Bayesian Logic Programs

Summer School on Relational Data Mining
17 and 18 August 2002, Helsinki, Finland

Kristian Kersting, Luc De Raedt
Albert-Ludwigs University

Freiburg, Germany
Real-world applications

uncertainty

probability theory
discrete, continuous

Bayesian networks

complex, structured domains

logic
objects, relations, functors

Logic Programming (Prolog)

Bayesian logic programs
Outline

• Bayesian Logic Programs
  • Examples and Language
  • Semantics and Support Networks
• Learning Bayesian Logic Programs
  • Data Cases
  • Parameter Estimation
  • Structural Learning
Bayesian Logic Programs

• Probabilistic models structured using logic
• Extend Bayesian networks with notions of objects and relations
• Probability density over (countably) infinitely many random variables
• Flexible discrete-time stochastic processes
• Generalize pure Prolog, Bayesian networks, dynamic Bayesian networks, dynamic Bayesian multinets, hidden Markov models,...
Bayesian Networks

• One of the successes of AI
• State-of-the-art to model uncertainty, in particular the degree of belief
• Advantage [Russell, Norvig 96]: "strict separation of qualitative and quantitative aspects of the world"
• Disadvantage [Breese, Ngo, Haddawy, Koller, ...]: Propositional character, no notion of objects and relations among them
Stud farm (Jensen ´96)

- The colt John has been born recently on a stud farm.
- John suffers from a life threatening hereditary carried by a recessive gene. The disease is so serious that John is displaced instantly, and the stud farm wants the gene out of production, his parents are taken out of breeding.
- What are the probabilities for the remaining horses to be carriers of the unwanted gene?
Bayesian networks [Pearl ´88]

Based on the stud farm example [Jensen ´96]
Bayesian networks [Pearl ´88]

Based on the stud farm example [Jensen ´96]

(Conditional) Probability distribution

<table>
<thead>
<tr>
<th>P(bt_john)</th>
<th>bt_henry</th>
<th>bt_irene</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1.0,0.0,0.0)</td>
<td>aa</td>
<td>aa</td>
</tr>
<tr>
<td>(0.5,0.5,0.0)</td>
<td>aa</td>
<td>aA</td>
</tr>
<tr>
<td>(0.0,1.0,0.0)</td>
<td>aa</td>
<td>AA</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0.33,0.33,0.33)</td>
<td>AA</td>
<td>AA</td>
</tr>
</tbody>
</table>

P(bt_cecily=aA|bt_john=aA)=0.1499
P(bt_john=AA|bt_ann=aA)=0.6906
P(bt_john=AA)=0.9909
Bayesian networks (contd.)

- acyclic graphs
- probability distribution over a finite set \( X_1, \ldots, X_n \) of random variables:

\[
P(X_1, \ldots, X_n) = P(X_1 | X_2, \ldots, X_n) \cdot P(X_2 | X_3, \ldots, X_n) \cdot \ldots \cdot P(X_n)
\]

\[
= P(X_1 | Pa(X_1)) \cdot P(X_2 | Pa(X_2)) \cdot \ldots \cdot P(X_n | Pa(X_n))
\]

\[
= \prod_{i=1}^{n} P(X_i | Pa(X_i))
\]
From Bayesian Networks to Bayesian Logic Programs
From Bayesian Networks to Bayesian Logic Programs
From Bayesian Networks to Bayesian Logic Programs
From Bayesian Networks to Bayesian Logic Programs

\[ \text{Pa}(bt\_fred) \]

\[ \begin{align*}
&\text{bt_fred} | \text{bt_unknown1, bt_ann.} \\
&\text{bt_dorothy} | \text{bt_ann, bt_brian.} \\
&\text{bt_eric} | \text{bt_brian, bt_cecily.} \\
&\text{bt_gwenn} | \text{bt_ann, bt_unknown2.} \\
&\text{bt_henry} | \text{bt_fred, bt_dorothy.} \\
&\text{bt_irene} | \text{bt_eric, bt_gwenn.} \\
&\text{bt_john} \\
\end{align*} \]
From Bayesian Networks to Bayesian Logic Programs

\[ P\left( bt\_fred \right) \]

\[ bt\_fred \mid bt\_unknown1, bt\_ann. \]
\[ bt\_dorothy \mid bt\_ann, bt\_brian. \]
\[ bt\_eric \mid bt\_brian, bt\_cecily. \]
\[ bt\_gwenn \mid bt\_ann, bt\_unknown2. \]

\[ bt\_henry \mid bt\_fred, bt\_dorothy. \]
\[ bt\_irene \mid bt\_eric, bt\_gwenn. \]

\[ bt\_john \mid bt\_henry, bt\_irene. \]

