CSE 573 Knowledge Representation: Propositional, FO & Markov Logic

Dan Weld
(With some slides from Mausam, Stuart Russell, Dieter Fox, Henry Kautz, Pedro Domingos, Min-Yen Kan...)

Irrationally held truths may be more harmful than reasoned errors.
- Thomas Huxley (1825-1895)

Project Presentations

- Friday 12/7
- Length = 4, 6, 7 or 8 min (includes questions) – practice!
- Default = your laptop; else mail me slides (.ppt or .pdf) by 9am Fri
  - Bring slides on a backup USB memory.
- Every team member should talk for some part of the presentation
- Subtopics to cover:
  - Aspirations & reality of what you built
  - Demo?
  - Suprises (What was harder or easier than expected?)
  - What did you learn?
  - Experiments & validation
  - Plans for remaining week
  - Who did what

Final Reports (see web page)

- Goals for the project
- System design and algorithmic choices
- Sample screens of typical usage scenarios (if applicable)
- Experiments and results
- Anything you considered surprising or that you learned.
  - What would you do differently if you could?
- Conclusions and ideas for future work
- Appendices
- No limit on length, but we appreciate good organization and tight, precise writing. Points off for rambling and repetition.

Experiments

- Clearly state question being asked
- Kinds of experiments
  - Informal user study
  - Formal user study
  - System (or module) performance comparison
    - Baselines
    - Ablation experiments

Presenting Results

Graphs vs tables
Chartjunk
Data / ink ratio
Visualization integrity

Previously

- CSPs are a special (factored) kind of search problem:
  - States defined by values (domains) of a fixed set of variables
  - Goal test defined by constraints on variable values
- Backtracking = DFS - one legal variable assigned per node
- Heuristics
  - Variable ordering: min remaining values
  - Value ordering: least constraining value
Previously

- CSPs are a special (factored) kind of search problem:
  - States defined by values (domains) of a fixed set of variables
  - Goal test defined by constraints on variable values
- Backtracking = DFS - one legal variable assigned per node
- Variable ordering and value selection heuristics help
- Forward checking prevents assignments that fail later

\[ \begin{align*}
Q_1 & \quad Q_2 \\
\text{Row 1} & \quad \text{Row 2} \\
\text{Row 3} & \quad \text{Row 4}
\end{align*} \]

### Forward checking prevents assignments that fail later

- Constraint propagation (e.g., arc consistency)
  - does additional work to constrain values and detect inconsistencies
- Constraint graph representation
  - Allows analysis of problem structure
- Tree-structured CSPs can be solved in linear time
- Local (stochastic) search often effective in practice
  - Iterative min-conflicts

### Algorithms

- Blind search
- Heuristic search
- Mini-max & Expectimax
- MDPs & POMDPs
- Reinforcement learning
- State estimation

### Knowledge Representation

- Separate knowledge from algorithms
- HMMs
- Bayesian networks
- Propositional logic
- First-order logic
- Description logic
- Constraint networks
- Markov logic networks
- ...

### Overview

- Knowledge Representation & Reasoning
- Propositional Logic
  - Foundations: Syntax, semantics & inference
  - Algorithms: DPLL, Resolution, WalkSAT
  - Tractable subsets
- First-Order Logic
- Markov Logic
Semantics
- **Syntax**: which arrangements of symbols are *legal*  
  - (Def "sentences")
- **Semantics**: what the symbols *mean* in the world  
  - (Mapping between symbols and worlds)

Models
- Logicians often think in terms of models, which are formally structured worlds with respect to which truth can be evaluated  
  - In propositional case, each model = truth assignment  
  - Set of models can be enumerated in a truth table
- We say m is a model of a sentence α if α is true in m
- M(α) is the set of all models of α
- Then KB |= α iff M(KB) ⊆ M(α)
  - E.g., KB = (P ∨ Q) ∧ (¬P ∨ R)  
    α = (P ∨ R)
  - How to check?  
    - One way is to enumerate all elements in the truth table - slow

