JMLR: Workshop and Conference Proceedings 45:349–361, 2015 ACML 2015

Statistical Unfolded Logic Learning

Wang-Zhou Dai DAIWZ@LAMDA.NJU.EDU.CN
Zhi-Hua Zhou ZHOUZH@LAMDA.NJU.EDU.CN
National Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing 210046, China

Editor: Geoffrey Holmes and Tie-Yan Liu

Abstract

During the past decade, Statistical Relational Learning (SRL) and Probabilistic Inductive Logic Programming (PILP), owing to their strength in capturing structure information, have attracted much attention for learning relational models such as weighted logic rules. Typically, a generative model is assumed for the structured joint distribution, and the learning process is accomplished in an enormous relational space. In this paper, we propose a new framework, i.e., Statistical Unfolded Logic (SUL) learning. In contrast to learning rules in the relational space directly, SUL propositionalizes the structure information into an attribute-value data set, and thus, statistical discriminative learning which is much more efficient than generative relational learning can be executed. In addition to achieving better generalization performance, SUL is able to conduct predicate invention that is hard to be realized by traditional SRL and PILP approaches. Experiments on real tasks show that our proposed approach is superior to state-of-the-art weighted rules learning approaches.

Keywords: Relational Learning, Weighted Logic Rules, Structure Learning

1. Introduction

In many tasks, it is often crucial to capture and exploit structure information of the data. Thus, the induction of probabilistic relational knowledge in the form of weighted logic rules has attracted much attention. The task can be described as: Given a relational dataset consisting of two sets of ground facts, i.e., examples and background knowledge, to generate a series of weighted logic rules consistent with examples in terms of evidence relations in background knowledge.

A natural idea is to learn a structured probabilistic model to represent the joint distribution of the domain, by Statistical Relational Learning (SRL) Getoor and Taskar (2007) or Probabilistic Inductive Logic Programming (PILP) De Raedt and Kersting (2008). This problem, however, is a combinatorial optimization problem that requires repeatedly enumerating rule structures and estimating corresponding parameters. The enormous combinations of relations and arguments make the task notoriously difficult. Thus, SRL and PILP systems have tried different ways for the optimization. By assuming the independence between ground facts, Kok and Domingos (2005) employed pseudo-likelihood to speedup parameter estimation; this approach has been facilitated with various search techniques for rule structure space reduction Mihalkova and Mooney (2007); Kok and Domingos (2009, 2010); Bellodi and Riguzzi (2012, 2013). One representative of this approach, RDN-Boost Natarajan et al. (2012), transforms the pseudo-likelihood estimation into a regression task solved

by functional gradient boosting, leading to the state-of-the-art performance. Another approach, e.g. ALEPH++-MLN Huynh and Mooney (2008), is to use first-order logic rule learners to produce a set of candidate rules, and then generate weighted logic rules based on the candidate set, though the optimal solution is not guaranteed to exist in the hypothesis space reduced by the first-order logic rule learners.

In this paper, we propose the Statistical Unfolded Logic (SUL) learning framework. We search for a relational feature set $\mathcal{F}$ to extract an attribute-value dataset from relational space, such that existing statistical learning approaches can be applied, and then translate the learned statistical model back into weighted logic rules. This framework is realized by the SUL-Path approach. There is no independence assumption about ground facts during the process of attribute-value data extraction and weighted logic rule translation. Moreover, our approach searches for $\mathcal{F}$ based on relational pathfinding Richards and Mooney (1992), and this feature space is guaranteed to be complete for learning definitive weighted logic rules. Furthermore, the efficiency of our approach enables predicate invention that is difficult to be realized by traditional SRL and PILP approaches. Experimental results show that our approach is superior to state-of-the-art rule learners in performance.

The rest of this paper is organized as follows. Section 2 presents some preliminaries. Section 3 proposes the SUL framework. Section 4 develops the SUL-Path approach, followed by experimental results in Section 5. Finally, Section 6 concludes.

