Applications of Probabilistic Logic Programming

Luc De Raedt
ILP 2015, Kyoto
• Artificial Intelligence
  • the endeavor to make intelligent machines.

• Early days 1970s-1980s
  • focus on high-level reasoning
  • model driven

• limitations
  • assuming perfect knowledge about the world
  • assumed a deterministic world
  • no learning (from data)
Person: Pick up a big red block.
Computer: OK.
Person: Grasp the pyramid.
Computer: I don't understand which pyramid you mean.
Person (changing their mind): Find a block which is taller than the one you are holding and put it into the box.
Computer: By "it", I assume you mean the block which is taller than the one I am holding.
Computer: OK.
Person: What does the box contain?
Computer: The blue pyramid and the blue block.
Person: What is the pyramid supported by?
Computer: The box.
Person: How many blocks are not in the box?
Computer: Four of them.
Person: Is at least one of them narrower than the one which I told you to pick up?
Computer: Yes, the red cube.
etc.

*Reality is harder
- Details are important! For reasoning, planning...
- We cannot ignore position, orientation, shape, physics, etc...
- High-level concepts still useful (objects, properties and relations, background knowledge)*

Diagram adapted from Winograd, Understanding Natural Language (1972)
Historical perspective

- Next phase in AI 90s-00s
  - focus on processing low-level sensory information
  - *probabilistic graphical models, neural networks, svms* in vision, robotics, natural language processing ...
  - *data-driven* - uses data to learn
  - coping with *uncertainty*
- limitations
  - lack of high-level reasoning
  - does not deal with objects and relations
  - has a hard time to deal with knowledge
Now is the time

- to integrate these two views
  - the technology is getting ready
  - the applications demand it
  - and it is happening
- I believe the key challenge for Artificial Intelligence is to bridge the gap between low-level perception and high-level reasoning
Why is this relevant?

Bring me the tea pot and the sugar
The Robot Grasping Task

Object: pan
Part: handle
Task: P&P on table

Object: bottle
Part: top
Task: pass

Object: bottle
Part: middle
Task: Pass

Object: cup
Part: top
Task: Pour out
The Robot Grasping Task

Object: pan  
Part: handle  
Task: P&P on table

Object: bottle  
Part: top  
Task: pass

Object: bottle  
Part: middle  
Task: Pass

Object: cup  
Part: top  
Task: Pour out
This requires dealing with

- Structured environments
  - objects, and
  - relationships amongst them
- and possibly
  - using background knowledge
- cope with uncertainty
- learn from data
Probabilistic Logic Programming

Distribution Semantics [Sato, ICLP 95]:
probabilistic choices + logic program
→ distribution over possible worlds

OVERVIEW paper [Kimmig, De Raedt, MLJ 15]

e.g., PRISM, ICL, ProbLog, LPADs, CP-logic, ...

- multi-valued switches
- probabilistic alternatives
- probabilistic facts
- annotated disjunctions
- causal-probabilistic laws
Extensions of basic PLP

Distribution Semantics [Sato, ICLP 95]: probabilistic choices + logic program → distribution over possible worlds

- decisions
- constraints
- continuous RVs
- time & dynamics
- programming constructs
- semiring labels
- prob. rule learning
Overview

• Extensions motivated by applications
  • Distributional clauses — continuous distributions for use in robotics
  • Dynamics — for use in robotics and planning
  • Decision Theoretic ProbLog for biological network inference
  • Probabilistic rule learning — use with NELL
  • Semiring labels — kernels

• Focus on ProbLog / kLog line of research at KU Leuven
PART 1: Recap PLP
ProbLog by example:

A bit of gambling

- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)
ProbLog by example:

A bit of gambling

- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

probabilistic fact: heads is true with probability 0.4 (and false with 0.6)
A bit of gambling

- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

\[ 0.4 :: \text{heads}. \quad \text{annotated disjunction: first ball is red with probability 0.3 and blue with 0.7} \]
\[ 0.3 :: \text{col}(1,\text{red}); \quad 0.7 :: \text{col}(1,\text{blue}) \leftarrow \text{true.} \]
ProbLog by example:

A bit of gambling

- toss (biased) coin & **draw ball from each urn**
- win if (heads and a red ball) or (two balls of same color)

0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue) <- true.
0.2 :: col(2,red); 0.3 :: col(2,green);
    0.5 :: col(2,blue) <- true.

**annotated disjunction:** second ball is red with probability 0.2, green with 0.3, and blue with 0.5
ProbLog by example:

A bit of gambling

- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue) <- true.
0.2 :: col(2,red); 0.3 :: col(2,green);
     0.5 :: col(2,blue) <- true.

win :- heads, col(_,red).
ProbLog by example:

A bit of gambling

- toss (biased) coin & draw ball from each urn
- **win if (heads and a red ball) or (two balls of same color)**

0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue) <- true.
0.2 :: col(2,red); 0.3 :: col(2,green);
   0.5 :: col(2,blue) <- true.

\[ \text{win} :\text{heads, col(\_,red).} \]
\[ \text{win} :\text{col(1,C), col(2,C).} \]

**logical rule** encoding

**background knowledge**
ProbLog by example:

A bit of gambling

- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

<table>
<thead>
<tr>
<th>0.4 :: heads.</th>
<th>probabilistic choices</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.3 :: col(1,red); 0.7 :: col(1,blue) &lt;- true.</td>
<td></td>
</tr>
<tr>
<td>0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue) &lt;- true.</td>
<td></td>
</tr>
</tbody>
</table>

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
Questions

0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue) <- true.
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue) <- true.

win :- heads, col(_,red).
win :- col(1,C), col(2,C).

- Probability of \textbf{win}
- Probability of \textbf{win} given \textbf{col(2,green)}?
- Most probable world where \textbf{win} is true?

\textbf{marginal probability}

\textbf{conditional probability}

\textbf{MPE inference}
Possible Worlds

0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue) <- true.
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue) <- true.

win :- heads, col(_,red).
win :- col(1,C), col(2,C).

$0.4 \times 0.3 \times 0.3 \quad (1-0.4)\times0.3 \times 0.2 \quad (1-0.4)\times0.3 \times 0.3$
All Possible Worlds

0.024

0.036

0.056

0.084

0.036

0.054

0.084

0.126

0.060

0.090

0.140

0.210

0.036

0.054

0.084

0.126

0.060

0.090

0.140

0.210
Distribution Semantics
(with probabilistic facts)

\[ P(Q) = \sum_{F \cup R \models Q} \prod_{f \in F} p(f) \prod_{f \notin F} 1 - p(f) \]

query

subset of probabilistic facts

sum over possible worlds where \( Q \) is true

Prolog rules

probability of possible world

query

subset of probabilistic facts

sum over possible worlds where \( Q \) is true

Prolog rules

probability of possible world

[Sato, ICLP 95]
Alternative view: CP-Logic

throws(john).
0.5::throws(mary).

