MARKOV LOGIC

Overview

- Representation
- Inference
- Learning
- Applications
- Discussion
Propositional Logic

- **Atoms**: Symbols representing propositions
- **Logical connectives**: $\neg$, $\land$, $\lor$, etc.
- **Knowledge base**: Set of formulas
- **World**: Truth assignment to all atoms
- Every KB can be converted to **CNF**
  - **CNF**: Conjunction of clauses
  - **Clause**: Disjunction of literals
  - **Literal**: Atom or its negation
- **Entailment**: Does KB entail query?

First-Order Logic

- **Atom**: Predicate(Variables,Constants)
  
  E.g.: $\text{Friends}(\text{Anna},x)$
- **Ground atom**: All arguments are constants
- **Quantifiers**: $\forall$, $\exists$
- **This talk**: Finite, Herbrand interpretations
Markov Networks

- Undirected graphical models

- Potential functions defined over cliques

\[ P(x) = \frac{1}{Z} \prod_c \Phi_c(x_c) \]

\[ Z = \sum_x \prod_c \Phi_c(x_c) \]

<table>
<thead>
<tr>
<th>Smoking</th>
<th>Cancer</th>
<th>( \Phi(S,C) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>4.5</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>4.5</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>2.7</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>4.5</td>
</tr>
</tbody>
</table>

Markov Networks

- Undirected graphical models

- Log-linear model:

\[ P(x) = \frac{1}{Z} \exp \left( \sum_i w_i f_i(x) \right) \]

\[ f_i(\text{Smoking},\text{Cancer}) = \begin{cases} 
1 & \text{if } \neg \text{Smoking} \lor \text{Cancer} \\
0 & \text{otherwise}
\end{cases} \]

\[ w_i = 0.51 \]
**Probabilistic Knowledge Bases**

$\text{PKB} = \text{Set of formulas and their probabilities} + \text{Consistency} + \text{Maximum entropy}
= \text{Set of formulas and their weights}
= \text{Set of formulas and their potentials}
(1 \text{ if formula true, } \phi_i \text{ if formula false})

\[
P(\text{world}) = \frac{1}{Z} \prod_i \phi_i^{n_i(\text{world})}
\]

**Markov Logic**

- A *Markov Logic Network (MLN)* is a set of pairs $(F, w)$ where
  - $F$ is a formula in first-order logic
  - $w$ is a real number
- An MLN defines a Markov network with
  - One node for each grounding of each predicate in the MLN
  - One feature for each grounding of each formula $F$ in the MLN, with the corresponding weight $w$
Example

\[ \neg \text{Friends}(\text{Anna, Bob}) \]

\[ \text{Friends}(\text{Anna, Bob}) \]

\[ \neg \text{Happy}(\text{Bob}) \quad \text{Happy}(\text{Bob}) \]

Example

\[ \neg \text{Friends}(\text{Anna, Bob}) \]

\[ \neg \text{Friends}(\text{Anna, Bob}) \vee \text{Happy}(\text{Bob}) \]

\[ \text{Friends}(\text{Anna, Bob}) \]

\[ \neg \text{Happy}(\text{Bob}) \quad \text{Happy}(\text{Bob}) \]
Example

\[ P(\neg\text{Friends}(\text{Anna, Bob}) \lor \text{Happy}(\text{Bob})) = 0.8 \]

Example

\[ \Phi(\neg\text{Friends}(\text{Anna, Bob}) \lor \text{Happy}(\text{Bob})) = 1 \]
\[ \Phi(\text{Friends}(\text{Anna, Bob}) \land \neg\text{Happy}(\text{Bob})) = 0.75 \]
### Example

\[ w(\Phi(\neg\text{Friends}(\text{Anna, Bob}) \lor \text{Happy}(\text{Bob}))) \]

\[ = \log(1/0.75) = 0.29 \]

<table>
<thead>
<tr>
<th>¬Friends(Anna, Bob)</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Friends(Anna, Bob)</td>
<td>0.75</td>
</tr>
<tr>
<td>¬Happy(Bob)</td>
<td></td>
</tr>
<tr>
<td>Happy(Bob)</td>
<td>1</td>
</tr>
</tbody>
</table>

### Overview

- Representation
- Inference
- Learning
- Applications
- Discussion
Theorem Proving

\[ TP(KB, \text{Query}) \]
\[ KB_Q \leftarrow KB \cup \{\neg \text{Query}\} \]
\[ \text{return } \neg \text{SAT(CNF}(KB_Q)) \]