2. Preliminaries

In this paper, relations are defined with First-order logic (FOL), a first-order alphabet contains constants, variables, functions and predicates. Constants represent objects in domain, e.g. `1`, `anna`. Variables range over the constants, e.g. `X`, `Person`. Functions represent mappings from tuples of objects to objects, e.g. `s(X)` is identical to $\texttt{X} + 1$. Predicates represent relations among objects or attributes of objects, e.g. `friends/2` means "friends" relation is applied on 2 objects, the number behind the slash is called *arity*. A *term* is a constant, a variable or a function symbol immediately followed by a bracketed n-tuple of terms, e.g. `bob`, `X` and `s(s(s(0)))`. An *atom* is a predicate symbol applied to a tuple of terms, e.g. `greater_than(X,s(s(0)))`. A *ground term* is a term containing no variables. A *ground atom* is an atom whose arguments are ground terms. A *definite clause* is a formula of the form "$\texttt{A:-B}_1\texttt{,...,B}_n$", where `A` and $\texttt{B}_i$ represents the atoms that form the rule's *head* and *body* respectively, the variables appear in the head must appears in the body. A *possible world* or *Herbrand interpretation* assigns a truth value to each possible ground atom in domain. A clause is *satisfiable* only if there is an assignment of truth values to ground atoms that make the clause true. An atom A is *provable* from an hypothesis H only if there exists a proof from H to A, which is denoted as $H \vdash A$.

The joint distribution of a relational domain is defined on possible worlds of the domain. A probabilistic atom is written as $p\texttt{::A}$ representing the probability of `A` being true is p. The joint distribution is structured by weighted logic rules, which are parameterized definitive clauses written as "$w\texttt{::A:-B}_1\texttt{,...,B}_n$", where w is called the weight of rule.

3. The SUL Framework

Formally, our task is to learn a set of weighted logic rules $\mathcal{R}$ to define a target concept `t/n`. The training data consists of two sets of ground atoms as background knowledge $\mathcal{B}$ and training examples $\mathcal{E}$. Positive and negative examples are denoted as $\mathcal{E}^+$ and $\mathcal{E}^-$ respectively.

Suppose the ground truth rule set is $\mathcal{R}^*$, the ultimate target of weighted rule learning is to recover $\mathcal{R}^*$ or at least reconstruct the equivalence class of $\mathcal{R}^*$ to model $P(\mathbf{T}|\mathcal{R},\mathcal{B})$, where $\mathbf{T} = (t_1, \ldots, t_n)$ stands for all possible ground atoms of the target concept `t/n`.

Ideally, the search for weighted logic rules should be directly guided by the goal of maximizing their contribution to the predictive accuracy. It is a well-known fact that statistical learning is very efficient for such kind of tasks. Thus we wish to formulate the weighted rules learning task as a statistical learning problem to exploit the benefits. A straightforward idea to accomplish this target is extracting an attribute-value dataset $\mathcal{D}$ from the original relational dataset, then existing statistical learning method can be directly executed. The training examples of $\mathcal{D}$ should be identical to the original $\mathcal{E}$, but the feature space $\mathcal{F}$ is difficult to be determined. Observing that structure of a rule is of the form "`A:-B`$_1$`,...,B`$_n$", naturally, we can use first-order logic relations `B`$_i$ as features.

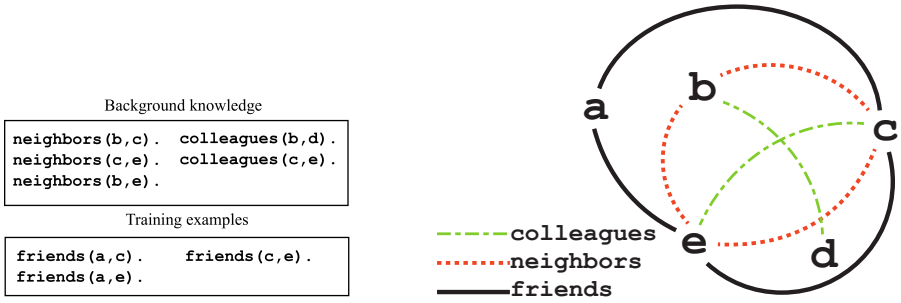
Thus, if we limit the size of $\mathcal{F}$, then we can instantly narrow down the hypothesis space for learning rule structures. Some of traditional Inductive Logic Programming algorithms use similar techniques to learn first-order rules Fürnkranz et al. (2012), however, they are impractical for SRL problem. The reduction of hypothesis may result in the risk of losing optimal solution $\mathcal{R}^*$ and its equivalence class. Lets define $MB(\mathbf{T})$ as the Markov Blanket of $\mathbf{T}$, then $P(\mathbf{T}|MB(T)) = P(\mathbf{T}|\mathcal{R},\mathcal{B})$ Koller and Friedman (2009). Hence, once a relational feature set $\mathcal{F}$ that satisfies $\mathcal{F} \supset MB(T)$ is obtained, it is guaranteed that the extracted dataset $\mathcal{D}$ is able to produce the optimal solution.

