Advanced Machine Learning Techniques for Temporal, Multimedia, and Relational Data

Vibhav Gogate

The University of Texas at Dallas

Many slides courtesy of Pedro Domingos

Statistical Relational Learning: Motivation

- Most learners assume i.i.d. data (independent and identically distributed)
  - One type of object
  - Objects have no relation to each other
- Real applications: dependent, variously distributed data
  - Multiple types of objects
  - Relations between objects

Examples

- Web search
- Information extraction
- Natural language processing
- Perception
- Medical diagnosis
- Computational biology
- Social networks
- Ubiquitous computing
- Etc.

Costs and Benefits of SRL

- **Benefits**
  - Better predictive accuracy
  - Better understanding of domains
  - Growth path for machine learning
- **Costs**
  - Learning is much harder
  - Inference becomes a crucial issue
  - Greater complexity for user

Goal and Progress

- **Goal:**
  Learn from non-i.i.d. data as easily as from i.i.d. data
- Progress to date
  - Burgeoning research area
  - We're "close enough" to goal
  - Easy-to-use open-source software available
- Lots of research questions (old and new)

Plan

- We have the elements:
  - **Probability** for handling uncertainty
  - **Logic** for representing types, relations, and complex dependencies between them
  - **Learning** and **inference** algorithms for each
- Figure out how to put them together
- Tremendous leverage on a wide range of applications

Disclaimers

- Not a complete survey of statistical relational learning
- Or of foundational areas
- Focus is practical, not theoretical
- Assumes basic background in logic, probability and statistics, etc.
- Please ask questions
- Tutorial and examples available at **alchemy.cs.washington.edu**
- **New version of alchemy available on my website**
  - **http://www.hlt.utdallas.edu/~vgogate/software.html**

Markov Logic

- An approach for statistical relational learning
- Most developed approach to date
- Many other approaches can be viewed as special cases
- Main focus of rest of this tutorial

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 $\Rightarrow$ 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

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

Example: Friends & Smokers

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

Example: Friends & Smokers

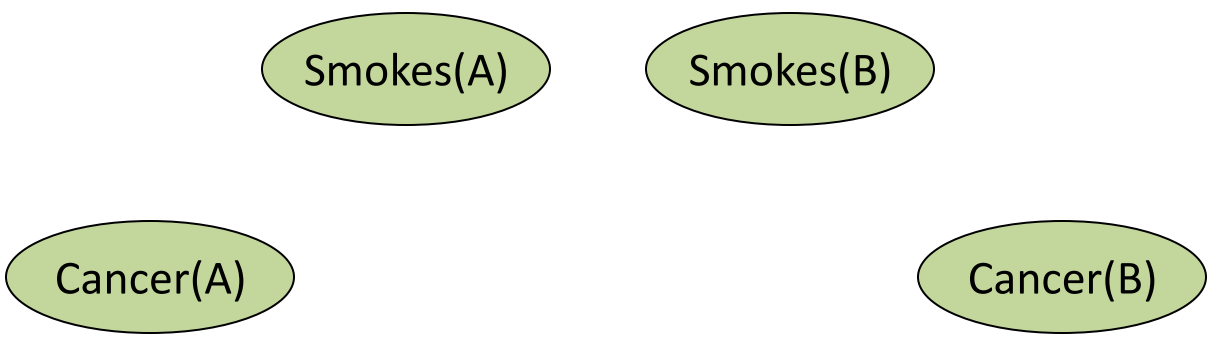
1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