<table>
<thead>
<tr>
<th>( P(bt_john) )</th>
<th>bt_henry</th>
<th>bt_irene</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,0,0,0,0)</td>
<td>aa</td>
<td>aa</td>
</tr>
<tr>
<td>(0.5,0,5,0,0)</td>
<td>aa</td>
<td>aA</td>
</tr>
<tr>
<td>(0,0,1,0,0)</td>
<td>aa</td>
<td>AA</td>
</tr>
<tr>
<td>(0,0,0,1,0)</td>
<td>aA</td>
<td>aa</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0.33,0.33,0.33)</td>
<td>AA</td>
<td>AA</td>
</tr>
</tbody>
</table>
From Bayesian Networks to Bayesian Logic Programs

% apriori nodes
bt_ann.  bt_brian.
bt_cecily. bt_unknown1. bt_unknown1.

% aposteriori nodes
bt_henry | bt_fred, bt_dorothy.
bt_irene | bt_eric, bt_gwenn.
bt_fred  | bt_unknown1, bt_ann.
bt_dorothy| bt_brian, bt_ann.
bt_eric   | bt_brian, bt_cecily.
bt_gwenn  | bt_unknown2, bt_ann.
bt_john   | bt_henry, bt_irene.

Domain
e.g. finite, discrete, continuous

(conditional) probability distribution

<table>
<thead>
<tr>
<th>P(bt_john)</th>
<th>bt_henry</th>
<th>bt_irene</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1.0,0.0,0.0)</td>
<td>aa</td>
<td>aa</td>
</tr>
<tr>
<td>(0.5,0.5,0.0)</td>
<td>aa</td>
<td>aA</td>
</tr>
<tr>
<td>(0.0,1.0,0.0)</td>
<td>aa</td>
<td>AA</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(0.33,0.33,0.33)</td>
<td>AA</td>
<td>AA</td>
</tr>
</tbody>
</table>
From Bayesian Networks to Bayesian Logic Programs

% apriori nodes

\[ \text{bt(ann)}, \text{ bt(brian)}, \text{ bt(cecily)}, \text{ bt(unknown1)}, \text{ bt(unknown1)}. \]

% aposteriori nodes

\[ \begin{align*}
\text{bt(henry)} & | \text{ bt(fred), bt(dorothy)}. \\
\text{bt(irene)} & | \text{ bt(eric), bt(gwenn)}. \\
\text{bt(fred)} & | \text{ bt(unknown1), bt(ann)}. \\
\text{bt(dorothy)} & | \text{ bt(brian), bt(ann)}. \\
\text{bt(eric)} & | \text{ bt(brian), bt(cecily)}. \\
\text{bt(gwenn)} & | \text{ bt(unknown2), bt(ann)}. \\
\text{bt(john)} & | \text{ bt(henry), bt(irene)}. \\
\end{align*} \]

<table>
<thead>
<tr>
<th>( P(\text{bt(john)}) )</th>
<th>( \text{bt(henry)} )</th>
<th>( \text{bt(irene)} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>((1,0,0,0,0,0))</td>
<td>aa</td>
<td>aa</td>
</tr>
<tr>
<td>((0,5,0,0,5,0))</td>
<td>aa</td>
<td>aA</td>
</tr>
<tr>
<td>((0,0,1,0,0,0))</td>
<td>aa</td>
<td>AA</td>
</tr>
<tr>
<td>((0,0,1,0,0,0))</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>((0,33,0,33,0,33))</td>
<td>AA</td>
<td>AA</td>
</tr>
</tbody>
</table>

(conditional) probability distribution
From Bayesian Networks to Bayesian Logic Programs

% ground facts / apriori

father(unkown1,fred).  mother(ann,fred).
father(brian,dorothy). mother(ann, dorothy).
father(brian,eric).    mother(cecily,eric).
father(unkown2,gwenn). mother(ann,gwenn).
father(fred,henry).    mother(cecily,gwenn).
father(eric,irene).    mother(gwenn,irene).
father(henry,john).    mother(irene,john).