Satisfiability (DPLL)

\[ \text{SAT}(\text{CNF}) \]
\hspace{1em} \text{if } \text{CNF is empty return True} \]
\hspace{1em} \text{if } \text{CNF contains empty clause return False} \]
\hspace{1em} \text{choose an atom } A \]
\hspace{1em} \text{return SAT(CNF}(A)) \lor \text{SAT(CNF}(\neg A)) \]
First-Order Theorem Proving

- **Propositionalization**
  1. Form all possible ground atoms
  2. Apply propositional theorem prover

- **Lifted Inference: Resolution**
  - Resolve pairs of clauses until empty clause derived
  - Unify literals by substitution, e.g.: $x=Bob$ unifies $\text{Friends}(\text{Anna}, x)$ and $\text{Friends}(\text{Anna}, \text{Bob})$

\[
\begin{align*}
\neg \text{Friends}(\text{Anna}, x) & \vee \text{Happy}(x) \\
\text{Friends}(\text{Anna}, \text{Bob}) & \\
\hline
\text{Happy}(\text{Bob})
\end{align*}
\]

---

Probabilistic Theorem Proving

**Given** Probabilistic knowledge base $K$

**Query formula** $Q$

**Output** $P(Q|K)$
**Weighted Model Counting**

- ModelCount(CNF) = # worlds that satisfy CNF
- Assign a weight to each literal
- Weight(world) = \( \prod \) weights(true literals)
- Weighted model counting:
- **Given** CNF \( C \) and literal weights \( W \)
- **Output** \( \Sigma \) weights(worlds that satisfy \( C \))

PTP is reducible to lifted WMC

---

**Example**

\( \text{Friends}(Anna, Bob) \)

\[
\begin{array}{c|c|c}
\text{¬Friends}(Anna, Bob) & \text{Friends}(Anna, Bob) & \text{¬Happy}(Bob) & \text{Happy}(Bob) \\
\hline
0.75 & 1 & & \\
\end{array}
\]
Example

\[ P(\text{Happy}(\text{Bob}) \mid \text{Friends}(\text{Anna, Bob})) = \frac{1}{1 + 0.75} = 0.57 \]

Example

If \( P(\neg \text{Friends}(\text{Anna, Bob}) \lor \text{Happy}(\text{Bob})) = 0.8 \)
Then \( P(\text{Happy}(\text{Bob}) \mid \text{Friends}(\text{Anna, Bob})) = \frac{1}{1 + 0.75} \approx 0.5 \)
Example

\[ P(\neg \text{Friends(Anna, } x) \lor \text{Happy}(x)) = 0.8 \]

Example

\[ P(\neg \text{Friends(Anna, } x) \lor \text{Happy}(x)) = 0.8 \]

\[ \text{Friends(Anna, } x) \]

\[ \neg \text{Friends(Anna, } x) \]

\[ \lor \text{Happy}(x) \]

\[ \neg \text{Happy}(x) \quad \text{Happy}(x) \]

\[ \text{Bob} \quad \neg \text{Bob} \]

\[ \text{Friends(Anna, Bob)} \]

\[ \text{0.75} \quad 1 \]

\[ \neg \text{Happy}(\text{Bob}) \quad \text{Happy}(\text{Bob}) \]
Propositional Case

- All conditional probabilities are ratios of partition functions:

\[ P(\text{Query} \mid PKB) = \frac{\sum_{\text{worlds}} 1_{\text{Query}}(\text{world}) \prod_{i} \Phi_{i}(\text{world})}{Z(PKB)} \]

\[ = \frac{Z(PKB \cup \{(\text{Query},0)\})}{Z(PKB)} \]

- All partition functions can be computed by weighted model counting
Conversion to CNF + Weights

\[
\text{WCNF}(PKB) \\
\text{for all } (F_i, \Phi_i) \in PKB \text{ s.t. } \Phi_i > 0 \text{ do} \\
\quad PKB \leftarrow PKB \cup \{(F_i \iff A_i, 0)\} \setminus \{(F_i, \Phi_i)\} \\
\quad CNF \leftarrow \text{CNF}(PKB) \\
\text{for all } \neg A_i \text{ literals do } W_{\neg A_i} \leftarrow \Phi_i \\
\text{for all other literals } L \text{ do } w_L \leftarrow 1 \\
\text{return } (CNF, \text{weights})
\]