0.8 :: break <- throws(mary).
0.6 :: break <- throws(john).

probabilistic causal laws

\[
P(\text{break}) = 0.6 \times 0.5 \times 0.8 + 0.6 \times 0.5 \times 0.2 + 0.6 \times 0.5 + 0.4 \times 0.5 \times 0.8
\]
CP-logic [Vennekens et al.]

E.g., “throwing a rock at a glass breaks it with probability 0.3 and misses it with probability 0.7”

\[(\text{Broken}(G):0.3) \lor (\text{Miss} 0.7) \leftarrow \text{ThrowAt}(G).\]

Note that the actual non-deterministic event (“rock flying at glass”) is implicit.

Slides CP-logic courtesy Joost Vennekens
Semantics

\[(\text{Broken}(G) \; 0.3) \lor (\text{Miss} \; 0.7)\]

\[\overset{\text{ThrowAt}(G)}{\leftarrow}\]

Probability tree is an execution model of theory iff:
- Each tree-transition matches causal law
- The tree cannot be extended
- Each execution model defines the same probability distribution over final states

Slides CP-logic courtesy Joost Vennekens
Continuous Distributions
Distributional Clauses (DC)

- Discrete- and continuous-valued random variables
Distributional Clauses (DC)

• Discrete- and continuous-valued random variables

random variable with Gaussian distribution

\[ \text{length}(\text{Obj}) \sim \text{gaussian}(6.0, 0.45) :\text{type}(\text{Obj}, \text{glass}). \]
Discrete- and continuous-valued random variables

```
length(Obj) ~ gaussian(6.0,0.45) :- type(Obj,glass).
stackable(OBot,OTop) :-
  ≈length(OBot) ≥ ≈length(OTop),
  ≈width(OBot) ≥ ≈width(OTop).
```

Comparing values of random variables

[Gutmann et al, TPLP 11; Nitti et al, IROS 13]
Discrete- and continuous-valued random variables

\[ \text{length}(\text{Obj}) \sim \text{gaussian}(6.0, 0.45) \text{:- type(Obj,}\text{glass}). \]
\[ \text{stackable}(\text{OBot,OTop}) \text{:-}
\begin{align*}
&\approx \text{length}(\text{OBot}) \geq \approx \text{length}(\text{OTop}), \\
&\approx \text{width}(\text{OBot}) \geq \approx \text{width}(\text{OTop}). 
\end{align*} \]
\[ \text{ontype}(\text{Obj,plate}) \sim \text{finite}([0 : \text{glass}, 0.0024 : \text{cup},
\quad 0 : \text{pitcher}, 0.8676 : \text{plate},
\quad 0.0284 : \text{bowl}, 0 : \text{-serving},
\quad 0.1016 : \text{none}]) \text{:- obj(Obj), on(Obj,O2), type(O2,plate)}. \]

random variable with
discrete distribution

[Gutmann et al, TPLP 11; Nitti et al, IROS 13]
Discrete- and continuous-valued random variables

- length(Obj) ~ gaussian(6.0, 0.45) :- type(Obj, glass).
- stackable(OBot, OTop) :-
  ≃ length(OBot) ≥ ≃ length(OTop),
  ≃ width(OBot) ≥ ≃ width(OTop).
- ontype(Obj, plate) ~ finite([0 : glass, 0.0024 : cup,
  0 : pitcher, 0.8676 : plate,
  0.0284 : bowl, 0 : serving,
  0.1016 : none])
  :- obj(Obj), on(Obj, O2), type(O2, plate).
Distributional Clauses (DC)

- Defines a generative process (as for CP-logic)
- Tree can become infinitely wide
  - Sampling
- Well-defined under reasonable assumptions
- See Gutmann et al TPLP 11
Probabilistic Programs

- Distributional clauses / PLP similar in spirit
  - to e.g. BLOG, ... but embedded in existing logic and programming language
  - to e.g. Church but use of logic instead of functional programming ...

- natural possible world semantics and link with prob. databases.

- somewhat harder to do meta-programming
Markov Logic

Key differences

• programming language

• Dist. Sem. uses least-fix point semantics
  • can express transitive closure of relation

• this cannot be expressed in FOL (and Markov Logic), requires second order logic

• \( p(X,Y) :- p(X,Z), p(Z,Y). \)
Inference in PLP

- As in Prolog and logic programming
  - proof-based
- As in Answer Set Programming
  - model based
- As in Probabilistic Programming
  - sampling
Inference

Given:
- program
- queries
- evidence

Find:
- marginal probabilities
- conditional probabilities
- MPE state
- knowledge compilation

1. using proofs
2. using models

logical reasoning

data structure

probabilistic inference
Inference for DC

n ~ uniform([1, 2, 3, 4, 5, 6, 7, 8, 9, 10]).
color(X) ~ uniform([grey, blue, black]) ← material(X) ~ metal.
color(X) ~ uniform([black, brown]) ← material(X) ~ wood.
materila(X)~finite([0.3:wood,0.7:metal])←n~N,between(1,N,X).
drawn(Y) ~ uniform(L) ← n ~ N, findall(X, between(1, N, X), L).
size(X) ~ beta(2, 3) ← material(X) ~ metal.
size(X) ~ beta(4, 2) ← material(X) ~ wood.
Given a goal $G$ and the global variables $w_q^{(i)}, iq, x_P^{(i)}$, applying a rule produces a new goal $G'$ and modifies the global variables:

1. $G'$ is the new goal obtained from $G$ using a kind of SLD-resolution step;
2. if a new variable $r$ is sampled with value $v$,
   - set $w_q^{(i)} \leftarrow w_q^{(i)} \frac{p(r=v|x_P^{(i)})}{g(r=v|x_P^{(i)})}$ (based on LW) and
   - $x_P^{(i)} \leftarrow x_P^{(i)} \cup \{r = v\}$. 
   In addition, if $r = Val \in iq$ with $r$ grounded, then:
   - $iq \leftarrow iq\theta$ with $\theta = \{Val = v\}$.
3. if a new atom $a$ is proved set $x_P^{(i)} \leftarrow x_P^{(i)} \cup \{a\}$.
1: (color(2) \sim= \text{black}); w_q^{(i)} = 1; x^{P(i)} = \emptyset