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

Example: Friends & Smokers

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

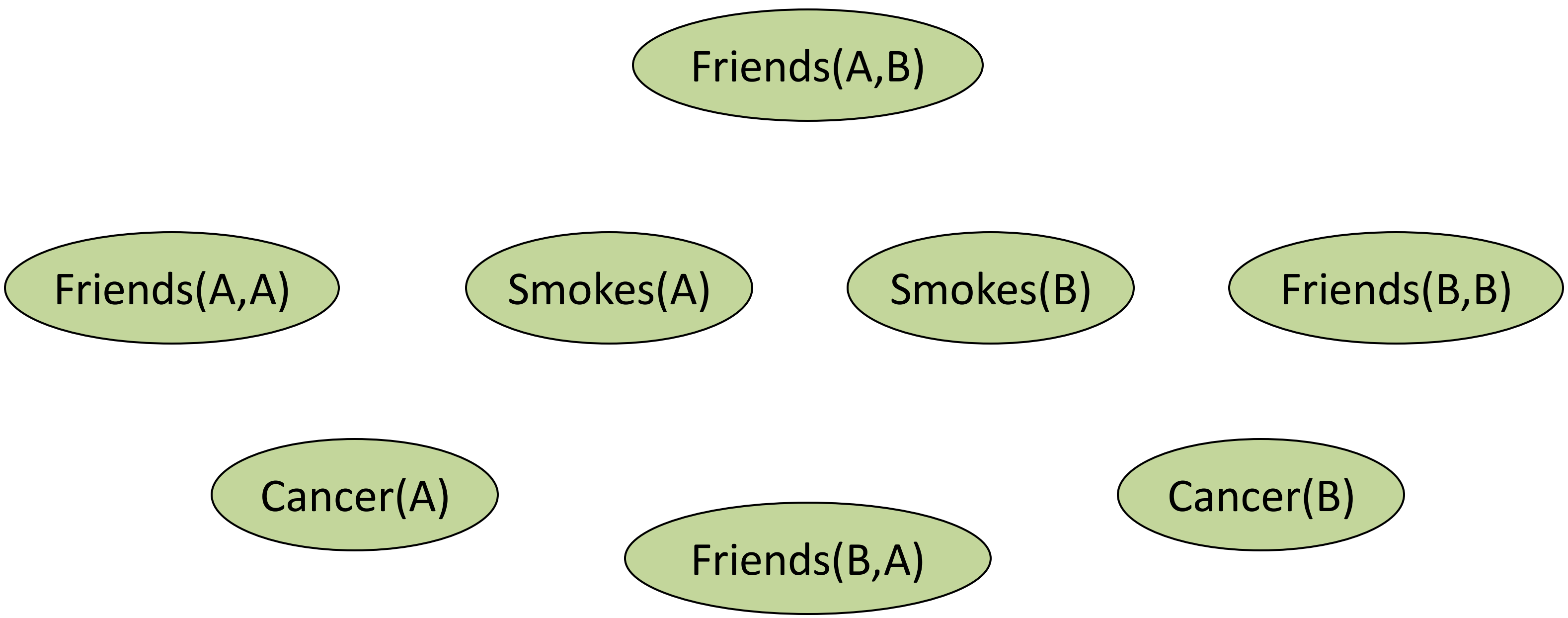
Two constants: **Anna** (A) and **Bob** (B)



Example: Friends & Smokers

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

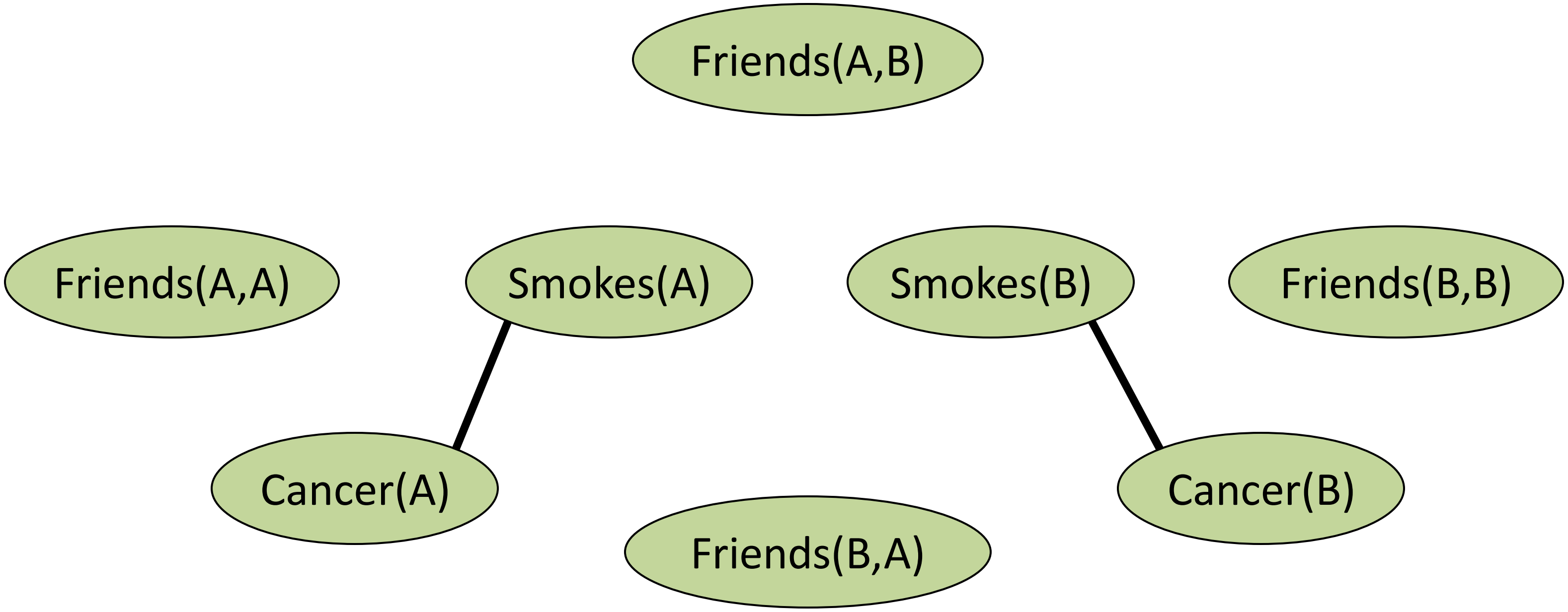
Two constants: **Anna** (A) and **Bob** (B)