% rules / aposteriori

bt(X) | father(F,X), bt(F), mother(M,X), bt(M).

(conditional) probability distribution

<table>
<thead>
<tr>
<th>P(bt(X))</th>
<th>father(F,X)</th>
<th>bt(F)</th>
<th>mother(M,X)</th>
<th>bt(M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1.0,0,0,0.0)</td>
<td>true</td>
<td>Aa</td>
<td>true</td>
<td>aa</td>
</tr>
</tbody>
</table>

...
Dependency graph
= Bayesian network

- bt(ann)
- bt(brian)
- bt(cecily)
- bt(dorothy)
- bt(eric)
- bt(unknown2)
- bt(unknown1)
- bt(fred)
- bt(henry)
- bt(john)
- bt(gwenn)

Relationships:
- mother(ann,dorothy)
- father(brian,dorothy)
- father(brian,eric)
- mother(cecily,eric)
- mother(ann,gwenn)
- mother(dorothy,henry)
- mother(eric,irene)
- mother(irene,john)
- father(unknown1,fred)
- mother(ann,fred)
- father(fred,henry)
- father(eric,irene)
- father(henry,john)
- mother(gwenn,irene)
Dependency graph = Bayesian network
Bayesian Logic Programs
- a first definition

A BLP $B$ consists of

- a finite set of Bayesian clauses.
- To each clause $c$ in $B$ a conditional probability distribution $\text{cpd}(c)$ is associated:

\[
\text{cpd}(c) = P(\text{head}(c)|\text{body}(c))
\]

- Proper random variables $\sim LH(B)$
- graphical structure $\sim$ dependency graph
- Quantitative information $\sim$ CPDs
Bayesian Logic Programs - Examples

MC

% apriori nodes
nat(0).

% aposteriori nodes
nat(s(X)) | nat(X).

HMM

% apriori nodes
state(0).

% aposteriori nodes
state(s(Time)) | state(Time).
output(Time) | state(Time)

DBN

% apriori nodes
n1(0).

% aposteriori nodes
n1(s(TimeSlice)) | n2(TimeSlice).
n2(TimeSlice) | n1(TimeSlice).
n3(TimeSlice) | n1(TimeSlice), n2(TimeSlice).
Associated CPDs

- represent generically the CPD for each ground instance of the corresponding Bayesian clause.

<table>
<thead>
<tr>
<th>P(bt(X))</th>
<th>father(F,X)</th>
<th>bt(F)</th>
<th>mother(M,X)</th>
<th>bt(M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1.0,0.0,0.0)</td>
<td>true</td>
<td>Aa</td>
<td>true</td>
<td>aa</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>(0.33,0.33,0.33)</td>
<td>false</td>
<td>AA</td>
<td>false</td>
<td>AA</td>
</tr>
</tbody>
</table>

Multiple ground instances of clauses having the same head atom?
Combining Rules

Multiple ground instances of clauses having the same head atom?

% ground facts as before
% rules
  bt(X) | father(F,X), bt(F).
  bt(X) | mother(M,X), bt(M).

cpd(bt(john)|father(henry,john), bt(henry)) and
cpd(bt(john)|mother(henry,john), bt(irene))

But we need!!

cpd(bt(john)|father(henry,john), bt(henry), mother(irene,john), bt(irene))
Combining Rules (contd.)

- Any algorithm which
  - combines a set of PDFs
    \[
    \left\{ \text{cpd}\left( A \mid A_{i_1}, \ldots, A_{i_n} \right) \mid 1 \leq i \leq m \right\}
    \]
  - into the (combined) PDFs
    \[
    \text{cpd}\left( A \mid B_1, \ldots, B_k \right)
    \]
  - has an empty output if and only if the input is empty
  - E.g. noisy-or, regression, ...

