Markov Logic: Representation

Overview

- Statistical relational learning
- Markov logic
- Basic inference
- Basic learning

Statistical Relational Learning

Goals:

- Combine (subsets of) logic and probability into a single language
- Develop efficient inference algorithms
- Develop efficient learning algorithms
- Apply to real-world problems

L. Getoor & B. Taskar (eds.), *Introduction to Statistical Relational Learning,* MIT Press, 2007.

Plethora of Approaches

- 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]
- Bayesian logic [Milch et al., 2005]
- Markov logic [Richardson & Domingos, 2006]
- And many others!

Key Dimensions

- **Logical language**
  First-order logic, Horn clauses, frame systems
- **Probabilistic language**
  Bayes nets, Markov nets, PCFGs
- **Type of learning**
  - Generative / Discriminative
  - Structure / Parameters
  - Knowledge-rich / Knowledge-poor
- **Type of inference**
  - MAP / Marginal
  - Full grounding / Partial grounding / Lifted

First-Order Logic

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

Example: Friends & Smokers

Smoking causes cancer.

Friends have similar smoking habits.

Example: Friends & Smokers

$$\forall x\ Smokes(x) \Rightarrow Cancer(x)$$

$$\forall x, y\ Friends(x, y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$$

Inference in First-Order Logic

- Traditionally done by theorem proving (e.g.: Prolog)
- Propositionalization followed by model checking turns out to be faster (often a lot)
- **Propositionalization:**
  Create all ground atoms and clauses
- **Model checking:** Satisfiability testing
- Two main approaches:
  - **Backtracking** (e.g.: DPLL)
  - **Stochastic local search** (e.g.: WalkSAT)

Satisfiability

- **Input:** Set of clauses
  (Convert KB to conjunctive normal form (CNF))
- **Output:** Truth assignment that satisfies all clauses, or failure
- The paradigmatic NP-complete problem
- **Solution:** Search
- **Key point:**
  Most SAT problems are actually easy
- **Hard region:** Narrow range of
  #Clauses / #Variables

Backtracking

- Assign truth values by depth-first search
- Assigning a variable deletes false literals and satisfied clauses
- Empty set of clauses: Success
- Empty clause: Failure
- Additional improvements:
  - **Unit propagation** (unit clause forces truth value)
  - **Pure literals** (same truth value everywhere)

The DPLL Algorithm

```
if CNF is empty then
    return true
else if CNF contains an empty clause then
    return false
else if CNF contains a pure literal $x$ then
    return DPLL(CNF($x$))
else if CNF contains a unit clause $\{u\}$ then
    return DPLL(CNF($u$))
else
    choose a variable $x$ that appears in CNF
    if DPLL(CNF($x$)) = true then return true
    else return DPLL(CNF($\neg x$))
```

Stochastic Local Search

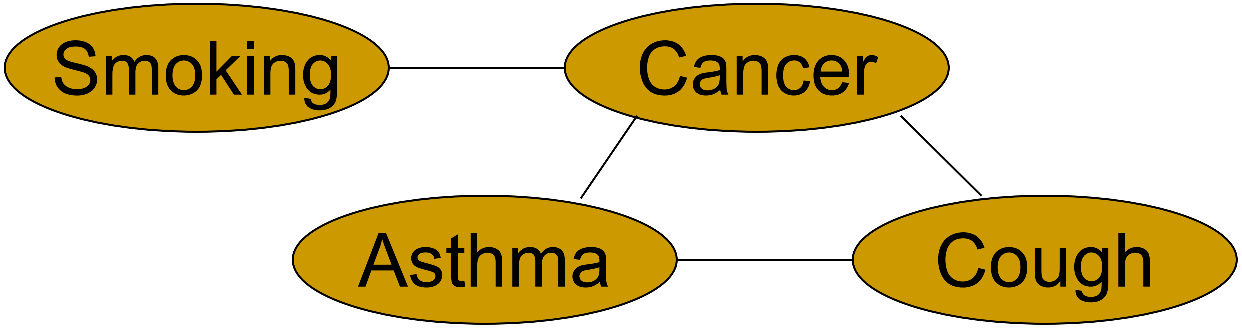
- Uses complete assignments instead of partial
- Start with random state
- Flip variables in unsatisfied clauses
- Hill-climbing: Minimize # unsatisfied clauses
- Avoid local minima: Random flips
- Multiple restarts

The WalkSAT Algorithm

```
for $i \leftarrow 1$ to max-tries do
    solution = random truth assignment
    for $j \leftarrow 1$ to max-flips do
        if all clauses satisfied then
            return solution
        $c \leftarrow$ 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
```

