Probabilistic (Logic) Programming Concepts*

Angelika Kimmig

Department of Computer Science, KU Leuven, Belgium
angelika.kimmig@cs.kuleuven.be

The substantial interest in statistical relational learning [3], probabilistic (inductive) logic programming [1] and probabilistic programming languages [6] has resulted in a wide variety of different formalisms, models and languages, with applications in structured, uncertain domains such as natural language processing, bioinformatics, and activity recognition. The multitude of probabilistic languages that exists today provides evidence for the richness and maturity of the field, but on the other hand, makes it hard to get an appreciation and understanding of the relationships and differences between the different languages.

Furthermore, most arguments in the literature about the relationship amongst these languages are about the expressiveness of these languages, that is, they state (often in an informal way) that one language is more expressive than another one (implying that the former could be used to emulate the latter). By now, it is commonly accepted that the more interesting question is concerned with the underlying concepts that these languages employ and their effect on the inference mechanisms, as their expressive power is often very similar. However, a multitude of different probabilistic primitives exists, which makes it hard to appreciate their relationships.[1]

To alleviate these difficulties and obtain a better understanding of the field we identify a number of core probabilistic programming concepts and relate them to one another. We cover the basic concepts representing different types of random variables, but also general modeling concepts such as negation or time and dynamics, and programming constructs such as meta-calls and ways to handle sets. While doing so, we focus on probabilistic extensions of logic programming languages because this is (arguably) the first and best studied probabilistic programming paradigm. It has been studied for over 20 years starting with the seminal work of David Poole [5] and Taisuke Sato [7], and now includes languages such as the independent choice logic (ICL), stochastic logic programs (SLPs), PRISM, Bayesian logic programs (BLPs), CLP($\mathcal{BN}$), logic programs with annotated disjunctions (LPADs), P-log, Dyna, CP-logic, ProbLog, and programming with personalized Pagerank (PROPPR). Another reason for focussing on probabilistic extensions of logic programming languages is that the concepts are all embedded within the same host language, so we can focus on semantics rather than syntax. At the same time, we also relate the concepts to

* Extended abstract of [2]

[1] Throughout the paper we use the term *primitive* to denote a particular syntactic and semantic construct that is available in a particular probabilistic programming language, and the term *concept* to denote the underlying notion. Different primitives may hence realize the same concept.

alternative probabilistic programming languages such as IBAL, Bayesian logic (BLOG), Church and Figaro and to some extent also to statistical relational learning models such as relational Bayesian networks (RBNs), probabilistic relational models (PRMs) and Markov logic. Most statistical relational learning approaches employ a knowledge-based model construction approach, in which the logic is used as a template for constructing a graphical model; see [4] for a recent overview. Typical probabilistic programming languages, on the other hand, employ a variant of Sato's distribution semantics [7], in which random variables directly correspond to ground facts and a traditional program specifies how to deduce further knowledge from these facts. This difference explains why we introduce the concepts in the context of the distribution semantics, and discuss approaches to knowledge-based model construction separately.

Inference, that is, evaluating the probability distribution defined by a program or model, is a key challenge in probabilistic programming and statistical relational learning. Furthermore, the choice of inference approach often influences which probabilistic primitives can be supported. Enormous progress has been made in the past few years w.r.t. probabilistic inference and numerous inference procedures have been contributed. Therefore, we also identify some core classes of inference mechanisms for probabilistic programming and discuss which ones to use for which probabilistic concept. Inference in probabilistic languages also is an important building block of approaches that learn the structure and/or parameters of such models from data. Given the variety of approaches that exist today, a discussion of learning is beyond the scope of this paper.

To summarize, the key contributions of this paper are (1) the identification of a number of core concepts that are used by various probabilistic languages, (2) a discussion of the execution mechanisms that they require, and (3) a positioning of state-of-the-art probabilistic languages and implementations w.r.t. these concepts.

References

1. De Raedt, L., Frasconi, P., Kersting, K., Muggleton, S. (eds.): Probabilistic Inductive Logic Programming — Theory and Applications, Lecture Notes in Artificial Intelligence, vol. 4911. Springer (2008)
2. De Raedt, L., Kimmig, A.: Probabilistic (logic) programming concepts. Machine Learning 100(1), 5–47 (2015)
3. Getoor, L., Taskar, B. (eds.): An Introduction to Statistical Relational Learning. MIT Press (2007)
4. Kimmig, A., Mihalkova, L., Getoor, L.: Lifted graphical models: A survey. Machine Learning 99(1), 1–45 (2015)
5. Poole, D.: Logic programming, abduction and probability. In: Fifth Generation Computing Systems. pp. 530–538 (1992)
6. Roy, D., Mansinghka, V., Winn, J., McAllester, D., Tenenbaum, D. (eds.): Probabilistic Programming. NIPS Workshop (2008)
7. Sato, T.: A statistical learning method for logic programs with distribution semantics. In: Proceedings of the 12th International Conference on Logic Programming (ICLP-95) (1995)