# Learning Hierarchical Probabilistic Logic Programs

Arnaud Nguembang Fadja, Fabrizio Riguzzi, Evelina Lamma

University of Ferrara

Keywords: **Hierarchical Probabilistic Logic Programming, Deep Neural Networks, Arithmetic Circuits.**

## Probabilistic Logic Programming (PLP) and Hierarchical PLP

- Probabilistic logic programming is a powerful tool for reasoning with **uncertain** relational models.
- Learning probabilistic logic programs is expensive due to the high cost of inference.
- We consider a restriction of the language of Logic Programs with Annotated Disjunctions called hierarchical PLP in which clauses and predicates are hierarchically organized.
- Generic clause $C$:

$$C = p(\vec{X}) : \pi :- \phi(\vec{X}, \vec{Y}), b_1(\vec{X}, \vec{Y}), \ldots, b_m(\vec{X}, \vec{Y})$$

where $\phi(\vec{X}, \vec{Y})$ is a conjunction of literals for the input predicates.

1

## Generic Hierarchical PLP: https://rdcu.be/cmCIA

$$
\begin{aligned}
C_1 = r(\vec{X}) : \pi_1 &:- \phi_1, b_{1\_1}, \ldots, b_{1\_m_1} \\
&\ldots \\
C_n = r(\vec{X}) : \pi_n &:- \phi_n, b_{n\_1}, \ldots, b_{n\_m_n} \\
C_{1\_1\_1} = r_{1\_1}(\vec{X}) : \pi_{1\_1\_1} &:- \phi_{1\_1\_1}, b_{1\_1\_1\_1}, \ldots, b_{1\_1\_1\_m_{111}} \\
&\ldots \\
C_{1\_1\_n_{11}} = r_{1\_1}(\vec{X}) : \pi_{1\_1\_n_{11}} &:- \phi_{1\_1\_n_{11}}, b_{1\_1\_n_{11}\_1}, \ldots, b_{1\_1\_n_{11}\_m_{11n_{11}}} \\
&\ldots \\
C_{n\_1\_1} = r_{n\_1}(\vec{X}) : \pi_{n\_1\_1} &:- \phi_{n\_1\_1}, b_{n\_1\_1\_1}, \ldots, b_{n\_1\_m_{n11}} \\
&\ldots \\
C_{n\_1\_n_{n1}} = r_{n\_1}(\vec{X}) : \pi_{n\_1\_n_{n1}} &:- \phi_{n\_1\_n_{n1}}, b_{n\_1\_n_{n1}\_1}, \ldots, b_{n\_1\_n_{n1}\_m_{n1n_{n1}}} \\
&\ldots
\end{aligned}
$$

2

## Example

- UW-CSE domain, the objective is to predict the "advised_by" relation

$C_1$ = *advised_by*$(A, B) : 0.3 :-$
*student*$(A)$, *professor*$(B)$, *project*$(C, A)$, *project*$(C, B)$, $r_{11}(A, B, C)$.

$C_2$ = *advised_by*$(A, B) : 0.6 :-$
*student*$(A)$, *professor*$(B)$, *ta*$(C, A)$, *taught_by*$(C, B)$.

$C_{111} = r_{11}(A, B, C) : 0.2 :-$
*publication*$(D, A, C)$, *publication*$(D, B, C)$.

- $r$ = *advised_by*(*harry*, *ben*) where *harry* is a student, *ben* is a professor, they have two joint courses, two joint projects and two joint publications from each jointed project.

3

## Parameter learning for HIerarchical probabilistic Logic programs

- Given a HPLP $T$ with parameters $\Pi$, and a training set $E = \{e_1, \ldots, e_M, \text{not } e_{M+1}, \ldots, \text{not } e_N\}$ **PHIL** finds the values of $\Pi$ that maximize the log likelihood:

$$\arg\max_{\Pi} \sum_{i=1}^{M} \log P(e_i) + \sum_{i=M+1}^{N} \log(1 - P(e_i)) \quad (1)$$

$P(e_i)$ is the probability of example $e_i$.

- Or equivalently minimizing the sum of *cross entropy errors* for all the examples $err_i = -y_i \log(p_i) - (1 - y_i) \log(1 - p_i)$
- Deep PHIL (DPHIL) and Expectation Maximization PHIL (EMPHIL) learn the parameters using Gradient Descent and Expectation Maximization.

4

## Structure LEArning of Hierarchical Probabilistic logic programming

Given a set of interpretations, each containing positive and negative examples for a target predicate and facts for input predicates,
$I = \{e_1, \ldots, e_M, \text{not } e_{M+1}, \ldots, \text{not } e_N\}$, **SLEAHP** finds an HPLP with parameters $\Pi$ that maximizes the (log) likelihood

$$LL = \arg\max_{\Pi} \sum_{k}^{Nl} (\sum_{i=1}^{M} \log P(e_{i_k}) + \sum_{i=M+1}^{N} \log(1 - P(e_{i_k}))) \quad (2)$$

