EM over Binary Decision Diagrams for Probabilistic Logic Programs

Elena Bellodi   Fabrizio Riguzzi

ENDIF – University of Ferrara, Italy
elena.bellodi@unife.it, fabrizio.riguzzi@unife.it

Outline

Probabilistic Logic Programming

- Logic + Probability: useful to model domains with *complex* and *uncertain* relationships among entities
- Many approaches proposed in: *Logic Programming, Uncertainty in AI, Machine Learning, Databases*
- **Logic Programming**: Distribution Semantics [Sato, 1995]
  - Independent Choice Logic,PRISM, ProbLog, Logic Programs with Annotated Disjunctions (LPADs)[Vennekens et al., 2004],...
  - They define a probability distribution over normal logic programs (possible worlds)
  - They differ in the definition of the probability distribution
  - The distribution is extended to a joint distribution over worlds and queries
  - The probability of a query is obtained from this distribution by marginalization

Logic Programs with Annotated Disjunctions (LPAD)

- *Example: development of an epidemic or pandemic, if somebody has the flu and the climate is cold.*

$$
\begin{aligned}
C_1 &= \mathit{epidemic} : 0.6; \mathit{pandemic} : 0.3; \mathit{null{:}0.1} : -\mathit{flu}(X), \mathit{cold}. \\
C_2 &= \mathit{cold} : 0.7; \mathit{null{:}0.3}. \\
C_3 &= \mathit{flu}(\mathit{david}). \\
C_4 &= \mathit{flu}(\mathit{robert}).
\end{aligned}
$$

- Worlds obtained by selecting *only one atom from the head* of every grounding of each rule

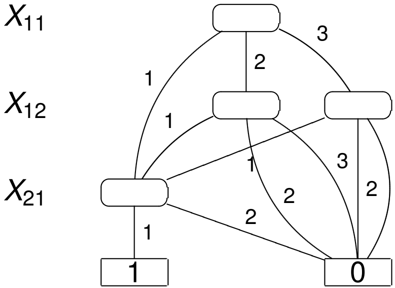
Inference

- Explanation: set of probabilistic choices that ensure the entailment of the goal
- Covering set of explanations: every world where the query is true is consistent with at least one explanation
- A covering set of explanations for :- epidemic. is $\{\kappa_1, \kappa_2\}$
  - $\kappa_1 = \{(C_1, \theta_1 = \{X/david\}, 1), (C_2, \{\}, 1)\}$
  - $\kappa_2 = \{(C_1, \theta_2 = \{X/robert\}, 1), (C_2, \{\}, 1)\}$
- Explanations are not mutually exclusive
- From a covering set of explanations the probability of the query Q is computed by means of Decision Diagrams

Multivalued Decision Diagrams (MDD)

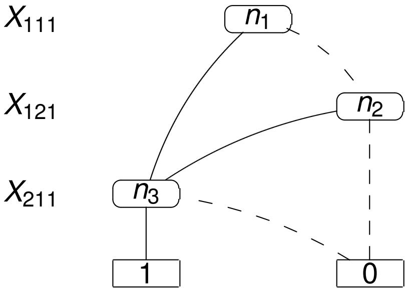
- Multivalued Decision Diagrams (MDDs) represent a Boolean function $f(\mathbf{X})$ on a set of multivalued variables X_{ij} $\rightarrow$ ground clause $C_i\theta_j$, with domain 1,..., $|head(C_i)|$
- In a MDD a path to a 1-leaf corresponds to an explanation for Q
- The various paths are mutually exclusive

$$f(\mathbf{X}) = (X_{11} = 1 \wedge X_{21} = 1) \vee (X_{12} = 1 \wedge X_{21} = 1)$$



Binary Decision Diagrams (BDD)

- MDDs can be converted into Binary Decision Diagrams with Boolean variables
- multivalued variable X_{ij} with n_i values $\rightarrow$ $n_i - 1$ Boolean variables $X_{ij1},\dots,X_{ijn_{i-1}}$
- from $f(\mathbf{X}) = (X_{11} = 1 \wedge X_{21} = 1) \vee (X_{12} = 1 \wedge X_{21} = 1)$
  to $f(\mathbf{X}) = ((X_{111} \wedge \overline{X_{112}}) \wedge X_{211}) \vee ((X_{121} \wedge \overline{X_{122}}) \wedge X_{211})$



Weight Learning for LPADs

- Problem: model of the domain known

  VS

  weights (numeric parameters) unknown
- Weight learning: inference of weights from data
- Given
  - a LPAD: a probabilistic logical model with **unknown probabilities**
  - data: a set of interpretations
  - **Find** the values of the **probabilities** that maximize the probability of the data given the model
- Expectation Maximization (EM) algorithm
  - iterative method for problems with incomplete data
  - *Expectation* step: *estimates missing data* given observed data + current estimate of parameters
  - *Maximization* step: *computes the parameters* using estimates of E step

