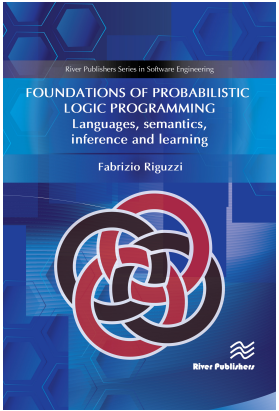
Probabilistic Inductive Logic Programming

Fabrizio Riguzzi

Department of Mathematics and Computer Science
University of Ferrara, Italy
fabrizio.riguzzi@unife.it

Dipartimento di Matematica e Informatica

Outline



Dipartimento di Matematica e Informatica

Probabilistic Logic Programming

- Distribution Semantics [Sato ICLP95]
- 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 interpretations (or queries)
- The probability of a query is obtained from this distribution

Probabilistic Logic Programming (PLP) Languages under the Distribution Semantics

- Probabilistic Logic Programs [Dantsin RCLP91]
- Probabilistic Horn Abduction [Poole NGC93], Independent Choice Logic (ICL) [Poole AI97]
- PRISM [Sato ICLP95]
- Logic Programs with Annotated Disjunctions (LPADs) [Vennekens et al. ICLP04]
- ProbLog [De Raedt et al. IJCAI07]
- They differ in the way they define the distribution over logic programs

PLP Online

- `http://cplint.eu`
  - Inference (knowledge compilation, Monte Carlo)
  - Parameter learning (EMBLEM)
  - Structure learning (SLIPCOVER, LEMUR)
- `https://dtai.cs.kuleuven.be/problog/`
  - Inference (knwoledge compilation, Monte Carlo)
  - Parameter learning (LFI-ProbLog)

PRISM

$$sneezing(X) \leftarrow flu(X), msw(flu\_sneezing(X), 1).$$
$$sneezing(X) \leftarrow hay\_fever(X), msw(hay\_fever\_sneezing(X), 1).$$
$$flu(bob).$$
$$hay\_fever(bob).$$

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

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

Dipartimento di Matematica e Informatica

Logic Programs with Annotated Disjunctions

$$sneezing(X) : 0.7 \ ; \ null : 0.3 \leftarrow flu(X).$$
$$sneezing(X) : 0.8 \ ; \ null : 0.2 \leftarrow hay\_fever(X).$$
$$flu(bob).$$
$$hay\_fever(bob).$$

- 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

Dipartimento di Matematica e Informatica

ProbLog

$$sneezing(X) \leftarrow flu(X), flu\_sneezing(X).$$
$$sneezing(X) \leftarrow hay\_fever(X), hay\_fever\_sneezing(X).$$
$$flu(bob).$$
$$hay\_fever(bob).$$
$$0.7 :: flu\_sneezing(X).$$
$$0.8 :: hay\_fever\_sneezing(X).$$

- 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 switch/clause
- Atomic choice: selection of the i-th atom for grounding $C\theta$ of 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 $C_1 = sneezing(X) : 0.7 \vee null : 0.3 \leftarrow flu(X).$, $(C_1, \{X/bob\}, 1)$
- Composite choice κ: consistent set of atomic choices
- The probability of composite choice κ is

$$P(\kappa) = \prod_{(C_i,\theta,k)\in\kappa} \Pi_{i,k}$$

Distribution Semantics

- Selection σ: a total composite choice (one atomic choice for every grounding of each clause)
- A selection σ identifies a logic program w_σ called world
- The probability of w_σ is $P(w_\sigma) = P(\sigma) = \prod_{(C_i,\theta,k)\in\sigma} \Pi_{i,k}$
- Finite set of worlds: $W_T = \{w_1, \ldots, w_m\}$
- $P(w)$ distribution over worlds: $\sum_{w\in W_T} P(w) = 1$

Distribution Semantics

- Ground query Q
- $P(Q|w) = 1$ if Q is true in w and 0 otherwise
- $P(Q) = \sum_w P(Q, w) = \sum_w P(Q|w)P(w) = \sum_{w \models Q} P(w)$

Example Program (PRISM) Worlds

http://cplint.eu/e/sneezing_simple_msw.pl

- 4 worlds

$$sneezing(X) \leftarrow flu(X), msw(flu\_sneezing(X), 1).$$
$$sneezing(X) \leftarrow hay\_fever(X), msw(hay\_fever\_sneezing(X), 1).$$
$$flu(bob).$$
$$hay\_fever(bob).$$

$msw(flu\_sneezing(bob), 1).$	$msw(flu\_sneezing(bob), 0).$
$msw(hay\_fever\_sneezing(bob), 1).$	$msw(hay\_fever\_sneezing(bob), 1).$
$P(w_1) = 0.7 \times 0.8$	$P(w_2) = 0.3 \times 0.8$
$msw(flu\_sneezing(bob), 1).$	$msw(flu\_sneezing(bob), 0).$
$msw(hay\_fever\_sneezing(bob), 0).$	$msw(hay\_fever\_sneezing(bob), 0).$
$P(w_3) = 0.7 \times 0.2$	$P(w_4) = 0.3 \times 0.2$

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

Dipartimento di Matematica e Informatica

Example Program (LPAD) Worlds

`http://cplint.eu/e/sneezing_simple.pl`

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

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

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

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

$$P(Q) = \sum_{w \in W_{\mathcal{T}}} P(Q, w) = \sum_{w \in W_{\mathcal{T}}} P(Q|w)P(w) = \sum_{w \in W_{\mathcal{T}} : w \models Q} P(w)$$

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

Dipartimento di Matematica e Informatica

Example Program (ProbLog) Worlds

`http://cplint.eu/e/sneezing_simple_pb.pl`

- 4 worlds

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

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

$hay\_fever\_sneezing(bob).$ $P(w_2) = 0.3 \times 0.8$

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

$P(w_4) = 0.3 \times 0.2$

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

Dipartimento di Matematica e Informatica

Logic Programs with Annotated Disjunctions

`http://cplint.eu/e/sneezing.pl`

$$strong\_sneezing(X) : 0.3 \vee moderate\_sneezing(X) : 0.5 \leftarrow flu(X).$$
$$strong\_sneezing(X) : 0.2 \vee moderate\_sneezing(X) : 0.6 \leftarrow hay\_fever(X).$$
$$flu(bob).$$
$$hay\_fever(bob).$$

- 9 worlds
- $P(strong\_sneezing(bob)) = 0.3 \times 0.2 + 0.3 \times 0.6 + 0.3 \times 0.2 + 0.5 \times 0.2 + 0.2 \times 0.2 = 0.44$

Expressive Power

- All languages under the distribution semantics have the same expressive power
- LPADs have the most general syntax
- There are transformations that can convert each one into the others

Reasoning Tasks

- Inference: we want to compute the probability of a query given the model and, possibly, some evidence
- Weight learning: we know the structural part of the model (the logic formulas) but not the numeric part (the weights) and we want to infer the weights from data
- Structure learning we want to infer both the structure and the weights of the model from data

Examples

Throwing coins `http://cplint.eu/e/coin.swinb`

```
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 `http://cplint.eu/e/trigger.pl`

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

Dipartimento di Matematica e Informatica

Examples

Mendel's inheritance rules for pea plants `http://cplint.eu/e/mendel.pl`

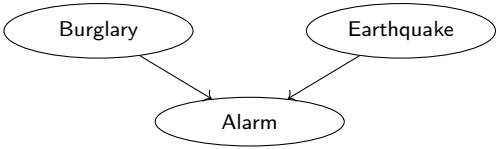
```
color(X,purple):-cg(X,_A,p).
color(X,white):-cg(X,1,w),cg(X,2,w).
cg(X,1,A):0.5 ; cg(X,1,B):0.5 :-
  mother(Y,X),cg(Y,1,A),cg(Y,2,B).
cg(X,2,A):0.5 ; cg(X,2,B):0.5 :-
  father(Y,X),cg(Y,1,A),cg(Y,2,B).
```

Probability of paths `http://cplint.eu/e/path.swinb`

```
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.
```

Dipartimento di Matematica e Informatica

Encoding Bayesian Networks



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	f
	0.1	0.9

earthq	t	f
	0.2	0.8

http://cplint.eu/e/alarm.pl

```
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).
```

Dipartimento di Matematica e Informatica

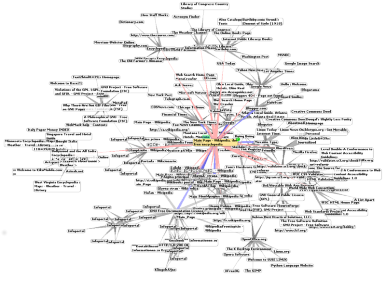
Applications

- Link prediction: given a (social) network, compute the probability of the existence of a link between two entities (UWCSE)

```
advisedby(X, Y) :0.7 :-
  publication(P, X),
  publication(P, Y),
  student(X).
```

Applications

- Classify web pages on the basis of the link structure (WebKB)

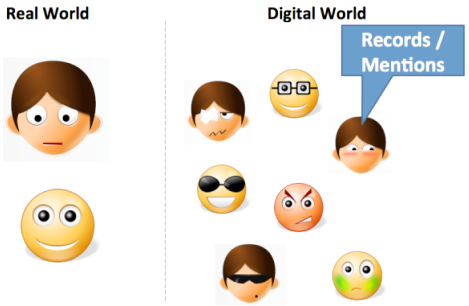


