Probabilistic Logics in Machine Learning

ML@Unife group members:
Elena Bellodi\textsuperscript{2}  
Giuseppe Cota\textsuperscript{1}  
Arnaud Nguembang Fadja\textsuperscript{1}  
Evelina Lamma\textsuperscript{1}  
Fabrizio Riguzzi\textsuperscript{2}  
Riccardo Zese\textsuperscript{1}

\textsuperscript{1} Dipartimento di Ingegneria – University of Ferrara

\textsuperscript{2} Dipartimento di Matematica e Informatica – University of Ferrara

name.surname@unife.it
Outline

1. Logic
2. Combining Logic and Probability
3. Applications
4. Inference
5. Learning

Probabilistic Logics in Machine Learning
Logic

Useful to model domains with complex relationships among entities

Various forms:

- First Order Logic
- Logic Programming
- Description Logics
First Order Logic

- Very expressive
- Open World Assumption
- Undecidable

\[
\forall x \ Intelligent(x) \rightarrow GoodMarks(x)
\]

\[
\forall x, y \ Friends(x, y) \rightarrow (Intelligent(x) \leftrightarrow Intelligent(y))
\]
Logic Programming

- A subset of First Order Logic
- Closed World Assumption
- Turing complete
- Prolog

```prolog
flu(bob).
hay_fever(bob).
sneezing(X) ← flu(X).
sneezing(X) ← hay_fever(X).
```
Description Logics

- Subsets of First Order Logic
- Open World Assumption
- Decidable, efficient inference
- Special syntax using concepts (unary predicates) and roles (binary predicates)

```
fluffy : Cat
tom : Cat
Cat ⊑ Pet
∃hasAnimal.Pet ⊑ NatureLover
(kevin, fluffy) : hasAnimal
(kevin, tom) : hasAnimal
```

1. [http://www.cs.man.ac.uk/~ezolin/dl/](http://www.cs.man.ac.uk/~ezolin/dl/), a nice Web application that shows decidability and complexity issues depending on which operator you decide to include/exclude in your logic, with extensive references
Combining Logic and Probability

- Logic does not handle well uncertainty
- Graphical models do not handle well relationships among entities
- Solution: combine the two
- Many approaches proposed in the areas of Logic Programming, Uncertainty in AI, Machine Learning, Databases, Knowledge Representation
  - Distribution Semantics [Sato ICLP95]: given a query, find the worlds (normal logic programs) where the query is true and summing their probability
Logic Programs with Annotated Disjunctions

\[
\begin{align*}
sneezing(X) : 0.7 \lor \text{null} : 0.3 & \leftarrow \text{flu}(X). \\
sneezing(X) : 0.8 \lor \text{null} : 0.2 & \leftarrow \text{hay\_fever}(X). \\
\text{flu}(bob). \\
\text{hay\_fever}(bob). \\
\end{align*}
\]

- Distributions over the head of rules
- \textit{null} does not appear in the body of any rule
- Worlds obtained by selecting one atom from the head of every grounding of each clause
Combining Logic and Probability

Example Program (LPAD) Worlds

```
sneezing(bob) ← flu(bob).
sneezing(bob) ← hay_fever(bob).
flu(bob).
hay_fever(bob).
P(w₁) = 0.7 × 0.8

sneezing(bob) ← flu(bob).
null ← hay_fever(bob).
flu(bob).
hay_fever(bob).
P(w₂) = 0.3 × 0.8

sneezing(bob) ← flu(bob).
null ← hay_fever(bob).
flu(bob).
hay_fever(bob).
P(w₃) = 0.7 × 0.2

null ← flu(bob).
null ← hay_fever(bob).
flu(bob).
hay_fever(bob).
P(w₄) = 0.3 × 0.2

P(Q) = \sum_{w \in W_T} P(Q, w) = \sum_{w \in W_T} P(Q|w)P(w) = \sum_{w \in W_T: w|=Q} P(w)

• sneezing(bob) is true in 3 worlds
• P(sneezing(bob)) = 0.7 × 0.8 + 0.3 × 0.8 + 0.7 × 0.2 = 0.94
```
sneezing(X) ← flu(X), flu_sneezing(X).

sneezing(X) ← hay_fever(X), hay_fever_sneezing(X).

flu(bob).

hay_fever(bob).

