Probabilistic Logic Languages

Fabrizio Riguzzi

UNIVERSITÀ DEGLI STUDI DI FERRARA
- EX LABORE FRUCTUS -

Outline

Combining Logic and Probability

- Useful to model domains with complex and uncertain relationships among entities
- Many approaches proposed in the areas of Logic Programming, Uncertainty in AI, Machine Learning, Databases
- 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 queries
- 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]
- They differ in the way they define the distribution over logic programs

Independent Choice Logic

$$
\begin{array}{l}
sneezing(X) \leftarrow flu(X), flu\_sneezing(X).\\
sneezing(X) \leftarrow hay\_fever(X), hay\_fever\_sneezing(X).\\
flu(bob).\\
hay\_fever(bob).\\
\\
disjoint([flu\_sneezing(X) : 0.7, null : 0.3]).\\
disjoint([hay\_fever\_sneezing(X) : 0.8, null : 0.2]).
\end{array}
$$

- Distributions over facts by means of disjoint statements
- *null* does not appear in the body of any rule
- Worlds obtained by selecting one atom from every grounding of each disjoint statement

Independent Choice Logic

- 4 worlds

$$\begin{array}{l} sneezing(X) \leftarrow flu(X), flu\_sneezing(X). \\ sneezing(X) \leftarrow hay\_fever(X), hay\_fever\_sneezing(X). \\ flu(bob). \\ hay\_fever(bob). \end{array}$$

$$\begin{array}{ll} flu\_sneezing(bob). & null. \\ hay\_fever\_sneezing(bob). & hay\_fever\_sneezing(bob). \\ P(w_1) = 0.7 \times 0.8 & P(w_2) = 0.3 \times 0.8 \\ & \\ flu\_sneezing(bob). & null. \\ null. & null. \\ P(w_3) = 0.7 \times 0.2 & P(w_4) = 0.3 \times 0.2 \end{array}$$

- $sneezing(bob)$ is true in 3 worlds
- $P(sneezing(bob)) = 0.7 \times 0.8 + 0.3 \times 0.8 + 0.7 \times 0.2 = 0.94$

PRISM

$$\begin{array}{l} sneezing(X) \leftarrow flu(X), msw(flu\_sneezing(X), 1). \\ sneezing(X) \leftarrow hay\_fever(X), msw(hay\_fever\_sneezing(X), 1). \\ flu(bob). \\ hay\_fever(bob). \end{array}$$

$$\begin{array}{l} values(flu\_sneezing(\_X), [1, 0]). \\ values(hay\_fever\_sneezing(\_X), [1, 0]). \\ : -set\_sw(flu\_sneezing(\_X), [0.7, 0.3]). \\ : -set\_sw(hay\_fever\_sneezing(\_X), [0.8, 0.2]). \end{array}$$

- Distributions over *msw* facts (random switches)
- Worlds obtained by selecting one value for every grounding of each *msw* statement

Logic Programs with Annotated Disjunctions

$$
\begin{array}{l}
sneezing(X) : 0.7 \vee null : 0.3 \leftarrow flu(X). \\
sneezing(X) : 0.8 \vee null : 0.2 \leftarrow hay\_fever(X). \\
flu(bob). \\
hay\_fever(bob).
\end{array}
$$

- Distributions over the head of rules
- *null* does not appear in the body of any rule
- Worlds obtained by selecting one atom from the head of every grounding of each clause

ProbLog

$$
\begin{array}{l}
sneezing(X) \leftarrow flu(X), flu\_sneezing(X). \\
sneezing(X) \leftarrow hay\_fever(X), hay\_fever\_sneezing(X). \\
flu(bob). \\
hay\_fever(bob). \\
0.7 :: flu\_sneezing(X). \\
0.8 :: hay\_fever\_sneezing(X).
\end{array}
$$

- Distributions over facts
- Worlds obtained by selecting or not every grounding of each probabilistic fact