Probabilistic Theorem Proving

\[
\text{PTP}(PKB, \text{Query}) \\
PKB_Q \leftarrow PKB \cup \{(\text{Query}, 0)\} \\
\text{return } \text{WMC}(\text{WCNF}(PKB_Q)) \\
/ \text{WMC}(\text{WCNF}(PKB))
\]
Probabilistic Theorem Proving

\[ \text{PTP}(\text{PKB}, \text{Query}) \]
\[ \text{PKB}_Q \leftarrow \text{PKB} \cup \{(\text{Query}, 0)\} \]
\[ \text{return } \frac{\text{WMC}(\text{WCNF}(\text{PKB}_Q))}{\text{WMC}(\text{WCNF}(\text{PKB}))} \]

Compare:

\[ \text{TP}(\text{KB}, \text{Query}) \]
\[ \text{KB}_Q \leftarrow \text{KB} \cup \{\neg \text{Query}\} \]
\[ \text{return } \neg \text{SAT}(\text{CNF}(\text{KB}_Q)) \]

Weighted Model Counting

\[ \text{WMC}(\text{CNF}, \text{weights}) \]
\[ \text{if } \text{all clauses in } \text{CNF} \text{ are satisfied} \]
\[ \text{return } \prod_{A \in A(\text{CNF})} (w_A + w_{\neg A}) \]
\[ \text{if } \text{CNF} \text{ has empty unsatisfied clause } \text{return } 0 \]
Weighted Model Counting

\[ \text{WMC}(CNF, \text{weights}) \]

if all clauses in \( CNF \) are satisfied
return \( \prod_{A \in A(CNF)} (w_A + w_{\neg A}) \)
if \( CNF \) has empty unsatisfied clause return 0
if \( CNF \) can be partitioned into \( CNFs \) \( C_1, \ldots, C_k \)
sharing no atoms
return \( \prod_{i=1}^{k} \text{WMC}(C_i, \text{weights}) \)

Decomp. Step

| 31 |

Pedro Domingos, Kristian Kersting
Combining Logic and Probability: Languages, Algorithms and Applications

-------------------------------------

Weighted Model Counting

\[ \text{WMC}(CNF, \text{weights}) \]

if all clauses in \( CNF \) are satisfied
return \( \prod_{A \in A(CNF)} (w_A + w_{\neg A}) \)
if \( CNF \) has empty unsatisfied clause return 0
if \( CNF \) can be partitioned into \( CNFs \) \( C_1, \ldots, C_k \)
sharing no atoms
return \( \prod_{i=1}^{k} \text{WMC}(C_i, \text{weights}) \)
choose an atom \( A \)
return \( w_A \text{WMC}(CNF \mid A, \text{weights}) + w_{\neg A} \text{WMC}(CNF \mid \neg A, \text{weights}) \)

Splitting Step

| 32 |

Pedro Domingos, Kristian Kersting
Combining Logic and Probability: Languages, Algorithms and Applications
First-Order Case

- PTP schema remains the same
- Conversion of PKB to hard CNF and weights:
  New atom in $F_i \Leftrightarrow A_i$ is now $\text{Predicate}_i(\text{variables in } F_i, \text{constants in } F_i)$
- New argument in WMC:
  Set of substitution constraints of the form $x = A, x \neq A, x = y, x \neq y$
- Lift each step of WMC

Lifted Weighted Model Counting

$\text{LWMC}(\text{CNF}, \text{substs}, \text{weights})$

if all clauses in $\text{CNF}$ are satisfied

\[ \text{return } \prod_{A \in A(\text{CNF})} (w_A + w_{\neg A})^{n_A(\text{substs})} \]

if $\text{CNF}$ has empty unsatisfied clause

\[ \text{return } 0 \]
Lifted Weighted Model Counting

\[ \text{LWMC} (CNF, \text{substs}, \text{weights}) \]

- if all clauses in \( CNF \) are satisfied
  
  \[ \text{return } \prod_{A \in A(CNF)} (w_A + w_{\neg A})^{n_A(\text{substs})} \]

- if \( CNF \) has empty unsatisfied clause return \( 0 \)

- if there exists a lifted decomposition of \( CNF \)
  
  \[ \text{return } \prod_{i=1}^{k} [\text{LWMC}(CNF_{i,1}, \text{substs}, \text{weights})]^m_i \]

- choose an atom \( A \)
  
  \[ \text{return } \sum_{i=1}^{l} n_i w_A^{f_i} w_{\neg A}^{f_i} \text{LWMC}(CNF \mid \sigma_j, \text{substs}_j, \text{weights}) \]
Extensions