Markov Networks

- **Undirected** graphical models



- Potential functions defined over cliques

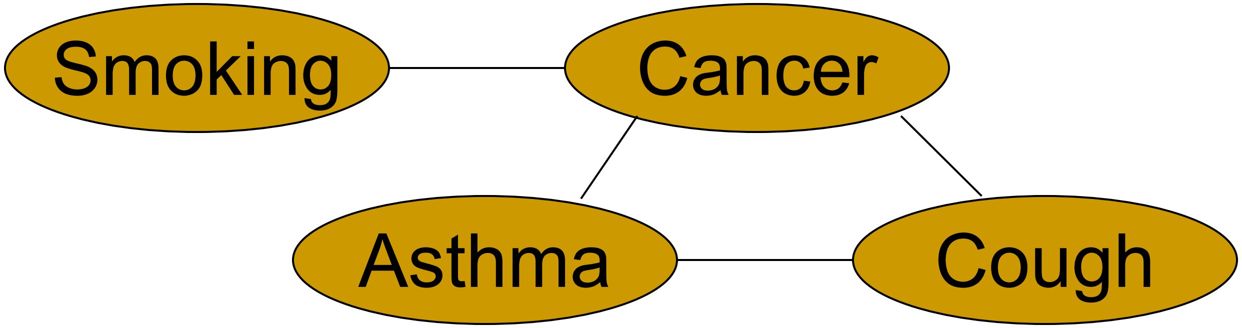
$$P(x) = \frac{1}{Z} \prod_c \Phi_c(x_c)$$

$$Z = \sum_x \prod_c \Phi_c(x_c)$$

Smoking	**Cancer**	**Φ(S,C)**
False	False	4.5
False	True	4.5
True	False	2.7
True	True	4.5

Markov Networks

- **Undirected** graphical models



- Log-linear model:

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

Weight of Feature *i*

Feature *i*

$$f_1(\text{Smoking}, \text{ Cancer}) = \begin{cases} 1 & \text{if } \neg \text{ Smoking } \vee \text{ Cancer} \\ 0 & \text{otherwise} \end{cases}$$

$$w_1 = 1.5$$

Hammersley-Clifford Theorem

If Distribution is strictly positive (P(x) > 0)

And Graph encodes conditional independences

Then Distribution is product of potentials over cliques of graph

Inverse is also true.

("Markov network = Gibbs distribution")

Markov Nets vs. Bayes Nets

Property	Markov Nets	Bayes Nets
Form	Prod. potentials	Prod. potentials
Potentials	Arbitrary	Cond. probabilities
Cycles	Allowed	Forbidden
Partition func.	Z = ?	Z = 1
Indep. check	Graph separation	D-separation
Indep. props.	Some	Some
Inference	MCMC, BP, etc.	Convert to Markov

Inference in Markov Networks

- Computing probabilities
  - Markov chain Monte Carlo
  - Belief propagation
- MAP inference

Markov Logic

- **Logical language:** First-order logic
- **Probabilistic language:** Markov networks
  - **Syntax:** First-order formulas with weights
  - **Semantics:** Templates for Markov net features
- **Learning:**
  - **Parameters:** Generative or discriminative
  - **Structure:** ILP with arbitrary clauses and MAP score
- **Inference:**
  - **MAP:** Weighted satisfiability
  - **Marginal:** MCMC with moves proposed by SAT solver
  - Partial grounding + Lazy inference / Lifted inference

Markov Logic: Intuition

- 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)

$$P(\text{world}) \propto \exp\left(\sum \text{weights of formulas it satisfies}\right)$$

Markov Logic: 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

$$\forall x\; Smokes(x) \Rightarrow Cancer(x)$$

$$\forall x, y\; Friends(x, y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$$

Example: Friends & Smokers

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

Example: Friends & Smokers

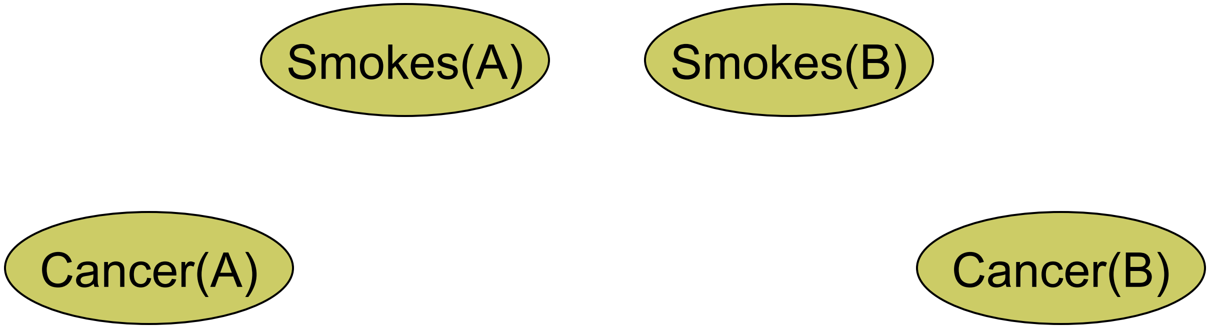
1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

