ISSUES OF A BAYESIAN NETWORK-BASED FIRST-ORDER PROBABILISTIC KNOWLEDGE REPRESENTATION LANGUAGE

András MILLINGHOFFER
Advisors: Péter ANTAL, Tadeusz DOBROWIECKI

I. Introduction

In the area of knowledge representation one of the most widely spread systems is that of Bayesian networks. Despite their advantageous properties shown in the handling of informal, subjective knowledge and uncertainties, they have some drawbacks which prevent them from being applicable in larger domains. The currently used monolithic Bayesian networks do not support the reuse of previously acquired knowledge (i.e. already constructed model parts) and are unable to perform inference about multiple objects simultaneously; briefly, they lack object-orientedness and first-orderedness. Recently there has been many attempts to overcome these disadvantages, in this paper we overview the most important ones, and motivated by them we summarize the desired properties of a representation language and introduce a framework for handling the instances of the above mentioned models.

The paper is organized as follows: in Section II. we overview the framework of Bayesian networks and the underlying principles of Bayesian statistics, in Section III. the main trends of extensions to Bayesian networks are enumerated. In Section IV. we introduce the elements of the implemented system, and in Section V. we sketch the planned direction of further developments.

II. The Bayesian framework

A. Bayesian statistics

Uncertainty is an inherent property of knowledge acquisition processes: it may result from many sources, e.g. the applied method itself, the way how data is gathered or the lack or ignorance of knowledge. In the framework of Bayesian statistics these uncertainties are described through probabilistic quantities, accepting their subjectivist interpretation, considering them measures of prior beliefs about the domain, contrary to the frequentist interpretation which is more prevalent in engineering practice. In Bayesian statistics, observation data is considered to be generated by an ensemble of models parameterized by stochastic variables. In practice, like in the case of Bayesian networks, this parametrization is often structured hierarchically.

The basic task in Bayesian statistics is to calculate the posterior probability of an event conditional on the prior knowledge and observations. In case of *predictive inference*, we are interested in the probability of a generic event, the probability of which can be calculated by a sum or integral over the possible models (in the below example discrete model structures with continuous parameters):

$$P(x|D) = \sum_k P(G_k|D) \int P(x|\theta_k, G_k) P(\theta_k|G_k, D) d\theta_k \tag{1}$$

Parametric inference is the task of computing the probability of a given model, the corresponding equation can be derived from Bayes' theorem:

$$P(\theta, G|D) = \frac{P(D|\theta, G)P(\theta, G)}{P(D)} \tag{2}$$

$$P(G|D) = \frac{\int P(D|\theta, G)P(\theta, G)d\theta}{P(D)} \tag{3}$$

Since the above equations are intractable in practice, inference is performed using Monte Carlo methods, taking into account only some representative model instances, or, as an extremum, only the most probable one.

The main advantages of Bayesianism over classical statistics can be summarized as follows:

- Information about the parameters are described as a probabilistic distribution over them, hence every statistical inference is a direct probabilistic statement.
- Parameter estimation can be regarded as an inversion task formalized by Bayes' theorem, i.e. inference is based purely on data.
- Prior distribution are capable of representing any kind of knowledge or even the total lack of it.
- Posteriors can be used as priors in the consecutive steps of knowledge updating, and can be considered the representations of the phases of knowledge acquisition.
- Through Bayes' theorem, Bayesian inference combines information in prior knowledge and observation data in a normative manner.
- Using posterior distributions instead of point estimations as results of inference takes into account not only the most probable configuration.

B. Bayesian networks

The most common implementations of the above principles are Bayesian networks. They represent the modeled domain by a directed acyclic graph (DAG), the nodes of which are (discrete) stochastic variables (the entities of the domain); while the edges can be regarded as direct probabilistic dependencies. Besides the structure of the graph there is a conditional dependency model assigned to each variable, describing how the given node depends on its parents, hence a Bayesian network implements the previously mentioned hierarchic model structure. These local dependency models suggest an important property of Bayesian networks: the represented distribution $P(X_1, ..X_n)$ factorizes w.r.t. the DAG G if

$$P(X_1, ..X_n) = \prod_{i=1}^{n} P(X_i|Parents(X_i)). \tag{4}$$

Hence a Bayesian network can be defined as the minimal DAG obeying the above condition, assuming that each of its nodes represent a domain variable. Though this definition concentrates on the independencies of the domain, in practice, Bayesian networks are regarded as pairs $BN(G, \theta)$ of the DAG structure G, and the set of corresponding parameters θ describing the local dependencies.

III. Extensions of Bayesian networks

There are two main trends aiming at unifying the capabilities of first-order logic and Bayesian networks. The one which starts from the basis of logic tries to extend it with probabilistic statements. As described in [1] the two possible semantics introducing probabilities about logical statements are those based on object-frequency and on possible world. In the KBMC methodology [2] Bayesian networks are constructed from a logical knowledge base to support the current query. Stochastic logic programs [3] and Bayesian logic programs [4] stem from inductive logic programming.

The other basic approach, which is closer to ours, is to introduce object-oriented properties in Bayesian networks. The basic idea of object-oriented Bayesian networks (OOBNs) [5] is to build up larger networks from smaller objects, network fragments. These can be constructed separately from each other, so forming a library of reusable elements. Among the defined classes the usual inheritance and containment relations may exist, what further extends the knowledge engineering advantages of the framework. Models of a given situation can then be built up from these smaller fragments.

Probabilistic relational models (PRMs) [6] are very similar to OOBNs, but apart from the differences between their formal definitions, they further extend OOBNs with structural uncertainties. In case of *reference uncertainty* the members of the relations cannot be exactly identified, only a set of them; *existence uncertainty* means inferring about unknown objects, defined by its relations.

