CS839:
Probabilistic Graphical Models

Lecture 17: Markov Logic Networks

Theo Rekatsinas

Slides by Pedro Domingos

Project Deliverables

- Proposal due: Nov 8
- Mid-report due: Nov 27
- Proposal presentations: Dec 11

Logical and Statistical AI

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

We Need to Unify the Two

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

Goal and Progress

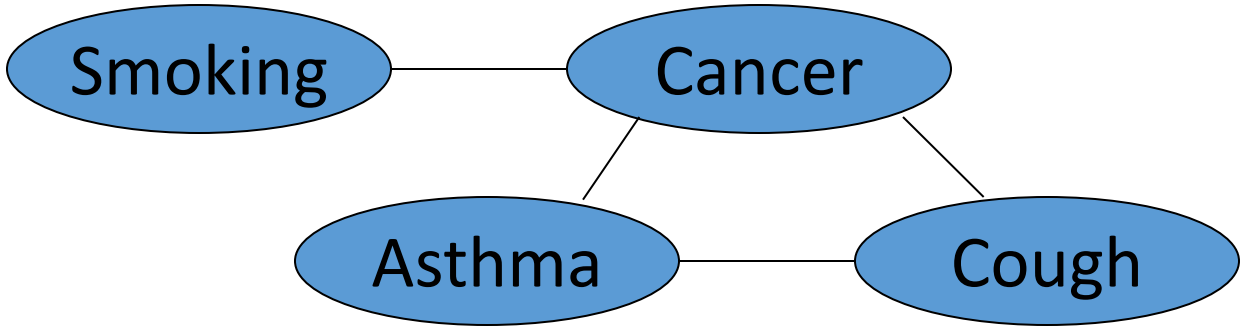
- **Goal:**
  Make statistical relational AI as easy as purely statistical or purely logical AI
- Progress to date
  - Was a very hot area of research
  - After the boom of DL it is fading away with few exceptions

Overview

- Motivation
- **Foundational areas**
  - **Probabilistic inference**
  - Statistical learning
  - Logical inference
  - Inductive logic programming
- Putting the pieces together
- Applications

Markov Networks

- **Undirected** graphical models



- Potential functions defined over cliques

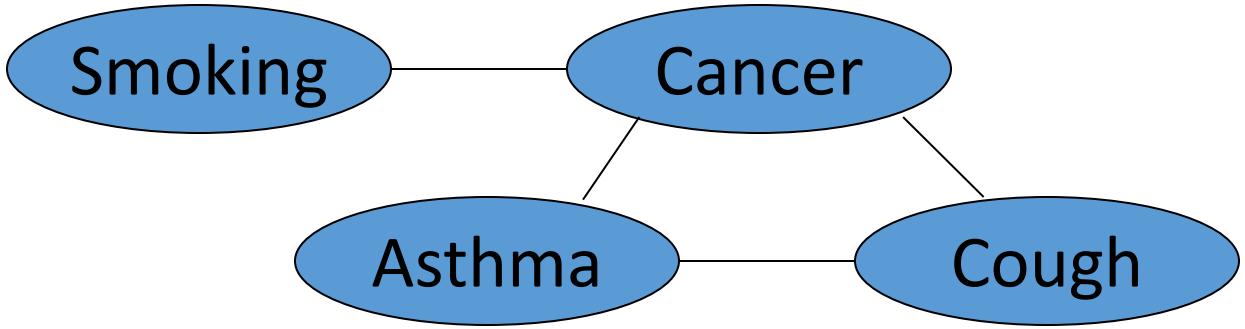
$$P(x) = \frac{1}{Z}\prod_{c}\Phi_c(x_c)$$

$$Z = \sum_{x}\prod_{c}\Phi_c(x_c)$$

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:

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

Weight of Feature *i*

Feature *i*

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

$$w_1 = 1.5$$

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 = ? **global**	Z = 1 **local**
Indep. check	Graph separation	D-separation
Indep. props.	Some	Some
Inference	MCMC, BP, etc.	Convert to Markov

Inference in Markov Networks

- **Goal**: Compute marginals & conditionals of

$$P(X) = \frac{1}{Z}\exp\left(\sum_i w_i f_i(X)\right) \qquad Z = \sum_X \exp\left(\sum_i w_i f_i(X)\right)$$

- Exact inference is #P-complete
- Conditioning on Markov blanket is easy:

$$P(x \mid MB(x)) = \frac{\exp\left(\sum_i w_i f_i(x)\right)}{\exp\left(\sum_i w_i f_i(x=0)\right) + \exp\left(\sum_i w_i f_i(x=1)\right)}$$

