International Journal of Approximate Reasoning 91 (2017) 216–232

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

www.elsevier.com/locate/ijar

Causal inference in `cplint` [☆]

Fabrizio Riguzzi [a,*], Giuseppe Cota [b], Elena Bellodi [b], Riccardo Zese [b]

[a] *Dipartimento di Matematica e Informatica – Università di Ferrara, Via Saragat 1, 44122, Ferrara, Italy*
[b] *Dipartimento di Ingegneria – Università di Ferrara, Via Saragat 1, 44122, Ferrara, Italy*

A R T I C L E   I N F O

Article history:
Received 10 March 2017
Received in revised form 12 September 2017
Accepted 13 September 2017
Available online 22 September 2017

Keywords:
Probabilistic Logic Programming
Distribution semantics
Logic programs with annotated disjunctions
ProbLog
Causal inference
Statistical relational artificial intelligence

A B S T R A C T

`cplint` is a suite of programs for reasoning and learning with Probabilistic Logic Programming languages that follow the distribution semantics. In this paper we describe how we have extended `cplint` to perform causal reasoning. In particular, we consider Pearl's *do* calculus for models where all the variables are measured. The two `cplint` modules for inference, PITA and MCINTYRE, have been extended for computing the effect of actions/interventions on these models. We also executed experiments comparing exact and approximate inference with conditional and causal queries, showing that causal inference is often cheaper than conditional inference.

1. Introduction

Identifying cause-effect relationships among variables or events is one of the main objectives of science. How to extract such relationships from data and how to use them to make predictions have been fiercely debated. The connection between correlation and causation is particularly subtle and has mislead many authors. The famous sentence "correlation does not imply causation" is often used in statistics to warn against confounding the two. A correlation between two variables means that there is an association between them, but it does not imply that one causes the other. As a matter of fact, it could happen that there is a third factor that causes both producing the correlation.

Pearl in [1] showed that it is possible to represent causality by means of graphical models. Bayesian networks, in particular, are directed acyclic graphs that represent probabilistic dependencies between variables in a very intuitive way. In order to represent causality, Pearl [1] extended them into *causal Bayesian networks*, i.e. Bayesian networks where an arc from a node A to a node B means that A *directly causally influences* B. Computing the effect of actions can be performed in causal Bayesian networks by removing the edges that point to the nodes that represent the actions. The probability distribution of some variables E when performing action A, indicated as $P(E|do(A))$, denotes the effect of actions and is at the basis of Pearl's *do calculus*. This distribution can be computed by probabilistic inference on the mutilated network.

Representing probabilistic information in Logic Programming has been pursued by many authors. The distribution semantics [2] is one of the most widely used semantics for Probabilistic Logic Programming (PLP). This semantics is at the basis of many languages, such as Independent Choice Logic [3], PRISM [4], Logic Programs with Annotated Disjunctions (LPADs) [5] and ProbLog [6].

[☆] This paper is part of the Virtual special issue on Probabilistic Logic Programming 2016, edited by James Cussens and Arjen Hommersom.
[*] Corresponding author.
E-mail addresses: fabrizio.riguzzi@unife.it (F. Riguzzi), giuseppe.cota@unife.it (G. Cota), elena.bellodi@unife.it (E. Bellodi), riccardo.zese@unife.it (R. Zese).

https://doi.org/10.1016/j.ijar.2017.09.007
0888-613X/© 2017 Elsevier Inc. All rights reserved.

CP-logic [7] is a PLP language for causal reasoning. CP-logic programs (or CP-theories) are syntactically very similar to LPADs and they are given a semantics based on probability trees that represent possible courses of events. The authors proved that their semantics is suitable for representing causation and the effects of causal laws. For legal CP-logic programs, the semantics of LPADs and that of CP-logic coincide.

The authors in [7] also showed that the effect of actions in *do* calculus style can be computed from CP-theories by modifying the theory itself and computing the probability of the query from the modified theory. The modification involves adding facts for positive actions or removing (instantiations) of rules for negative actions.

