Deep Probabilistic Logic Programming

Arnaud Nguembang Fadja   Evelina Lamma   Fabrizio Riguzzi

Dipartimento di Ingegneria – University of Ferrara

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

Introduction

- 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.
- Hierarchical PLP is truth-functional and equivalent to the product fuzzy logic.
- Inference then is much cheaper as a simple dynamic programming algorithm similar to PRISM is sufficient

Probabilistic Logic Programming

- Distribution Semantics [Sato ICLP95]
- A probabilistic logic program defines a probability distribution over normal logic programs (called instances or possible worlds or simply worlds)
- The distribution is extended to a joint distribution over worlds and interpretations (or queries)
- The probability of a query is obtained from this distribution

PLP under the Distribution Semantics

- 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:

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

,

- 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:

$$\begin{array}{l} advisedby(A,B) : 0.3 :- \\ \quad student(A), professor(B), project(C,A), project(C,B). \\ advisedby(A,B) : 0.6 :- \\ \quad student(A), professor(B), ta(C,A), taughtby(C,B). \end{array}$$

Distribution Semantics

- Case of no function symbols: finite Herbrand universe, finite set of groundings of each clause
- Atomic choice: selection of the k-th atom for grounding $C_i\theta_j$ of clause C_i
- Represented with the triple (C_i, θ_j, k)
- Example $C_1 = advisedby(A, B) :$
  $0.3 :- student(A), professor(B), project(C, A), project(C, B).$,
  $(C_1, \{A/bob, B/harry, C/p1\}, 1)$
- Composite choice κ: consistent set of atomic choices
- The probability of composite choice κ is

$$P(\kappa) = \prod_{(C_i,\theta_j,k)\in\kappa} \pi_{ik}$$

Distribution Semantics

- Selection σ: a total composite choice (one atomic choice for every grounding of each clause)
- A selection σ identifies a logic program w_σ called world
- The probability of w_σ is $P(w_\sigma) = P(\sigma) = \prod_{(C_i,\theta_j,k)\in\sigma} \pi_{ik}$
- Finite set of worlds: $W_T = \{w_1, \ldots, w_m\}$
- $P(w)$ distribution over worlds: $\sum_{w\in W_T} P(w) = 1$

Distribution Semantics

- Ground query q
- We consider only *sound* LPADs, where each possible world has a total well-founded model, so $w \models q$ means that the query q is true in the well-founded model of the program w.
- $P(q|w) = 1$ if q is true in w and 0 otherwise
- $P(q) = \sum_w P(q, w) = \sum_w P(q|w)P(w) = \sum_{w \models q} P(w)$

Example

- UW-CSE domain: the objective is to predict the “advised by” relation between students and professors.

$$
\begin{array}{l}
advisedby(A,B) : 0.3 :- \\
\quad student(A), professor(B), project(C,A), project(C,B). \\
advisedby(A,B) : 0.6 :- \\
\quad student(A), professor(B), ta(C,A), taughtby(C,B).
\end{array}
$$

- $q = advisedby(harry, ben)$ where *harry* is a student, *ben* is a professor, they have one joint project and *ben* teaches one course where *harry* is a TA. So

$$P(advisedby(harry, ben)) = 0.3 \cdot 0.6 + 0.3 \cdot 0.4 + 0.7 \cdot 0.6 = 0.72$$

Hierarchical PLP

- We want to compute the probability of atoms for a predicate r: $r(\boldsymbol{t})$, where $\boldsymbol{t}$ is a vector of constants.
- $r(\boldsymbol{t})$ 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.

Hierarchical PLP

- 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.
- Extreme form of program stratification: stronger than acyclicity [Apt NGC91] because it is imposed on the predicate dependency graph, and is also stronger than stratification [Chandra, Harel JLP85] that allows clauses with positive literals built on predicates in the same layer.
- It prevents inductive definitions and recursion in general, thus making the language not Turing-complete.

