unfortunately, searching the parameter space of PN-OWL for an optimum is 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 Fuzzy 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 [8,30–35], 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 [22–24]) is a natural extension in which one relies on fuzzy DLs [3,4] and fuzzy ILP (see *e.g.*, [36]) instead: while, in the former works the authors propose *fuzzy* FOIL-like algorithms that are inspired by fuzzy ILP variants such as [37–39], in other works such as e.g. [40,41] 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 instances of a target class as possible 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[9];

[8] We conjecture this may be due to a slight difference in the 5-fold partitioning, even if, in theory, this should not have been the case as we have used the same random seeds. In any case, the ontologies used for each of the five runs of the cross-validation are now provided.

[9] Tabular data can easily be mapped into OWL 2 ontologies as illustrated in [16].

- 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,[10] 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. We will leave it for future work to look into it.

Discrete boosting has been considered in [42], while [16] is a real-valued fuzzy variant of discrete AdaBoost and differentiates from the previous one by using a descent-like gradient algorithm to search for the best alternative and is inspired by [43]. Notably, this also deviates from 'fuzzy' rule learning AdaBoost variants, such as [44–48].

The works [49,50] consider *Kernel Methods* for inducing concept descriptions, [51–53] are inspired by *Decision Trees/Random Forests*, while [54,55] consider essentially a *Naive Bayes* approach. Last but not least, [56] is inspired by *Genetic Programming* to induce concept expressions, while [57] is based on the *Reinforcement Learning* framework. Finally, [58] 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 5, typically ontology-based learning algorithms as those listed above do not rely on numerical datatype properties (except [16,22,23]), 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 3.2). To this end, let us recall that the construction of the fuzzy membership functions (see also *e.g.*, [17,18]), is a non-negligible ingredient of fuzzy ontology-based learning algorithms with numerical datatypes.

6. Conclusions & future work

We have presented a two-stage algorithm, called PN-OWL that is inspired by 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 positive examples as possible without compromising too much the number of non-positive examples they cover (the false positives); *(ii)* at the N-stage, it generates a set of fuzzy inclusion axioms, the N-rules, that cover as many false positives of the P-stage as possible; *(iii)* 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 a target class or not.

We have also conducted an extensive evaluation, comparing the new algorithm with Fuzzy FOIL-$\mathcal{DL}$. The results show that, despite the conceptual simplicity of PN-OWL, it outperforms Fuzzy FOIL-$\mathcal{DL}$ in terms of effectiveness. However, identifying an optimal parameter configuration is much more time-consuming.

For future work, in addition to exploring other learning methods and the areas highlighted throughout the paper, we envision several aspects worth investigating in more detail: *(i)* we aim to investigate the computational aspects. For some ontologies, a learning run can currently take up to a week given our available resources. In this regard, we would like to investigate both parallelization methods and 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 stage learner both in the P-stage and the N-stage (*cf.* lines 4 and 16 of the PN-OWL algorithm). In this regard, we would like to investigate how to plug in alternative learners, such as, e.g., FUZZY OWL-BOOST [16] and to verify their effectiveness; and *(iii)* we would also like to assess the impact of alternative scoring functions to information gain (cf. Eq. (A.1)) within our setting, including various choices of t-norms and aggregation functions.

Lastly, we would like to extend hypothesis language $\mathcal{EL}(\mathbf{D})$ with so-called *threshold concepts* [59] 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. Informally, the intended meaning of $C[\geq d]$ (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. As an example from our ongoing implementation, referring to the `Wine Quality` ontology, we generated an expression of the form

$$(\exists \texttt{alcohol.alcohol\_H})[\leq 0.786] \sqcap (\exists \texttt{sulfates.sulfates\_H})[\geq 0.289] \sqcap (\exists \texttt{ph.ph\_M})[\leq 0.106] \sqsubseteq \texttt{GoodRedWine}$$

By considering that `alcohol_H` is defined as $rs(10.729, 11.979)$, `sulfates_H` is defined as $rs(0.555, 0.712)$, and `ph_M` is defined as $tri(3.090, 3.228, 3.382)$, one may verify that the above mentioned rule may be rephrased in the following way: *"if, for an individual (wine) a, the alcohol level is less or equal to* 11.712*, the sulfates level is greater or equal to* 0.6 *and the pH level is either less or equal to* 3.105 *or greater or equal to* 3.366*, then classify a as an instance of* `GoodRedWine`*".*[11]

CRediT authorship contribution statement

Franco Alberto Cardillo: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Franca Debole:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Umberto Straccia:** Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization.

