Hybrid Logical Bayesian Networks

Irma Ravkic, Jan Ramon, and Jesse Davis

Department of Computer Science
KU Leuven
Celestijnenlaan 200A, B-3001 Heverlee, Belgium
irma.ravkic@cs.kuleuven.be

Abstract. Probabilistic logical models have proven to be very successful at modelling uncertain, complex relational data. Most current models and implementations focus on modelling domains that only have discrete variables. Yet many real-world problems are hybrid and have both discrete and continuous variables. In this paper we focus on the Logical Bayesian Network (LBN) formalism. This paper discusses our work in progress in developing hybrid LBNs, which offer support for continuous variables. We provide a brief sketch for basic parameter learning and inference algorithms for them.

1 Introduction

Real-world problems are hybrid in that they have discrete and continuous variables. Additionally, it is necessary to model the uncertain nature and complex structure inherent in these problems. Most existing formalisms cannot cope with all these challenges. Hybrid Bayesian networks [1] can model uncertainty about both discrete and continuous variables, but not relationships between objects in the domain. On the other hand, probabilistic logical models (PLM) [2–4] can model uncertainty in relational domains, but many formalisms restricted themselves to discrete data. More recently, several approaches have been proposed that augment PLMs in order to model hybrid relational domains. These include Adaptive Bayesian Logic Programs [5], ProbLog with continuous variables [6], and Hybrid Markov Logic Networks [7].

In this paper, we focus on upgrading another PLM framework called Logical Bayesian Networks [8] such that they can model continuous variables. From our perspective, LBNs have several important advantages. One, they clearly distinguish the different components (i.e., the random variables, dependencies among the variables, and the CPDs of each variable) of a relational probabilistic model. Two, the CPDs are easily interpretable by humans, which is not the case in other formalisms, such as those based on Markov random fields. This paper reports on our preliminary work in progress on developing hybrid LBNs. We show how LBNs can naturally represent continuous variables. We also discuss a basic parameter learning algorithm and how Gibbs sampling can be used for inference.

2 Background

We briefly provide background on Logical Bayesian Networks, and Gibbs sampling.

2.1 Logical Bayesian Networks

A propositional Bayesian network (BN) compactly represents a probability distribution over a set of random variables $X = \{X_1, \ldots, X_n\}$. A BN is a directed, acyclic graph that contains a node for each variable $X_i \in X$. Each node in the graph has a conditional probability distribution $\theta_{X_i|Parents(X_i)}$ that gives the probability distribution over the values that a variable can take for each possible setting of its parents. A BN encodes the following probability distribution:

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

Logical Bayesian Networks upgrade propositional BNs to work with relational data [8]. LBNs contain four components: *random variable declarations*, *conditional dependencies*, *Conditional Probability Distributions (CPDs)* and a *logic program*. Semantically, an LBN induces a Bayesian network. Given a set of constants, the first two components of the LBN define the structure of the Bayesian network. The random variable declarations define which random variables appear in the network whereas conditional dependency relationships define the arcs that connect the nodes. Finally, the conditional probability functions determine the conditional probability distribution associated with each node in the network. We will illustrate each of these components using the well-known *university* example [9]. The logical predicates in this problem are `student/1`, `course/1`, and `takes/2`. Random variables start with capital letters and constants with lower-case letters. The logical predicates can then be used to define random variables as follows:

```
random(intelligence(S)):- student(S).
random(difficulty(C)):- course(C).
random(grade(S,C)):- takes(S,C).
random(ranking(S)):- student(S).
```

Conditional dependencies are represented by a set of clauses. The clauses state which variables depend on each other and determine which edges are included in a ground LBN. They take the following form:

```
grade(S,C) | intelligence(S),difficulty(C).
ranking(S) | grade(S,C) :- takes(S,C).
```

A CPD is associated with each conditional dependency in a LBN. In principle, any CPD is possible. However, LBNs typically make use of logical probability trees. A logical probability tree is a binary tree where each internal node contains a logical test (conjunction of literals) and each leaf contains a probability distribution for a particular attribute. Examples are sorted down the tree based on whether they satisfy the logical test at an internal node.

The logic program component contains a set of facts and clauses that describes the background knowledge for a specific problem. It generates the ground Bayesian network. In the university example it may contain facts such as:

```
student(mary).
student(peter).
course(math).
takes(mary,math).
takes(peter,math).
```

The LBN specified in the running example induces a Bayesian network shown in Figure 1.

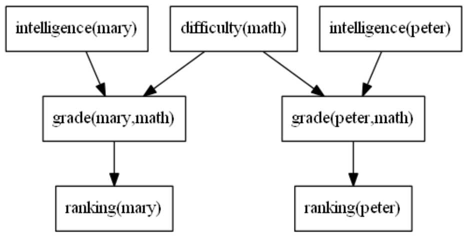

Fig. 1. Bayesian network induced by the simple *university* example

Notice that the logic program defines which random variables appear in the model, i.e., different interpretations of a logical program represent a different set of random variables.

2.2 Gibbs Sampling

Gibbs sampling is an instance of a Markov Chain Monte Carlo (MCMC) algorithm. It estimates a joint probability distribution over a set of random variables by simulating a sequence of draws from the distribution. It is commonly used in practice when joint distributions over variables are not known or are complicated, but local dependency distributions are given and are simple. To sample a value for a particular variable it is sufficient to take into account only its Markov blanket. The time needed for Gibbs sampling to converge to a stationary distribution is dependent on the starting point and therefore in practice some number of examples are ignored (*burn-in* period). For more details see [10].

3 Our Approach

We now describe how we augment LBNs to model continuous variables.