Hierarchical PLP

- Generic clause C:

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

  where $\phi(\boldsymbol{X}, \boldsymbol{Y})$ is a conjunction of literals for the input predicates using variables $\boldsymbol{X}, \boldsymbol{Y}$.
- $b_i(\boldsymbol{X}, \boldsymbol{Y})$ for $i = 1, \ldots, m$ is a literal built on a hidden predicate.
- $\boldsymbol{Y}$ is a possibly empty vector of variables existentially quantified with scope the body.
- Literals for hidden predicates must use the whole set of variables $\boldsymbol{X}, \boldsymbol{Y}$.
- The predicate of each $b_i(\boldsymbol{X}, \boldsymbol{Y})$ does not appear elsewhere in the body of C or in the body of any other clause.

Hierarchical PLP

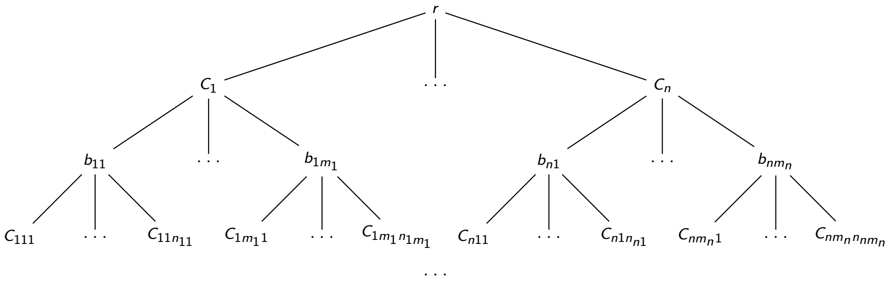
- A generic program defining r is thus:

$$
\begin{aligned}
C_1 = r(\boldsymbol{X}) : \pi_1 \quad &:- \quad \phi_1, b_{11}, \ldots, b_{1m_1} \\
&\cdots \\
C_n = r(\boldsymbol{X}) : \pi_n \quad &:- \quad \phi_n, b_{n1}, \ldots, b_{nm_n} \\
C_{111} = r_{11}(\boldsymbol{X}) : \pi_{111} \quad &:- \quad \phi_{111}, b_{1111}, \ldots, b_{111m_{111}} \\
&\cdots \\
C_{11n_{11}} = r_{11}(\boldsymbol{X}) : \pi_{11n_{11}} \quad &:- \quad \phi_{11n_{11}}, b_{11n_{11}1}, \ldots, b_{11n_{11}m_{11n_{11}}} \\
&\cdots \\
C_{n11} = r_{n1}(\boldsymbol{X}) : \pi_{n11} \quad &:- \quad \phi_{n11}, b_{n111}, \ldots, b_{n11m_{n11}} \\
&\cdots \\
C_{n1n_{n1}} = r_{n1}(\boldsymbol{X}) : \pi_{n1n_{n1}} \quad &:- \quad \phi_{n1n_{n1}}, b_{n1n_{n1}1}, \ldots, b_{n1n_{n1}m_{n1n_{n1}}} \\
&\cdots
\end{aligned}
$$

Example

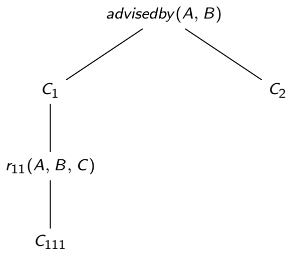
$$
\begin{aligned}
C_1 &= advisedby(A,B) : 0.3 :- \\
&\quad student(A), professor(B), project(C,A), project(C,B), \\
&\quad r_{11}(A,B,C). \\
C_2 &= advisedby(A,B) : 0.6 :- \\
&\quad student(A), professor(B), ta(C,A), taughtby(C,B). \\
C_{111} &= r_{11}(A,B,C) : 0.2 :- \\
&\quad publication(D,A,C), publication(D,B,C).
\end{aligned}
$$