Example: Friends & Smokers

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

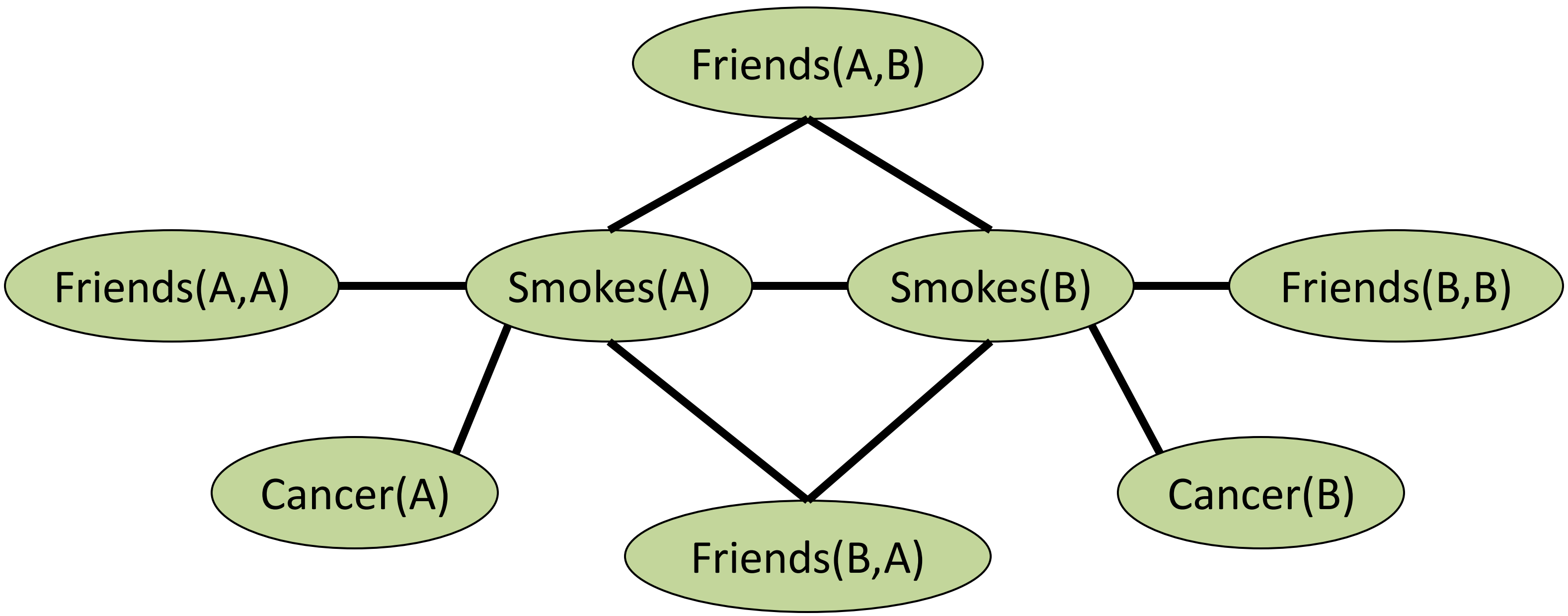
Two constants: **Anna** (A) and **Bob** (B)



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:

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

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

Markov logic

$\forall x \; Smokes(x) \Rightarrow Cancer(x)$	1.5
$\forall x, y \; Smokes(x) \wedge Friends(x, y) \Rightarrow Smokes(y)$	1.1

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

World ω:
S(A),¬C(A), F(A,A), ¬ F(A,B), F(B,A), F(B,B), ¬ S(B), ¬ C(B)]

$Smokes(A) \Longrightarrow Cancer(A), \exp(1.5)$

$\boxed{Smokes(B) \Longrightarrow Cancer(B), \exp(1.5)}$

$\boxed{Smokes(A) \wedge Friends(A, A) \Longrightarrow Smokes(A), \exp(1.1)}$

$\boxed{Smokes(A) \wedge Friends(A, B) \Longrightarrow Smokes(B), \exp(1.1)}$

$\boxed{Smokes(B) \wedge Friends(B, A) \Longrightarrow Smokes(A), \exp(1.1)}$

$\boxed{Smokes(B) \wedge Friends(B, B) \Longrightarrow Smokes(B), \exp(1.1)}$

$\mathbf{n_1 = 1}$
$\mathbf{n_2 = 4}$

Probability of ω is proportional to the product of exponentiated weights of satisfied ground formulas

