MR3931324 68-02 60A99 62G99 62H99 68N17 68Q55

Riguzzi, Fabrizio (I-FERR-MI; Ferrara)

★**Foundations of probabilistic logic programming.**

Languages, semantics, inference and learning.
With a foreword by Agostino Dovier.
River Publishers Series in Software Engineering.
River Publishers, *Gistrup*, 2018. *xxxiii*+387 *pp.* *ISBN* 978-87-7022-018-7; 978-87-7022-017-0

Probabilistic logic programming (PLP) is an attempt to find a bridge between computational logic and modern machine learning methods. This book gives a nice, self-contained introduction to PLP. It consists of 12 chapters, which can be grouped into five parts: preliminaries, languages, semantics, inference, and learning.

Preliminaries. The first chapter is devoted to the material which can equip the reader with the necessary background to understand the subsequent topics. It starts by recalling notions related to orders, lattices, ordinals, mapping, and fixpoints, followed by a concise introduction to logic programming and its semantics, probability theory, and probabilistic graphical models.

Languages. The existing approaches to combining logic programming with probability can be classified into two categories: those based on Sato's Distribution Semantics (DS-based) and those based on Knowledge Base Model Construction (KBMC-based).

The second chapter of the book is dedicated to these approaches. It starts with DS-based languages that do not contain function symbols. Here, the idea is to use a probabilistic logic program to define a probability distribution over normal logic programs (worlds). Then, for a given query, the distribution is extended to a joint distribution of the query and the worlds, from which the probability of the query is computed by summing over the worlds. To define the distribution over programs, random choices are encoded for clauses, and the set of choices induces the probability distribution. Different DS-based languages may choose different encodings or different ways of stating probabilities for choices, but in the end, all of them have the same expressive power. Logic Programs with Annotated Disjunctions (LPAD), ProbLog, Probabilistic Horn Abduction, Independent Choice Logic, PRISM, and CP-logic are the languages considered in this part. Well-chosen examples help the reader to understand their syntax and semantics. This part also contains a section on a translation of LPADs into Bayesian networks and a discussion on semantics for non-sound probabilistic logic programs.

The KBMC-based languages are represented by three examples: Bayesian Logic Programs (BLPs), CLP(BN), and the Prolog Factor Language (PFL). In these languages, a program is a template for generating a ground probabilistic graphical model. BLPs generalize both logic programs and Bayesian networks. Ground atoms correspond to random variables and clauses represent the idea of conditional probability densities, defining the dependencies between ground atoms. CLP(BN) is based on constraint logic programming and uses Bayesian networks to represent the joint probability distribution over terms constructed from Skolem symbols. Probabilistic dependencies are expressed by constraints. PFL is an extension of Prolog for representing first-order probabilistic models. In general, the treatment of KBMC-based languages is brief. The author does not go into as much detail as he does for DS-based languages. This is not surprising, since the emphasis of the book is on DS-based languages.

This part ends with a brief discussion of a couple of frameworks for probabilistic logic programming that do not follow the distribution semantics, and on probabilistic logic languages that are not based on logic programming.

Semantics. This part includes semantics for programs with function symbols (Chapter

3) and semantics for hybrid programs (Chapter 4). For programs with function symbols, the set of ground instances (grounding) of a clause is infinite and the number of worlds is uncountable. Therefore, the number of atomic choices in a selection that defines a world is infinite. It may lead to the consequence that the probability of each individual world is zero, and the semantics defined for function-free programs is not applicable anymore. The material presented in Chapter 3 deals with this problem. Semantics for programs with function symbols is defined based on sets of choices that ensure that the query is true. Such sets are called explanations. They identify sets of worlds that entail the query. Then the probability of a query q is a function of the set of all possible explanations for q (called the covering set of explanations for q).

Hybrid probabilistic programs include continuous random variables alongside discrete ones. In Chapter 4, some hybrid languages are introduced and their semantics is presented. Hybrid ProbLog is one such language, which extends ProbLog with continuous probabilistic facts. The number of continuous variables is finite. The approach to defining the semantics of hybrid ProbLog programs is based on discretization techniques. The space of possible assignments of continuous variables is discretized into intervals. The actual values within intervals do not play a role. The semantics of a hybrid program is then defined based on the semantics of the discretized program. For this approach to work, there are some restrictions imposed on the occurrences of continuous variables. Other languages considered in this chapter are Distributional Clauses, an extension of PRISM that includes continuous random variables with a Gaussian or gamma distribution, hybrid programs on cplint, and Probabilistic Constraint Logic Programming.