0.7 :: flu_sneezing(X).

0.8 :: hay_fever_sneezing(X).

- Distributions over facts
- Worlds obtained by selecting or not every grounding of each probabilistic fact
Example Program (ProbLog) Worlds

- 4 worlds

\[ \text{sneezing}(X) \leftarrow \text{flu}(X), \text{flu}_\text{sneezing}(X). \]
\[ \text{sneezing}(X) \leftarrow \text{hay}_\text{fever}(X), \text{hay}_\text{fever}_\text{sneezing}(X). \]
\[ \text{flu}(bob). \]
\[ \text{hay}_\text{fever}(bob). \]
\[ \text{flu}_\text{sneezing}(bob). \]
\[ \text{hay}_\text{fever}_\text{sneezing}(bob). \]
\[ \text{flu}_\text{sneezing}(bob). \]
\[ \text{hay}_\text{fever}_\text{sneezing}(bob). \]

\[ P(w_1) = 0.7 \times 0.8 \]
\[ P(w_2) = 0.3 \times 0.8 \]
\[ P(w_3) = 0.7 \times 0.2 \]
\[ P(w_4) = 0.3 \times 0.2 \]

- \text{sneezing}(bob) is true in 3 worlds

\[ P(\text{sneezing}(bob)) = 0.7 \times 0.8 + 0.3 \times 0.8 + 0.7 \times 0.2 = 0.94 \]
Probabilistic Logic Programming Online

- [http://cplint.eu/](http://cplint.eu/)
  - Inference (knowledge compilation, Monte Carlo)
  - Parameter learning (EMBLEM)
  - Structure learning (SLIPCOVER)

  - Inference (knowledge compilation, Monte Carlo)
  - Parameter learning (LFI-ProbLog)
Expressive Power

- All languages under the distribution semantics have the same expressive power
- LPADs have the most general syntax
- There are transformations that can convert each one into the others
- ProbLog to LPAD: direct mapping
Description Logics

- DISPONTE: “DIstribution Semantics for Probabilistic ONTologiiEs” [Riguzzi et al. SWJ15]
- Probabilistic axioms:
  - $p :: E$
    - e.g., $p :: C \sqsubseteq D$ represents the fact that we believe in the truth of $C \sqsubseteq D$ with probability $p$.
- DISPONTE applies the distribution semantics of probabilistic logic programming to description logics
World $w$: regular DL KB obtained by selecting or not the probabilistic axioms

Probability of a query $Q$ given a world $w$: $P(Q|w) = 1$ if $w \models Q$, 0 otherwise

Probability of $Q$

$$P(Q) = \sum_w P(Q, w) = \sum_w P(Q|w)P(w) = \sum_{w:w \models Q} P(w)$$
Example

0.4 :: fluffy : Cat
0.3 :: tom : Cat
0.6 :: Cat ⊑ Pet
∃hasAnimal.Pet ⊑ NatureLover
(kevin, fluffy) : hasAnimal
(kevin, tom) : hasAnimal

\[ P(\text{kevin} : \text{NatureLover}) = 0.4 \times 0.3 \times 0.6 + 0.4 \times 0.7 \times 0.6 + 0.6 \times 0.3 \times 0.6 = 0.348 \]
Reasoning Tasks

