Probabilistic Inductive Constraint Logic

Fabrizio Riguzzi[1], Elena Bellodi[2], Riccardo Zese[2], Giuseppe Cota[2], and Evelina Lamma[2]

[1] Dipartimento di Matematica e Informatica – University of Ferrara
Via Saragat 1, I-44122, Ferrara, Italy
[2] Dipartimento di Ingegneria – University of Ferrara
Via Saragat 1, I-44122, Ferrara, Italy
`[fabrizio.riguzzi,elena.bellodi,riccardo.zese,`
`giuseppe.cota,evelina.lamma]@unife.it`

Abstract. Probabilistic logic models are used ever more often to deal with the uncertain relations that are typical of the real world. However, these models usually require expensive inference and learning procedures. Very recently the problem of identifying tractable languages has come to the fore. In this paper we consider the models used by the Inductive Constraint Logic (ICL) system, namely sets of integrity constraints, and propose a probabilistic version of them. A semantics in the style of the distribution semantics is adopted, where each integrity constraint is annotated with a probability. These probabilistic constraint logic models assign a probability of being positive to interpretations. This probability can be computed in linear time in the number of ground instantiations of the constraints. Parameter learning can be performed using an L-BFGS procedure. We also propose the system PASCAL for "ProbAbiliStic inductive ConstrAint Logic" that learns both the structure and the parameters of these models. PASCAL has been tested on a process mining dataset giving promising results.

1 Introduction

Probabilistic logic models are gaining popularity due to their successful application in a variety of fields, such as natural language processing, information extraction, bioinformatics, semantic web, robotics and computer vision.

However, these models usually require expensive inference and learning procedures. Very recently the problem of identifying tractable languages has come to the fore. Proposals such as Tractable Markov Logic [10], Tractable Probabilistic Knowledge Bases [18, 15] and fragments of probabilistic logics [4, 14] strive to achieve tractability by limiting the form of sentences.

In the ILP field, the learning from interpretation settings [9, 2] offers advantages in terms of tractability with respect to the learning from entailment setting. First, learning first-order clausal theories is tractable [9] in the sense that given fixed bounds on the maximal length of clauses and the maximal arity of literals, such theories are polynomial-sample polynomial-time PAC-learnable. Second, examples in learning from interpretations can be considered in isolation

[2], so coverage tests are local and learning algorithms take a time that is linear in the number of examples.

A particularly interesting system that learns from interpretations is Inductive Constraint Logic (ICL) [8]. It performs discriminative learning and it generates models in the form of sets of integrity constraints.

Our aim is to consider a probabilistic version of the sets of integrity constraints with a semantics in the style of the distribution semantics [17]. Each integrity constraint is annotated with a probability and a model assigns a probability of being positive to interpretations. This probability can be computed in linear time given the number of groundings of the constraints. Parameter learning can be performed using an L-BFGS procedure. We also propose the system PASCAL for ”ProbAbiliStic inductive ConstrAint Logic” that learns both the structure and the parameters of these models. PASCAL has been tested on a process mining dataset giving promising results.

The paper is organized as follows: Section 2 introduces integrity constraints and ICL, Section 3 presents probabilistic constraints, Section 4 illustrates PASCAL, Section 5 discusses related work, Section 6 describes the experiments performed and Section 7 concludes the paper.

2 Inductive Constraint Logic

ICL [8] performs discriminative learning from interpretations. It learns logical theories in the form of Constraint Logic Theories (CLTs).

A CLT T is a set of integrity constraints (ICs) C of the form

$$L_1, \dots, L_b \to A_1; \dots; A_h \tag{1}$$

where the L_is are logical literals and the A_j are logical atoms. $L_1, \dots, L_b$ is called the *body* of the IC and is indicated with $Body(C)$. $A_1; \dots; A_h$ is called the *head* and is indicated with $Head(C)$. The semicolon stands for disjunction, so the head is a disjunction of atoms and the body is a conjunction of literals.

Together with a CLT T, we may be given a background knowledge B on the domain which is a normal logic program.