[10] If A is categorical then $cond$ is either of the forms $A = v$ or $A \neq v$.

[11] Note that w.r.t. `alcohol_H`, $rs(10.729, 11.979)(x) \leq 0.786$, for $x \leq 11.712$. The other cases are similar.

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

Input: A satisfiable crisp KB $\mathcal{K}$, target concept name T not occurring in $\mathcal{K}$, a set P (resp. N) of positive (resp. negative) examples, confidence threshold $\theta \in (0,1]$, negative coverage percentage $\eta \in [0,1]$
Output: A hypothesis, *i.e.*, a set $h_T = \{\langle C_i \sqsubseteq T, \delta_i \rangle | 1 \leq i \leq k\}$ of fuzzy $\mathcal{EL}(\mathbf{D})$ GCIs
1: $h_T \leftarrow \emptyset, Pos \leftarrow P, \phi \leftarrow \top \sqsubseteq T$;
2: **while** ($Pos \neq \emptyset$) **and** ($\phi \neq$ **null**) **do** //Loop until no improvement
3: $\phi \leftarrow$ LEARN-ONE-AXIOM($\mathcal{K}, T, Pos, P, N, \theta, \eta$); //Learn one fuzzy $\mathcal{EL}(\mathbf{D})$ GCI of the form $C \sqsubseteq T$
4: **if** $\phi \in h_T$ **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, as per Eq. (16)
8: $h_T \leftarrow h_T \cup \{\langle \phi, \delta \rangle\}$; //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_T;

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data and code are accessible from link within the paper

Acknowledgements

This research was partially supported by TAILOR, a project funded by EU Horizon 2020 research and innovation programme under (GA No 952215). This paper is also supported by the FAIR (Future Artificial Intelligence Research, GA No PE00000013) project funded by the NextGenerationEU program within the PNRR-PE-AI scheme (M4C2, investment 1.3, line on Artificial Intelligence). Finally, this work has also been partially supported by the H2020 STARWARS Project (GA No. 101086252), a 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 computing services and resources they made available.

Appendix A. The stage learner Fuzzy FOIL-$\mathcal{DL}$

The procedure invocations FUZZYSTAGELEARNER at steps 4 and 17 in the Algorithm 1 are calls to Fuzzy FOIL-$\mathcal{DL}$ [16,22,24], 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 (steps 9 and 10). To induce an axiom (step 3), LEARN-ONE-AXIOM is invoked (see Algorithm 3),

Refinement operator The refinement operator (see Table A.5) is the same as in [16,23,24,60]. It takes as input a concept C and generates a more specific concept description candidates D. In Table A.5, $\mathbf{A}_\mathcal{K}$ 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 $\mathbf{D}$ a fuzzy concrete domain. Moreover, for concepts C and D, let $C \sqsubseteq_\mathcal{K} D$ be a macro for $\mathcal{K} \models C \sqsubseteq D$, then $\sqsubset_\mathcal{K}$ is the strict sub-relation of $\sqsubseteq_\mathcal{K}$, *i.e.*, if $C \sqsubseteq_\mathcal{K} D$ then $D \sqsubseteq_\mathcal{K} C$ does not hold.

The GCI scoring function The scoring function used in Fuzzy FOIL-$\mathcal{DL}$ is the *Information Gain* function [16,23,24,60]. 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'.

Table A.5
Downward Refinement Operator.

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

Algorithm 3 LEARN-ONE-AXIOM.

Input: A satisfiable crisp KB $\mathcal{K}$, target concept name T not occurring in $\mathcal{K}$, set Pos of positive examples still to be covered, training sets P, N of positive and negative examples, respectively, confidence threshold $\theta \in (0,1]$, negative coverage percentage $\eta \in [0,1]$
Output: A fuzzy $\mathcal{EL}(\mathbf{D})$ GCI of the form $C \sqsubseteq T$

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

The LEARN-ONE-AXIOM *algorithm* The LEARN-ONE-AXIOM algorithm is illustrated in Algorithm 3: lines 1 - 2 are simple initialisation steps. Steps 3-16 are the main loop from which we may exit if the stopping criterion is satisfied, in step 6 we compute all new refinements, which are then scored in steps 7-12 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 17). 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.

