Reasoning with Probabilistic Logic Languages

Fabrizio Riguzzi

Outline

1. Inference for PLP under DS
2. Explanation Based Inference Algorithm
3. Inference with Tabling
4. Inference in Simpler Settings
5. Approximate Inference
6. Inference by Conversion to Bayesian Networks
7. Learning Parameters
8. Directions for Future Work

Inference for PLP under DS

- Computing the probability of a query (no evidence)
- Explanation based:
  - find explanations for queries
  - make the explanations mutually exclusive
    - by means of an iterative splitting algorithm (Ailog2 [Poole, 2000])
    - by means of Binary Decision Diagrams (ProbLog [De Raedt et al., 2007], `cplint` [Riguzzi, 2007, Riguzzi, 2009], PITA [Riguzzi and Swift, 2010])
- Bayesian Network based:
  - Convert to BN
  - Use BN inference algorithms (CVE [Meert et al., 2009])
  - Lifted inference

ProbLog

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

- Distributions over facts

Definitions

- Composite choice κ: consistent set of atomic choices (C, θ, i) with $i \in \{1, 2\}$
- Explanation κ for a query Q: Q is true in every world compatible with κ (every world of ω_κ)
- A set of composite choices K covering with respect to Q: every world w in which Q is true is such that $w \in \omega_K$.
- Example:

$$K_1 = \{\{(C_1, \{X/david\}, 1)\}, \{(C_2, \{X/david\}, 1)\}\} \tag{1}$$

is covering for $sneezing(david)$.

Finding Explanations

- All explanations for the query are collected
- ProbLog: source to source transformation for facts, use of dynamic database
- `cplint`: meta-interpretation
- 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

$$f_K(\mathbf{Y}) = \bigvee_{\kappa \in K} \bigwedge_{(C,\theta,i) \in \kappa} (Y_{C\theta} = i)$$

  where $Y_{C\theta}$ is a random variable whose domain is $1, 2$ and $P(Y_{C\theta} = i) = P_0(C, i)$
- Binary domain: we use a Boolean variable $X_{C\theta}$ to represent $(Y_{C\theta} = 1)$
- $\neg X_{C\theta}$ represents $(Y_{C\theta} = 2)$

Example

A set of covering explanations for $\textit{sneezing}(\textit{david})$ is $K = \{\kappa_1, \kappa_2\}$

$\kappa_1 = \{(C_1, \{X/\textit{david}\}, 1)\}$

$\kappa_2 = \{(C_2, \{X/\textit{david}\}, 1)\}$

$K = \{\kappa_1, \kappa_2\}$

$f_K(\mathbf{Y}) = (Y_{C_1\{X/\textit{david}\}} = 1) \vee (Y_{C_1\{X/\textit{david}\}} = 1).$

$X_1 = (Y_{C_1\{X/\textit{david}\}} = 1)$

$X_2 = (Y_{C_2\{X/\textit{david}\}} = 1)$

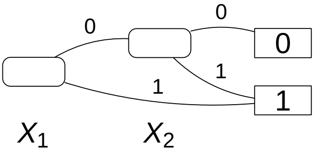
$f_K(\mathbf{X}) = X_1 \vee X_2.$

$P(f_K(\mathbf{X})) = P(X_1 \vee X_2)$

$P(f_K(\mathbf{X})) = P(X_1) + P(X_2) - P(X_1)P(X_2)$

- In order to compute the probability, we must make the explanations mutually exclusive
- [De Raedt et al., 2007]: Binary Decision Diagram (BDD)

Binary Decision Diagrams



$$f_K(\mathbf{X}) = X_1 \times f_K^{X_1}(\mathbf{X}) + \neg X_1 \times f_K^{\neg X_1}(\mathbf{X})$$

$$P(f_K(\mathbf{X})) = P(X_1)P(f_K^{X_1}(\mathbf{X})) + (1 - P(X_1))P(f_K^{\neg X_1}(\mathbf{X}))$$

$$P(f_K(\mathbf{X})) = 0.7 \cdot P(f_K^{X_1}(\mathbf{X})) + 0.3 \cdot P(f_K^{\neg X_1}(\mathbf{X}))$$