- \( P(A \mid B) \) and \( P(A \mid C) \)
- CR
- \( P(A \mid B, C) \)
Bayesian Logic Programs - a definition

A BLP $B$ consists of

- a finite set of Bayesian clauses.
- To each clause $c$ in $B$ a conditional probability distribution $\text{cpd}(c)$ is associated:
  \[
  \text{cpd}(c) = P\left(\text{head}(c) | \text{body}(c)\right)
  \]
- To each Bayesian predicate $p$ a combining rule $\text{cr}(p)$ is associated to combine CPDs of multiple ground instances of clauses having the same head
- Proper random variables $\sim \text{LH}(B)$
- Graphical structure $\sim$ dependency graph
- Quantitative information $\sim$ CPDs and CRs
Outline

• Bayesian Logic Programs
  • Examples and Language
  • Semantics and Support Networks
• Learning Bayesian Logic Programs
  • Data Cases
  • Parameter Estimation
  • Structural Learning
Discrete-Time Stochastic Process

• Family \( \{ X_t, t \in J \} \) of random variables \( X_t \)
  over a domain \( X \), where \( J \subseteq \{0, 1, 2, \ldots \} \)

• for each linearization of the partial order
  induced by the dependency graph a Bayesian
  logic program specifies a discrete-time
  stochastic process
Theorem of Kolmogorov

Existence and uniqueness of probability measure

• \( X \) : a Polish space
• \( H(J) \) : set of all non-empty, finite subsets of \( J \)
• \( P_I \) : the probability measure over \( X^I, I \in H(J) \)

• If the projective family \( \left( P_I \right)_{I \in H(J)} \) exists then there exists a unique probability measure
Consistency Conditions

- Probability measure \( P_i, I \in H(J) \) is represented by a finite Bayesian network which is a subnetwork of the dependency graph over \( LH(B) \): Support Network

- (Elimination Order): All stochastic processes represented by a Bayesian logic program \( B \) specify the same probability measure over \( LH(B) \).
Support network

• Support network \( N(x) \) of \( x \in LH(B) \) is the induced subnetwork of

\[
S = \{x\} \cup \{y \in LH(B) \mid y \text{ is influencing } x\}
\]

• Support network \( N(x) \) of \( x \subset LH(B) \) is defined as

\[
N(x) = \bigcup_{x \subseteq x} N(x)
\]

• Computation utilizes And/Or trees
Queries using And/Or trees

- A probabilistic query
  \[ ?- \forall_{1, \ldots, n} Q_1 \ldots Q_n \mid E_1 = e_1, \ldots, E_m = e_m. \]
  asks for the distribution
  \[ P(Q_1, \ldots, Q_n \mid E_1 = e_1, \ldots, E_m = e_m). \]

- Or node is proven if at least one of its successors is provable.
- And node is proven if all of its successors are provable.

  \[ ?- \text{bt(eric)}. \]
Consistency Condition (contd.)

Projective family \( \left( P_I \right)_{I \in H(J)} \) exists if

- the dependency graph is acyclic, and
- every random variable is influenced by a finite set of random variables only

well-defined Bayesian logic program
Relational Character

% ground facts
bt(ann).  bt(brian).
bt(cecily).  bt(unknown1).
father(unknown1,fred).  mother(ann,fred).
father(brian,dorothy).  mother(ann, dorothy).
father(brian,eric).  mother(cecily,eric).
father(unknown2,gwenn).  mother(ann,gwenn).
father(fred,henry).  mother(dorothy,henry).
father(eric,irene).  mother(gwenn,irene).
father(henry,john).  mother(irene,john).

% rules
bt(X) | father(F,X), bt(F), mother(M,X), bt(M).

% ground facts
bt(petra).  bt(bsilvester).
bt(anne).  bt(wilhelm).
bt(beate).
father(silvester,claudien).  mother(beate,claudien).
father(wilhelm,marta).  mother(anne, marthe).
...

% ground facts
bt(susanne).  bt(ralf).
bt(peter).  bt(uta).
father(ralf,luca).  mother(susanne,luca).
...

% ground facts
bt(petra).  bt(bsilvester).
bt(anne).  bt(wilhelm).
bt(beate).
father(silvester,claudien).  mother(beate,claudien).
father(wilhelm,marta).  mother(anne, marthe).
...

% rules
bt(X) | father(F,X), bt(F), mother(M,X), bt(M).

