

Software Tools for Probabilistic Inductive Logic Programming

Kristian Kersting¹ and Luc De Raedt¹

Institute for Computer Science, Machine Learning Lab, University of Freiburg,
Georges-Koehler-Allee, Building 079, 79112 Freiburg, Germany

1 Probabilistic Inductive Logic Programming

Inductive Logic Programming [6] combines techniques from machine learning with the representation of logic programming. It aims at inducing general rules from specific observations and background knowledge. Few ILP systems are capable of representing or processing probabilities. Consequently, they have problems to handle uncertainty explicitly. On the other hand, traditional probabilistic models such as *Bayesian networks*, *hidden Markov models*, *stochastic context-free grammars*, etc. have a rigid structure and therefore have problems representing a variable number of objects and general relations among objects. Recently, various first-order logical extensions of traditional probabilistic models have been proposed. More over, first approaches for learning such *probabilistic-logical models* (PLMs) have been developed. For more information we refer to the PLMs repository at <http://www.informatik.uni-freiburg.de/~kersting/plmr/>.

We will present software tools for two PLMs: *Bayesian logic programs* and *logical hidden Markov models*. In both tools, we intended to approach the learning problem by extending *Inductive Logic Programming* techniques with *probabilistic reasoning* mechanisms. This also explains the title of the demo proposal, *Software Tools for Probabilistic Inductive Logic Programming*.

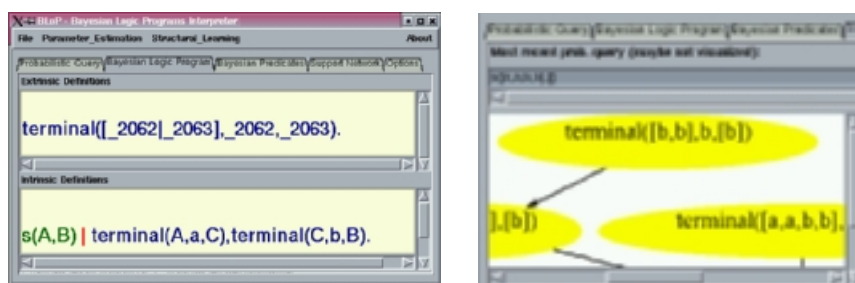
2 BLoP: A System for Bayesian Logic Programs

Bayesian logic programs (BLPs) [2, 3, 1] combine Bayesian networks with definite clause logic. Atoms are considered to represent sets of random variables. Like Bayesian networks, BLPs separate the quantitative from the qualitative aspects of the world. For instance, to represent that "the maternal genetic information $mc(X)$ of a person X is influenced by the maternal $mc(M)$ and paternal $pc(M)$ genetic information of X 's mother M " a *conditional probability table* is associated to a *Bayesian clause* $bt(X) \mid mother(M, X), mc(M), pc(M)$:

$mother(Y, X)$	$mc(Y)$	$pc(Y)$	$P(mc(X))$
<i>true</i>	<i>a</i>	<i>a</i>	(0.98, 0.01, 0.01)
<i>true</i>	<i>b</i>	<i>a</i>	(0.01, 0.98, 0.01)
...
<i>false</i>	<i>a</i>	<i>a</i>	(0.33, 0.33, 0.33)
...

Bayesian logic programs generalize both Bayesian networks as well as logic programs.

We will present our software tool BLOP. It is written in Sicstus Prolog calling a Bayesian network inference engine (such as NETICA, HUGIN, or MATLAB). BLOP supports (1) a graphical user interface written in TCLTK, (2) inference, (3) EM and gradient-based parameter estimation, (3) sampling, and (4) learning of intensional rules. To illustrate BLOP, an example from the field of genetics will be employed. Genetics provide a natural application domain for BLPs. The biological laws of inheritance have a probabilistic nature and require the representation of the relational familial structure of the individuals under study. Example screenshots are:



3 Fulham: A System for Logical Hidden Markov Models

Logical hidden Markov Models (LOHMMs) [4, 5] are an extension of hidden Markov models that allows for *logical sequences*, i.e., sequences of atoms in a first order logic. Many sequences occurring in real-world problems exhibit a natural logical representation. For example for user modelling, UNIX command sequence can be represented by `emacs(ecsqaru.tex)`, `ls`, `latex(ecsqaru.tex)`, ... The main idea of LOHMMs is to use abstract symbols and states instead of specific symbols and states. These abstractions, obtained by summarizing sets of states, are represented as logical atoms. Abstract states, i.e. atoms are connected by abstract transitions such as