```
coursePage(Page1): 0.3 :- linkTo(Page2,Page1),coursePage(Page2).
coursePage(Page1): 0.6 :- linkTo(Page2,Page1),facultyPage(Page2).
...
coursePage(Page): 0.9 :- has('syllabus',Page).
...
```

Dipartimento di Matematica e Informatica

Applications

- Entity resolution: identify identical entities in text or databases

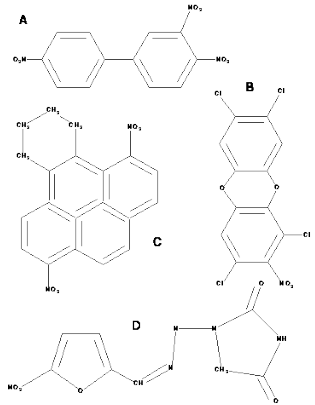



```
samebib(A,B):0.9 :-
samebib(A,C), samebib(C,B).
sameauthor(A,B):0.6 :-
  sameauthor(A,C), sameauthor(C,B).
sametitle(A,B):0.7 :-
  sametitle(A,C), sametitle(C,B).
samevenue(A,B):0.65 :-
  samevenue(A,C), samevenue(C,B).
samebib(B,C):0.5 :-
  author(B,D),author(C,E),sameauthor(D,E).
samebib(B,C):0.7 :-
  title(B,D),title(C,E),sametitle(D,E).
samebib(B,C):0.6 :-
  venue(B,D),venue(C,E),samevenue(D,E).
samevenue(B,C):0.3 :-
  haswordvenue(B,logic),
  haswordvenue(C,logic).
...
```

Dipartimento di Matematica e Informatica

Applications

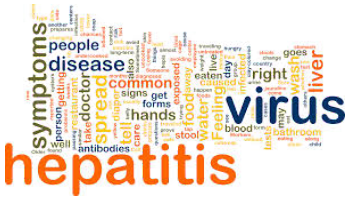
- Chemistry: given the chemical composition of a substance, predict its mutagenicity or its carcenogenicity




```
active(A):0.4 :-
   atm(A,B,c,29,C),
   gteq(C,-0.003),
   ring_size_5(A,D).
active(A):0.6:-
   lumo(A,B), lteq(B,-2.072).
active(A):0.3 :-
   bond(A,B,C,2),
   bond(A,C,D,1),
   ring_size_5(A,E).
active(A):0.7 :-
   carbon_6_ring(A,B).
active(A):0.8 :-
   anthracene(A,B).
...
```

Applications

- Medicine: diagnose diseases on the basis of patient information (Hepatitis), influence of genes on HIV, risk of falling of elderly people



Inference for PLP under DS

- Computing the probability of a query (no evidence)
- Knowledge compilation:
  - compile the program to an intermediate representation
    - Binary Decision Diagrams (BDD) (ProbLog [De Raedt et al. IJCAI07], `cplint` [Riguzzi AIIA07,Riguzzi LJIGPL09], PITA [Riguzzi & Swift ICLP10])
    - deterministic, Decomposable Negation Normal Form circuit (d-DNNF) (ProbLog2 [Fierens et al. TPLP15])
    - Sentential Decision Diagrams (ProbLog2 [Fierens et al. TPLP15])
  - compute the probability by weighted model counting

Inference for PLP under DS

- Bayesian Network based:
  - Convert to BN
  - Use BN inference algorithms (CVE [Meert et al. ILP09])
- Lifted inference

Knowledge Compilation

- Assign Boolean random variables to the probabilistic rules
- Given a query Q, compute its explanations, assignments to the random variables that are sufficient for entailing the query
- Let K be the set of all possible explanations
- Build a Boolean formula $F(Q)$
- Transform it into an intermediate representation: BDD, d-DNNF, SDD
- Perform Weighted Model Counting (WMC)

ProbLog

$$sneezing(X) \leftarrow flu(X), flu\_sneezing(X).$$
$$sneezing(X) \leftarrow hay\_fever(X), hay\_fever\_sneezing(X).$$
$$flu(bob).$$
$$hay\_fever(bob).$$
$$C_1 = 0.7 :: flu\_sneezing(X).$$
$$C_2 = 0.8 :: hay\_fever\_sneezing(X).$$

Definitions

- Composite choice κ: consistent set of atomic choices (C_i, θ_j, l) with $l \in \{1, 2\}$, example $\kappa = \{(C_1, \{X/bob\}, 1)\}$
- Set of worlds compatible with κ: $\omega_\kappa = \{w_\sigma | \kappa \subseteq \sigma\}$
- Explanation κ for a query Q: Q is true in every world of ω_κ, example $Q = sneezing(bob)$ and $\kappa = \{(C_1, \{X/bob\}, 1)\}$
- A set of composite choices K is covering with respect to Q: every world w in which Q is true is such that $w \in \omega_K$ where $\omega_K = \bigcup_{\kappa \in K} \omega_\kappa$
- Example:

$$K_1 = \{\{(C_1, \{X/bob\}, 1)\}, \{(C_2, \{X/bob\}, 1)\}\} \quad (1)$$

  is covering for $sneezing(bob)$.

Dipartimento di Matematica e Informatica

Finding Explanations

- All explanations for the query are collected
- ProbLog: source to source transformation for facts, use of dynamic database
- `cplint` (PITA): source to source transformation, addition of an argument to predicates

Explanation Based Inference Algorithm

- K = set of explanations found for Q, the probability of Q is given by the probability of the formula

$$f_K(\mathbf{X}) = \bigvee_{\kappa \in K} \bigwedge_{(C_i,\theta_j,l) \in \kappa} (X_{C_i\theta_j} = l)$$

  where $X_{C_i\theta_j}$ is a random variable whose domain is $1, 2$ and $P(X_{C_i\theta_j} = l) = \Pi_{i,l}$
- Binary domain: we use a Boolean variable X_{ij} to represent $(X_{C_i\theta_j} = 1)$
- $\overline{X_{ij}}$ represents $(X_{C_i\theta_j} = 2)$

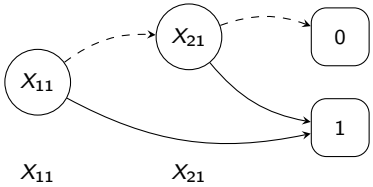
Example

A set of covering explanations for $sneezing(bob)$ is $K = \{\kappa_1, \kappa_2\}$
$\kappa_1 = \{(C_1, \{X/bob\}, 1)\} \quad \kappa_2 = \{(C_2, \{X/bob\}, 1)\}$
$K = \{\kappa_1, \kappa_2\}$
$f_K(\mathbf{X}) = (X_{C_1\{X/bob\}} = 1) \vee (X_{C_2\{X/bob\}} = 1).$
$X_{11} = (X_{C_1\{X/bob\}} = 1) \quad X_{21} = (X_{C_2\{X/bob\}} = 1)$
$f_K(\mathbf{X}) = X_{11} \vee X_{21}.$
$P(f_K(\mathbf{X})) = P(X_{11} \vee X_{21}) = P(X_{11}) + P(X_{21}) - P(X_{11})P(X_{21})$

- In order to compute the probability, we must make the explanations mutually exclusive
- Compute the Weighted Model Count
- [De Raedt at. IJCAI07]: Binary Decision Diagram (BDD)

Binary Decision Diagrams

- A BDD for a function of Boolean variables is a rooted graph that has one level for each Boolean variable
- A node n in a BDD has two children: one corresponding to the 1 value of the variable associated with n and one corresponding the 0 value of the variable
- The leaves store either 0 or 1.

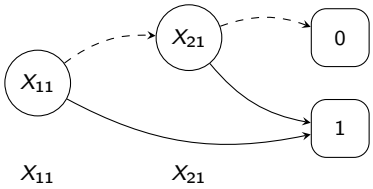


Dipartimento di Matematica e Informatica

Binary Decision Diagrams

- BDDs can be built by combining simpler BDDs using Boolean operators
- While building BDDs, simplification operations can be applied that delete or merge nodes
- Merging is performed when the diagram contains two identical sub-diagrams
- Deletion is performed when both arcs from a node point to the same node
- A reduced BDD often has a much smaller number of nodes with respect to the original BDD

Binary Decision Diagrams



$$f_K(\mathbf{X}) = X_{11} \times f_K^{X_{11}}(\mathbf{X}) + \overline{X_{11}} \times f_K^{\overline{X_{11}}}(\mathbf{X})$$

$$P(f_K(\mathbf{X})) = P(X_{11})P(f_K^{X_{11}}(\mathbf{X})) + (1 - P(X_{11}))P(f_K^{\overline{X_{11}}}(\mathbf{X}))$$

$$P(f_K(\mathbf{X})) = 0.7 \cdot P(f_K^{X_{11}}(\mathbf{X})) + 0.3 \cdot P(f_K^{\overline{X_{11}}}(\mathbf{X}))$$

Dipartimento di Matematica e Informatica

Probability from a BDD

- Dynamic programming algorithm [De Raedt et al IJCAI07]

