MCINTYRE: A Monte Carlo Algorithm for Probabilistic Logic Programming

Fabrizio Riguzzi

ENDIF – University of Ferrara, Italy
fabrizio.riguzzi@unife.it
Probabilistic Logic Languages

- Combine logic and probability
- Logic Programming: Distribution Semantics [Sato, 1995]
- A probabilistic logic program defines a probability distribution over normal logic programs (called instances or possible worlds or simply worlds)
- The distribution is extended to a joint distribution over worlds and a query
- The probability of a query is obtained from this distribution
Probabilistic Logic Programming (PLP) Languages under the Distribution Semantics

- Probabilistic Logic Programs [Dantsin, 1991]
- Probabilistic Horn Abduction [Poole, 1993], Independent Choice Logic (ICL) [Poole, 1997]
- PRISM [Sato, 1995]
- Logic Programs with Annotated Disjunctions (LPADs) [Vennekens et al., 2004]
- ProbLog [De Raedt et al., 2007]
Logic Programs with Annotated Disjunctions Example

\[ C_1 = \text{epidemic} : 0.6; \text{pandemic} : 0.3 : \neg \text{flu}(X), \text{cold}. \]

\[ C_2 = \text{cold} : 0.7. \]

\[ C_3 = \text{flu}(\text{david}). \]

\[ C_4 = \text{flu}(\text{robert}). \]

- Distributions over the head of rules
- The clause contains implicitly an extra head *null* with probability 0.1 that does not appear in the body of any rule
- Worlds obtained by selecting one atom from the head of every grounding of each clause
- 18 worlds in this example
The query \textit{epidemic} is true in this world, while \textit{pandemic} is false.
ProbLog Example

The ProbLog program equivalent to the example LPAD is

\[
\begin{align*}
C_{11} &= \text{epidemic} : \neg \text{flu}(X), \text{cold}, f1(X). \\
C_{12} &= \text{pandemic} : \neg \text{flu}(X), \text{cold}, \text{problog\_not}(f1(X)), f2(X). \\
C_{13} &= 0.6 :: f1(X). \\
C_{14} &= 0.75 :: f2(X). \\
C_{21} &= \text{cold} : \neg f3. \\
C_{22} &= 0.7 :: f3. \\
C_3 &= \text{flu}(david). \\
C_4 &= \text{flu}(robert).
\end{align*}
\]

- Distributions over facts
- Worlds obtained by selecting or not every grounding of each probabilistic fact
- 32 worlds in this example
**Distribution Semantics**

- **Case of no function symbols:** finite Herbrand universe, finite set of groundings of each clause
- **Atomic choice:** selection of the \(i\)-th atom for grounding \(C\theta\) of clause \(C\)
  - represented with the triple \((C, \theta, i)\)
- **Composite choice** \(\kappa\): consistent set of atomic choices
  - \(\kappa = \{(C_1, \{X/david\}, 1), (C_1, \{X/david\}, 2)\}\) not consistent
- The probability of composite choice \(\kappa\) is
  \[
P(\kappa) = \prod_{(C, \theta, i) \in \kappa} P_0(C, i)
  \]
### Distribution Semantics

- **Selection** $\sigma$: a total composite choice (one atomic choice for every grounding of each clause)

  \[ \sigma = \{(C_1, \{X/\text{david}\}, 1), (C_1, \{\text{robert}\}, 1), (C_2, \{\}, 1)\} \]

- A selection $\sigma$ identifies a logic program $w_\sigma$ called world

- The probability of $w_\sigma$ is $P(w_\sigma) = P(\sigma) = \prod_{(C, \theta, i) \in \sigma} P_0(C, i)$

- Finite set of worlds: $W_T = \{w_1, \ldots, w_m\}$

- $P(w)$ distribution over worlds: $\sum_{w \in W_T} P(w) = 1$

- Query $Q$: $P(Q|w) = 1$ if $Q$ is true in $w$ and 0 otherwise

- $P(Q) = \sum_w P(Q, w) = \sum_w P(Q|w)P(w) = \sum_{w|\models Q} P(w)$
## Inference

- **Exact inference**
  - *Finding explanations* for the query and then making them mutually exclusive by means of BDDs [De Raedt et al., 2007, Riguzzi, 2009, Riguzzi and Swift, 2010].
  - #P-complete [Valiant, 1979]

- **Approximate inference:**
  - *k*-best [Kimmig et al., 2011, Bragaglia and Riguzzi, 2011]: compute a lower bound by finding only the *k* most probable explanations for a query and then builds a BDD from them.
  - Bounded approximation
    [Kimmig et al., 2011, Bragaglia and Riguzzi, 2011]: compute a lower bound and an upper bound of the probability of the query by using iterative deepening.
  - Monte Carlo [Kimmig et al., 2011, Bragaglia and Riguzzi, 2011]: sample the worlds and tests the query in the samples.
Monte Carlo

