Boosting DL Concept Learners

Nicola Fanizzi, Giuseppe Rizzo(✉), and Claudia d'Amato

LACAM – Dipartimento di Informatica & CILA,
Università degli Studi di Bari Aldo Moro, Bari, Italy
{nicola.fanizzi,giuseppe.rizzo1,claudia.damato}@uniba.it
http://lacam.di.uniba.it

Abstract. We present a method for boosting relational classifiers of individual resources in the context of the *Web of Data*. We show how weak classifiers induced by simple concept learners can be enhanced producing strong classification models from training datasets. Even more so the comprehensibility of the model is to some extent preserved as it can be regarded as a sort of concept in disjunctive form. We demonstrate the application of this approach to a weak learner that is easily derived from learners that search a space of hypotheses, requiring an adaptation of the underlying heuristics to take into account weighted training examples. An experimental evaluation on a variety of artificial learning problems and datasets shows that the proposed approach enhances the performance of the basic learners and is competitive, outperforming current concept learning systems.

1 Introduction and Motivation

The capability of assigning individual resources to their respective classes plays a central role in many applications related to the *Web of Data* [7] for carrying out tasks such as *completion* or *type* and *link prediction*. Indeed, these tasks can be regarded as relying on the classification of new individual resources, i.e. the prediction of their membership w.r.t. given classes (e.g. see [13]) or the fillers of given properties for fixed subjects [11,15]. Developing such a capability on the ground of sole semantic features ascertained via purely logic-based methods, can be computationally expensive and sometimes also scarcely effective, owing to the inherent incompleteness of the knowledge bases in such a context. Improvements in computational complexity, accuracy and predictiveness may be often obtained with a limited trade-off with the comprehensibility [3] of the classification model. To reach this compromise, a different approach can be pursued by *boosting* a simple (greedy) learner, that amounts to generate a sequence of weak classifiers for the target class(es) to form stronger classification models [16].

In the case of concept learning, weak learners can be derived from full-fledged conventional learners, such as those that search a space of hypotheses guided by some heuristics (e.g. DL-Foil [5,6], Celoe or other algorithms in the DL-Learner suite [2]): the boosting process iterates the production of weak classifiers for the ensemble model adjusting the weights of the examples to drive

P. Hitzler et al. (Eds.): ESWC 2019, LNCS 11503, pp. 68–83, 2019.
https://doi.org/10.1007/978-3-030-21348-0_5

the focus towards covering *harder* examples. The induced classifiers will have the desirable property that the class assigned to an instance depends on the classifiers that cover it. This property makes classifications easier to understand, and is made possible by requiring: (a) suitable constraints from the adopted weak learner; (b) the use of a generalization of the basic boosting *meta-learning* schema [16]. The resulting classification model is simpler and formally better-understood than other ensemble models [3]. Even more so, it can scale better on large datasets and can be extremely effective.

In this work, we show how to apply a *constrained confidence-rated boosting* strategy to a weak learner extracted from existing concept learners, namely a simplified version of their core refinement-search routine, with a different heuristic function that takes into account the *hardness* of the examples to drive the search for refinements. The resulting solution can be regarded as a generalized framework allowing the application of the proposed boosting schema to any conventional learners, that is also those adopting different classification models, e.g. terminological decision trees [13]. Moreover this schema tends to produce more *stable* classifiers.

We performed an empirical evaluation of the proposed approach, applying the boosting schema to a weak learner, wDLF, derived from the latest release of DL-Foil [6]. For a comparative evaluation, we also considered the conventional concept learners DL-Foil [5] and Celoe [10], from the DL-Learner suite [2]. The experiments, carried out on artificial learning problems of varying difficulty with different datasets, confirmed empirically the expected enhancement of the ensemble models with respect to the weak classifiers induced by the basic learners and show an improved effectiveness also with respect to the conventional competitors. Interestingly, we observed that generally small numbers of iterations (hence of weak classifiers) were necessary to obtain accurate strong classifiers so that (efficiency and) comprehensibility of the resulting models is preserved to some extent.

The paper is organized as follows. In the next section, related works are reviewed. In Sect. 3 we recall the learning problem cast as an error minimization task. In Sect. 4 the basics of boosting are presented and we show how to apply it to the specific learning problem, while Sect. 5 shows how to adapt available (weak) concept learners as auxiliary routines for the presented boosting method. In Sect. 6 a comparative empirical evaluation with conventional systems on various datasets and related learning problems is reported. Further possible developments are suggested in Sect. 7.

2 Related Work

Ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. Specifically, boosting refers to a family of algorithms that are able to convert weak learners to strong learners. The main principle is to fit a sequence of weak learners to weighted versions of the data. Examples that were

misclassified by earlier rounds get their weight increased. The predictions made by weak hypotheses are then combined through a weighted majority vote to produce the final strong prediction. The most common implementation of boosting is AdaBoost_M1 [16].

One of the first rule-learning systems based on a generalized version of AdaBoost is SLIPPER [3]. It applied the AdaBoost scheme to weak learners based on a simple and efficient generate-and-prune rule construction method that produces rule-sets, where each propositional rule is labeled with a confidence value used in the final voting procedure determining the classification predicted by the whole model.

One of the first applications of the boosting approach to First-Order rule learners involved Foil [12]. Another system, C^2RIB [8], produces strong binary classifiers in the form of clausal rule-sets, with each clause endowed with its confidence (and a single default clause for the negative case). The main difference to our proposal is that these algorithms are designed for a binary classification problem, where only positive and negative examples are available. We target ternary settings that can accommodate the case of uncertain-membership individuals w.r.t. the target concept which generally abound in the linked datasets due to the lack of disjointness axioms in the related vocabularies (ontologies).