Satisfiability, Validity, & Entailment
- S is *satisfiable* if it is true in some model (aka world, interpretation)
- S is *unsatisfiable* if it is false all models
- S is *valid* if it is true in all models
- S1 entails S2 if wherever S1 is true S2 is also true

Propositional Logic
- **Syntax**
  - Atomic sentences: P, Q, ...
  - Connectives: ∧, ∨, ¬, ⇒
- **Semantics**
  - Model = an assignment of T/F values to every atomic sentence  
  - Truth Tables

<table>
<thead>
<tr>
<th>P</th>
<th>Q</th>
<th>¬P</th>
<th>P ∨ Q</th>
<th>P ∧ Q</th>
<th>P ⇒ Q</th>
<th>P ⇔ Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
</tr>
<tr>
<td>false</td>
<td>true</td>
<td>true</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
</tr>
<tr>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
<td>true</td>
</tr>
</tbody>
</table>

Types of Reasoning (Inference)
- **Deduction (showing entailment, |=)**
  
  S = question

  Prove that KB |= S

  Two approaches:
  - Rules to derive new formulas from old (inference)
  - Show (KB ∧ ¬S) is unsatisfiable

- **Model Finding (showing satisfiability)**
  
  S = description of problem

  Show S is satisfiable
  
  A kind of constraint satisfaction
Propositional Logic: Inference Algorithms

1. Backward & Forward Chaining
2. Resolution (Proof by Contradiction)
3. Exhaustive Enumeration
4. DPLL (Davis, Putnam Loveland & Logemann)
5. GSAT

Wumpus World

- Performance measure
  - Gold: +1000, death: -1000
  - -1 per step, -10 for using the arrow
- Environment
  - Squares adjacent to wumpus are smelly
  - Squares adjacent to pit are breezy
  - Glitter if gold is in the same square
  - Shooting kills wumpus if you are facing it
  - Shooting uses up the only arrow
  - Grabbing picks up gold if in same square
  - Releasing drops the gold in same square
- Sensors: Stench, Breeze, Glitter, Bump, Scream
- Actuators: Left turn, Right turn, Forward, Grab, Release, Shoot

Exploring a wumpus world

[Diagram of a wumpus world]

[Diagram of a wumpus world]

[Diagram of a wumpus world]

[Diagram of a wumpus world]
Exploring a wumpus world

Wumpus world sentences: KB

Let $P_{i,j}$ be true if there is a pit in $[i, j]$.
Let $B_{i,j}$ be true if there is a breeze in $[i, j]$.

KB:

- $\neg P_{1,1}$
- $\neg B_{1,1}$

"Pits cause breezes in adjacent squares"

$B_{1,1} \Leftrightarrow (P_{1,2} \lor P_{2,1})$
$B_{2,1} \Leftrightarrow (P_{1,1} \lor P_{2,2} \lor P_{3,1})$

Propositional Logic:

Inference Algorithms

1. Backward & Forward Chaining
2. Resolution (Proof by Contradiction)
3. Exhaustive Enumeration
4. DPLL (Davis, Putnam Loveland & Logemann)
5. GSAT

Representing Formulae

- CNF = Conjunctive Normal Form
  - Conjunction ($\land$) of Disjunctions ($\lor$)
- Represent as set of sets
  - $((A, B), (\neg A, C), (\neg C))$
  - $((\neg A), (A))$
  - $()$
  - $((A))$
  - $()$

Inference 4: DPLL

(Enumeration of Partial Models)

[Davis, Putnam, Loveland & Logemann 1962]

Version 1

dpll_1(pa) {
    if (pa makes $F$ false) return false;
    if (pa makes $F$ true) return true;
    choose $P$ in $F$;
    if (dpll_1(pa $\cup\{P=0\}$)) return true;
    return dpll_1(pa $\cup\{P=1\}$);
}

Returns true if $F$ is satisfiable, false otherwise

DPLL Version 1

$(a \lor b \lor c)$
$(a \lor \neg b)$
$(a \lor \neg c)$
$(\neg a \lor c)$
**Improving DPLL**

If literal \( L_i \) is true, then clause \( (L_i \lor L_j \lor \ldots) \) is true
If clause \( C_i \) is true, then \( C_i \land C_j \land \ldots \) has the same value as \( C_i \land \ldots \)
Therefore: Okay to delete clauses containing true literals!

