PROBABILISTIC AND DIFFERENTIABLE PROGRAMMING
V9: Probabilistic Programming II

Özgür L. Özçep
Universität zu Lübeck
Institut für Informationssysteme
Probabilistic Logic Programs (PLP)

• devised by Poole and Sato in the 90s.
• built on top of the programming language Prolog
• upgrade directed graphical models
• Generalises probabilistic databases (Suciu et al.)
• combines the advantages / expressive power of programming languages (Turing equivalent) and graphical models
• Implementations: see next page
PLP Systems

- PRISM [https://www.prismmodelchecker.org/](https://www.prismmodelchecker.org/)
- Yap Prolog [https://github.com/vscosta/yap-6.3](https://github.com/vscosta/yap-6.3) includes
  - ProbLog1
  - cplint
  - CLP(BN)
  - LP2
- AILog2 [http://artint.info/code/ailog/ailog2.html](http://artint.info/code/ailog/ailog2.html)
- SLPs [http://stoics.org.uk/~nicos/sware/pepl](http://stoics.org.uk/~nicos/sware/pepl)
- DC [https://code.google.com/p/distributional-clauses](https://code.google.com/p/distributional-clauses)
Today’s Agenda (in classical linear form)

Probabilistic Logic Programming

1. Modeling
2. Reasoning
3. Learning
MODELING
Motivation (suffering from Vennitis)

Dealing with Uncertainty

Reasoning with Relational data

Various formalisms
Here: PLP

Learning
Motivation (suffering from Vennitia)

**Distribution Semantics** (Sato, 95):
probabilistic choices + logic program
→ distribution over possible worlds

Several possible worlds

0.8::stress(ann).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

Atoms as RVs

Prolog/ logic programming

One world

stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :- influences(Y,X), smokes(Y).

Parameter learning,
Adapted relational
Learning techniques

Various formalisms
Here: PLP

Atoms as RVs

Several possible worlds

Distribution Semantics (Sato, 95):
probabilistic choices + logic program
→ distribution over possible worlds
The motto: Logic everywhere

See also IFIS course Information systems
The Logic programming (LP) paradigm

- The other big three paradigms of programming
  - Imperative (e.g. C)
  - Functional (e.g., Lisp)
  - Object-oriented (e.g. Java)
- Distinguishing feature of LP: Problem solving by specifying the “What” not the “How to”
- Abstracting from
  - Control structures
  - Memory layout
  - Process direction
- Prominent examples: Prolog, Datalog, ASP (Answer set programming)
Science of logic investigates mathematical structures (static and dynamic) and formal languages to describe them by specifying a logic given by

- **syntax** (well-formed formula)
- **semantics** (truth conditions for sentences, entailment notion)
- **calculus** (provability, inference)

Introductory logic textbooks with CS in mind

- (Huth, Ryan 00)
- (Ben-Ari 01)
Where is the logic in logic programming?

- Specification of a domain with a set of formula (sometimes called a knowledge base)
  - Formula specified by truth-condition semantics as in logic
  - In Prolog: formula are facts or rules

- Specification of the problem as a query (also a formula)
  - Query is Boolean or has variables to be bound

- Solving a problem according a logical calculus
  - try to infer (bindings for) query w.r.t. the knowledge base using rules
  - In Prolog use resolution
Prolog

• Prolog: Programmation en Logique

• Invented around 1970 when there was high interest in
  – Theorem proving
  – Language processing with formal grammars

• Protagonists
  – R. Kowalski: Theoretical contribution with SL-Resolution
  – A. Colmerauer and P. Roussel: developer
A bit of gambling with ProbLog

- Toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

• 0.4 :: heads.
• 0.3 :: col(1,red); 0.7 :: col(1, blue).
• 0.2 :: col(2,red); 0.3 :: col(2,green);
  0.5 :: col(2,blue).
  Probabilistic choices

• win :- heads, col(_,red).
• win :- col(1,C), col(2,C).
  Consequences

• Probabilistic fact: Heads with probability 0.4
• annotated disjunction: first ball is red with probability 0.3 and blue with 0.7
• annotated disjunction: second ball is red with probability 0.2, green with 0.3, and blue with 0.5