CLTs can be used to classify Herbrand interpretations, i.e., sets of ground facts. Given a Herbrand interpretation I (simply interpretation in the following) and a background knowledge B, we use B to complete the information in I: instead of simply considering I, we consider $M(B \cup I)$ where M indicates the model according to the Prolog semantics (i.e. Clark completion [5]) and I is interpreted as a set of ground facts. In this way, all the facts of I are true in $M(B \cup I)$, moreover $M(B \cup I)$ can contain new facts derived from I using B.

Given an interpretation I, a background knowledge B and a CLT T we can ask whether T is true in I given B, i.e., whether T covers I given B or, in other words, whether I is a *positive interpretation*. Formally, an IC C is *true in an interpretation I given a background knowledge B*, written $M(B \cup I) \models C$, if, for every substitution θ for which $Body(C)$ is true in $M(B \cup I)$, there exists a disjunct in $Head(C)$ that is true in $M(B \cup I)$. If $M(B \cup I) \models C$ we say that

C *satisfies the interpretation* I *given* B; if $M(B \cup I) \not\models C$ we say that C does not satisfy I. A CLT T is *true in an interpretation* I *given* B if every IC of T is true in it and we write $M(B \cup I) \models T$. We also say T *satisfies* I *given* B, that T *covers* I *given* B or that I *is positive given* T *and* B.

In a *range-restricted* logical clause all the variables that appear in the head also appear in the body. [6] showed that the truth of a range-restricted IC in an interpretation I with range-restricted background knowledge B can be tested by asking the goal

$$? - Body(C), \neg Head(C).$$

against a Prolog database containing the atoms of I as facts together with the rules of the normal program B. By $\neg Head(C)$ we mean $\neg A_1, \ldots, \neg A_h$ so the query is

$$? - L_1, \ldots, L_b, \neg A_1, \ldots, \neg A_h. \tag{2}$$

If the query fails, C is true in I given B, otherwise C is false in I given B. If B is range-restricted, every answer to an atomic query Q against $B \cup I$ completely instantiates Q, i.e., it produces an element of $M(B \cup I)$. So the queries $\neg A_j$ are ground when they are called and no floundering occurs.

Example 1. The Bongard Problems were introduced by the Russian scientist M. Bongard in his book [3]. Each problem consists of a number of pictures, some positive and some negative. The goal is to discriminate between the two classes.

The pictures contain squares, triangles, circles, ... with different properties, such as small, large, pointing down, ... and different relationships between them, such as inside, above, ... Figure 1 shows some of these pictures.

Each picture can be described by an interpretation. Consider the left picture. It consists of a large triangle that includes a small square that, in turn, includes a small triangle. This picture can be described using the interpretation

$$\begin{aligned} I_l = \{&triangle(0), large(0), square(1), small(1), inside(1,0),\\ &triangle(2), inside(2,1)\} \end{aligned}$$

Moreover, suppose you are given the background knowledge B:

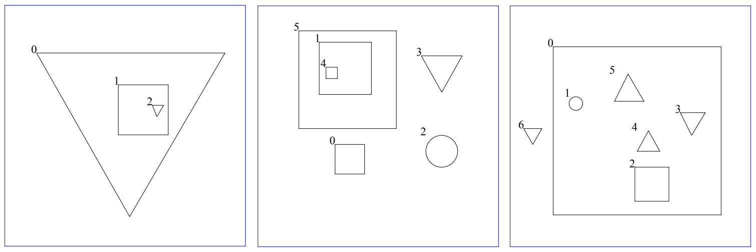

Fig. 1. Bongard pictures.

$$in(A,B) \leftarrow inside(A,B).$$
$$in(A,D) \leftarrow inside(A,C), in(C,D).$$

Thus $M(B \cup I_l)$ will contain the atoms $in(1,0)$, $in(2,1)$ and $in(2,0)$. The IC

$$C_1 = triangle(T), square(S), in(T,S) \rightarrow false$$

states that a figure satisfying the IC cannot contain a triangle inside a square. C_1 is false in I_l given B because triangle 2 is inside square 1.

In the central picture instead C is true given B because the only triangle is outside any square.