Program Tree



Example

$$
\begin{aligned}
C_1 &= advisedby(A, B) : 0.3 :- \\
&\quad student(A), professor(B), project(C, A), project(C, B), \\
&\quad r_{11}(A, B, C). \\
C_2 &= advisedby(A, B) : 0.6 :- \\
&\quad student(A), professor(B), ta(C, A), taughtby(C, B). \\
C_{111} &= r_{11}(A, B, C) : 0.2 :- \\
&\quad publication(D, A, C), publication(D, B, C).
\end{aligned}
$$



Hierarchical PLP

- Writing programs in hierarchical PLP may be unintuitive for humans because of the need of satisfying the constraints and because the hidden predicates may not have a clear meaning.
- The structure of the program should be learned by means of a specialized algorithm
- Hidden predicates generated by a form of predicate invention.

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 $C_{\boldsymbol{p}i} = a_{\boldsymbol{p}} : \pi_{\boldsymbol{p}i} :- \ b_{\boldsymbol{p}i1}, \ldots, b_{\boldsymbol{p}im_{\boldsymbol{p}}}$. where $\boldsymbol{p}$ is a path in the program tree
- $P(b_{\boldsymbol{p}i1}, \ldots, b_{\boldsymbol{p}im_{\boldsymbol{p}}}) = \prod_{i=k}^{m_{\boldsymbol{p}}} P(b_{\boldsymbol{p}ik})$ and $P(b_{\boldsymbol{p}ik}) = 1 - P(a_{\boldsymbol{p}ik})$ if $b_{\boldsymbol{p}ik} = \neg a_{\boldsymbol{p}ik}$.
- 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

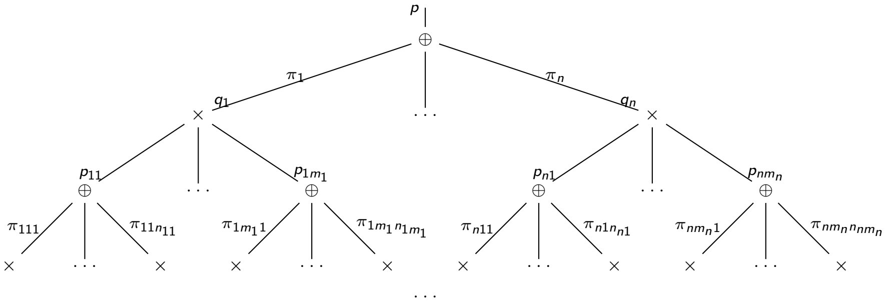
- Hidden predicates: to compute $P(a_{\boldsymbol{p}})$ we need to take into account the contribution of every ground clause for the predicate of $a_{\boldsymbol{p}}$.
- Suppose these clauses are $\{C_{\boldsymbol{p}1}, \ldots, C_{\boldsymbol{p}o_{\boldsymbol{p}}}\}$.
- If we have two clauses, $P(a_{\boldsymbol{p}i}) = 1 - (1 - \pi_{\boldsymbol{p}1} \cdot P(body(C_{\boldsymbol{p}1})) \cdot (1 - \pi_{\boldsymbol{p}2} \cdot P(body(C_{\boldsymbol{p}2})))$
- $p \oplus q \triangleq 1 - (1 - p) \cdot (1 - q)$.
- 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:

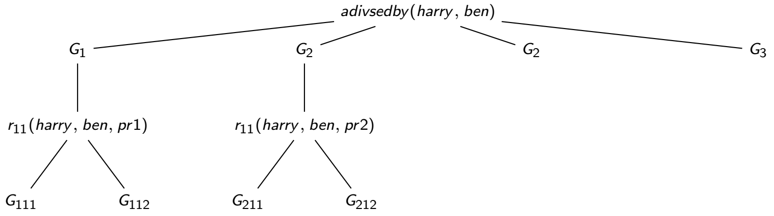


- The arithmetic circuit can be interpreted as a deep neural network where nodes have the activation functions $\times$ and $\oplus$