If literal \( L_i \) is false, then clause \( (L_i \lor L_j \lor \ldots) \) has the same value as \( (L_i \lor L_j \lor \ldots) \)
Therefore: Okay to delete clauses containing false literals!

---

**DPLL version 2**

\[
dpll_2(F, \text{literal}) \{
    \text{remove clauses containing literal}
    \text{if (F contains no clauses) return true;}
    \text{shorten clauses containing } \neg \text{literal}
    \text{if (F contains empty clause) return false;}
    \text{choose } V \text{ in } F;
    \text{if (dpll_2(F, } \neg \text{V)) return true;}
    \text{return dpll_2(F, V) ;}
\}
\]

Partial assignment corresponding to a node is the set of chosen literals on the path from the root to the node.

---

**Benefit**

- Like forward checking
- Can backtrack before getting to leaf

---

**Structure in Clauses**

- **Unit Literals**
  A literal that appears in a singleton clause
  \[
  \{\neg b \ c \ \neg c \ a \ b \ e \ (d \ b) \ (e \ a \neg c)\}
  \]
  Might as well set it true! And simplify
  \[
  \{\neg b \} \quad \{a \neg b e \ (d b)\} \quad \{(d)\}
  \]

- **Pure Literals**
  A symbol that always appears with same sign
  \[
  \{\neg a \neg b \ c \ (\neg c \ d \ (\neg e) \ (\neg a \neg b \ e) \ (d \ b) \ (e \ a \neg c)\}
  \]
  Might as well set it true! And simplify
  \[
  \{a \neg b \ c\} \quad \{\neg a \neg b \ e\} \quad \{e \ a \neg c\}
  \]
DPLL (for real)
Davis - Putnam - Loveland - Logemann

dpll(F, literal){
    remove clauses containing literal
    if (F contains no clauses) return true;
    shorten clauses containing \neg literal
    if (F contains empty clause)
        return false;
    if (F contains a unit or pure L)
        return dpll(F, L);
    choose V in F;
    if (dpll(F, \neg V)) return true;
    return dpll(F, V);
}

Compare with DPLL Version 1

Heuristic Search in DPLL

- Heuristics are used in DPLL to select a (non-unit, non-pure) proposition for branching

- Idea: identify a most constrained variable
  - Likely to create many unit clauses
- MOM's heuristic:
  - Most occurrences in clauses of minimum length

Success of DPLL

- 1962 - DPLL invented
- 1992 - 300 propositions
- 1997 - 600 propositions (satz)
- Additional techniques:
  - Learning conflict clauses at backtrack points
  - Randomized restarts
  - 2002 (zChaff) 1,000,000 propositions - encodings of hardware verification problems

Other Ideas?

- How else could we solve SAT problems?
WalkSat (Take 1)

- **Local** search (Hill Climbing + Random Walk) over space of complete truth assignments
  - With prob \( p \): flip any variable in any unsatisfied clause
  - With prob \( 1-p \): flip best variable in any unsat clause
    - best = one which minimizes #unsatisfied clauses

Refining Greedy Random Walk

- Each flip
  - makes some false clauses become true
  - breaks some true clauses, that become false
- Suppose \( s1 \rightarrow s2 \) by flipping \( x \). Then:
  \[
  \#\text{unsat}(s2) = \#\text{unsat}(s1) - \text{make}(s1,x) + \text{break}(s1,x)
  \]
- Idea 1: if a choice breaks nothing, it's likely good!
- Idea 2: near the solution, only the break count matters
  - the make count is usually 1

Walksat (Take 2)

\[
\text{state} = \text{random truth assignment};
\]
while ! GoalTest(state) do
  clause := random member \( \{ C \mid C \text{ is false in state} \} \);
  for each \( x \) in clause do compute break\([x]\);
  if exists \( x \) with break\([x]=0 \) then
    var := \( x \);
  else
    with probability \( p \) do
      with probability \( p \) do
        var := random member \( \{ x \mid x \text{ is in clause} \} \);
      else
        var := arg min \{ break\([x]\) \mid x \text{ is in clause} \};
    endif
  endif
  state\([var]\) := 1 - state\([var]\);
end
return state;