Appendix B. Computational aspects of PN-OWL

In this section we outline the computational aspects of PN-OWL and its sub-procedure Fuzzy FOIL-$\mathcal{DL}$, employed in *e.g.*, [16,23,24,60]. Specifically, we show how to compute the classifier prediction value $h_T(a)$ of an individual a in finite time.

Let us recall that, by assumption, the input KB $\mathcal{K}$ of PN-OWL is crisp, which means also that *e.g.*, w.l.o.g. we may assume that all GCIs occurring in $\mathcal{K}$ are of the form $\top \sqsubseteq E$ as $C \sqsubseteq D$ is equivalent to $\top \sqsubseteq \neg C \sqcup D$. Furthermore, the hypothesis space is finite as we impose a maximum number of conjuncts and depth of existential nestings to occur in the to be learnt fuzzy GCIs. Also, for convenience, let us consider the language fuzzy $\mathcal{EL}_w(\mathbf{D})$ allowing so-called *weighted concepts* of the form $\alpha \cdot C$, where $\alpha \in [0,1]$ and C is a fuzzy $\mathcal{EL}(\mathbf{D})$ concept. $\alpha \cdot C$ is interpreted, by a fuzzy interpretation $\mathcal{I}$, as $\alpha \cdot C^{\mathcal{I}}$ (the degree of truth being an instance of C is multiplied by the constant α). Note that a weighted concept is a special case of the *weighted sum* concept $\sum_i \alpha_i \cdot C_i$, which has been exploited in [16] for fuzzy GCI learning (but see also *e.g.*, [4,19]).

We recap that the learnt fuzzy GCIs are of the form $\langle C \sqsubseteq X, \alpha \rangle$, for $X \in \{P, N\}$, where X neither occurs in $\mathcal{K}$ nor in C, and C is a fuzzy $\mathcal{EL}(\mathbf{D})$ concept expression (Eq. (13)) and $\mathcal{K}$ is a crisp KB. Now, the following can be proven:

Proposition 3. *Given a crisp KB $\mathcal{K}$, a fuzzy GCI of the form $\langle C \sqsubseteq X, \alpha \rangle$, where X occurs neither in $\mathcal{K}$ nor in C, the following three statements are equivalent for all $\beta \in [0,1]$:*

1. $\mathcal{K} \cup \{\langle C \sqsubseteq X, \alpha \rangle\} \models \langle a{:}X, \beta \rangle$

2. $\mathcal{K} \cup \{\alpha \cdot C \sqsubseteq X\} \models \langle a{:}X, \beta\rangle$
3. $\mathcal{K} \models \langle a{:}\alpha \cdot C, \beta\rangle$.

Proof. The equivalence among 1. and 2. is obvious. So, let us show the equivalence between 2. and 3. Clearly if $\mathcal{K} \models \langle a{:}\alpha \cdot C, \beta\rangle$ then $\mathcal{K} \cup \{\alpha \cdot C \sqsubseteq X\} \models \langle a{:}X, \beta\rangle$. Vice-versa, assume $\mathcal{K} \cup \{\alpha \cdot C \sqsubseteq X\} \models \langle a{:}X, \beta\rangle$ and suppose to the contrary that $\mathcal{K} \not\models \langle a{:}\alpha \cdot C, \beta\rangle$. Therefore, there is a model $\mathcal{I}$ of $\mathcal{K}$ such that $\alpha \cdot C^{\mathcal{I}}(a^{\mathcal{I}}) < \beta$. Now, let $\mathcal{J}$ be an interpretation that is as $\mathcal{I}$, except that $X^{\mathcal{J}}(x) = \alpha \cdot C^{\mathcal{I}}(x)$, for all domain elements $x \in \Delta^{\mathcal{J}} = \Delta^{\mathcal{I}}$. Therefore, $\mathcal{J}$ satisfies $\alpha \cdot C \sqsubseteq X$. But, X does not occur in $\mathcal{K}$, so $\mathcal{J}$ is a model of $\mathcal{K}$: in fact, $\mathcal{J}$ is identical to $\mathcal{I}$ for all entities occurring in $\mathcal{K}$ and $\mathcal{I}$ is a model of $\mathcal{I}$. Consequently, $\mathcal{J}$ is model of $\mathcal{K}$ such that $X^{\mathcal{J}}(a^{\mathcal{J}}) = \alpha \cdot C^{\mathcal{I}}(a^{\mathcal{I}}) < \beta$ and, thus, $\mathcal{K} \cup \{\alpha \cdot C \sqsubseteq X\} \not\models \langle a{:}X, \beta\rangle$, contrary to our assumption, which concludes. □

