ProbLog: A Probabilistic Prolog and Its Applications to Link

Luc De Raedt
Overview

A simple Probabilistic Logic, a kind of probabilistic Database

Motivated by a real-life application in biological networks

Various techniques for supporting the analysis of such networks

• querying, probabilistic EBL and analogy, theory compression on the basis of examples
Overview

Motivation

**ProbLog:** a simple Probabilistic Logic (De Raedt, Kimmig, Toivonen, IJCAI 07)

**Probabilistic Explanation Based Learning** (Kimmig, De Raedt & Toivonen, ECML 2007)

**Compressing** ProbLog programs (De Raedt, Kersting, Kimmig, Revoredo, Toivonen, ILP 06, MLJ submitted)
Motivation
Daughter in Florence
A model

Model as a graph

- places are nodes
- streets are links
  - are unreliable (shops, monuments, musea, gelateria, ...)
  - labels on streets?
  - degree of belief that the street brings her from a to b: probability labels on links
  - links independent of one another
Questions to ask

Query: what is the probability that she can arrive at Ponte Vecchio?
- that there is a path \([NP \text{ hard - two-terminal network reliability}]\)

Most likely path
- path with highest probability

Explanation: what is the best generalized path to take?
- generalizing specific street names into properties (labels)

Analogy - EBL: can I use this explanation for other goals?
- for finding the Duomo / for walking in Venice?

Compression: give her a compressed map to memorize?
- given examples of desirable (start,end) pairs and undesirable ones, find a compressed map.
### Database Information

<table>
<thead>
<tr>
<th>Nodetype</th>
<th># nodes</th>
<th>avg. degree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Article</td>
<td>330970</td>
<td>6.9</td>
</tr>
<tr>
<td>BiologicalProcess</td>
<td>10744</td>
<td>6.8</td>
</tr>
<tr>
<td>CellularComponent</td>
<td>1807</td>
<td>16.2</td>
</tr>
<tr>
<td>Domain</td>
<td>15727</td>
<td>99.8</td>
</tr>
<tr>
<td>Gene</td>
<td>395611</td>
<td>6.1</td>
</tr>
<tr>
<td>GeneCluster</td>
<td>362155</td>
<td>2.4</td>
</tr>
<tr>
<td>HomologGroup</td>
<td>35478</td>
<td>28.5</td>
</tr>
<tr>
<td>MeSHHeading</td>
<td>22997</td>
<td>2.6</td>
</tr>
<tr>
<td>MolecularFunction</td>
<td>7922</td>
<td>7.3</td>
</tr>
<tr>
<td>OMIM</td>
<td>15253</td>
<td>34.3</td>
</tr>
<tr>
<td>Protein</td>
<td>741856</td>
<td>5.4</td>
</tr>
<tr>
<td>StructuralProperty</td>
<td>26425</td>
<td>3.3</td>
</tr>
<tr>
<td>Tissue</td>
<td>348</td>
<td>14.9</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td>1,968,951</td>
<td>7,008,607 edges</td>
</tr>
</tbody>
</table>

- Probabilities measure:
  - reliability (data source, method used)
  - relevance (of edge type w.r.t. query)
  - rarity (specificity, vertex degree)

**BioMine Project @ Helsinki** [Sevon, Toivonen et al.]
Subgraph Extracted
Questions to ask

Query: what is the probability that gene A is connected to disease B?

Explanation: what is the most likely explanation for this connection?

Explanation Based Learning: can I use this explanation to find other genes?

Compression: identify key interactions?

• given examples of interesting (gene,disease) interactions and uninteresting ones, find a compressed network.
ProbLog: Semantics and Inference
ProbLog Programs

ProbLog = Prolog + Probability Labels

- Each fact/clause has a probability label (value between 0 and 1). It indicates the probability that the fact/clause holds/belongs to the target program.