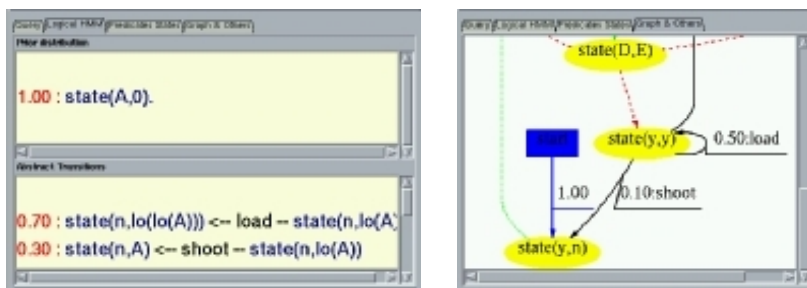
$$\begin{aligned}
 0.8 : \text{latex}(\text{File}, \text{tex}) &\xleftarrow{\text{emacs}(\text{File})} \text{emacs}(\text{File}, \text{tex}) \\
 0.2 : \quad \text{xdvi}(\text{File}) &\xleftarrow{\text{emacs}(\text{File})} \text{emacs}(\text{File}, \text{tex}) .
 \end{aligned}$$

Both transitions specify how to proceed from states matching `emacs(File, tex)`. Consider e.g. `emacs(ecsqaru, tex)`. Indeed, according to the first transition, the probability is 0.8 that the resulting state is `latex(ecsqaru, tex)` and according to the second one it is 0.2 that it is `xdvi(ecsqaru)`. In both cases `emacs(ecsqaru)` will be observed.

There are two important motivations for using logical atoms. First, *variables* in the atoms allow us to make abstraction of specific symbols. E.g., the logical atom `emacs(X, luc)` represents all files that user Luc could edit using Emacs. Second, *unification* allows us to share information among hidden states and between

hidden states and observations. E.g., the sequence `emacs(X, luc), latex(X, luc)` represents that the same file is used as an argument for both Emacs and \LaTeX .

We will present our software tool FULHAM for LOHMMs. FULHAM supports (1) a graphical user interface written in Tcl/Tk, (2) scaled forward and backward inference algorithms, (3) Viterbi algorithm, (4) forward Sampling, and (5) Baum-Welch and gradient-based parameter estimation. We will illustrate FULHAM with examples form user modelling and computational biology. In the latter, LOHMMs can led to more compact models (120 parameters corresponding to an HMM with more than 62000 parameters). Example screenshots are:



Acknowledgements The authors would like to thank Tapani Raiko for his valuable work on FULHAM.

References

1. K. Kersting and L. De Raedt. Adaptive Bayesian Logic Programs. In C. Rouveirol and M. Sebag, editors, *Proceedings of the Eleventh Conference on Inductive Logic Programming (ILP-01)*, volume 2157 of *LNCS*, Strasbourg, France, 2001. Springer.
2. K. Kersting and L. De Raedt. Bayesian logic programs. Technical Report 151, University of Freiburg, Institute for Computer Science, April 2001. (submitted).
3. K. Kersting and L. De Raedt. Towards Combining Inductive Logic Programming and Bayesian Networks. In C. Rouveirol and M. Sebag, editors, *Proceedings of the Eleventh Conference on Inductive Logic Programming (ILP-2001)*, volume 2157 of *LNCS*, Strasbourg, France, 2001. Springer.
4. K. Kersting, T. Raiko, and L. De Raedt. Logical Hidden Markov Models (Extended Abstract). In J. A. Games and A. Salmeron, editors, *Proceedings of the First European Workshop on Probabilistic Graphical Models (PGM-02)*, Cuenca, Spain, November 2002.
5. K. Kersting, T. Raiko, S. Kramer, and L. De Raedt. Towards discovering structural signatures of protein folds based on logical hidden markov models. In R. B. Altman, A. K. Dunker, L. Hunter, T. A. Jung, and T. E. Klein, editors, *Proceedings of the Pacific Symposium on Biocomputing*, pages 192 – 203, Kauai, Hawaii, USA, 2003. World Scientific.
6. S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19(20):629–679, 1994.