2b on (7):
2: (material(2) \sim= \text{metal}, \text{sample(color(2), } D_{\text{color(2)}}, \text{color(2) } \sim= \text{black}) ; w_q^{(i)} = 1; x^{P(i)} = \emptyset

2b on (9):
3: (n \sim= N, \text{between(1, } N, 2), \text{sample(material(2), } D_{\text{material(2)}}, \text{material(2) } \sim= \text{metal, sample(color(2), } D_{\text{color(2)}}, \text{color(2) } \sim= \text{black}) ; w_q^{(i)} = 1; x^{P(i)} = \emptyset

2b on (6):
4: (\text{sample(n, } D_n), n \sim= N, \text{between(1, } \sim(n), 2), \text{sample(material(2), } D_{\text{material(2)}}, \text{material(2) } \sim= \text{metal, sample(color(2), } D_{\text{color(2)}}, \text{color(2) } \sim= \text{black}) ; w_q^{(i)} = 1; x^{P(i)} = \emptyset

3b:
5: (n \sim= 3, \text{between(1, } 3, 2), \text{sample(material(2), } D_{\text{material(2)}}, \text{material(2) } \sim= \text{metal, sample(color(2), } D_{\text{color(2)}}, \text{color(2) } \sim= \text{black}) ; w_q^{(i)} = 1; x^{P(i)} = \{n = 3\}

2a followed by 1a
6: (\text{sample(material(2), } D_{\text{material(2)}}, \text{material(2) } \sim= \text{metal, sample(color(2), color(2) } \sim= \text{black}) ; w_q^{(i)} = 1; x^{P(i)} = \{n = 3\}

3b:
7: (\text{material(2) } \sim= \text{metal, sample(color(2), } D_{\text{color(2)}}, \text{color(2) } \sim= \text{black})

\quad w_q^{(i)} = 1; x^{P(i)} = \{n = 3, \text{material(2) } = \text{wood}\}

\text{fail, backtracking to 1}
can cope with evidence like $\text{color}(1) = \text{color}(2)$
and $\text{size}(1) = 0.356, \text{size}(1)=\text{size}(2), \ldots$
outperforms BLOG ... unification + LW
Affordances with DCs
Affordances

- Model captures action opportunities
  - What can one do with an object?
- Three main aspects:
  - Object (properties):
    - Measured from perceptual devices
    - shape, size, ...
  - Action:
    - Applied physical manipulation
    - Tap, Push, Grab
  - Effects:
    - Measurable features after action
    - displacement, orientation, ...

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(O, A)$</td>
<td>$E$</td>
<td>Effect prediction</td>
</tr>
<tr>
<td>$(O, E)$</td>
<td>$A$</td>
<td>Action recognition/planning</td>
</tr>
<tr>
<td>$(A, E)$</td>
<td>$O$</td>
<td>Object recognition/selection</td>
</tr>
</tbody>
</table>
Learning relational affordances

Learn probabilistic model

From two object interactions
Generalize to N

Moldovan et al. ICRA 12, 13, 14, PhD 15
Learning relational affordances

Learn probabilistic model

From two object interactions
Generalize to $N$

*Moldovan* et al. ICRA 12, 13, 14, PhD 15
Learning relational affordances between two objects (learnt by experience)

Right Arm

Examples
Learning relational affordances between two objects (learnt by experience)

Right Arm

Examples
What is an affordance?

Clip 8: Relational O before (l), and E after the action execution (r).

Table 1: Example collected O, A, E data for action in Figure 8

<table>
<thead>
<tr>
<th>Object Properties</th>
<th>Action</th>
<th>Effects</th>
</tr>
</thead>
<tbody>
<tr>
<td>$shape_{O_{Main}}$ : sprism</td>
<td>$tap(10)$</td>
<td>$displX_{OMain} : 10.33cm$</td>
</tr>
<tr>
<td>$shape_{O_{Sec}}$ : sprism</td>
<td></td>
<td>$displY_{OMain} : -0.68cm$</td>
</tr>
<tr>
<td>$distX_{O_{Main},O_{Sec}} : 6.94cm$</td>
<td></td>
<td>$displX_{O_{Sec}} : 7.43cm$</td>
</tr>
<tr>
<td>$distY_{O_{Main},O_{Sec}} : 1.90cm$</td>
<td></td>
<td>$displY_{O_{Sec}} : -1.31cm$</td>
</tr>
</tbody>
</table>

- Formalism — related to STRIPS but models delta
- but also joint probability model over A, E, O
Learning relational affordances

Clip 4: Pipeline for table-top two-arm object manipulation.

- 1a) learn a Linear Continuous Gaussian (LCG) Bayesian Network (BN) from single arm and simultaneous two-arm exploratory data,
- 1b) from the LCG model, build the two-arm continuous domain relational affordance model in a PPL,
- 2) build a state transition model from the relational affordance model, and
- 3) infer best action to execute to reach goal (step repeated until goal reached).
Remaining challenge

- Learn DC model directly
- Work on planning with DC (Nitti et al., ECML, EWRL '15)
Occluded Object Search

• How to achieve a specific configuration of objects on the shelf?

• Where’s the orange mug?

• Where’s something to serve soup in?

• Models of objects and their spatial arrangement
Dynamics
Dynamics: Evolving Networks

- **Travian**: A massively multiplayer real-time strategy game
  - Commercial game run by TravianGames GmbH
  - ~3.000.000 players spread over different “worlds”
  - ~25.000 players in one world

[Thon et al. MLJ 11]
World Dynamics

Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

Can we build a model
of this world?
Can we use it for playing
better?

[Thon, Landwehr, De Raedt, ECML08]
World Dynamics

Fragment of world with

- ~10 alliances
- ~200 players
- ~600 cities

alliances color-coded

Can we build a model of this world?
Can we use it for playing better?

[Thon, Landwehr, De Raedt, ECML08]
World Dynamics

Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

Can we build a model of this world?
Can we use it for playing better?

[Thon, Landwehr, De Raedt, ECML08]
World Dynamics

Fragment of world with

~10 alliances  
~200 players  
~600 cities

alliances color-coded

Can we build a model of this world?  
Can we use it for playing better?

[Thon, Landwehr, De Raedt, ECML08]
World Dynamics

Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

Can we build a model of this world?
Can we use it for playing better?

[Thon, Landwehr, De Raedt, ECML08]
World Dynamics

Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

Can we build a model of this world?
Can we use it for playing better?

[Thon, Landwehr, De Raedt, ECML08]
Causal Probabilistic Time-Logic (CPT-L)

how does the world change over time?

