Capita Selecta AI 2019-2020

# Assignment: Probabilistic Programming

The goal of this assignment is to apply probabilistic programming technology to perform inference in probabilistic graphical models and ProbLog programs.

- You can work alone or in a team of two students. Do not share results between teams.
- Send a `yourlastnames.zip` file containing (1) a report describing your approach, design choices and results as `yourlastnames-report.pdf` and (2) CNF-files and scripts to repeat the experiments to wannes.meert@cs.kuleuven.be using the **Belnet FileSender service**[1] by Monday April 6, 2020.
- All answers to tasks should be in your report, **not** in the code or other files.
- Grades for this part are fully determined by the submission (no presentation).
- Questions must be asked in the Toledo discussion forum.

## 1 Probabilistic Inference Using Weighted Model Counting (6/10)

One of the most performant techniques to compute the marginal or conditional probability of a query given a probabilistic graphical model (PGM) or probabilistic programs is to reduce the problem to weighted model counting (WMC). This entails that a PGM such as a Bayesian network is represented as a propositional knowledge base in conjunctive normal form (CNF) with weights associated to the propositional variables. You will build a small compiler based on such approach.

Throughout the assignment we will work with the following ProbLog program:

```
person(a). person(b). person(c).
0.2::stress(X) :- person(X).
0.1::friends(X,Y) :- person(X), person(Y).
0.3::smokes(X) :- stress(X).
0.4::smokes(X) :- friends(X,Y), smokes(Y).
query(smokes(a)).
```

### 1.1 SRL to CNF (1/5)

ProbLog, a Statistical Relational Learning formalism, is a generalization of PGM that allows one to express complex relations. Similar to PGMs, probabilistic inference for ProbLog can be reduced to a Weighted Model Counting task. Read about this approach in Fierens et al. [2015] (sections 1-6.1).

Task 1.1.1 Write the encoding for the ProbLog program as logic formula and associated weights. For partial credit you can restrict the friends relationships to `0.1::friends(a,b)` and `0.1::friends(a,c)` to avoid cycles (also for the next tasks).

Task 1.1.2 Show the intermediate steps to translate the given program to a CNF.

### 1.2 SRL to PGM (1/5)

A ProbLog program can be translated to an equivalent Bayesian network. In ProbLog, multiple rules with the same head can be considered as a noisy-OR structure in PGM [Díez and Druzdzel, 2006, Sec 4.1.1]. You can also use the ProbLog tutorial for inspiration.

Task 1.2.1 Show a complete and correct equivalent Bayesian network for the given ProbLog program and query.

[1] `https://filesender.belnet.be`

### 1.3 PGM to CNF (2/5)

Read about this approach in Chavira and Darwiche [2008] (sections 1-3, optionally also 4) and familiarize yourself with ENC1 (the encoding of Chavira and Darwiche) and ENC2 (the encoding of Sang, Beam and Kautz).

Task 1.3.1 Write the logic encoding for the Bayesian network (with full CPTs) from Task 1.2.1 using ENC1.

Task 1.3.2 Write the logic encoding for the Bayesian network (with full CPTs) from Task 1.2.1 using ENC2.

Task 1.3.3 Can you come up with a more compact variant of either ENC1 or ENC2 by exploiting the knowledge that the Bayesian network expresses noisy-OR structures?

### 1.4 Weighted Model Counting (2/5)

WMC can be performed by applying a search algorithm on the CNF or by compiling the CNF into a structure on which WMC can be performed in polynomial time with respect to the size of the structure. An advantage of the standardization to CNF is that multiple model counters can be applied, each with their own advantages and disadvantages. An exhaustive overview of exact model counters is available on `http://beyondnp.org`.

Task 1.4.1 Use the SDD package[2] and one other exact weighted model counter, and apply them to the four logic encodings from the previous tasks (for the given query). Compute and report the WMC (at least 4 digits behind the decimal point). Can you interpret the WMC as probabilities?

Task 1.4.2 What is the smallest circuit for each model you found and using which hyperparameters?

Task 1.4.3 Use WMC to compute the probabilities $P(smokes(a))$ and $P(smokes(a)|friends(a,b) = \top, friends(a,c) = \top)$. Explain how you perform this computation and show the results.

Task 1.4.4 Explain in one paragraph the main theoretical differences between the two weighted model counters.

## 2 Lifted Inference (2/10)

In this part, you are tasked with applying lifted inference concepts to the models discussed before. Use the rules introduced in the lecture slides. You can also use the approach by Van den Broeck et al. [2014].

Task 2.1 Write down the formula to compute the probability for query $P(smokes(a))$ using the lifted inference rules for probabilistic databases as seen in the lecture.

Task 2.2 Can you apply lifted inference techniques from probabilistic databases to construct alternative encodings for the ProbLog program (that can be expressed as a CNF). Show the resulting circuit size and compare to the previous circuit sizes.

## 3 Parameter learning (2/10)

One of the key tasks in machine learning is learning the parameters that fit a given dataset. In this task you implement your own Expectation-Maximisation learning algorithm based on Gutmann et al. [2011]. As this is a rather technical paper it is not required to read this paper and the relevant formula (Eq. 3) is repeated here in a simpler form:

$$\widehat{p_n} = \frac{1}{M} \sum_{m=1}^{M} P(f_n \mid I_m) \tag{1}$$

where M is the number of interpretations or examples and $I_m$ is the $m$-th interpretation[3]. The parameter $p_n$ is the probability for a probabilistic fact $p_n :: f_n$. Repeat applying this formula to all $p_n$ until the values converge or a maximum number of iterations is reached.

For the following tasks you use this variant of the ProbLog program and adapt your encoding from section 1.1:

[2] `http://reasoning.cs.ucla.edu/sdd/`

[3] For examples, see `https://dtai.cs.kuleuven.be/problog/tutorial/learning/01_bayes.html`

```
person(a). person(b). person(c).
0.2::stress(X) :- person(X).
t(_)::friends(a,b).
t(_)::friends(a,c).
t(_)::friends(b,a).
t(_)::friends(b,c).
t(_)::friends(c,a).
t(_)::friends(c,b).
1.0::smokes(X) :- stress(X).
1.0::smokes(X) :- friends(X,Y), smokes(Y).
query(stress(a)).
query(stress(b)).
query(stress(c)).
query(smokes(a)).
query(smokes(b)).
query(smokes(c)).
```

Task 3.1 Included with the assignment is the file `data.pl` with 1000 examples. Use these examples and your encoding together with the above formula to estimate parameters $p_n$ to learn who is friends with who. Explain your algorithm and show the learned parameters.

Task 3.2 Repeat the previous task with the first 100 examples. Show the results and explain the difference with the previous task.

## References


Daan Fierens, Guy Van den Broeck, Ingo Thon, Bernd Gutmann, and Luc De Raedt. Inference in probabilistic logic programs using weighted CNFs. *Theory and Practice of Logic Programming*, 15, 2015.

Francisco Javier Díez and M.J. Druzdzel. Canonical probabilistic models for knowledge engineering. Technical report cisiad-06-01, UNED, Madrid, Spain, 2006.

Mark Chavira and Adnan Darwiche. On probabilistic inference by weighted model counting. *Artificial Intelligence*, 172(6): 772–799, 2008.

Guy Van den Broeck, Wannes Meert, and Adnan Darwiche. Skolemization for weighted first-order model counting. In *Proceedings of the 14th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, pages 1–10, 2014.

Bernd Gutmann, Ingo Thon, and Luc De Raedt. Learning the parameters of probabilistic logic programs from interpretations. *European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD)*, pages 581–596, 2011.