P(bt(X))  father(X,F)  bt(F)  mother(X,M)  bt(M)
(1.0,0.0,0.0)  true  Aa  true  aa
...
(0.33,0.33,0.33)  false  AA  false  AA
Bayesian Logic Programs

- Summary

• First order logic extension of Bayesian networks
• constants, relations, functors
• discrete and continuous random variables
• ground atoms = random variables
• CPDs associated to clauses
• Dependency graph = (possibly) infinite Bayesian network
• Generalize dynamic Bayesian networks and definite clause logic (range-restricted)
Applications

- Probabilistic, logical
  - Description and prediction
  - Regression
  - Classification
  - Clustering
- Computational Biology
  - APrIL IST-2001-33053
- Web Mining
- Query approximation
- Planning, ...
Other frameworks

- Probabilistic Horn Abduction [Poole 93]
- Distributional Semantics (PRISM) [Sato 95]
- Stochastic Logic Programs [Muggleton 96; Cussens 99]
- Relational Bayesian Nets [Jaeger 97]
- Probabilistic Logic Programs [Ngo, Haddawy 97]
- Object-Oriented Bayesian Nets [Koller, Pfeffer 97]
  Probabilistic Frame-Based Systems [Koller, Pfeffer 98]
  Probabilistic Relational Models [Koller 99]
Outline

• Bayesian Logic Programs
  • Examples and Language
  • Semantics and Support Networks
  • Learning Bayesian Logic Programs
    • Data Cases
    • Parameter Estimation
    • Structural Learning
Learning Bayesian Logic Programs

Data + Background Knowledge → learning algorithm → $A \mid E, B.$

<table>
<thead>
<tr>
<th>E B</th>
<th>P(A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>e b</td>
<td>.9   .1</td>
</tr>
<tr>
<td>e b</td>
<td>.7   .3</td>
</tr>
<tr>
<td>e b</td>
<td>.8   .2</td>
</tr>
<tr>
<td>e b</td>
<td>.99  .01</td>
</tr>
</tbody>
</table>
Why Learning Bayesian Logic Programs?

Learning within Bayesian network

Inductive Logic Programming

Learning within Bayesian Logic Programs

Of interest to different communities?

• scoring functions, pruning techniques, theoretical insights, ...
What is the data about?

\[
\begin{align*}
&\begin{cases}
m(\text{ann, dorothy}) = \text{true}, f(\text{brian, dorothy}) = \text{true}, \\
p(\text{brian}) = b, bt(\text{ann}) = a, bt(\text{brian}) = ?, bt(\text{dorothy}) = a
\end{cases} \\
&\begin{cases}
m(\text{cecily, fred}) = \text{true}, f(\text{henry, fred}) = \text{true}, bt(\text{cecily}) = ab, \\
bt(\text{henry}) = b, bt(\text{fred}) = ?, m(\text{kim, bob}) = \text{true}, f(\text{fred, bob}) = \text{true}, \\
bt(\text{kim}) = ?, bt(\text{bob}) = b
\end{cases}
\end{align*}
\]

A data case \(D_i \in D\) is a partially observed joint state of a finite, nonempty subset \(x \subset LH(B)\).
Learning Task

Given:
- set $D = \{D_1, \ldots, D_n\}$ of data cases
- a Bayesian logic program $B$

Goal: for each $c \in B$ the parameters

$$\lambda(c) = \{\lambda(c)_1, \ldots, \lambda(c)_{e(c)}\}$$

of $\text{cpd}(c)$ that best fit the given data
Parameter Estimation (contd.)

- „best fit“ ~ ML-Estimation

\[ ?^* = \arg \max_{? \in H} P_B(?) (D) \]

where the hypothesis space \( H \) is spanned by the product space over the possible values of

\[ ? := \bigcup_{c \in B} ? (c) \]
Parameter Estimation (contd.)

\[ ?^* = \arg \max_{? \in H} P_{B(?)} (D) \]

\[ = \arg \max_{? \in H} \ln P_{B(?)} (D) \]

Assumption:

D1,...,DN are independently sampled from indentical distributions (e.g. totally separated families),

\[ = \arg \max_{? \in H} \ln \prod_i P_{B(?)} (D_i) \]

\[ = \arg \max_{? \in H} \sum_i \ln P_{B(?)} (D_i) \]
Parameter Estimation (contd.)