5

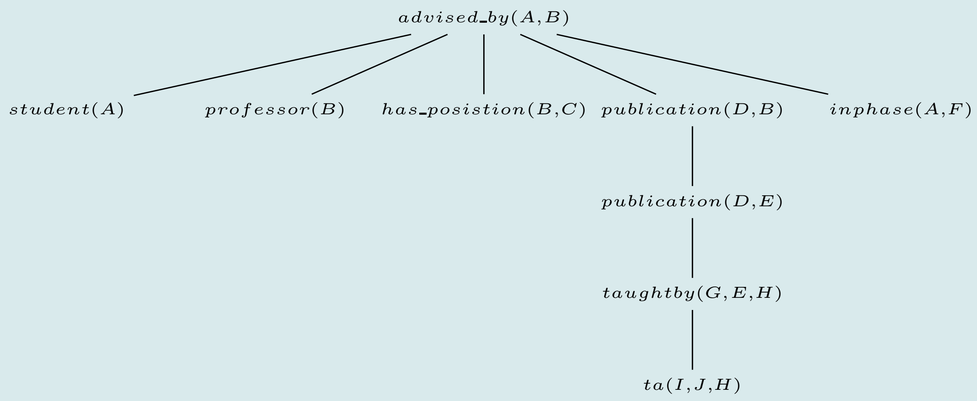
## Structure Learning: SLEAHP algorithm

1. **Input**: a set of interpretations and a target predicate (examples).
2. Generate a set of bottom clauses from a language bias.
3. Generate a program tree from the bottom clauses.
4. **Randomly** generate an initially **large** HPLP from the tree.
5. Apply a regularized version of PHIL on the HPLP.
6. Discard Clauses with very small probability values.

6

## Structure Learning: program tree from a bottom clause

*advised_by*$(A, B) :-$ *student*$(A)$, *professor*$(B)$, *has_position*$(B, C)$, *publication*$(D, B)$, *publication*$(D, E)$, *inphase*$(A, F)$, *taughtby*$(G, E, H)$, *ta*$(I, J, H)$.




7

## Structure Learning: initial HPLP generated from the tree

*advised_by*$(A, B) : 0.5 :-$ *student*$(A)$.
*advised_by*$(A, B) : 0.5 :-$ *professor*$(B)$.
*advised_by*$(A, B) : 0.5 :-$ *has_position*$(B, C)$.
*advised_by*$(A, B) : 0.5 :-$ *publication*$(D, B)$, *hidden_1*$(A, B, D)$.
*advised_by*$(A, B) : 0.5 :-$ *inphase*$(A, E)$.
*hidden_1*$(A, B, D) : 0.5 :-$ *publication*$(D, F)$, *hidden_11*$(A, B, D, F)$.
*hidden_11*$(A, B, D, F) : 0.5 :-$ *taughtby*$(G, F, H)$, *hidden_111*$(A, B, D, F, G, H)$.
*hidden_111*$(A, B, D, F, G, H) : 0.5 :-$ *ta*$(I, J, H)$.

- Learning HPLPs on web browser: **phil on SWISH**
- Link: https://cplint.eu/e/phil/phil_examples.swinb
- Manual: https://arnaudfadja.github.io/phil/
- Paper: https://rdcu.be/cmCIA

8

## Experiments: Average time for PHIL

| Time | Mutagenesis | Carcinogenesis | Mondial | UWCSE | PAKDD15 |
|---|---|---|---|---|---|
| DPHIL | *2.8573* | 178.2680 | 265.4160 | **0.0884** | 34.8170 |
| $DPHIL_1$ | 5.2059 | 177.4270 | 311.3280 | 0.2910 | 20.5704 |
| $DPHIL_2$ | 5.4450 | 88.5100 | 301.1392 | 0.2214 | 4.9517 |
| EMPHIL | 4.4430 | *106.5540* | 270.6688 | 0.2890 | 6.6106 |
| $EMPHIL_1$ | 4.8940 | *181.0890* | 317.4202 | 1.0000 | 7.0691 |
| $EMPHIL_2$ | 5.0046 | 146.8440 | **245.3830** | *0.8372* | 6.6334 |
| $EMPHIL_{B_1}$ | 2.6478 | 85.3210 | 248.1978 | 0.1572 | 6.0990 |
| $EMPHIL_{B_2}$ | **1.5820** | **80.4270** | *261.3612* | 0.1266 | 8.8722 |
| $EMPHIL_{B_3}$ | 1.4937 | 94.6850 | 274.9598 | 0.1190 | 6.0985 |
| EMBLEM | 125.62 | - | - | 0.9666 | 154.51 |
| ProbLog2 | 722.00 | 38685.00 | 1607.60 | 161.40 | 596.90 |
| Tuffy | - | - | - | - | - |

9

## Experiments: Average time for SLEAHP

| Time | Mutagenesis | Carcinogenesis | Mondial | UWCSE | PAKDD15 |
|---|---|---|---|---|---|
| $SLEAHP_{G_1}$ | 41.8250 | **48.7600** | *59.5054* | *219.6410* | 192.4396 |
| $SLEAHP_{G_2}$ | 47.1344 | 10524.0900 | **14.0470** | *194.9706* | *162.9938* |
| $SLEAHP_{E_1}$ | 48.0152 | *303.0570* | 60.8316 | 387.6650 | 151.7217 |
| $SLEAHP_{E_2}$ | *45.9245* | 92.3820 | 61.0996 | 312.2604 | 68.2432 |
| $SLEAHP_B$ | **13.1478** | 1399.0090 | 14.6698 | 295.6734 | **61.5347** |
| SLIPCOVER | 74610.7000 | 17419.4500 | 650.3630 | **141.3600** | 242.7077 |
| PROBFOIL+ | 1726.6000 | 15433.0000 | - | - | - |
| MLN-BC | 91.7700 | 59.73500 | 56.5007 | 376.2356 | * |
| MLN-BT | 360.8563 | 87.5020 | 53.5568 | 891.4226 | 915.5794 |
| RDN-B | 183.7140 | 61.5000 | 192.0487 | 501.1176 | 793.7661 |

- PHIL and SLEAHP Provide similar performance w.r.t SOTA in terms of area under the ROC and PR curves but appear to be scalable.

10

Department of Engineering Ferrara

Arnaud Nguembang Fadja
arnaud.nguembafadja@unife.it

http://www.arnaud-fadja.it/ http://ml.unife.it/