The learning from interpretations setting of ILP [9, 2] is the following.
Given

- a set $\mathcal{I}^+ = \{I_1, \ldots, I_Q\}$ of positive interpretations (positive examples)
- a set $\mathcal{I}^- = \{I_{Q+1}, \ldots, I_R\}$ of negative interpretations (negative examples)
- a normal logic program B (background knowledge)
- a hypothesis space $\mathcal{H}$

Find: an hypothesis $T \in \mathcal{H}$ such that

- for all $I^+ \in \mathcal{I}^+$, $M(B \cup I^+) \models T$
- for all $I^- \in \mathcal{I}^-$, $M(B \cup I^-) \not\models T$

Thus we look for a CLT that discriminates the positive interpretations from the negative interpretations.

ICL learning from interpretations uses a covering loop on the negative examples, an approach that is dual to the usual covering loop of top-down ILP algorithms. ICL starts from an empty theory and adds one IC at a time. After the addition of the IC, the set of negative examples that are ruled out by the IC are removed from $\mathcal{I}^-$ and the loop ends when no more ICs can be generated or when $\mathcal{I}^-$ becomes empty (all the negative examples are ruled out). ICL is shown in Algorithm 1.

Algorithm 1 Function ICL

```
1: function ICL($\mathcal{I}^+, \mathcal{I}^-, B, \mathcal{H}$)
2:     $T \leftarrow \emptyset$
3:     repeat
4:         $C \leftarrow$ FindBestIC($\mathcal{I}^+, \mathcal{I}^-, B, \mathcal{H}$)
5:         if $C \neq \emptyset$ then
6:             $T \leftarrow T \cup \{C\}$
7:             Remove from $\mathcal{I}^-$ all interpretations that are false for $C$
8:         end if
9:     until $C = \emptyset$ or $\mathcal{I}^-$ is empty
10: return $T$
11: end function
```

The IC to be added in every iteration of the covering loop is returned by the procedure FindBestIC. It uses a beam search with $P(\ominus|\overline{C})$ as a heuristic function, where $P(\ominus|\overline{C})$ is the probability that an input example is negative given that is ruled out by the IC C, i.e., the precision on negative examples. The search starts from the IC *true* $\rightarrow$ *false* that rules out all the negative examples but also all the positive examples and gradually refines that clause in order to make it more general. The maximum size of the beam is a user-defined parameter. The heuristic of each generated refinement is compared with the one of the best IC found so far and, if its value is higher, the best IC is updated. At the end of the refinement cycle, the best IC found so far is returned.

The refinement operator exploits θ-subsumption for defining a generality relation among ICs: an IC C θ-subsumes an IC D, written $C \leq_\theta D$, if there exists a substitution θ such that $C\theta \subseteq D$ where C and D are seen as logical clauses (sets of literals). The generality relation for ICs is defined in terms of θ-subsumption as for learning from entailment but in the opposite direction: an IC D is more general than an IC C ($D \leq_g C$) if $C \leq_\theta D$. So *true* $\rightarrow$ *false* is the most specific constraint and the search in FindBestIC proceeds bottom up.

Refinements are obtained by using a refinement operator that adds a literal to the body or head of the IC or applies a substitution.

3 Probabilistic Inductive Constraint Logic

A Probabilistic Constraint Logic Theory (PCLT) is a set of probabilistic integrity constraints (PICs) of the form

$$p_i \ :: \ L_1, \ldots, L_b \rightarrow A_1; \ldots; A_h \tag{3}$$

Each constraint C_i is associated with a probability $p_i \in [0,1]$ and a PCLT T is a set $\{(C_1, p_1), \ldots, (C_n, p_n)\}$. A PCLT T defines a probability distribution on ground constraint logic theories called worlds in this way: for each grounding of each IC, we include the IC in a world with probability p_i and we assume all groundings to be independent. The notion of world as a theory is similar to notion of world in ProbLog [7] where a world is a normal logic program. Let us assume that constraint C_i has n_i groundings called $C_{i1}, \ldots, C_{in_i}$. Let us call the ICs C_{ij} *instantiations* of C_i. Thus, the probability of a world w is given by the product:

$$P(w) = \prod_{i=1}^{n} \prod_{C_{ij} \in w} p_i \prod_{C_{ij} \notin w} (1 - p_i).$$