• Logical rule encoding background knowledge
Queries

0.4 :: heads.
0.3 :: col(1, red); 0.7 :: col(1, blue).
0.2 :: col(2, red); 0.3 :: col(2, green); 0.5 :: col(2, blue).

\[
\begin{align*}
\text{win} & :- \text{heads, col(\_, red)}.
\text{win} & :- \text{col(1,C), col(2,C)}.
\end{align*}
\]

- Probability of \textbf{win}? (marginal probability)

- Probability of \textbf{win} given \textbf{col(2,green)}? (conditional probability)

- Most probable world where \textbf{win} is true? (Most probable explanation (MPE))
Possible Worlds

0.4 :: heads.
0.3 :: col(1, red); 0.7 :: col(1, blue).
0.2 :: col(2, red); 0.3 :: col(2, green); 0.5 :: col(2, blue).

win :- heads, col(_, red).
win :- col(1, C), col(2, C).

0.4 x 0.3 x 0.3
Possible Worlds

0.4 :: heads.
0.3 :: col(1,red); 0.7 :: col(1, blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
All Possible Worlds

(remember the discussion in V8 on traces)
Most likely world with $W$ (\texttt{win} = true)?

<table>
<thead>
<tr>
<th>Probability</th>
<th>World 1</th>
<th>World 2</th>
<th>World 3</th>
<th>World 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.024</td>
<td>H R R W</td>
<td>H R R W</td>
<td>H B R W</td>
<td>B R W</td>
</tr>
<tr>
<td>0.036</td>
<td>H R G W</td>
<td>R G W</td>
<td>H B G W</td>
<td>B G</td>
</tr>
<tr>
<td>0.036</td>
<td>H R B W</td>
<td>R B W</td>
<td>H B B W</td>
<td>B B W</td>
</tr>
<tr>
<td>0.060</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.056</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.054</td>
<td>H B G W</td>
<td>B G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.084</td>
<td>H B G W</td>
<td>B G</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.090</td>
<td>H B B W</td>
<td>B B W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.140</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.126</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.084</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.056</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.210</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MPE inference
\[ P(\text{win} = \text{true}) = P(\text{win}) = \sum_{\text{world with } w=\text{true}} = 0.562 \]
\[ P(\text{win}|\text{col}(2, \text{green})) = \frac{0.036}{0.3} = 0.12 \]

\[ \frac{P(\text{win}, \text{col}(2, \text{green}))}{P(\text{col}(2, \text{green}))} = \frac{\Sigma}{\Sigma} \]
Distribution semantics (Sato, 95)

Distribution semantics with probabilistic facts (Sato 95)

\[
P(Q) = \sum_{F \cup R \models Q} \prod_{f \in F} p(f) \prod_{f \notin F} (1 - p(f))
\]

where

- \( Q \) = query
- \( F \) = subset of facts (assumed to hold in a possible world)
- \( R \) = Prolog rules
- \( F \cup R \models Q \): Summing condition: possible worlds where \( Q \) is true

Probability of possible world
INFERENCE
The challenge: disjoint sum problem

0.4 :: heads(1).
0.7 :: heads(2).
0.5 :: heads(3).

win :- heads(1).
win :- heads(1), heads(3).  \% win \leftrightarrow h(1) \lor (h(2) \land h(3))

- \newcommand{\P}{P}
\newcommand{\win}{\text{win}}
\P(\win) = \P\left(h(1) \lor (h(s) \land h(3))\right) \neq \P(h(1)) + \P(h(2) \land h(3))

- Rather should be

- = \P(h(1)) + \P(h(2) \land h(3)) - \P(h(1) \land h(2) \land h(3))
Idea: Weighted Model Counting (WMC)