[Thon et al, MLJ 11]
Causal Probabilistic Time-Logic (CPT-L)

how does the world change over time?

\[ 0.4 \cdot \text{conquest}(\text{Attacker}, C) ; \ 0.6 \cdot \text{nil} \leftarrow \text{city}(C, \text{Owner}), \text{city}(C2, \text{Attacker}), \text{close}(C, C2). \]

if \textbf{cause} holds at time \( T \)

[Thon et al, MLJ 11]
Causal Probabilistic Time-Logic (CPT-L)

how does the world change over time?

one of the **effects** holds at time $T+1$

$$0.4::\text{conquest}(\text{Attacker}, C); 0.6::\text{nil} \leftarrow \text{city}(C, \text{Owner}), \text{city}(C2, \text{Attacker}), \text{close}(C, C2).$$

if **cause** holds at time $T$

[Thon et al, MLJ '11]
Causal Probabilistic Time-Logic (CPT-L)

how does the world change over time?

one of the effects holds at time $T+1$

$$0.4::\text{conquest}(\text{Attacker}, C); 0.6::\text{nil} \leftarrow \text{city}(C, \text{Owner}), \text{city}(C2, \text{Attacker}), \text{close}(C, C2).$$

if cause holds at time $T$

[Thon et al, MLJ 11]
Limitations CPT-L

Inference slow / scalability

• uses knowledge compilation method
• compile formula for \( P(I_{t+1}|I_{[0,t]}) \)
• exponential in number of time steps

Learning: fully observable

No continuous distributions
Relational Tracking

- Track people or objects over time? Even if temporarily hidden?
- Recognize activities?
- Infer object properties?
Relational State Estimation over Time

Magnetism scenario

- object tracking
- category estimation from interactions

Box scenario

- object tracking even when invisible
- estimate spatial relations

[Nitti et al, IROS 13]
Magnetic scenario

- 3 object types: magnetic, ferromagnetic, nonmagnetic

- Nonmagnetic objects do not interact

- A magnet and a ferromagnetic object attract each other

- Magnetic force that depends on the distance

- If an object is held magnetic force is compensated.
Magnetic scenario

- 3 object types: magnetic, ferromagnetic, nonmagnetic

\[ \text{type}(X)_t \sim \text{finite}([1/3:\text{magnet},1/3:\text{ferromagnetic},1/3:\text{nonmagnetic}]) \leftarrow \text{object}(X). \]

- 2 magnets attract or repulse

\[ \text{interaction}(A,B)_t \sim \text{finite}([0.5:\text{attraction},0.5:\text{repulsion}]) \leftarrow \text{object}(A), \text{object}(B), A < B, \text{type}(A)_t = \text{magnet}, \text{type}(B)_t = \text{magnet}. \]

- Next position after attraction

\[ \text{pos}(A)_{t+1} \sim \text{gaussian}(\text{middlepoint}(A,B)_t, \text{Cov}) \leftarrow \text{near}(A,B)_t, \text{not}(\text{held}(A)), \text{not}(\text{held}(B)), \text{interaction}(A,B)_t = \text{attr}, \frac{c}{\text{dist}(A,B)_t^2} > \text{friction}(A)_t. \]

\[ \text{pos}(A)_{t+1} \sim \text{gaussian}(\text{pos}(A)_t, \text{Cov}) \leftarrow \text{not}(\text{attraction}(A,B)). \]
Speed 0x

Queries
(updated every 5 steps)

on(X,Y):
[1.0:(3,(table)), 1.0:(4,(table))]
inside(X,Y):
[]
tr_inside(X,Y):
[]

Particles

Box ID=4  Cube ID=3
Speed 0x

Queries
(updated every 5 steps)

on(X,Y):
[1.0:(3,(table)), 1.0:(4,(table))]
inside(X,Y):
[]
tr_inside(X,Y):
[]

Particles

Box ID=4  Cube ID=3
DC Particle Filter (DCPF)

- Dynamic Distributional Clauses
- Particle Filter

Goal

Flexible (relational) state representation
Fast inference (state estimation) in general models

“A particle filter for hybrid relational domains” IROS 2013
D. Nitti, T. De Laet, L. De Raedt
Dynamic Distributional Clauses

Prior distribution $p(x_0)$
State transition model $p(x_t|x_{t-1}, u_t)$
Measurement model $p(z_t|x_t)$
Other rules: $p(x'_t|x''_t)$
Particle Filter  
(Sequential Monte Carlo)

- Based on sampling → approximate inference
- Particles (samples) to represent \( \text{bel}(x_t) \)
Classical Particle Filter vs DCPF

- **Classical PF**
  - Fixed set of random variables
  - Update the entire state

- **DCPF**
  - **Adaptive state (particle):** the number of facts / random variables can change over time
  - Particles are partial interpretations
  - Expressive language

\[
X_t^{(i)} = \begin{array}{c}
1.1 \\
2.3 \\
10.3
\end{array}
\quad X_{t+1}^{(i)} = \begin{array}{c}
1.2 \\
2.1 \\
10.5
\end{array}
\]

- Pos(1)=(0, 3)
- Pos(2)=(0, 1)
- right(X,Y)
- near(X,Y)
- interaction(X,Y)
- type(X) \sim \{1/3:magnet,...\}
- [...]
Optimized inference: partial state

**Distributional Clauses Particle Filter (DCPF)**

- **Sampled**
  - Pos(1) = (0, 3)
  - Pos(2) = (0, 1)
  - right(X, Y)
  - near(X, Y)
  - interaction(X, Y)
  - type(X) \sim [1/3: magnet, ...]
  - ...

- **Marginalized**
  - Pos(1) = (0, 2)
  - Pos(2) = (0, 1)
  - near(1, 2) = true
  - type(1) = nonmagnetic
  - right(X, Y)
  - near(X, Y)
  - interaction(X, Y)
  - type(X) \sim [1/3: magnet, ...]
  - ...

**Classical particle filter**

- Pos(1) = (0, 3)
  - Pos(2) = (0, 1)
  - near(1, 2) = false
  - near(2, 1) = false
  - interaction(1, 2) = none
  - type(1) = nonmagnetic
  - type(2) = nonmagnetic
  - ...

- Pos(1) = (0, 2)
  - Pos(2) = (0, 1)
  - near(1, 2) = true
  - near(2, 1) = true
  - interaction(1, 2) = none
  - type(1) = nonmagnetic
  - type(2) = nonmagnetic
  - [...]