$P(w)$ so defined is a probability distribution over the set of worlds W. The probability $P(\oplus|w, I)$ of the positive class given an interpretation I, a background knowledge B and a world w is defined as the probability that w satisfies I and is given by $P(\oplus|w, I) = 1$ if $M(B \cup I) \models w$ and 0 otherwise. The probability $P(\oplus|I)$ of the positive class given an interpretation I and a background B is the probability of a PCLT T satisfying I. From now on we always assume B as given and we do not mention it again. $P(\oplus|I)$ is given by

$$P(\oplus|I) = \sum_{w \in W} P(\oplus, w|I) = \sum_{w \in W} P(\oplus|w, I)P(w|I) = \sum_{w \in W, M(B \cup I) \models w} P(w) \tag{4}$$

The probability $P(\ominus|I)$ of the negative class given an interpretation I is the probability of I not satisfying T and is given by $1 - P(\oplus|I)$.

Computing $P(\oplus|I)$ with Formula (4) would be impractical as there is an exponential number of worlds. We can associate a Boolean random variable X_{ij} to each instantiated constraint C_{ij} with the meaning that $X_{ij} = 1$ in a world if C_{ij} is included in the world. So $P(X_{ij}) = p_i$ and $P(\overline{X_{ij}}) = 1 - p_i$. Let $\mathbf{X}$ be the set of the X_{ij} variables. These variables are all mutually independent. A valuation ν is an assignment of a truth value to all variables in $\mathbf{X}$. There is clearly a one to one correspondence between worlds and valuations. A valuation can be represented as a set containing X_{ij} or $\overline{X_{ij}}$ for each X_{ij}.

Suppose a ground IC C_{ij} is violated in I. The worlds where X_{ij} holds in the respective valuation are thus excluded from the summation in Formula (4). We must keep only the worlds where $\overline{X_{ij}}$ holds in the respective valuation for all ground constraints C_{ij} violated in I. So I satisfies all the worlds where the formula

$$\phi = \bigwedge_{i=1}^{n} \bigwedge_{M(B\cup I)\not\models C_{ij}} \overline{X_{ij}}$$

is true in the respective valuations, so

$$P(\oplus|I) = P(\phi) = \prod_{i=1}^{n}(1-p_i)^{m_i} \tag{5}$$

where m_i is the number of instantiations of C_i that are not satisfied in I, since the random variables are all mutually independent. So $P(\oplus|I)$ can be computed in a time that is linear in the number of ground constraints.

Example 2 (Example 1 continued). Consider the PCLT

$$\{C_1 = 0.5 \ :: \ \mathit{triangle}(T), \mathit{square}(S), \mathit{in}(T,S) \rightarrow \mathit{false}\}$$

In the left picture of Figure 1 the body of C_1 is true for the single substitution $T/2$ and $S/1$ thus $m_1 = 1$ and $P(\oplus|I_l) = 0.5$. In the right picture of Figure 1 the body of C_1 is true for three couples (triangle, square) thus $m_1 = 3$ and $P(\oplus|I_r) = 0.125$.

4 Learning Probabilistic Constraint Logic Theories

Let us consider first the parameter learning problem that can be expressed as follows.

Given

- a PCLT theory T
- a set $\mathcal{I}^+ = \{I_1, \ldots, I_Q\}$ of positive interpretations
- a set $\mathcal{I}^- = \{I_{Q+1}, \ldots, I_R\}$ of negative interpretations
- a normal logic program B (background knowledge)

Find: the parameters of T such that the likelihood

$$L = \prod_{q=1}^{Q} P(\oplus|I_q) \prod_{r=Q+1}^{R} P(\ominus|I_r)$$

is maximized. The likelihood can be unfolded to

$$L = \prod_{q=1}^{Q}\prod_{l=1}^{n}(1-p_l)^{m_{lq}} \prod_{r=Q+1}^{R}\left(1-\prod_{l=1}^{n}(1-p_l)^{m_{lr}}\right) \tag{6}$$

where m_{iq} (m_{ir}) is the number of instantiations of C_i that are false in I_q (I_r) and n is the number of ICs. Let us compute the derivative of the likelihood with respect to the parameter p_i. We first aggregate the positive examples

$$L = \prod_{l=1}^{n}(1-p_l)^{m_{l+}} \prod_{r=Q+1}^{R}\left(1-\prod_{l=1}^{n}(1-p_l)^{m_{lr}}\right) \tag{7}$$