- Each fact/clause independent of other clauses

- Best viewed as a probabilistic database

- No other restrictions / assumptions

- Semantics not new (cf. Dantsin 89, Distribution Semantics by Sato 95, pD by Fuhr 00, etc.)
In ProbLog

1.0: definition of PATH in PROLOG

\[ p_A : \text{arc}(a,b) \text{ with name } A \]

Of course in real application
protein interacts with protein
protein belongs to pathway, etc.
A ProbLog program \( \{ p_1 : c_1, \ldots, p_n : c_n \} \) defines a probability distribution over programs \( L \subseteq \{ c_1, \ldots, c_n \} \)

\[
P(L|Program) = \prod_{c_i \in L} p_i \prod_{c_j \notin L} (1 - p_j)
\]

The probability of a goal \( q \)

\[
P(q|Program) = \sum_{L \subseteq \{ c_1, \ldots, c_n \}} P(q|L) \times P(L|Program)
\]

\[
P(q|L) = 1 \text{ if } L \models q
\]
\[
= 0 \text{ otherwise}
\]
\[ P(\text{path}(a,c)) = P(AB + EDC + EFB + AFDC) \]
0.9 : edge(a,c). 0.7 : edge(c,b).
0.6 : edge(d,c). 0.9 : edge(d,b).
1.0 : path(X,Y) :- edge(X,Y).
1.0 : path(X,Y) :- edge(X,Z), path(Z,Y).

?- p(d,b).
:- e(d,b).   :- e(d,A),p(A,b).

\[ d = (p_1 \land e_4) \lor (p_2 \land e_3 \land p_1 \land e_2) \]
ProbLog Evaluation

Disjoint SUM (NP-hard)

\[ P(\text{path}(a,c)) = P(AB + EDC + EFB + AFDC) \]
Disjoint Sum

\[ P(AB + AC) = P(AB) + P(A\neg BC) \]
Exact Inference

Inclusion - Exclusion:

- Experimentation with Hyspirit [Fuhr 2000] has shown that this is infeasible for the evaluation of about 10 or more conjuncts.

Some results in "Network Reliability"

- up to 100 conjuncts

Using BDDs

- up to 100000 conjuncts

Approximations are needed:

- for larger problems
Binary Decision Diagrams

- Efficient graphical representation for Boolean functions
- Widely used in computer architecture and verification
- Can be seen as variant of decision trees
  - Queries: values of single Boolean variables
    - Solid edge: variable true
    - Dashed edge: variable false
  - Fixed variable ordering on all paths
  - Shared use of identical subtrees
  - Nodes having two identical children left out
- Probability of reaching 1-terminal from the root: traverse diagram and combine probabilities of child nodes.
Algorithm 1 Probability calculation for BDDs

```
Probability(input: BDD node n)
1  if n is the 1-terminal then return 1
2  if n is the 0-terminal then return 0
3  let $p_n$ be the probability of the clause represented by $n$’s random variable
4  let $h$ and $l$ be the high and low children of $n$
5  $prob(h) := \text{Probability}(h)$
6  $prob(l) := \text{Probability}(l)$
7  return $p_n \cdot prob(h) + (1 - p_n) \cdot prob(l)$
```
Approximative Inference

Two approaches

- **Monte-Carlo**: sample programs and solve query
  \[ P(q|\text{Program}) = \sum_{L \subseteq \{c_1, \ldots, c_n\}} P(q|L) \times P(L|\text{Program}) \]

- **Bounded Inference** ([Poole 92]) using BDDs and iterative deepening
Monte Carlo

Basic idea (unpublished / ongoing)

• sample a subprogram
• search for first proof
• repeat and use fraction

Use confidence interval of 95%

Optimization

• incrementally store successful proofs in prefix tree to avoid re-computation
Bounded Inference

Alternative approach (inspired on [Poole 92])

- Incremental, levelwise search for paths using maximal length
- Level with maximal length $m$ creates two sets of paths:
  - $low$: all paths of at most $m$ edges
  - $up$: all paths of $m$ edges that did not reach the goal node yet (but might reach it with additional edges)
- $low \subseteq \text{paths} \subseteq up$
- Same principle for sets of proofs / DNF formulae in ProbLog
- Bounds on probability obtained using BDDs for $low$ and $up$:
  \[ P(low) \leq P(q|T) \leq P(up) \]
- Stop if $|P(up) - P(low)| \leq \delta$ for some small $\delta$
\[ P(\text{path}(a,c)) = P(AB+EDC+EFB+AFDC) \]

\[ P(AB+EDC+EFB) \leq P(\text{path}(a,c)) \]

\[ \leq P(AB+EDC+EFB+AFD) \]
Bounds

\[ \text{low} = (p_1 \land e_4) \]
\[ \text{up} = (p_1 \land e_4) \lor (p_2 \land e_3) \lor (p_2 \land e_4) \]
\[ d = (p_1 \land e_4) \lor (p_2 \land e_3 \land p_1 \land e_2) \]
Experimental data