Two constants: **Anna** (A) and **Bob** (B)

Example: Friends & Smokers

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$

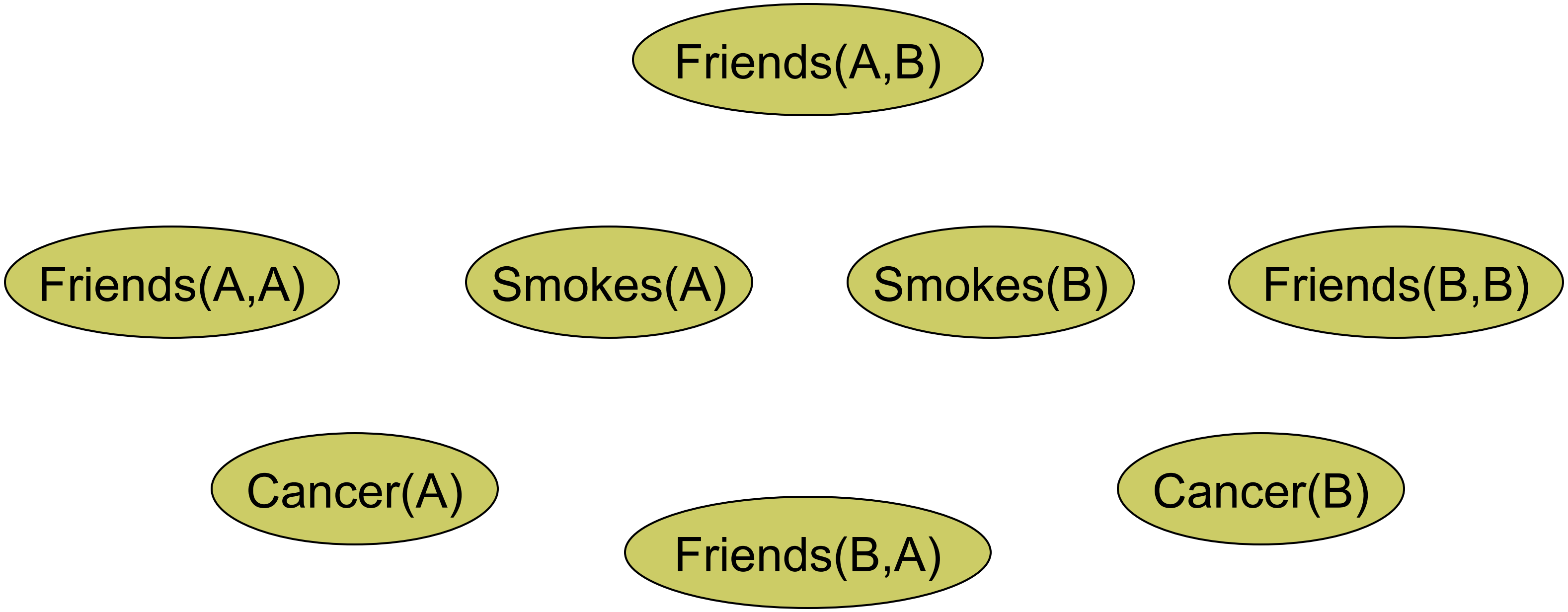
Two constants: **Anna** (A) and **Bob** (B)



Example: Friends & Smokers

1.5	$\forall x \; Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y \; Friends(x,y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

Two constants: **Anna** (A) and **Bob** (B)


Friends(A,B)

Friends(A,A)

Smokes(A)

Smokes(B)

Friends(B,B)

Cancer(A)