where $m_{l+} = \sum_{q=1}^{Q} m_{lq}$. Then the partial derivative with respect to p_i is

$$
\begin{aligned}
\frac{\partial L}{\partial p_i} &= \frac{\partial \prod_{l=1}^{n}(1-p_l)^{m_{l+}}}{\partial p_i} \prod_{r=Q+1}^{R}\left(1-\prod_{l=1}^{n}(1-p_l)^{m_{lr}}\right) && (8)\\
&+\prod_{l=1}^{n}(1-p_l)^{m_{l+}} \frac{\partial \prod_{r=Q+1}^{R}\left(1-\prod_{l=1}^{n}(1-p_l)^{m_{lr}}\right)}{\partial p_i} && (9)\\
&= -m_{i+}(1-p_i)^{m_{i+}-1} \prod_{l=1,l\neq i}^{n}(1-p_l)^{m_{l+}} \prod_{r=Q+1}^{R}\left(1-\prod_{l=1}^{n}(1-p_l)^{m_{lr}}\right) && (10)\\
&+\prod_{l=1}^{n}(1-p_l)^{m_{l+}} \sum_{r=Q+1}^{R} m_{ir}(1-p_i)^{m_{ir}-1} \prod_{l=1,l\neq i}^{n}(1-p_l)^{m_{lr}} && (11)\\
&\cdot \prod_{r'=Q+1,r'\neq r}^{R}\left(1-\prod_{l=1}^{n}(1-p_l)^{m_{lr'}}\right) && (12)\\
&= -m_{i+}(1-p_i)^{m_{i+}-1} \prod_{l=1,l\neq i}^{n}(1-p_l)^{m_{l+}} \frac{(1-p_i)^{m_{i+}}}{(1-p_i)^{m_{i+}}} \prod_{r=Q+1}^{R}\left(1-\prod_{l=1}^{n}(1-p_l)^{m_{lr}}\right) && (13)\\
&+\prod_{l=1}^{n}(1-p_l)^{m_{l+}} \sum_{r=Q+1}^{R} m_{ir}\frac{\prod_{l=1}^{n}(1-p_l)^{m_{lr}}}{1-p_i} \prod_{r'=Q+1,r'\neq r}^{R}\left(1-\prod_{l=1}^{n}(1-p_l)^{m_{lr'}}\right) && (14)\\
&\cdot\frac{1-\prod_{l=1}^{n}(1-p_l)^{m_{lr}}}{1-\prod_{l=1}^{n}(1-p_l)^{m_{lr}}} && (15)\\
&= -\frac{m_{i+}(1-p_i)^{m_{i+}-1}L}{(1-p_i)^{m_i}} + \sum_{r=Q+1}^{R}\frac{m_{ir}\prod_{l=1}^{n}(1-p_l)^{m_{lr}}L}{(1-p_i)(1-\prod_{l=1}^{n}(1-p_l)^{m_{lr}})} && (16)\\
&= -\frac{m_{i+}L}{1-p_i} + \sum_{r=Q+1}^{R}\frac{m_{ir}\prod_{l=1}^{n}(1-p_l)^{m_{lr}}L}{(1-p_i)(1-\prod_{l=1}^{n}(1-p_l)^{m_{lr}})} && (17)
\end{aligned}
$$

$$= \frac{L}{1-p_i}\left(\sum_{r=Q+1}^{R} \frac{m_{ir}\prod_{l=1}^{n}(1-p_l)^{m_{lr}}}{1-\prod_{l=1}^{n}(1-p_l)^{m_{lr}}} - m_{i+}\right) \qquad (18)$$

$$= \frac{L}{1-p_i}\left(\sum_{r=Q+1}^{R} m_{ir}\frac{P(\oplus|I_r)}{P(\ominus|I_r)} - m_{i+}\right) \qquad (19)$$

The equation $\frac{\partial L}{\partial p_i} = 0$ does not admit a closed form solution so we must use optimization to find the maximum of L.