In this paper we discuss how we implemented this process in practice in `cplint`,[1] a suite of programs for reasoning and learning in PLP. The two main inference modules of `cplint`, PITA and MCINTYRE, have been suitably extended for performing the *do* calculus, thus computing the effects of actions. PITA performs exact inference by knowledge compilation while MCINTYRE performs approximate inference by sampling so their extensions allow to perform both exact and approximate causal inference.

Like [7], we assume that the causal structure of the model is fully known. Pearl's *do* calculus is more general, as it allows to compute the effect of actions also on models with unknown variables. Exploiting the full power of the *do* calculus in PLP is a very interesting direction for future work.

We present two domains to illustrate causal reasoning in PLP: the famous Simpson's paradox and a viral marketing problem. We have also conducted experiments on the latter with an increasing number of members of the social network and compared exact and approximate conditional inference with exact and approximate causal inference. The results show that performing causal inference is often much less expensive than conditional inference, as expected, since actions remove dependencies among random variables.

The paper is organized as follows. Section 2 introduces preliminaries about distribution semantics, causal reasoning and PLP. Section 3 and Section 4 describe the modules PITA and MCINTYRE respectively, together with their extension for causal reasoning; Section 5 shows some notable examples of causal inference, namely the Simpson's paradox and the viral marketing problem. Section 6 illustrates related work. Section 7 reports on the experiments performed and Section 8 concludes the paper.

2. Preliminaries

2.1. Probabilistic Logic Programming

The field of PLP has seen many different proposals for integrating logic programming and probability theory. We here concentrate on the Distribution Semantics (DS) [2] because it is one of the most widely used. The basic idea of the DS is that a probabilistic logic program defines a probability distribution over a set of normal logic programs (called *worlds*) that is extended to a joint probability of programs and truth values of a ground query. The probability of the query is then obtained from this joint distribution by marginalization.

We present the DS for LPADs for their general syntax. LPADs are sets of disjunctive clauses in which each atom in the head is annotated with a probability.

Formally a *Logic Program with Annotated Disjunctions* (LPADs) [5] consists of a finite set of annotated disjunctive clauses. An annotated disjunctive clause C_i is of the form

$$h_{i1} : \Pi_{i1}; \ldots ; h_{in_i} : \Pi_{in_i} \leftarrow b_{i1}, \ldots , b_{im_i}.$$

In such a clause the semicolon stands for disjunction, $h_{i1}, \ldots h_{in_i}$ are logical atoms and $b_{i1}, \ldots , b_{im_i}$ are logical literals, $\Pi_{i1}, \ldots , \Pi_{in_i}$ are real numbers in the interval [0, 1] such that $\sum_{k=1}^{n_i} \Pi_{ik} \leq 1$. If $\sum_{k=1}^{n_i} \Pi_{ik} < 1$, the head of the annotated disjunctive clause implicitly contains an extra atom *null* that does not appear in the body of any clause and whose annotation is $1 - \sum_{k=1}^{n_i} \Pi_{ik}$.

Example 1. The following LPAD T from [8] encodes a very simple model of the development of an epidemic or a pandemic:

$$C_1 = epidemic : 0.6; \; pandemic : 0.3 \leftarrow flu(X), cold.$$
$$C_2 = cold : 0.7.$$
$$C_3 = flu(david).$$
$$C_4 = flu(robert).$$

An epidemic or a pandemic may arise if somebody has the flu and the climate is cold. We are uncertain whether the climate is cold and we know for sure that David and Robert have the flu.

We discuss the DS for the case in which the program does not contain function symbols so that its Herbrand base is finite.[2] An *atomic choice* is a selection of the k-th atom for a grounding $C_i\theta_j$ of a probabilistic clause C_i and is represented by

[1] http://sites.unife.it/ml/cplint.

[2] For the distribution semantics for programs with function symbols see [2,9,10].