- Real biological graph $G$ around four random Alzheimer genes (5220 nodes, 11530 edges)
- Example query: connection between two of the genes
- 10 sequences of subgraphs $G_1 \subset G_2 \subset \ldots$ of sizes 200, 400, \ldots edges obtained by randomly subsampling edges from $G$
- Each $G_i$ consists of exactly one connected component and contains both genes used in the query.
- Approximate inference using interval width $\delta = 0.01$
- First level of search contains proofs with up to 4 clauses, this bound is incremented by one clause on each level.
Some results

Results for 10 graphs with 1400 edges each
6 to 13 levels taking 15 seconds to 4 minutes
Scalability

- Good scalability as often only small fraction of proofs needed for approximation
- Example query solved for graphs with up to 1400 to 4600 edges, depending on the random sample
- Runtimes from some seconds up to four hours for larger graphs
- Runtimes influenced by many factors, difficult to predict based on size of graph

![Graph showing scalability](image)
Current trends

Memory versus time trade-off

Typically: when BDD not too large, BDD faster, otherwise MC.

Confidence intervals reasonable
Inference

Approximate inference

• allows one to compute god approximations of the success probability of complex queries

• scales pretty well

Current/future work in inference

• incorporate background knowledge / ontology

• pose really complex queries -- full power of Prolog
Probabilistic EBL
Explanation-Based Learning

Popular in ML in the 1980s

• largely speed-up learning
• knowledge based
• role of abduction

Probabilistic EBL:

• using ProbLog (but can be adapted towards other logics)
• finding explanations of examples
• use explanations to find similar examples
Explanation based learning:

• compute a proof-tree of an example

• **generalize** the proof-tree by not expanding *operational* predicates

• **explanation** = *collection of leafs of the generalized proof-tree*
The CUP example

Figure 1: Generalized Explanation for the Cup Example.

cup(X) :- partof(B,X), bottom(B), flat(B),
partof(H,X), handle(H), light(X),
partof(C,X), concavity(C), partof(C,X),
up_pointing(C).

from [Mooney and Zelle, 94]
Table 2: A Sample EBG Problem.

Goal Concept:
cup(x) :- stable(x), liftable(x), open_vessel(x).

Training Example:
owner(obj1,fred). light(obj1). color(obj1,red).
partof(h1,obj1). handle(h1). bottom(b1).
partof(b1,obj1). flat(b1). concavity(c1).
partof(c1,obj1). up_pointing(c1).

Domain Theory:
stable(X) :- partof(Y,X), bottom(Y), flat(Y).
liftable(X) :- graspable(X), light(X).
graspable(X) :- partof(Y,X), handle(Y).
open_vessel(X) :- partof(Y,X), concavity(Y),
                 up_pointing(Y).

Operationality Criterion: Concept definition must be expressed in terms of structural features used to describe examples (e.g. light, handle, flat, etc.).
Most Likely Proof

Disjoint SUM (NP-hard)

\[ P(\text{path}(a,c)) = P(\text{AB} + \text{EDC} + \text{EFB} + \text{AFDC}) \]

Most Likely Proof

\[ \text{MLP}(\text{path}(a,c)) = \arg \max_{x \in \{\text{AB, EDC, EFB, AFDC}\}} P(X) \]
Most likely proof

Finding the most likely proof

• use (iterative deepening) best-first SLD-resolution

• obtain a conjunctive expression

• compute its probability
  - only product, no disjoint sum problem
  - easier & much faster to compute
Probabilistic Explanation Based learning:

- compute the most likely generalized proof of example

\[
\prod_{c \in \text{Proof}} p_c = \prod_{i \in \text{GenProof}} p_i \times \prod_{j \in \text{OperPred}} p_j
\]

- stop expanding the proof tree at operational predicates
- collect leafs of the tree
Example EBL

1.0 : related(X,Y) :- similar(X,Y).
0.8 : related(X,Y) :- similar(X,Z), related(Z,Y).
0.8 : similar(a,c).
0.7 : similar(c,b).
0.6 : similar(d,c).
0.9 : similar(d,b).
0.7 : similar(e,c).
0.5 : similar(f,a).

?- r(a,b).
?- r(d,b).
?- r(X,Y).