A different trade-off between accuracy and comprehensibility of classifiers for semantically annotated resources was pursued through tree-based models (see below). A sort of bagging algorithm for the induction of *terminological random forests* has been devised [13] aiming at ensemble models to enhance accuracy and stability. This idea has been further developed in the algorithms for *evidential random forests* [14] as classification models that integrate uncertainty reasoning in prediction by resorting to the *Dempster-Shafer Theory*.

As regards concept learning, supervised approaches grounded on the idea of *generalization as search*, enacted by operators that are specifically designed for DL languages, have been proposed and implemented in learning systems like YinYang [9], DL-Foil [5,6] and DL-Learner [2]. YinYang produces concept descriptions by resorting to the notion of *counterfactuals* [9] for specializing overly general concepts by ruling out negative examples for their extensions. DL-Foil is grounded on a sequential covering strategy and exploits a (downward) refinement operator to traverse the search space and a heuristic, inspired by *information gain*, to select among the candidate specializations [5,6]. DL-Learner is a suite that collects the implementation of various concept learning algorithms which essentially differ in the heuristics adopted as scoring functions for candidate refinements. In particular Celoe [10] performs an accuracy-driven search for solutions biased towards (syntactically) simpler concept definitions.

Alternative classification models, such as logical decision trees, are targeted by different strategies. Specifically, *terminological decision trees* [13], i.e. decision trees with test-nodes containing DL concept descriptions, and procedures for their conversion into disjunctive descriptions, adopt a *divide-and-conquer* learning strategy (instead of the *separate-and-conquer* strategy embraced by most of the algorithms). Similarly, Parcel and its extension SPaCEL [17] adopt the

separate-and-conquer strategy but it is based on the use of a set of workers generating partial solutions, which are subsequently combined to generate a complete concept description. Both PARCEL and SPACEL use CELOE as a subroutine for building partial solutions, thus the quality of the final concept strictly depends on the quality of such class expressions.

Another related approach to concept learning is based on *bisimulation* [18], which amounts to a recursive partitioning of a set of individuals, similarly to terminological decision trees. However, instead of resorting to a refinement operator for producing partial solutions, the method exploits a set of pre-computed *selectors*, i.e. tests that are used to partition the set of individuals.

3 The Learning Problem

We consider knowledge bases (KBs) defined through representation languages adopted for the vocabularies in the Web of Data that can ultimately be mapped onto *Description Logics* (DLs) [1].

Basically, a KB can be regarded as a set $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$ of inclusion axioms (in the *TBox* $\mathcal{T}$) in terms of *subsumption* ($\sqsubseteq$), and assertions (in the *ABox* $\mathcal{A}$) that describe, respectively, concepts (and relations) and individuals.

Given a limited set of training examples, i.e. individuals labeled with an intended membership w.r.t. a given target concept, the goal is to induce a function (*classification model* or *classifier*) that is able to predict the correct membership for further unseen individuals. More formally, the problem is:

Given:

- a knowledge base $\mathcal{K} = \mathcal{T} \cup \mathcal{A}$, with $\mathsf{Ind}(\mathcal{A})$ denoting the set of its individuals;
- a *target concept* C associated with an unknown classification function $f_C : \mathsf{Ind}(\mathcal{A}) \rightarrow \{+1, -1, 0\}$ such that, for each $a \in \mathsf{Ind}(\mathcal{A})$, the possible values (*labels*) correspond, respectively, to the cases $\mathcal{K} \models C(a)$, $\mathcal{K} \models \neg C(a)$, and uncertain membership (default);
- a set of classifiers, or *space of hypotheses*: $\mathcal{H} = \{h : \mathsf{Ind}(\mathcal{A}) \rightarrow \{+1, -1, 0\}\}$
- a *training set* $\mathbf{T} = \mathbf{Ps} \cup \mathbf{Ns} \cup \mathbf{Us}$ of examples, i.e. individual-label pairs, where each individual is associated with the unique known membership-label l for f_C, that corresponds to its intended classification (*ground truth*) w.r.t. C:
  - $\mathbf{Ps} = \{(a, +1) \mid a \in \mathsf{Ind}(\mathcal{A}),\ f_C(a) = +1\}$; *positive examples*
  - $\mathbf{Ns} = \{(b, -1) \mid b \in \mathsf{Ind}(\mathcal{A}),\ f_C(b) = -1\}$; *negative examples*
  - $\mathbf{Us} = \{(c, 0) \mid c \in \mathsf{Ind}(\mathcal{A}),\ f_C(c) = 0\}$ *uncertain-membership examples*

Learn: a classifier $H \in \mathcal{H}$ that minimizes the *empirical risk of error* $\hat{R}$ on $\mathbf{T}$:

$$H = \arg\min_{h \in \mathcal{H}} \hat{R}(h, \mathbf{T}) = \arg\min_{h \in \mathcal{H}} \frac{1}{n} \sum_{(x,l) \in \mathbf{T}} L(h(x), l)$$

where L is a *loss function* $L : \{-1, 0, +1\}^2 \rightarrow \mathbb{R}$, a cost function penalizing incorrect label assignments by the classifier. The accuracy of the induced classifier can be then assessed by measuring $\hat{R}(H, \mathbf{T}')$ on a separate *test set* $\mathbf{T}'$ of examples (i.e. $\mathbf{T} \cap \mathbf{T}' = \emptyset$).