Put everything inside of a restart loop.
Parameters: \( p \), max_flips, max_runs

Random 3-SAT

- Random 3-SAT
  - sample uniformly from space of all possible 3-clauses
  - \( n \) variables, \( l \) clauses
- Which are the hard instances?
  - around \( l/n = 4.3 \)

Special Syntactic Forms

- **General Form:**
  \((q \land \neg r) \rightarrow s) \land (s \land t)\)
- **Conjunction Normal Form (CNF)**
  \(\neg (q \lor r \lor s) \land (\neg s \lor \neg t)\)
  Set notation: \( \{ (\neg q, r, s), (\neg s, \neg t) \} \)
  empty clause \( () \equiv \text{false} \)
- **Binary clauses:** 1 or 2 literals per clause
  \(\neg q \lor r \quad (\neg s \lor \neg t)\)
- **Horn clauses:** 0 or 1 positive literal per clause
  \(\neg q \lor \neg r \lor s \quad (\neg s \lor \neg t)\)
  \((q \land r) \rightarrow s \quad (s \land t) \rightarrow \text{false})\)

Prop. Logic Themes

- **Expressiveness**
  Expressive but awkward
  No notion of objects, properties, or relations
  Number of propositions is fixed
  **Brittle**
- **Tractability**
  NP in general
  Completeness / speed tradeoff
  Horn clauses, binary clauses
Overview

- Knowledge Representation & Reasoning
- Propositional Logic
- First-Order Logic
  - Foundations: Syntax, semantics & inference
  - Algorithms: Chaining, Resolution, Compilation to SAT
  - Tractable subsets
- Markov Logic

Propositional Logic vs. First Order

<table>
<thead>
<tr>
<th>Ontology</th>
<th>Propositional Symbols</th>
<th>Objects, Properties, Relations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
<td>Atomic sentences</td>
<td>Sentences have structure: terms father-of(mother-of(X))</td>
</tr>
<tr>
<td>Semantics</td>
<td>Truth Tables</td>
<td>Interpretations (Much more complicated)</td>
</tr>
<tr>
<td>Inference</td>
<td>Algorithm</td>
<td>DPLL, WalkSAT, Fast in practice</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unification, Forward, Backward chaining, Prolog, theorem proving</td>
</tr>
<tr>
<td>Complexity</td>
<td>NP-Complete</td>
<td>Semi-decidable</td>
</tr>
</tbody>
</table>

FOL Definitions

- **Constants**: a, b, dog33.
  - Name a specific object.
- **Variables**: X, Y.
  - Refer to an object without naming it.
- **Functions**: dad-of
  - Mapping from objects to objects.
- **Terms**: dad-of(dog33)
  - Refer to objects
- **Atomic Sentences**: in(dad-of(dog33), food6)
  - Can be true or false
  - Correspond to propositional symbols P, Q

More Definitions

- **Quantifiers**:
  - ∀ Forall
  - ∃ There exists
- **Examples**
  - Dumbo is grey
    grey(dumbo)
  - Elephants are grey
    ∀ x elephant(x) ⇒ grey(x)
  - There is a grey elephant
    ∃ x elephant(x) ∧ grey(x)

Quantifier / Connective Interaction

1. ∀x E(x) ∧ G(x)
   - “x is an elephant”
   - “x has the color grey”
2. ∀x E(x) ⇒ G(x)
3. ∃x E(x) ∧ G(x)
4. ∃x E(x) ⇒ G(x)

Nested Quantifiers:

- **Examples**
  - Every dog has a tail
    ∀ d ∃ t has(d,t)
  - Everyone shares a tail
    ∀ d ∃ t ∀ v has(d,t)
  - Someone is loved by everyone
    ∃ x ∀ y loves(y,x)
Wumpus world in prop logic

Let \( P_{i,j} \) be true if there is a pit in \([i, j]\).

