Markov Logic Network

Matthew Richardson
and
Pedro Domingos

IE598 2016, Present by Hao Wu (haowu4)

Motivation - Unifying Logic and Probability

- Logic and probability are two most import way of reasoning.
- “Classic” AI favors logic approaches, which is mostly rule based.
  - Theorem proofing.
  - Cannot deal with uncertainty, very limited success.
- “Modern” AI approaches are dominated by more probabilistic methods, which handles the uncertainty and noise in real data.
  - Deep Learning, PGM and etc.
  - Huge success
- So why we still want to have Logic? (Why not learn everything?)

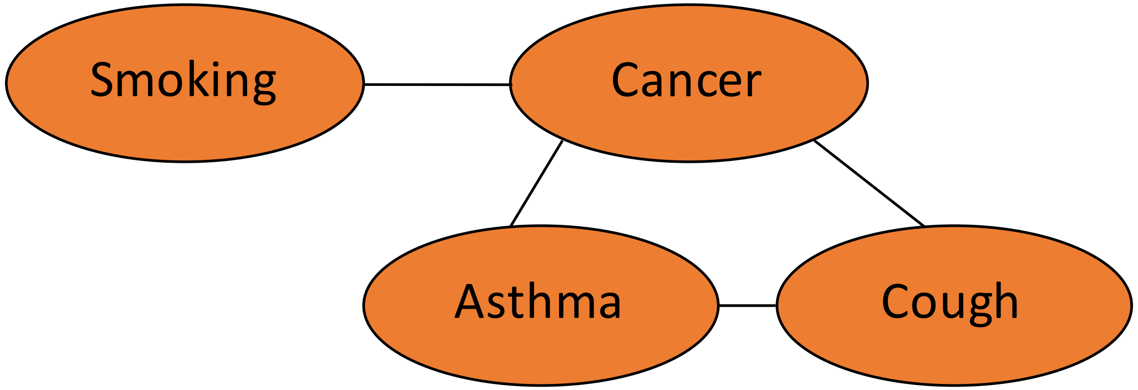
Why logic is still interesting

- Logic, especially, First-order logic provide a expressive, compact and elegant way to express knowledge.
  - It only take 30+ line to write down the rule of Sudoku in Prolog (and the same code can also solve it). How many data do you need to learn everything from scratch?
- We want a nice way to represent and solve our problems (efficiently).
  - Use expert knowledge to help the data driven system.
- Markov Logic is a way to connects Logic and Probability.
  - Logic handles complexity.
  - Probability handles uncertainty.

Background: Markov Network

- Potential functions defined over cliques

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



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

Syntax for First-Order Logic

Sentence → AtomicSentence
| Sentence Connective Sentence
| Quantifier Variable Sentence
| ¬Sentence
| (Sentence)

AtomicSentence → Predicate(Term, Term, ...)
| Term=Term

Term → Function(Term,Term,...)
| Constant
| Variable

Connective → ∨ | ∧ | ⇒ | ⇔

Quanitifier → ∃ | ∀

Constant → A | John | Car1

Variable → x | y | z |...

Predicate → Brother | Owns | ...

Function → father-of | plus | ...