```
1: function Prob(node)
2:     if node is a terminal then
3:         return 1
4:     else
5:         if TableProb(node.pointer) ≠ null then
6:             return TableProb(node)
7:         else
8:             p0 ←Prob(child_0(node))
9:             p1 ←Prob(child_1(node))
10:            if child_0(node).comp then
11:                p0 ← (1 − p0)
12:            end if
13:            Let π be the probability of being true of var(node)
14:            Res ← p1 · π + p0 · (1 − π)
15:            Add node.pointer → Res to TableProb
16:            return Res
17:        end if
18:    end if
19: end function
```

Logic Programs with Annotated Disjunctions

$$
\begin{array}{lll}
C_1 = & strong\_sneezing(X) : 0.3 \vee moderate\_sneezing(X) : 0.5 & \leftarrow \quad flu(X). \\
C_2 = & strong\_sneezing(X) : 0.2 \vee moderate\_sneezing(X) : 0.6 & \leftarrow \quad hay\_fever(X). \\
C_3 = & flu(bob). & \\
C_4 = & hay\_fever(bob). &
\end{array}
$$

- Distributions over the head of rules
- More than two head atoms

Dipartimento di Matematica e Informatica

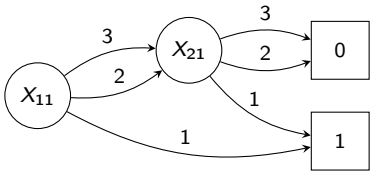
Example

A set of covering explanations for $strong\_sneezing(bob)$ is $K = \{\kappa_1, \kappa_2\}$
$\kappa_1 = \{(C_1, \{X/bob\}, 1)\}$
$\kappa_2 = \{(C_2, \{X/bob\}, 1)\}$
$X_{11} = X_{C_1\{X/bob\}}$
$X_{21} = X_{C_2\{X/bob\}}$
$f_K(\mathbf{X}) = (X_{11} = 1) \vee (X_{21} = 1).$
$P(f_X) = P(X_{11} = 1) + P(X_{21} = 1) - P(X_{11} = 1)P(X_{21} = 1)$

- To make the explanations mutually exclusive: Multivalued Decision Diagram (MDD)

Dipartimento di Matematica e Informatica

Multivalued Decision Diagrams



$$f_K(\mathbf{X}) = \bigvee_{l \in |X_{11}|} (X_{11} = l) \wedge f_K^{X_{11}=l}(\mathbf{X})$$

$$P(f_K(\mathbf{X})) = \sum_{l \in |X_{11}|} P(X_{11} = l) P(f_K^{X_{11}=l}(\mathbf{X}))$$

$$f_K(\mathbf{X}) = (X_{11} = 1) \wedge f_K^{X_{11}=1}(\mathbf{X}) + (X_{11} = 2) \wedge f_K^{X_{11}=2}(\mathbf{X}) + (X_{11} = 3) \wedge f_K^{X_{11}=3}(\mathbf{X})$$

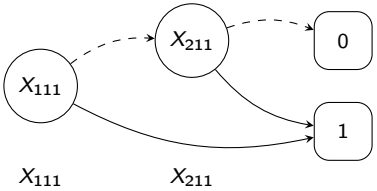
$$f_K(\mathbf{X}) = 0.3 \cdot P(f_K^{X_{11}=1}(\mathbf{X})) + 0.5 \cdot P(f_K^{X_{11}=2}(\mathbf{X})) + 0.2 \cdot P(f_K^{X_{11}=3}(\mathbf{X}))$$

Manipulating Multivalued Decision Diagrams

- Use an MDD package
- Convert to BDD, use a BDD package: BDD packages more developed, more efficient
- Conversion to BDD
  - Log encoding
  - Binary splits: more efficient

Transformation to a Binary Decision Diagram

- For a variable X_{ij} having n values, we use $n-1$ Boolean variables $X_{ij1}, \ldots, X_{ijn-1}$
- $X_{ij} = l$ for $l = 1, \ldots n-1$: $\overline{X_{ij1}} \wedge \overline{X_{ij2}} \wedge \ldots \wedge \overline{X_{ijl-1}} \wedge X_{ijl}$,
- $X_{ij} = n$: $\overline{X_{ij1}} \wedge \overline{X_{ij2}} \wedge \ldots \wedge \overline{X_{ijn-1}}$.
- Parameters: $P(X_{ij1}) = P(X_{ij} = 1) \ldots P(X_{ijl}) = \frac{P(X_{ij}=l)}{\prod_{m=1}^{l-1}(1-P(X_{ijm}))}$.



Examples of BDDs

`http://cplint.eu/e/sneezing_simple.pl`
`http://cplint.eu/e/sneezing.pl`
`http://cplint.eu/e/path.swinb`

Conditional Inference

- Computing $P(q|e)$
- Use $P(q|e) = \frac{P(q,e)}{P(e)}$
- Build BDDs for e (BDD_e) and q (BDD_q)
- The BDD for q, e is $BDD_{q,e} = BDD_e \wedge BDD_q$
- $P(q, e) = \frac{P(BDD_{q,e})}{P(BDD_e)}$
- Example: `http://cplint.eu/e/threesideddice.pl`

ProbLog2

- ProbLog2 allows probabilistic intensional facts of the form

$$\Pi :: f(X_1, X_2, \ldots, X_n) \leftarrow Body$$

with *Body* a conjunction of calls to non-probabilistic facts that define the domains of the variables $X_1, X_2, \ldots, X_n$.

- ProbLog2 allows annotated disjunctions in LPAD style of the form

$$\Pi_{i1} :: h_{i1} \ ; \ \ldots \ ; \ \Pi_{in_i} :: h_{in_i} \leftarrow b_{i1}, \ldots, b_{im_i}$$

which are equivalent to an LPAD clauses of the form

$$h_{i1} : \Pi_{i1} \ ; \ \ldots \ ; \ h_{in_i} : \Pi_{in_i} \leftarrow b_{i1}, \ldots, b_{im_i}$$

and are handled by translating them into Boolean probabilistic facts

ProbLog2

- ProbLog2 converts the program into a weighted Boolean formula and then performs Weighted Model Counting (WMC)
- Weighted Boolean formula: a formula over a set of variables $\mathbf{V} = \{V_1, \ldots, V_n\}$ associated with a weight function $w(\cdot)$ that assigns a real number to each literal built on $\mathbf{V}$.
- Weight of assignment $\omega = \{V_1 = v_1, \ldots, V_n = v_n\}$:

$$w(\omega) = \prod_{l \in \omega} w(l)$$

- Given weighted Boolean formula ϕ, the weighted model count of ϕ, $WMC_{\mathbf{V}}(\phi)$, with respect to the set of variables $\mathbf{V}$, is

$$WMC_{\mathbf{V}}(\phi) = \sum_{\omega \in SAT(\phi)} w(\omega).$$

where $SAT(\phi)$ is the set of assignments satisfying ϕ.

Dipartimento di Matematica e Informatica

ProbLog2

- ProbLog2 converts the program into a weighted formula in three stesp:
  1. Grounding $\mathcal{P}$ yielding a program $\mathcal{P}_g$, taking into account q and e in order to consider only the part of the program that is relevant to the query given the evidence.
  2. Converting the ground rules in $\mathcal{P}_g$ to an equivalent Boolean formula ϕ_r
  3. Taking into account the evidence and defining a weight function. A Boolean formula ϕ_e representing the evidence is conjoined with ϕ_r obtaining formula ϕ and a weight function is defined for all atoms in ϕ.

Example

Program

$0.1 :: burglary.$
$0.2 :: earthquake.$
$0.7 :: hears\_alarm(X) \leftarrow person(X).$
$alarm \leftarrow burglary.$
$alarm \leftarrow earthquake.$
$calls(X) \leftarrow alarm, hears\_alarm(X).$
$person(mary).$
$person(john).$

$q = burglary$ $e = calls(john)$

Relevant ground program

$0.1 :: burglary.$
$0.2 :: earthquake.$
$0.7 :: hears\_alarm(john).$
$alarm \leftarrow burglary.$
$alarm \leftarrow earthquake.$
$calls(john) \leftarrow alarm, hears\_alarm(john).$

The relevant ground program is now converted to an equivalent Boolean formula. The conversion is not merely syntactical as logic programming makes the Closed World Assumption while first order logic doesn't.

Dipartimento di Matematica e Informatica

Example

$alarm \leftrightarrow burglary \vee earthquake$
$calls(john) \leftrightarrow alarm \wedge hears\_alarm(john)$
$calls(john)$

- The weight function $w(\cdot)$ is defined as: for each probabilistic fact $\Pi :: f$, f is assigned weight Π and $\neg f$ is assigned weight $1 - \Pi$. All the other literals are assigned weight 1.

Dipartimento di Matematica e Informatica