A direct consequence of Proposition 3 is

Proposition 4. *Given a crisp KB $\mathcal{K}$, GCIs of the form $\langle C \sqsubseteq X, \alpha\rangle$, where X neither occurs in $\mathcal{K}$ nor in C, then the following holds:*

$$bed(\mathcal{K} \cup \{\langle C \sqsubseteq X, \alpha\rangle\}, a{:}X) \quad = \quad bed(\mathcal{K}, a{:}\alpha \cdot C)\ .$$

Now, it is easily verified that Propositions 3 and 4 can be generalised to the case in which we consider a set h_X of rules $\langle C_i \sqsubseteq X, \alpha_i\rangle$ $(1 \leq i \leq n)$ instead. That is,

Proposition 5. *Given a crisp KB $\mathcal{K}$, a set h_X of GCIs of the form $\langle C_i \sqsubseteq X, \alpha_i\rangle$ $(1 \leq i \leq n)$, where X neither occurs in $\mathcal{K}$ nor in C_i, then the following holds:*

$$bed(\mathcal{K} \cup h_X, a{:}X) \quad = \quad bed(\mathcal{K}, a{:}\alpha_1 \cdot C_1 \sqcup \ldots \sqcup \alpha_n \cdot C_n)\ .$$

The main computational aspects of PN-OWL involve the computation of the values p_a, n_a, confidence (Eq. (16)), support (Eq. (17)), (θ) coverage and information gain, where the latter (see Eq. (A.1)) refers to the computation of the confidence and cardinality. Now, by definition, confidence and support boil down to compute the cardinality of a concept (Eq. (4)), which in turn depends on the best entailment degree of a concept assertion. By Propositions 4 and 5, also the computation of the values p_a, n_a (and, thus $h_T(a)$) and (θ) coverage can be computed via the best entailment degree of concept assertions and, thus, we may focus on the latter problem only. That is, on determining $bed(\mathcal{K}, a{:}C)$, where $\mathcal{K}$ is a crisp KB and C is either an $\mathcal{EL}_w(\mathbf{D})$ concept or a disjunction of $\mathcal{EL}_w(\mathbf{D})$ concepts.

Now, if the truth space is restricted to be finite, *e.g.*, $L_n = \{0, \frac{1}{n}, \frac{2}{n}, \ldots, \frac{(n-1)}{n}, 1\}$ (for integer $n > 1$), then $bed(\mathcal{K}, a{:}C)$ can be computed in finite time. In fact, in that case, following [4,62–64], we may map a fuzzy KB into a crisp KB and determine $bed(\mathcal{K}, a{:}C)$ by means of a finite number of crisp $\mathcal{SROIQ}(\mathbf{D})$, *viz.* OWL 2, KB satisfiability tests. However, such a solution is not computationally efficient.

We instead, in our implemented system, compute the *bed* by means of the so-called *Operational Research-based* (OR-based) fuzzy tableau algorithm method [4,19,61]. These are based on the fact that[12]

$$bed(\mathcal{K}, a{:}C) = \min x.\ s.t.\ \mathcal{K} \cup \{\langle a{:}\neg C, 1 - x\rangle\} \text{ is satisfiable}, \tag{B.1}$$

where x is a $[0, 1]$ valued variable.

The general idea of these algorithms is the following. We start constructing a *completion-forest* for $\mathcal{K} \cup \{\langle a{:}\neg C, 1 - x\rangle\}$ in which the application of satisfiability preserving *completion rules* generate new fuzzy assertion axioms together with inequations over $[0, 1]$-valued variables. These inequations have to hold in order to respect the semantics of the DL constructors. Then we minimize the original variable x such that all constraints are satisfied. Due to the language restrictions we have in our setting, we end up with a *bounded Mixed Integer Linear Program* (MILP) optimization problem [65,66].

For the uninformed reader, we succinctly recap roughly that a completion-forest $\mathcal{F}$ for $\mathcal{K}$ is a collection of trees whose distinguished roots are arbitrarily connected by edges. The nodes can be abstract or concrete, depending if they represent abstract or concrete individuals.