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

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

Marginal/Counting Inference

Probabilistic Theorem Proving problem

Given Probabilistic knowledge base *K*

Query formula *Q*

Output $P(Q|K)$

Compare to:

Logical Theorem proving

Given Knowledge base *K*

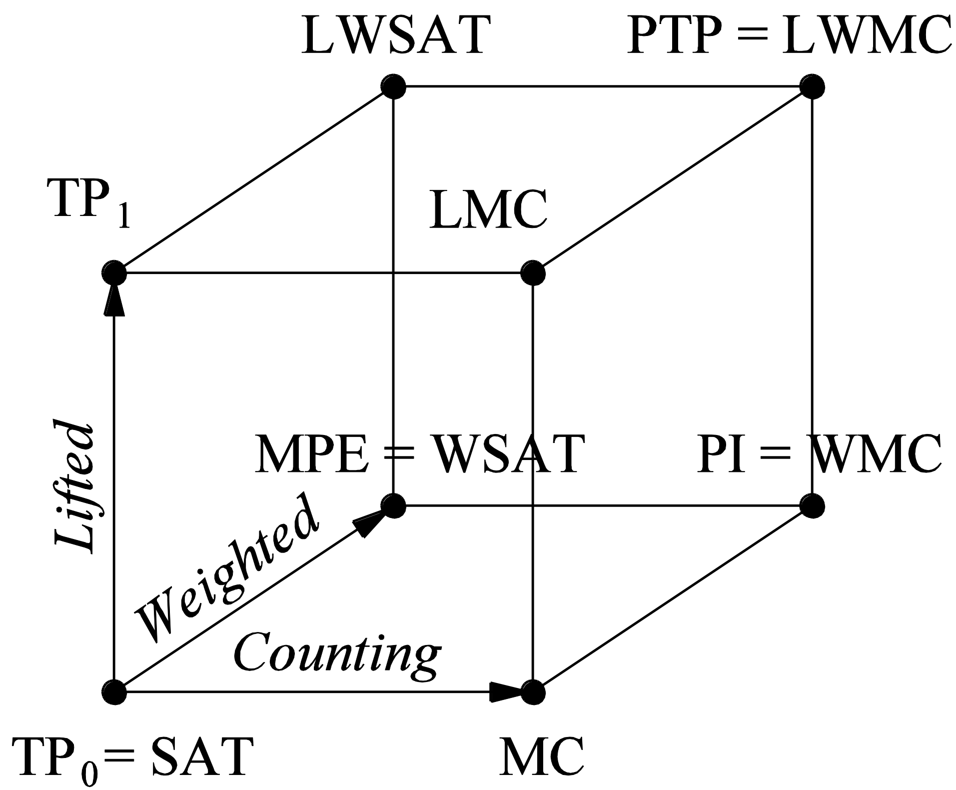
Query formula *Q*

Output: Does K entail Q

Lifted Weighted Model Counting

- ModelCount(CNF) = # worlds that satisfy CNF
- Assign a weight to each literal
- Weight(world) = product of literals that are true in the world
- Weighted model counting:
  - Sum of weights of all world that satisfy CNF
- Lifted Weighted model counting:
  - Each literal is first-order literal

Inference Problems



PTP is reducible to LWMC

Weighted Model Counting

WMC(*CNF, weights*)

if all clauses in *CNF* are satisfied

return $\prod_{A \in \mathrm{A}(CNF)} (w_A + w_{\neg A})$

if *CNF* has empty unsatisfied clause **return** 0

Base Case

Weighted Model Counting

WMC(*CNF, 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 $\prod_{i=1}^{k} WMC(C_i, weights)$

Decomp. Step

Weighted Model Counting

WMC(*CNF, weights*)

if all clauses in *CNF* are satisfied

return $\prod_{A \in \mathrm{A}(CNF)} (w_A + w_{\neg A})$

if *CNF* has empty unsatisfied clause **return** 0

if *CNF* can be partitioned into CNFs $C_1, \ldots, C_k$ sharing no atoms

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

choose an atom *A*

return $w_A WMC(CNF \mid A, weights) + w_{\neg A} WMC(CNF \mid \neg A, weights)$

Splitting Step

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
  Predicate$_i$(variables in F_i, 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