- Gibbs sampling exploits this

MCMC: Gibbs Sampling

state ← random truth assignment
for *i* ← 1 **to** *num-samples* **do**
    for each variable *x*
        sample *x* according to P(*x*|*neighbors*(*x*))
        state ← *state* with new value of *x*
P(*F*) ← fraction of states in which *F* is true

Other Inference Methods

- Many variations of MCMC
- Belief propagation (sum-product)
- Variational approximation
- Exact methods

Overview

- Motivation
- **Foundational areas**
  - Probabilistic inference
  - **Statistical learning**
  - Logical inference
  - Inductive logic programming
- Putting the pieces together
- Applications

Learning Markov Networks

- Learning parameters (weights)
  - Generatively
  - Discriminatively
- Learning structure (features)
- In this tutorial: Assume complete data
  (If not: EM versions of algorithms)

Generative Weight Learning

- Maximize likelihood or posterior probability
- Numerical optimization (gradient or 2nd order)
- No local maxima

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

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

Expected no. times feature *i* is true according to model

- Requires inference at each step (slow!)

Pseudo-Likelihood

$$PL(x) \equiv \prod_i P(x_i \mid neighbors(x_i))$$

- Likelihood of each variable given its neighbors in the data
- 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

[Which can lead to disasterous results]

Discriminative Weight Learning

- Maximize conditional likelihood of query (y) given evidence (x)

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

No. of true groundings of clause *i* in data

Expected no. true groundings according to model

- Approximate expected counts by counts in MAP state of y given x

Structure Learning

- How to learn the structure of a Markov network?
  - … not too different from learning structure for a Bayes network: discrete search through space of possible graphs, trying to maximize data probability….

Overview

- Motivation
- Foundational areas
  - Probabilistic inference
  - Statistical learning
  - **Logical inference**
  - Inductive logic programming
- Putting the pieces together
- Applications

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

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)

Overview

- Motivation
- Foundational areas
  - Probabilistic inference
  - Statistical learning
  - Logical inference
  - **Inductive logic programming**
- Putting the pieces together
- Applications

Rule Induction

- **Given:** Set of positive and negative examples of some concept
  - **Example:** $(x_1, x_2, \ldots, x_n, y)$
  - *y:* **concept** (Boolean)
  - $x_1, x_2, \ldots, x_n$*:* **attributes** (assume Boolean)
- **Goal:** Induce a set of rules that cover all positive examples and no negative ones
  - **Rule:** $x_a \wedge x_b \wedge \ldots \Rightarrow y$ (x_a: Literal, i.e., x_i or its negation)
  - Same as **Horn clause**: *Body* $\Rightarrow$ *Head*
  - Rule *r* **covers** example *x* iff *x* satisfies body of *r*
- ***Eval(r):*** Accuracy, info. gain, coverage, support, etc.

Learning a Single Rule

head ← *y*
body ← Ø
repeat
    for each literal *x*
        r_x ← *r* with *x* added to *body*
        Eval(r_x)
    body ← *body* ^ best *x*
until no *x* improves *Eval(r)*
return *r*

[For Eval(r): something like a one-sided version of information gain works pretty well – see Quinlan's FOIL-W]

Learning a Set of Rules

$R \leftarrow \emptyset$
$S \leftarrow$ *examples*
repeat
    learn a single rule *r*
    $R \leftarrow R \cup \{ r \}$
    $S \leftarrow S -$ positive examples covered by r
until S contains no positive examples
r**eturn** *R*

First-Order Rule Induction

- y and x_i are now predicates with arguments
  E.g.: y is Ancestor(x,y), x_i is Parent(x,y)
- Literals to add are predicates or their negations
- Literal to add must include at least one variable already appearing in rule
- Adding a literal changes # groundings of rule
  E.g.: Ancestor(x,z) ^ Parent(z,y) $\Rightarrow$ Ancestor(x,y)
- *Eval(r)* must take this into account
  E.g.: Multiply by # positive groundings of rule still covered after adding literal

[Issues in learning first-order rules]

- First-order rules can be expensive to evaluate
  - Circuit(x,n) ← Edge(x,z1),Edge(z1,z2),…,Edge(zn,x), z1!=z2,z1!=z2,…,z1!=zn,z2!=z3,…,z{n-1}!=zn.
- First-order theories can have long proofs
  - Eg, Ackerman's function
- First-order rules can be very expressive
  - F(a,b,c,d,y) ← Nand(a,b,z1),Nor(c,d,z2),Xor(z2,z2,z3),Not(z3,y)