\[ \hat{\theta}^* = \arg \max_{\theta \in \Theta} \sum_i \ln P_{B(\theta)}(D_i) \]

\[ \arg \max_{\theta \in \Theta} \sum_i \ln P_{N(\vartheta)}(D_i) \]

\[ \arg \max_{\theta \in \Theta} \sum_i \ln P_{\mathbb{N}(\vartheta)}(D_i) \]

\[ \hat{N}(\vartheta) = \bigcup_i N_{\vartheta}(\text{var}(D_i)) \]

\( \text{N}(\vartheta) \) is an ordinary BN
Parameter Estimation (contd.)

- Reduced to a problem within Bayesian networks:
  given structure,
  partially observed random variables

- EM
  [Dempster, Laird, Rubin, ´77], [Lauritzen, ´91]

- Gradient Ascent
  [Binder, Koller, Russel, Kanazawa, ´97], [Jensen, ´99]
Decomposable CRs

- Parameters of the clauses and not of the support network.

Single ground instance
of a Bayesian clause

Multiple ground instance
of the same Bayesian clause

CPD for Combining Rule
Gradient Ascent

Goal: Computation of

\[ \frac{\partial \ln P_N(?) (D)}{\partial \lambda_i} \]

\[ \ln P_N(?) (D) \]
Gradient Ascent

\[
\frac{\partial \ln P_{N(?)}(D)}{\partial \text{cpd}(c)_{jk}} = \sum_{\text{subst.}\theta} \sum_{j',k'} \frac{\partial \ln P_{N(?)}(D)}{\partial \text{cpd}(c\theta)_{j'k'}} \times \frac{\partial \text{cpd}(c\theta)_{j'k'}}{\partial \text{cpd}(c)_{jk}}
\]

\[
= \sum_{\text{subst.}\theta} \frac{\partial \ln P_{N(?)}(D)}{\partial \text{cpd}(c\theta)_{jk}}
\]

\[
= \begin{cases} 
1, & \text{if } j = j', k = k' \\
0, & \text{if } j \neq j', k \neq k'
\end{cases}
\]

Bayesian ground clause

Bayesian clause
Gradient Ascent

\[
\frac{\partial \ln P_{N(\theta)}(D)}{\partial \text{cpd}(c)_{jk}} = \sum_{\text{subst. } \theta} \frac{\partial \ln P_{N(\theta)}(D)}{\partial \text{cpd}(c_{\theta})_{jk}}
\]

\[
= \sum_{\text{subst. } \theta} \sum_{i=1}^{n} P_{N(\theta)}(\text{head}(c_{\theta}) = u_j, \text{body}(c_{\theta}) = u_k | D_i) \text{cpd}(c)_{jk}
\]
**Algorithm**

Table 1. A simplified skeleton of the algorithm for adaptive Bayesian logic programs.

```plaintext
function Basic-ABLP(B, D) returns a modified Bayesian logic program

inputs: B, a Bayesian logic program; associated pdfs are parameterized by λ
D, a finite set of data cases

λ ← INITIAL_PARAMETERS
N ← SUPPORTNETWORK(B, D)
repeat until Δλ ≈ 0
    Δλ ← 0
    set pdfs of N according to λ
    for each D_t ∈ D
        set the evidence in N from D_t
        for each clause c ∈ B
            for each ground instance c_θ s.t. \{head(c_θ)\} ∪ body(c_θ) ⊆ N
                for each single parameter λ(c_θ)_t
                    Δλ(c)_t ← Δλ(c)_t + (∂ log P_N(D_t)/∂ λ(c_θ)_t)
                    Δλ ← PROJECTIONONTOCONSTRAINTSURFACE(Δλ)
                λ ← λ + α · Δλ
            return B
```
1. Initialize parameters

2. **E-Step and M-Step**, i.e. compute **expected counts** for each clause and treat the expected count as counts

$$\text{cpd}(c)_{jk} \leftarrow \frac{\sum_{\text{subst. } \theta} \sum_{i=1}^{n} P_{N(?)}(\text{head}(c\theta) = u_j, \text{body}(c\theta) = u_k | D_i)}{\sum_{\text{subst. } \theta} \sum_{i=1}^{n} P_{N(?)}(\text{body}(c\theta) = u_k | D_i)}$$