Logical/First-order Structure

- Exploit Symmetry in the first-order representation

$R(x) \vee S(x), v$

Independent $\left\{ \begin{array}{l} R(A) \vee S(A), v \\ R(B) \vee S(B), v \\ R(C) \vee S(C), v \\ R(D) \vee S(D), v \end{array} \right.$

$$Z = \prod_{X \in \{A,B,C,D\}} Z[x\backslash X]$$

Linear time

$R(x) \vee S(x), v$

Independent And Indentical $\left\{ \begin{array}{l} R(A) \vee S(A), v \\ R(B) \vee S(B), v \\ R(C) \vee S(C), v \\ R(D) \vee S(D), v \end{array} \right.$

$$Z = (Z[x\backslash X])^4$$

Constant time

Lifted/First-order Structure: POWER RULE

- Of course, you cannot always take powers and solve it efficiently
- Following conditions must be satisfied for a variable x:
  - "x" must appear in every predicate symbol in the formula
  - If there is another unifiable variable "y", then "x" and "y" must appear in the same position in every predicate in every formula
- MLN: R(x,y) v S(x,z) and R(y,z) v T(y,u)
  - $Z=[Z[x/A, y/A]]^n$
- MLN: R(x,y) v S(x,z) and R(z,y) v T(y,u)
  - cannot apply.

Lifted/First-order Structure: BINOMIAL RULE

- Applies to singleton atoms
  - Condition on singleton atoms in a special way
- MLN: (f=R(x) v S(x,y) v T(y), v)
  - If domain-size of x is "n", naïve conditioning on R(x) yields 2^n truth-assignments
- BINOMIAL RULE: Condition on (n+1)-truth assignments

$$Z(f, v) = \sum_{i=0}^{n} \binom{n}{i} Z(f_{R,i}, v)$$

$f_{R,i}$ *is obtained from* f *by setting exactly "i" groundings of* R *to True*

Lifted Weighted Model Counting

LWMC(*CNF, substs, weights*)

if all clauses in *CNF* are satisfied

return $\prod_{A \in \mathrm{A}(CNF)} (w_A + w_{\neg A})^{n_A(substs)}$

if *CNF* has empty unsatisfied clause **return** 0

Base Case

Lifted Weighted Model Counting

LWMC(*CNF, substs, weights*)

if all clauses in *CNF* are satisfied

return $\prod_{A \in \mathrm{A}(CNF)} (w_A + w_{\neg A})^{n_A(substs)}$

if *CNF* has empty unsatisfied clause **return** 0

if there exists a lifted decomposition of *CNF*

return $\prod_{i=1}^{k} [LWMC(CNF_{i,1}, substs, weights)]^{m_i}$

Decomp.
Step.
Power
Rule

Lifted Weighted Model Counting

LWMC(*CNF, substs, weights*)

if all clauses in *CNF* are satisfied

return $\prod_{A \in \mathrm{A}(CNF)} (w_A + w_{\neg A})^{n_A(substs)}$

if *CNF* has empty unsatisfied clause **return** 0

if there exists a lifted decomposition of *CNF*

return $\prod_{i=1}^{k} [LWMC(CNF_{i,1}, substs, weights)]^{m_i}$

choose an atom *A*

return $\sum_{i=1}^{l} n_i w_A^{t_i} w_{\neg A}^{f_i} LWMC(CNF \mid \sigma_j, substs_j, weights)$

Splitting Step Binomial Rule

Approximate Inference

WMC(*CNF, weights*)