Further constraints can be introduced on the set of solutions as preference criteria (e.g. specific loss functions or syntactic measures of complexity). Such constraints are aimed at avoiding hypotheses that overfit $\mathbf{T}$ remaining poorly predictive w.r.t. $\mathbf{T}'$ [4].

It is also important to note that the notion of negative example, say $(b, -1)$, corresponding to $\mathcal{K} \models \neg C(b)$, is stronger if compared to other settings, yet it is more coherent with the underlying open-world semantics that is adopted in the targeted setting. Other related approaches favour a weaker binary setting where positive and *non-positive* examples are considered, namely $\mathcal{K} \not\models C(b)$, which leads to targeting (extensionally) narrower concepts, hence classifiers that may turn out to be poorly predictive w.r.t. newly acquired instances as long as the KB population evolves.

Example 1 (simple problem). Given the following KB, suppose the goal is finding an hypothesis for classifying the individuals w.r.t. the concept `Father`.

$\mathcal{K} = \{\mathtt{Man} \sqsubseteq \mathtt{Person}, \mathtt{Woman} \sqsubseteq \mathtt{Person}, \mathtt{Artist} \sqsubseteq \mathtt{Person}, \mathtt{Man} \sqsubseteq \neg\mathtt{Woman}\} \cup$
$\{\mathtt{Man}(a), \mathtt{Man}(b), \mathtt{Man}(c), \mathtt{Woman}(d), \mathtt{Woman}(f), \mathtt{Artist}(e), \mathtt{Robot}(g),$
$\mathtt{hasChild}(a, d), \mathtt{hasChild}(b, e)\}$.

A training set $\mathbf{T}$ may include: $\mathbf{Ps} = \{a, b\}$ (instances of `Man` with a known child), $\mathbf{Ns} = \{d, f\}$ (instances of `Woman`) and $\mathbf{Us} = \{c, e, g\}$.

4 Boosting a Concept Learner

In this section we present the proposed boosting-based framework. The framework is grounded on the application of a boosting-schema to concept learners. Specifically, boosting is used to create an ensemble classifier made up of class expressions. A *weak learner* is employed to find a single classifier (e.g. a rule, a clause or, as in our case, a single class expression), essentially by means of the same process used in the inner loops of the typical learning algorithms based on *sequential covering* (such as DL-Foil or Celoe). The heuristics adopted to guide the construction of the classifier can be derived from the formal analysis of boosting-based learners.

4.1 Basics of Boosting

Boosting is a meta-learning method that is applicable to (weak) learners to get ensemble models that enhance the performance of the single classifiers. The original algorithm AdaBoost_M1 relies on a weak learner that is repeatedly invoked to get a series of *weak hypotheses* whose predictiveness is only required to be slightly better than random guessing. On each round, the weak learner generates a single classifier that determines changes in the distribution of the weights associated with the examples: the weights of misclassified examples get increased so that better classifiers can be produced in the next rounds knowing the harder examples to focus on. The weak hypotheses produced in various rounds are finally combined into a *strong hypothesis*, an ensemble that makes predictions via a majority vote procedure.

Algorithm 1. Generalized AdaBoost

```
input T = {(e_i, l_i)}_{i=1}^N
output H: Hypothesis {strong hypothesis}
for i = 1 to N do w_i^(1) = 1/N      {weight initialization}
for t = 1 to T do   {T is the number of iterations/hypotheses}
    h_t ← WeakLearner(T, w^(t))      {next weak hypothesis}
    Choose α_t ∈ ℝ
    for i = 1 to N do w_i^(t+1) ← w_i^(t) exp(−α_t l_i h_t(e_i))/Z_t
H ← λx.sign(Σ_{i=1}^T α_t h_t(x))
return H
```

A generalization of AdaBoost [16] produces models with a confidence value associated with each weak hypothesis that is used to weight their predictions (optionally even *declining* to vote/predict by assigning a null confidence to *uncovered* instances). The ensemble prediction takes into account the confidence values associated with each weak classifier. Formally, a set of examples $\mathbf{T} = \{(e_1, l_1), ..., (e_m, l_m)\}$ is given, where each instance e_i belongs to a domain $\mathbf{E}$ and each label l_i is in $\{-1, +1\}$. Also, a weak learning algorithm is assumed to require a distribution of weights $\boldsymbol{w}$ over the training examples (initially considered as uniform) indicating their *hardness*. In the generalized setting, the weak learner computes a weak hypothesis h_t of the form $h_t : \mathbf{E} \rightarrow \mathbb{R}$ where, given an instance x, the *sign* of $h_t(x)$ is interpreted as the predicted label while its *magnitude* $|h_t(x)|$ is interpreted as the amount of confidence in the prediction. The weak hypothesis may even abstain from predicting the label ($h_t(x) = 0$).

This boosting algorithm is described in Algorithm 1; here the constant Z_t ensures the normalization of each $\boldsymbol{w}^{(\cdot)}$ and parameter α_t depends on the weak learner. The returned strong hypothesis is indicated as a lambda abstraction $\lambda x.\text{sign}(\cdot)$.

4.2 Learning Confidence-Rated Hypotheses in DL

In related settings [3,8] the targeted strong hypothesis H is a set of hypotheses (rules/clauses), each associated with a confidence value. A single *default hypothesis* is associated with a negative confidence (i.e., the case when a prediction cannot be made confidently). To classify an individual, the sum of the confidences of all hypotheses covering it is computed, then the prediction is made according to the sign of this sum.