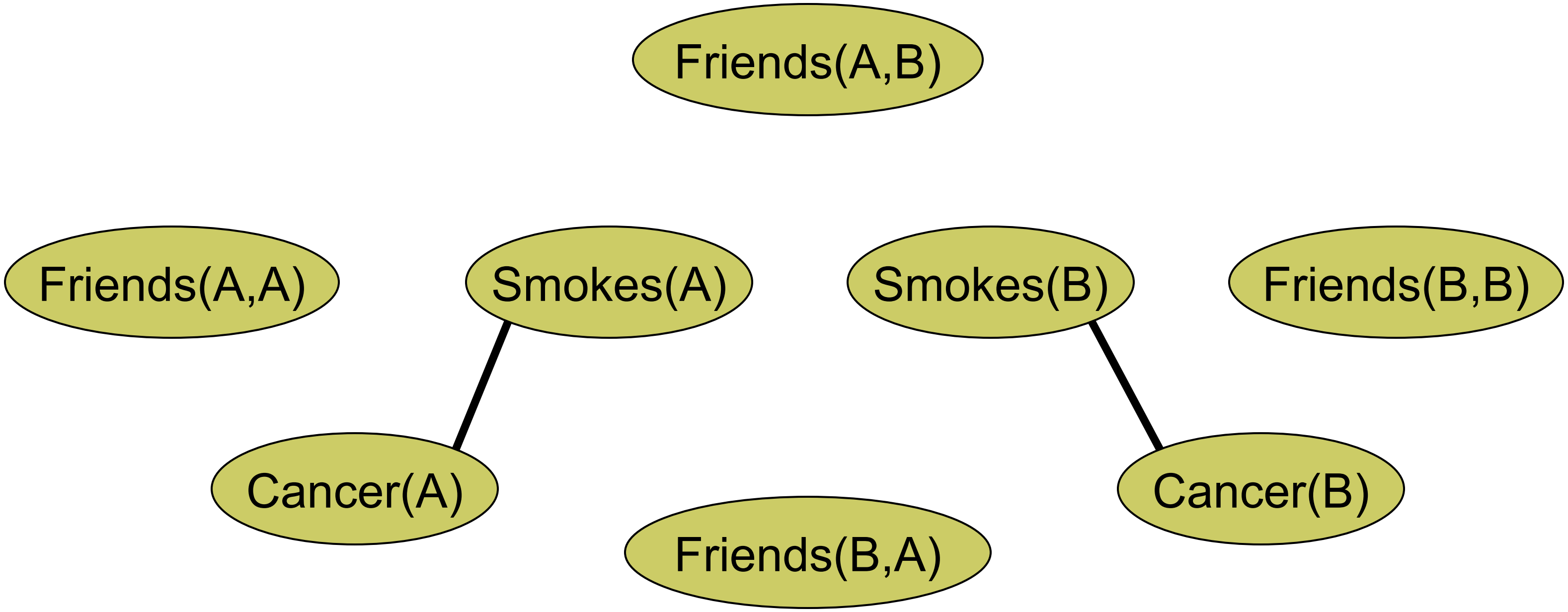
Friends(B,A)

Cancer(B)

Example: Friends & Smokers

1.5	$\forall x \; Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y \; Friends(x,y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

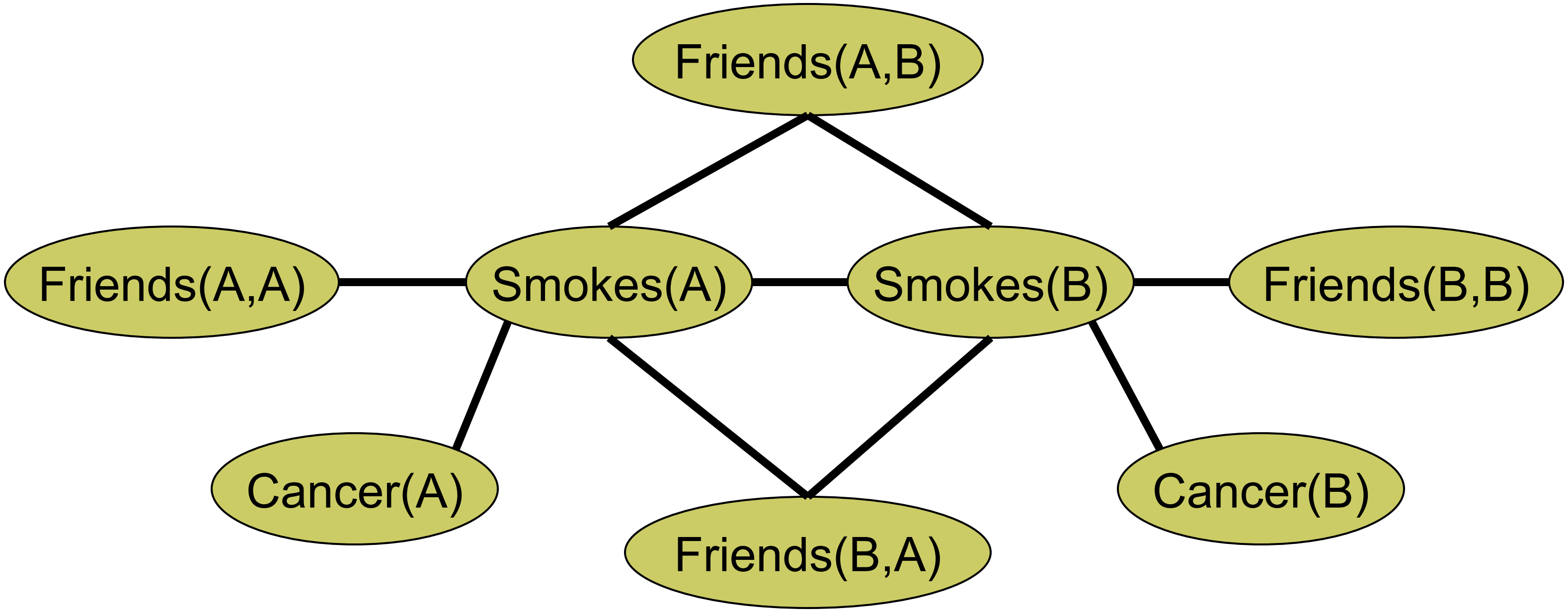
Two constants: **Anna** (A) and **Bob** (B)



