Learning the Parameters of Deep Probabilistic Logic Programs

Arnaud Nguembang Fadja    Fabrizio Riguzzi    Evelina Lamma

Dipartimento di Ingegneria – University of Ferrara

Dipartimento di Matematica e Informatica – University of Ferrara
[arnaud.nguembafadja,fabrizio.riguzzi,evelina.lamma]@unife.it
• 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.

• Inference is then much cheaper.
A PLP language under the distribution semantics with a general syntax is Logic Programs with Annotated Disjunctions (LPADs).

Heads of clauses are disjunctions in which each atom is annotated with a probability.

LPAD $T$ with $n$ clauses: $T = \{C_1, \ldots, C_n\}$.

Each clause $C_i$ takes the form:

$$h_{i1} : \pi_{i1}; \ldots; h_{iv_i} : \pi_{iv_i} \leftarrow b_{i1}, \ldots, b_{iu_i}$$

Each grounding $C_i \theta_j$ of a clause $C_i$ corresponds to a random variable $X_{ij}$ with values $\{1, \ldots, v_i\}$

The random variables $X_{ij}$ are independent of each other.
Example

- UW-CSE domain:

```
advisedby(A, B) : 0.3 :-
    student(A), professor(B), project(C, A), project(C, B).

advisedby(A, B) : 0.6 :-
    student(A), professor(B), ta(C, A), taughtby(C, B).
```
Hierarchical PLP

• We want to compute the probability of atoms for a predicate \( r: r(\mathbf{t}) \), where \( \mathbf{t} \) is a vector of constants.
• \( r(\mathbf{t}) \) can be an example in a learning problem and \( r \) a target predicate.
• A specific form of an LPADs defining \( r \) in terms of the input predicates.
• The program defined \( r \) using a number of input and hidden predicates disjoint from input and target predicates.
• Each rule in the program has a single head atom annotated with a probability.
• The program is hierarchically defined so that it can be divided into layers.
• Each layer contains a set of hidden predicates that are defined in terms of predicates of the layer immediately below or in terms of input predicates.
Hierarchical PLP

• Generic clause $C$:

$$C = p(X) : \pi : \neg \phi(X,Y), b_1(X,Y), \ldots, b_m(X,Y)$$

where $\phi(X,Y)$ is a conjunction of literals for the input predicates using variables $X, Y$.

• $b_i(X,Y)$ for $i = 1, \ldots, m$ is a literal built on a hidden predicate.

• $Y$ is a possibly empty vector of variables existentially quantified with scope the body.

• Literals for hidden predicates must use the whole set of variables $X, Y$.

• The predicate of each $b_i(X,Y)$ does not appear elsewhere in the body of $C$ or in the body of any other clause.
Hierarchical PLP

- A generic program defining $r$ is thus:

$$C_1 = r(X) : \pi_1 : - \phi_1, b_{11}, \ldots, b_{1m_1}$$

$$\ldots$$

$$C_n = r(X) : \pi_n : - \phi_n, b_{n1}, \ldots, b_{nm_n}$$

$$C_{111} = r_{11}(X) : \pi_{111} : - \phi_{111}, b_{1111}, \ldots, b_{111m_{111}}$$

$$\ldots$$

$$C_{11n_{11}} = r_{11}(X) : \pi_{11n_{11}} : - \phi_{11n_{11}}, b_{11n_{11}1}, \ldots, b_{11n_{11}m_{11n_{11}}}$$

$$\ldots$$

$$C_{n1n_1} = r_{n1}(X) : \pi_{n1n_1} : - \phi_{n1n_1}, b_{n1n_11}, \ldots, b_{n1n_1m_{n1n_1}}$$

$$\ldots$$

$$C_{n1n_1n_1} = r_{n1}(X) : \pi_{n1n_1n_1} : - \phi_{n1n_1n_1}, b_{n1n_1n_11}, \ldots, b_{n1n_1n_1m_{n1n_1n_1}}$$

$$\ldots$$
Example

\[C_1 = \text{advisedby}(A, B) : 0.3 : - \\
\quad \text{student}(A), \text{professor}(B), \text{project}(C, A), \text{project}(C, B), \]
\[r_{11}(A, B, C).\]

\[C_2 = \text{advisedby}(A, B) : 0.6 : - \\
\quad \text{student}(A), \text{professor}(B), \text{ta}(C, A), \text{taughtby}(C, B).\]

\[C_{111} = r_{11}(A, B, C) : 0.2 : - \\
\quad \text{publication}(D, A, C), \text{publication}(D, B, C).\]
Inference

• Generate the grounding.
• Each ground probabilistic clause is associated with a random variable whose probability of being true is given by the parameter of the clause and that is independent of all the other clause random variables.
• Ground clause \( C_{pi} = a_p : \pi_{pi} :- b_{pi1}, \ldots, b_{pim_p} \), where \( p \) is a path in the program tree

\[
P(b_{pi1}, \ldots, b_{pim_p}) = \prod_{i=k}^{mp} P(b_{pik}) \quad \text{and} \quad P(b_{pik}) = 1 - P(a_{pik}) \text{ if } b_{pik} = \neg a_{pik}.
\]
• If \( a \) is a literal for an input predicate, then \( P(a) = 1 \) if \( a \) belongs to the example interpretation and \( P(a) = 0 \) otherwise.
Inference

- Hidden predicates: to compute $P(a_p)$ we need to take into account the contribution of every ground clause for the predicate of $a_p$.
- Suppose these clauses are $\{C_{p1}, \ldots, C_{pn}\}$.
- For one clause, $P(a_p) = \pi_{p1} \cdot P(body(C_{p1}))$
- If we have two clauses, 
  
  $$P(a_p) = 1 - (1 - \pi_{p1} \cdot P(body(C_{p1})) \cdot (1 - \pi_{p2} \cdot P(body(C_{p2})))$$