3.1 Representation

It is relatively natural to incorporate continuous random variables in the LBNs. We do so by adding a new random variable declaration that indicates whether a variable is continuous and what its distribution is. For example, we could make the following declaration:

```
randomGaussian(numHours(C)):- course(C).
```

This states that `numHours(C)` is a Gaussian distributed continuous random variable if `C` is a course. Currently, we only allow Gaussian continuous variables, but it is straightforward to incorporate other distributions.
After being declared, continuous random variables can appear in conditional dependency clauses. For example:

```
numHours(C) | difficulty(C).
```

This clause states that the number of hours spent studying for a course `C` depends on the difficulty of the course. Currently, we add a restriction that a discrete random variable cannot have continuous parents. This is a common restriction in hybrid BNs as well.

Logical CPDs can easily accommodate continuous variables by adding a Gaussian distribution in an appropriate leaf as in Figure 2. A Gaussian distribution with mean μ and standard deviation σ is:

$$N(\mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2 / 2\sigma^2} \tag{2}$$

3.2 Parameter Learning and Inference

When learning parameters we assume a known structure, that is, the structure of the probability tree is given. The examples are sets of interpretations where each interpretation refers to a particular instantiation of all random variables. We estimate the maximum likelihood of parameters. In the discrete case, this corresponds to a frequency of a specific variable value in a dataset. In the continuous case, this corresponds to computing the sample mean and standard deviation.

For estimating the mean and standard deviation we used a two-pass algorithm. It first computes the sample mean:

$$\mu = \frac{\sum_{i=1}^{n} y_i}{n} \tag{3}$$

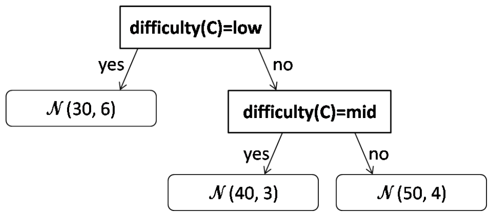

Fig. 2. Logical probability tree for `numHours(C)`

The standard deviation is calculated in the second pass through data by using:

$$\sigma = \frac{\sqrt{\sum_{i=1}^{n}(y_i - \mu)^2}}{n-1} \tag{4}$$

For inference, we have implemented Gibbs sampling which allows us to estimate the posterior probability of some variables given (a possibly empty) set of evidence variables. When querying continuous variables, we can answer several types of queries. We can estimate its mean value. Alternatively, we can estimate the probability that its value falls into some interval (i.e., estimate its cumulative distribution). We sample a continuous variable given its Markov blanket by generating a value from a Gaussian distribution given the appropriate mean and standard deviation coming from the CPD defined by its associated conditional dependency clause. A discrete variable having a continuous node as its child is sampled by using its Markov blanket, and the probability of a continuous child given its parents is computed using Equation (2).

4 Experiments

Currently, we have an implementation that works for small datasets. We have done preliminary experiments using synthetically generated data from the university domain that we have used as a running example in the paper. We augmented the task description with two continuous variables: `numHours/1` and `attendance/1`. The first one represents the number of hours a student spends studying for a particular course, and the second one denotes the number of hours students spent in class. We added two conditional dependency clauses making use of these variables:

```
numHours(C) | difficulty(C).
attendance(C) | satisfaction(S,C):-takes(S,C)
```

The first clause was described in Subsection 3.1 and the second clause states that a student is more likely to attend a class if (s)he enjoys the lectures.

To test the parameter learning, we generated synthetic datasets based of varying size. Unsurprisingly, we found that we could learn accurate estimates of the parameters. In terms of inference, we randomly selected some atoms as queries and some as evidence. On small examples, we were able to check that the Gibbs sampler converged to the correct value after a reasonable number of iterations.

5 Conclusions

In this paper we presented a preliminary work on representation, learning and querying of hybrid logical Bayesian networks. Building on this preliminary work, in the future we will study other conditional probability models (e.g., using Poisson distributions), learning and inference in large-scale networks, and the application of hybrid LBNs in bio-medical applications.

Acknowledgements

This work is supported by the Research Fund K.U.Leuven (OT/11/051),

References

1. Murphy, K.: Inference and learning in hybrid Bayesian networks. University of California, Berkeley, Computer Science Division (1998)
2. Kersting, K., De Raedt, L.: 1 bayesian logic programming: Theory and tool. Statistical Relational Learning (2007) 291
3. De Raedt, L., Kimmig, A., Toivonen, H.: Problog: A probabilistic prolog and its application in link discovery. In: Proceedings of the 20th international joint conference on Artifical intelligence. (2007) 2468–2473
4. Richardson, M., Domingos, P.: Markov logic networks. Machine learning **62**(1) (2006) 107–136
5. Kersting, K., De Raedt, L.: Adaptive bayesian logic programs. Inductive Logic Programming (2001) 104–117
6. Gutmann, B., Jaeger, M., De Raedt, L.: Extending problog with continuous distributions. Inductive Logic Programming (2011) 76–91
7. Wang, J., Domingos, P.: Hybrid markov logic networks. In: Proceedings of the 23rd national conference on Artificial intelligence. Volume 2. (2008) 1106–1111
8. Fierens, D., Blockeel, H., Bruynooghe, M., Ramon, J.: Logical bayesian networks and their relation to other probabilistic logical models. Inductive Logic Programming (2005) 121–135
9. Friedman, N., Getoor, L., Koller, D., Pfeffer, A.: Learning probabilistic relational models. In: International Joint Conference on Artificial Intelligence. Volume 16. (1999) 1300–1309
10. Casella, G., George, E.: Explaining the gibbs sampler. American Statistician (1992) 167–174