Overview

- Motivation
- Foundational areas
  - Probabilistic inference
  - Statistical learning
  - Logical inference
  - Inductive logic programming
- **Putting the pieces together**
- Applications

[Combinations of first-order and statistical learning methods]

- Stochastic logic programs
  - Horn clause programs + probabilities
- Probabilistic relational models (PRMs)
  - Bayes networks defined by “frames”
- Relational Markov networks (PRMs)
  - Markov networks defined by SQL queries
- Bayesian logic (BLOG), Hierarchical Bayesian Compiler (HBC)
  - Bayes networks defined by special language
- **Markov logic networks**

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

Markov Logic

- Most developed approach to date
- Many other approaches can be viewed as special cases

- [Main focus of rest of this lecture]

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)

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

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

Example: Friends & Smokers

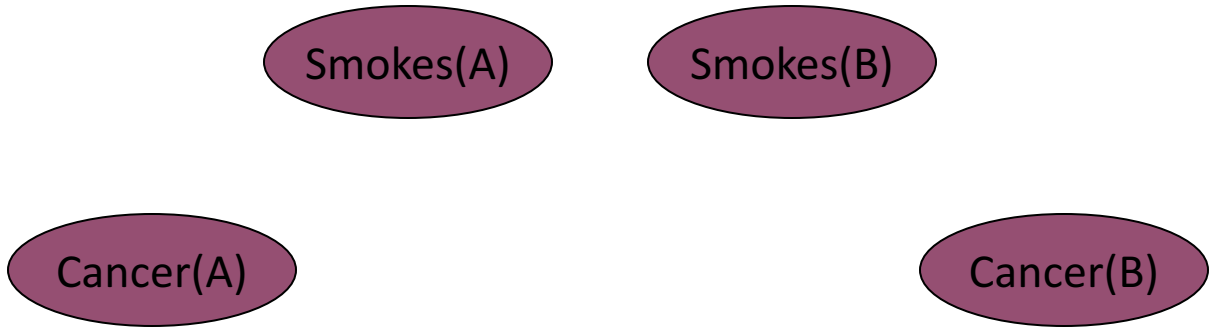
1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$

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

Example: Friends & Smokers

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

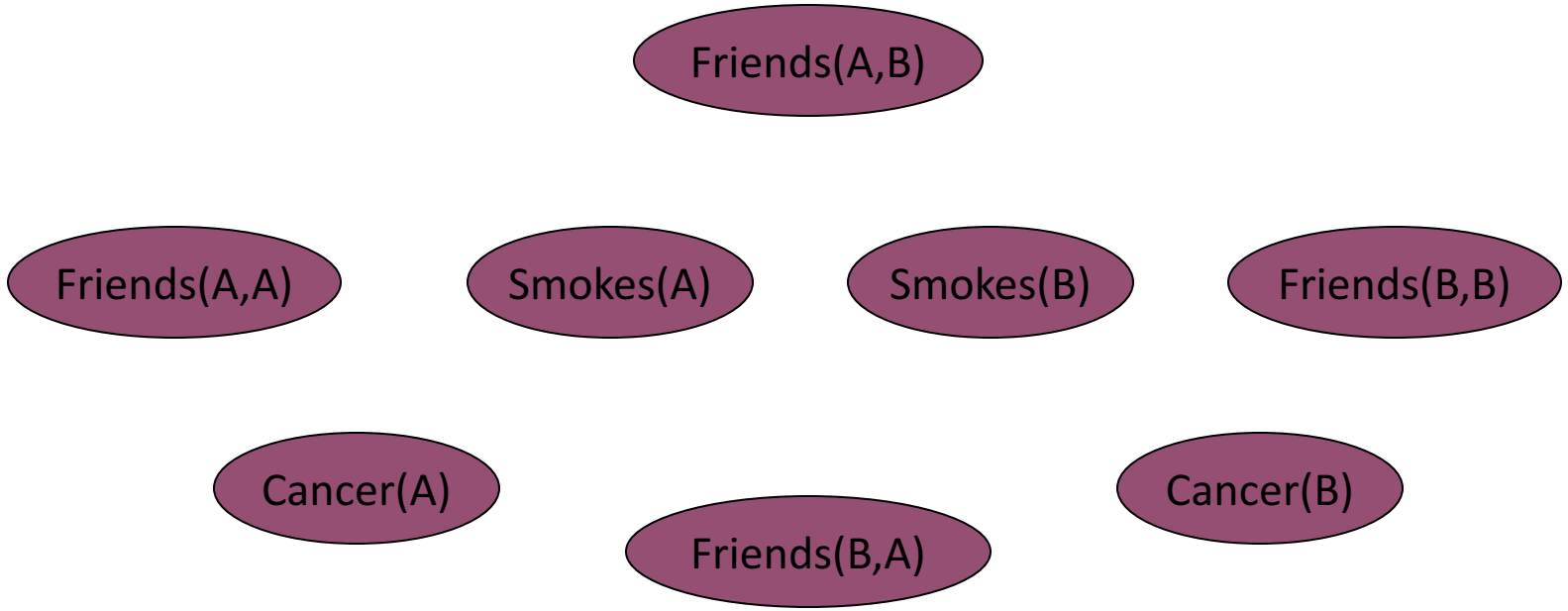
Two constants: **Anna** (A) and **Bob** (B)