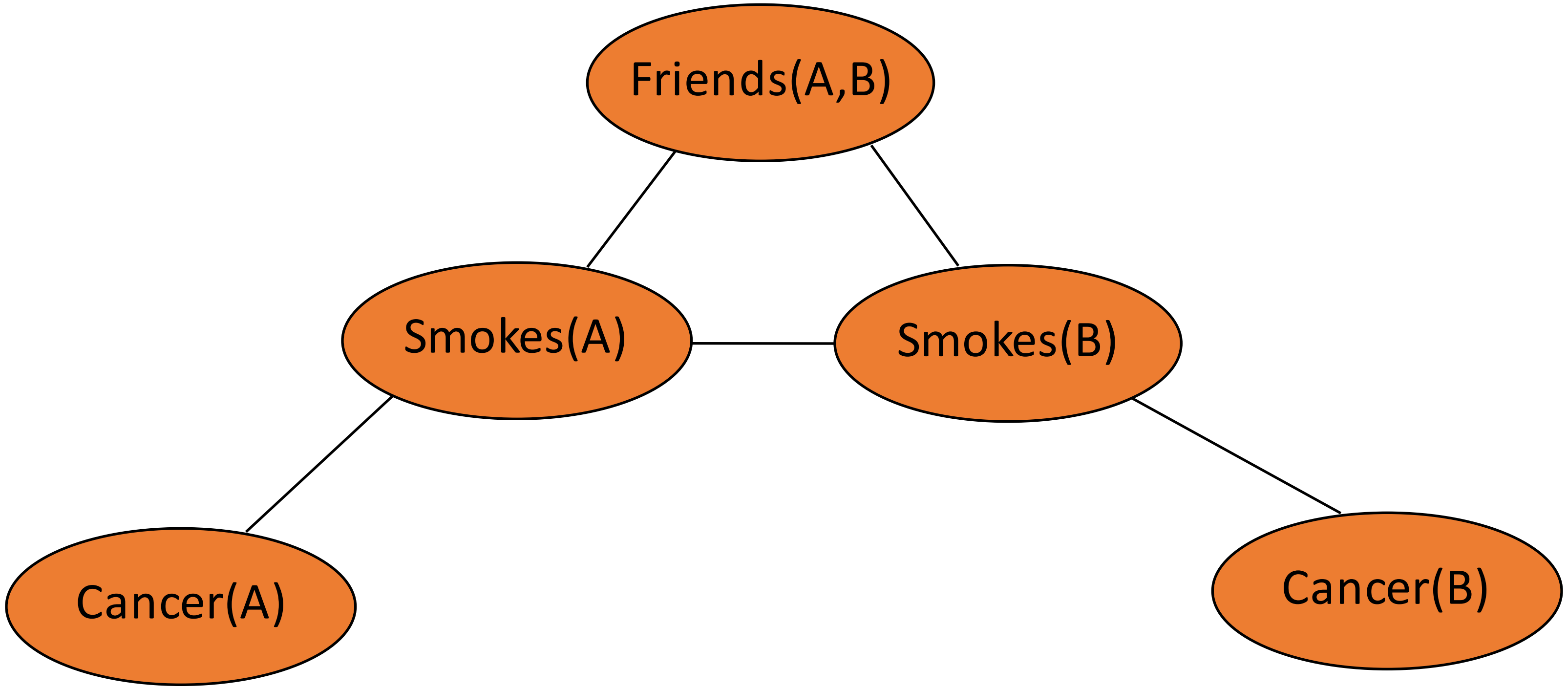
Comparision

FOL :

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

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

MRF:



Markov Logic Network

- 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

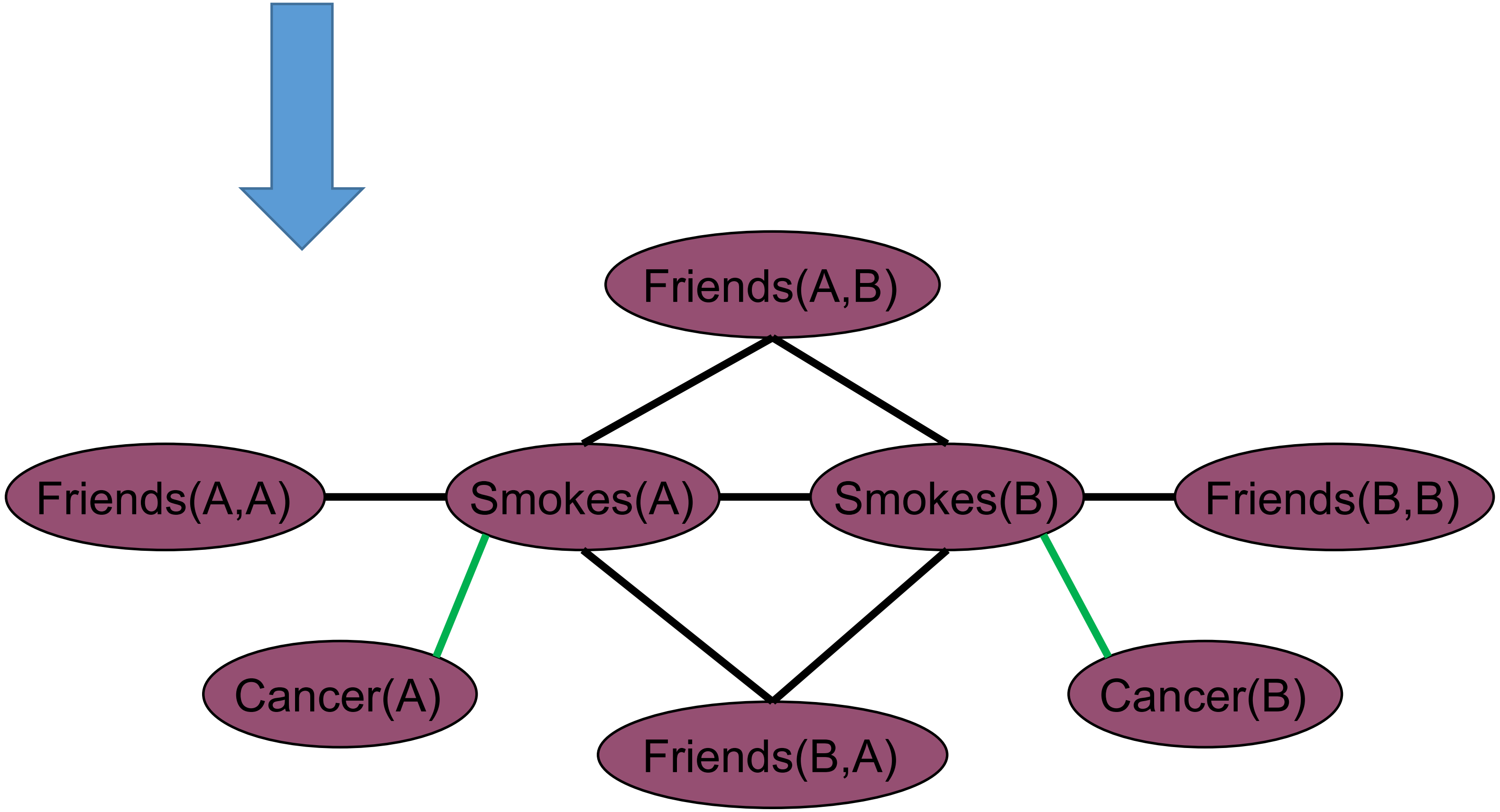
1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow \left(Smokes(x) \Leftrightarrow Smokes(y)\right)$