Our solution deviates from these settings to address the presented *ternary* classification problems: the (unique) label l_i associated with each $e_i \in \mathsf{Ind}(\mathcal{A})$ is selected from $\{+1, -1, 0\}$. The targeted strong hypothesis H is an ensemble of concepts (i.e. class expressions) C_t, each associated with a confidence c_t.

Example 2 (confidence-rated ensemble model). Given the learning problem in Example 1, an induced ensemble model H may be:

$$\{\texttt{Person}\ [0.4], \neg\texttt{Woman}\ [0.8], \texttt{Person}\ [0.5], \exists\texttt{hasChild}.\top\ [0.9]\}.$$

Algorithm 2. DL Concept Learner Boosting: BOOST(WEAKLEARNER,$\mathbf{T}$): M

```
input WEAKLEARNER: Learning Function
      T = {(e_i, l_i)}_{i=1,...,N}  { training set }
output M: Classification Model {list of concepts with their confidence}
for i = 1 to N do w_i^(1) = 1/N
for t = 1 to T do
    C_t ← WEAKLEARNER(T, w^(t)) {concept learner}
    Let ĉ_t be computed as in Eq. (2), using α̂_t and h_t based on C_t
    for each i = 1 to N do w'_i ← w_i^(t) exp(−ĉ_t) {weights update}
    Z_t ← Σ_{i=1}^N w'_i
    for each i = 1 to N do  w_i^(t+1) ← w'_i / Z_t {weights normalization}
return {C_t [ĉ_t]}_{t=1,...,T}
```

The algorithm will generate only concepts associated with a positive confidence – that is, classifiers that can predict a definite membership w.r.t. the target class. Moreover, an implicit *default classifier* can be associated with a negative confidence (i.e., the case when a classification cannot be predicted confidently).

Coverage. As a consequence of representing hypotheses as DL concepts (with their semantics), the notion of *coverage* has to differ from the standard settings for binary problems (based on instances that trigger rules), namely:

Definition 1 (coverage). *A positive, resp. negative, example x is* covered *by a concept C when $\mathcal{K} \models C(x)$, resp. $\mathcal{K} \models \neg C(x)$. An uncertain-membership example is* covered *by C when both $\mathcal{K} \not\models C(x)$ and $\mathcal{K} \not\models \neg C(x)$ hold.*

This new notion has an impact on the setting of the confidence values: a positive confidence is assigned to covered examples whereas a negative one is assigned to uncovered examples. Therefore, denoting with h_t the classifier $h_t : \mathbf{E} \to \{-1, 0, +1\}$, that predicts a label based on concept C_t along with the entailments in Definition 1, the confidence associated to C_t can be defined as follows:

$$c_t = -\alpha_t \cdot (-1)^{I(l_i, h_t(e_i))}$$

where $I(\cdot, \cdot)$ denotes the indicator function for the *identity relation* (returning 1 in case of identity and 0 otherwise).

Hence the sign of c_t corresponds to the cases of instances e_i that are covered/uncovered by C_t (alternatively, uncovered examples may be assigned to a null confidence by h_t, similarly to the original setting).

The resulting boosting procedure is shown as Algorithm 2. Essentially it creates a model as a list of class expressions together with their confidence. This representation can be regarded as a generalization of a *Disjunctive Normal Form* adopted by many DL concept learners. The model can be made more compact by merging equivalent concepts in a single hypothesis whose confidence sums up the related confidence values:

Example 3 (compacted ensemble model). An ensemble model derived from the one presented in Example 2 is: $\{\texttt{Person}\ [0.9], \neg\texttt{Woman}\ [0.8], \exists\texttt{hasChild}.\top\ [0.9]\}$.

Algorithm 3. Strong Classifier: CLASSIFY$(M, x) : l_x$

```
input {C_t [ĉ_t]}_{t=1,...,T}: Classification Model
      x: Ind(A)  {query individual}
output l_x ∈ {−1, 0, +1}   {predicted label}
for each l ∈ {−1, 0, +1} do s_l ← 0
for t = 1 to T do
   if K ⊨ C_t(x) do s_{+1} ← s_{+1} + ĉ_t {i.e. when h_t(x) = +1}
   else if K ⊨ ¬C_t(x) do s_{−1} ← s_{−1} + ĉ_t {i.e. when h_t(x) = −1}
        else s_0 ← s_0 + ĉ_t {i.e. when h_t(x) = 0}
return argmax_{l∈{−1,0,+1}} s_l
```

This represents a *strong classifier* that is used for classifying unseen instances. The related procedure is illustrated in Algorithm 3. Note that, differently from the original binary setting, to get a ternary classifier, the confidence values are collected separately per each label, with 0 as default case. The final classification is decided with a majority vote.

Optimization. In Algorithm 2, $\boldsymbol{w}$ is normalized using Z_t, that depends on c_t:

$$Z_t = \sum_{i=1,\ldots,N} w_i^{(t)} \exp(-c_t) = \sum_{i=1,\ldots,N} w_i^{(t)} \exp\left(-\left(-\alpha_t \cdot (-1)^{I(l_i, h_t(e_i))}\right)\right) \quad (1)$$

It was shown [3,16] that to minimize the training error, on each round the weak learner should pick the h_t and α_t which lead to the *smallest* value of Z_t. Thus, given a generated C_t, its confidence c_t, based on α_t and h_t, should be set to minimize Z_t. To this purpose the sum can be split up separating covered from uncovered examples. This simplifies the arguments of the exponential assigning opposite signs to the confidence magnitude α_t:

$$Z_t = \sum_{\substack{i=1,\ldots,N \\ I(l_i, h_t(e_i))=1}} w_i^{(t)} \exp(-\alpha_t) + \sum_{\substack{i=1,\ldots,N \\ I(l_i, h_t(e_i))=0}} w_i^{(t)} \exp(+\alpha_t).$$