Probability from a BDD

- Dynamic programming algorithm [De Raedt et al., 2007]

function $\textsc{Prob}(n)$
  if n is a terminal note **then**
    return $value(n)$
  else
    return $\textsc{Prob}(child_1(n)) \times p(v(n)) + \textsc{Prob}(child_0(n)) \times (1 - p(v(node)))$
  end if
end function

Logic Programs with Annotated Disjunctions

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

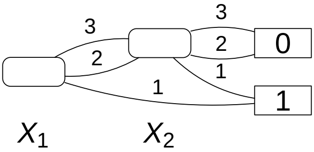
- Distributions over the head of rules
- More than two head atoms

Example

A set of covering explanations for $strong\_sneezing(david)$ is
$K = \{\kappa_1, \kappa_2\}$
$\kappa_1 = \{(C_1, \{X/david\}, 1)\}$
$\kappa_2 = \{(C_2, \{X/david\}, 1)\}$
$K = \{\kappa_1, \kappa_2\}$
$X_1 = X_{C_1\{X/david\}}$
$X_2 = X_{C_2\{X/david\}}$
$f_K(\mathbf{X}) = (X_1 = 1) \vee (X_2 = 1).$
$P(f_X) = P(X_1 = 1) + P(X_2 = 1) - P(X_1 = 1)P(X_2 = 1)$

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

Multivalued Decision Diagrams



$$f_K(\mathbf{X}) = \bigvee_{i \in |X_1|} (X_1 = i) \wedge f_K^{X_1=i}(\mathbf{X})$$

$$P(f_K(\mathbf{X})) = \sum_{i \in |X_1|} P(X_1 = i) P(f_K^{X_1=i}(\mathbf{X}))$$

$$f_K(\mathbf{X}) = (X_1 = 1) \wedge f_K^{X_1=1}(\mathbf{X}) + (X_1 = 2) \wedge f_K^{X_1=2}(\mathbf{X}) + (X_3 = 3) \wedge f_K^{X_3=1}(\mathbf{X})$$

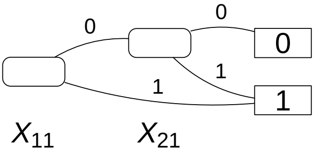
$$f_K(\mathbf{X}) = 0.3 \cdot P(f_K^{X_1=1}(\mathbf{X})) + 0.5 \cdot P(f_K^{X_1=2}(\mathbf{X})) + 0.2 \cdot P(f_K^{X_3=1}(\mathbf{X}))$$

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_1 having n values, we use $n-1$ Boolean variables $X_{11}, \dots, X_{1n-1}$
- $X_1 = i$ for $i = 1, \dots n-1$: $\overline{X_{11}} \wedge \overline{X_{12}} \wedge \dots \wedge \overline{X_{1i-1}} \wedge X_{1i}$,
- $X_1 = n$: $\overline{X_{11}} \wedge \overline{X_{12}} \wedge \dots \wedge \overline{X_{1n-1}}$.
- Parameters: $P(X_{11}) = P(X_1 = 1) \dots P(X_{1i}) = \frac{P(X_1=i)}{\prod_{j=1}^{i-1}(1-P(X_{1i-1}))}$.



Tabling

- Idea: maintain in a table both subgoals encountered in a query evaluation and answers to these subgoals
- If a subgoal is encountered more than once, the evaluation reuses information from the table rather than re-performing resolution against program clauses
- Important consequences:
  - Tabling ensures termination for a wide class of programs
  - Tabling can be used to evaluate programs with negation according to the Well-Founded Semantics
  - Tabling integrates closely with Prolog: a predicate p/n is evaluated using SLDNF by default, the predicate is made to use tabling by a directive such as *:- table p/n* that is added by the user or compiler.