- Each abstract node v is labelled with a set $\mathcal{L}(v)$ of concepts C. If $C \in \mathcal{L}(v)$ then we consider a variable $x_{v:C}$. The intuition is that v is an instance of C to degree greater or equal than the value of the variable $x_{v:C}$.
- Each concrete node c is labelled with a set $\mathcal{L}(c)$ of fuzzy predicates. For each concrete node c, we consider a variable x_c representing the value of the concrete individual c.
- Each edge $\langle v, w\rangle$ is labelled with a set $\mathcal{L}(v, w)$ of roles r and if $r \in \mathcal{L}(v, w)$ then we consider a variable $x_{(v,w):r}$ representing the degree of being $\mathcal{L}(v, w)$ an instance of r.

[12] Note that $\langle a{:}\neg C, 1 - x\rangle$ expresses informally that $a{:}C \leq x$.

Table B.6
Examples of OR-based fuzzy completion rules.

(var_1)	If $x \in \{x_{v:A}, x_{(v,w):r}\}$ occurs in $C_{\mathcal{F}}$ then $C_{\mathcal{F}} = C_{\mathcal{F}} \cup \{x \in \{0,1\}\}$
(var_2)	If $x \in \{x_{v:C}\}$ occurs in $C_{\mathcal{F}}$ and C not crisp then $C_{\mathcal{F}} = C_{\mathcal{F}} \cup \{x \in L_n\}$
$(\sqcap)$	If $C \sqcap D \in \mathcal{L}(v)$, node v is not indirectly blocked and $\{C, D\} \nsubseteq \mathcal{L}(v)$ then *(i)* add C, D to $\mathcal{L}(v)$, and *(ii)* $C_{\mathcal{F}} = C_{\mathcal{F}} \cup \{x_{v:C} \otimes x_{v:D} \geq x_{v:C \sqcap D}\}$
$(\sqcup)$	If $C \sqcup D \in \mathcal{L}(v)$, node v is not indirectly blocked and $\{C, D\} \nsubseteq \mathcal{L}(v)$ then *(i)* add C, D to $\mathcal{L}(v)$, and *(ii)* $C_{\mathcal{F}} = C_{\mathcal{F}} \cup \{x_{v:C} \oplus x_{v:D} \geq x_{v:C \sqcup D}\}$
$(\top)$	If $\top \sqsubseteq C \in \mathcal{K}$, node v is not indirectly blocked and $C \notin \mathcal{L}(v)$ then add C to $\mathcal{L}(v)$.

The forest has associated a set $C_{\mathcal{F}}$ of MILP constraints of the form $l \leq l'$, $l = l'$, $x_i \in L_n$, $y_i \in \{0,1\}$, where l, l' are linear expressions using the variables occurring in the forest.

Now, we initialize a forest $\mathcal{F}$ for $\mathcal{K} \cup \{\langle a{:}\neg C, 1 - x\rangle\}$ to contain:

- a root node a, for each individual a occurring in $\mathcal{K}$;
- an edge $\langle a, b\rangle$, for each role assertion $(a,b){:}r \in \mathcal{K}$;
- for each $a{:}D \in \mathcal{K}$, we add both D to $\mathcal{L}(a)$ and $x_{a:D} \geq 1$ to $C_{\mathcal{F}}$;
- for $\langle a{:}\neg C, 1 - x\rangle$, we add both $\neg C$ to $\mathcal{L}(a)$ and $x_{a:\neg C} \geq 1 - x$ to $C_{\mathcal{F}}$;
- for each $(a,b){:}r \in \mathcal{K}$, we add both r to $\mathcal{L}(a,b)$ and $x_{(a,b):r} \geq 1$ to $C_{\mathcal{F}}$.

$\mathcal{F}$ is then expanded by repeatedly applying completion rules and the completion-forest is *complete* when none of the completion rules are applicable. To guarantee termination, we apply the same *blocking strategies* (*viz.* pairwise blocking) as for $\mathcal{SROIQ}$ [67]. Then, the MILP problem on the set of constraints $C_{\mathcal{F}}$ is solved over L_n.

The OR-based completion rules for fuzzy datatypes can be found in [4,61], while the OR-based completion rules dealing with weighted concepts can be found in [4,19]. Concerning the OR-based completion rules to be applied to $\mathcal{K}$, as $\mathcal{K}$ is crisp, they are as for crisp $\mathcal{SROIQ}$ [67], but naturally transformed into their (OR-based) version (see, *e.g.*, [4] for a significant excerpt). Examples of completion rules are shown in Table B.6 ($\otimes = \min, \oplus = \max$).