- Inference: we want to compute the probability of a query given the model and, possibly, some evidence.
- Weight learning: we know the structural part of the model (the logic formulas) but not the numeric part (the weights) and we want to infer the weights from data.
- Structure learning: we want to infer both the structure and the weights of the model from data.
Link prediction: given a (social) network, compute the probability of the existence of a link between two entities (UWCSE)

\[
\text{advisedby}(X, Y) : 0.7 :- \\
\quad \text{publication}(P, X), \\
\quad \text{publication}(P, Y), \\
\quad \text{student}(X).
\]
Classify web pages on the basis of the link structure (WebKB)

coursePage(Page1): 0.3 :- linkTo(Page2, Page1), coursePage(Page2).
coursePage(Page1): 0.6 :- linkTo(Page2, Page1), facultyPage(Page2).
...
coursePage(Page): 0.9 :- has('syllabus', Page).
...
Entity resolution: identify identical entities in text or databases

samebib(A,B): 0.9 :-
   samebib(A,C), samebib(C,B).
sameauthor(A,B): 0.6 :-
   sameauthor(A,C), sameauthor(C,B).
sametitle(A,B): 0.7 :-
   sametitle(A,C), sametitle(C,B).
samevenue(A,B): 0.65 :-
   samevenue(A,C), samevenue(C,B).
samebib(B,C): 0.5 :-
   author(B,D), author(C,E), sameauthor(D,E).
samebib(B,C): 0.7 :-
   title(B,D), title(C,E), sametitle(D,E).
samebib(B,C): 0.6 :-
   venue(B,D), venue(C,E), samevenue(D,E).
samevenue(B,C): 0.3 :-
   haswordvenue(B,logic),
   haswordvenue(C,logic).
...
Chemistry: given the chemical composition of a substance, predict its mutagenicity or its carcenerogenicity

active(A): 0.4 :-
atm(A,B,c,29,C),
gteq(C,-0.003),
ring_size_5(A,D).
active(A): 0.6 :-
lumo(A,B), lteq(B,-2.072).
active(A): 0.3 :-
bond(A,B,C,2),
bond(A,C,D,1),
ring_size_5(A,E).
active(A): 0.7 :-
carbon_6_ring(A,B).
active(A): 0.8 :-
anthracene(A,B).
...
Medicine: diagnose diseases on the basis of patient information (Hepatitis), influence of genes on HIV, risk of falling of elderly people
Inference for probabilistic logic programming under distribution semantics

- Computing the probability of a query
- Knowledge compilation:
  - compile the program to an intermediate representation
  - compute the probability by weighted model counting
- Bayesian Network based:
  - Convert to Bayesian Network
  - Use Bayesian Network inference algorithms (CVE [Meert et al. ILP09])
- Lifted inference
Inference problem is #P hard
For large models inference is intractable
Approximate inference
- Monte Carlo: draw samples of the truth value of the query
- Iterative deepening: gives a lower and an upper bound
- Compute only the best $k$ explanations: branch and bound, gives a lower bound
PITA and MCINTYRE

- PITA performs a program transformation technique and uses techniques alternative to tabling and answer subsumption on knowledge compilation to compute the probability of queries.

- MCINTYRE performs approximate inference by sampling. It uses a different program transformation technique than that of PITA. It is able to handle continuous random variables.
Inference in DISPONTE

- The probability of a query \( Q \) can be computed according to the distribution semantics by first finding the explanations for \( Q \) in the knowledge base.
- Explanation: subset of axioms of the KB that is sufficient for entailing \( Q \).
- All the explanations for \( Q \) must be found, corresponding to all ways of proving \( Q \).
- Application of knowledge compilation to explanations.
Example

\[ E_1 = 0.4 \::\: \text{fluffy} : \text{Cat} \]
\[ E_2 = 0.3 \::\: \text{tom} : \text{Cat} \]
\[ E_3 = 0.6 \::\: \text{Cat} \sqsubseteq \text{Pet} \]
\[ \exists \text{hasAnimal}. \text{Pet} \sqsubseteq \text{NatureLover} \]
\[ (\text{kevin}, \text{fluffy}) : \text{hasAnimal} \]
\[ (\text{kevin}, \text{tom}) : \text{hasAnimal} \]

- \[ Q = \text{kevin} : \text{NatureLover} \] has two explanations:
  \[ \{ (E_1), (E_3) \} \]
  \[ \{ (E_2), (E_3) \} \]

- \[ P(Q) = 0.4 \times 0.6 \times (1 - 0.3) + 0.3 \times 0.6 = 0.348 \]
BUNDLE and TRILL

- BUNDLE performs inference over DISPONTE knowledge bases. It exploits an underlying ontology reasoner able to return all explanations for a query and applies knowledge compilation for computing the probability.