Tabling for Probabilistic Inference

- PITA (Probabilistic Inference with Tabling and Answer subsumption) [Riguzzi and Swift, 2010] (a package of XSB Prolog)
- All the explanations for a goal have to be found
- It makes sense to store the explanations for subgoals with tabling
- Associate to each answer (ground atom) a BDD representing its explanations
- Combine BDDs by using the Boolean operators offered by BDD manipulating packages
- Library for manipulating BDD directly in Prolog (interface to CUDD)
- A BDD is represented in Prolog by an integer indicating the address of its root node
- Casting for integer-pointer conversion

Library Predicates

- `init`, `end`: for allocation and deallocation of a BDD manager
- `zero(-BDD)`, `one(-BDD)`, `and(+BDD1,+BDD2,-BDDO)`, `or(+BDD1,+BDD2,-BDDO)`, `not(+BDDI,-BDDO)`: BDD operations
- `add_var(+N_Val,+Probs,-Var)`: addition of a new multi-valued variable with `N_Val` values and parameters `Probs`
- `equality(+Var,+Value,-BDD)`: `BDD` represents `Var=Value`
- `ret_prob(+BDD,-P)`: returns the probability of the formula encoded by `BDD`

Tabling

- Add an extra argument to each atom for storing a BDD
- When an answer $p(\mathbf{x}, bdd)$ is found, *bdd* represents the explanations for $p(\mathbf{x})$
- If the program is range restricted, $p(\mathbf{x})$ is ground
- Use program transformation to obtain a Prolog program from an LPAD

Answer Subsumption

- A feature of tabling in XSB Prolog
- Use a lattice on terms to combine different answers for the same goal
- The bottom element and the join operator of the lattice have to be specified in the tabling directives
- E.g `:-table path(X,Y,or/3-zero/1)` means that, if two answers `path(a,b,bdd0)` and `path(a,b,bdd1)` are found, the single answer `path(a,b,bdd)` will be stored in the table where `or(bdd0,bdd1,bdd)`

Program Transformation

- *get_var_n(+R,+S,+Probs,-Var)* wraps *add_var/3*

$$
\begin{array}{l}
get\_var\_n(R, S, Probs, Var) \leftarrow \\
\quad (var(R, S, Var) \rightarrow \\
\quad\quad true \\
\quad ; \\
\quad\quad length(Probs, L), \\
\quad\quad add\_var(L, Probs, Var), \\
\quad\quad assert(var(R, S, Var)) \\
\quad).
\end{array}
$$

- Atom $A = p(\bar{t})$: $PITA(A) = p(\bar{t}, BDD)$
- Literal $\neg A$: $PITA(\neg A) = (PITA(A) \rightarrow one(BDD); not(BDD, BDD'))$

Program Transformation

The disjunctive clause

$C_r = H_1 : \alpha_1 \vee \ldots \vee H_n : \alpha_n \leftarrow L_1, \ldots, L_m.$

is transformed into the set of clauses $PITA(C_r)$

$$\begin{array}{rl}
PITA(C_r, 1) = PITA(H_1) \leftarrow & one(BB_0), \\
& PITA(L_1), and(BB_0, B_1, BB_1), \\
& \ldots, \\
& PITA(L_m), and(BB_{m-1}, B_m, BB_m), \\
& get_var_n(r, VC, [\alpha_1, \ldots, \alpha_n], Var), \\
& equality(Var, 1, BB), and(BB_m, BB, BDD).
\end{array}$$

$\ldots$

$$\begin{array}{rl}
PITA(C_r, n) = PITA(H_n) \leftarrow & one(BB_0), \\
& PITA(L_1), and(BB_0, B_1, BB_1), \\
& \ldots, \\
& PITA(L_m), and(BB_{m-1}, B_m, BB_m), \\
& get_var_n(r, VC, [\alpha_1, \ldots, \alpha_n], Var), \\
& equality(Var, n, BB), and(BB_m, BB, BDD).
\end{array}$$