if all clauses in *CNF* are satisfied

return $\prod_{A \in \mathrm{A}(CNF)} (w_A + w_{\neg A})$

if *CNF* has empty unsatisfied clause **return** 0

if *CNF* can be partitioned into CNFs $C_1, \ldots, C_k$ sharing no atoms

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

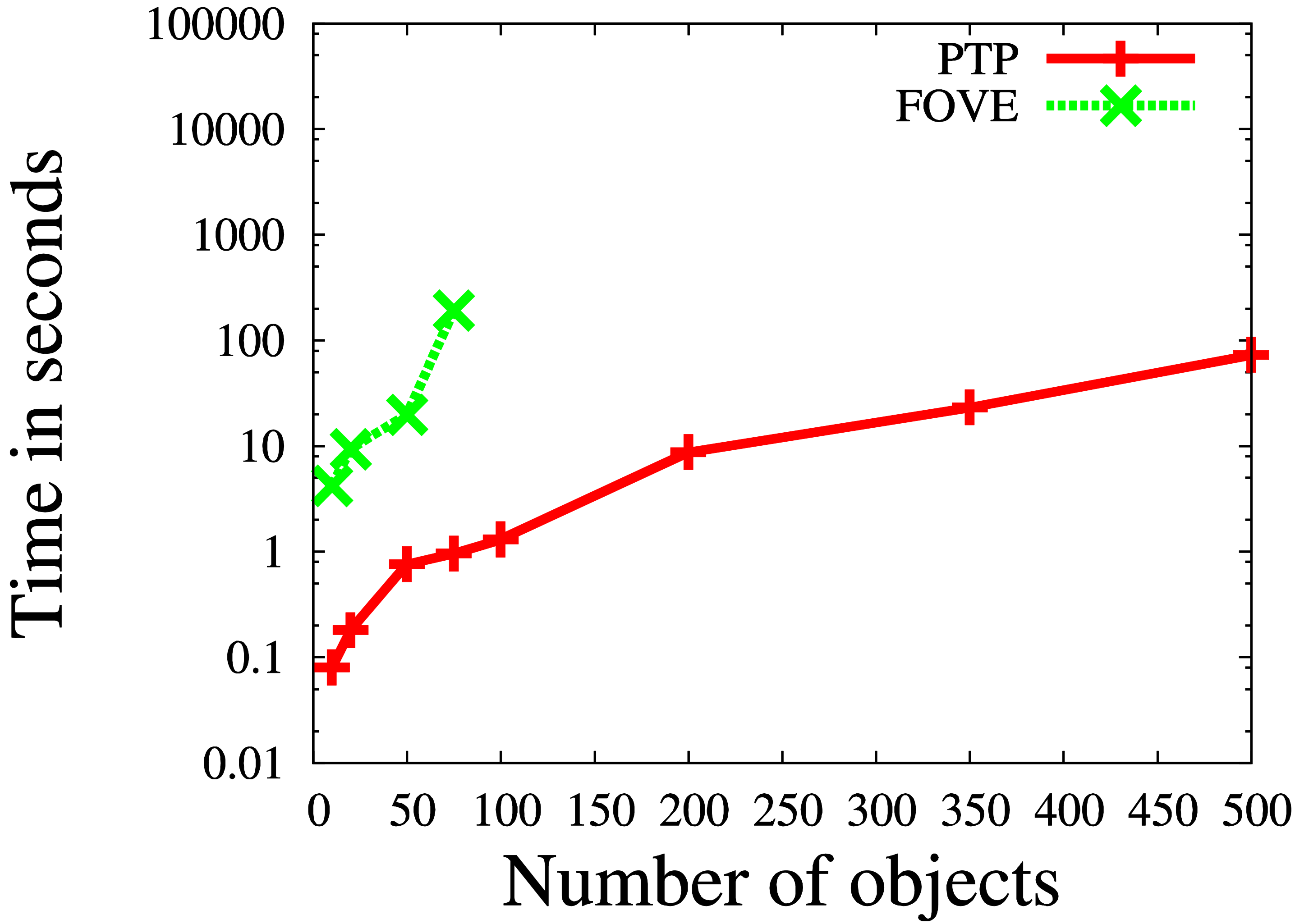
choose an atom *A*

return $\frac{w_A}{Q(A \mid CNF, weights)} WMC(CNF \mid A, weights)$

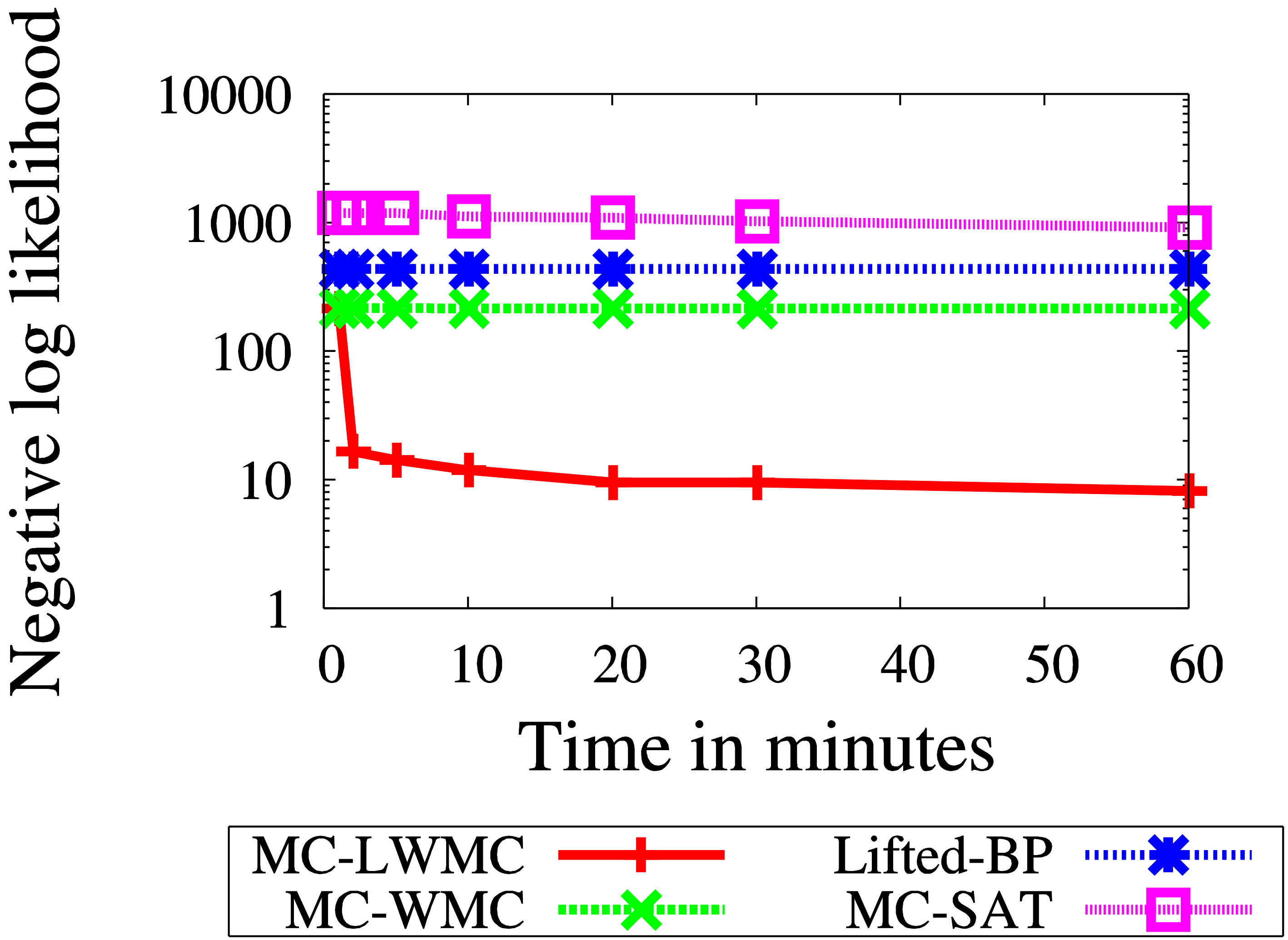
with probability $Q(A \mid CNF, weights)$, etc.