Distribution Semantics

- Case of no function symbols: finite Herbrand universe, finite set of groundings of each disjoint statement/switch/clause
- Atomic choice: selection of the i-th atom for grounding $C\theta$ of disjoint statement/switch/clause C
  - represented with the triple (C, θ, i)
  - a ProbLog fact $p :: F$ is interpreted as $F : p \vee null : 1 - p$.
- Example $C_1 = disjoint([flu\_sneezing(X) : 0.7, null : 0.3])$, $(C_1, \{X/bob\}, 1)$
- Composite choice κ: consistent set of atomic choices
- $\kappa = \{(C_1, \{X/bob\}, 1), (C_1, \{X/bob\}, 2)\}$ not consistent
- The probability of composite choice κ is

$$P(\kappa) = \prod_{(C,\theta,i)\in\kappa} P_0(C, i)$$

Distribution Semantics

- Selection σ: a total composite choice (one atomic choice for every grounding of each disjoint statement/clause)
- $\sigma = \{(C_1, \{X/bob\}, 1), (C_2, \{X/bob\}, 1)\}$

$$C_1 = disjoint([flu\_sneezing(X) : 0.7, null : 0.3]).$$
$$C_2 = disjoint([hay\_fever\_sneezing(X) : 0.8, null : 0.2]).$$

- A selection σ identifies a logic program w_σ called world
- The probability of w_σ is $P(w_\sigma) = P(\sigma) = \prod_{(C,\theta,i)\in\sigma} P_0(C, i)$
- Finite set of wrolds: $W_T = \{w_1, \ldots, w_m\}$
- $P(w)$ distribution over worlds: $\sum_{w\in W_T} P(w) = 1$

Distribution Semantics

- Herbrand base $H_T = \{A_1, \ldots, A_n\}$
- $P(a_j|w) = 1$ if A_j is true in w and 0 otherwise
- $P(a_j) = \sum_w P(a_j, w) = \sum_w P(a_j|w)P(w) = \sum_{w \models A_j} P(w)$

Example Program (ICL)

- 4 worlds

$$sneezing(X) \leftarrow flu(X), flu\_sneezing(X).$$
$$sneezing(X) \leftarrow hay\_fever(X), hay\_fever\_sneezing(X).$$
$$flu(bob).$$
$$hay\_fever(bob).$$

$flu\_sneezing(bob).$	$null.$
$hay\_fever\_sneezing(bob).$	$hay\_fever\_sneezing(bob).$
$P(w_1) = 0.7 \times 0.8$	$P(w_2) = 0.3 \times 0.8$

$flu\_sneezing(bob).$	$null.$
$null.$	$null.$
$P(w_3) = 0.7 \times 0.2$	$P(w_4) = 0.3 \times 0.2$

- $sneezing(bob)$ is true in 3 worlds
- $P(sneezing(bob)) = 0.7 \times 0.8 + 0.3 \times 0.8 + 0.7 \times 0.2 = 0.94$

Example Program (LPAD)

- 4 worlds

$sneezing(bob) \leftarrow flu(bob).$
$sneezing(bob) \leftarrow hay\_fever(bob).$
$flu(bob).$
$hay\_fever(bob).$
$P(w_1) = 0.7 \times 0.8$

$null \leftarrow flu(bob).$
$sneezing(bob) \leftarrow hay\_fever(bob).$
$flu(bob).$
$hay\_fever(bob).$
$P(w_2) = 0.3 \times 0.8$

$sneezing(bob) \leftarrow flu(bob).$
$null \leftarrow hay\_fever(bob).$
$flu(bob).$
$hay\_fever(bob).$
$P(w_3) = 0.7 \times 0.2$

$null \leftarrow flu(bob).$
$null \leftarrow hay\_fever(bob).$
$flu(bob).$
$hay\_fever(bob).$
$P(w_4) = 0.3 \times 0.2$

- $sneezing(bob)$ is true in 3 worlds
- $P(sneezing(bob)) = 0.7 \times 0.8 + 0.3 \times 0.8 + 0.7 \times 0.2 = 0.94$

Example Program (ProbLog)