Example

Clause

$strong\_sneezing(X) : 0.3 \vee moderate\_sneezing(X) : 0.5 \leftarrow flu(X).$

is translated into

$$
\begin{array}{ll}
strong\_sneezing(X, BDD) \leftarrow & one(BB_0), \\
& flu(X, B_1), and(BB_0, B_1, BB_1), \\
& get\_var\_n(1, [X], [0.3, 0.5, 0.2], Var), \\
& equality(Var, 1, BB), \\
& and(BB_1, BB, BDD). \\
moderate\_sneezing(X, BDD) \leftarrow & one(BB_0), \\
& flu(X, B_1), and(BB_0, B_1, BB_1), \\
& get\_var\_n(1, [X], [0.3, 0.5, 0.2], Var), \\
& equality(Var, 2, BB), \\
& and(BB_1, BB, BDD).
\end{array}
$$

Example

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

:-table path(X,Y,or/3-zero/1),edge(X,Y,or/3-zero/1).
path(X,X,One):-one(One).
path(X,Y,BDD):- path(X,Z,BDD0), edge(Z,Y,BDD1),
  and(BDD0,BDD1,BDD).
edge(a,b,BDD):-.
  get_var(3,[],[0.3,0.7],Var),
  equality(Var,0,BDD).
....
```

Query

- Query: `path(a,b)`

```
:-init,
  path('HGNC_620','HGNC_983',BDD),
  ret_prob(BDD,P),
  end.
```

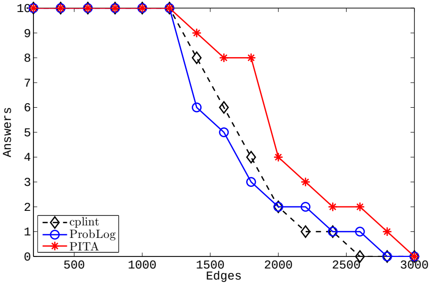
Experiments

- 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

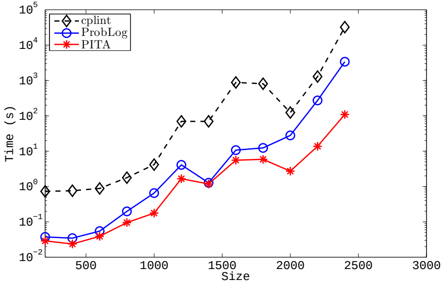
```
path(X,Y) :- path(X,Y,[X],Z).
path(X,Y,V,[Y|V]) :- arc(X,Y).
path(X,Y,V0,V1) :- arc(X,Z),append(V0,_S,V1),
\+ member(Z,V0),path(Z,Y,[Z|V0],V1).
arc(X,Y):-edge(X,Y).
arc(X,Y):-edge(Y,X).
edge('EntrezProtein_33339674','HGNC_620'):0.515062.
```

Dataset from [De Raedt et al., 2007]

Number of solved subgraphs

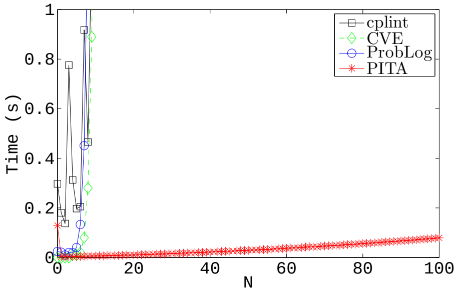


Average time



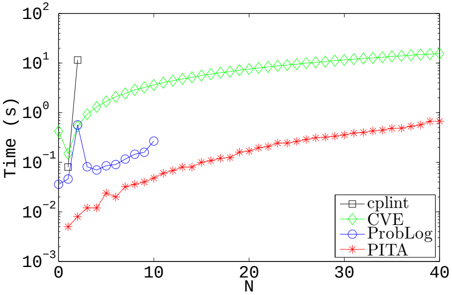
Game of dice

```
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).
```



Blood Type [Meert et al., 2009]