- TRILL resolves the same problem of BUNDLE, but it is completely implemented in Prolog. It is part of a framework containing also TRILL$^P$ and TORNADO, both based on TRILL.
  - Different approaches for collecting the explanations.
Available online at http://trill.ml.unife.it/
Pets example http://trill.ml.unife.it/trill_on_swish/example/peoplePets.owl
Combination of probabilistic logic programming and description logics

- Unified framework exploiting LPAD and DISPONTE Description Logics
- Allows the combination of different closure assumptions
- SPHERE: algorithm combining PITA and TRILL
  - Able to answer queries and computing their probability
Parameter Learning

- Problem: given a set of interpretations, a program, find the parameters maximizing the likelihood of the interpretations (or of instances of a target predicate)
- The interpretations record the truth value of ground atoms, not of the choice variables
- Unseen data: relative frequency can’t be used
Parameter Learning

- An Expectation-Maximization algorithm must be used:
  - Expectation step: the distribution of the unseen variables in each instance is computed given the observed data
  - Maximization step: new parameters are computed from the distributions using relative frequency
  - End when likelihood does not improve anymore
EMBLEM

- EM over Bdds for probabilistic Logic programs Efficient Mining
- Input: an LPAD; logical interpretations (data); *target* predicate(s)
- all ground atoms in the interpretations for the target predicate(s) correspond to as many queries
- use of knowledge compilation to encode the explanations for each query $Q$
EDGE

- Em over bDds for description IoGics paramEter learning
- EDGE is inspired to EMBLEM [Bellodi and Riguzzi, IDA 2013]
- Takes as input a DL theory and a number of examples that represent queries.
- EDGE computes the explanations of each example using BUNDLE

**EDGE^MR** (EDGE powered by MapReduce) parallelizes EDGE:
- The examples are divided in chunks and associated to different processes, since each example is independent of the others
- The learning phase is spread on all the workers as well
## EDGE: Experiments

<table>
<thead>
<tr>
<th></th>
<th>edu.gov.uk</th>
<th>DBPedia</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EDGE</td>
<td>ARs</td>
</tr>
<tr>
<td>AUCPR</td>
<td>0.9702 ± 0.029</td>
<td>0.8804 ± 0.016</td>
</tr>
<tr>
<td>AUCROC</td>
<td>0.9796 ± 0.017</td>
<td>0.9158 ± 0.017</td>
</tr>
</tbody>
</table>
Structure Learning for LPADs

- Given a trivial LPAD or an empty one, a set of interpretations (data)
- *Find the model and the parameters* that maximize the probability of the data (log-likelihood)
- SLIPCOVER: Structure LearnIng of Probabilistic logic program by searching OVER the clause space
- *Parameter learning* by means of EMBLEM [Riguzzi & Bellodi TPLP 2015]
Experiments - Area Under the PR Curve