Let \( B_{i,j} \) be true if there is a breeze in \([i, j]\).

\[
\text{KB:} \\
\neg P_{1,1} \\
\neg B_{1,1} \\
\text{“Pits cause breezes in adjacent squares”} \\
B_{1,1} \iff (P_{1,2} \lor P_{2,1}) \\
B_{2,1} \iff (P_{1,1} \lor P_{2,2} \lor P_{3,1}) \\
\]

Semantics

- **Syntax**: a description of the legal arrangements of symbols
  
  e.g. “(Def “sentences”)

- **Semantics**: what the arrangement of symbols means in the world

Models

- Logicians often think in terms of models, which are formally structured worlds with respect to which truth can be evaluated
  
  - In propositional case, each model = truth assignment
  
  - Set of models can be enumerated in a truth table

  - E.g. \( \text{KB} = (P \lor Q) \land \neg (P \lor Q) \)

  \( \models \) is the set of all models of \( \alpha \)

  - Then \( \text{KB} \models \alpha \) iff \( M(\text{KB}) \subseteq M(\alpha) \)

Satisfiability, Validity, & Entailment

- \( S \) is valid if it is true in all models

- \( S \) is satisfiable if it is true in some model

- \( S \) is unsatisfiable if it is false all model

Propositional Logic: **SEMANTICS**

- Possible models are TRUTH ASSIGNMENTS
  
  - Assignment to each variable either T or F
  
  - Assignment of T or F to each connective

<table>
<thead>
<tr>
<th>Symbols:</th>
<th>P</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>Truth:</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Models:</th>
<th>T</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>T</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td></td>
</tr>
</tbody>
</table>
First Order Logic: Worlds

• Depiction of one possible world

Models = Mappings

syntactic tokens $\rightarrow$ world elements

Another interpretation, same assumptions

- Constants:
  - Richard
  - John
- Functions:
  - Leg(p, l)
  - On(x, y)
- Relations:
  - King(p)

FOL Reasoning

• FO Forward & Backward Chaining
• FO Resolution
• Many other types of theorem proving
• Specialized provers for restricted representations
  - Description logics
  - Horn Clauses
• Compilation to SAT

Compilation to Prop. Logic I

• Typed Logic
  - $\forall_{\text{city}} a, b$ connected(a, b)
• Finite Universe
  - Cities: seattle, tacoma, enumclaw
• Equivalent propositional formula:

Compilation to Prop. Logic II

• Universe
  - Cities: Seattle, Chicago
  - Firms: Microsoft, Boeing
• First-Order formula
  - $\forall_{\text{city}} c \exists_{\text{firm}} f$ hasHQ(c, f)
• Equivalent propositional formula?
Hey!

- You said FO Inference is semi-decidable
- But you compiled it to SAT
  - Which is NP Complete
- So now we can always do the inference?!?
  - Tho it might take exponential time...

- Something seems wrong here...???

Restricted Forms of FO Logic

- Known, Finite Universes
  - Compile to SAT
- Description Logics (Frame Systems)
  - Ban certain types of expressions
- Horn Clauses
  - Aka Prolog
- Function-Free Horn Clauses
  - Aka Datalog

KR with Description Logics

Assertions

Abox

mother(jane)  
child-of(jane, bob)  
...

Term Defs

Tbox

person

father  
mother  
grandmother

Logical and Statistical AI

<table>
<thead>
<tr>
<th>Field</th>
<th>Logical approach</th>
<th>Statistical approach</th>
</tr>
</thead>
<tbody>
<tr>
<td>Knowledge representation</td>
<td>First-order logic</td>
<td>Graphical models</td>
</tr>
<tr>
<td>Automated reasoning</td>
<td>Satisfiability testing</td>
<td>Markov chain</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Monte Carlo</td>
</tr>
<tr>
<td>Machine learning</td>
<td>Inductive logic programming</td>
<td>Neural networks</td>
</tr>
<tr>
<td>Planning</td>
<td>Classical planning</td>
<td>Markov decision processes</td>
</tr>
<tr>
<td>Natural language processing</td>
<td>Definite clause grammars</td>
<td>Prob. context-free grammars</td>
</tr>
</tbody>
</table>