```
mchrom(Person,a):0.90 ; mchrom(Person,b):0.05 ; mchrom(Person,null):0.05 :-
  mother(Mother,Person), pchrom(Mother,a   ), mchrom(Mother,a   ).
mchrom(Person,a):0.49 ; mchrom(Person,b):0.49 ; mchrom(Person,null):0.02 :-
  mother(Mother,Person), pchrom(Mother,b   ), mchrom(Mother,a   ).
.....
pchrom(Person,a):0.90 ; pchrom(Person,b):0.05 ; pchrom(Person,null):0.05 :-
  father(Father,Person), pchrom(Father,a   ), mchrom(Father,a   ).
......
bloodtype(Person,a):0.90 ; bloodtype(Person,b):0.03 ; bloodtype(Person,ab):0.03 ;
  bloodtype(Person,null):0.04 :- pchrom(Person,a   ),mchrom(Person,a   ).
bloodtype(Person,a):0.03 ; bloodtype(Person,b):0.03 ; bloodtype(Person,ab):0.90 ;
  bloodtype(Person,null):0.04 :- pchrom(Person,b   ),mchrom(Person,a   ).
```



Simpler setting: PRISM

- The PRISM system consider a simpler setting
  - the probability of a conjunction (A,B) is computed as the product of the probabilities of A and B (independence assumption)
  - the probability of a disjunction (A;B) is computed as the sum of the probabilities of A and B (exclusiveness assumption).
- The program has to be written so that these requirements are met
- Not always possible

Simpler setting: PRISM

- Not all programs satisfy the two conditions
- Coin, Pea plants, Blood type both
- Russian roulette satisfies and
- Dice satisfies or
- Path does not satisfy any

```
p:- a,b.
a:0.3 ; b:0.4.
```

```
q:-a,b.
a:-c.
b:-c.
c:0.2.
```

- do not satisfy and: $P(p) = 0$, $P_{PRISM}(p) = 0.12$, $P(q) = 0.2$, $P_{PRISM}(q) = 0.04$

PRISM simpler setting

- PITA can be optimized for PRISM simpler setting
- The disjunctive clause
  $C_r = H_1 : \alpha_1 \vee \ldots \vee H_n : \alpha_n \leftarrow L_1, \ldots, L_m.$
  is transformed into the set of clauses $PITA'(C_r)$

$$\begin{aligned}
PITA'(C_r, 1) = PITA(H_1) \leftarrow \quad & one(BB_0),\\
& PITA(L_1), and(BB_0, B_1, BB_1), \ldots,\\
& PITA(L_m), and(BB_{m-1}, B_m, BB_m),\\
& equality([\alpha_1, \ldots, \alpha_n], 1, BB),\\
& and(BB_m, BB, B).
\end{aligned}$$

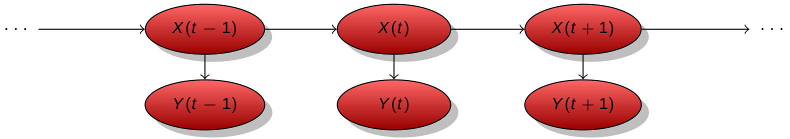
$\ldots$

$$\begin{aligned}
PITA'(C_r, n) = PITA(H_n) \leftarrow \quad & one(BB_0),\\
& PITA(L_1), and(BB_0, B_1, BB_1), \ldots,\\
& PITA(L_m), and(BB_{m-1}, B_m, BB_m),\\
& equality([\alpha_1, \ldots, \alpha_n], n, BB),\\
& and(BB_m, BB, B).
\end{aligned}$$

PRISM simpler setting

```
equality(Probs,N,P):- nth(N,Probs,P).
or(A,B,C):- C is A+B.
and(A,B,C):- C is A*B.
zero(0.0).
one(1.0).
not(P,P1):- P1 is 1-P.
ret_prob(P,P).
```

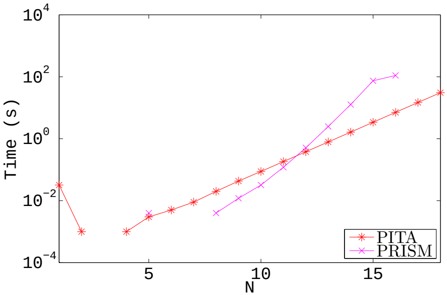
Hidden Markov Models