3. If not converged, iterate to 2
Experimental Evidence

• [Koller, Pfeffer ´97]
  support network is a good approximation

• [Binder et al. ´97]
  equality constraints speed up learning

| m(M,X) | f(F,X) | pdf (c)(h(X)|h(M),h(F)) |
|--------|--------|-------------------------|
| true   | true   | N(0.5*h(M)+0.5*h(F),s) |
| true   | false  | N(165,s)                |
| false  | true   | N(165,s)                |
| false  | false  | N(165,s)                |

• 100 data cases
• constant step-size
• Estimation of means
  • 13 iterations
• Estimation of the weights
  • sum = 1.0
Outline

• Bayesian Logic Programs
  • Examples and Language
  • Semantics and Support Networks
• Learning Bayesian Logic Programs
  • Data Cases
  • Parameter Estimation
• Structural Learning
Structural Learning

- Combination of Inductive Logic Programming and Bayesian network learning
- Datalog fragment of Bayesian logic programs (no functors)
- Intensional Bayesian clauses
<table>
<thead>
<tr>
<th>Idea - CLAUDIEN</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>learning from interpretations</strong></td>
</tr>
<tr>
<td>• all data cases are Herbrand interpretations</td>
</tr>
<tr>
<td>• a hypothesis should reflect what is in the data</td>
</tr>
</tbody>
</table>

---
What is the data about?

\[
\begin{cases}
  m(ann, dorothy) = \text{true}, f(brian, dorothy) = \text{true}, \\
  pc(brian) = b, bt(ann) = a, bt(brian) = ?, bt(dorothy) = a \\
  \vdots
\end{cases}
\]

\[
\begin{cases}
  m(cecily, fred) = \text{true}, f(henry, fred) = \text{true}, \\
  bt(cecily) = ab, bt(henry) = b, bt(fred) = ?, \\
  m(kim, bob) = \text{true}, f(fred, bob) = \text{true}, \\
  bt(kim) = ?, bt(bob) = b
\end{cases}
\]
Claudien -
Learning From Interpretations

- $D$ : set of data cases
- $C$ : set of all clauses that can be part of hypotheses

$H \subseteq C$ (logically) valid iff $\forall D_i \in D : H$ is logically true in $D_i$

$H \subseteq C$ logical solution iff $H$ is a logically maximally general valid hypothesis

$H \subseteq C$ probabilistic solution iff $H$ is (logically) valid and the Bayesian network induced by $B$ on $D$ is acyclic
Learning Task

Given:

• set \( D = \{D_1, \ldots, D_n\} \) of data cases
• a set \( H \) of Bayesian logic programs
• a scoring function \( \text{score}_D : H \to \mathbb{R} \)

Goal: probabilistic solution \( H^* \in H \)

• matches the data best according to \( \text{score}_D \)
Algorithm

Let $H$ be an initial (valid) hypothesis;
$S(H) := \text{score}_D(H)$;
repeat
  $H' := H$;
  $S(H') := S(H)$;
  foreach $H'' \in \rho_{g}(H') \cup \rho_{s}(H')$ do
    if $H''$ is (logically) valid on $D$ then
      if the Bayesian networks induced by $H''$ on the data are acyclic
        then
          if $\text{score}_D(H'') > S(H)$ then
            $H := H''$;
            $S(H) := S(H'')$;
          end
        end
    end
  until $S(H') = S(H)$;
Return $H$;

Algorithm 1: A greedy algorithm for searching the structure of Bayesian logic programs.
Example

Original Bayesian logic program

```
mc(X) | m(M, X), mc(M), pc(M).
p(X) | f(F, X), mc(F), pc(F).
bt(X) | mc(X), pc(X).
```

Data cases

```
{m(ann, john)=true, pc(ann)=a, mc(ann)=?,
f(eric, john)=true, pc(eric)=b, mc(eric)=a,
mc(john)=ab, pc(john)=a, bt(john) = ? }
```

...
Original Bayesian logic program

\[
\begin{align*}
mc(X) & \mid m(M,X), mc(M), pc(M). \\
\text{pc}(X) & \mid f(F,X), mc(F), pc(F). \\
\text{bt}(X) & \mid mc(X), pc(X).
\end{align*}
\]