We can optimize the likelihood with Limited-memory BFGS (L-BFGS) [16], an optimization algorithm in the family of quasi-Newton methods that approximates the Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm using a limited amount of computer memory. L-BFGS requires the computation of L and of its derivative at various points. These are given by equations (7) and (19). As can be seen from these equations, computing these quantities is linear in the number of examples.

Secondly, the structure learning problem can be expressed as
Given

– a set $\mathcal{I}^+ = \{I_1, \ldots, I_Q\}$ of positive interpretations
– a set $\mathcal{I}^- = \{I_{Q+1}, \ldots, I_R\}$ of negative interpretations
– a normal logic program B (background knowledge)

Find: a PCLT T such that the likelihood

$$L = \prod_{q=1}^{Q} P(\oplus|I_q) \prod_{r=Q+1}^{R} P(\ominus|I_r)$$

is maximized.

This problem can be solved by first identifying good candidate clauses and then searching for a theory guided by the LL of the data. Candidate clauses are found by building an initial beam of top clauses - generated similarly to Progol's bottom clauses [13] according to a user-defined language bias - and by refining them through revision operators in a beam search. For a user-defined number NS of times, two interpretations are selected, one positive and one negative, for building the bottom clauses, so the initial beam will contain $2NS$ clauses. We first generate candidate clauses instead of using directly a covering loop because coverage is probabilistic, so the probability of negative interpretations of being excluded may increase as more ICs are false in the interpretations. To score the revisions, we can learn the clauses' parameter by function LearnParams and use the resulting log likelihood (LL'') as the heuristic. This function takes as input a theory, the examples and the background knowledge and returns the theory with updated parameters. The scored refinements are inserted back into the beam in order of heuristic. If the beam exceeds a maximum predefined size, the last element is removed.

Refinements in line 9 are found using a revision operator that adds a literal from the top clause to the clause under refinement. The top clause is the most general clause in the language bias and is obtained by saturation from the atoms in an interpretation, ensuring that the input/output modes of the atoms specified by the language bias are respected, similarly to [13]. Refinements as well must respect the input-output modes of the bias declarations, must be connected (i.e., each body literal must share a variable with the head or a previous body literal) and the total number of clause literals must not exceed a user-defined number ML.

The output of this search phase is represented by a list CC of candidate clauses sorted on decreasing LL.

The second phase is a greedy search in the space of theories starting with an empty theory T. Then one clause Cl at a time is added from the CC list. After each addition, parameter learning is run on the new theory T' and the log likelihood LL' of the data is computed as the score of T'. If LL' is better than the current best, the clause is kept in the theory, otherwise it is discarded. The PASCAL system that implements this strategy is shown in Algorithm 2.

Algorithm 2 Function PASCAL

```
1: function PASCAL($\mathcal{I}^+, \mathcal{I}^-, B, T, \mathcal{H}, NS, ML, BeamSize, MaxSteps$)
2:     $CC = []$
3:     $Steps = 1$
4:     $Beam \leftarrow$ InitialBeam($NS$)                          ▷ generation of Bottom Clauses
5:     repeat
6:         $NewBeam = []$
7:         while $Beam$ is not empty do                              ▷ Clause search
8:             Remove the first couple $((Cl, TopCl), LL)$ from $Beam$
9:             Find all refinements $Ref$ of $Cl$ with at most $ML$ literals
10:            for all $(Cl', TopCl') \in Ref$ do
11:                $(\{Cl''\}, LL'') \leftarrow$ LearnParams($\{Cl'\}, \mathcal{I}^+, \mathcal{I}^-, B$)
12:                $NewBeam \leftarrow$ Insert($(Cl'', TopCl'), LL'', NewBeam$)
13:                if $size(NewBeam) > BeamSize$ then
14:                    Remove the last element of $NewBeam$
15:                end if
16:                $CC \leftarrow$ Insert($(Cl'', LL''), CC$)
17:            end for
18:        end while
19:        $Beam \leftarrow NewBeam$
20:        $Steps = Steps + 1$
21:    until $Steps > MaxSteps$
22:    $T \leftarrow \emptyset$, $LL \leftarrow -\infty$                        ▷ Theory search
23:    repeat
24:        Remove the first couple $(Cl, ClLL)$ from $CC$
25:        $(T', LL') \leftarrow$ LearnParams($T \cup \{Cl\}, \mathcal{I}^+, \mathcal{I}^-, B$)
26:        if $LL' > LL$ then
27:            $T \leftarrow T'$, $LL \leftarrow LL'$
28:        end if
29:    until $CC$ is empty
30:    return $T$
31: end function
```