?- s(a,b).
?- s(d,b).
?- s(X,Y).

?- s(a,A),r(A,b).
?- s(d,B),r(B,b).
?- s(X,Z),r(Z,Y).

?- r(a,b).
fail

?- r(X,Y).

?- s(a,A),r(A,b).

?- s(d,b).

?- s(X,Z),r(Z,Y).

?- r(c,b).

?- s(X,Z),r(Z,Y).

?- r(c,b).

?- s(X,Z),r(Z,Y).

?- r(c,b).

?- s(c,b).

?- s(c,b).

?- s(c,b).

?- s(c,b).

?- s(X,Z),s(Z,Y).

Fig. 2. (a) The successful branch of the SLD tree for related(a,b). (b) The corresponding branch for general goal related(X,Y), where bold atoms are part of the explanation and the bar marks the position to continue the proof. (c) A partial SLD
Obtained explanations

\[
e_{\text{path}}(A,B) \leftarrow \text{node}(A, \text{gene}), \text{edge}(A, C, \text{belongs.to}), \text{node}(C, \text{homologgroup}), \text{edge}(B, C, \text{refers.to}), \text{node}(E, \text{phenotype}), \text{nodes.distinct}([B, C, A]).
\]
\[
e_{\text{path}}(A,B) \leftarrow \text{node}(A, \text{gene}), \text{edge}(A, C, \text{codes.for}), \text{node}(C, \text{protein}), \text{edge}(D, C, \text{subsumes}), \text{node}(D, \text{protein}), \text{edge}(D, E, \text{interacts.with}), \text{node}(E, \text{protein}), \text{edge}(B, E, \text{refers.to}), \text{node}(B, \text{phenotype}), \text{nodes.distinct}([B, E, D, C, A])
\]
\[
e_{\text{path}}(A,B) \leftarrow \text{node}(A, \text{gene}), \text{edge}(A, C, \text{participates.in}), \text{node}(C, \text{pathway}), \text{edge}(D, C, \text{participates.in}), \text{node}(D, \text{gene}), \text{edge}(D, E, \text{codes.for}), \text{node}(E, \text{protein}), \text{edge}(B, E, \text{refers.to}), \text{node}(B, \text{phenotype}), \text{nodes.distinct}([B, E, D, C, A])
\]
\[
e_{\text{path}}(A,B) \leftarrow \text{node}(A, \text{gene}), \text{edge}(A, C, \text{is.found.in}), \text{node}(C, \text{cellularcomponent}), \text{edge}(D, C, \text{is.found.in}), \text{node}(D, \text{protein}), \text{edge}(B, D, \text{refers.to}), \text{node}(B, \text{phenotype}), \text{nodes.distinct}([B, D, C, A])
\]

\[\text{Fig. 4.}\] One explanation for connect\((G1, G2, P)\), where double circles mark answer variables, and node types used are gene \((g)\), phenotype \((pt)\) and homologgroup \((hg)\).
PEBL Issues

Use explanation to solve similar tasks

- find explanation for gene being connected to disease
- use it to locate other genes being connected to disease

Easy to generalize to multiple examples

- find max likely generalized common proof tree

Transfer across domains

- from Alzheimer to Asthma
Similarity-based EBL

Given positive example
Find explanation
Use explanation to retrieve further examples
Use ProbLog to compute probabilities of examples and rank them
Experiments

Biological networks for Alzheimer and Asthma genes of different sizes generated

Questions:

• Does PEBL produce meaningful examples?
• Can we find common explanations?
• Can we apply transfer learning?