- Ground out
- Put formula in CNF (conjunctive normal form)
- Weights
- Call WMC

\[
\begin{align*}
0.4 & \Rightarrow \text{heads}(1). \\
0.7 & \Rightarrow \text{heads}(2). \\
0.5 & \Rightarrow \text{heads}(3). \\
\text{win} & \Leftarrow \text{heads}(1). \\
\text{win} & \Leftarrow \text{heads}(1), \text{heads}(3). \quad \% \text{win} \Leftarrow \text{heads}(1) \lor (\text{heads}(2) \land \text{heads}(3))
\end{align*}
\]

\[
\begin{align*}
\text{weights:} & \quad \neg \text{win} \lor \text{heads}(1) \\
& \land (\neg \text{win} \lor \text{heads}(1) \lor \text{heads}(3)) \\
& \land (\text{win} \lor \neg \text{heads}(1)) \\
& \land (\text{win} \lor \neg \text{heads}(2) \land \neg \text{heads}(3))
\end{align*}
\]

\[
\begin{align*}
\text{weights:} & \quad \text{heads}(1) \Rightarrow 0.4 \quad \neg \text{heads}(1) \Rightarrow 0.6 \\
& \text{heads}(2) \Rightarrow 0.7 \quad \neg \text{heads}(2) \Rightarrow 0.3 \\
& \text{heads}(3) \Rightarrow 0.5 \quad \neg \text{heads}(3) \Rightarrow 0.5
\end{align*}
\]
Recap on some terminology from logic

- A propositional formula is in **conjunctive normal form** (CNF) iff it is a conjunction of disjunctions of literals
- **Literals** = proposition symbol or its negation
- Every propositional formula can be transformed into CNF (using distribution, de Morgan rules and double negation elimination)
Recap on some terminology from logic

For the example note that
- \( A \leftrightarrow B \) and \( (A \rightarrow B) \land (B \rightarrow A) \) are equivalent
- \( A \rightarrow B \) is equivalent to \( \neg A \lor B \)
- Interpretations \( I_i \) (truth value assignments) can also be recorded in set notation (as done in the following)
- E.g. \( I_2 = \{ \neg A, B \} \) or even shorter: \( I_2 = \{ B \} \) (considering only the propositional variables with value 1)

<table>
<thead>
<tr>
<th>( A )</th>
<th>( B )</th>
<th>( A \leftrightarrow B )</th>
<th>( (A \rightarrow B) \land (B \rightarrow A) )</th>
<th>( (\neg A \lor B) \land (\neg B \lor A) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

in CNF
Recap on some terminology from logic

• **Grounding**
  - Idea: “Propositionalize“ rules
  - Technically: Instantiate all variables with all possible constant combinations
  - E.g. `successfulStudent(X):- lovesLogic(X)` over constants `{a,b}`
    - `successfulStudent(a):- lovesLogic(a),`
    - `successfulStudent(b):- lovesLogic(b)`

  - (Grounding not used actually on the slides before, as rules contained no variables)
Weighted Model Counting

\[ WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l) \]

where

- \( \phi \): propositional formula in CNF
  (resulting from problog program or any other statistical relational model (SRL))
- \( I_V \): interpretation of propositional variables
  (in set notation; corresponds to possible world)
- \( w(l) \): weight of literal
  (for \( p : f \) one assigns \( w(f) = p, w(\neg f) = 1 - p \))

For \( \phi = Q \):

\[ WMC(Q) = \sum_{F \cup R \models Q} \prod_{f \in F} p(f) \prod_{f \notin F} 1 - p(f) \]
Weighted Model Counting

- Simple WMC solvers based on a generalisation of DPLL algorithm for SAT (Davis Putnam Logeman Loveland algorithm)
- Current solvers often use knowledge compilation – here an OBDD (ordered binary decision diagram), many variations s-dDNNF, SDDs, (see also following lectures V10-V13)

\[
\text{win} \leftrightarrow h(1) \lor (h(2) \land h(3))
\]
Weighted Model Counting

- Simple WMC solvers based on a generalisation of DPLL algorithm for SAT (Davis Putnam Logeman Loveland algorithm)
- Current solvers often use knowledge compilation – here an OBDD (ordered binary decision diagram), many variations s-dDNNF, SDDs, (see also following lectures V10-V13)

\[
\text{win } \leftrightarrow \ h(1) \lor (h(2) \land h(3))
\]