5 Related Work

The approach here presented is very similar in spirit to the distribution semantics [17]: a probabilistic theory defines a distribution over non-probabilistic theories by assuming independence among the choices in probabilistic constructs. The distribution semantics has emerged as one of the most successful approaches in Probabilistic Logic Programming (PLP) and underlies many languages such as Probabilistic Horn Abduction, Independent Choice Logic, PRISM, Logic Programs with Annotated Disjunctions and ProbLog.

In the distribution semantics, the aim is to compute the probability that a ground atom is true. However, performing such inference requires an expensive procedure that is usually based on knowledge compilation. For example, ProbLog [7] builds a Boolean formula and compiles it into a Binary Decision Diagram from which the computation of the probability is linear in the size of the diagram. However, the compilation procedure is #P in the number of variables. On the contrary, computing the probability of the positive class given an interpretation in a PCLT is linear in the number of variables.

This places PCLTs in the recent line of research devoted to identifying tractable probabilistic languages. PCLTs exploit the research done in the ILP subfield of learning from interpretations for achieving tractability.

The assumption of independence of the constraints may seem strong. Of course, not assuming independence may result in a finer modeling of the domain. However, this would preclude the nice computational properties of PCLTs. Achieving tractability requires approximations and we think that the independence of the constraints is a reasonable assumption, similar to the independence of probabilistic choices in the distribution semantics for PLP.

Each constraint may have a different number of violated groundings, which may result in a larger weight associated with constraints with many groundings. However, the parameters are learned from data in order to maximize the likelihood, so the parameters are adjusted to take into account the number of groundings.

A system related to PASCAL is 1BC [11] that induces first-order features in the form of conjunctions of literals and combines them using naive Bayes in order to classify examples. First-order features are similar to integrity constraints with an empty head: they check the existence of values for the variables that satisfy the conjunction. The probability of a feature is computed by relative frequency in 1BC. This can lead to suboptimal results if compared to PASCAL, where the probabilities are optimized to maximize the likelihood.

6 Experiments

PASCAL has been implemented in SWI-Prolog [19] and has been applied to the process mining dataset considered in [1]. For performing L-BFGS we ported the YAP-LBFGS library developed by Bernd Gutmann to SWI-Prolog. This library is based on libLBFGS. PASCAL is compared to the DPML system [12] for performing process mining with ILP. DPML implements a learning algorithm very similar to ICL (Algorithm 1).

The dataset collects the careers of students enrolled at the Faculty of Engineering of the University of Ferrara from 2004 to 2009. Each career records the main events such as all the chronological enrollments, the exams taken and the conclusion of the career (degree or not). The dataset records 776 interpretations each corresponding to a different student career. The careers of students who graduated are positive interpretations while those who did not finish their studies are negative interpretations. In particular, it contains 304 positive and 472 negative interpretations. All experiments have been performed on GNU/Linux machines with an Intel Xeon Haswell E5-2630 v3 (2.40GHz) CPU and 128 GB RAM.

PASCAL offers the following parameters: the size *BeamSize* of the beam, the maximum number of literals *ML* per clause refinement, the maximum number *NS* of iterations for building the bottom clauses, the maximum number *MaxSteps* of clause search iterations. They have been set as *BeamSize*=100, *ML*=10, *NS*=4, *MaxSteps*=50 respectively. DPML offers the following parameters: the maximum number of literals in the head and in the body of a rule (set to 8); the minimal accuracy for each rule (set to 0.75); the beam size (set to 10), and the number of clause refinement iterations (set to 50). Five-fold cross validation has been applied, with each fold containing 60 or 61 positive and 94 or 95 negative examples.

From the probability estimates on the test examples we computed the average accuracy and the Area Under the Precision Recall curve and ROC curve (AUCPR and AUCROC respectively). Accuracy is computed as the average of the greatest accuracies over the folds. Since the theories returned by DPML are sharp, in order to compute their AUCPR and AUCROC we annotated each learned clause with probability 1.0 and applied the same testing technique as PASCAL. Table 1 reports the log-likelihood over the test set, the accuracy, the AUCPR, AUCROC and learning time averaged over the folds for both systems. These results show that a probabilistic approach at learning constraint models may produce better results than a sharp approach realized by a purely logical system.