- 4 worlds

$$
\begin{array}{l}
sneezing(X) \leftarrow flu(X), flu\_sneezing(X).\\
sneezing(X) \leftarrow hay\_fever(X), hay\_fever\_sneezing(X).\\
flu(bob).\\
hay\_fever(bob).
\end{array}
$$

$$
\begin{array}{ll}
flu\_sneezing(bob). & \\
hay\_fever\_sneezing(bob). & hay\_fever\_sneezing(bob).\\
P(w_1) = 0.7 \times 0.8 & P(w_2) = 0.3 \times 0.8\\
 & \\
flu\_sneezing(bob). & \\
 & \\
P(w_3) = 0.7 \times 0.2 & P(w_4) = 0.3 \times 0.2
\end{array}
$$

- $sneezing(bob)$ is true in 3 worlds
- $P(sneezing(bob)) = 0.7 \times 0.8 + 0.3 \times 0.8 + 0.7 \times 0.2 = 0.94$

Examples

Throwing coins

```
heads(Coin):1/2 ; tails(Coin):1/2 :-
  toss(Coin),\+biased(Coin).
heads(Coin):0.6 ; tails(Coin):0.4 :-
  toss(Coin),biased(Coin).
fair(Coin):0.9 ; biased(Coin):0.1.
toss(coin).
```

Russian roulette with two guns

```
death:1/6 :- pull_trigger(left_gun).
death:1/6 :- pull_trigger(right_gun).
pull_trigger(left_gun).
pull_trigger(right_gun).
```

Examples

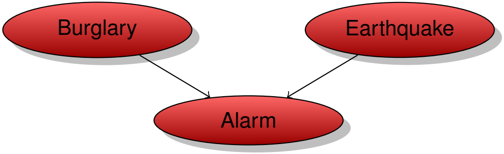
Mendel's inheritance rules for pea plants

```
color(X,purple):-cg(X,_A,p).
color(X,white):-cg(X,1,w),cg(X,2,w).
cg(X,1,A):0.5 ; cg(X,1,B):0.5 :-
  mother(Y,X),cg(Y,1,A),cg(Y,2,B).
cg(X,2,A):0.5 ; cg(X,2,B):0.5 :-
  father(Y,X),cg(Y,1,A),cg(Y,2,B).
```

Probability of paths

```
path(X,X).
path(X,Y):-path(X,Z),edge(Z,Y).
edge(a,b):0.3.
edge(b,c):0.2.
edge(a,c):0.6.
```

Encoding Bayesian Networks



burg	t	f
	0.1	0.9

earthq	t	f
	0.2	0.8

alarm	t	f
b=t,e=t	1.0	0.0
b=t,e=f	0.8	0.2
b=f,e=t	0.8	0.2
b=f,e=f	0.1	0.9

```
burg(t):0.1 ; burg(f):0.9.
earthq(t):0.2 ; earthq(f):0.8.
alarm(t):-burg(t),earthq(t).
alarm(t):0.8 ; alarm(f):0.2:-burg(t),earthq(f).
alarm(t):0.8 ; alarm(f):0.2:-burg(f),earthq(t).
alarm(t):0.1 ; alarm(f):0.9:-burg(f),earthq(f).
```

Try it yourself

- Go to `http://cplint.lamping.unife.it/`

Expressive Power

- All these languages have the same expressive power
- LPADs have the most general syntax
- There are transformations that can convert each one into the others
- ICL, PRISM: direct mapping
- ICL, PRISM to LPAD: direct mapping

LPADs to ICL

- Clause C_i with variables $\overline{X}$

$$H_1 : p_1 \vee \ldots \vee H_n : p_n \leftarrow B.$$

is translated into

$$\begin{array}{l} H_1 \leftarrow B, choice_{i,1}(\overline{X}). \\ \vdots \\ H_n \leftarrow B, choice_{i,n}(\overline{X}). \end{array}$$

$$disjoint([choice_{i,1}(\overline{X}) : p_1, \ldots, choice_{i,n}(\overline{X}) : p_n]).$$