Unfortunately, we can never know the structure of $\mathcal{R}^*$ in advance, and thus it might be very difficult to find an optimal Markov Blanket $MB(\mathbf{T})$. To solve this problem, we can find a way to structure the space of possible features, then search for $\mathcal{F}$ according to prior domain knowledge.

Here we formally propose the *Statistical Unfolded Logic* learning (SUL) framework, it is divided into 3 stages:

1. **Unfolding.** SUL first searches for a set of relational features $\mathcal{F}$ that may be useful for learning the target concept `t/n` according to particular prior knowledge. In relational domain, the quantity of positive examples is usually much lesser than that of false examples. To ensure that $\mathcal{F}$ contains enough knowledge for defining `t/n`, $\mathcal{F}$ should at least be *complete* for the input positive examples $\mathcal{E}^+$, which means $\forall e^+ \in \mathcal{E}^+$, $\exists \mathtt{F} \in \mathcal{F}$ s.t. $\mathtt{F}, \mathcal{B} \vdash e^+$. Notice that, if the target relational domain holds the closed world assumption, i.e. the non-existent facts are assumed to be false, SUL can extract the ground atoms $\{e | e \notin \mathcal{E},\ \exists \mathtt{F} \in \mathcal{F}$ s.t. $\mathtt{F}, \mathcal{B} \vdash e\}$ as additional negative examples to enhance the original $\mathcal{E}$.

2. **Learning.** In this stage, $\mathcal{F}$ serves as a feature space to generate feature vectors. For each example $e \in \mathcal{E}$, SUL queries a Prolog engine for e's satisfaction situation of $\forall \mathtt{F} \in \mathcal{F}$. The satisfaction results are recorded as attribute values of feature vectors.



(*a*) Relational dataset. (*b*) Hypergraph of the data.

Figure 1: Example of `friends` dataset (symmetric atoms omitted).

After the dataset construction, SUL uses statistical learning to learn a statistical model.

3. **Folding.** In order to interpret the learned statistical model in the original relational domain, SUL finally translates the learned statistical model back into weighted logic rules. After the folding step, the weights of rules are the scores given by the statistical discriminative models. For example for decision trees, the weights of each rule is the purity of the leaf that standing for the rule.

Seeing this problem from the perspective of first-order rule learning, the relational features $\mathcal{F}$ act as definitions of candidate invented predicates. Owing to the efficiency of statistical learning, SUL can quickly identify which predicates are truly beneficial for describing the hidden structure of the target domain. As a consequence, SUL naturally supports *predicate invention*.

4. The SUL-Path approach

Apparently, the choice of relational features $\mathcal{F}$ is crucial to the performance of SUL algorithms. In this section, we propose the SUL-Path approach based on the "relational path" assumption Richards and Mooney (1992) as an implementation of the SUL framework. The outline of SUL-Path algorithm is presented in algorithm 1, we describe the details of SUL-Path in the following sections.

4.1. Rule Learning

In order to describe the proposed approach more clearly, we first define a hypergraph of a relational domain as follow:

Definition 1 (Relational hypergraph): *Given a relational dataset, its hypergraph is* $\mathcal{G} = (V, L)$*, where each vertex* $v \in V$ *is a constant in domain, each hyperedge* $l \in L$ *is a ground atom in domain.*

An example of relational hypergraph generation is illustrated with Figure 1.

The relational path assumption suggests that there usually exist fixed-length paths of relations linking the set of terms that satisfy a target concept Richards and Mooney (1992). According to this assumption, we define the first kind of SUL-Path feature as follow:

Algorithm 1 *SUL-Path($\mathcal{B}, \mathcal{E}$)*

Input: Background knowledge $\mathcal{B}$, Examples $\mathcal{E} = \mathcal{E}^+ \cup \mathcal{E}^-$
Output: weighted rule set $\mathcal{R}$
Start:
Initialize hypergraph $\mathcal{G}$ with $\mathcal{B}$.
$\mathcal{F} = \phi$
for positive example $e^+ \in \mathcal{E}^+$ **do**
  Search for ground paths of e^+ in $\mathcal{G}$
  Generalize the ground paths to *path features* $\mathcal{P}$
  Search for *local paths* and *branch features* $\mathcal{P}'$ of $\mathtt{p}_i \in \mathcal{P}$
  add $\{\mathcal{P} \cup \mathcal{P}'\}$ to $\mathcal{F}$
end for
Initialize an $|\mathcal{F}|$-dimensional attribute-value dataset $D = \phi$
for each example $e_k \in \mathcal{E}$ **do**
  Count its satisfaction for $\mathcal{F}$ as the feature vector f_k
  Label f_k according to $\mathcal{E}$ and add it to the attribute-value dataset D
end for
Use Weka to learn a model h for D
Translate h into weighted rules $\mathcal{R}$
Return: $\mathcal{R}$.

Definition 2 (Path feature): *Given a hypergraph $\mathcal{G} = (V, L)$ and an example of target concept* `t(`v_1`,...,`v_n`)`*, a* ground path *for the example is a set of sequential hyperedges $(l_1, \ldots, l_k)$ that links $\{v_1,\ldots,v_n\}$. A* path feature *for example* `t(`v_1`,...,`v_2`)` *is a generalization of the ground path linking $\{v_1,\ldots,v_n\}$, i.e. substitute the constants of the ground path with unique variables.*

For the example illustrated with Figure 1, consider the positive example `friends(c,e)`, a ground path of it is `colleagues(c,e)`, which can be generalized to a path feature `colleagues(X,Y)`.

Based on path features, we define the second kind of SUL-Path feature to enhance the feature space as follow:

Definition 3 (Local path feature): *Given a path (feature) $L = (l_1, \ldots, l_k)$, the vertices that p_l has crossed are $V_l = v_1, \ldots, v_m$, a* local path feature *is a path feature that connects any subset of V_l.*

For example, the local paths are very common in the Cora dataset. The background knowledge are binary predicates describing authors, titles and venues of published papers, the target concepts are binary predicates to duplicate those fields of a bibliography. A path feature of one of the target concept `sameAuthor/2` could be:

`p(`X_1`,`X_2`,`X_3`,`X_4`,`X_5`):-author(`X_3`,`X_1`),title(`X_3`,`X_4`),title(`X_5`,`X_4`),author(`X_5`,`X_2`).`

then "`venue(`X_3`,`X_6`),venue(`X_5`,`X_6`)`" forms a local path that connects the vertices X_3 and X_5 on the path feature. Intuitively, local paths contain additional information about a path feature, they may be helpful to describe the hidden relational structure of the domain as well.

Some structural informations about attributes of objects can not form paths. For the `friends` example, if we know some single-arity facts like `male(X)` or `female(X)` in background knowledge, they might be also useful for learning the target concept. Thus, we define the last kind of SUL-Path feature:

Definition 4 (Branch feature): *Given a set of vertices V, a* branch feature *is a path L, which has crossed a set of vertices V_L , s.t. $|V \cap V_L| = 1$, and the only intersected vertex is an end vertex of L.*

This kind of feature is called "branch feature" because on path features, they look like branches that spread out but never come back.

As to the structure of a weighted logic rule, it is easy to see that:

Proposition 1 *Structure of any definitive logic rule that defines the target concept `t/n` can be represented by a set of branch features plus a set of connected hypergraphs (the sets could be empty).*

This proposition can be proved directly from the definition of definitive clauses.

We can show that, with the three kinds of features defined above, SUL-Path is able to learn all forms of definitive logic rule structure.

Theorem 1 (Completeness of SUL-Path features): *SUL-Path is able to learn any structure of rules with path feature, local path feature and branch feature.*

Proof From proposition 1 we can learn that any form of definitive rule can be represented by a set of branch features and connected hypergraphs. Apparently, SUL-Path is able to learn branch features with branch feature, therefore we only have to prove that the 3 types of relational features are able to express any connected hypergraph.

Without loss of generality, we assume that a connected sub-hypergraph $\mathcal{G}'$ for a rule contains 2 arguments from its head as its end vertices, then we can find at least 1 path feature `p` on this sub-hypergraph. For each vertex v on the path feature, we can traverse start from v and stop at any end vertices v' on $\mathcal{G}'$. If v' is on `p`, then the trajectory vv' is a local path feature, otherwise vv' is either a branch feature started from v or a path that composed by local path features and a path feature. ■

Although the 3 kinds of relational features are complete for weighted rules learning, the size of $\mathcal{F}$ grows exponentially with the maximum length of the different kind of paths. According to Richards and Mooney (1992), SUL-Path assumes that the optimal hypothesis is restricted in a feature space formed by short paths. Our results in the experiments proved this assumption empirically.