Approximate Splitting Step

Link Prediction



Coreference (Cora)



MAP/MPE Inference

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

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

Query

Evidence

MAP/MPE Inference

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

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

MAP/MPE Inference

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

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

MAP/MPE Inference

- **Problem:** Find most likely state of world given evidence
$$\max_y \sum_i w_i n_i(x, y)$$

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

The MaxWalkSAT Algorithm

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

But … Memory Explosion

- **Problem:**
  If there are **n** constants and **k** distinct logical variables in each formula, we get O($\mathbf{n}^k$) ground formulas
- **Solution:**
  Exploit sparseness; ground clauses lazily
  – LazySAT algorithm [Singla & Domingos, 2006]
  – Fast WALKSAT by grounding to monadic first-order logic (In progress)
  – Lifted MPE (in progress)

Learning

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

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 Lifted sampling or MaxWalkSAT for inference

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 search, shortest-first search

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**
- **http://www.hlt.utdallas.edu/~vgogate/software**

	Alchemy	Prolog	BUGS
Represent-ation	F.O. Logic + Markov nets	Horn clauses	Bayes nets
Inference	Probabilistic Theorem proving	Theorem proving	Gibbs sampling
Learning	Parameters & structure	No	Params.
Uncertainty	Yes	No	Yes
Relational	Yes	Yes	No

Applications

- Statistical parsing
- Semantic processing
- Bayesian networks
- Relational models
- Robot mapping
- Planning and MDPs
- Practical tips
- Basics
- Logistic regression
- Hypertext classification
- Information retrieval
- Entity resolution
- Hidden Markov models
- Information extraction

Running Alchemy

- Programs
  - Infer
  - Learnwts
  - Learnstruct
  - LiftedInfer
- Options
- MLN file
  - Types (optional)
  - Predicates
  - Formulas
- Database files

Uniform Distribn.: Empty MLN

Example: Unbiased coin flips

Type: `flip = { 1, … , 20 }`

Predicate: `Heads(flip)`

$$P(Heads(f)) = \frac{\frac{1}{Z}e^{0}}{\frac{1}{Z}e^{0} + \frac{1}{Z}e^{0}} = \frac{1}{2}$$

Binomial Distribn.: Unit Clause

Example: Biased coin flips

Type: `flip = { 1, … , 20 }`

Predicate: `Heads(flip)`

Formula: `Heads(f)`

Weight: Log odds of heads: $w = \log\left(\frac{p}{1-p}\right)$

$$P(\text{Heads(f)}) = \frac{\frac{1}{Z}e^{w}}{\frac{1}{Z}e^{w} + \frac{1}{Z}e^{0}} = \frac{1}{1+e^{-w}} = p$$

By default, MLN includes unit clauses for all predicates

(captures marginal distributions, etc.)

Multinomial Distribution

Example: Throwing die

Types: `throw = { 1, … , 20 }`
`face = { 1, … , 6 }`

Predicate: `Outcome(throw,face)`

Formulas: `Outcome(t,f) ^ f != f' => !Outcome(t,f').`
`Exist f Outcome(t,f).`

Too cumbersome!