Now letting

$$W_c = \sum_{i|I(l_i, h_t(e_i))=1} w_i^{(t)} \quad \text{and} \quad W_u = \sum_{i|I(l_i, h_t(e_i))=0} w_i^{(t)},$$

it can be further simplified as follows:

$$Z_t = W_c \exp(-\alpha_t) + W_u \exp(+\alpha_t)$$

The value that minimizes Z_t, obtained solving $\frac{\partial}{\partial \alpha_t} Z_t = 0$, is $\alpha_t = \ln\left(W_c / W_u\right)/2$ or, equivalently [16], its approximation $\hat{\alpha}_t$ obtained adding a *smoothing factor* s^{-1}, with $s = 2N$:

$$\hat{\alpha}_t = 0.5 \ln\left[(W_c + 1/s)/(W_u + 1/s)\right]. \quad (2)$$

Then the *smoothed* confidence $\hat{c}_t$ of any hypothesis is bounded from above by $0.5 \ln(s)$ [3]. The analysis also suggests an objective function to be used by the weak learner [16]. Plugging the value of α_t in the equation above:

$$Z_t = 1 - \left(\sqrt{W_c} - \sqrt{W_u}\right)^2 .$$

As noted in [3], a hypothesis h_t minimizes Z_t iff it maximizes $|\sqrt{W_c} - \sqrt{W_u}|$. Such a h_t may be negatively correlated with the coverage, hence its confidence c_t is negative. Considering only positively correlated hypotheses, the objective function to be maximized by the weak learner searching for a suitable C_t is:

$$\sqrt{W_c} - \sqrt{W_u}$$

These considerations will be taken into account in the definition of the score function exploited by the weak learners.

5 Designing a Simple and Fast Weak Learner

In principle, any algorithm could be exploited as a weak learner for the presented ensemble schema, provided that it can return a hypothesis, in the form of a concept or any alternative classifier, based on a set of weighted training examples.

Considering an algorithm that adopts a sequential covering approach for the construction of concepts, one can simply detach and adapt the inner loop for the construction of a disjunct to get a weak learner straightforwardly.

Essentially, it should be required that the underlying search heuristic is adapted to focus towards covering the examples in accordance with their weight distribution, that has to be provided as an additional argument. We will showcase how to make this customization adopting the DL-Foil specializing routine.

5.1 wDLF: A Weak Learner Derived from DL-Foil

The original DL-Foil algorithm [6] features a main routine based on sequential covering that produces a disjunctive concept made up of conjunctive partial descriptions, computed by the specialization procedure, that cover as many positives as possible while iteratively ruling out all of the negative examples. Each conjunctive description is provided by a downward refinement operator traversing the search space driven by a heuristic. Candidate refinements are scored by means of a measure inspired to the *information gain* (see below).

While the role of the main routine is intended to be played by the boosting meta-algorithm, the specialization procedure can be reused as a weak learner, provided that its heuristic biases the search towards concept descriptions that cover especially those examples indicated by the weight distribution as *harder*.

Algorithm 4 illustrates the adapted algorithm wDLF, that can be passed as a weakLearner to the boosting schema. Similarly to the mentioned original procedure that performs a (greedy) heuristic search aiming at (good) specializations of a general description of the target concept, wDLF seeks for proper

Algorithm 4. The Weak Learner wDLF($\mathbf{T}, \boldsymbol{w}$): C

```
1  {constant values set according to the configuration}
2  const ϵ: ℝ {minimal score threshold}
3        N_S: ℕ {number of specializations to be generated}
4  input T = Ps ∪ Ns ∪ Us: training set {i.e. labeled individuals}
5        w: Distribution of the examples {weights}
6  output C: Concept {best in terms of score (otherwise default)}
7  best ← 0
8  C ← INIT(T) {e.g. start with ⊤}
9  repeat
10   for i = 1 to N_S
11     repeat
12       C_ρ ← GETRANDOMREFINEMENT(ρ, C)
13       if (∃(p, +1) ∈ Ps : K ⊨ C_ρ(p)) then
14       score ← SCORING(C, C_ρ, T, w)
15       if (score > best) then
16         C ← C_ρ
17         best ← score
18     until (best > 0)    {weakened constraint w.r.t. the original algo.}
19 until (best > ϵ)
20 return C      {⊤ as default classifier}
```

refinements (i.e. non equivalent concepts in a descending path through the subsumption hierarchy). Starting with the initialization of C with a general concept, the algorithm generates N_S candidate refinements by repeatedly invoking GETRANDOMREFINEMENT. This function randomly selects a specialization C_ρ from the set of the refinements computed by the operator ρ applied to C (see [6] for more details). The resulting concept description must cover at least a positive example and yield a minimal score fixed with a threshold. Note that for speeding up the induction of a weak classifier, unlike the case original more complex routine employed by DL-FOIL, in wDLF the incorrect coverage of some negative or null-labeled examples may be tolerated.

5.2 Heuristics

The heuristic employed in [6] to guide the search for good candidate concepts as possible refinement of the current concept was a *gain* g computed as follows:

$$g(C, C') = p' \cdot \left[\log \frac{p'}{p' + n' + u'} - \log \frac{p}{p + n + u} \right]$$

where p', n' and u' represent, respectively, the numbers of positive, negative and uncertain-membership examples covered by the refinement C', and p, n and u stand for the corresponding numbers of examples covered by C.