Propositional Logic


Propositional  
Logic


First-Order  
Logic


Propositional  
Logic


Probabilistic Graphical  
Models (Bayes Nets)


Uncertainty


We Need to Unify the Two

- The real world is complex and uncertain
- Logic handles complexity
- Probability handles uncertainty

Progress to Date

- Probabilistic logic [Nilsson, 1986]
- Statistics and beliefs [Halpern, 1990]
- Knowledge-based model construction [Wellman et al., 1992]
- Stochastic logic programs [Muggleton, 1996]
- Probabilistic relational models [Friedman et al., 1999]
- Relational Markov networks [Taskar et al., 2002]
- Etc.
- Here at UW: MLNs [Richardson & Domingos, 2004]

Markov Logic

- Syntax: Weighted first-order formulas
- Semantics: Templates for Markov nets
- Inference: WalkSAT, MCMC, KBMC
- Learning: Voted perceptron, pseudo-likelihood, inductive logic programming
- Software: Alchemy
- Applications: Information extraction, link prediction, etc.

Overview

- Motivation
- Background
- Markov logic
- Inference
- Learning
- Software
- Applications
- Discussion

Markov Networks

- Undirected graphical models

\[ P(x) = \frac{1}{Z} \prod \Phi_i(x_i) \]

\[ Z = \sum x \prod \Phi_i(x_i) \]

<table>
<thead>
<tr>
<th>Feature</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Smoking</td>
<td>4.5</td>
</tr>
<tr>
<td>Cancer</td>
<td>4.5</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>

Markov Networks

- Undirected graphical models

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

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

\[ w_i = 1.5 \]
**First-Order Logic**

- **Constants, variables, functions, predicates**
  E.g.: Anna, x, MotherOf(x), Friends(x, y)
- **Grounding**: Replace all variables by constants
  E.g.: Friends (Anna, Bob)
- **World (model, interpretation)**: Assignment of truth values to all ground predicates

**Overview**

- **Motivation**
- **Background**
- **Markov logic**
- **Inference**
- **Learning**
- **Software**
- **Applications**
- **Discussion**

**Markov Logic**

- A logical KB is a set of **hard constraints** on the set of possible worlds
- Let’s make them **soft constraints**: When a world violates a formula, it becomes less probable, not impossible
- Give each formula a **weight**
  (Higher weight ⇒ Stronger constraint)
  \[ P(\text{world}) = \exp\left(\sum \text{weights of formulas it satisfies}\right) \]

**Definition**

- 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
- Together with a set of constants, it 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: Friends & Smokers**

- Smoking causes cancer.
- Friends have similar smoking habits.

**Example: Friends & Smokers**

\[ \forall x \, \text{Smokes}(x) \Rightarrow \text{Cancer}(x) \]
\[ \forall x, y \, \text{Friends}(x, y) \Rightarrow (\text{Smokes}(x) \Leftrightarrow \text{Smokes}(y)) \]
Example: Friends & Smokers

1.5 $\forall x \text{ Smokes}(x) \Rightarrow \text{Cancer}(x)$
1.1 $\forall x, y \text{ Friends}(x, y) \Rightarrow (\text{Smokes}(x) \Leftrightarrow \text{Smokes}(y))$

Two constants: Anna (A) and Bob (B)
Markov Logic Networks

- MLN is template for ground Markov nets
- Probability of a world $x$:
  \[ P(x) = \frac{1}{Z} \exp \left( \sum w_i(x) \right) \]
  
  Weight of formula $i$ 
  No. of true groundings of formula $i$ in $x$

- Typed variables and constants greatly reduce size of ground Markov net
- Functions, existential quantifiers, etc.
- Infinite and continuous domains

Relation to Statistical Models

- Special cases:
  - Markov networks
  - Bayesian networks
  - Log-linear models
  - Exponential models
  - Max. entropy models
  - Gibbs distributions
  - Boltzmann machines
  - Logistic regression
  - Hidden Markov models
  - Conditional random fields

- Obtained by making all predicates zero-arity
- Markov logic allows objects to be interdependent (non-i.i.d.)