Example: Friends & Smokers

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

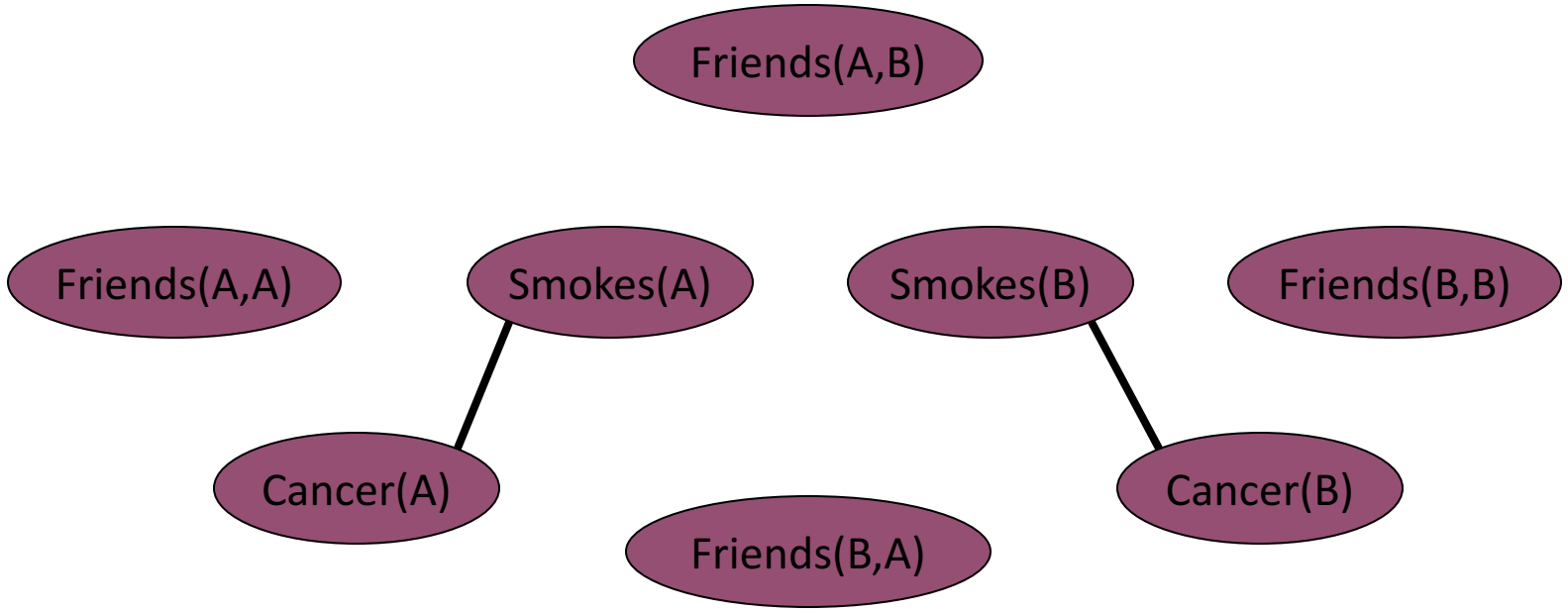
Two constants: **Anna** (A) and **Bob** (B)