Following the previous discussion on optimization, a different heuristic is introduced to take into account the weights distribution:

$$s(C, C', \mathbf{T}, \boldsymbol{w}) = \left(\sqrt{W_c(C')} - \sqrt{W_u(C')} \right) - \left(\sqrt{W_c(C)} - \sqrt{W_u(C)} \right)$$

where $W_c(\cdot)$ and $W_u(\cdot)$ for the concepts C and C' are computed as illustrated in Sect. 4.2. The differences regarding a previous concept can be stored in a data structure for future use to save some computation.

Table 1. Facts concerning the ontologies employed in the experiments

Ontology	Language	#concepts	#object props	#datatype props	#individuals
BioPAX	$\mathcal{ALCHF}(\mathbf{D})$	28	19	30	323
NTN	$\mathcal{SHIF}(\mathbf{D})$	47	27	8	724
HDisease	$\mathcal{SHIF}(\mathbf{D})$	1499	10	15	639
Financial	$\mathcal{ALCIF}$	60	17	0	1000
Geoskills	$\mathcal{SHIF}$	596	23	0	2532

6 Empirical Evaluation

This section illustrates the design and the outcomes of a comparative empirical evaluation aimed at assessing the effectiveness of a boosted weak learner[1] against both the weak learner (see Sect. 4) and full-fledged algorithms. Details concerning the design and outcomes are provided in the following.

6.1 Design and Setup of the Experiments

In order to assess the effectiveness of the compared systems, the quality of the classifiers induced by each of them was evaluated against a baseline of target concepts. To this purpose, we considered publicly available Web ontologies featuring different expressiveness, numbers of individuals and numbers of concepts/roles. The salient characteristics are summarized in Table 1.

Each ontology was considered for producing 15 artificial problems/target concepts and related training & test sets, generated according to the randomized procedure described in [6]. A target concept description per problem was generated by randomly picking concepts/roles from the respective signatures and combining them through basic DL concept constructors: union, conjunction, complement, universal or existential restriction. All the available individuals were labeled as positive, negative, or uncertain-membership examples per target concept via the instance-checking service offered by a reasoner[2]. Note that this procedure a concept is picked only when both positive and negative examples are available. To allow the repetition of the experiments, a seed value can be set in the configuration for the internal pseudo-random generator: 1 in this case.

We adopted the weak learner wDLF described in Algorithm 4 as a baseline and compared the proposed approach against the latest releases of DL-Foil [6] and Celoe [10], that are based on a sequential covering algorithm. In the configuration of both wDLF and DL-Foil, the maximum number of candidate specializations to be evaluated per turn is a required parameter (N_S). This value was empirically set to 20.

[1] The source code, the datasets, the ontologies and supplemental material are publicly available at: https://bitbucket.org/grizzo001/DLbooster/src/master/.

[2] JFact was used: http://jfact.sourceforge.net.

BOOST(WDLF), the boosted version of WDLF, required the maximal number of weak hypotheses (T in Algorithm 2) as an additional parameter. It was set to 10, as larger values were observed not to produce significant improvements of the performance of the resulting strong learners. Throughout the experiments the performance of the ensemble classifiers was not significantly influenced by the value of the hyper-parameter T. A preliminary analysis (*grid search*) had been carried out to find out a good value for T testing the following values: 2, 5, 10, 20, and 30. Thus the increase of performance of the resulting ensembles w.r.t. the baseline of $T = 1$ was assessed in terms of average accuracy. As a result $T = 10$ was recognized as a reasonable tradeoff between ensemble complexity (efficiency) and effectiveness over the various problems and datasets.

As regards CELOE, a proper tuning of the refinement operator was also required. In particular, the operator was configured so that it could return concept descriptions obtained through all the available concept constructors, possibly also using the datatype properties available in the ontologies. Another required parameter is the time-out used to stop the training phase. This value was set to 10 s (larger values were also preliminarily tested but they brought no significant improvement while slowing down the learning process).

For a deeper insight in the cases of incorrect coverage, the following performance indices have been measured (on the test sets):

match rate (M%) cases of test examples with correct coverage;
commission error rate (C%) cases of incorrect coverage of test examples with definite membership: opposite class predicted;
omission error rate (O%) cases of incorrect coverage of test examples with definite membership: the model was unable to predict the membership;
induction rate (I%): cases of test examples with unknown membership: the model predicted a definite membership.

The *.632 bootstrap* resampling method was adopted to assess estimates of these indices with 30 runs of the experiment per problem, randomly varying the partitions of the available examples in training and test sets.

6.2 Aggregate Results and Discussion

The experiments showed interesting results. As an analytic *per problem* presentation would require a lot of space (a total of 75 problems) we report in Table 2 the outcomes (and standard deviations) averaged over the 15 learning problems.

As expected, BOOST(WDLF) produced accurate ensemble classifiers that significantly outperformed those induced by the weak learner. The performance was also equal or slightly superior compared to DL-FOIL and significantly better than CELOE, which tended to produce very simple and poorly predictive definitions often made up of a single (complemented) concept which erroneously covered many negative examples (see the next section for some examples).