Relation to First-Order Logic

- Infinite weights $\Rightarrow$ First-order logic
- Satisfiable KB, positive weights $\Rightarrow$
  Satisfying assignments = Modes of distribution
- Markov logic allows contradictions between formulas

Overview

- Motivation
- Background
- Markov logic
- Inference
- Learning
- Software
- Applications
- Discussion

MAP/MPE Inference

- **Problem**: Find most likely state of world given evidence

\[ \arg \max_y P(y \mid x) \]

Query Evidence

MAP/MPE Inference

- **Problem**: Find most likely state of world given evidence

\[ \arg \max_y \frac{1}{Z_x} \exp \left( \sum w_i(x, y) \right) \]
MAP/MPE Inference

- **Problem:** Find most likely state of world given evidence

\[
\arg \max_y \sum_j w_j n_j(x, y)
\]

- This is just the weighted MaxSAT problem
- Use weighted SAT solver (e.g., MaxWalkSAT [Kautz et al., 1997])
- Potentially faster than logical inference (!)

The WalkSAT Algorithm

```python
for i ← 1 to max-tries do
    solution = random truth assignment
    for j ← 1 to max-flips do
        if all clauses satisfied then
            return solution
        c ← random unsatisfied clause with probability p
        flip a random variable in c
        else flip variable in c that maximizes number of satisfied clauses
    return failure
```

The MaxWalkSAT Algorithm

```python
for i ← 1 to max-tries do
    solution = random truth assignment
    for j ← 1 to max-flips do
        if \( \sum \text{weights}(\text{sat. clauses}) > \text{threshold} \) then
            return solution
        c ← random unsatisfied clause with probability p
        flip a random variable in c
        else flip variable in c that maximizes \( \sum \text{weights}(\text{sat. clauses}) \)
    return failure, best solution found
```

But ... Memory Explosion

- **Problem:**
  - if there are \( n \) constants
  - and the highest clause arity is \( c \),
  - the ground network requires \( \mathcal{O}(n^c) \) memory

- **Solution:**
  - Exploit sparseness; ground clauses lazily
  - \( \rightarrow \) LazySAT algorithm [Singla & Domingos, 2006]

Computing Probabilities

- \( P(\text{Formula} | \text{MLN}, C) = ? \)
- MCMC: Sample worlds, check formula holds
- \( P(\text{Formula}_1 | \text{Formula}_2, \text{MLN}, C) = ? \)
- If \( \text{Formula}_2 = \) Conjunction of ground atoms
  - First construct min subset of network necessary to answer query (generalization of KBMC)
  - Then apply MCMC (or other)
- Can also do lifted inference [Braz et al, 2005]
Ground Network Construction

\[
\text{network} \leftarrow \emptyset \\
\text{queue} \leftarrow \text{query nodes} \\
\text{repeat} \\
\quad \text{node} \leftarrow \text{front}(\text{queue}) \\
\quad \text{remove node from queue} \\
\quad \text{add node to network} \\
\quad \text{if node not in evidence then} \\
\quad \quad \text{add neighbors(node) to queue} \\
\text{until queue = } \emptyset
\]

MCMC: Gibbs Sampling

\[
\text{state} \leftarrow \text{random truth assignment} \\
\text{for } i \leftarrow 1 \text{ to num-samples do} \\
\quad \text{for each variable } x \\
\quad \quad \text{sample } x \text{ according to } P(x| \text{neighbors}(x)) \\
\quad \quad \text{state} \leftarrow \text{state with new value of } x \\
\quad P(F) \leftarrow \text{fraction of states in which } F \text{ is true}
\]

But ... Insufficient for Logic

- **Problem:** Deterministic dependencies break MCMC
  Near-deterministic ones make it very slow

- **Solution:**
  Combine MCMC and WalkSAT
  → MC-SAT algorithm [Poon & Domingos, 2006]