Example: Friends & Smokers

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

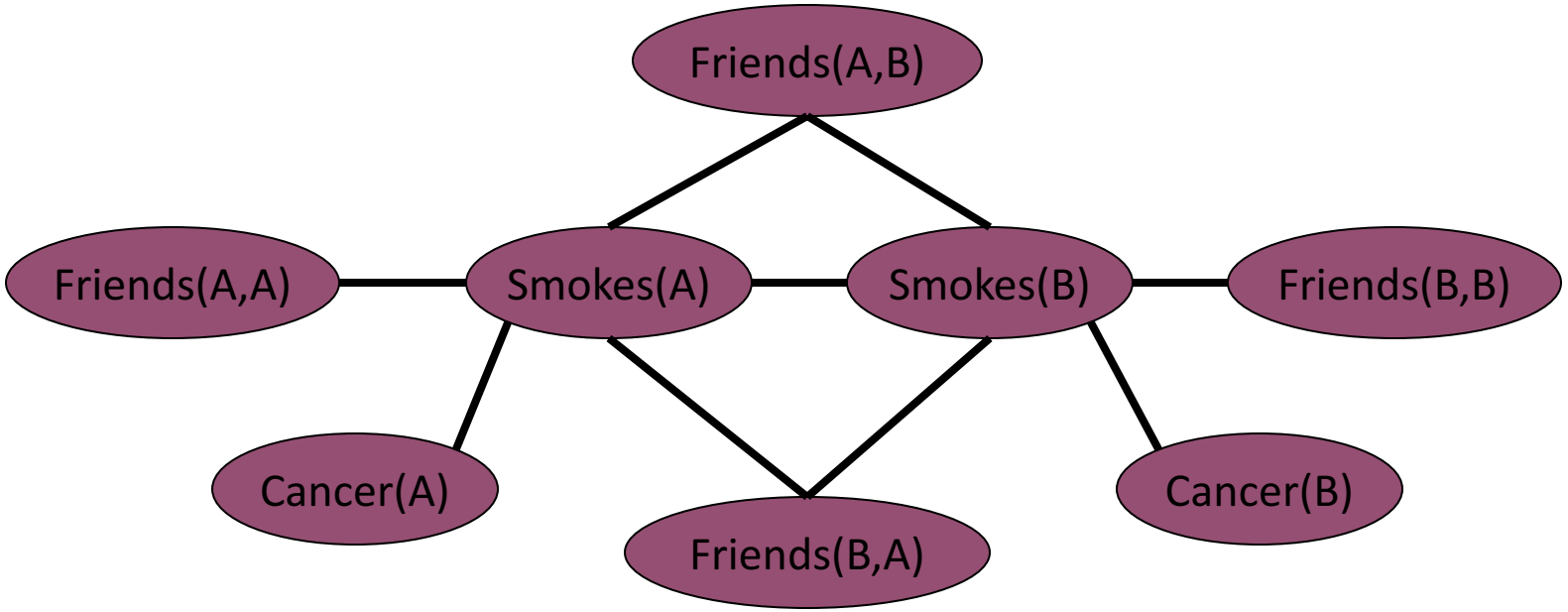
Two constants: **Anna** (A) and **Bob** (B)



Example: Friends & Smokers

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

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



Markov Logic Networks

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

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

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

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.)

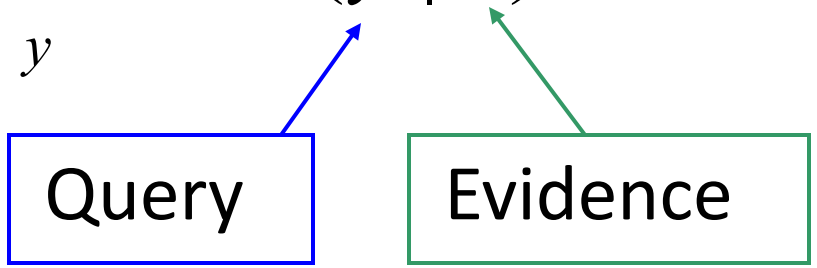
Relation to First-Order Logic

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

MAP/MPE Inference

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

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



MAP/MPE 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/MPE Inference

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

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

MAP/MPE 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])
- 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 the highest clause arity is **c**,
  the ground network requires **O(n^c)** memory
- **Solution:**
  Exploit sparseness; ground clauses lazily
  → LazySAT algorithm [Singla & Domingos, 2006]

Computing Probabilities

- P(Formula|MLN,C) = ?
- MCMC: Sample worlds, check formula holds
- P(Formula1|Formula2,MLN,C) = ?
- If Formula2 = 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

```
network ← Ø
queue ← query nodes
repeat
    node ← front(queue)
    remove node from queue
    add node to network
    if node not in evidence then
        add neighbors(node) to queue
until queue = Ø
```

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]

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

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

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 MC-SAT 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, shortest-first, bottom-up
  [Kok & Domingos, 2005; Mihalkova & Mooney, 2007]

Overview

- Motivation
- Foundational areas
  - Probabilistic inference
  - Statistical learning
  - Logical inference
  - Inductive logic programming
- Putting the pieces together
- **Applications**

Applications

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

Text Classification

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

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

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

Text Classification

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

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

Information Retrieval

```
InQuery(word)
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*.