```
hmm(O):-hmm1(_,O).
hmm1(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.
```

HMM

Time for computing $P(hmm([a, \ldots, a]))$ as a function of sequence length



Exponential cost

Counting Explanations

- The optimized PITA can be used to count explanations when explanations for different goals can not be incompatible
- We have to modify `equality` as

  `equality(_Probs,_N,1).`
- In the Biomine network, series 1, the number of paths is

Edges	200	400	600	800	1000	1200	...
Explanations	10	42	380	1280	3,480	61,2140	...

- The definition of `path` implies that these are also the counts of the number of distinct paths from source to target that do not contain loops

Further Optimization

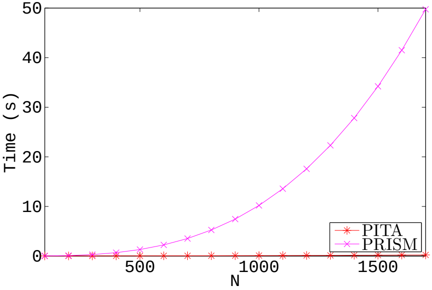
- [Christiansen and Gallagher, 2009] proposed to remove non-discriminating arguments, resulting in a program whose computation trees are isomorphic to those of the original program
- The results of the original program can be reconstructed from trace of the transformed program
- Useful with tabling: calls of a tabled predicate differing only in the non-discriminating arguments will merge into a single call
- Much smaller table and larger chance that the current call has a match in the table

```
hmm(O):-hmm(q1,O).
hmm(end,[]).
hmm(Q,[L|O]):-
  Q\= end,
  next_state(Q,Q1,S0),letter(Q,L,S0),
  hmm(Q1,O).
```

HMM

Time for computing $P(hmm([a, \ldots, a])$ as a function of sequence length

It should increase linearly



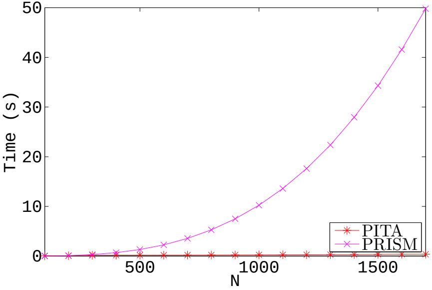
Computing the Viterbi Path

- Viterbi path: most probable explanation, its probability is the Viterbi probability

```
equality(R,S,Probs,N,e([(R,S,N)],P)):-
  nth(N,Probs,P).
or(e(E1,P1),e(_E2,P2),e(E1,P1)):- P1 > P2,!.
or(e(_E1,_P1),e(E2,P2),e(E2,P2)).
and(e(E1,P1),e(E2,P2),e(E3,P3)):-
  P3 is P1*P2,
  append(E1,E2,E3).
zero(e(null,0)).
one(e([],1)).
ret_prob(B,B).
```

HMM Viterbi Path

Time for computing the Viterbi path and probability of $hmm([a, \ldots, a])$ as a function of sequence length



Possibilistic Logic

- $\Pi(\phi)$: possibility of a logical formula ϕ, the degree of compatibility of ϕ with the available knowledge
- $N(\phi)$: necessity of a logical formula ϕ, the degree of certainty of ϕ given the available knowledge
- Relation: $N(\phi) = 1 - \Pi(\neg\phi)$
- Possibilistic Logic Program: set of formulas for the form (ϕ, α) where ϕ is a program clause

$$H \leftarrow L_1, \ldots, L_n.$$

- Meaning of (ϕ, α): $N(\phi) \geq \alpha$
- Inference: compute the maximum value of α such that $N(Q) \geq \alpha$ holds for a query Q.

Possibilistic Logic

- Inference rules:
  - $(\phi, \alpha), (\psi, \beta) \vdash (R(\phi, \phi), \min(\alpha, \beta))$ where $R(\phi, \phi)$ is the resolvent of ϕ and ψ
  - $(\phi, \alpha), (\phi, \beta) \vdash (\phi, \max(\alpha, \beta))$
- In PITA, interpret the formula $H : \alpha \leftarrow B_1, \ldots, B_n$ as $(H \leftarrow B_1, \ldots, B_n, \alpha)$

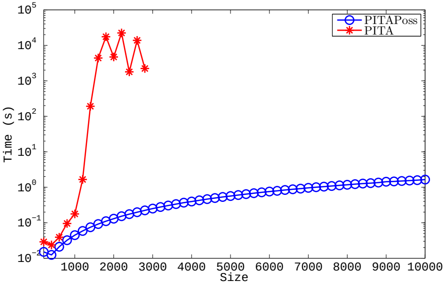
```
equality([P|T],_N,P).
or(A,B,C):- C is max(A,B).
and(A,B,C):- C is min(A,B).
zero(0.0).
one(1.0).
ret_prob(P,P).
```

PITA for Possibilistic Logic

- The possibilistic program

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

computes the least unsure path in a graph, i.e., the path with maximal weight, the weight of a path being the weight of its weakest link.



Approximate Inference

- Inference problem is #P hard
- For large models inference is intractable
- Approximate inference
  - Monte Carlo: draw samples of the truth value of the query
  - Iterative deepening: gives a lower and an upper bound
  - Compute only the best k explanations: branch and bound, gives a lower bound

Monte Carlo

- The disjunctive clause
  $C_r = H_1 : \alpha_1 \vee \ldots \vee H_n : \alpha_n \leftarrow L_1, \ldots, L_m.$
  is transformed into the set of clauses $MC(C_r)$
  $MC(C_r, 1) = H_1 \leftarrow L_1, \ldots, L_m, sample\_head(n, r, VC, NH), NH = 1.$
  $\ldots$
  $MC(C_r, n) = H_1 \leftarrow L_1, \ldots, L_m, sample\_head(n, r, VC, NH), NH = n.$
- Definition of `sample_head`:

```
:- table sample_head/4.
sample_head(NHead,R,VC,NH):- sample(NHead,NH),
```

- Sample truth value of query `Q`:

```
...
  (call(Q)->  NT1 is NT+1 ; NT1 =NT),
...
```

Monte Carlo

- The proportion of successes in a Bernoulli trial process is in the binomial proportion confidence interval

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

- Algorithm:
- $n := 0, nt := 0$
- Repeat
  - Test query n' times, nt' successes
  - $n := n + n', nt := nt + nt', \hat{p} = nt/n$
  - Compute interval size s
- until $s < \delta$
- return $\hat{p}, s$

Approximate Inference

- Iterative deepening: build the derivation tree only up to a certain depth,
- Completed derivations give a lower bound, completed plus incomplete derivations an upper bound
- How to do it efficiently?
- Best-k explanations: each time an explanation is found, update the set of explanations
- Cut a derivation if its probability falls below that of the k-th best explanation
- Sub case: best-1 explanation: Viterbi explanation

Inference by Conversion to Bayesian Networks

- Convert the program to a BN, perform inference on the BN with belief propagation, variable elimination, etc.
- Problem: grounding the program
- With function symbols, infinite grounding
- Even without function symbols, the grounding can be huge (exponential size)
- Most of the network is irrelevant to the query

Grounding

- Use a lifted inference algorithm
- Build only the relevant network and apply an inference algorithm
- Combination of the two approaches

Lifted Belief Propagation

- Belief propagation: nodes exchange messages, at convergence the marginal probability of each node can be extracted
- Correct for polytrees, approximate for general DAGs
- Lifted Belief Propagation: exploit the symmetries in the network to group nodes that exchange equal or similar messages into super nodes
- Perform belief propagation between super nodes taking into account the cardinalities of the messages

Building the Relevant Network

- Bayes Ball [Shachter, 1998]: algorithm for identifying the portion of a network that is relevant to query and evidence
- First-Order Bayes Ball [Meert et al., 2010]: lifted version of Bayes Ball
- Then apply a (lifted) inference algorithm

Learning Parameters

- Problem: given a set of interpretations, a program, find the parameters maximizing the likelihood of the interpretations (or of instances of a target predicate)
- Exploit the equivalence with BN to use BN learning algorithms
- The interpretations record the truth value of ground atoms, not of the choice variables
- Unseen data: relative frequency can't be used
- An Expectation-Maximization algorithm must be used:
  - 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

Learning Parameters

- [Thon et al., 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., 2008] independently proposed a similar algorithm
- CoPrEM [Gutmann et al., 2010] is the adaptation of EM to ProbLog

Learning Parameters

- EM can get trapped into local maxima
- Information Bottleneck: uses an evaluation function with a parameter
- When the parameter is 0, the maximum is easy to find
- When the parameter is 1, the function is the EM evaluation function, difficult to optimize
- Optimize the function with a deterministic annealing strategy: start with the parameter = 0 and then gradually increase it to 1, in the hope of finding an optimum better than EM