---

Classical particle filter
Inference in DCPF

Two steps:

Query $p(z_{t+1} | x_{t+1})$ (weighting + part of sampling step)
Query $p(x_{t+1} | x_t, u_{t+1})$ (to complete the sampling step)

Particles are partial interpretations

$bel(x_t)$ fully represented by $\{x_t^{(i)}\} \cup \text{Program}$

History $\{x_{0:t-1}^{(i)}\}$ not necessary

Issue: particles (interpretations) may grow till becoming complete
Ongoing Work

- Parameter learning [Nitti, ICRA 2014]
- Integrate with planning [Nitti ECML, EWRL 15]
- Larger Experiments
- Connection to probabilistic programming
- Applications in robotics (also to learn affordances)
ProbLog for activity recognition from video

- Separation between low-level events (LLE) and high-level events (HLE)
  - LLE: *walking, running, active, inactive, abrupt*
  - HLE: *meeting, moving, fighting, leaving_object*
- Probabilistic Logic approach: *Event Calculus in ProbLog* (Prob-EC) to infer the high-level events from an algebra of low-level events.
- Example:
  
  \[
  \text{initiatedAt(} \text{fighting}(P_1, P_2) = \text{true}, T) \leftarrow \\
  \text{happensAt(} \text{abrupt}(P_1), T), \\
  \text{holdsAt(} \text{close}(P_1, P_2, 44) = \text{true}, T), \\
  \text{not happensAt(} \text{inactive}(P_2), T). \\
  \]

[Skarlatidis et al, TPLP 13]
Biomine Network
Notch receptor processing
BiologicalProcess
GO:GO:0007220

presenilin 2
Gene
EntrezGene:81751
Biomine Network -participates_in 0.220 BiologicalProcess

-participates_in 0.220

Gene

presenilin 2 Gene EntrezGene:81751
Graphs & Randomness

ProbLog, Phenetic, Prism, ICL, Probabilistic Databases, ...

• all based on a “random graph” model

Stochastic Logic Programs, ProPPR, PCFGs, ...

• based on a “random walk” model

• connected to PageRank
Phenetic

Causes: Mutations
- All related to similar phenotype
- Effects: Differentially expressed genes
- 27,000 cause effect pairs

Interactions network:
- 3063 nodes
- Genes
- Proteins
- 16794 edges
- Molecular interactions
- Uncertain

Goal: connect causes to effects through common subnetwork
- = Find mechanism
- Techniques:
  - DTProbLog [Van den Broeck]
  - Approximate inference

DT-ProbLog
decision theoretic version

Can we find the mechanism connecting causes to effects?

[De Maeyer et al., Molecular Biosystems 13, NAR 15]
Figure 1. Overview of PheNetic, a web service for network-based interpretation of ‘omics’ data. The web service uses as input a genome wide interaction network for the organism of interest, a user generated molecular profiling data set and a gene list derived from these data. Interaction networks for a wide variety of organisms are readily available from the web server. Using the uploaded user-generated molecular data the interaction network is converted into a probabilistic network: edges receive a probability proportional to the levels measured for the terminal nodes in the molecular profiling data set. This probabilistic interaction network is used to infer the sub-network that best links the genes from the gene list. The inferred sub-network provides a trade-off between linking as many genes as possible from the gene list and selecting the least number of edges.
Viral Marketing

Which advertising strategy maximizes expected profit?

Lenny
Moe
Apu
Seymour
Homer
Bart
Marge
Ralph
Lisa
Maggie

[Van den Broeck et al, AAAI 10]
Viral Marketing

**Which strategy gives the maximum expected utility?**

*Viral Marketing* [Van den Broeck et al, AAAI 10]
DTProbLog

person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
DTProbLog

? :: marketed(P) :- person(P).

decision fact: true or false?

person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
DTProbLog

? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

bues(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
bues(X) :- marketed(X), buy_marketing(X).

probabilistic facts + logical rules

person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
DTProbLog

? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).

utility facts: cost/reward if true

person(1).
person(2).
person(3).
person(4).
frend(1,2).
frend(2,1).
frend(2,4).
frend(3,4).
frend(4,2).
DTProbLog

? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).

person(1).
person(2).
person(3).
person(4).

friend(1,2).
frend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).

person(1).
person(2).
person(3).
person(4).

friend(1,2).
frend(2,1).
frend(2,4).
frend(3,4).
frend(4,2).
DTProbLog

? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).

marketed(1)   marketed(3)
DTProbLog

? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).

marketed(1)    marketed(3)
bmt(2,1)        bt(2,4)    bm(1)

person(1).
person(2).
person(3).
person(4).
friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
DTProbLog

? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).

marketed(1) marketed(3)
bt(2,1) bt(2,4) bm(1)
buys(1) buys(2)

person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
DTProbLog

? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).

utility = $-3 + -3 + 5 + 5 = 4$

probability = 0.0032

marketed(1) marketed(3)
bt(2,1) bt(2,4) bm(1)
buys(1) buys(2)
DTProbLog

? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).

utility = −3 + −3 + 5 + 5 = 4
probability = 0.0032

world contributes 0.0032×4 to expected utility of strategy
DTProbLog

? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).

person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).

task: find strategy that maximizes expected utility
solution: using ProbLog technology
A true application

A tool for Computational Biology
Based on decision theoretic variation of ProbLog
ProbLog / Prob. Programming for prototyping
More specialised inference engine was needed
also some special purpose approximations
Probabilistic Rule learning
Information Extraction in NELL

### Recently-Learned Facts

<table>
<thead>
<tr>
<th>instance</th>
<th>iteration</th>
<th>date learned</th>
<th>confidence</th>
</tr>
</thead>
<tbody>
<tr>
<td>kelly_andrews is a female</td>
<td>826</td>
<td>29-mar-2014</td>
<td>98.7</td>
</tr>
<tr>
<td>investment_next_year is an economic sector</td>
<td>829</td>
<td>10-apr-2014</td>
<td>95.3</td>
</tr>
<tr>
<td>shibenik is a geopolitical entity that is an organization</td>
<td>829</td>
<td>10-apr-2014</td>
<td>97.2</td>
</tr>
<tr>
<td>quality_web_design_work is a character trait</td>
<td>826</td>
<td>29-mar-2014</td>
<td>91.0</td>
</tr>
<tr>
<td>mercedes_benz_cls_by_carlsson is an automobile manufacturer</td>
<td>829</td>
<td>10-apr-2014</td>
<td>95.2</td>
</tr>
<tr>
<td>social_work is an academic program at the university rutgers university</td>
<td>827</td>
<td>02-apr-2014</td>
<td>93.8</td>
</tr>
<tr>
<td>dante wrote the book the_divine_comedy</td>
<td>826</td>
<td>29-mar-2014</td>
<td>93.8</td>
</tr>
<tr>
<td>willie_aames was born in the city los_angeles</td>
<td>831</td>
<td>16-apr-2014</td>
<td>100.0</td>
</tr>
<tr>
<td>kitt_peak is a mountain in the state or province arizona</td>
<td>831</td>
<td>16-apr-2014</td>
<td>96.9</td>
</tr>
<tr>
<td>greenwich is a park in the city london</td>
<td>831</td>
<td>16-apr-2014</td>
<td>100.0</td>
</tr>
</tbody>
</table>