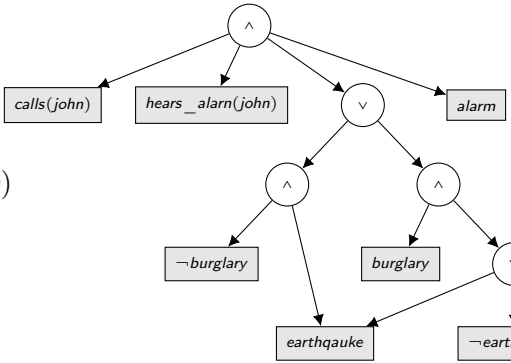
Knowledge Compilation

- By knowledge compilation, ProbLog2 translates ϕ to a smooth d-DNNF Boolean formula
- A NNF formula is a rooted directed acyclic graph in which each leaf node is labeled with a literal and each internal node is labeled with a conjunction or disjunction.
- Smooth d-DNNF satisfy also
  - Decomposability (D): for every conjunction node, no couple of children of the node has any variable in common
  - Determinism (d): for every disjunction node, every couple of children represents formulas that are logically inconsistent with each other.
  - Smoothness: for every disjunction node, all children use exactly the same set of variables.

Knowledge Compilation

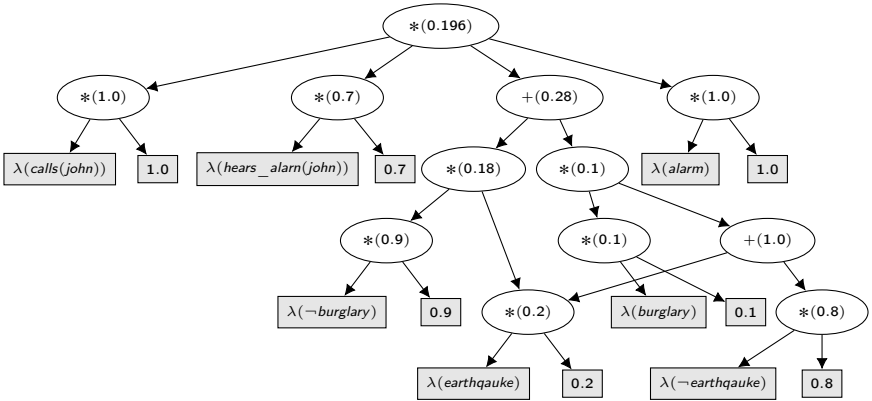
- Compilers for d-DNNF usually start from formulas in CNF (c2d [Darwiche ECAI04], Dsharp [Muise et al CAI12])

$alarm \leftrightarrow burglary \vee earthquake$
$calls(john) \leftrightarrow alarm \wedge hears\_alarm(john)$
$calls(john)$



Dipartimento di Matematica e Informatica

d-DNNF Circuit



Knowledge Compilation

- This transformation is equivalent to transforming the weighted formula into

$$WMC(\phi) = \sum_{\omega \in SAT(\phi)} \prod_{l \in \omega} w(l)\lambda(l) = \sum_{\omega \in SAT(\phi)} \prod_{l \in \omega} w(l) \prod_{l \in \omega} \lambda(l)$$

- Given the arithmetic circuit, the WMC can be computed by evaluating the circuit bottom-up after having assigned the value 1 to all the indicator variables and their weight to the literals
- $WMC_{\mathbf{V}}(\phi) = P(e)$: The value computed for the root is the probability of evidence

Dipartimento di Matematica e Informatica

Knowledge Compilation

- It is possible to compute the probability of any evidence, provided that it extends the initial evidence
- To compute $P(e, l_1 \dots l_n)$ for any conjunction of literals $l_1, \dots, l_n$ it is enough to set the indicator variables as $\lambda(l_i) = 1$, $\lambda(\neg l_i) = 0$ (where $\neg\neg a = a$) and $\lambda(l) = 1$ for the other literals l, and evaluate the circuit.
- In fact the value $f(l_1 \dots l_n)$ of the root node will give:

$$f(l_1 \dots l_n) = \sum_{\omega \in SAT(\phi)} \prod_{l \in \omega} w(l) \prod_{l \in \omega} \begin{cases} 1, & \text{if } \{l_1 \dots l_n\} \subseteq \omega \\ 0, & \text{otherwise} \end{cases} =$$

$$\sum_{\omega \in SAT(\phi), \{l_1 \dots l_n\} \subseteq \omega} \prod_{l \in \omega} w(l) =$$

$$P(e, l_1 \dots l_n)$$

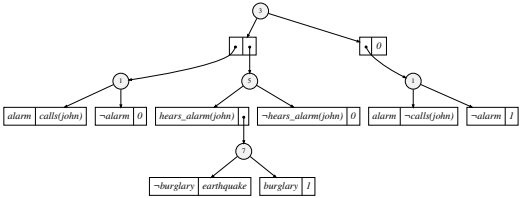
- So in theory one could build the circuit for formula ϕ_r only,
- The formula for evidence however usually simplifies the compilation process

Conditional Queries

- To answer conditional queries $P(q|e)$ use $P(q|e) = \frac{P(q,e)}{P(e)}$
- $P(e) = WMC_(\phi)$
- $P(q, e) = f(q)$

SDDs

- More recently, ProbLog2 has also included the possibility of compiling the Boolean function to Sentential Decision Diagrams (SDDs)



- An SDD [Darwiche 11] contains two types of nodes: decision nodes, represented as circles, and elements, represented as paired boxes.
- Elements are the children of decision nodes and each box in an element can contain a pointer to a decision node or a terminal node, either a literal or the constants 0 or 1.
- A decision node with children $(p_1, s_1), \ldots, (p_n, s_n)$ represents the function $(p_1 \wedge s_1) \vee \ldots \vee (p_n \wedge s_n)$.

Dipartimento di Matematica e Informatica

Reasoning Tasks

- Inference: we want to compute the probability of a query given the model and, possibly, some evidence
- Weight learning: we know the structural part of the model (the logic formulas) but not the numeric part (the weights) and we want to infer the weights from data
- Structure learning we want to infer both the structure and the weights of the model from data

Dipartimento di Matematica e Informatica

Parameter Learning

Definition (Learning Problem)

Given an LPAD $\mathcal{P}$ with unknown parameters and two sets $E^+ = \{e_1, \ldots, e_T\}$ and $E^- = \{e_{T+1}, \ldots, e_Q\}$ of ground atoms (positive and negative examples), find the value of the parameters $\mathbf{\Pi}$ of $\mathcal{P}$ that maximize the likelihood of the examples, i.e., solve

$$\arg\max_{\mathbf{\Pi}} P(E^+, {\sim}E^-) = \arg\max_{\mathbf{\Pi}} \prod_{t=1}^{T} P(e_t) \prod_{t=T+1}^{Q} P({\sim}e_t).$$

Predicates for the atoms in E^+ and E^-: target because the objective is to be able to better predict the truth value of atoms for them.

Parameter Learning

- Looking for the maximum likelihood parameters of the disjunctive clauses
- The random variables associated to clauses not observed in the dataset, which contains only derived atoms.
- Relative frequency cannot be used
- Expectation Maximization

Parameter Learning for ProbLog and LPADs

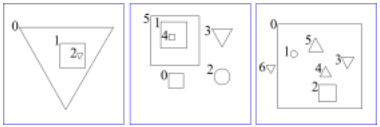
- [Thon et al. ECML 2008] proposed an adaptation of EM for CPT-L, a simplified version of LPADs
- The algorithm computes the counts efficiently by repeatedly traversing the BDDs representing the explanations
- [Ishihata et al. ILP 2008] independently proposed a similar algorithm
- LFI-ProbLog [Gutamnn et al. ECML 2011]: EM for ProbLog on BDDs
- EMBLEM [Riguzzi & Bellodi IDA 2013] adapts [Ishihata et al. ILP 2008] to LPADs

Parameter Learning

- Typically, the LPAD $\mathcal{P}$ has two components:
  - a set of rules, annotated with parameters
  - a set of certain ground facts, representing background knowledge on individual cases of a specific world
- Useful to provide information on more than one world: a background knowledge and sets of positive and negative examples for each world
- Description of one world: *mega-interpretation* or *mega-example*
- Positive examples encoded as ground facts of the mega-interpretation and the negative examples as suitably annotated ground facts (such as $neg(a)$ for negative example a)
- The task then is maximizing the product of the likelihood of the examples for all mega-interpretations.

Example: Bongard Problems

- Introduced by the Russian scientist M. Bongard
- Pictures containing shapes with different properties, such as small, large, pointing down, ... and different relationships between them, such as inside, above, ...
- Some positive and some negative
- Problem: discriminate between the two classes.



Dipartimento di Matematica e Informatica

Data

Each mega-example encodes a single picture

Models