Overview

- Motivation
- Background
- Markov logic
- Inference
- Learning
- Software
- 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
  \[
  \frac{\partial}{\partial w_i} \log P_w(x) = \frac{n_i(x)}{E_i} - \bar{E}_i
  \]
- Requires inference at each step (slow!)
Pseudo-Likelihood

\[ PL(x) = \prod_i P(x_i \mid \text{neighbors}(x_i)) \]

- 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 \( x \)
  \[
  \frac{\partial}{\partial w_j} \log P_j(y \mid x) = \frac{n_j(x, y)}{\sum_k n_k(x, y)} - \mathbb{E}_x [n_j(x, y)]
  \]
- Approximate expected counts by counts in MAP state of \( y \) given \( x \)

Voted Perceptron

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

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

Voted Perceptron for MLNs

- HMMs are special case of MLNs
- Replace Viterbi by MaxWalkSAT
- Network can now be arbitrary graph

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

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 [Kok & Domingos, 2005]
  - Shortest-first [Kok & Domingos, 2005]
  - Bottom-up [Mihalkova & Mooney, 2007]
Overview

- Motivation
- Background
- Markov logic
- Inference
- Learning
- Software
- Applications
- Discussion

Alchemy

Open-source software including:
- Full first-order logic syntax
- Generative & discriminative weight learning
- Structure learning
- Weighted satisfiability and MCMC
- Programming language features
  alchemy.cs.washington.edu

<table>
<thead>
<tr>
<th>Representation</th>
<th>Alchemy</th>
<th>Prolog</th>
<th>BUGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>F.O. Logic + Markov nets</td>
<td>Horn clauses</td>
<td>Bayes nets</td>
<td></td>
</tr>
<tr>
<td>Inference</td>
<td>Model checking, MC-SAT</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>

Applications

- Information extraction*
- Entity resolution
- Link prediction
- Collective classification
- Web mining
- Natural language processing

* Markov logic approach won LLL-2005 information extraction competition [Riedel & Klein, 2005]

Information Extraction

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


H. Poon & P. Domingos, Sound and Efficient Inference
Parag Singla and Pedro Domingos, "Memory-Efficient Inference in Relational Domains" (AAAI-06).


H. Poon & P. Domingos, Sound and Efficient Inference

Segmentation

Entity Resolution

Entity Resolution

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

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)
### Types and Predicates

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

#### Evidence
- `Token(token, position, citation)`
- `InField(position, field, citation)`
- `SameField(field, citation, citation)`
- `SameCit(citation, citation)`

### Formulas

- `Token(\langle t, i, c \rangle) \Rightarrow InField(i, f, c)`
- `InField(i, f, c) \iff InField(i+1, f, c)`
- `f \neq f' \Rightarrow (!InField(i, f, c) \lor !InField(i, f', c))`
- `Token(\langle t, i, c \rangle) \land InField(i, f, c) \land Token(\langle t, i', c \rangle) \land InField(i', f, c') \Rightarrow SameField(f, c, c')`
- `SameField(f, c, c') \iff SameCit(c, c')`
- `SameField(f, c, c') \land SameField(f, c', c'') \Rightarrow SameField(f, c, c'')`
- `SameCit(c, c') \land SameCit(c', c'') \Rightarrow SameCit(c, c'')`

### Formulas

- `Token(\langle t, i, c \rangle) \Rightarrow InField(i, f, c)`
- `InField(i, f, c) \iff InField(i+1, f, c)`
- `f \neq f' \Rightarrow (!InField(i, f, c) \lor !InField(i, f', c))`
- `Token(\langle t, i, c \rangle) \land InField(i, f, c) \land Token(\langle t, i', c \rangle) \land InField(i', f, c') \Rightarrow SameField(f, c, c')`
- `SameField(f, c, c') \iff SameCit(c, c')`
- `SameField(f, c, c') \land SameField(f, c', c'') \Rightarrow SameField(f, c, c'')`
- `SameCit(c, c') \land SameCit(c', c'') \Rightarrow SameCit(c, c'')`
Formulas

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

Results: Segmentation on Cora

![Graph showing precision and recall for different token combinations]

Results: Matching Venues on Cora

![Graph showing precision and recall for different token combinations]