LPADs to ProbLog

- Clause C_i with variables $\overline{X}$

$$H_1 : p_1 \vee \ldots \vee H_n : p_n \leftarrow B.$$

is translated into

$$\begin{array}{l} H_1 \leftarrow B, f_{i,1}(\overline{X}). \\ H_2 \leftarrow B, not(f_{i,1}(\overline{X})), f_{i,2}(\overline{X}). \\ \vdots \\ H_n \leftarrow B, not(f_{i,1}(\overline{X})), \ldots, not(f_{i,n-1}(\overline{X})). \\ \\ \pi_1 :: f_{i,1}(\overline{X}). \\ \vdots \\ \pi_{n-1} :: f_{i,n-1}(\overline{X}). \end{array}$$

where $\pi_1 = p_1$, $\pi_2 = \frac{p_2}{1-\pi_1}$, $\pi_3 = \frac{p_3}{(1-\pi_1)(1-\pi_2)}, \ldots$

- In general $\pi_i = \frac{p_i}{\prod_{j=1}^{i-1}(1-\pi_j)}$

Conversion to Bayesian Networks

- PLP can be converted to Bayesian networks
- Conversion for an LPAD T
- For each atom A in H_T a binary variable A
- For each clause C_i in the grounding of T

$$H_1 : p_1 \vee \ldots \vee H_n : p_n \leftarrow B_1, \ldots B_m, \neg C_1, \ldots, \neg C_l$$

  a variable CH_i with $B_1, \ldots, B_m, C_1, \ldots, C_l$ as parents and H_1, ..., H_n and *null* as values
- The CPT of CH_i is

	$\ldots$	$B_1 = 1, \ldots, B_m = 1, C_1 = 0, \ldots, C_l = 0$	$\ldots$
$CH_i = H_1$	0.0	p_1	0.0
$\ldots$			
$CH_i = H_n$	0.0	p_n	0.0
$CH_i = null$	1.0	$1 - \sum_{i=1}^{n} p_i$	1.0

Conversion to Bayesian Networks

- Each variable A corresponding to atom A has as parents all the variables CH_i of clauses C_i that have A in the head.
- The CPT for A is:

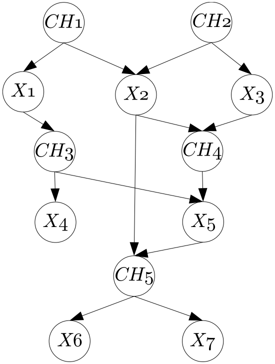
	at least one parent equal to A	remaining columns
$A = 1$	1.0	0.0
$A = 0$	0.0	1.0

Conversion to Bayesian Networks

$$
\begin{aligned}
C_1 &= x1 : 0.4 \vee x2 : 0.6. \\
C_2 &= x2 : 0.1 \vee x3 : 0.9. \\
C_3 &= x4 : 0.6 \vee x5 : 0.4 \leftarrow x1. \\
C_4 &= x5 : 0.4 \leftarrow x2, x3. \\
C_5 &= x6 : 0.3 \vee x7 : 0.2 \leftarrow x2, x5.
\end{aligned}
$$

CH_1, CH_2	$x1, x2$	$x1, x3$	$x2, x2$	$x2, x3$
$x2 = 1$	1.0	0.0	1.0	1.0
$x2 = 0$	0.0	1.0	0.0	0.0

$x2, x5$	t,t	t,f	f,t	f,f
$CH_5 = x6$	0.3	0.0	0.0	0.0
$CH_5 = x7$	0.2	0.0	0.0	0.0
$CH_5 = null$	0.5	1.0	1.0	1.0



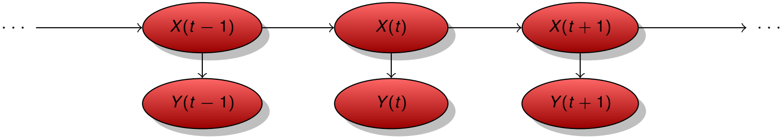
Function Symbols

- What if function symbols are present?
- Infinite, countable Herbrand universe
- Infinite, countable Herbrand base
- Infinite, countable grounding of the program T
- Uncountable W_T
- Each world infinite, countable
- $P(w) = 0$
- Semantics not well-defined