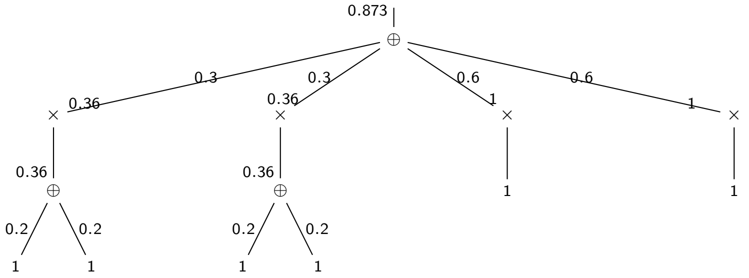
Example

$$
\begin{aligned}
G_1 &= \mathit{advisedby}(\mathit{harry},\mathit{ben}) : 0.3 :- \\
&\quad \mathit{student}(\mathit{harry}), \mathit{professor}(\mathit{ben}), \mathit{project}(\mathit{pr}1,\mathit{harry}), \\
&\quad \mathit{project}(\mathit{pr}1,\mathit{ben}), r_{11}(\mathit{harry},\mathit{ben},\mathit{pr}1). \\
G_2 &= \mathit{advisedby}(\mathit{harry},\mathit{ben}) : 0.3 :- \\
&\quad \mathit{student}(\mathit{harry}), \mathit{professor}(\mathit{ben}), \mathit{project}(\mathit{pr}2,\mathit{harry}), \\
&\quad \mathit{project}(\mathit{pr}2,\mathit{ben}), r_{11}(\mathit{harry},\mathit{ben},\mathit{pr}2). \\
G_3 &= \mathit{advisedby}(\mathit{harry},\mathit{ben}) : 0.6 :- \\
&\quad \mathit{student}(\mathit{harry}), \mathit{professor}(\mathit{ben}), \mathit{ta}(c1,\mathit{harry}), \mathit{taughtby}(c1,\mathit{ben}). \\
G_4 &= \mathit{advisedby}(\mathit{harry},\mathit{ben}) : 0.6 :- \\
&\quad \mathit{student}(\mathit{harry}), \mathit{professor}(\mathit{ben}), \mathit{ta}(c2,\mathit{harry}), \mathit{taughtby}(c2,\mathit{ben}). \\
G_{111} &= r_{11}(\mathit{harry},\mathit{ben},\mathit{pr}1) : 0.2 :- \\
&\quad \mathit{publication}(p1,\mathit{harry},\mathit{pr}1), \mathit{publication}(p1,\mathit{ben},\mathit{pr}1). \\
G_{112} &= r_{11}(\mathit{harry},\mathit{ben},\mathit{pr}1) : 0.2 :- \\
&\quad \mathit{publication}(p2,\mathit{harry},\mathit{pr}1), \mathit{publication}(p2,\mathit{ben},\mathit{pr}1). \\
G_{211} &= r_{11}(\mathit{harry},\mathit{ben},\mathit{pr}2) : 0.2 :- \\
&\quad \mathit{publication}(p3,\mathit{harry},\mathit{pr}2), \mathit{publication}(p3,\mathit{ben},\mathit{pr}2). \\
G_{212} &= r_{11}(\mathit{harry},\mathit{ben},\mathit{pr}2) : 0.2 :- \\
&\quad \mathit{publication}(p4,\mathit{harry},\mathit{pr}2), \mathit{publication}(p4,\mathit{ben},\mathit{pr}2).
\end{aligned}
$$

Example



$adivsedby(harry, ben)$
G_1
G_2
G_2
G_3
$r_{11}(harry, ben, pr1)$
$r_{11}(harry, ben, pr2)$
G_{111}
G_{112}
G_{211}
G_{212}



