UiO : **University of Oslo**

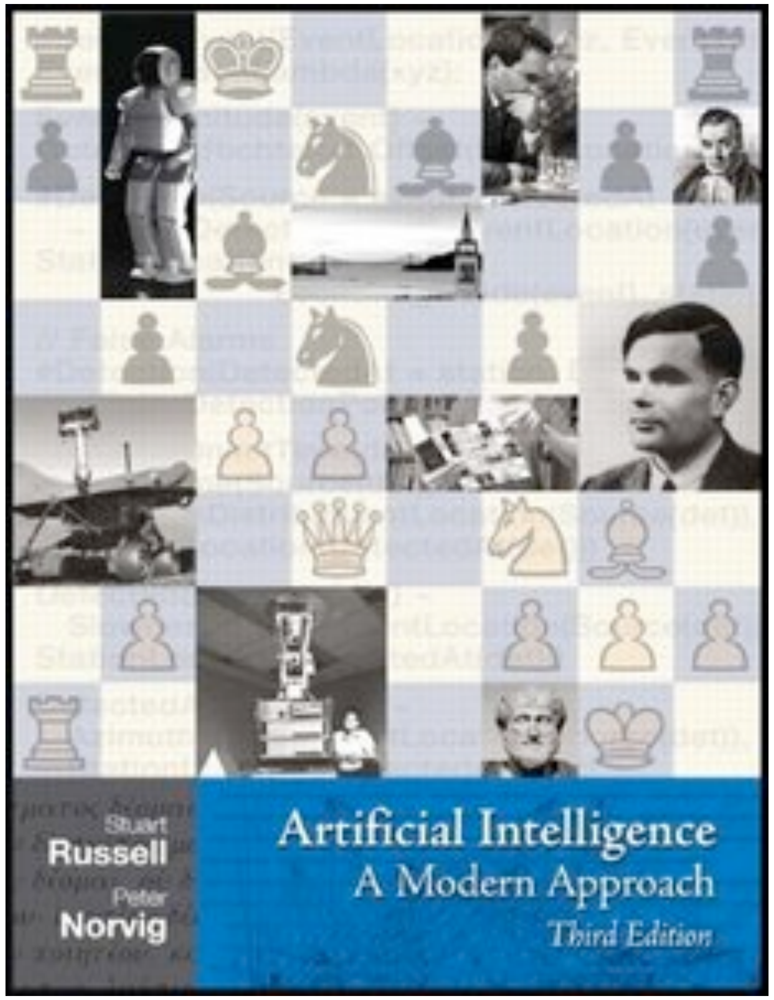
A short introduction to *Statistical Relational Learning*

Pierre Lison
University of Oslo, Dep. of Informatics

INF5830: Natural Language Processing
November 30, 2011

Introduction

- My favourite AI textbook: Russell & Norvig's AIMA
- Quick look at table of contents, *in terms of representations:*



chap 3-7: atomic
& factored models

chap 8-12+: relational
first-order models

chap 13-18+: probabilistic models

combination of the two?

Motivation

- Machine learning traditionally relies on fixed sets of features
  - Expressive power = propositional logic
- Problems for capturing many real-world domains
  - Need to encode *objects* and *relations* between them
  - Need to express *generic facts* about these objects
- Can we combine the expressive power of first-order logic with probability theory?

Domain example

- Assume you have a database of people, where for each person *p*, you know:
  - whether he/she smokes: *smokes(p)*
  - his/her group of friends: {*q* : *friends(p,q)*}
- You would like to determine for each person *p* the probability of *cancer(p)*
- Complex network of dependencies between friends, their smoking habits, and correlated cancer

Domain example (2)

- Assume we have some prior knowledge about our domain, which we can write as a set of first-order formulae:

$$\forall x : smokes(x) \Rightarrow cancer(x)$$
$$\forall x, y : friends(x, y) \wedge smokes(y) \Rightarrow cancer(x)$$
$$\forall x : \neg(\exists y : friends(x, y)) \Rightarrow smokes(x)$$
$$\forall x, y : friends(x, y) \Rightarrow (smokes(x) \Leftrightarrow smokes(y))$$
$$\forall x, y, z : (friends(x, y) \wedge friends(y, z) \wedge x \neq z) \Rightarrow friends(x, z)$$

- Problem: logic can only express hard constraints («all-or-nothing»)

Domain example (3)

- Alternatively, you could try to define a probabilistic model
- Solves the problem of soft correlations
- But a standard probabilistic model cannot capture generic constraints such as «friends of friends are also friends»
  - set of random variables is fixed and finite, and each variable has a fixed domain of alternative values