- \( h(1) \rightarrow 0.4 \quad \neg h(1) \rightarrow 0.6 \)
- \( h(2) \rightarrow 0.7 \quad \neg h(2) \rightarrow 0.3 \)
- \( h(3) \rightarrow 0.5 \quad \neg h(3) \rightarrow 0.5 \)
More inference

• Many variations / extensions

• Approximate inference

• Lifted inference (lifting from propositional to first order)
  – infected(X) :- contact(X,Y), sick(Y).
LEARNING
Parameter Learning: an example

• Webpage classification model
• For each Class1, Class2 and each Word

\[
\text{link\_class(Source,Target, Class1, Class2).}
\]
\[
\text{word\_class(Word,Class).}
\]
\[
\text{class(Page,C) :- has\_word(Page,W), word\_class(W,C).}
\]
\[
\text{class(Page,C) :- links\_to(OtherPage,Page),}
\]
\[
\text{class(OtherPage,OtherClass),}
\]
\[
\text{link\_class(OtherPage,Page,OtherClass,C).}
\]
Sampling interpretations

\[ P(\text{fact}) = \frac{\#(\text{fact true})}{\# \text{interpretations}} \]
Partial interpretations

• Not all facts are observed
  – Note: this is different from some facts Being false

• Use for this some form of the EM-algorithm (Expectation maximization)
  – Expected count used instead of count
  – \(P(Q|E)\) – conditional queries
Reminder: EM: How it Works on Naive Bayes

- Consider the following data,
  - N examples with Boolean attributes $X_1, X_2, X_3, X_4$

- which we want to categorize in one of three possible values of class $C = \{1,2,3\}$ (hidden, no observations given)

- We use a Naive Bayes classifier with hidden variable $C$
Reminder: EM: General Idea

• The algorithm starts from “invented” (e.g., randomly generated) information to solve the learning problem, i.e.
  • Determine the network parameters (CPT in Bayesian networks)

• It then refines this initial guess by cycling through two basic steps
  • **Expectation (E):** update the data with predictions generated via the current model
  • **Maximization (M):** given the updated data, update the model parameters using the Maximum Likelihood (ML) approach

✓ This is the same step that is used for learning parameters for fully observable networks
EM Cycle

**Expected Counts** ("Augmented data")

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</th>
<th>$X_3$</th>
<th>$X_4$</th>
<th>$C$</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
<tr>
<td>$t$</td>
<td>$f$</td>
<td>$t$</td>
<td>$t$</td>
<td>$1$</td>
<td></td>
</tr>
<tr>
<td>$t$</td>
<td>$f$</td>
<td>$t$</td>
<td>$t$</td>
<td>$2$</td>
<td></td>
</tr>
<tr>
<td>$t$</td>
<td>$f$</td>
<td>$t$</td>
<td>$t$</td>
<td>$3$</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
</tr>
</tbody>
</table>

**Probabilities**

- $P(C)$
- $P(X_1|C)$
- $P(X_2|C)$
- $P(X_3|C)$
- $P(X_4|C)$

**Note:** Actually you never generate any data in E-step but expected counts
Learning Rules/Structures

Information Extraction in NELL

instances for many different relations

degree of certainty

NELL:  http://rtw.ml.cmu.edu/rtw/
ProbFOIL

• Upgrade rule-learning to a probabilistic setting within a relational learning / inductive logic programming setting
  – Works with a probabilistic logic program instead of a deterministic one.

• Introduce ProbFOIL, an adaption of Quinlan’s FOIL

• Apply to probabilistic databases like NELL
Example in Pro Log

surfing(X) :- not rain(X), windOK(X). %H
surfing(X) :- not rain(X), sunshine(X).

rain(e1). %B
windOK(e1).
sunshine(e1).