Bayesian logic (BLOG) [7] models also resemble the structure of OOBNs and PRMs, although they are defined from the basis of predicate logic. They complete the types of uncertainties with *identity uncertainty*: in the case of this, the objects of the domain do not have a unique identifier attribute, so they can be distinguished from each other only probabilistically.

A. *Desired properties of a knowledge representation language*

Motivated by the above trends we can now enumerate the desired properties of a language describing a knowledge base about a domain, which successfully extend monolithic Bayesian networks towards first-orderedness.

- The possible worlds of the language are the model structures and their parameterizations, that is, only the domain-related elements are probabilistic, the basic pieces of knowledge, like logical axioms and functions are not.
- A distribution over them: according to the principles of Bayesian statistics, any kind of domain knowledge is represented by a distribution over the corresponding domain elements.
- Textual annotations assigned to model elements: these may contain important informal knowledge, and tractable first-orientedness can be achieved by means of these.
- The query language, beyond expressing basic queries about the instantiations of the model variables, must be supplemented with string manipulation functions, in order to make it capable of handling queries which are first-ordered in the referred annotations.
- Evidently, the system must have effective algorithms for inference, and as an application of generic inference methods for the calculation of posteriors over parameters and structure, i.e. for learning.

Furthermore, the model structures and parameterizations must follow the paradigm of object-orientedness: they must support a hierarchic, decomposed description of network modules and inheritance between them. The description also has to be able to handle the structural uncertainties described in Section III..

IV. The implemented framework

A. *Elements of the representation language*

The system implemented so far mainly concentrates on the object-oriented properties of the model. The description language consists of the following elements:

- type: simply speaking, defines the range of the corresponding variable, their task is to ensure that only compatible variables will be connected in the model.
- CPD (conditional probability distribution): defines how a child variable depends on its parents, can be regarded as a probabilistic function mapping the current values of the parental variables, to the distribution of the child. It can be a table containing the conditional probability values in the simplest case, or it can be some more complex function e.g. a decision tree.
- TDM (typed dependency map): similarly to CPDs, describes a probabilistic function from some input types to some output types, however, it has an inner structure, that is, it may consist of multiple CPDs and TDMs, defining a Bayesian network module or class.

Of the above elements, TDMs are the key of modularity: by the inclusion of further TDMs they support a hierarchically decomposed structure description.

B. *The implemented system*

The framework currently consists by two parts: a graphical user interface for editing and visualizing models, developed within the confines of the project Pythia, and an inference engine capable of loading models stored in XML files, and performing inference in them, implemented by the author.

V. Prior definition using graph-grammars

An important factor of the inference equation Eq. 2 is the term $P(G, \theta)$, encoding the prior knowledge and beliefs about the domain. In real-world applications it would be intractable to define this distribution by an exhaustive enumeration of the possible structures, however, the language based on TDMs offers an easy and tractable method for this task. Since the model structure hierarchically decomposed by the TDMs naturally resembles a top-down description, it is a straightforward to define a graph grammar that generates the set of possible structures.

Such a grammar would consist of a set of rewrite rules of the form:

$$C_i \rightarrow A_{i1}(P_{i1})|...|A_{in}(P_{in}), \tag{5}$$

where C_i, the condition part consists of a nonterminal subgraph and the corresponding annotation, which, if matched by a part of the current graph may be replaced by one of A_{ij}-s, with probability P_{ij}. Since the matching of the model parts is based on the textual annotations, this grammar-based model generation procedure may be extended to other elements of the model, e.g. the conditional dependency models. In this way the (possibly unnormalized) prior probability of a model can be defined as the product of the probabilities of the rules applied in its derivation.

For the sake of simplicity, the grammar should be retained at a context free level, so that the grammar-related computations remain tractable, and conflicts between overlapping rule conditions can be avoided.

The grammar-related part of the system has to support the following types of queries:

- parsing: deciding whether a given structure can be derived by the grammar, and the probability of it.
- membership: deciding whether a symbol occurred in the parsing tree of the model.
- parameter learning: finding the set of probabilities assigned to the rules, which maximize the posterior probabilities of the models.

VI. Conclusion

In the paper we have overviewed the recent trends aiming at extending monolithic Bayesian models towards first-order logic. Motivated by these we could enumerate the necessary properties of a language by which we can describe a knowledge base containing uncertain knowledge. We have introduced the basic elements of the language and sketched the future steps of its development.

References

[1] J. Y. Halpern, "An analysis of first-order logics of probability," in *Proceedings of IJCAI-89, 11th International Joint Conference on Artificial Intelligence*, pp. 1375–1381, Detroit, US, 1989.

[2] M. P. W. nad J. S. Breese and R. P. Goldman, "From knowledge bases to decision models," *The Knowledge Engineering Review*, 7(1):35–53, 1992.

[3] S. Muggleton, "Stochastic logic programs," in *Proceedings of the 5th International Workshop on Inductive Logic Programming*, L. De Raedt, Ed., p. 29. Department of Computer Science, Katholieke Universiteit Leuven, 1995.

[4] K. Kersting and L. D. Raedt, "Bayesian logic programs," in *Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming*, J. Cussens and A. Frisch, Eds., pp. 138–155, 2000.

[5] D. Koller and A. Pfeffer, "Object-oriented bayesian networks," in *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pp. 302–313, 1997.

[6] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer, "Learning probabilistic relational models," in *IJCAI*, pp. 1300–1309, 1999.

[7] B. Milch, B. Marthi, and S. Russell, "BLOG: Relational modeling with unknown objects," in *Proceedings of the ICML-04 Workshop on Statistical Relational Learning*, 2004.