Instances for many different relations

Degree of certainty

NELL: http://rtw.ml.cmu.edu/rtw/
Rule learning in NELL (I)

- Original approach
- Make probabilistic data deterministic
- run classic rule-learner (variant of FOIL)
- re-introduce probabilities on learned rules and predict
Rule learning in NELL (2)

• Newer Page Rank Based Approach (Cohen et al.) -- ProPPR
  • Change the underlying model, from random graph / database to random walk one;
  • No longer “degree of belief” assigned to facts;
  • more like stochastic logic programs
  • Learn rules / parameters
Probabilistic Rule Learning

• Learn the rules directly in a PLP setting
• Generalize relational learning and inductive logic programming directly towards probabilistic setting
• Traditional rule learning/ILP as a special case
• Apply to probabilistic databases like NELL
• ILP 10, IJCAI 15
Pro Log

surfing(X) :- not pop(X) and windok(X).
surfing(X) :- not pop(X) and sunshine(X).

pop(e1). windok(e1). sunshine(e1).

?-surfing(e1). e
no

B U H \models e (H does not cover e)

An ILP example
ProbLog

a probabilistic Prolog

\[ p1:: \text{surfing}(X) :- \text{not pop}(X) \text{ and } \text{windok}(X). \]

\[ p2:: \text{surfing}(X) :- \text{not pop}(X) \text{ and } \text{sunshine}(X). \]

\[ 0.2:: \text{pop}(e1). \]

\[ 0.7:: \text{windok}(e1). \]

\[ 0.6:: \text{sunshine}(e1). \]

\[ ?- P(\text{surfing}(e1)). \]

\[ \text{gives } (1-0.2) \times 0.7 \times p1 + (1-0.2) \times 0.6 \times (1-0.7) \times p2 = P(B \cup H \mid= e) \]

\[ \text{not pop } \times \text{windok } \times p1 + \text{not pop } \times \text{sunshine } \times (\text{not windok}) \times p1 \]

probability that the example is covered
Inductive Probabilistic Logic Programs

Given

a set of example facts $e \in E$ together with the probability $p$ that they hold

a background theory $B$ in ProbLog

a hypothesis space $L$ (a set of clauses)

Find

$$\arg \min_H \text{loss}(H, B, E) = \arg \min_H \sum_{e_i \in E} |P_s(B \cup H \models e) - p_i|$$
Observations

Propositional versus first order

- traditional rule learning = propositional
- inductive logic programming = first order

Deterministic case

- all probabilities 0 or 1
- traditional rule learning / ILP as special case
Analysis

1) the true positive part

\[ t_{p,i} = \min(p_{i}, p_{H,i}) \]

2) the true negative part

\[ t_{n,H,i} = \min(n_{i}, n_{H,i}) \]

3) the false positive part

\[ f_{p,H,i} = \max(0, n_{i} - t_{n,H,i}) \]

4) the false negative part

\[ f_{n,H,i} = \max(0, p_{i} - t_{p,H,i}) \]

Example

\[ p_{i} \]

under-estimate

\[ t_{n,H,i} \]

\[ f_{n,H,i} \]

\[ p_{H,i} \]

\[ t_{p,H,i} \]

target

probability

\[ 0 \]

\[ 1 \]

over-estimate

\[ t_{n,H,i} \]

\[ f_{p,H,i} \]

\[ p_{H,i} \]

\[ t_{p,H,i} \]
Analysis
Rule learning

Interesting properties

- adding a rule is monotonic, this can only increase the probability of an example
- adding a condition to a rule is anti-monotonic, this can only decrease the probability of an example
- several rules may be needed to cover an example
  - use all examples all of the time (do not delete them while learning), do not forget the positives
  - disjoint sum problem
ProbFOIL

Quinlan’s well-known FOIL algorithm combined with ProbLog and probabilistic examples and background knowledge

Essentially a vanilla sequential covering algorithm with m-estimate as local score and accuracy as global score.
Criteria

\[
\text{precision} = \frac{TP}{TP + FP}
\]

\[
\text{m-estimate} = \frac{TP + m \cdot \frac{P}{N}}{TP + FP + m}
\]

\[
\text{recall} = \frac{TP}{TP + FN}
\]

\[
\text{accuracy} = \frac{TP + TN}{TP + TN + FP + FN}
\]

Avoiding overfitting using significance test
We now present our algorithm for learning probabilistic clauses, which is a generalization of the mFOIL rule learning algorithms. The outline of the algorithm is shown as Algorithm 1. It follows the typical separate-and-conquer approach (also known as sequential covering) that is commonly used in rule learning algorithms. The outer loop of the algorithm, labeled ProbFOIL, starts from an empty set of clauses and repeatedly adds clauses to the hypothesis until no more improvement is observed with respect to a global scoring function. The clause to be added is obtained by the function LearnRule, which greedily searches for the clause that maximizes a local scoring function.

The resulting algorithm is very much like the standard rule-learning algorithm known from the literature (cf. [16,31]).

Algorithm 1

The ProbFOIL$^+$ learning algorithm

1: function ProbFOIL$^+$ (target)  \hfill $\triangleright$ target is the target predicate
2:     $H := \emptyset$
3:     while true do
4:         clause := LearnRule($H$, target) \hfill $\triangleright$ Start with an empty (probabilistic) body
5:         if GlobalScore($H$) < GlobalScore($H \cup \{\text{clause}\}$) then
6:             $H := H \cup \{\text{clause}\}$
7:         else
8:             return $H$
9:  
10: function LearnRule($H$, target) \hfill $\triangleright$ Grow rule
11:     candidates := \{x :: target $\leftarrow$ true\} \hfill $\triangleright$ Generate all refinements
12:     bestrule := (x :: target $\leftarrow$ true) \hfill $\triangleright$ Reject unsuited refinements
13:     while candidates $\neq \emptyset$ do
14:         nextcandidates := \emptyset
15:         for all x :: target $\leftarrow$ body $\in$ candidates do
16:             for all literal $\in \rho$(target $\leftarrow$ body) do
17:                 if not RejectRefinement($H$, bestrule, x :: target $\leftarrow$ body) then
18:                     nextcandidates := nextcandidates $\cup$ \{x :: target $\leftarrow$ body $\land$ literal\}
19:                     if LocalScore ($H$, x :: target $\leftarrow$ body $\land$ literal) $>$ LocalScore($H$, bestrule) then
20:                         bestrule := (x :: target $\leftarrow$ body $\land$ literal) \hfill $\triangleright$ Update best rule
21:         candidates := nextcandidates
22:     return bestrule
Extended rule learning

Learn rules with probability $x::\text{head} :- \text{body}$

What changes?