<table>
<thead>
<tr>
<th>System</th>
<th>HIV</th>
<th>UW-CSE</th>
<th>Mondial</th>
</tr>
</thead>
<tbody>
<tr>
<td>SLIPCOVER</td>
<td>0.82 ± 0.05</td>
<td>0.11 ± 0.08</td>
<td>0.86 ± 0.07</td>
</tr>
<tr>
<td>SLIPCASE</td>
<td>0.78 ± 0.05</td>
<td>0.03 ± 0.01</td>
<td>0.65 ± 0.06</td>
</tr>
<tr>
<td>LSM</td>
<td>0.37 ± 0.03</td>
<td>0.07 ± 0.02</td>
<td>-</td>
</tr>
<tr>
<td>ALEPH++</td>
<td>-</td>
<td>0.05 ± 0.01</td>
<td>0.87 ± 0.07</td>
</tr>
<tr>
<td>RDN-B</td>
<td>0.28 ± 0.06</td>
<td>0.28 ± 0.06</td>
<td>0.77 ± 0.07</td>
</tr>
<tr>
<td>MLN-BT</td>
<td>0.29 ± 0.04</td>
<td>0.18 ± 0.07</td>
<td>0.74 ± 0.10</td>
</tr>
<tr>
<td>MLN-BC</td>
<td>0.51 ± 0.04</td>
<td>0.06 ± 0.01</td>
<td>0.59 ± 0.09</td>
</tr>
<tr>
<td>BUSL</td>
<td>0.38 ± 0.03</td>
<td>0.01 ± 0.01</td>
<td>-</td>
</tr>
<tr>
<td>System</td>
<td>Carcinogenesis</td>
<td>Mutagenesis</td>
<td>Hepatitis</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------</td>
<td>-----------------</td>
<td>----------------</td>
</tr>
<tr>
<td>SLIPCOVER</td>
<td>0.60</td>
<td>0.95 ± 0.01</td>
<td>0.80 ± 0.01</td>
</tr>
<tr>
<td>SLIPCASE</td>
<td>0.63</td>
<td>0.92 ± 0.08</td>
<td>0.71 ± 0.05</td>
</tr>
<tr>
<td>LSM</td>
<td>-</td>
<td>-</td>
<td>0.53 ± 0.04</td>
</tr>
<tr>
<td>ALEPH++</td>
<td>0.74</td>
<td>0.95 ± 0.01</td>
<td>-</td>
</tr>
<tr>
<td>RDN-B</td>
<td>0.55</td>
<td>0.97 ± 0.03</td>
<td>0.88 ± 0.01</td>
</tr>
<tr>
<td>MLN-BT</td>
<td>0.50</td>
<td>0.92 ± 0.09</td>
<td>0.78 ± 0.02</td>
</tr>
<tr>
<td>MLN-BC</td>
<td>0.62</td>
<td>0.69 ± 0.20</td>
<td>0.79 ± 0.02</td>
</tr>
<tr>
<td>BUSL</td>
<td>-</td>
<td>-</td>
<td>0.51 ± 0.03</td>
</tr>
</tbody>
</table>
LEAP

- LEArning Probabilistic description logics
- **Learns the structure** of a probabilistic KB by taking as input a DL KB and a number of positive and negative examples
- learns (acyclic) concept expressions $C_i$ for one or more Target classes
- EDGE is run on the extended theory to compute the log-likelihood of the data $LL$ and the updated parameters

- **LEAP$^{MR}$** (LEAP powered by MapReduce) parallelizes LEAP by exploiting EDGE$^{MR}$
LEAP: Experiments

<table>
<thead>
<tr>
<th>Carcinogenesis</th>
<th>EDGE</th>
<th>LEAP</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUCPR</td>
<td>0.5340 ± 0.108</td>
<td>0.8006 ± 0.240</td>
<td>0.0603</td>
</tr>
<tr>
<td>AUCROC</td>
<td>0.4452 ± 0.051</td>
<td>0.7980 ± 0.246</td>
<td>0.0360</td>
</tr>
</tbody>
</table>
Conclusions

- Logics are useful in domains with complex relationships between entities.
- Combination with probability allows the management of real world domains, with uncertain information.
- PITA, MCINTYRE, EMBLEM, and SLIPCOVER allows reasoning on LPAD.
  - Are all part of the framework cplint.
- BUNDLE, TRILL (and its extensions), EDGE, and LEAP allows reasoning on DISPONTE description logics.
THANKS FOR LISTENING AND ANY QUESTIONS?
References