Example: Friends & Smokers

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x,y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

Two constants: **Anna** (A) and **Bob** (B)



Markov Logic Networks

- MLN is **template** for ground Markov nets
- Probability of a world x:

$$P(x)=\frac{1}{Z}\exp\left(\sum_i \underbrace{w_i}_{\text{Weight of formula } i}\,\underbrace{n_i(x)}_{\text{No. of true groundings of formula } i \text{ in } x}\right)$$

- **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
  - Markov random fields
  - 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.)

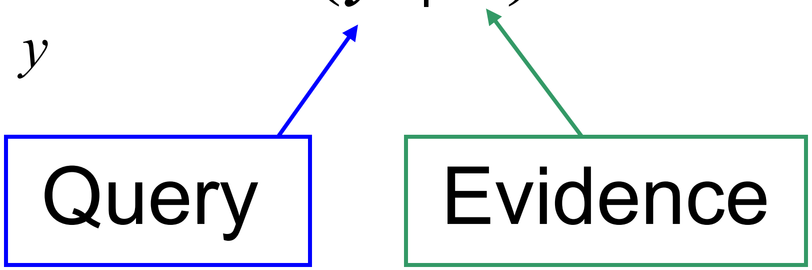
Relation to First-Order Logic

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

MAP Inference

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

$$\arg\max_{y} \; P(y \mid x)$$



Query

Evidence

MAP Inference

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

$$\arg\max_{y} \frac{1}{Z_x} \exp\left(\sum_i w_i n_i(x, y) \right)$$

MAP Inference

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

$$\arg\max_{y} \sum_{i} w_i n_i(x, y)$$

MAP Inference

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

$$\arg\max_{y} \sum_{i} w_i n_i(x, y)$$

- This is just the weighted MaxSAT problem
- Use weighted SAT solver (e.g., MaxWalkSAT [Kautz et al., 1997]

The MaxWalkSAT Algorithm

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

Computing Probabilities

- P(Formula|MLN,C) = ?
- Brute force: Sum probs. of worlds where formula holds
- MCMC: Sample worlds, check formula holds
- P(Formula1|Formula2,MLN,C) = ?
- Discard worlds where Formula 2 does not hold
- In practice: More efficient alternatives

Learning

- Data is a relational database
- For now: Closed world assumption (if not: EM)
- Learning parameters (weights)
  - Similar to learning weights for Markov networks
- Learning structure (formulas)
  - A form of inductive logic programming
  - Also related to learning features for Markov nets

Weight Learning

- Parameter tying: Groundings of same clause

$$\frac{\partial}{\partial w_i}\log P_w(x) = \boxed{n_i(x)} - \boxed{E_w\left[n_i(x)\right]}$$

No. of times clause *i* is true in data

Expected no. times clause *i* is true according to MLN

- Generative learning: Pseudo-likelihood
- Discriminative learning: Cond. likelihood, use MaxWalkSAT for inference

Alchemy

Open-source software including:

- Full first-order logic syntax
- Inference (MAP and conditional probabilities)
- Weight learning (generative and discriminative)
- Structure learning
- Programming language features

alchemy.cs.washington.edu