Multinomial Distrib.: ! Notation

Example: Throwing die

Types: `throw = { 1, … , 20 }`
`face = { 1, … , 6 }`

Predicate: `Outcome(throw,face!)`

Formulas:

Semantics: Arguments without "!" determine arguments with "!".

Also makes inference more efficient (triggers blocking).

Multinomial Distrib.: + Notation

Example: Throwing biased die

Types: `throw = { 1, … , 20 }`

`face = { 1, … , 6 }`

Predicate: `Outcome(throw,face!)`

Formulas: `Outcome(t,+f)`

Semantics: Learn weight for each grounding of args with "+".

Logistic Regression

Logistic regression: $$\log\left(\frac{P(C=1 \mid \mathbf{F}=\mathbf{f})}{P(C=0 \mid \mathbf{F}=\mathbf{f})}\right)=a+\sum b_i f_i$$

Type: `obj = { 1, ... , n }`
Query predicate: `C(obj)`
Evidence predicates: F_i`(obj)`
Formulas: `a   C(x)`
b_i F_i`(x) ^ C(x)`

Resulting distribution: $$P(C=c,\mathbf{F}=\mathbf{f})=\frac{1}{Z}\exp\left(ac+\sum_i b_i f_i c\right)$$

Therefore: $$\log\left(\frac{P(C=1 \mid \mathbf{F}=\mathbf{f})}{P(C=0 \mid \mathbf{F}=\mathbf{f})}\right)=\log\left(\frac{\exp\left(a+\sum b_i f_i\right)}{\exp(0)}\right)=a+\sum b_i f_i$$

Text Classification

```
page = { 1, … , n }
word = { … }
topic = { … }

Topic(page,topic!)
HasWord(page,word)

!Topic(p,t)
HasWord(p,+w) => Topic(p,+t)
```

For all w, t pairs we will learn a weight
Which denotes how indicative of a topic a particular word is

Hypertext Classification

```
Topic(page,topic!)
HasWord(page,word)
Links(page,page)

HasWord(p,+w) => Topic(p,+t)
Topic(p,t) ^ Links(p,p') => Topic(p',t)
```

Use hyperlinks to help classify text

Cf. S. Chakrabarti, B. Dom & P. Indyk, "Hypertext Classification Using Hyperlinks," in *Proc. SIGMOD-1998*.

Information Retrieval

```
InQuery(word)// Suppose word is in our search query
HasWord(page,word)
Relevant(page)

InQuery(+w) ^ HasWord(p,+w) => Relevant(p)
Relevant(p) ^ Links(p,p') => Relevant(p')
```

Cf. L. Page, S. Brin, R. Motwani & T. Winograd, "The PageRank Citation Ranking: Bringing Order to the Web," Tech. Rept., Stanford University, 1998.

Entity Resolution

Problem: Given database, find duplicate records

```
HasToken(token,field,record)
SameField(field,record,record)
SameRecord(record,record)

HasToken(+t,+f,r) ^ HasToken(+t,+f,r')
   => SameField(f,r,r')
SameField(+f,r,r') => SameRecord(r,r')
SameRecord(r,r') ^ SameRecord(r',r")
   => SameRecord(r,r")
```

Cf. A. McCallum & B. Wellner, "Conditional Models of Identity Uncertainty with Application to Noun Coreference," in *Adv. NIPS 17*, 2005.

Entity Resolution

Can also resolve fields:

```
HasToken(token,field,record)
SameField(field,record,record)
SameRecord(record,record)

HasToken(+t,+f,r) ^ HasToken(+t,+f,r’)
   => SameField(f,r,r’)
SameField(f,r,r’) <=> SameRecord(r,r’)
SameRecord(r,r’) ^ SameRecord(r’,r”)
   => SameRecord(r,r”)
SameField(f,r,r’) ^ SameField(f,r’,r”)
   => SameField(f,r,r”)
```

More: P. Singla & P. Domingos, “Entity Resolution with Markov Logic”, in *Proc. ICDM-2006*.

Hidden Markov Models

```
obs = { Obs1, … , ObsN }
state = { St1, … , StM }
time = { 0, … , T }

State(state!,time)
Obs(obs!,time)

State(+s,0)
State(+s,t) => State(+s',t+1)
Obs(+o,t) => State(+s,t)
```

Practical Tips

- Add all unit clauses (the default)
- Implications vs. conjunctions
- Open/closed world assumptions
- How to handle uncertain data:
  `R(x,y) => R'(x,y)` (the "HMM trick")
- Controlling complexity
  - Low clause arities
  - Low numbers of constants
  - Short inference chains
- Use the simplest MLN that works
- Cycle: Add/delete formulas, learn and test