* And we need a database that contains constants for grounding.

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 Network: Definition

- Each ground formula defines a clique

$$P(X=x) = \frac{1}{Z}\exp\left(\sum_i w_i n_i(x)\right) = \frac{1}{Z}\prod_i \phi_i(x_{\{i\}})^{n_i(x)}$$

- $n_i(x)$ is the number of true grounding of formula i
- $x_{\{i\}}$ is the state (truth value) of atoms in formula i

$$\phi_i(x_{\{i\}}) = e^{w_i}$$

Markov Logic Networks

- A **template** for ground Markov Random Field.
- Can have type to reduce the number of predicate X constants.
  - i.e. Human can only be friend with another human.

- Expressivity:
  - When set all weight to infinite large, it becomes FOL.
  - *Every probability distribution over discrete or finite- precision numeric variables can be represented as a Markov logic network.*

Inference (Same as inference on MRF*)

*Sometime need a little twist for MCMC style inference

- MAP Inference:

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

- Conditional Inference

$$\begin{aligned} P(F_1|F_2, L, C) &= P(F_1|F_2, M_{L,C}) \\ &= \frac{P(F_1 \wedge F_2|M_{L,C})}{P(F_2|M_{L,C})} \\ &= \frac{\sum_{x \in \mathcal{X}_{F_1} \cap \mathcal{X}_{F_2}} P(X=x|M_{L,C})}{\sum_{x \in \mathcal{X}_{F_2}} P(X=x|M_{L,C})} \end{aligned}$$

Learning

- Learn from a database
- Can to learn both weights (parameters) and FOL formula(structure):
  - Learning weights.
    - By optimize likelihood.
  - Learning formula: (Inductive Logic Programming)
    - An ILP system will derive a hypothesised logic program which entails all the positive and none of the negative examples.
    - Use existing Inductive logic programming system.

Learning weight

- Optimize likelihood. (Generative approach)

$$f(w) = \log P(X = x) = \sum_i w_i n_i(x) - \log Z$$

$$Z = \sum_x \exp\left(\sum_i w_i n_i(x')\right)$$

  - Generalized too hard, do Pseudo-likelihood instead.
    - Counting true groundings of a first order clause in a KB is #P complete

$$\log PL(x) = \sum_l \log P(X_l = x_l \mid MB(x_l))$$

- Optimize conditional likelihood. (Discriminative approach)

$$f(w) = \log P(Y = y \mid X = x) = \sum_i w_i n_i(y, x) - \log Z_x$$

$$Z_x = \sum_{y'} \exp\left(\sum_i w_i n_i(y', x)\right)$$

Application - Entity resolution (Citation DB)

- Author(bib,author) Title(bib,title) Venue(bib,venue)
- HasWord(author,word)
- HasWord(title,word)
- HasWord(venue,word)
- SameAuthor(author1,author2)
- SameTitle(title1,title2)
- SameVenue(venue1,venue2)
- SameBib(bib1,bib2)

Application - Entity resolution

- Title(b1,t1) $\wedge$ Title(b2,t2) $\wedge$ HasWord(t1,+w) $\wedge$ HasWord(t2,+w) $\Rightarrow$ SameBib(b1,b2)
- Author(b1,a1) $\wedge$ Author(b2,a2) $\wedge$ SameBib(b1,b2) $\Rightarrow$ SameAuthor(a1,a2)
- Author(b1,a1) $\wedge$ Author(b2,a2) $\wedge$ SameAuthor(a1,a2) $\Rightarrow$ Samebib(b1,b2)