In summary, at the *unfolding* stage, SUL-Path builds a hypergraph and searches for a set of features of positive examples as $\mathcal{F}$. During the *learning* stage, for each $e \in \mathcal{E}$, SUL-Path counts the number of satisfaction of $\forall \mathtt{F} \in \mathcal{F}$ to construct their feature vectors. When an attribute-value dataset is ready, SUL-Path learns decision tree models by Weka-3.6 Hall et al. (2009). In the *unfolding* stage, SUL-Path translates the tree models back into weighted rules like RPT Neville et al. (2003).

4.2. Predicate Invention

Candidate predicates generation of SUL-Path is accomplished by the relational feature construction in its unfolding stage. After the learning stage, useful predicates from $\mathcal{F}$ are naturally selected by the learned decision trees.

Both feature construction and predicate invention can be considered to be *constructive induction* Kramer (1995). Generally speaking, the goal of constructive induction is an increase in accuracy and a decrease in complexity of a hypothesis Wnek and Michalski (1994). Different from learning in an existed feature space, constructive induction requires more domain knowledge. For SUL-Path, the short relational path assumption acts as such kind of domain knowledge.

Normally, the number of candidate predicates could be potentially very large because the predicate invention problem is also a combinatorial optimization problem in relational space. Even for SUL-Path, the short relational path assumption results in a candidate set that exponentially grows with the maximum path length. Thus, preserving all of the candidates is apparently unrealistic. Owing to the efficiency of decision tree learning, SUL-Path is able to keep the truly functional predicates in a feasible time.

5. Experiments

We now present the experimental results of SUL-Path. The experiments are conducted on 5 target concepts from 2 benchmark real world datasets.

5.1. Datasets

Cora. This dataset is a collection of citations to computer science papers, processed into 5 folds for the task of duplicating the citations Poon and Domingos (2007). This domain has 5 types of 3079 constants, 10 predicates and 687422 atoms. Evidence predicates are relations like `author(Bib,Author)`,`title(Bib,Title)`, `venue(Bib,Venue)`, and so on. Target concepts to be learned are `sameAuthor/2`, `sameBib/2`, `sameTitle/2` and `sameVenue/2` that duplicates the corresponding 4 types of constants. For simplicity, we denote the tasks as *author*, *bib*, *venue* and *title* in the following section.

UW-CSE. This dataset was prepared by Domingos and Lowd (2009), describing relationships in an academic department. It is divided into 5 independent folds. This domain has 9 types of 929 constants, 12 of predicates and 260254 atoms. The evidence predicates define students, faculty, and describe their relationships. The target concept `advisedBy(Person, Person)` judges whether a person is advised by another person, this task is referred as *advisedby* in the following part of this section. We omitted 9 equality predicates following Natarajan et al. (2012).

Among the two datasets, UW-CSE is smaller but it has a more complex structure, Cora has more ground atoms but has a simpler structure. Both the datasets satisfy closed world assumption, but negative examples are incomplete, specifically, UW-CSE does not contain any negative example.

5.2. Compared Methods

We compared 4 algorithms on all the tasks:

RDN-Boost Natarajan et al. (2012). This algorithm learns transforms the weighted rules learning problem into regressions that estimates pseudo-likelihoods. We used the parameters in Natarajan et al. (2012) for our experiments. RDN-Boost also requires a predefined "mode" as background knowledge to reduce the search space of rule structure. In the experiments we enumerated all possible schemes and achieved the best performance (on some tasks our results are even better than the result reported in Natarajan et al. (2012)). It also needs a set of negative examples, since the negative examples provided by Cora and UW-CSE are incomplete, we ran RDN-Boost with both the original and a complete negative training set, the two methods are denoted as *RDN-Boost-org* and *RDN-Boost-total* respectively.

Alchemy Domingos and Lowd (2009). This is a widely used Statistical Relational Learning toolbox. In the experiments, we are using Alchemy 2.0 version. We ran Alchemy's `learnstruct` and `learnweight` programs with its default settings.