Game of dice

```
on(0,1):1/3 ; on(0,2):1/3 ; on(0,3):1/3.
on(T,1):1/3 ; on(T,2):1/3 ; on(T,3):1/3 :-
  T1 is T-1, T1>=0, on(T1,F), \+ on(T1,3).
```

Hidden Markov Models




```
hmm(S,O):-hmm(q1,[],S,O).
hmm(end,S,S,[]).
hmm(Q,S0,S,[L|O]):-
  Q\= 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.
```

Distribution Semantics with Function Symbols

- Semantics proposed for ICL and PRISM, applicable also to the other languages
- Definition of a probability measure μ over W_T
- μ assign a probability to every element of an algebra Ω of subsets of W_T, i.e. a set of subsets closed under union and complementation
- The algebra Ω is the set of sets of worlds identified by a finite set of finite composite choices

Knowledge-Based Model Construction

- The probabilistic logic theory is used directly as a template for generating an underlying complex graphical model [Breese et al., 1994].
- Languages: CLP(BN), Markov Logic

CLP(BN)

- Variables in a CLP(BN) program can be random
- Their values, parents and CPTs are defined with the program
- To answer a query with uninstantiated random variables, CLP(BN) builds a BN and performs inference
- The answer will be a probability distribution for the variables
- Probabilistic dependencies expressed by means of CLP constraints

```
{ Var = Function with p(Values, Dist) }
{ Var = Function with p(Values, Dist, Parents) }
```

CLP(BN)

```
....
course_difficulty(Key, Dif) :-
{ Dif = difficulty(Key) with p([h,m,l],
[0.25, 0.50, 0.25]) }.
student_intelligence(Key, Int) :-
{ Int = intelligence(Key) with p([h, m, l],
[0.5,0.4,0.1]) }.
....
registration(r0,c16,s0).
registration(r1,c10,s0).
registration(r2,c57,s0).
registration(r3,c22,s1).
```

CLP(BN)

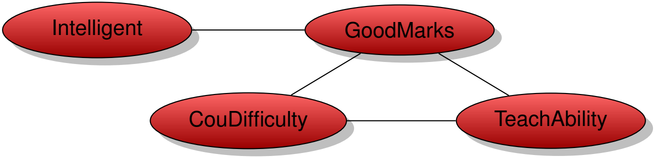
```
....
registration_grade(Key, Grade):-
registration(Key, CKey, SKey),
course_difficulty(CKey, Dif),
student_intelligence(SKey, Int),
{ Grade = grade(Key) with
 p([a,b,c,d],
%h h  h m  h l  m h  m m  m l  l h  l m  l l
[0.20,0.70,0.85,0.10,0.20,0.50,0.01,0.05,0.10,
 0.60,0.25,0.12,0.30,0.60,0.35,0.04,0.15,0.40,
 0.15,0.04,0.02,0.40,0.15,0.12,0.50,0.60,0.40,
 0.05,0.01,0.01,0.20,0.05,0.03,0.45,0.20,0.10 ],
 [Int,Dif]))
}.
.....
```

CLP(BN)

```
?- [school_32].
   ?- registration_grade(r0,G).
p(G=a)=0.4115,
p(G=b)=0.356,
p(G=c)=0.16575,
p(G=d)=0.06675 ?
?- registration_grade(r0,G),
   student_intelligence(s0,h).
p(G=a)=0.6125,
p(G=b)=0.305,
p(G=c)=0.0625,
p(G=d)=0.02 ?
```

Markov Networks

- Undirected graphical models



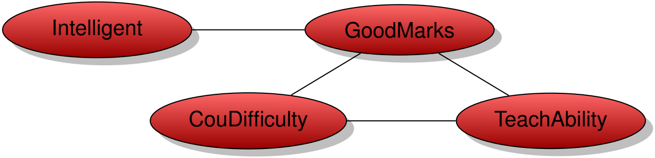
- Each clique in the graph is associated with a potential ϕ_i

$$P(\mathbf{x}) = \frac{\prod_i \phi_i(\mathbf{x_i})}{Z}$$

$$Z = \sum_{\mathbf{x}} \prod_i \phi_i(\mathbf{x_i})$$

Intelligent	GoodMarks	$\phi_i(V, T)$
false	false	4.5
false	true	4.5
true	false	1.0
true	true	4.5