EMBLEM: *EM over Bdds for probabilistic Logic programs Efficient Mining*

- EM over BDDs proposed in [Ishihata et al., 2008]
- Input: a LPAD; logical interpretations (data); *target* predicate(s)
- all ground atoms in the interpretations for the target predicate(s) correspond to as many queries
- BDDs encode the disjunction of explanations for each query Q
- EM algorithm directly over the BDDs
  - *missing data*: the number of times that *i-th* head atom has been selected from groundings of the clauses used in the proof of the queries

EM Algorithm

- *Expectation step* (synthesis)
  1. Computes $P(X_{ijk} = x, Q)$ and $P(Q)$
  2. expected counts $\mathbf{E}[c_{ikx}] = \frac{\sum_{j \in g(i)} P(X_{ijk}=x,Q)}{P(Q)}$
     for all rules C_i and $k = 1, ..., n_i - 1$, where c_{ikx} is the number of times a binary variable X_{ijk} takes value $x \in \{0, 1\}$, and for all values of $j \in g(i) = \{j|\theta_j \textit{ is a substitution grounding } C_i\}$
- *Maximization step*
  - Updates parameters π_{ik} representing $P(X_{ijk} = 1)$
  - π_{ik} = $E[c_{ik1}|Q]$ / ($E[c_{ik0}|Q]$ + $E[c_{ik1}|Q]$)

Expectation Computation

- $P(X_{ijk} = x, Q) = \sum_{n \in N(Q), v(n)=X_{ijk}} F(n)B(child_x(n))\pi_{ikx} = \sum_{n \in N(Q), v(n)=X_{ijk}} e^x(n)$
  - π_{ikx} is π_{ik} if $x = 1$ and $(1 - \pi_{ik})$ if $x = 0$
  - $F(n)$ is the forward probability, the probability mass of the paths from the root to n
  - $B(n)$ is the backward probability, the probability mass of paths from n to the 1-leaf
  - $e^x(n)$ is the probability mass of paths from the root to the 1 leaf passing through the x branch of n

Computation of the forward probability

```
procedure GetForward(root)
    F(root) = 1   F(n) = 0 for all nodes          ▷ BDD traversed from root to leaves
    for l = 1 to levels do                                        ▷ BDD levels
        for all node ∈ Nodes(l) do                          ▷ Nodes of one level
            Let X_ijk be v(node), the variable associated to node
            if child_0(node) is not terminal then   ▷ node's child connected by 0-branch
                F(child_0(node)) = F(child_0(node)) + F(node) · (1 − π_ik)   ▷ π_ik: probability
                Add child_0(node) to Nodes(level(child_0(node)))
            end if
            if child_1(node) is not terminal then
                F(child_1(node)) = F(child_1(node)) + F(node) · π_ik
                Add child_1(node) to Nodes(level(child_1(node)))
            end if
        end for
    end for
end procedure
```

- For all nodes of a level the forward probabilities *of their children* are computed, by using probabilities π_{ik} associated to the outgoing edges

Computation of the backward probability

function GETBACKWARD(*node*)
  if *node* is a terminal **then** ▷ BDD traversed recursively from root up to leaves
    return *value*(*node*) ▷ leaves return 0 or 1
  else
    Let X_{ijk} be $v(node)$
    $B(child_0(node))$ =GETBACKWARD($child_0(node)$) ▷ recursive calls
    $B(child_1(node))$ =GETBACKWARD($child_1(node)$)
    $e^0(node) = F(node) \cdot B(child_0(node)) \cdot (1 - \pi_{ik})$ ▷ $F(node)$ from `GetForward`
    $e^1(node) = F(node) \cdot B(child_1(node)) \cdot \pi_{ik}$
    $\eta^0(i,k) = \eta^0_t(i,k) + e^0(node)$ ▷ update of $\eta^x(i,k)$ to **build** $P(X_{ijk} = x, Q)$
    $\eta^1(i,k) = \eta^1_t(i,k) + e^1(node)$
    return $B(child_0(node)) \cdot (1 - \pi_{ik}) + B(child_1(node)) \cdot \pi_{ik}$
  end if
end function

- at the end of all recursive calls, the function returns B(root) = probability of the query P(Q)

Experiments - settings

- EMBLEM is implemented in Yap Prolog
- Comparison with other systems
  - for learning and inference under the distribution semantics:
    - RIB [Riguzzi and di Mauro, 2011]
    - CEM [Riguzzi, 2007]
    - LeProblog [De Raedt et al., 2007]
  - for learning and inference in Markov Logic Networks: Alchemy
- Datasets composed of 5 mega-interpretations → Five-fold cross validation
- Performance evaluation
  - Area Under the PR (Precision-Recall) Curve

Experiments - datasets

- **IMDB** - the Internet Movie DataBase: movies, actors, directors.
- Input LPADs
  - target predicate $sameperson(per1, per2)$

$$
\begin{array}{l}
sameperson(X,Y):p :- movie(M,X), movie(M,Y).\\
sameperson(X,Y):p :- actor(X), actor(Y), workedunder(X,Z), workedunder(Y,Z).\\
sameperson(X,Y):p :- gender(X,Z), gender(Y,Z).\\
sameperson(X,Y):p :- director(X), director(Y), genre(X,Z), genre(Y,Z).
\end{array}
$$