Initial hypothesis

\[
\begin{align*}
mc(X) & \mid m(M,X). \\
\text{pc}(X) & \mid f(F,X). \\
\text{bt}(X) & \mid mc(X).
\end{align*}
\]
Example

Original Bayesian logic program

\[
\begin{align*}
mc(X) & \mid m(M, X), mc(M), pc(M). \\
pc(X) & \mid f(F, X), mc(F), pc(F). \\
b(t)(X) & \mid mc(X), pc(X).
\end{align*}
\]

Initial hypothesis

\[
\begin{align*}
mc(X) & \mid m(M, X). \\
pc(X) & \mid f(F, X). \\
b(t)(X) & \mid mc(X).
\end{align*}
\]
Example

Original Bayesian logic program

\[
\begin{align*}
mc(X) & \mid m(M,X), mc(M), pc(M). \\
pc(X) & \mid f(F,X), mc(F), pc(F). \\
bt(X) & \mid mc(X), pc(X).
\end{align*}
\]

Initial hypothesis

\[
\begin{align*}
mc(X) & \mid m(M,X). \\
pc(X) & \mid f(F,X). \\
bt(X) & \mid mc(X).
\end{align*}
\]

Refinement

\[
\begin{align*}
mc(X) & \mid m(M,X). \\
pc(X) & \mid f(F,X). \\
bt(X) & \mid mc(X), pc(X).
\end{align*}
\]
Example

Original Bayesian logic program

| mc(X) | m(M,X), mc(M), pc(M). |
| pc(X) | f(F,X), mc(F), pc(F). |
| bt(X) | mc(X), pc(X). |

Initial hypothesis

| mc(X) | m(M,X). |
| pc(X) | f(F,X). |
| bt(X) | mc(X). |

Refinement

| mc(X) | m(M,X). |
| pc(X) | f(F,X). |
| bt(X) | mc(X), pc(X). |

Refinement

| mc(X) | m(M,X), mc(X). |
| pc(X) | f(F,X). |
| bt(X) | mc(X), pc(X). |
Example

Original Bayesian logic program

\[
\begin{align*}
  mc(X) & \mid m(M,X), mc(M), pc(M). \\
  pc(X) & \mid f(F,X), mc(F), pc(F). \\
  bt(X) & \mid mc(X), pc(X).
\end{align*}
\]

Initial hypothesis

\[
\begin{align*}
  mc(X) & \mid m(M,X). \\
  pc(X) & \mid f(F,X). \\
  bt(X) & \mid mc(X).
\end{align*}
\]

Refinement

\[
\begin{align*}
  mc(X) & \mid m(M,X). \\
  pc(X) & \mid f(F,X). \\
  bt(X) & \mid mc(X), pc(X).
\end{align*}
\]
Example

Original Bayesian logic program

\[
\begin{align*}
mc(X) & \mid m(M, X), mc(M), pc(M). \\
p(c)(X) & \mid f(F, X), mc(F), pc(F). \\
bt(X) & \mid mc(X), pc(X). 
\end{align*}
\]

Initial hypothesis

\[
\begin{align*}
mc(X) & \mid m(M, X). \\
p(c)(X) & \mid f(F, X). \\
b(t)(X) & \mid mc(X). 
\end{align*}
\]

Refinement

\[
\begin{align*}
mc(X) & \mid m(M, X). \\
p(c)(X) & \mid f(F, X). \\
b(t)(X) & \mid mc(X), pc(X). 
\end{align*}
\]
Properties

- All relevant random variables are known
- First order equivalent of Bayesian network setting
- Hypothesis postulates true regularities in the data
- Logical solutions as initial hypotheses
- Highlights Background Knowledge
Example Experiments

mc(X) | m(M, X), mc(M), pc(M).
pc(X) | f(F, X), mc(F), pc(F).
bt(X) | mc(X), pc(X).

Data: sampling from 2 families, each 1000 samples
Score: LogLikelihood
Goal: learn the definition of bt
Conclusion

• EM-based and Gradient-based method to do ML parameter estimation

• Link between ILP and learning Bayesian networks

• CLAUDIEN setting used to define and to traverse the search space

• Bayesian network scores used to evaluate hypotheses
Thanks!