```
begin(model(2)).
pos.
triangle(o5).
config(o5,up).
square(o4).
in(o4,o5).
circle(o3).
triangle(o2).
config(o2,up).
in(o2,o3).
triangle(o1).
config(o1,up).
end(model(2)).

begin(model(3)).
neg(pos).
circle(o4).
circle(o3).
in(o3,o4).
....
```

Keys

```
pos(2).
triangle(2,o5).
config(2,o5,up).
square(2,o4).
in(2,o4,o5).
circle(2,o3).
triangle(2,o2).
config(2,o2,up).
in(2,o2,o3).
triangle(2,o1).
config(2o1,up).

neg(pos(3)).
circle(3,o4).
circle(3,o3).
in(3,o3,o4).
....
```

Program

Theory for parameter learning and background

```
pos:0.5 :-
  circle(A),
  in(B,A).
pos:0.5 :-
  circle(A),
  triangle(B).
```

The task is to tune the two parameters
`http://cplint.eu/e/bongard.pl`

EMBLEM

- The interpretations record the truth value of ground atoms, not of the random variables
- Unseen data: relative frequency can't be used
- Expectation-Maximization algorithm:
  - Expectation step: the distribution of the unseen variables in each instance is computed given the observed data
  - Maximization step: new parameters are computed from the distributions using relative frequency
  - End when likelihood does not improve anymore

EMBLEM

- EM over Bdds for probabilistic Logic programs Efficient Mining [Bellodi and Riguzzi IDA 2013]
- Input: an LPAD; logical interpretations (data); *target* predicate(s)
- All ground atoms in the interpretations for the target predicate(s) correspond to as many queries
- BDDs encode the explanations for each query
- Expectations computed with two passes over the BDDs

EMBLEM

- EMBLEM encodes multi-valued random variable with Boolean random variables
- Variable X_{ij} associated with grounding θ_j of clause C_i having n values.
- Encoding using $n-1$ Boolean variables $X_{ij1}, \ldots, X_{ijn-1}$.
- Equation $X_{ij} = k$ for $k = 1, \ldots n-1$ represented by

$$\overline{X_{ij1}} \wedge \ldots \wedge \overline{X_{ijk-1}} \wedge X_{ijk}$$

- Equation $X_{ij} = n$ represented by

$$\overline{X_{ij1}} \wedge \ldots \wedge \overline{X_{ijn-1}}.$$

- Parameters:

$$\begin{aligned} P(X_{ij1}) &= P(X_{ij} = 1) \\ &\ldots \\ P(X_{ijk}) &= \frac{P(X_{ij} = k)}{\prod_{l=1}^{k-1}(1 - P(X_{ijk-1}))} \end{aligned}$$

EMBLEM

- Let X_{ijk} for $k = 1, \ldots, n_i - 1$ and $j \in g(i)$ be the Boolean random variables associated with grounding $C_i\theta_j$ of clause C_i of $\mathcal{P}$ where n_i is the number of head atoms of C_i and $g(i)$ is the set of indices of grounding substitutions of C_i.

Example

`http://cplint.eu/e/epidemic.pl`

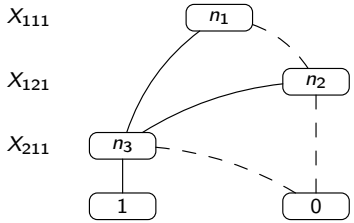
$C_1 = epidemic : 0.6 \; ; \; pandemic : 0.3 \leftarrow flu(X), cold.$

$C_2 = cold : 0.7.$

$C_3 = flu(david).$

$C_4 = flu(robert).$

- Clause C_1: two groundings, first: X_{111} and X_{112}, latter: X_{121} and X_{122}.
- C_2: single grounding, random variable X_{211}.



Dipartimento di Matematica e Informatica

EMBLEM

- EMBLEM alternates between the two phases:
  - Expectation: compute $\mathbf{E}[c_{ik0}|e]$ and $\mathbf{E}[c_{ik1}|e]$ for all examples e, rules C_i in $\mathcal{P}$ and $k = 1, \ldots, n_i - 1$, where c_{ikx} is the number of times a variable X_{ijk} takes value x for $x \in \{0, 1\}$, with j in $g(i)$.

$$\mathbf{E}[c_{ikx}|e] = \sum_{j \in g(i)} P(X_{ijk} = x|e).$$

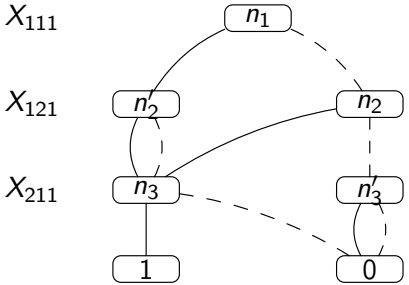
  - Maximization: compute π_{ik} for all rules C_i and $k = 1, \ldots, n_i - 1$.

$$\pi_{ik} = \frac{\sum_{e \in E} \mathbf{E}[c_{ik1}|e]}{\sum_{q \in E} \mathbf{E}[c_{ik0}|e] + \mathbf{E}[c_{ik1}|e]}$$

Dipartimento di Matematica e Informatica

EMBLEM

- $P(X_{ijk} = x|e)$ is given by $P(X_{ijk} = x|e) = \frac{P(X_{ijk}=x,e)}{P(e)}$.
- Consider a BDD for an example e built by applying only the merge rule



EMBLEM

- $P(e)$ is given by the sum of the probabilities of all the paths in the BDD from the root to a 1 leaf
- To compute $P(X_{ijk} = x, e)$ we need to consider only the paths passing through the x-child of a node n associated with variable X_{ijk} so

$$P(X_{ijk} = x, e) = \sum_{n \in N(X_{ijk})} \pi_{ikx} F(n) B(child_x(n)) = \sum_{n \in N(X_{ijk})} e^x(n)$$

  - $F(n)$ is the *forward probability*, the probability mass of the paths from the root to n,
  - $B(n)$ is the *backward probability*, the probability mass of paths from n to the 1 leaf.

EMBLEM

- BDD obtained by also applying the deletion rule: paths where there is no node associated with X_{ijk} can also contribute to $P(X_{ijk} = x, e)$.
- Suppose the BDD was obtained deleting node m 0-child of n associated with variable X_{ijk}
- Outgoing edges of m both point to $child_0(n)$.
- The probability mass of the two paths that were merged was $e^0(n)(1 - \pi_{ik})$ and $e^0(n)\pi_{ik}$ for the paths passing through the 0-child and 1-child of m respectively
- The first quantity contributes to $P(X_{ijk} = 0, e)$, the latter to $P(X_{ijk} = 1, e)$.

Dipartimento di Matematica e Informatica

GetForward

```
1: procedure GetForward(root)
2:     F(root) = 1
3:     F(n) = 0 for all nodes
4:     for l = 1 to levels do                    ▷ levels is the number of levels of the BDD rooted at root
5:         Nodes(l) = ∅
6:     end for
7:     Nodes(1) = {root}
8:     for l = 1 to levels do
9:         for all node ∈ Nodes(l) do
10:            let X_ijk be v(node), the variable associated with node
11:            if child_0(node) is not terminal then
12:                F(child_0(node)) = F(child_0(node)) + F(node) · (1 − π_ik)
13:                add child_0(node) to Nodes(level(child_0(node)))
14:            end if
15:            if child_1(node) is not terminal then
16:                F(child_1(node)) = F(child_1(node)) + F(node) · π_ik
17:                add child_1(node) to Nodes(level(child_1(node)))
18:            end if
19:        end for
20:    end for
21: end procedure
```

GetBackward

1: **function** GetBackward($node$)
2: **if** $node$ is a terminal **then**
3: **return** $value(node)$
4: **else**
5: let X_{ijk} be $v(node)$
6: $B(child_0(node))$ =GetBackward($child_0(node)$)
7: $B(child_1(node))$ =GetBackward($child_1(node)$)
8: $e^0(node) = F(node) \cdot B(child_0(node)) \cdot (1 - \pi_{ik})$
9: $e^1(node) = F(node) \cdot B(child_1(node)) \cdot \pi_{ik}$
10: $\eta^0(i,k) = \eta^0(i,k) + e^0(node)$
11: $\eta^1(i,k) = \eta^1(i,k) + e^1(node)$
12: take into account deleted paths
13: **return** $B(child_0(node)) \cdot (1 - \pi_{ik}) + B(child_1(node)) \cdot \pi_{ik}$
14: **end if**
15: **end function**

EMBLEM

1: **function** EMBLEM($E, \mathcal{P}, \epsilon, \delta$)
2: build $BDDs$
3: $LL = -inf$
4: **repeat**
5: $LL_0 = LL$
6: LL = Expectation($BDDs$)
7: Maximization
8: **until** $LL - LL_0 < \epsilon \vee LL - LL_0 < -LL \cdot \delta$
9: return LL, π_{ik} for all i, k
10: **end function**

EMBLEM