- Unit propagation, etc.
- Caching / Memoization
- Knowledge-based model construction

Approximate Inference

\[ WMC(CNF, weights) \]

if all clauses in CNF are satisfied

\[ \text{return } \prod_{A \in \text{Atoms}(CNF)} (w_A + w_{\neg A}) \]

if CNF has empty unsatisfied clause return 0

if CNF can be partitioned into CNFs \( C_1, \ldots, C_k \) sharing no atoms

\[ \text{return } \prod_{i=1}^k WMC(C_i, weights) \]

choose an atom \( A \)

\[ \text{return } \frac{w_A}{Q(A | CNF, weights)} WMC(CNF | A, weights) \] with probability \( Q(A | CNF, weights) \), etc.
MPE Inference

- Replace sums by maxes
- Use branch-and-bound for efficiency
- Do traceback

Overview

- Representation
- Inference
- Learning
- Applications
- Discussion
Learning

- Data is a relational database
- Closed world assumption (if not: EM)
- Learning parameters (weights)
  - Generatively
  - Discriminatively
- Learning structure (formulas)

Generative Weight Learning

- Maximize likelihood
- Use gradient ascent or L-BFGS
- No local maxima
  \[
  \frac{\partial}{\partial w_i} \log P_w(x) = n_i(x) - E_w[n_i(x)]
  \]
  - \(n_i(x)\): No. of true groundings of clause \(i\) in data
  - \(E_w[n_i(x)]\): Expected no. true groundings according to model
- Requires inference at each step (slow!)
Pseudo-Likelihood

$$PL(x) \equiv \prod_{i} P(x_i \mid \text{neighbors}(x_i))$$

- Likelihood of each variable given its neighbors in the data [Besag, 1975]
- Does not require inference at each step
- Consistent estimator
- Widely used in vision, spatial statistics, etc.
- But PL parameters may not work well for long inference chains

Discriminative Weight Learning

- Maximize conditional likelihood of query ($y$) given evidence ($x$)
  $$\frac{\partial}{\partial w_i} \log P_w(y \mid x) = n_i(x, y) - E_w[n_i(x, y)]$$
  - No. of true groundings of clause $i$ in data
  - Expected no. true groundings according to model
- Expected counts can be approximated by counts in MAP state of $y$ given $x$
Voted Perceptron

- Originally proposed for training HMMs discriminatively [Collins, 2002]
- Assumes network is linear chain

\[
\begin{align*}
    w_i &\leftarrow 0 \\
    \text{for } t &\leftarrow 1 \text{ to } T \text{ do} \\
    y_{MAP} &\leftarrow \text{Viterbi}(x) \\
    w_i &\leftarrow w_i + \eta \left[ \text{count}_{i}(y_{Data}) - \text{count}_{i}(y_{MAP}) \right] \\
    \text{return } \sum_t w_i / T
\end{align*}
\]

Voted Perceptron for MLNs

- HMMs are special case of MLNs
- Replace Viterbi by prob. theorem proving
- Network can now be arbitrary graph

\[
\begin{align*}
    w_i &\leftarrow 0 \\
    \text{for } t &\leftarrow 1 \text{ to } T \text{ do} \\
    y_{MAP} &\leftarrow \text{PTP}(\text{MLN } U \{x\}, y) \\
    w_i &\leftarrow w_i + \eta \left[ \text{count}_{i}(y_{Data}) - \text{count}_{i}(y_{MAP}) \right] \\
    \text{return } \sum_t w_i / T
\end{align*}
\]
Structure Learning

- Generalizes feature induction in Markov nets
- Any inductive logic programming approach can be used, but . . .
- Goal is to induce any clauses, not just Horn
- Evaluation function should be likelihood
- Requires learning weights for each candidate
- Turns out not to be bottleneck
- Bottleneck is counting clause groundings
- Solution: Subsampling

Structure Learning

- **Initial state:** Unit clauses or hand-coded KB
- **Operators:** Add/remove literal, flip sign
- **Evaluation function:** Pseudo-likelihood + Structure prior
- **Search:**
  - Beam, shortest-first [Kok & Domingos, 2005]
  - Bottom-up [Mihalkova & Mooney, 2007]
  - Relational pathfinding [Kok & Domingos, 2009, 2010]
Alchemy

Open-source software including:

- Full first-order logic syntax
- MAP and marginal/conditional inference
- Generative & discriminative weight learning
- Structure learning
- Programming language features

alchemy.cs.washington.edu

<table>
<thead>
<tr>
<th>Represent-ation</th>
<th>Alchemy</th>
<th>Prolog</th>
<th>BUGS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>F.O. Logic + Markov nets</td>
<td>Horn clauses</td>
<td>Bayes nets</td>
</tr>
<tr>
<td>Inference</td>
<td>Probabilistic thm. proving</td>
<td>Theorem proving</td>
<td>Gibbs sampling</td>
</tr>
<tr>
<td>Learning</td>
<td>Parameters &amp; structure</td>
<td>No</td>
<td>Params.</td>
</tr>
<tr>
<td>Uncertainty</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Relational</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
Overview

- Representation
- Inference
- Learning
- Applications
- Discussion

Applications to Date

- Natural language processing
- Information extraction
- Entity resolution
- Link prediction
- Collective classification
- Social network analysis
- Robot mapping
- Activity recognition
- Scene analysis
- Computational biology
- Probabilistic Cyc
- Personal assistants
- Etc.
Information Extraction

Parag Singla and Pedro Domingos, “Memory-Efficient Inference in Relational Domains” (AAAI-06).


Segmentation

Parag Singla and Pedro Domingos, “Memory-Efficient Inference in Relational Domains” (AAAI-06).


Entity Resolution

Parag Singla and Pedro Domingos, “Memory-Efficient Inference in Relational Domains” (AAAI-06).


State of the Art

- Segmentation
  - HMM (or CRF) to assign each token to a field
- Entity resolution
  - Logistic regression to predict same field/citation
  - Transitive closure
- Alchemy implementation: Seven formulas

Types and Predicates

\[
\text{token} = \{\text{Parag, Singla, and, Pedro, ...}\} \\
\text{field} = \{\text{Author, Title, Venue}\} \\
\text{citation} = \{C1, C2, ...\} \\
\text{position} = \{0, 1, 2, ...\}
\]

Token(token, position, citation) \\
InField(position, field, citation) \\
SameField(field, citation, citation) \\
SameCit(citation, citation)
Types and Predicates

- \( \text{token} = \{ \text{Parag}, \text{Singla}, \text{and}, \text{Pedro}, \ldots \} \)
- \( \text{field} = \{ \text{Author}, \text{Title}, \text{Venue}, \ldots \} \)
- \( \text{citation} = \{ \text{C1}, \text{C2}, \ldots \} \)
- \( \text{position} = \{ 0, 1, 2, \ldots \} \)

- \( \text{Token} (\text{token}, \text{position}, \text{citation}) \)
- \( \text{InField} (\text{position}, \text{field}, \text{citation}) \)
- \( \text{SameField} (\text{field}, \text{citation}, \text{citation}) \)
- \( \text{SameCit} (\text{citation}, \text{citation}) \)
Types and Predicates

token = {Parag, Singla, and, Pedro, ...}
field = {Author, Title, Venue}
citation = {C1, C2, ...}
position = {0, 1, 2, ...}

Token(token, position, citation)

InField(position, field, citation)

SameField(field, citation, citation)

SameCit(citation, citation)

InField(position, position, citation)

Query

Formulas

Token(+t, i, c) => InField(i, +f, c)
InField(i, +f, c) <=> InField(i+1, +f, c)
f != f’ => (!InField(i, +f, c) v !InField(i, +f’, c))

Token(+t, i, c) ^ InField(i, +f, c) ^ Token(+t, i’, c’) ^ InField(i’, +f, c’) => SameField(+f, c, c’)
SameField(+f, c, c’) <= SameCit(c, c’)
SameField(f, c, c’) ^ SameField(f, c’, c”) => SameField(f, c, c”)
SameCit(c, c’) ^ SameCit(c’, c”) => SameCit(c, c”)

Pedro Domingos, Kristian Kersting
Combining Logic and Probability: Languages, Algorithms and Applications
Formulas

\[
\text{Token}(+t, i, c) \Rightarrow \text{InField}(i, +f, c)
\]
\[
\text{InField}(i, +f, c) \iff \text{InField}(i+1, +f, c)
\]
\[
f \neq f' \Rightarrow (!\text{InField}(i, +f, c) \lor !\text{InField}(i, +f', c))
\]

\[
\text{Token}(+t, i, c) \land \text{InField}(i, +f, c) \land \text{Token}(+t, i', c') \land \text{InField}(i', +f, c') \Rightarrow \text{SameField}(+f, c, c')
\]
\[
\text{SameField}(+f, c, c') \iff \text{SameCit}(c, c')
\]
\[
\text{SameField}(f, c, c') \land \text{SameField}(f, c', c'') \Rightarrow \text{SameField}(f, c, c'')
\]
\[
\text{SameCit}(c, c') \land \text{SameCit}(c', c'') \Rightarrow \text{SameCit}(c, c'')
\]
Formulas