  - target predicate $samemovie(mov1, mov2)$

$$
\begin{array}{l}
samemovie(X,Y):p :- movie(X,M), movie(Y,M), actor(M).\\
samemovie(X,Y):p :- movie(X,M), movie(Y,M), director(M).\\
samemovie(X,Y):p :- movie(X,A), movie(Y,B), actor(A), director(B), workedunder(A,B).\\
samemovie(X,Y):p :- movie(X,A), movie(Y,B), director(A), director(B), genre(A,G), genre(B,G).
\end{array}
$$

Average of the AUCPR

Dataset	EMBLEM	RIB	LeProblog	CEM	Alchemy
IMDB-SP	**0.202**	0.199	0.096	0.202	0.107
IMDB-SM	**1.000**	memory error	0.933	0.537	0.820

Experiments - datasets

- **CORA**: citations to computer science research papers [Singla and Domingos, 2005]
- target predicate: $samebib(cit1, cit2)$, to determine which citations are referring to the same paper
- Input LPADs
  - 559 clauses

$$
\begin{array}{l}
samebib(B,C):p:-author(B,D),author(C,E),sameauthor(D,E).\\
samebib(B,C):p:-title(B,D),title(C,E),sametitle(D,E).\\
samebib(B,C):p:-venue(B,D),venue(C,E),samevenue(D,E).\\
samevenue(B,C):p:-haswordvenue(B,W^*),haswordvenue(C,W^*).\ ^*\text{W instantiated to all words}\\
sametitle(B,C):p:-haswordtitle(B,W^*),haswordtitle(C,W^*).\\
sameauthor(B,C):p:-haswordauthor(B,W^*),haswordauthor(C,W^*).
\end{array}
$$

  - + 4 transitive rules (CoraT)

$$
samebib/title/author/venue(A,B):p:- \quad samebib/title/author/venue(A,C),\ samebib/title/author/venue(C,B).
$$

Average of the AUCPR

Dataset	EMBLEM	RIB	LeProblog	CEM	Alchemy
CORA	**0.995**	0.939	0.905	0.995	0.469
CORAT	**0.991**	not appl.	0.970	memory error	memory error

Experiments - datasets

- **UWCSE**: information about the Computer Science department of the University of Washington [Kok and Domingos, 2010]
- target predicate: *advisedBy*/2, a person is advised by another person
- Input LPAD: 86 clauses, such as

$advisedby(S, P) : p : -courselevel(C, level\_500), taughtby(C, P, Q), ta(C, S, Q).$
$tempadvisedby(S, P) : p : -courselevel(C, level\_500), taughtby(C, P, Q), ta(C, S, Q).$
$professor(P) : p : -courselevel(C, level\_500), taughtby(C, P, Q).$

Average of the AUCPR

EMBLEM	RIB	LeProblog	CEM	Alchemy
0.883	0.588	0.270	0.644	0.294

Conclusions and future works

- `EMBLEM` can be applied to all languages based on the distribution semantics, since there are transformations with linear complexity that can convert a program in one language into the others
- Able to solve greater problems, where other algorithms do not terminate
- Higher PR areas with same learning time as the fastest other algorithm
- Higher PR areas with longer learning time

- *In progress*: structure learning of LPADs (clauses+parameters)

References I

De Raedt, L., Kimmig, A., and Toivonen, H. (2007).
ProbLog: A probabilistic prolog and its application in link discovery.
In *International Joint Conference on Artificial Intelligence*, pages 2462–2467. AAAI Press.

Ishihata, M., Kameya, Y., Sato, T., and Minato, S. (2008).
Propositionalizing the em algorithm by bdds.
Technical Report TR08-0004, Dep. of Computer Science, Tokyo Institute of Technology.

Kok, S. and Domingos, P. (2010).
Learning markov logic networks using structural motifs.
pages 551–558. Omnipress.

Riguzzi, F. (2007).
A top-down interpreter for LPAD and CP-Logic.
In Basili, R. and Pazienza, M. T., editors, *Proceedings of the 10th Congress of the Italian Association for Artificial Intelligence*, volume 4733 of *LNCS*, pages 109–120. Springer.

Riguzzi, F. and di Mauro, N. (2011).
Applying the information bottleneck to statistical relational learning.
Machine Learning.
To appear.

References II

Sato, T. (1995).
A statistical learning method for logic programs with distribution semantics.
In *International Conference on Logic Programming*, pages 715–729. MIT Press.

Singla, P. and Domingos, P. (2005).
Discriminative training of Markov logic networks.
In *National Conference on Artificial Intelligence*, pages 868–873. AAAI Press/The MIT Press.

Vennekens, J., Verbaeten, S., and Bruynooghe, M. (2004).
Logic programs with annotated disjunctions.
In *International Conference on Logic Programming*, volume 3131 of *LNCS*, pages 195–209. Springer.