connect(G1,G2,P):- path(G1,P), path(G2,P), path(G1,G2).
## Experiments

<table>
<thead>
<tr>
<th></th>
<th>depth</th>
<th>nodes</th>
<th>edges</th>
<th>ag</th>
<th>ng</th>
<th>pt</th>
<th>pos</th>
<th>neg</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alz1</td>
<td>4</td>
<td>122</td>
<td>259</td>
<td>14</td>
<td>15</td>
<td>3</td>
<td>182</td>
<td>2254</td>
</tr>
<tr>
<td>Alz2</td>
<td>5</td>
<td>658</td>
<td>3544</td>
<td>17</td>
<td>20</td>
<td>4</td>
<td>272</td>
<td>5056</td>
</tr>
<tr>
<td>Alz3</td>
<td>4</td>
<td>351</td>
<td>774</td>
<td>72</td>
<td>33</td>
<td>3</td>
<td>5112</td>
<td>27648</td>
</tr>
<tr>
<td>Alz4</td>
<td>5</td>
<td>3364</td>
<td>17666</td>
<td>130</td>
<td>55</td>
<td>6</td>
<td>16770</td>
<td>187470</td>
</tr>
<tr>
<td>Ast1</td>
<td>4</td>
<td>127</td>
<td>241</td>
<td>7</td>
<td>12</td>
<td>2</td>
<td>42</td>
<td>642</td>
</tr>
<tr>
<td>Ast2</td>
<td>5</td>
<td>381</td>
<td>787</td>
<td>11</td>
<td>12</td>
<td>2</td>
<td>110</td>
<td>902</td>
</tr>
</tbody>
</table>

Table 1. Graph characteristics: search depth used during graph extraction, numbers of nodes and edges, number of genes annotated resp. not annotated with the corresponding disease and number of phenotypes, number of positive and negative examples for connecting two genes and a phenotype.
## Results

<table>
<thead>
<tr>
<th></th>
<th>pos(1)</th>
<th>pos(3)</th>
<th>pos(5)</th>
<th>pos_n</th>
<th>pos_a</th>
<th>prec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alz1</td>
<td>0.95</td>
<td>2.53</td>
<td>3.95</td>
<td>6.91</td>
<td>16.82</td>
<td>0.46</td>
</tr>
<tr>
<td>Alz2</td>
<td>0.84</td>
<td>2.24</td>
<td>3.60</td>
<td>7.37</td>
<td>18.65</td>
<td>0.42</td>
</tr>
<tr>
<td>Alz3</td>
<td>0.99</td>
<td>2.64</td>
<td>4.09</td>
<td>23.20</td>
<td>126.09</td>
<td>0.48</td>
</tr>
<tr>
<td>Alz4</td>
<td>0.84</td>
<td>2.23</td>
<td>3.58</td>
<td>7.37</td>
<td>18.80</td>
<td>0.42</td>
</tr>
<tr>
<td>Ast1</td>
<td>0.09</td>
<td>0.26</td>
<td>0.44</td>
<td>2.07</td>
<td>2.07</td>
<td>0.02</td>
</tr>
<tr>
<td>Ast2</td>
<td>0.08</td>
<td>0.23</td>
<td>0.38</td>
<td>2.00</td>
<td>2.00</td>
<td>0.01</td>
</tr>
<tr>
<td></td>
<td>1.00</td>
<td>3.00</td>
<td>4.86</td>
<td>6.86</td>
<td>10.57</td>
<td>0.23</td>
</tr>
<tr>
<td></td>
<td>0.86</td>
<td>2.86</td>
<td>4.71</td>
<td>6.86</td>
<td>14.56</td>
<td>0.22</td>
</tr>
<tr>
<td></td>
<td>1.00</td>
<td>2.71</td>
<td>4.14</td>
<td>6.86</td>
<td>28.00</td>
<td>0.24</td>
</tr>
<tr>
<td></td>
<td>0.86</td>
<td>2.29</td>
<td>3.43</td>
<td>5.14</td>
<td>28.00</td>
<td>0.15</td>
</tr>
<tr>
<td></td>
<td>1.00</td>
<td>3.00</td>
<td>4.86</td>
<td>17.14</td>
<td>17.14</td>
<td>0.34</td>
</tr>
<tr>
<td></td>
<td>0.86</td>
<td>2.57</td>
<td>4.29</td>
<td>16.57</td>
<td>16.57</td>
<td>0.20</td>
</tr>
</tbody>
</table>

Table 2. Averaged results over all examples learned on Alz1 (left column) resp. Ast1 (right column) and evaluated on 6 different graphs (lines Alz1–4, Ast1–2): number of positives among the first $k$ answers (pos($k$)), number of positives returned before the first negative (pos_n), absolute number of positives among examples with non-zero probability (pos_a), and precision w.r.t. all examples with non-zero probability (prec).
In total 26 explanations for Alz1, 3 for Ast 1
One explanation shared by 12 examples, most explanations shared by multiple examples
PEBL: conclusions

EBL in a new *probabilistic* context

- no speed-up learning, but
- analogy, reasoning by cases, similarity
- good predictive power
- explanations by themselves of interest

*Current* ongoing work:

- local pattern mining
Compression
Map for Daughter

Suppose

- my daughter always starts from the hotel
- certain places are of interest (musea, monuments)
- certain places are to be avoided (shops, gelateria, pubs, ...)