Markov Networks



- If all the potential are strictly positive, we can use a log-linear model (where the f_is are features)

$$P(\mathbf{x}) = \frac{\exp(\sum_i w_i f_i(\mathbf{x_i}))}{Z}$$

$$Z = \sum_{\mathbf{x}} \exp(\sum_i w_i f_i(\mathbf{x_i})))$$

$$f_i(Intelligent, GoodMarks) = \begin{cases} 1 & \text{if } \neg\text{Intelligent}\vee\text{GoodMarks} \\ 0 & \text{otherwise} \end{cases}$$

$$w_i = 1.5$$

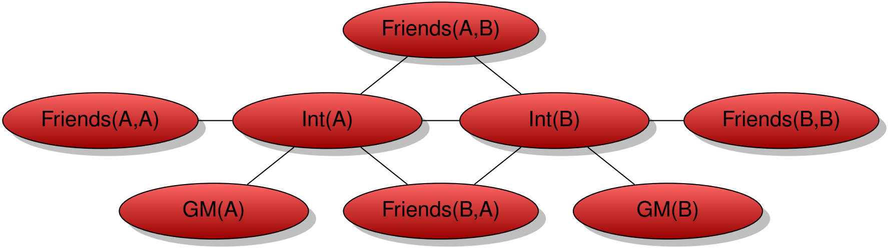
Markov Logic

- A Markov Logic Network (MLN) is a set of pairs (F, w) where F is a formula in first-order logic w is a real number
- Together with a set of constants, it defines a Markov network with
  - One node for each grounding of each predicate in the MLN
  - One feature for each grounding of each formula F in the MLN, with the corresponding weight w

Markov Logic Example

$$1.5 \quad \forall x \; \mathit{Intelligent}(x) \rightarrow \mathit{GoodMarks}(x)$$
$$1.1 \quad \forall x, y \; \mathit{Friends}(x, y) \rightarrow (\mathit{Intelligent}(x) \leftrightarrow \mathit{Intelligent}(y))$$

- Constants Anna (A) and Bob (B)



Markov Networks

- Probability of an interpretation **x**

$$P(\mathbf{x}) = \frac{\exp(\sum_i w_i n_i(\mathbf{x_i}))}{Z}$$

- $n_i(\mathbf{x_i})$ = number of true groundings of formula F_i in **x**
- Typed variables and constants greatly reduce size of ground Markov net

References I

Breese, J. S., Goldman, R. P., and Wellman, M. P. (1994).
Introduction to the special section on knowledge-based construction of probabilistic and decision models.
IEEE Transactions On Systems, Man and Cybernetics, 24(11):1577–1579.

Dantsin, E. (1991).
Probabilistic logic programs and their semantics.
In *Russian Conference on Logic Programming*, volume 592 of *LNCS*, pages 152–164. Springer.

De Raedt, L., Kimmig, A., and Toivonen, H. (2007).
Problog: A probabilistic prolog and its application in link discovery.
In *International Joint Conference on Artificial Intelligence*, pages 2462–2467.

References II

Poole, D. (1993).
Logic programming, abduction and probability - a top-down anytime algorithm for estimating prior and posterior probabilities.
New Gener. Comput., 11(3):377–400.

Poole, D. (1997).
The Independent Choice Logic for modelling multiple agents under uncertainty.
Artif. Intell., 94(1–2):7–56.

Sato, T. (1995).
A statistical learning method for logic programs with distribution semantics.
In *International Conference on Logic Programming*, pages 715–729.

References III

Vennekens, J., Verbaeten, S., and Bruynooghe, M. (2004).
Logic programs with annotated disjunctions.
In *International Conference on Logic Programming*, volume 3131 of *LNCS*, pages 195–209. Springer.