```
1: function Expectation(BDDs)
2:     LL = 0
3:     for all BDD ∈ BDDs do
4:         for all i do
5:             for k = 1 to n_i − 1 do
6:                 η^0(i,k) = 0;  η^1(i,k) = 0
7:             end for
8:         end for
9:         for all variables X do
10:            ς(X) = 0
11:        end for
12:        GetForward(root(BDD))
13:        Prob=GetBackward(root(BDD))
14:        take into account deleted paths
15:        for all i do
16:            for k = 1 to n_i − 1 do
17:                E[c_ik0] = E[c_ik0] + η^0(i,k)/Prob
18:                E[c_ik1] = E[c_ik1] + η^1(i,k)/Prob
19:            end for
20:        end for
21:        LL = LL + log(Prob)
```

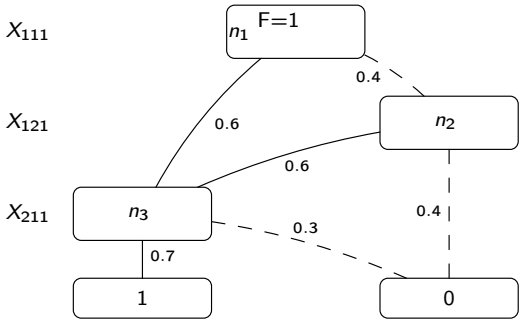
Dipartimento di Matematica e Informatica

EMBLEM

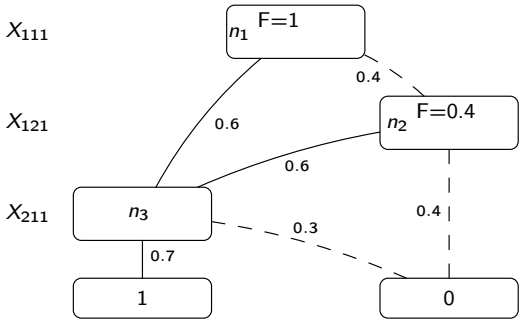
1: **procedure** Maximization
2: **for all** i **do**
3: **for** $k = 1$ to $n_i - 1$ **do**
4: $\pi_{ik} = \frac{\mathbf{E}[c_{ik\mathbf{1}}]}{\mathbf{E}[c_{ik\mathbf{0}}]+\mathbf{E}[c_{ik\mathbf{1}}]}$
5: **end for**
6: **end for**
7: **end procedure**

Dipartimento di Matematica e Informatica

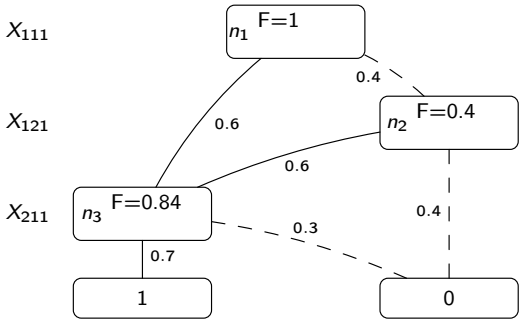
Example



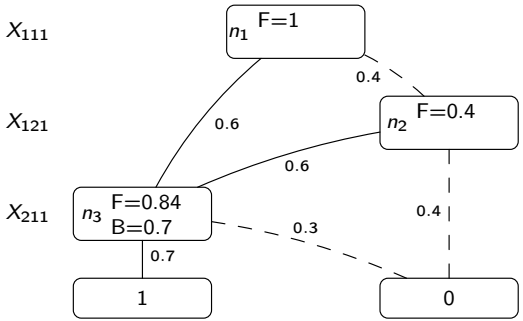
Example



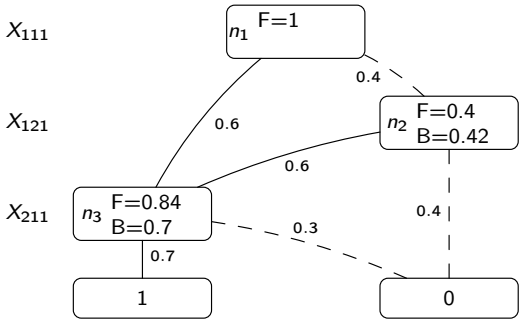
Example



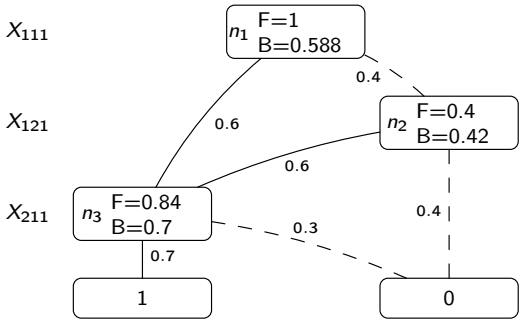
Example



Example



Example


X_{111}
X_{121}
X_{211}
n_1 F=1 B=0.588
n_2 F=0.4 B=0.42
n_3 F=0.84 B=0.7
0.4
0.6
0.6
0.4
0.3
0.7
1
0

ProbLog2

- ProbLog2 includes LFI-ProbLog [Gutmann et al PKDD 2011] that learns the parameters of ProbLog programs from partial interpretations.
- Partial interpretations specify the truth value of some but not necessarily all ground atoms.
- $\mathcal{I} = \langle I_T, I_F \rangle$: the atoms in I_T are true and those in I_F are false.
- $\mathcal{I} = \langle I_T, I_F \rangle$ can be associated with a conjunction $q(\mathcal{I}) = \bigwedge_{a \in I_T} a \wedge \bigwedge_{a \in I_F} \sim a$.

Dipartimento di Matematica e Informatica

LFI-ProbLog

Definition (LFI-ProbLog learning problem)

Given a ProbLog program $\mathcal{P}$ with unknown parameters and a set $E = \{\mathcal{I}_1, \ldots, \mathcal{I}_T\}$ of partial interpretations (the examples), find the value of the parameters $\mathbf{\Pi}$ of $\mathcal{P}$ that maximize the likelihood of the examples, i.e., solve

$$\arg\max_{\mathbf{\Pi}} P(E) = \arg\max_{\mathbf{\Pi}} \prod_{t=1}^{T} P(q(\mathcal{I}_t))$$

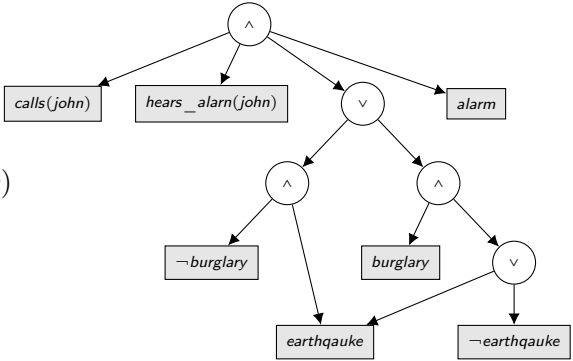
Dipartimento di Matematica e Informatica

LFI-ProbLog

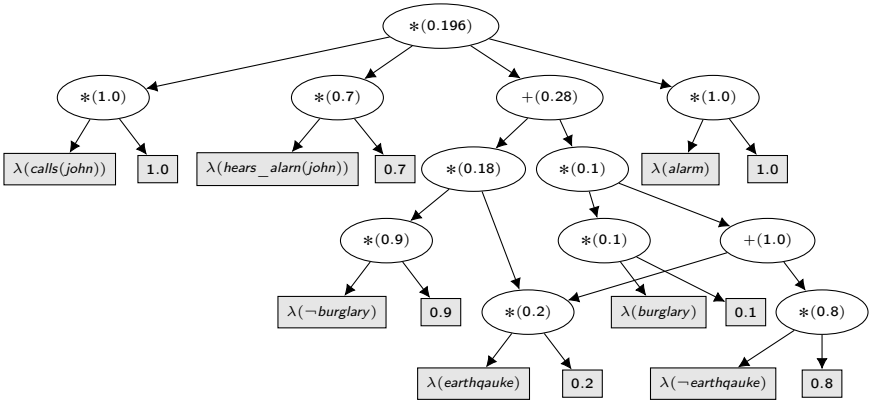
- EM algorithm
- A d-DNNF circuit for each partial interpretation $\mathcal{I} = \langle I_T, I_F \rangle$ by using the ProbLog2 inference algorithm with the evidence $q(\mathcal{I})$.
- A Boolean random variable X_{ij} is associated with each ground probabilistic fact $f_i\theta_j$.
- For each example $\mathcal{I}$, variable X_{ij} and $x \in \{0, 1\}$, LFI-ProbLog computes $P(X_{ij} = x|\mathcal{I})$.
- LFI-ProbLog computes $P(X_{ij} = x|\mathcal{I})$ by computing $P(X_{ij} = x, \mathcal{I})$ using Procedure CircP

Example of a d-DNNF Formula

$alarm \leftrightarrow burglary \vee earthquake$
$calls(john) \leftrightarrow alarm \wedge hears\_alarm(john)$
$calls(john)$



Example of a d-DNNF Circuit



Computing Expectations

$$WMC(\phi) = \sum_{\omega \in SAT(\phi)} \prod_{l \in \omega} w(l)\lambda_l = \sum_{\omega \in SAT(\phi)} \prod_{l \in \omega} w(l) \prod_{l \in \omega} \lambda_l$$

$$P(e) = \sum_{\omega \in SAT(\phi)} \prod_{l \in \omega} w(l)$$

- We want to compute $P(q|e)$ for all atoms $q \in Q$.
- Partial derivative $\frac{\partial f}{\partial \lambda_q}$ for an atom q:

$$\frac{\partial f}{\partial \lambda_q} = \sum_{\omega \in SAT(\phi), q \in \omega} \prod_{l \in \omega} w(l) \prod_{l \in \omega, l \neq q} \lambda_l = \sum_{\omega \in SAT(\phi), q \in \omega} \prod_{l \in \omega} w(l) = P(e, q)$$

Dipartimento di Matematica e Informatica

Computing Expectations

- If we compute the partial derivatives of f for all indicator variables λ_q, we get $P(q, e)$ for all atoms q.
- $v(n)$: value of each node n
- $d(n) = \frac{\partial v(r)}{\partial v(n)}$.
- $d(r) = 1$
- By the chain rule of calculus, for an arbitrary non-root node n with p indicating its parents

$$d(n) = \sum_p \frac{\partial v(r)}{\partial v(p)} \frac{\partial v(p)}{\partial v(n)} = \sum_p d(p) \frac{\partial v(p)}{\partial v(n)}.$$

Computing Expectations

- If p is a multiplication node with n' indicating its children

$$\frac{\partial v(p)}{\partial v(n)} = \frac{\partial v(n) \prod_{n' \neq n} v(n')}{\partial v(n)} = \prod_{n' \neq n} v(n').$$

- If p is an addition node with n' indicating its children

$$\frac{\partial v(p)}{\partial v(n)} = \frac{\partial v(n) + \sum_{n' \neq n} v(n')}{\partial v(n)} = 1.$$

- $+p$ an addition parent of n and $*p$ a multiplication parent of n:

$$d(n) = \sum_{+p} d(+p) + \sum_{*p} d(*p) \prod_{n' \neq n} v(n').$$

- If $v(n) \neq 0$.

$$d(n) = \sum_{+p} d(+p) + \sum_{*p} d(*p) v(*p) / v(n).$$

CircP

1: **procedure** CircP(*circuit*)
2: assign values to leaves
3: **for all** non-leaf node n with children c (visit children before parents) **do**
4: **if** n is an addition node **then**
5: $v(n) \leftarrow \sum_c v(c)$
6: **else**
7: $v(n) \leftarrow \prod_c v(c)$
8: **end if**
9: **end for**
10: $d(r) \leftarrow 1$, $d(n) = 0$ for all non-root nodes
11: **for all** non-root node n (visit parents before children) **do**
12: **for all** parents p of n **do**
13: **if** p is an addition parent **then**
14: $d(n) = d(n) + d(p)$
15: **else**
16: $d(n) \leftarrow d(n) + d(p)v(p)/v(n)$
17: **end if**
18: **end for**
19: **end for**
20: **end procedure**

Dipartimento di Matematica e Informatica

Reasoning Tasks

- Inference: we want to compute the probability of a query given the model and, possibly, some evidence
- Weight learning: we know the structural part of the model (the logic formulas) but not the numeric part (the weights) and we want to infer the weights from data
- Structure learning we want to infer both the structure and the weights of the model from data

Structure Learning for LPADs

- Given a set of interpretations (data)
- *Find the model and the parameters* that maximize the probability of the data (log-likelihood)
- SLIPCOVER: Structure LearnIng of Probabilistic logic program by searching OVER the clause space [Riguzzi & Bellodi TPLP 2015]
  1. Beam search in the space of clauses to find the promising ones
  2. Greedy search in the space of probabilistic programs guided by the LL of the data.
- *Parameter learning* by means of EMBLEM

SLIPCOVER

- Cycle on the set of predicates that can appear in the head of clauses, either target or background
- For each predicate, beam search in the space of clauses
- The initial set of beams is generated by building a set of *bottom clauses* as in Progol [Muggleton NGC 1995]
- Bottom clause: most specific clause covering an example

Language Bias

- Mode declarations as in Progol
- Syntax