- value of $x$ determines prob. of coverage of example

$x = 1$  

$x = 0$
Extended rule learning

Express local score as a function of $x$

Compute optimal value of $x$
In order to test probabilistic rule learning for facts extracted by NELL, we used the NELL athlete dataset, which has already been used in the context of meta-interpretive learning of higher-order dyadic Datalog [36]. This dataset contains 10130 facts. The number of facts per predicate is listed in Table 5. The unary predicates in this dataset are deterministic, whereas the binary predicates have a probability attached.

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Facts</th>
</tr>
</thead>
<tbody>
<tr>
<td>athletecoach(person,person)</td>
<td>18</td>
</tr>
<tr>
<td>athleteplayssport(person,sport)</td>
<td>1921</td>
</tr>
<tr>
<td>athleteplaysinleague(person,league)</td>
<td>872</td>
</tr>
<tr>
<td>coachesinleague(person,league)</td>
<td>93</td>
</tr>
<tr>
<td>teamhomestadium(team,stadium)</td>
<td>198</td>
</tr>
<tr>
<td>athleteplayssportsteamposition(person,position)</td>
<td>255</td>
</tr>
<tr>
<td>athlete(person)</td>
<td>1909</td>
</tr>
<tr>
<td>coach(person)</td>
<td>624</td>
</tr>
<tr>
<td>male(person)</td>
<td>7</td>
</tr>
<tr>
<td>organization(league)</td>
<td>1</td>
</tr>
<tr>
<td>personafrica(person)</td>
<td>1</td>
</tr>
<tr>
<td>personaustralia(person)</td>
<td>22</td>
</tr>
<tr>
<td>personeurope(person)</td>
<td>1</td>
</tr>
<tr>
<td>personus(person)</td>
<td>6</td>
</tr>
<tr>
<td>sportsleague(league)</td>
<td>18</td>
</tr>
<tr>
<td>sportsteamposition(position)</td>
<td>22</td>
</tr>
<tr>
<td>athleteplayssportsteamposition(person,position)</td>
<td>1909</td>
</tr>
<tr>
<td>athletecoach(person,person)</td>
<td>18</td>
</tr>
<tr>
<td>athleteplayssport(person,sport)</td>
<td>1921</td>
</tr>
<tr>
<td>athleteplaysinleague(person,league)</td>
<td>872</td>
</tr>
<tr>
<td>coachesinleague(person,league)</td>
<td>93</td>
</tr>
<tr>
<td>teamhomestadium(team,stadium)</td>
<td>198</td>
</tr>
<tr>
<td>athleteplayssportsteamposition(person,position)</td>
<td>255</td>
</tr>
<tr>
<td>athlete(person)</td>
<td>1909</td>
</tr>
<tr>
<td>coach(person)</td>
<td>624</td>
</tr>
<tr>
<td>male(person)</td>
<td>7</td>
</tr>
<tr>
<td>organization(league)</td>
<td>1</td>
</tr>
<tr>
<td>personafrica(person)</td>
<td>1</td>
</tr>
<tr>
<td>personaustralia(person)</td>
<td>22</td>
</tr>
<tr>
<td>personeurope(person)</td>
<td>1</td>
</tr>
<tr>
<td>personus(person)</td>
<td>6</td>
</tr>
<tr>
<td>sportsleague(league)</td>
<td>18</td>
</tr>
<tr>
<td>sportsteamposition(position)</td>
<td>22</td>
</tr>
</tbody>
</table>

Table 5 also shows the types that were used for the variables in the base declarations for the predicates. As indicated in Section 4.5, this typing of the variables forms a syntactic restriction on the possible groundings and ensures that arguments are only instantiated with variables of the appropriate type. Furthermore, the LearnRule function of the ProbFOIL algorithm is based on mFOIL and allows for a number of variable constraints. To reduce the search space, we imposed that unary predicates that are added to a candidate rule during the learning process can only use variables that have already been introduced. Binary predicates can introduce at most one new variable.

Relational probabilistic rule learning

In order to illustrate relational probabilistic rule learning with ProbFOIL in the context of NELL, we will learn rules and report their respective accuracy for each binary predicate with more than 500 facts. In order to show ProbFOIL’s speed, also the runtimes are reported. Unless indicated otherwise, both the m-estimate’s m value and the beam width were set to 1. The value of p for rule significance was set to 0.9. The rules are postprocessed such that only range-restricted rules are obtained. Furthermore, to avoid a bias towards the majority class, the examples are balanced, i.e., negative examples are added to balance the number of positives.

Anton: negative examples are removed?

8 Kindly provided by Tom Mitchell and Jayant Krishnamurthy (CMU).

9 The dataset in ProbFOIL format can be downloaded from [removed for double-blind review].
athleteplaysforteam\( (A,B) \) :- coachesteam\( (A,B) \).
0.875:: athleteplaysforteam\( (A,B) \) :- teamhomestadium\( (B,C) \), athletehomestadium\( (A,C) \).
0.99080::athleteplaysforteam\( (A,B) \) :- teamhomestadium\( (B,\_\_\_\_) \), male\( (A) \), athleteplayssport\( (A,\_\_\_\_) \).
0.75::athleteplaysforteam\( (A,B) \) :- teamhomestadium\( (B,\_\_\_\_) \), athleteplaysinleague\( (A,C) \), teamplaysinleague\( (B,C) \), athlete\( (A) \).
0.75::athleteplaysforteam\( (A,B) \) :- teamplayssport\( (B,C) \), athleteplayssport\( (A,C) \), coach\( (A) \), teamplaysinleague\( (B,\_\_\_\_) \).
0.97555::athleteplaysforteam\( (A,B) \) :- personus\( (A) \), teamplayssport\( (B,\_\_\_\_) \).
0.762::athleteplaysforteam\( (A,B) \) :- teamplayssport\( (B,C) \), athleteplayssport\( (A,C) \), personmexico\( (A) \), teamplaysinleague\( (B,\_\_\_\_) \).
0.52571::athleteplaysforteam\( (A,B) \) :- teamplayssport\( (B,C) \), athleteplayssport\( (A,C) \), athleteplaysinleague\( (A,\_\_\_\_) \), teamplaysinleague\( (B,\_\_\_\_) \), athlete\( (A) \), teamplayssport\( (B,C) \).
0.50546::athleteplaysforteam\( (A,B) \) :- teamplayssport\( (B,\_\_\_\_) \), teamplaysinleague\( (B,C) \), athleteplaysinleague\( (A,C) \), athleteplayssport\( (A,\_\_\_\_) \), teamplayssport\( (B,C) \).
0.50::athleteplaysforteam\( (A,B) \) :- teamplayssport\( (B,\_\_\_\_) \), teamplaysinleague\( (B,C) \), athleteplaysinleague\( (A,C) \).
0.52941::athleteplaysforteam\( (A,B) \) :- teamplayssport\( (B,\_\_\_\_) \), teamhomestadium\( (B,\_\_\_\_) \), coach\( (A) \), teamplaysinleague\( (B,\_\_\_\_) \).
0.55287::athleteplaysforteam\( (A,B) \) :- teamplayssport\( (B,\_\_\_\_) \), teamplaysinleague\( (B,C) \), athleteplaysinleague\( (A,C) \), athlete\( (A) \).
0.46875::athleteplaysforteam\( (A,B) \) :- teamplayssport\( (B,\_\_\_\_) \), teamplaysinleague\( (B,\_\_\_\_) \), coach\( (A) \), teamhomestadium\( (B,\_\_\_\_) \).
Experiments