- Idea: sample a world, test the query and update counters
- The fraction of worlds where the query is true is the probability of the query
- Problem: worlds are obtained from a grounding of the program which has an exponential size
- Solution: on demand sampling, sample only the clauses that are involved in a branch of the SLDNF tree for the goal
- Samples must be consistent, i.e., the same alternative must be sampled from a grounding of a clause
Monte Carlo

- **ProbLog algorithm** [Kimmig et al., 2011]
  - Source to source transformation, the probabilistic facts are turned into normal clauses that update global structures
  - Ground probabilistic facts: an array with an element for each fact that stores sampled true, sampled false or not yet sampled
  - When a probabilistic fact is called, if it has not been sampled then it is sampled and stored in the array.
  - Non-ground probabilistic facts: samples for groundings are stored in the internal database of Yap

- **cplint algorithm** [Bragaglia and Riguzzi, 2011]:
  - Meta-interpretation: two arguments of the meta-interpreter predicate are used, one for keeping the input set of choices and one for the output set of choices
MCINTYRE: “Monte Carlo INference wiTh Yap REcord”
Source to source transformation
The disjunctive clause

\[ C_i = h_{i1} : \Pi_{i1} \lor \ldots \lor h_{in} : \Pi_{in} : \neg b_{i1}, \ldots, b_{im_i}. \]

where the parameters sum to 1, is transformed into the set of clauses \( MC(C_i) \):

\[
MC(C_i, 1) = \begin{align*}
h_{i1} : \neg b_{i1}, \ldots, b_{im_i}, \\
\text{sample\_head}(ParList, i, VC, NH), NH = 1.
\end{align*}
\]

\[
\ldots
\]

\[
MC(C_i, n_i) = \begin{align*}
h_{in} : \neg b_{i1}, \ldots, b_{im_i}, \\
\text{sample\_head}(ParList, i, VC, NH), NH = n_i.
\end{align*}
\]

where \( VC \) is a list containing each variable appearing in \( C_i \) and \( ParList \) is \([\Pi_{i1}, \ldots, \Pi_{in_i}]\).
If the parameters do not sum up to 1 the last clause (the one for \textit{null}) is omitted.

Basically, we create a clause for each head and we sample a head index at the end of the body with \texttt{sample\_head/4}.

If this index coincides with the head index, the derivation succeeds, otherwise it fails.

For example, clause $C_1$ of epidemic example becomes

$MC(C_1, 1) = \textit{epidemic} : \neg \textit{flu}(X), \textit{cold},$

$\texttt{sample\_head}([0.6, 0.3, 0.1], 1, [X], NH), \; NH = 1.$

$MC(C_1, 2) = \textit{pandemic} : \neg \textit{flu}(X), \textit{cold},$

$\texttt{sample\_head}([0.6, 0.3, 0.1], 1, [X], NH), \; NH = 2.$
sample_head/4 samples an index from the head of a clause and uses the builtin Yap predicates recorded/3 and recorda/3 for retrieving or adding an entry to the internal database.

sample_head/4 is at the end of the body

Range restricted programs: all the variables appearing in the head also appear in positive literals in the body

When calling sample_head/4 all the variables of the clause have been grounded.
sample_head(_ParList,R,VC,NH):-
  recorded(exp,(R,VC,NH),_),!.
sample_head(ParList,R,VC,NH):-
  sample(ParList,NH),
  recorda(exp,(R,VC,NH),_).

sample(ParList, HeadId) :-
  random(Prob),
  sample(ParList, 0, 0, Prob, HeadId).

sample([HeadProb|Tail], Index, Prev, Prob, HeadId) :-
  Succ is Index + 1,
  Next is Prev + HeadProb,
  (Prob =< Next ->
   HeadId = Index
  ;
   sample(Tail, Succ, Next, Prob, HeadId)
 ).
Tabling can be effectively used to avoid re-sampling the same atom.

To take a sample from the program we use the following predicate:

```prolog
sample(Goal):-
    abolish_all_tables,
    eraseall(exp),
    call(Goal).
```
A fixed number of samples \( n \) is taken and the fraction \( \hat{p} \) of samples in which the query succeeds is computed.

Confidence interval of \( \hat{p} \): given by the central limit theorem to approximate the binomial distribution with a normal distribution.

The 95% binomial proportion confidence interval is

\[
\hat{p} \pm z_{1-\alpha/2} \sqrt{\frac{\hat{p}(1-\hat{p})}{n}}
\]

where \( z_{1-\alpha/2} \) is the \( 1 - \alpha / 2 \) percentile of a standard normal distribution (\( \alpha = 0.05 \)).

If the width of the interval is below \( \delta \), MCINTYRE stops and returns \( \hat{p} \)

This estimate of the interval is good for a sample size larger than 30 and if \( \hat{p} \) is not too close to 0 or 1.