```
modeh(RecallNumber,PredicateMode).
modeb(RecallNumber,PredicateMode).
```

- `RecallNumber` can be a number or *. Usually *. Maximum number of answers to queries to include in the bottom clause

Mode Declarations

- `PredicateMode` template of the form:

```
p(ModeType, ModeType,...)
```

- `ModeType` can be:
  - Simple:
    - `+T` input variables of type `T`;
    - `-T` output variables of type `T`; or
    - `#T`, `-#T` constants of type `T`.
  - Structured: of the form `f(..)` where `f` is a function symbol and every argument can be either simple or structured. For example:

Dipartimento di Matematica e Informatica

Mode Declarations

```
modeb(1,mem(+number,+list)).
modeb(1,dec(+integer,-integer)).
modeb(1,mult(+integer,+integer,-integer)).
modeb(1,plus(+integer,+integer,-integer)).
modeb(1,(+integer)=(#integer)).
modeb(*,has_car(+train,-car))
modeb(1,mem(+number,[+number|+list])).
```

Bottom Clause $\bot$

- Most specific clause covering an example e
- Form: $e \leftarrow B$
- B: set of ground literals that are true regarding the example e
- B obtained by considering the constants in e and querying the data for true atoms regarding these constants
- Values for output arguments are used as input arguments for other predicates
- A map from types to lists of constants is kept, it is enlarged with constants in the answers to the queries and the procedure is iterated a user-defined number of times
- `#T` arguments are instantiated in calls, `-#T` aren't and the values after the call are added to the list of constants
- `-#T` arguments can be used to retrieve values for `T`, `#T` can't

Dipartimento di Matematica e Informatica

Bottom Clause $\perp$

- Initialize to empty a map m from types to lists of values
- Pick a $modeh(r, s)$, an example e matching s, add to $m(T)$ the values of $+T$ arguments in e
- For $i = 1$ to d
  - For each $modeb(r, s)$

Dipartimento di Matematica e Informatica

Bottom Clause $\perp$

- For each possible way of building a query q from s by replacing $+T$ and $\#T$ arguments with constants from $m(T)$ and all other arguments with variables
  - Find all possible answers for q and put them in a list L
  - $L' := r$ elements sampled from L
  - For each $l \in L'$, add the values in l corresponding to $-T$ or $-\#T$ to $m(T)$

Dipartimento di Matematica e Informatica

Bottom Clause $\bot$

- Example:

$e = father(john, mary)$
$BG = \{parent(john, mary), parent(david, steve),$
$parent(kathy, mary), female(kathy), male(john), male(david)\}$
$modeh(*, father(+person, +person)).$
$modeb(*, parent(+person, -person)).$ $modeb(*, parent(-\#person, +person)).$
$modeb(*, male(+person)).$ $modeb(*, female(\#person)).$
$e \leftarrow B = father(john, mary) \leftarrow parent(john, mary), male(john),$
$parent(kathy, mary), female(kathy).$

Dipartimento di Matematica e Informatica

Bottom Clause $\perp$

- The resulting ground clause $\perp$ is then processed by replacing each term in a + or - placemarker with a variable
- An input variable (+T) must appear as an output variable with the same type in a previous literal and a constant (#T or -#T) is not replaced by a variable.

$\perp = father(X, Y) \leftarrow parent(X, Y), male(X), parent(kathy, Y), female(kathy).$

Dipartimento di Matematica e Informatica

Determination

`determination(pred1/n1,pred2/n2).`

- indicates that `pred2/n2` can appear in the body of clauses for predicate `pred1/n1`
- As in Progol

Head Declarations

- To generate clauses with more than two head atoms, head declarations of the form

$$modeh(r, [s_1, \ldots, s_n], [a_1, \ldots, a_n], [P_1/Ar_1, \ldots, P_k/Ar_k])$$

- $s_1, \ldots, s_n$ are schemas
- $a_1, \ldots, a_n$ are atoms such that a_i is obtained from s_i by replacing placemarkers with variables
- P_i/Ar_i are the predicates admitted in the body.
- $a_1, \ldots, a_n$ are used to indicate which variables should be shared by the atoms in the head.
- The generation of a bottom clause is the same except for the fact that the goal to call is composed of more than one atom.

Head Declarations

- Goal $a_1, \ldots, a_n$ is called and r answers that ground all a_is are kept
- Resulting bottom clauses $a_1 \ ; \ \ldots \ ; \ a_n :- b_1, \ldots, b_m$

SLIPCOVER

- The initial beam associated with predicate P/Ar of h will contain the clause with the empty body $h : 0.5.$ for each bottom clause $h :- b_1, \ldots, b_m$ or clauses with an empty body of the form

$$a_1 : \frac{1}{n+1} \; ; \; \ldots \; ; \; a_n : \frac{1}{n+1}.$$

- In each iteration of the cycle over predicates, it performs a beam search in the space of clauses for the predicate.
- The beam contains couples $(Cl, LIterals)$ where $Literals = \{b_1, \ldots, b_m\}$
- For each clause Cl of the form $Head :- Body$, the refinements are computed by adding a literal from $Literals$ to the body.

SLIPCOVER

- The tuple (Cl', $Literals'$) indicates a refined clause Cl' together with the new set $Literals'$
- EMBLEM is then executed for a theory composed of the single refined clause.
- LL is used as the score of the updated clause (Cl'', $Literals'$).
- (Cl'', $Literals'$) is then inserted into a list of promising clauses.
- Two lists are used, TC for target predicates and BC for background predicates.
- These lists ave a maximum size

Dipartimento di Matematica e Informatica

SLIPCOVER

- After the clause search phase, SLIPCOVER performs a greedy search in the space of theories:
  - it starts with an empty theory and adds a target clause at a time from the list TC.
  - After each addition, it runs EMBLEM and computes the LL of the data as the score of the resulting theory.
  - If the score is better than the current best, the clause is kept in the theory, otherwise it is discarded.
- Finally, SLIPCOVER adds all the clauses in BC to the theory and performs parameter learning on the resulting theory.

Execution Example

- UW-CSE dataset: 22 different predicates, such as `advisedby/2`, `yearsinprogram/2` and `taughtby/3`.
- The aim is to predict the predicate `advisedby/2`
- The language bias includes

```
modeh(*,advisedby(+person,+person)).
modeh(*,[advisedby(+person,+person),tempadvisedby(+person,+person)],
  [advisedby(A,B),tempadvisedby(A,B)],
  [professor/1,student/1,hasposition/2,inphase/2,publication/2,
  taughtby/3,ta/3,courselevel/2,yearsinprogram/2]).

modeh(*,[student(+person),professor(+person)],
  [student(P),professor(P)],
  [hasposition/2,inphase/2,taughtby/3,ta/3,courselevel/2,
  yearsinprogram/2,advisedby/2,tempadvisedby/2]).

modeh(*,[inphase(+person,pre_quals),inphase(+person,post_quals),
  inphase(+person,post_generals)],
  [inphase(P,pre_quals),inphase(P,post_quals),inphase(P,post_generals)],
  [professor/1,student/1,taughtby/3,ta/3,courselevel/2,
  yearsinprogram/2,advisedby/2,tempadvisedby/2,hasposition/2]).
```

Dipartimento di Matematica e Informatica

Execution Example

- *modeb* declarations such as

```
modeb(*,courselevel(+course, -level)).
modeb(*,courselevel(+course, #level)).
```

Execution Example

- Example of a two-head bottom clause generated from the first *modeh* declaration