Table 2. Aggregate outcomes of the experiments (averages and standard deviations)

Dataset	Index	wDLF	Boost(wDLF)	DL-Foil	Celoe
Biopax	M%	71.38 ± 20.42	**96.05 ± 01.59**	95.73 ± 03.74	94.53 ± 01.17
	C%	11.69 ± 09.07	01.59 ± 01.33	**00.13 ± 00.20**	03.24 ± 00.85
	O%	14.67 ± 17.20	**00.00 ± 00.00**	01.90 ± 03.31	01.62 ± 00.38
	I%	02.26 ± 00.40	02.36 ± 00.33	02.23 ± 00.40	**00.61 ± 00.18**
NTN	M%	95.84 ± 03.04	**98.89 ± 02.14**	97.78 ± 05.05	97.41 ± 00.15
	C%	01.61 ± 01.51	**00.00 ± 00.00**	00.05 ± 00.07	**00.00 ± 00.00**
	O%	00.06 ± 00.18	01.11 ± 02.14	02.17 ± 05.00	**00.00 ± 00.00**
	I%	02.49 ± 02.15	**00.00 ± 00.00**	00.01 ± 00.01	02.59 ± 00.15
HDisease	M%	77.18 ± 02.47	**90.33 ± 00.98**	88.75 ± 01.09	88.08 ± 01.09
	C%	00.07 ± 00.14	02.80 ± 01.10	00.04 ± 00.10	**00.00 ± 00.00**
	O%	22.18 ± 02.58	**03.25 ± 00.14**	03.64 ± 01.30	07.69 ± 00.90
	I%	**00.00 ± 00.01**	03.62 ± 01.15	07.57 ± 01.42	04.23 ± 00.16
Financial	M%	75.27 ± 10.05	93.42 ± 00.87	**93.52 ± 01.02**	87.40 ± 04.74
	C%	13.63 ± 05.18	00.32 ± 00.23	**00.22 ± 00.21**	06.33 ± 04.33
	O%	04.03 ± 10.76	**00.00 ± 00.00**	**00.00 ± 00.00**	00.00 ± 00.01
	I%	07.06 ± 01.80	06.26 ± 00.75	06.26 ± 00.88	**06.26 ± 00.52**
Geoskills	M%	72.16 ± 15.36	**84.15 ± 03.15**	82.60 ± 04.69	50.20 ± 02.31
	C%	17.47 ± 14.24	02.00 ± 01.35	**00.00 ± 00.00**	23.66 ± 02.61
	O%	07.86 ± 02.67	11.33 ± 02.46	13.33 ± 04.43	**01.34 ± 00.12**
	I%	**02.52 ± 02.32**	**02.52 ± 02.32**	04.07 ± 04.09	24.80 ± 00.89

The classifiers induced by Boost(wDLF) showed more stability (a limited standard deviation) than those learned by DL-Foil and also by wDLF. The latter showed a worse performance in the experiments with all the KBs but NTN. In particular, it proved to be very sensitive to the distribution of the training data and, like Celoe, tended to induce simple descriptions that ended up misclassifying many test individuals. While commission was mainly due to induced concepts not overlapping with the targets, thus leading to misclassify positives, the omission cases were due to the simplicity of the induced concepts that made it difficult to predict a definite membership. Consequently, the match rate was lower w.r.t. the experiments with DL-Foil. Boost(wDLF) showed that the results could be improved both over all of the learning problems and also over the various runs of the boostrap procedure, i.e. different choices of training and test sets. Also, with the ensemble models, omission cases were totally avoided in the experiments with BioPax, or limited (with NTN, HDisease).

This was likely due to weak classifiers based on concepts that favored the prediction of a definite membership representing the majority class in the training set. Due to this bias, a limited increase of the commission rate w.r.t. DL-Foil and Celoe was also observed in the experiments. In the experiments with NTN,

HDisease and GeoSkills a lower induction rate was observed with ensemble classifiers compared to the rate observed with other classifiers.

While the larger induction rate marked in the experiments with DL-Foil were due to the approximation of target concepts with a large number of uncertain-membership individuals in the training/test sets, Boost(wDLF)'s strong models favored the prediction of an uncertain membership. This resembles the *vanishing gradient*, a phenomenon that affects neural networks training: this basically implies that weak models are initially induced with a considerable decrease of the weights for the correctly covered examples, hence the subsequent models have a negligible effect on these weights. As a result, the ensemble model quickly converges towards a solution that is sufficiently correct w.r.t. the available instances.

While it may represent a downside for the predictiveness of the classifiers, this aspect can enhance the readability of the ensemble models compared to the lengthy concept descriptions sometimes produced by DL-Foil to fit the data.

Biopax

- Boost(wDLF): {utilityClass [0.82], entity [14E-5]}
- DL-Foil: (catalysis ⊔ pathwayStep) ⊓ ¬openControlledVocabulary ⊓ ¬protein ⊓ ∀STRUCTURE.sequenceInterval
- Celoe: ¬sequence

NTN

- Boost(wDLF): { Man [3.41], Man ⊓ ∀ knows.⊤ [3.63], Man ⊓ ∀ ethnicityOf.⊤ [3.63], Man ⊓ ∀ parentOf.⊤ [3.63], Man ⊓ ∀ hasEnemy.⊤ [3.63] }
- DL-Foil: Man ⊓ ((CitizenshipAttribute ⊓ ∃locationOf.⊤) ⊔ ∀spouseOf.Woman)
- Celoe: Man

HDisease

- Boost(wDLF): { Lab_Tests [2.72], ¬Urine_specific_gravity_Class [0.99], Abnormal_findings_on_examination_of_other_body_fluids_substances_and_tissues_without_diagnosis_Class [3.35], Abnormal_findings_in_specimens_from_respiratory_organs_and_thorax_Abnormal_immunological_findings_Class [3.13]}
- DL-Foil: Abnorma_findings_on_examination_of_other_body_fluids_substances_and_tissues_without_diagnosis_Class ⊔ (Acute_abdomen_Class ⊓ ¬Toxoplasma_oculopathy) ⊓ ¬Pseudomonas_aeruginosa_mallei_pseudomallei_as_the_cause_of_diseases_classified_to_other_chapters_Class))
- Celoe: NOT_CIE_Disease