ALEPH++-MLN Huynh and Mooney (2008). This is an algorithm that learns Markov Logic Networks (MLN) discriminatively. It uses ALEPH Srinivasan, an Inductive Logic Programming algorithm, to discriminatively learn first-order logic rules as candidate MLN structures. Then it uses exact inferences with L1 regularization to estimate the parameters and select the structures. The ALEPH program also needs a predefined mode setting, we used the same modes as to RDN-Boost, the parameter setting is same as Huynh and Mooney (2008) suggests.

SUL-Path. This is the algorithm proposed by this work. Because the two datasets satisfy the closed world assumption, SUL-Path can unfold additional negative examples by itself. The maximum lengths of the path features is set at 4, 4, 4, 4 and 2 for the 5 tasks respectively. We tested SUL-Path with different statistical learners, including J48, AdaboostM1 with 50 3-layers REPTrees as base learner and Random Forest with 300 4-layers random trees. They are denoted as *SUL-Path-J48*, *SUL-Path-Boost* and *SUL-Path-RF* in the following section. Because the number of positive examples is far more less than the number of negative examples, we use Weka's `CostsensitiveClassifier` to train all the models above. The cost of a false negative is set as twice of the cost of a false positive. Parameters of Weka algorithms are chosen by 5-fold cross validation on training data.

5.3. Performance Comparison

To evaluate the models learned by the algorithms under the global distribution, our test data contain all possible ground atoms of the target concepts. For the five tasks: *author*, *bib*, *title*, *venue*, *advisedby*, our test data in each fold have 952, 67144, 2287, 8587 and 3343 examples respectively on average. Because in relational domains, the number of positive examples is far more less than the number of negative examples, following Kok and Domingos (2010) and Natarajan et al. (2012), we used AUC value instead of precision and F-measure as evaluation criteria to reduce the class imbalance problem.

Table 1 presents the average AUC value of all algorithms on each task. Because UW-CSE dataset does not contain negative training example, for RDN-Boost methods only RDN-Boost-total was applied. In *bib* and *venue* task, the Alchemy algorithm did not finish in 100 hours. The ILP procedure ALEPH Srinivasan in ALEPH++-MLN crashes on the 4 Cora tasks due to out of memory of Prolog, so no results were obtained.

Table 1: Average AUC values of each approach on 5 tasks.

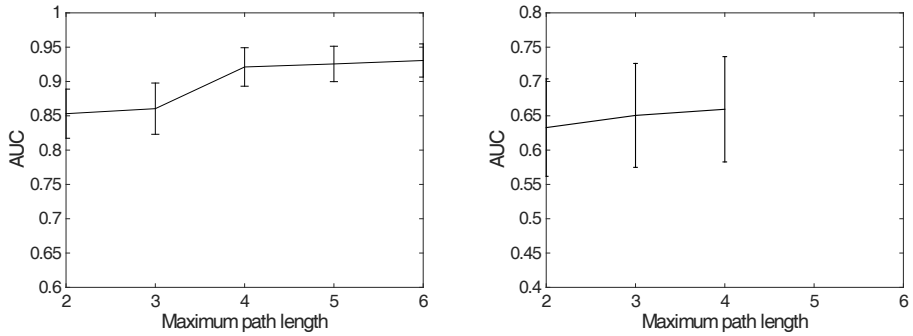
AUC	*author*	*bib*	*title*	*venue*	*advisedby*
SUL-Path-J48	0.994 ± 0.004	0.926 ± 0.025	0.921 ± 0.031	0.839 ± 0.046	0.633 ± 0.071
SUL-Path-Boost	$\mathbf{0.998 \pm 0.002}$	$\mathbf{0.998 \pm 0.001}$	$\mathbf{0.987 \pm 0.011}$	$\mathbf{0.977 \pm 0.011}$	0.975 ± 0.015
SUL-Path-RF	$\mathbf{0.998 \pm 0.002}$	0.989 ± 0.009	0.982 ± 0.017	0.946 ± 0.023	$\mathbf{0.992 \pm 0.006}$
RDN-Boost-org	0.985 ± 0.014	0.916 ± 0.021	0.706 ± 0.121	0.589 ± 0.040	-
RDN-Boost-total	0.986 ± 0.012	0.949 ± 0.016	0.729 ± 0.126	0.590 ± 0.038	0.983 ± 0.014
Alchemy	0.597 ± 0.152	-	0.604 ± 0.216	-	0.393 ± 0.103
ALEPH++-MLN	-	-	-	-	0.127 ± 0.032

On average, SUL-Path-Boost performed best in 4 tasks. In both *author* and *advisedby* task, there was no significant difference between SUL-Path and RDN-Boost according to t-test; in other tasks, SUL-Path outperforms other compared methods significantly. ALEPH++-MLN performed worst in all 5 tasks because the initial rules obtained by ALEPH are learned in a non-probabilistic setting, its performance in our experiments is consistent with Kok and Domingos (2005).