Of course, in our implementation various strategies have been adopted to improve the execution time of PN-OWL and Fuzzy FOIL-$\mathcal{DL}$, whose description is, however, out of the scope of this paper.

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, vol. 3, 2015, pp. 1105–1181, Ch. XVI.
[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, Int. J. Approx. Reason. 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, in: CEUR Workshop Proc., vol. 2396, 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 Min. Knowl. Discov. 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] U. Straccia, Description logics with fuzzy concrete domains, in: Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence (UAI-05), AUAI Press, 2005, pp. 559–567.
[12] P. Hájek, Making fuzzy description logics more general, Fuzzy Sets Syst. 154 (1) (2005) 1–15.
[13] G.J. Klir, B. Yuan, Fuzzy sets and fuzzy logic: theory and applications, J. Chem. Inf. Comput. Sci. (1996) 619.
[14] B. Kosko, Counting with fuzzy sets, IEEE Trans. Pattern Anal. Mach. Intell. 8 (4) (1986) 556–557, https://doi.org/10.1109/TPAMI.1986.4767822.
[15] J.C. Bezdek, Pattern Recognition with Fuzzy Objective Function Algorithms, Springer, New York, NY, 1981.
[16] F.A. Cardillo, U. Straccia, Fuzzy OWL-BOOST: learning fuzzy concept inclusions via real-valued boosting, Fuzzy Sets Syst. 438 (2022) 164–186.
[17] 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.
[18] I. Huitzil, F. Bobillo, U. Straccia, J. Gomez-Romero, Fudge: fuzzy ontology building with consensuated fuzzy datatypes, Fuzzy Sets Syst. 401 (2020) 91–112.
[19] F. Bobillo, U. Straccia, Aggregation operators for fuzzy ontologies, Appl. Soft Comput. 13 (9) (2013) 3816–3830.
[20] V. Torra, Y. Narukawa, Modeling Decisions: Information Fusion and Aggregation Operators, Springer, Berlin, Heidelberg, 2007.
[21] U. Straccia, Reasoning within fuzzy description logics, J. Artif. Intell. Res. 14 (2001) 137–166.