Token(+t,i,c) => InField(i,+f,c)
InField(i,+f,c) <=> InField(i+1,+f,c)
f != f' => (!InField(i,+f,c) v !InField(i,+f',c))

Token(+t,i,c) ^ InField(i,+f,c) ^ Token(+t,i',c')
^ InField(i',+f,c') => SameField(+f,c,c')
SameField(+f,c,c') <=> SameCit(c,c')
SameField(f,c,c') ^ SameField(f,c',c'')
=> SameField(f,c,c'')
SameCit(c,c') ^ SameCit(c',c'') => SameCit(c,c'')
Formulas

Token(+t,i,c) => InField(i,+f,c)
InField(i,+f,c) <=> InField(i+1,+f,c)
f != f' => (!InField(i,+f,c) v !InField(i,+f',c))

Token(+t,i,c) ^ InField(i,+f,c) ^ Token(+t,i',c')
^ InField(i',+f,c') => SameField(+f,c,c')

SameField(+f,c,c') <=> SameCit(c,c')

SameField(f,c,c') ^ SameField(f,c',c'')
=> SameField(f,c,c'')

SameCit(c,c') ^ SameCit(c',c'') => SameCit(c,c'')
Formulas

\[ \text{Token}(t, i, c) \Rightarrow \text{InField}(i, +f, c) \]

\[ \text{InField}(i, +f, c) \Rightarrow \text{Token}(\text{"."}, i, c) \Rightarrow \text{InField}(i+1, +f, c) \]

\[ f \neq f' \Rightarrow (\neg \text{InField}(i, +f, c) \lor \neg \text{InField}(i, +f', c)) \]

\[ \text{Token}(t, i, c) \land \text{InField}(i, +f, c) \land \text{Token}(t, i', c') \]
\[ \land \text{InField}(i', +f, c') \Rightarrow \text{SameField}(+f, c, c') \]

\[ \text{SameField}(+f, c, c') \Leftrightarrow \text{SameCit}(c, c') \]

\[ \text{SameField}(+f, c, c') \Rightarrow \text{SameField}(+f, c', c'') \]

\[ \text{SameCit}(c, c') \Rightarrow \text{SameCit}(c', c'') \Rightarrow \text{SameCit}(c, c'') \]

Results: Segmentation on Cora

![Graph showing recall and precision for different segmentation methods.](image)
Results: Matching Venues on Cora

Overview

- Representation
- Inference
- Learning
- Applications
- Discussion
Foundations for Probabilistic Models

- Graphs are not enough
- We need logic

Logical Models vs. Graphical Models (I)

<table>
<thead>
<tr>
<th></th>
<th>Graphical models</th>
<th>Logical models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Required by probability theory</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Representable distributions</td>
<td>All (BNs)</td>
<td>All</td>
</tr>
<tr>
<td></td>
<td>Positive (MNs)</td>
<td></td>
</tr>
<tr>
<td>Context-free independences</td>
<td>Some</td>
<td>All</td>
</tr>
<tr>
<td>Context-specific independences</td>
<td>None</td>
<td>All</td>
</tr>
<tr>
<td>Normalization constraints</td>
<td>Some</td>
<td>All</td>
</tr>
</tbody>
</table>
## Logical Models vs. Graphical Models (II)

<table>
<thead>
<tr>
<th></th>
<th>Graphical models</th>
<th>Logical models</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inference</td>
<td>Exp(treewidth)</td>
<td>Circuit complexity</td>
</tr>
<tr>
<td>Visual aid</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Densely connected distrs.</td>
<td>Unreadable</td>
<td>Readable</td>
</tr>
<tr>
<td>First-order</td>
<td>Plates</td>
<td>All</td>
</tr>
<tr>
<td>Lifted inference</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Available technology</td>
<td>Lots, used</td>
<td>Lots, unused</td>
</tr>
</tbody>
</table>

Pedro Domingos, Kristian Kersting
Combining Logic and Probability: Languages, Algorithms and Applications