Table 1. Results of the experiments in terms of log-likelihood over the test set, Area Under the PR and ROC Curves, accuracy and learning time (in seconds) averaged over the folds.

System	LL	AUCROC	AUCPR	Accuracy	Time(s)
PASCAL	-302.664	0.923	0.851	0.889	568.509
DPML	-440.254	0.707	0.53	0.656	280.594

7 Conclusions

We have proposed a probabilistic extension of sets of integrity constraints together with the PASCAL algorithm for learning them. PASCAL exploits L-BFGS for tuning the parameters and constraint refinement for searching for good structures. Preliminary experiments on a process mining dataset show that the approach can overcome a probabilistic process mining system.

References

1. Bellodi, E., Riguzzi, F., Lamma, E.: Probabilistic declarative process mining. In: Bi, Y., Williams, M.A. (eds.) KSEM 2010. LNCS, vol. 6291, pp. 292–303. Springer, Heidelberg (2010)
2. Blockeel, H., De Raedt, L., Jacobs, N., Demoen, B.: Scaling up inductive logic programming by learning from interpretations. Data Min. Knowl. Discov. 3(1), 59–93 (1999)
3. Bongard, M.M.: Pattern Recognition. Hayden Book Co., Spartan Books (1970)
4. Van den Broeck, G.: On the completeness of first-order knowledge compilation for lifted probabilistic inference. In: NIPS 2011. pp. 1386–1394 (2011)
5. Clark, K.L.: Negation as failure. In: Gallaire, H., Minker, J. (eds.) Logic and Databases. Plenum Press, New York, USA (1978)
6. De Raedt, L., Dehaspe, L.: Clausal discovery. Mach. Learn. 26(2-3), 99–146 (1997)
7. De Raedt, L., Kimmig, A., Toivonen, H.: ProbLog: A probabilistic Prolog and its application in link discovery. In: IJCAI 2007. vol. 7, pp. 2462–2467. AAAI Press (2007)
8. De Raedt, L., Van Laer, W.: Inductive constraint logic. In: ALT 1995. LNAI, vol. 997, pp. 80–94. Springer, Heidelberg (1995)
9. De Raedt, L., Dzeroski, S.: First-order jk-clausal theories are pac-learnable. Artif. Intell. 70(1-2), 375–392 (1994)
10. Domingos, P., Webb, W.A.: A tractable first-order probabilistic logic. In: AAAI 2012. AAAI Press (2012)
11. Flach, P.A., Lachiche, N.: Naive bayesian classification of structured data. Mach. Learn. 57(3), 233–269 (2004)
12. Lamma, E., Mello, P., Riguzzi, F., Storari, S.: Applying inductive logic programming to process mining. In: ILP 2008. pp. 132–146. No. 4894 in LNAI, Springer (2008)
13. Muggleton, S.: Inverse entailment and Progol. New Generation Computing 13, 245–286 (1995)
14. Niepert, M., Van den Broeck, G.: Tractability through exchangeability: A new perspective on efficient probabilistic inference. In: AAAI 2014. pp. 2467–2475 (2014)
15. Niepert, M., Domingos, P.: Tractable probabilistic knowledge bases: Wikipedia and beyond. In: Workshop on Statistical Relational Artificial Intelligence. vol. WS-14-13. AAAI (2014)
16. Nocedal, J.: Updating quasi-newton matrices with limited storage. Math. Comput. 35(151), 773–782 (1980)
17. Sato, T.: A statistical learning method for logic programs with distribution semantics. In: ICLP 1995. pp. 715–729. MIT Press (1995)
18. Webb, W.A., Domingos, P.: Tractable probabilistic knowledge bases with existence uncertainty. In: Workshop on Statistical Relational Artificial Intelligence (2013)
19. Wielemaker, J., Schrijvers, T., Triska, M., Lager, T.: SWI-Prolog. Theor. Pract. Log. Prog. 12(1-2), 67–96 (2012)