- Application to LPADs: Relational Information Bottleneck [Riguzzi and Di Mauro, 2010]

Directions for Future Works

- Approximate inference: iterative deepening, best-K
- Lifted inference for PLP: lifted variable elimination, lifted (loopy) belief propagation, first-order Bayes ball
- PLP structure learning

References I

- Christiansen, H. and Gallagher, J. P. (2009).
  Non-discriminating arguments and their uses.
  In *Logic Programming, 25th International Conference, ICLP 2009, Pasadena, CA, USA, July 14-17, 2009. Proceedings*, volume 5649 of *Lecture Notes in Computer Science*, pages 55–69. 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.

- Gutmann, B., Kimmig, A., Kersting, K., and De Raedt, L. (2010).
  Parameter estimation in ProbLog from annotated queries.
  Technical Report CW 583, Department of Computer Science, Katholieke Universiteit Leuven, Belgium.

References II

- Ishihata, M., Kameya, Y., Sato, T., and Minato, S. (2008).
  Propositionalizing the em algorithm by bdds.
  In *Late Breaking Papers of the 18th International Conf. on Inductive Logic Programming*, pages 44–49.

- Meert, W., Struyf, J., and Blockeel, H. (2009).
  CP-Logic theory inference with contextual variable elimination and comparison to bdd based inference methods.
  In *ILP 2009*.

- Meert, W., Taghipour, N., and Blockeel, H. (2010).
  First-order bayes-ball.
  In Balcázar, J. L., Bonchi, F., Gionis, A., and Sebag, M., editors, *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2010, Barcelona, Spain,*

References III

September 20-24, 2010, Proceedings, Part II, volume 6322 of *Lecture Notes in Computer Science*, pages 369–384. Springer.

Poole, D. (2000).
Abducing through negation as failure: stable models within the independent choice logic.
J. Log. Program., 44(1-3):5–35.

Riguzzi, F. (2007).
A top down interpreter for LPAD and CP-logic.
In *Congress of the Italian Association for Artificial Intelligence*, number 4733 in LNAI, pages 109–120. Springer.

References IV

- Riguzzi, F. (2009).
  Extended semantics and inference for the Independent Choice Logic.
  Logic Journal of the IGPL.
  to appear.

- Riguzzi, F. and Di Mauro, N. (2010).
  Applying the information bottleneck approach to SRL: Learning LPAD parameters.
  In *Inductive Logic Programming, 20th International Conference, Firenze, Italy, June 27-30, 2010*.

References V

- Riguzzi, F. and Swift, T. (2010).
  Tabling and Answer Subsumption for Reasoning on Logic Programs with Annotated Disjunctions.
  In Hermenegildo, M. and Schaub, T., editors, *Technical Communications of the 26th Int'l. Conference on Logic Programming (ICLP'10)*, volume 7 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 162–171, Dagstuhl, Germany. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

- Shachter, R. D. (1998).
  Bayes-ball: The rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams.
  In *In Uncertainty in Artificial Intelligence*, pages 480–487. Morgan Kaufmann.

References VI

Thon, I., Landwehr, N., and Raedt, L. D. (2008).
A simple model for sequences of relational state descriptions.
In Daelemans, W., Goethals, B., and Morik, K., editors, *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML/PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part II*, volume 5212 of *Lecture Notes in Computer Science*, pages 506–521. Springer.