The reason why SUL-Path-J48 performed worse than other SUL-Path algorithms is very simple. After examined the datasets, we found that the positive examples are usually covered by separate rules, especially in the *advisedby* task. When other methods are learning multiple models, SUL-Path-J48 only kept one model and ignored other potentially informative rules.

Training times of all algorithms are shown in Table 2. All the experiments were carried on a cluster node with a 2.53 GHz $\times$ 12 cores CPU. We can see that the discriminative learning methods, including SUL-Path and ALEPH++-MLN, are much more efficient than generative learning methods. ALEPH++-MLN is the quickest one because the ILP procedure ALEPH only learned 4-8 non-recursive candidate rules each fold for MLN weight estimation. Comparing to the training time, the unfolding stage of SUL-Path is time consuming. The unfolding times (including relational feature searching and attribute value calculation) and the unfolded data size of SUL-Path are shown in Table 3. The unfolding time on several tasks were worse than RDN-Boost, but they could be improved if the implementation of pathfinding is optimized.

We also studied the influence of the maximum path length on SUL-Path's performance. Because the performance of SUL-Path-Boost and SUL-Path-RF are close to optimal, we ran the experiments with SUL-Path-J48, the results are displayed with Figure 2. Because size of feature set and unfolding time grow exponentially with the maximum path length, the *unfolding* of *advisedby* task did not stop after one hour when the maximum path length exceeds 4. We can see that on both tasks the performance grows with the maximum path length, but the rate of growth decreases after a certain threshold is reached. This results showed that when the maximum path length is large enough, the best solution of SUL-Path-J48 is contained in the hypothesis space. The results also empirically verified that the short relational path assumption is correct on typical relational domains.



(*a*) *title* (*b*) *advisedby*

Figure 2: Influence of maximum path length on performance of SUL-Path-J48.

5.4. Predicate Invention

Output of SUL-Path is a Prolog file which can be directly consulted with an ISO Prolog engine, it contains both invented predicates and weighted rules. For example in the *author* task, one weighted rule in the output file of SUL-Path-Boost is:

```
tree_0_score(X1,X2,S0):-
        ct_path_0_0(X1,X2,N1),N1>0.0,S0 is 1.0,!.
```

The predicate `tree_i_score(X1,X2,S)` is a scoring predicate for `sameAuthor(X1,X2)`, where `S0=1.0` is the weight of this rule; predicate `ct_path_*` counts the satisfactions of feature `path_*`; `path_0_0` is an invented predicate, whose name indicates that this predicate is the 0-th "local path" of the 0-th feature `path_0`. The two invented predicates are:

```
path_0(X1,X2,X3,X4,X5):-
        author(X3,X1),title(X3,X4),title(X5,X4),author(X5,X2).
path_0_0(X1,X2,X3,X4,X5):-
        path_0(X1,X2,X3,X4,X5),
        haswordauthor(X1,X6),haswordauthor(X2,X6).
```

Meanings of the two rules are straightforward: `path_0`'s means the authors `X1` and `X2` wrote

Table 2: Average training time of each algorithm.

Time	*author*	*bib*	*title*	*venue*	*advisedby*
SUL-J48	0.2s	87.2s	0.5s	1.5s	0.4s
SUL-Boost	1.1s	302.6s	2.2s	9.5s	1.1s
SUL-RF	1.6s	471.0s	4.6s	19.7s	1.6s
RDN-Boost	25.6s	947.8s	278.6s	204.1s	15.6s
Alchemy	87.2h	-	98.3h	-	91.3h
ALEPH++-MLN	-	-	-	-	0.07s

Table 3: Average unfolding time and unfolded data size of each task

Time & size	Feature construction	Attribute value calculation	feature size	data size
author	5.7s	13.9s	36	2428
bib	47.0s	1475.5s	64	457782
title	20.6s	80.8s	34	16396
venue	55.4s	491.6s	33	73354
advisedby	11.5s	22.7s	452	10625

paper `X3` and `X5` respectively, and the two papers have a same title `X4`. Predicate `path_0_0` says the two authors from `path_0` also have a same word in their name.