0.873
⊕
0.3
0.3
0.6
0.6
0.36
0.36
1
1
×
×
×
×
0.36
0.36
⊕
⊕
1
1
0.2
0.2
0.2
0.2
1
1
1
1

Building the Network

- The network can be built by performing inference using tabling and answer subsumption
- PITA(IND,IND) [Riguzzi CJ14] is a program transformation that adds an extra argument to each subgoal of the program and of the query to store the probability of answers to the subgoal
- When a subgoal returns, the extra argument will be instantiated to the probability of the ground atom that corresponds to the subgoal without the extra argument.
- In programs of hierarchical PLP, when a subgoal returns the original arguments are guaranteed to be instantiated.
- PITA(IND,IND) adds literals to bodies that combine the extra arguments of the subgoals

Building the Network

- The contributions of multiple groundings of multiple clauses are combined by means of tabling with answer subsumption.
- Tabling: keep a store of the subgoals encountered in a derivation together with answers to these subgoals.
- If one of the subgoals is encountered again, its answers are retrieved from the store rather than recomputing them.
- Tabling reduces computation time and ensures termination for a large class of programs [Swift TPLP12].
- Answer subsumption [Swift TPLP12] is a tabling feature that, when a new answer for a tabled subgoal is found, combines old answers with the new one.
- In PITA(IND, IND) the combination operator is probabilistic sum.

Parameter Learning

- Parameter learning by EM or backpropagation.
- Inference has to be performed repeatedly on the same program with different values of the parameters.
- PITA(IND,IND) can build a representation of the arithmetic circuit, instead of just computing the probability.
- Extra argument used to store a term representing the circuit
- Implementing EM would adapt the algorithm of [Bellodi and Riguzz IDA13] for hierarchical PLP.

Related Work

- [Sourek et al NIPS15]: build deep neural networks using a template expressed as a set of weighted rules.
- Nodes for ground atoms and ground rules
- Values of ground rule nodes aggregated to compute the value of atom nodes.
- AggregatIon in two steps, first the contributions of different groundings of the same rule sharing the same head and then the contributions of groundings for different rules.
- Proposal parametric in the activation functions of ground rule nodes.
- Example: two families of activation functions that are inspired by Lukasiewicz fuzzy logic.
- We build a neural network whose output is the probability of the example according to the distribution semantics.

Related Work

- Edward [Tran et al. ICLR17]: Turing-complete probabilistic programming language
- Programs in Edward define computational graphs and inference is performed by stochastic graph optimization using TensorFlow.
- Hierarchical PLP is not Turing-complete as Edward but ensures fast inference by circuit evaluation.
- Being based on logic it handles well domains with multiple entities connected by relationships.
- Similarly to Edward, hierarchical PLP can be compiled to TensorFlow

Related Work

- Probabilistic Soft Logic (PSL) [Bach et al. arXiv15]: Markov Logic with atom random variables taking continuous values in $[0, 1]$ and logic formulas interpreted using Lukasiewicz fuzzy logic.
- PSL defines a joint probability distribution over fuzzy variables, while the random variables in hierarchical PLP are still Boolean and the fuzzy values are the probabilities that are combined with the product fuzzy logic.
- The main inference problem in PSL is MAP rather than MARG as in hierarchical PLP.

Related Work

- Sum-product networks [Poon, Domingos UAI11]: hierarchical PLP circuits can be seen as sum-product networks where children of sum nodes are not mutually exclusive but independent and each product node has a leaf child that is associated to a hidden random variable.
- Sum-product networks represent a distribution over input data while programs in hierarchical PLP describe only a distribution over the truth values of the query.
- Inference in hierarchical PLP is in a way “lifted”: the probability of the ground atoms can be computed knowing only the sizes of the populations of individuals that can instantiate the existentially quantified variables

Conclusions and Future Work

- Conclusions
  - 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 or EM
- Future work
  - Develop backpropagation and EM formulas
  - Perform also structure learning

THANKS FOR LISTENING AND ANY QUESTIONS ?