- $p \oplus q \triangleq 1 - (1 - p) \cdot (1 - q)$.
- This operator is commutative and associative:
  
  $$\bigoplus_i p_i = 1 - \prod_i (1 - p_i)$$

- The operators $\times$ and $\oplus$ are respectively the t-norm and t-conorm of the product fuzzy logic [Hajek 98]: product t-norm and probabilistic sum.
Inference

- If the probabilistic program is ground, the probability of the example atom can be computed with the arithmetic circuit:

- The arithmetic circuit can be interpreted as a deep neural network where nodes have the activation functions $\times$ and $\oplus$. 
Example

\[ G_1 = \text{advisedby}(\text{harry}, \text{ben}) : 0.3 : - \]
\[ \text{student(harry), professor(ben), project(pr1, harry),} \]
\[ \text{project(pr1, ben), } r_{11}(\text{harry, ben, pr1}). \]

\[ G_2 = \text{advisedby}(\text{harry}, \text{ben}) : 0.3 : - \]
\[ \text{student(harry), professor(ben), project(pr2, harry),} \]
\[ \text{project(pr2, ben), } r_{11}(\text{harry, ben, pr2}). \]

\[ G_3 = \text{advisedby}(\text{harry}, \text{ben}) : 0.6 : - \]
\[ \text{student(harry), professor(ben), ta(c1, harry), taughtby(c1, ben).} \]

\[ G_4 = \text{advisedby}(\text{harry}, \text{ben}) : 0.6 : - \]
\[ \text{student(harry), professor(ben), ta(c2, harry), taughtby(c2, ben).} \]

\[ G_{111} = r_{11}(\text{harry, ben, pr1}) : 0.2 : - \]
\[ \text{publication(p1, harry, pr1), publication(p1, ben, pr1).} \]

\[ G_{112} = r_{11}(\text{harry, ben, pr1}) : 0.2 : - \]
\[ \text{publication(p2, harry, pr1), publication(p2, ben, pr1).} \]

\[ G_{211} = r_{11}(\text{harry, ben, pr2}) : 0.2 : - \]
\[ \text{publication(p3, harry, pr2), publication(p3, ben, pr2).} \]

\[ G_{212} = r_{11}(\text{harry, ben, pr2}) : 0.2 : - \]
\[ \text{publication(p4, harry, pr2), publication(p4, ben, pr2).} \]
Example

\[ G_{1} \xrightarrow{r_{11}(harry, ben, pr1)} G_{111} \quad G_{112} \]
\[ G_{2} \xrightarrow{r_{11}(harry, ben, pr2)} G_{211} \quad G_{212} \]

\[ G_{11} \oplus G_{21} = G_{2} \]

\[ G_{111} \times 0.36 = 0.2 \quad G_{112} \times 0.36 = 0.2 \]

\[ G_{211} \times 0.36 = 0.2 \quad G_{212} \times 0.36 = 0.2 \]

\[ 0.873 = 1.0 \]

\[ Fadja, Riguzzi and Lamma (UNIFE) \]

Hierarchical PLP
Figure: Arithmetic circuit.
Parameter Learning

- Given a Hierarchical PLP $T$ with parameters $\Pi$, an interpretation $I$ defining input predicates and a training set $E = \{e_1, \ldots, e_M, \text{not } e_{M+1}, \ldots, \text{not } e_N\}$ find the values of $\Pi$ that maximize the log likelihood:

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

where $P(e_i)$ is the probability assigned to $e_i$ by $T \cup I$.

- Maximizing the log likelihood can be equivalently seen as minimizing the sum of cross entropy errors $err_i$ for all the examples

$$err_i = -y_i \log(p_i) - (1 - y_i) \log(1 - p_i)$$ (2)

where $y_i = 1$ for positive example, $y_i = 0$ otherwise and $p_i$ the probability that the atom is true.
Parameter Learning

- Partial derivative of the error with respect to each node $v(n)$:

$$\frac{\partial \text{err}}{\partial v(n)} = \begin{cases} 
- \frac{1}{v(r)} d(n) & \text{if } e \text{ is positive}, \\
\frac{1}{1-v(r)} d(n) & \text{if } e \text{ negative.}
\end{cases}$$

where

$$d(n) = \begin{cases} 
d(p_n) \frac{v(p_n)}{v(n)} & \text{if } n \text{ is a } \oplus \text{ node,} \\
d(p_n) \frac{1-v(p_n)}{1-v(n)} & \text{if } n \text{ is a } \times \text{ node} \\
\sum_{p_n} d(p_n) . v(p_n) . (1 - \Pi_i) & \text{if } n \text{ is a leaf node } \Pi_i \\
-d(p_n) & \text{if } p_n = \text{not}(n)
\end{cases}$$

(3)

and $v(n)$, $p_n$ are respectively the value and the parent of the node $n$. 
Parameter Learning

- Build the ACs and initialize the parameters and the gradients.
- Perform the forward pass by computing the output of each node ($v(n)$) in the AC.
- Compute the gradient of the error w.r.t the output and back-propagate.
- Update the parameters using Adam optimizer.
- Until convergence or a certain condition is satisfied.
Conclusion and Future Work

• Conclusion
  • hierarchical PLP: a restriction of the language of LPADs that allows to perform inference quickly using a simple and cheap dynamic programming algorithm such as PITA(IND,IND).
  • Programs can be seen as arithmetic circuits/neural networks.
  • Parameters can be trained by gradient descent and back-propagation.

• Future work
  • Perform experiments.
  • Parameter learning by Expectation Maximization.
  • Perform also structure learning.
THANKS FOR LISTENING AND ANY QUESTIONS?