[22] F.A. Lisi, U. Straccia, Learning in description logics with fuzzy concrete domains, Fundam. Inform. 140 (3–4) (2015) 373–391.
[23] 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.
[24] F.A. Lisi, U. Straccia, A logic-based computational method for the automated induction of fuzzy ontology axioms, Fundam. Inform. 124 (4) (2013) 503–519.
[25] P. Westphal, L. Bühmann, S. Bin, H. Jabeen, J. Lehmann, SML-bench - a benchmarking framework for structured machine learning, Semant. Web 10 (2) (2019) 231–245.
[26] F. Bobillo, U. Straccia, Reasoning within fuzzy OWL 2 EL revisited, Fuzzy Sets Syst. 351 (2018) 1–40.
[27] The FuzzyDL-learner system, http://www.umbertostraccia.it/cs/software/FuzzyDL-Learner/.
[28] D. Dua, C. Graff, UCI machine learning repository, http://archive.ics.uci.edu/ml, 2017.
[29] 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.
[30] L. Badea, S. Nienhuys-Cheng, A refinement operator for description logics, in: Inductive Logic Programming, 2000, pp. 40–59.
[31] 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.
[32] C. d'Amato, Machine learning for the semantic web: lessons learnt and next research directions, Semant. Web 11 (1) (2020) 195–203.
[33] J. Lehmann, P. Hitzler, Foundations of refinement operators for description logics, in: Inductive Logic Programming, 2008, pp. 161–174.
[34] J. Lehmann, P. Hitzler, Concept learning in description logics using refinement operators, Mach. Learn. 78 (2010) 203–250.
[35] F.A. Lisi, D. Malerba, Ideal refinement of descriptions in AL-log, in: Inductive Logic Programming, 2003, pp. 215–232.
[36] M. Serrurier, H. Prade, Improving expressivity of inductive logic programming by learning different kinds of fuzzy rules, Soft Comput. 11 (2007) 459–466.
[37] 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.
[38] M. Drobics, U. Bodenhofer, E.-P. Klement, FS-FOIL: an inductive learning method for extracting interpretable fuzzy descriptions, Int. J. Approx. Reason. 32 (2–3) (2003) 131–152.
[39] 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.
[40] N. Fanizzi, G. Rizzo, C. d'Amato, F. Esposito, DLFoil: class expression learning revisited, in: Knowledge Engineering and Knowledge Management, 2018, pp. 98–113.
[41] G. Rizzo, N. Fanizzi, C. d'Amato, Class expression induction as concept space exploration: from DL-Foil to DL-Focl, Future Gener. Comput. Syst. 108 (2020) 256–272.
[42] N. Fanizzi, G. Rizzo, C. d'Amato, Boosting DL concept learners, in: The Semantic Web, 2019, pp. 68–83.
[43] R. Nock, F. Nielsen, A real generalization of discrete AdaBoost, Artif. Intell. J. 171 (1) (2007) 25–41.
[44] 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.
[45] J. Otero, L. Sánchez, Induction of descriptive fuzzy classifiers with the logitboost algorithm, Soft Comput. 10 (9) (2006) 825–835.
[46] 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.
[47] L. Sánchez, J. Otero, Boosting fuzzy rules in classification problems under single-winner inference, Int. J. Intell. Syst. 22 (9) (2007) 1021–1034.
[48] 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, in: Advances in Intelligent Systems and Computing, vol. 443, 2016, pp. 129–136.
[49] S. Bloehdorn, Y. Sure, Kernel methods for mining instance data in ontologies, in: The Semantic Web, vol. 4825, 2007, pp. 58–71.
[50] N. Fanizzi, C. d'Amato, F. Esposito, Induction of robust classifiers for web ontologies through kernel machines, J. Web Semant. 11 (2012) 1–13.
[51] 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.
[52] G. Rizzo, C. d'Amato, N. Fanizzi, F. Esposito, Tree-based models for inductive classification on the web of data, J. Web Semant. 45 (2017) 1–22.
[53] G. Rizzo, N. Fanizzi, C. d'Amato, F. Esposito, Approximate classification with web ontologies through evidential terminological trees and forests, Int. J. Approx. Reason. 92 (2018) 340–362.
[54] 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.
[55] M. Zhu, Z. Gao, J.Z. Pan, Y. Zhao, Y. Xu, Z. Quan, Tbox learning from incomplete data by inference in BelNet+, Knowl.-Based Syst. 75 (2015) 30–40.
[56] J. Lehmann, Hybrid learning of ontology classes, in: Machine Learning and Data Mining in Pattern Recognition, 2007, pp. 883–898.
[57] M. Nickles, A. Rettinger, Interactive relational reinforcement learning of concept semantics, Mach. Learn. 94 (2) (2014) 169–204.
[58] G. Rizzo, C. d'Amato, N. Fanizzi, An unsupervised approach to disjointness learning based on terminological cluster trees, Semant. Web 12 (3) (2021) 423–447.
[59] F. Bobillo, U. Straccia, The fuzzy ontology reasoner fuzzyDL, Knowl.-Based Syst. 95 (2016) 12–34.
[60] 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.
[61] F. Bobillo, U. Straccia, Fuzzy description logics with general t-norms and datatypes, Fuzzy Sets Syst. 160 (23) (2009) 3382–3402.
[62] F. Bobillo, U. Straccia, Reasoning with the finitely many-valued Łukasiewicz fuzzy description logic $\mathcal{SROIQ}$, Inf. Sci. 181 (2011) 758–778.
[63] F. Bobillo, U. Straccia, Fuzzy description logics under Gödel semantics, Int. J. Approx. Reason. 50 (3) (2009) 494–514.
[64] U. Straccia, Transforming fuzzy description logics into classical description logics, in: Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA-04), in: LNCS, vol. 3229, 2004, pp. 385–399.
[65] G.L. Nemhauser, Integer and Combinatorial Optimization, Wiley-Interscience, 1999.
[66] H.M. Salkin, K. Mathur, Foundations of Integer Programming, North-Holland, 1989.
[67] I. Horrocks, O. Kutz, U. Sattler, The even more irresistible $\mathcal{SROIQ}$, in: Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR-06), 2006, pp. 57–67.
[68] V. Vapnik, Principles of Risk Minimization for Learning Theory, Advances in Neural Information Processing Systems, 1991.