```
advisedby(A,B):0.5 :- professor(B),student(A),hasposition(B,C),
  hasposition(B,faculty),inphase(A,D),inphase(A,pre_quals),
  yearsinprogram(A,E),taughtby(F,B,G),taughtby(F,B,H),taughtby(I,B,J),
  taughtby(I,B,J),taughtby(F,B,G),taughtby(F,B,H),
  ta(I,K,L),ta(F,M,H),ta(F,M,H),ta(I,K,L),ta(N,K,O),ta(N,A,P),
  ta(Q,A,P),ta(R,A,L),ta(S,A,T),ta(U,A,O),ta(U,A,O),ta(S,A,T),
  ta(R,A,L),ta(Q,A,P),ta(N,K,O),ta(N,A,P),ta(I,K,L),ta(F,M,H).
```

Execution Example

- Example of a multi-head bottom clause generated from the second *modeh* declaration

```
student(A):0.33; professor(A):0.33 :- inphase(A,B),
  inphase(A,post_generals),
  yearsinprogram(A,C).
```

Execution Example

- Example of a refinement from the first bottom clause is
  `advisedby(A,B):0.5 :- professor(B).`
- EMBLEM is applied to the theory, the only parameter is updated obtaining:
  `advisedby(A,B):0.108939 :- professor(B).`
- The clause is further refined to
  `advisedby(A,B):0.108939 :- professor(B),hasposition(B,C).`

Dipartimento di Matematica e Informatica

Execution Example

- Example of a refinement that is generated from the second bottom clause is
  `student(A):0.33; professor(A):0.33 :- inphase(A,B).`
- Updated refinement after EMBLEM
  `student(A):0.5869;professor(A):0.09832 :- inphase(A,B).`

Dipartimento di Matematica e Informatica

Execution Example

- When searching the *space of theories* for the target predicate `advisedby`, SLIPCOVER generates the program:

  ```
  advisedby(A,B):0.1198 :- professor(B),inphase(A,C).
  advisedby(A,B):0.1198 :- professor(B),student(A).
  ```

  with a LL of -350.01.
- After EMBLEM we get:

  ```
  advisedby(A,B):0.05465 :- professor(B),inphase(A,C).
  advisedby(A,B):0.06893 :- professor(B),student(A).
  ```

  with a LL of -318.17.
- Since the LL increased, the last clause is retained and at the next iteration a new clause is added:

  ```
  advisedby(A,B):0.12032 :- hasposition(B,C),inphase(A,D).
  advisedby(A,B):0.05465 :- professor(B),inphase(A,C).
  advisedby(A,B):0.06893 :- professor(B),student(A).
  ```

Dipartimento di Matematica e Informatica

ProbFOIL+

- ProbFOIL+ [De Raedt et al IJCAI 2015] learn rules from probabilistic examples.

Definition (ProbFoil+ learning problem)

Given

1. a set of training examples $E = \{(e_1, p_1), \ldots, (e_T, p_T)\}$ where each e_i is a ground fact for a target predicate
2. a background theory $\mathcal{B}$ containing information about the examples in the form of a ProbLog program
3. a space of possible clauses $\mathcal{L}$

find a hypothesis $H \subseteq \mathcal{L}$ so that the absolute error $AE = \sum_{i=1}^{T} |P(e_i) - p_i|$ is minimized, i.e.,

$$\arg\min_{H \in \mathcal{L}} \sum_{i=1}^{T} |P(e_i) - p_i|$$

ProbFOIL+

- Form of clauses: $x :: h \leftarrow B$, with $x \in [0, 1]$.
- To be interpreted as
  $h \leftarrow B, prob(id)$.
  $x :: prob(id)$.
- Different from an LPAD $h : x \leftarrow B$, as this stands for the union of ground rules $h' : x \leftarrow B'$. obtained by grounding $h : x \leftarrow B$.

Dipartimento di Matematica e Informatica

ProbFOIL+

- ProbFOIL+ generalizes mFOIL and FOIL
- Covering loop: one rule is added to the theory at each iteration.
- Clause search loop: builds the rule by iteratively adding literals to the body.
- The covering loop ends when a condition based on a global scoring function is satisfied.
- Clause search loop: beam search using a local scoring function as the heuristic.

ProbFOIL+

```
1: function ProbFOIL+(target)
2:     H ← ∅
3:     while true do
4:         clause ← LearnRule(H, target)
5:         if GScore(H) < GScore(H ∪ {clause}) ∧ Significant(H, clause) then
6:             H ← H ∪ {clause}
7:         else
8:             return H
9:         end if
10:    end while
11: end function
```

ProbFOIL+

```
1: function LearnRule(H, target)
2:     candidates ← {x :: target ← true}
3:     best ← (x :: target ← true)
4:     while candidates ≠ ∅ do
5:         next_cand ← ∅
6:         for all x :: target ← body ∈ candidates do
7:             for all (target ← body, refinement) ∈ ρ(target ← body) do
8:                 if not Reject(H, best, (x :: target ← body, refinement)) then
9:                     next_cand ← next_cand ∪ {(x :: target ← body, refinement)}
10:                    if LScore(H, (x :: target ← body, refinement)) > LScore(H, best) then
11:                        best ← (x :: target ← body, refinement)
12:                    end if
13:                end if
14:            end for
15:        end for
16:        candidates ← next_cand
17:    end while
18:    return best
19: end function
```

ProbFOIL+

- Global scoring function: accuracy over the dataset, given by

$$accuracy_H = \frac{TP_H + TN_H}{T}$$

where T is number of examples and TP_H and TN_H are, respectively, the number of *true positives* and of *true negatives*

- Local scoring function: an *m-estimate* of the *precision*

$$m\text{-}estimate_H = \frac{TP_H + m\frac{P}{P+N}}{TP_H + FP_H + m}$$

Dipartimento di Matematica e Informatica

ProbFOIL+

- Each example e_i is associated with a probability p_i.
- An example (e_i, p_i) contributes a part p_i to the positive part of training set and $1 - p_i$ to the negative part: $P = \sum_{i=1}^{T} p_i$ and $N = \sum_{i=1}^{T} (1 - p_i)$.
- Hypothesis H assigns a probability $p_{H,i}$ to each example e_i
- The contribution $tp_{H,i}$ of example e_i to TP_H will be $p_{H,i}$ if $p_i > p_{H,i}$ and p_i otherwise, because if $p_i < p_{H,i}$ the hypothesis is overestimating e_i.
- The contribution $fp_{H,i}$ of example e_i to FP_H will be $p_{H,i} - p_i$ if $p_i < p_{H,i}$ and 0 otherwise, because if $p_i > p_{H,i}$ the hypothesis is underestimating e_i.
- $TP_H = \sum_{i=1}^{T} tp_{H,i}$, $FP_H = \sum_{i=1}^{T} fp_{H,i}$, $TN_H = N - FP_H$ and $FN_H = P - TP_H$

Dipartimento di Matematica e Informatica

ProbFOIL+

- LScore$(H, x :: C)$ computes the local scoring function for the addition of clause $C(x) = x :: C$ to H
- The heuristic depends on the value of $x \in [0, 1]$.
- Find the value of x that maximizes the score

$$M(x) = \frac{TP_{H \cup C(x)} + mP/T}{TP_{H \cup C(x)} + FP_{H \cup C(x)} + m}.$$

- We need to compute $TP_{H \cup C(x)}$ and $FP_{H \cup C(x)}$, $tp_{H \cup C(x),i}$ and $fp_{H \cup C(x),i}$ as a function of x.

Dipartimento di Matematica e Informatica

ProbFOIL+

- $M(x)$ is a piecewise function where each piece is of the form

$$\frac{Ax + B}{Cx + D}$$

with A, B, C and D constants.

- The derivative of a piece is

$$\frac{dM(x)}{dx} = \frac{AD - BC}{(Cx + D)^2}$$

- It is either 0 or different from 0 everywhere in each interval so the maximum of $M(x)$ can only occur at the x_is values that are the endpoints of the intervals.
- Compute the value of $M(x)$ for each x_i and pick the maximum.
- Ordering the x_i values

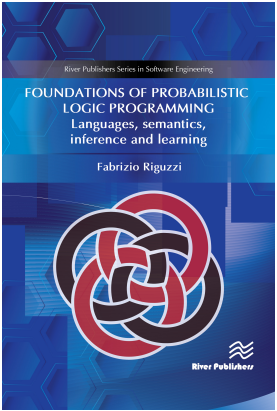
Dipartimento di Matematica e Informatica

ProbFOIL+

- ProbFOIL+ prunes refinements when
  - they cannot lead to a local score higher than the current best,
  - they cannot lead to a global score higher than the current best or
  - they are not significant, i.e., when they provide only a limited contribution.
- By adding a literal to a clause, the true positives and false positives can only decrease, so we can obtain an upper bound of the local score by setting the false positives to 0 and computing the m-estimate.
- By adding a clause to a theory, the true positives and false positives can only increase, so if the number of true positives of $H \cup C(x)$ is not larger than the true positives of H, the refinement $C(x)$ can be discarded.
- *significance test* based on the *likelihood ratio statistics*.

Conclusions

- Exciting field!
- Much is left to do:
  - Structure learning search strategies
  - Learning programs with continuous variables
  - Combining Deep Learning with PILP



Dipartimento di Matematica e Informatica

THANKS FOR LISTENING AND ANY QUESTIONS ?