Table 4: Precision for different experimental setups and parameters ($A: m = 1, p = 0.99, B: m = 1000, p = 0.90$).

<table>
<thead>
<tr>
<th>Setting train/test/rule</th>
<th>athleteplaysforteam</th>
<th>athleteplayssport</th>
<th>teamplaysinleague</th>
<th>athleteplayssport</th>
<th>teamplaysagainstteam</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>1: det/det/det</td>
<td>74.00</td>
<td>69.36</td>
<td>94.14</td>
<td>93.47</td>
<td>96.29</td>
</tr>
<tr>
<td>2: det/prob/det</td>
<td>73.51</td>
<td>69.57</td>
<td>97.53</td>
<td>94.85</td>
<td>96.70</td>
</tr>
<tr>
<td>3: det/prob/prob</td>
<td>74.67</td>
<td>69.82</td>
<td>95.88</td>
<td>94.74</td>
<td>96.35</td>
</tr>
<tr>
<td>4: det/prob/prob</td>
<td>77.25</td>
<td>73.87</td>
<td>96.53</td>
<td>96.04</td>
<td>98.00</td>
</tr>
<tr>
<td>5: det/prob/prob</td>
<td>74.76</td>
<td>69.97</td>
<td>95.85</td>
<td>94.69</td>
<td>96.44</td>
</tr>
<tr>
<td>6: prob/prob/det</td>
<td>75.83</td>
<td>73.11</td>
<td>93.40</td>
<td>93.76</td>
<td>94.44</td>
</tr>
<tr>
<td>7: prob/prob/prob</td>
<td>78.31</td>
<td>73.72</td>
<td>95.62</td>
<td>95.10</td>
<td>98.84</td>
</tr>
</tbody>
</table>

Table 3: Learned relational rules for the different predicates (fold 1).

0.9375::athleteplaysforteam(A,B) ← athleteledsportsteam(A,B).
0.9675::athleteplaysforteam(A,B) ← athleteledsportsteam(A,V1), teamplaysagainstteam(B,V1).
0.9375::athleteplaysforteam(A,B) ← athleteplayssport(A,V1), teamplayssport(B,V1).
0.5109::athleteplaysforteam(A,B) ← athleteplayssportinleague(A,V1), teamplayssportinleague(B,V1).
0.9070::athleteplayssport(A,B) ← athleteledsportsteam(A,V2), teamalsoknownas(V2,V1), teamplayssport(V1,B), teamplayssport(V2,B).
0.9070::athleteplayssport(A,B) ← athleteplaysforteam(A,V2), teamalsoknownas(V2,V1), teamplayssport(V1,B), teamplayssport(V2,B), teamalsoknownas(V1,V2).
0.9070::athleteplayssport(A,B) ← athleteplaysforteam(A,V1), teamplayssport(V1,B).
0.9286::athleteplaysinleague(A,B) ← athleteledsportsteam(A,V1), teamplayssportinleague(V1,B).
0.7868::athleteplaysinleague(A,B) ← athleteplaysforteam(A,V2), teamalsoknownas(V2,V1), teamplaysinleague(V1,B).
0.9384::athleteplaysinleague(A,B) ← athleteplayssport(A,V2), athleteplayssport(V1,V2), teamplaysinleague(V1,B).
0.9024::athleteplaysinleague(A,B) ← athleteplaysforteam(A,V1), teamplaysinleague(V1,B).
Contributions

Learning rules (or inducing logic programs) from uncertain/probabilistic data

A new problem formulation

Traditional rule learning (ILP) is the deterministic special case

Traditional rule learning principles apply directly (including ROC analysis)
Thanks!

http://dtai.cs.kuleuven.be/problog

Introduction.
Probabilistic logic programs are logic programs in which some of the facts are annotated with probabilities. ProbLog is a tool that allows you to intuitively build programs that do not only encode complex interactions between a large sets of heterogeneous components but also the inherent uncertainties that are present in real-life situations.

The engine tackles several tasks such as computing the marginals given evidence and learning from (partial) interpretations. ProbLog is a suite of efficient algorithms for various inference tasks. It is based on a conversion of the program and the queries and evidence to a weighted Boolean formula. This allows us to reduce the inference tasks to well-studied tasks such as weighted model counting, which can be solved using state-of-the-art methods known from the graphical model and knowledge compilation literature.

The Language. Probabilistic Logic Programming.
ProbLog makes it easy to express complex, probabilistic models.

```
0.3::stress(X) :- person(X).
0.2::influences(X,Y) :- person(X), person(Y).
```
• PRISM http://sato-www.cs.titech.ac.jp/prism/
• ProbLog2 http://dtai.cs.kuleuven.be/problog/
• Yap Prolog http://www.dcc.fc.up.pt/~vsc/Yap/ includes
  • ProbLog1
  • cplint https://sites.google.com/a/unife.it/ml/cplint
• CLP(BN)
• LP2
• PITA in XSB Prolog http://xsb.sourceforge.net/
• AILog2 http://artint.info/code/ailog/ailog2.html
• SLPs http://stoics.org.uk/~nicos/sware/pepl
• contdist http://www.cs.sunysb.edu/~cram/contdist/
• DC https://code.google.com/p/distributional-clauses
• WFOMC http://dtai.cs.kuleuven.be/ml/systems/wfomc
References


