Distributional Logic Programming: a brief overview

Nicos Angelopoulos
University of Edinburgh, Edinburgh, EH9 3JR, Scotland, UK
n.angelopoulos@ed.ac.uk

1 Introduction

Logic programming (LP) is an attractive formalism for representing crisp knowledge. Probabilistic extensions to logic programming have been previously proposed for the purpose of representing Bayesian priors ((Cussens, 2000; Angelopoulos & Cussens, 2001)). Here, we present an extension to the probabilistic aspects of their formalism based on probabilistic guards. Current approaches to logical-probabilistic formalisms include the (almost complete) replacement of non-determinism by probabilistic operator, the use of a primitive that appears within limited non-determinism and a clear separation of the two spaces. In the first category, SLPs under the semantics presented in (Cussens, 2000) replaces the SLD with sampling over pure programs, which only contain stochastic clauses.

An example of the second category is that of Prism, see for instance (Sato & Kameya, 2001). It provides a single probabilistic construct that instantiates an unbound variable from the elements of a list according to the probability values attached to each element. It was introduced with parameter learning in the context of PCFGs and hidden Markov models in mind.

2 Syntax

We extend the clausal syntax of Logic Programming with probabilistic guards that associate a resolution step to a probability that can be computed on-the-fly. The main intuition is that in addition to the logical relation a clause defines over the objects that appear as arguments in its head, it can also define a probability distribution over aspects of this relation. A Dlp probabilistic clause is an extension of the definite clause and it is of the form:

$$Expr : GVars \cdot Guard \sim PVars : Head \text{ :- } Body \tag{1}$$

Arithmetic expressions in the clause defined by (1) will be evaluated at resolution time to a probability value. In cases where this can be done successfully, the clauses will be used to define a distribution over the probabilistic variables ($PVars$). The distribution may depend on an arbitrary number of input terms via calls to the guard. We also allow goals that appear in the body of clause definitions to be labelled by a tuple of unary functions each wrapping an arithmetic expression. Each of the unary functions corresponds to the functions in $GVars$. The intuition behind labelled goals in the body of clauses ($Body$) is that often probability labels of recursive calls can be easily computed from their parent call thus the interpreter can avoid recomputing all or some of the guards. For a single probabilistic predicate all clauses must define the same set of probabilistic variables. In what follows we let $C_i^{\sim}$ denote the set of probabilistic variables of clause C_i.

By comparison to the standard LP $member/2$ relation, consider the predicate $pmember/2$:

(C_3) $\frac{1}{L}$: l(L) · length([H|T], L), $0 < L$ ~H:
pmember(H, [H|T]).
(C_4) $1 - \frac{1}{L}$: l(L) · length([H|T],L), $0 < L$ ~El:
pmember(El, [H|T]) :-
l(L-1): pmember(El,T).

These clauses have attached to them expressions which will be computed at resolution time. (C_3) is labelled by $\frac{1}{L}$ where L is the length of the input list. Similarly (C_4) claims the residual probability.

The recursive call has been augmented to carry forward the value of L as the length of T is one less than that of the input list and thus we avoid recomputing the guard. Intuitively, this program when queried by $? - pmember(X, List)$. for a concrete list it defines an equiprobable distribution over selection among the elements of the list. The three corresponding probabilities when $List = [a, b, c]$ are computed as $\frac{1}{3}, \frac{2}{3} \times \frac{1}{2}, \frac{1}{3} \times \frac{1}{2} \times 1$.

The full syntax is usually not necessary for simple predicate definition. Also, in the interest of clarity guard lines can be introduced to the programs which factor the guard section out. The example program thus becomes:

(G_1) L ·length(List, L), $0 < L$ ∼El : pmember(El,List).

(C'_3) $\frac{1}{L}$: G_1 : pmember(H, [H|T]).

(C'_4) $1 - \frac{1}{L}$: pmember(El, [H|T]) :-
L-1: pmember(El, T).

3 Example of a prior

(Chipman H, 1998) uses a prior over the set of classification trees T that depends on splitting individual nodes:$p(T) = \psi_{e_\eta} = \alpha(1 + e_\eta)^{-\beta}$ where d_η is the depth of node η and α and β, are user defined parameters controlling the size of the trees. The main part of the probabilistic program for constructing trees according to the presented prior is as follows:

$$
\begin{array}{lll}
(A_0) & cart(D, Cart) : - & \\
& & parameters(\alpha, \beta), \quad \psi_0 \text{ is } \alpha, \\
& \psi_0: & split(0, D, Cart).
\end{array}
$$

$$
\begin{array}{lll}
(A_1) & \psi_H: & split(E_H, D_H, c(F, Val, L, R)) : - \\
& & parameters(\alpha, \beta), \quad E_{H_1} \text{ is } E_H + 1, \quad \psi_{H_1} \text{ is } \alpha * E_{H_1}^{-\beta}, \\
& & r\_select(F, Val, D_H, L_H, R_H), \\
& \psi_{H_1}: & split(E_{H_1}, L_H, L), \\
& \psi_{H_1}: & split(E_{H_1}, R_H, R).
\end{array}
$$

$$(A_2)\ 1 - \psi_D: \quad split(D, D_H, l(D_H)). \qquad (A_3) : parameters(\alpha, \beta).$$

Clause (A_0) defines $Cart$ to be a valid representation of a tree given data D, and generated with probability equal to that described above. For each split at depth E_H the leaf nodes are considered in turn with a decision made for each of them as where to split the node or not. The node will either become an internal one via (A_1) or a leaf node by application of (A_2). Each time a split is considered (A_1) is selected with probability ψ_H and (A_2) with the complementary probability $1 - \psi_H$. The Dlp program captures the essence of the prior in an elegant and abstract way. (Angelopoulos & Cussens, 2005) has shown that such languages can be used for statistical inference via Bayesian model averaging. The MCMCMS system (`http://scibsfs.bch.ed.ac.uk/~nicos/sware/mcmcms`) supports both stochastic and distributional logic programs.

References

Angelopoulos, N., & Cussens, J. (2001). MCMC using tree-based priors on model structure. In *17th Uncertainty in AI (UAI-2001)*, pp. 16–23.

Angelopoulos, N., & Cussens, J. (2005). Exploiting Informative Priors for Bayesian Classification and Regression Trees. In *19th Int. Joint Conference on AI* Edinburgh, UK.

Chipman H, George E, M. R. (1998). Bayesian CART Model Search (with discussion). *Journal of the American Statistical Association*, *93*, 935–960.

Cussens, J. (2000). Stochastic logic programs: Sampling, inference and applications. In *16th Annual Conference on Uncertainty in Artificial Intelligence (UAI-2000)*, pp. 115–122.

Sato, T., & Kameya, Y. (2001). Parameter Learning of Logic Programs for Symbolic-statistical Modeling. *Journal of AI Research*, *15*, 391–454.