Fig. 1. Some ensembles produced in the experiments

6.3 Examples of Induced Ensembles

For a simple qualitative evaluation, Fig. 1 reports some ensembles produced in the experiments: concepts (to be interpreted disjunctively) together with their confidence. Note that in these ensembles, Boost(wDFL) was only able to induce classifiers made up of concepts names. Extremely short descriptions (such as those produced by Celoe or single runs of wDFL) often yielded larger commission error rates as they risk to cover negative examples. A final remark concerns the confidence values: they may exceed 1 (e.g. see the cases of NTN and Hdisease) also as a consequence of the compaction of the ensemble models.

7 Conclusions and Outlook

We presented a methodology for boosting concept learners trading the comprehensibility of the classification model for accurate ensemble classifiers. In particular, a constrained confidence boosting schema was have applied taking an adapted version of the inner specializing routine in DL-Foil as a weak learner. The underlying heuristic was modified to produce candidate concepts focused on covering hard examples. We have compared the boosted version of the algorithm with the weak learner, the original DL-Foil and Celoe from the DL-Learner suite. The experiments showed that Boost(wDLF) is able to better approximate target concepts, especially those with particularly complex definitions (e.g. featuring many nested sub-concepts).

We plan to investigate the implications of the method on efficiency and scalability. In particular, we intend to integrate a strategy for the *dynamic selection* of the ensemble size, even adopting a validation set to estimate an optimal value to ensure a good predictiveness of the strong classifier.

Also, we may apply the boosting strategy in combination to weak learners that adopt approximated inference mechanisms, like those built upon OCEL available in DL-Learner [2]. Another direction regards the investigation of different ensemble schemas and the exploitation of the inherent parallelism using frameworks for distributed data processing, such as Apache Spark.

References

1. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation and Applications, 2nd edn. Cambridge University Press, Cambridge (2007)
2. Bühmann, L., Lehmann, J., Westphal, P.: DL-Learner - a framework for inductive learning on the Semantic Web. J. Web Sem. **39**, 15–24 (2016)
3. Cohen, W.W., Singer, Y.: A simple, fast, and effective rule learner. In: Hendler, J., Subramanian, D. (eds.) AAAI 1999/IAAI 1999, pp. 335–342. AAAI/MIT Press, Menlo Park (1999)
4. De Raedt, L.: Logical and Relational Learning. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-68856-3
5. Fanizzi, N.: Concept induction in Description Logics using information-theoretic heuristics. Int. J. Semantic Web Inf. Syst. **7**(2), 23–44 (2011)
6. Fanizzi, N., Rizzo, G., d'Amato, C., Esposito, F.: DLFoil: class expression learning revisited. In: Faron Zucker, C., Ghidini, C., Napoli, A., Toussaint, Y. (eds.) EKAW 2018. LNCS (LNAI), vol. 11313, pp. 98–113. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03667-6_7
7. Heath, T., Bizer, C.: Linked Data: Evolving the Web into a Global Data Space. Morgan & Claypool, San Rafael (2011)
8. Hoche, S., Wrobel, S.: Relational learning using constrained confidence-rated boosting. In: Rouveirol, C., Sebag, M. (eds.) ILP 2001. LNCS (LNAI), vol. 2157, pp. 51–64. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44797-0_5
9. Iannone, L., Palmisano, I., Fanizzi, N.: An algorithm based on counterfactuals for concept learning in the Semantic Web. Appl. Intell. **26**(2), 139–159 (2007)

10. Lehmann, J., Auer, S., Bühmann, L., Tramp, S.: Class expression learning for ontology engineering. J. Web Sem. **9**, 71–81 (2011)
11. Melo, A., Völker, J., Paulheim, H.: Type prediction in noisy RDF knowledge bases using hierarchical multilabel classification with graph and latent features. Int. J. Artif. Intell. Tools **26**(2), 1–32 (2017)
12. Quinlan, J.R.: Boosting first-order learning. In: Arikawa, S., Sharma, A.K. (eds.) ALT 1996. LNCS, vol. 1160, pp. 143–155. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61863-5_42
13. Rizzo, G., d'Amato, C., Fanizzi, N., Esposito, F.: Tree-based models for inductive classification on the web of data. J. Web Sem. **45**, 1–22 (2017)
14. Rizzo, G., Fanizzi, N., d'Amato, C., Esposito, F.: Approximate classification with web ontologies through evidential terminological trees and forests. Int. J. Approx. Reason. **92**, 340–362 (2018)
15. Rowe, M., Stankovic, M., Alani, H.: Who will follow whom? Exploiting semantics for link prediction in attention-information networks. In: Cudré-Mauroux, P., et al. (eds.) ISWC 2012. LNCS, vol. 7649, pp. 476–491. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35176-1_30
16. Schapire, R.E., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. Mach. Learn. **37**(3), 297–336 (1999)
17. Tran, A.C., Dietrich, J., Guesgen, H.W., Marsland, S.: Parallel symmetric class expression learning. J. Mach. Learn. Res. **18**, 64:1–64:34 (2017)
18. Tran, T., Ha, Q., Hoang, T., Nguyen, L.A., Nguyen, H.S.: Bisimulation-based concept learning in description logics. Fundam. Inform. **133**(2–3), 287–303 (2014)