Empirically, the normal approximation works well as long as \( n\hat{p} > 5 \) and \( n(1 - \hat{p}) > 5 \).
Biomine Network

- Biomine network: network of biological concepts
- Each edge has a probability
- Dataset from [De Raedt et al., 2007]: 50 sampled subnetworks of size 200, 400, ..., 10000 edges
- Sampling repeated 10 times
- Linux PCs with Intel Core 2 Duo E6550 (2,333 MHz) and 4 GB of RAM
- Execution stopped after 24 hours

```prolog
path(X, X).
predicte(X, Y) :- X \neq Y, path(X, Z), arc(Z, Y).
arc(X, Y) :- edge(Y, X).
arc(X, Y) :- edge(X, Y).
edge('EntrezProtein_33339674', 'HGNC_620'): 0.515062.
...
```

- `path/2` tabled
Biomine Network

Solved graphs

Average execution times
Growng Head

- From [Meert et al., 2010]: propositional programs in which the head of clauses are of increasing size
- The program for size 4 is

\[
\begin{align*}
  a_0 & : \leftarrow a_1. \\
  a_1 & : 0.5. \\
  a_0 & : 0.5; a_1 : 0.5 \leftarrow a_2. \\
  a_2 & : 0.5. \\
  a_0 & : 0.33333; a_1 : 0.33333; a_2 : 0.33333 \leftarrow a_3. \\
  a_3 & : 0.5.
\end{align*}
\]

- No predicate is tabled
Sampling last

Sampling first

MCINTYRE
ProbLog
cplint
PITA

N
Time (s)
Bloodtype

From [Meert et al., 2010]: determining the blood type of a person on the basis of her chromosomes that in turn depend on those of her parents.

\[
\begin{align*}
\text{bloodtype(Person,a)}: & 0.90; \quad \text{bloodtype(Person,b)}: 0.03; \\
\text{bloodtype(Person,ab)}: & 0.03; \quad \text{bloodtype(Person,null)}: 0.04 :- \\
& \quad \text{pchrom(Person,a)}, \text{mchrom(Person,a)}. \\
\end{align*}
\]

\[
\begin{align*}
\text{mchrom(Person,a)}: & 0.90; \quad \text{mchrom(Person,b)}: 0.05; \\
\text{mchrom(Person,null)}: & 0.05 :- \\
& \quad \text{mother(Mother,Person)}, \text{pchrom(Mother,a)}, \text{mchrom(Mother,a)}. \\
\end{align*}
\]

\[
\begin{align*}
\text{mchrom(p,a)}: & 0.3; \quad \text{mchrom(p,b)}: 0.3; \quad \text{mchrom(p,null)}: 0.4. \\
\text{pchrom(p,a)}: & 0.3; \quad \text{pchrom(p,b)}: 0.3; \quad \text{pchrom(p,null)}: 0.4. \\
\end{align*}
\]

All the predicates are tabled.
Probabilistic Logic Languages

Bloodtype

![Graph showing time (s) vs. N for different tools: MCINTYRE, ProbLog, cplint, and PITA. The x-axis represents N ranging from 0 to 200, while the y-axis represents time in seconds on a logarithmic scale.]
From [Meert et al., 2010]: the clauses have bodies of increasing size. The program for size 4 is

\[
\begin{align*}
\text{a0:0.5} & : - \text{ a1.} \\
\text{a0:0.5} & : - \neg \text{ a1, a2.} \\
\text{a0:0.5} & : - \neg \text{ a1, \neg a2, a3.} \\
\text{a1:0.5} & : - \text{ a2.} \\
\text{a1:0.5} & : - \neg \text{ a2, a3.} \\
\text{a2:0.5} & : - \text{ a3.} \\
\text{a3:0.5} & .
\end{align*}
\]

No predicate is tabled
Growing body

All algorithms

Only Monte Carlo algorithms
From [Meert et al., 2010]: university domain with predicates such as taught_by/2, advised_by/2, course_level/2, phase/2, position/2, student/1 and others.

Programs of increasing size by considering an increasing number of students.

For both MCINTYRE and ProbLog all the predicates are tabled.
hidden Markov Model

hmm(O) :- hmm1(_,O).
hmm1(S,O) :- hmm(q1,[],S,O).
hmm(end,S,[]).
hmm(Q,S0,S,[L|O]) :- Q\neq end, next_state(Q,Q1,S0), letter(Q,L,S0), hmm(Q1,[Q|S0],S,O).
next_state(q1,q1,_)S) : 1/3;
next_state(q1,q2,_)S) : 1/3;
next_state(q1,end,_)S) : 1/3.
next_state(q2,q1,_)S) : 1/3;
next_state(q2,q2,_)S) : 1/3;
next_state(q2,end,_)S) : 1/3.
letter(q1,a,_)S) : 0.25
letter(q1,c,_)S) : 0.25
letter(q1,g,_)S) : 0.25
letter(q1,t,_)S) : 0.25
letter(q2,a,_)S) : 0.25
letter(q2,c,_)S) : 0.25
letter(q2,g,_)S) : 0.25
letter(q2,t,_)S) : 0.25.
Conclusions

- Probabilistic Logic Programming
- Distribution semantics
- Logic Programs with Annotated Disjunctions, ProbLog
- Approximate inference
- MCINTYRE: “Monte Carlo INference wiTh Yap REcord”
- Fast alternative to ProbLog

Thank you!

Questions?
References I


References II


References III