Inference. This is the largest part of the book and comprises four chapters: Exact Inference (Chapter 5), Lifted Inference (Chapter 6), Approximate Inference (Chapter 7), and Non-standard Inference (Chapter 8). There is a strong link with the material in the previous part, connecting semantics with statistical inference. It is in the tradition of logic programming, where connections between declarative and operational semantics are well explored.

In PLP, various inference tasks are considered. Given two conjunctions of ground literals, q (query) and e (evidence):

(1) the EVID task is to compute the probability of evidence: $P(e)$;
(2) the COND task is to compute the conditional probability distribution of the query given the evidence: $P(q|e)$;
(3) the MAP task (maximum a posteriori) is to compute the most likely value of a set of non-evidence atoms given the evidence; and
(4) the DISTR task is to compute the probability distribution or density of the non-ground arguments of a conjunction of literals.

Some more special tasks include MPE (most probable explanation), which is a special case of MAP considering the set of all non-evidence atoms, and CONDATOMS, which is a special case of COND assuming only atoms in q.

Exact inference aims at solving these tasks in an exact way and is performed either by dedicated algorithms for special cases, knowledge compilation, conversion to graphical models, or by lifted inference. The first three methods are discussed in Chapter 5. Inference in PRISM can be seen as an example of inference in a special case since this system requires subgoals to be independent and clauses in programs to be exclusive. The knowledge compilation approach is more general. It first converts the program, the query, and the evidence into a Boolean formula encoding the covering set of explanations, and then performs the actual compilation, converting the formula into a form from which the probability can be easily computed. In the book, this process is illustrated for various systems and different inference tasks. The chapter ends with a discussion of inference

for programs with function symbols and hybrid programs.

Lifted inference has been introduced for efficiency reasons since reasoning with real-world models is often very expensive. It tries to exploit symmetries in the model to achieve a speeding up in inference. Lifted versions of various techniques and their comparisons are discussed in Chapter 6.

Yet another way to address the complexity problem of exact inference is to relax it and consider some approximations. The approaches to approximate reasoning are classified into two groups: those that modify exact inference and those that are based on sampling. Various such techniques, used in different systems for different tasks, are defined and illustrated in Chapter 7.

The inference part of the book ends with the chapter on non-standard inference, where problems for languages that are similar to PLP are considered. This includes Possibilistic Logic Programming, Decision-Theoretic ProbLog, and Algebraic ProbLog.

Learning. This part is an attempt to connect logic programming with machine learning. PLP seems to be a promising approach in this direction. Two problems are addressed: Parameter Learning in Chapter 9, and Structure Learning in Chapter 10.

The parameter learning problem is formulated as follows: Given data (as a set of ground atoms or interpretations) and a probabilistic logic program, find the parameters of the program that maximize the probability of the data. It is illustrated how this problem is approached in several PLP systems.

The structure learning problem aims at generating whole programs (including their parameters and structure) from given data. This direction is related to Inductive Logic Programming, where data consists of three parts: positive and negative examples as sets of ground atoms, background knowledge in the form of a logic program, and a set of possible programs. The goal is to find a program among the possible ones which, together with the background knowledge, implies all positive examples and does not imply any negative ones. In the context of structure learning with PLP, one talks about Probabilistic Inductive Logic Programming (PILP). The author illustrates how learning is done in several PILP systems, where the exact problem statement depends on what kind of data is considered and what properties the final program should satisfy.

The book ends with a set of program examples and the illustration of how the cplint system answers queries to them (Chapter 11). This is a very instructive part, and I enjoyed reading it. Some open problems and future research directions are discussed in the concluding chapter (Chapter 12).

The book will be useful for readers with a background in computer science and artificial intelligence who want to learn about principles, techniques, and systems for probabilistic logic programming. *Temur Kutsia*