?- surfing(e1). % Query
No % Answer no, because surfing(e1) does not follow from H u B
Example in ProbLog

\[\begin{align*}
p1 &:: \text{surfing}(X) :- \neg \text{rain}(X), \text{windOK}(X). \quad \% \text{H} \\
p2 &:: \text{surfing}(X) :- \neg \text{rain}(X), \text{sunshine}(X).
\end{align*}\]

\[\begin{align*}
0.2 &:: \text{rain}(e1). \quad \% \text{B} \\
0.7 &:: \text{windOK}(e1). \\
0.6 &:: \text{Sunshine}(e1).
\end{align*}\]

?- P(\text{surfing}(e1)). \quad \% \text{Query}
\% \text{gives answer probability } P(\text{B U H } |\text{ e}) = \\
\% (1-0.2) \times 0.7 \times p1 + (1-0.2) \times 0.6 \times (1-0.7) \times p2 \\
\% \text{no rain x windok x p1 + no rain x sunshine x not windOk x p2}
\]

Note: probabilities \(p_1, p_2\) in front of rules are syntactic sugar.
Classical FOIL (Quinlan)

- **Input**
  - Prolog program (or any FOL theory)
  - Observed sequence of facts $E$ (such as `surfing(e1)`)
  - Space of hypotheses $L$

- **Output**: Hypothesis set $H \subseteq L$ (rules) s.t. $B \cup H \models E$

- Hypothesis space contains all admissible rules over the language up to some complexity
- Various heuristics
Inductive Probabilistic Logic Programming

- **Input**
  - a set of example facts $e \in E$ together with the probability $p$ that they hold
  - a background theory $B$ in ProbLog
    (note: $B$ may contain facts and rules, which we know to hold)
  - a hypothesis space $L$ (a set of clauses)

- **Output**

$$\arg\min_H \text{loss}(H, B, E) = \arg\min_H \sum_{e_i \in E} |P_s(B \cup H \models e_i) - p_i|$$

with optimal probabilities for rules.
Next weeks

• More details on the efficient representation of probabilities and formula.
APPENDIX
Probability theory basics reminder

Random variable (RV)

- possible worlds defined by assignment of values to random variables.
- **Boolean** random variables
e.g., Cavity (do I have a cavity?).
  Domain is $\langle$ true, false $\rangle$
- **Discrete** random variables
e.g., possible value of Weather is one of $\langle$ sunny, rainy, cloudy, snow $\rangle$
  - Domain values must be exhaustive and mutually exclusive
  - Elementary propositions are constructed by assignment of a value to a random variable:
    - Cavity = false (abbreviated as $\neg$ cavity)
    - Cavity = true (abbreviated as cavity)
- **(Complex) propositions** formed from elementary propositions and standard logical connectives, e.g., Weather = sunny $\lor$ Cavity = false

Probabilities

- Axioms (for propositions $a, b, \top = (a \lor \neg a)$, and $\bot = \neg \top$):
  - $0 \leq P(a) \leq 1$; $P(\top) = 1$; $P(\bot) = 0$
  - $(P(a \lor b) = P(a) + P(b) - P(a \land b)$
- Joint probability distribution of $X = \{X_1, \ldots, X_n\}$
  - $P(x_1, \ldots, x_n)$
  - gives the probability of every atomic event on $X$
- Conditional probability
  $P(a \mid b) = P(a \land b) / P(b)$ if $P(b) > 0$
- Chain rule
  $P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i \mid x_1, \ldots, x_{i-1})$
- Marginalization: $P(Y) = \sum_{z \in Z} P(Y, z)$
- Conditioning on $Z$:
  - $P(Y) = \sum_{z \in Z} P(Y \mid z)P(z)$ (discrete)
  - $P(Y) = \int P(Y \mid z)P(z)dz$ (continuous)
    $= \mathbb{E}_{z \sim P(z)}P(Y \mid z)$ (expected value notation)
- Bayes’ Rule
  $P(H \mid D) = \frac{P(D \mid H) \cdot P(H)}{P(D)} = \frac{P(D \mid H) \cdot P(H)}{\sum_h P(D \mid h)P(h)$
Color Convention in this Course

- Formulae, when occurring inline
- Newly introduced terminology and definitions
- Important results (observations, theorems) as well as emphasizing some aspects
- Examples are given with standard orange with possibly light orange frame
- Comments and notes in nearly opaque post-it
- Algorithms and program code
- Reminders (in the grey fog of your memory)
Today’s lecture is based on the following

- **Mainly**
    

- **A little bit**
  - Tutorial on
    

References