The problem:

- give her a compressed map containing the best routes to the interesting places and avoids the uninteresting ones
Example setting:
- Biological network
- Researcher interested in one specific disease (blue node)
- Some genes are known to be related to that disease (green nodes)
- Some genes are known NOT to be related to that disease (red nodes)
- Goal: automatically extract subgraph of given maximal size which best captures this knowledge
Example

- Positive examples:
  \text{path(green1,blue)}
  \text{path(green2,blue)}

- Negative examples:
  \text{path(red1,blue)}
  \text{path(red2,blue)}

- Goal: subgraph with at most $k$ edges giving high probability to positive and low probability to negative examples
Revision

**Given**

- a ProbLog program $T$
- Positive and Negative Exs. $P$ and $N$
- constant $k$

**Find:** the program $L \subseteq T$ of size at most $k$ that maximizes

$$\text{Likelihood}(L|E) = \prod_{p \in Pos} P(p|L) \prod_{n \in Neg} (1 - P(n|L))$$
Collect BDDs for (lower bound) approximations of examples
Consider the clauses / facts used as revision points
Perform a greedy search

BDDs for efficiency:

- recomputing prob. ex. with deleted clause is easy
- set corresponding boolean to 0
Fig. 3 Effect of deleting clause $s3$ in Example 2: (a) Initial BDD: $P(e|T) = 0.9 + 0.1 \cdot 0.8 \cdot 0.7 \cdot 0.6 = 0.9336$. (b) BDD after deleting $s3$ by setting its probability to 0: $P(e|T) = 0.9 + 0.1 \cdot 0.8 \cdot 0 \cdot 0.6 = 0.9$. 
Algorithm

**COMPRESS**(input: $S = \{p_1 : c_1, \ldots, p_n : c_n\}$, $E$, $k$, $\epsilon$)

1. **for** all $e \in E$
2.     **do call** **APPROXIMATE**(e, $S$, $\delta$) to get $DNF(low, e)$ and $BDD(e)$
3.     where $DNF(low, e)$ is the lower bound DNF formula for $e$
4.     and $BDD(e)$ is the BDD corresponding to $DNF(low, e)$
5. $R := \{p_i : c_i \mid b_i$ (indicator for clause $i$) occurs in a $DNF(low, e)\}$
6. $BDD(E) := \bigcup_{e \in E}\{BDD(e)\}$
7. improves := true
8. **while** ($|R| > k$ or improves) and $R \neq \emptyset$
9.     **do** $ll := LIKELIHOOD(R, BDD(E), \epsilon)$
10.     $i := \arg \max_{i \in R} LIKELIHOOD(R - \{i\}, BDD(E), \epsilon)$
11.     improves := ($ll \leq LIKELIHOOD(R - \{i\}, BDD(E), \epsilon)$)
12.     **if** improves or $|R| > k$
13.         **then** $R := R - \{i\}$
14. **return** $R$
Reduce to at most $k$ edges (greedy approach, reusing BDDs for scoring)

Example: Green and blue should be connected, red and blue not (all edges have probability 0.5)
Fig. 4 Evolvement of log-likelihood for 10 test runs with positive examples only (a) and with both positive and negative examples (b). Different starting points of lines reflect the number of clauses in the BDDs used in theory compression.
Fig. 5 Evolvement of log-likelihood for 10 test runs with both positive and negative examples (a). Evolvement of log-likelihood for settings with artificially implanted edges with negative and positive effects (b). In (b), curve color indicates the type of edge that was removed in the corresponding revision step. Likelihoods in the middle reflect the probability on artificial edges: topmost curve is for $p = 0.9$, going down in steps of size 0.1.
Compression

Does produce small and useful graphs

A form of theory revision for probabilistic logics
  • add new links / clauses ?

Ongoing direction
  • learning ProbLog programs from data
Conclusions

A simple but powerful Probabilistic Prolog

- logic provides an elegant framework for MGTS

Inference using approximation algorithms

New probabilistic framework for EBL

Compression applied to link discovery

One future direction -- use to make real discoveries