6. Conclusions

In this work, we presented a novel approach for weighted logic rule learning. It searches for a relational feature set to extract an attribute-value dataset from relational space, such that existing statistical learning approaches can be executed, and then translate the learned statistical model back into weighted logic rules. In contrast to assuming independent ground facts or using first-order logic rule learners to reduce the hypothesis space, our approach does not assume independence of ground facts and has the guarantee that the optimal solution exists in the transformed hypothesis space. Furthermore, our approach can invent new predicates, which is difficult to be realized by traditional SRL approaches. Experiments exhibit the advantages of our approach. In future work it will be interesting to theoretically study how to ensure the optimal solution is contained in the reduced hypothesis space that formed by the relational features.

7. Acknowledgment

This research was supported by the National Key Basic Research Program of China (2014CB340501) and the National Science Foundation of China (61333014, 61321491).

References

Elena Bellodi and Fabrizio Riguzzi. Learning the structure of probabilistic logic programs. In *Proceedings of the 22nd International Conference on Inductive Logic Programming*, pages 61–75, Dubrovnik, Croatia, September 2012.

Elena Bellodi and Fabrizio Riguzzi. Structure learning of probabilistic logic programs by searching the clause space. *CoRR/arXiv:1309.2080*, 2013.

Luc De Raedt and Kristian Kersting. *Probabilistic Inductive Logic Programming*. Springer, New York, NY, 2008.

Pedro Domingos and Daniel Lowd. *Markov Logic: An Interface Layer for Artificial Intelligence*. Morgan & Claypool Publishers, San Rafael, CA, 2009.

Johannes Fürnkranz, Dragan Gamberger, and Nada Lavrac. *Foundations of Rule Learning*. Springer, New York, NY, 2012.

Lise Getoor and Ben Taskar. *Introduction to statistical relational learning*. MIT press, Cambridge, MA, 2007.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H Witten. The weka data mining software: an update. *ACM SIGKDD explorations newsletter*, 11(1):10–18, June 2009.

Tuyen N. Huynh and Raymond J. Mooney. Discriminative structure and parameter learning for markov logic networks. In *Proceedings of the 25th International Conference of Machine Learning*, pages 416–423, Helsinki, Finland, June 2008.

Stanley Kok and Pedro Domingos. Learning the structure of markov logic networks. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 441–448, Bonn, Germany, August 2005.

Stanley Kok and Pedro Domingos. Learning markov logic network structure via hypergraph lifting. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 217–224, Montreal, Canada, June 2009.

Stanley Kok and Pedro Domingos. Learning markov logic networks using structural motifs. In *Proceedings of the 27th International Conference on Machine Learning*, pages 551–558, Haifa, Israel, June 2010.

Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, Cambridge, MA, 2009.

Stefan Kramer. Predicate invention: A comprehensive view. Technical Report 95-32, Austrian Research Institute for Artificial Intelligence, Vienna, Austria, 1995.

Lilyana Mihalkova and Raymond J. Mooney. Bottom-up learning of markov logic network structure. In *Proceedings of the 24th International Conference on Machine Learning*, pages 625–632, Corvallis, OR, June 2007.

Sriraam Natarajan, Tushar Khot, Kristian Kersting, Bernd Gutmann, and Jude W. Shavlik. Gradient-based boosting for statistical relational learning: The relational dependency network case. *Machine Learning*, 86(1):25–56, January 2012.

Jennifer Neville, David Jensen, Lisa Friedland, and Michael Hay. Learning relational probability trees. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 625–630, Washington, DC, August 2003.

Hoifung Poon and Pedro Domingos. Joint inference in information extraction. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence*, pages 913–918, Vancouver, Canada, July 2007.

Bradley L. Richards and Raymond J. Mooney. Learning relations by pathfinding. In *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 50–55, San Jose, CA, July 1992.

Ashwin Srinivasan. ALEPH. http://www.cs.ox.ac.uk/activities/machlearn/Aleph/aleph.html.

Janusz Wnek and RyszardS. Michalski. Hypothesis-driven constructive induction in AQ17-HCI: A method and experiments. *Machine Learning*, 14(2):139–168, February 1994.