Statistical relational learning

- Statistical relational learning (SRL):
  - Research subfield within AI / machine learning
  - deals with domains which exhibit *both* uncertainty and a complex relational structure
- Other terms: first-order probabilistic models, relational probabilistic models, etc.
- Addresses issues of representation, inference, and learning

SRL for NLP

- Why is statistical relational learning important for NLP?
  - Because language is full of *relational structures*, and learning algorithms should be able to exploit them
  - Because statistical relational learning allows us to compactly incorporate our *prior domain knowledge*
  - Because SRL has recently achieved state-of-the-art results in important NLP tasks such as reference resolution, information extraction and semantic parsing

SRL approaches

- Many approaches and frameworks:
  - Bayesian Logic, Markov Logic Networks, Probabilistic Relational Models, Relational Bayesian Networks, etc.
  - Two main «routes»: extensions of first-order logic to handle probabilities, and extensions of probabilistic models to capture relational structures
- I will here focus on a particular model: *Markov Logic Networks* (MLNs)
  - Following slides based on previous presentations on MLNs by Pedro Domingos and Daniel Lowd

Markov Logic Networks

- Key idea: add *weights* to first-order formulae!
  - The weight expresses the *strength* of the formula
  - Infinite weight = hard constraint (cannot be violated)
- A Markov Logic Network is a set of pairs (F_i, w_i)
  - F_i is a first-order formula, and w_i is its weight
- Example:

$$\begin{array}{l|l} 8.2 & \forall x : smokes(x) \Rightarrow cancer(x) \\ 1.7 & \forall x, y : friends(x,y) \Rightarrow (smokes(x) \Leftrightarrow smokes(y)) \end{array}$$

Reasoning with MLNs

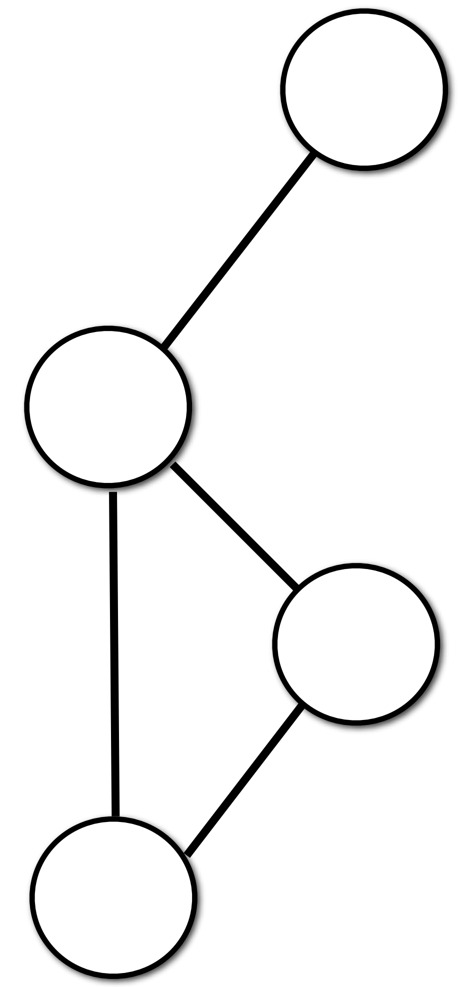
- MLNs are a nice knowledge representation formalism, but how can we use them for practical *inference* tasks?
  - I.e. how do we compute the probability of *cancer(Alice)*?
- A Markov Logic Network can be thought as a *template* for a (ground) Markov Network
  - Markov Network = undirected graphical model
  - Given a MLN and a set of constants (like *Alice* or *Robert*), we can directly generate an equivalent Markov Network
  - ... and use it for inference

Markov Network in a nutshell

- A Markov Network defines a joint probability distribution over a set of variables $X_1 \ldots X_n$
  - Network has a node for each variable
  - The nodes can be grouped into *cliques* (fully connected subgraph)
  - The joint distribution can then be factorised:

$$\Pr(X = x) = \frac{1}{Z} \prod_k \phi_k(x_{\{k\}})$$

where *k* is a clique and ϕ_k its potential function



Markov Network in a nutshell (2)

- Finally, the potential function ϕ_k is often rewritten as an exponentiated weighted sum over feature functions
- We can then rewrite the distribution as:

$$\Pr(X = x) = \frac{1}{Z} \exp(\sum_j w_j f_j(x))$$

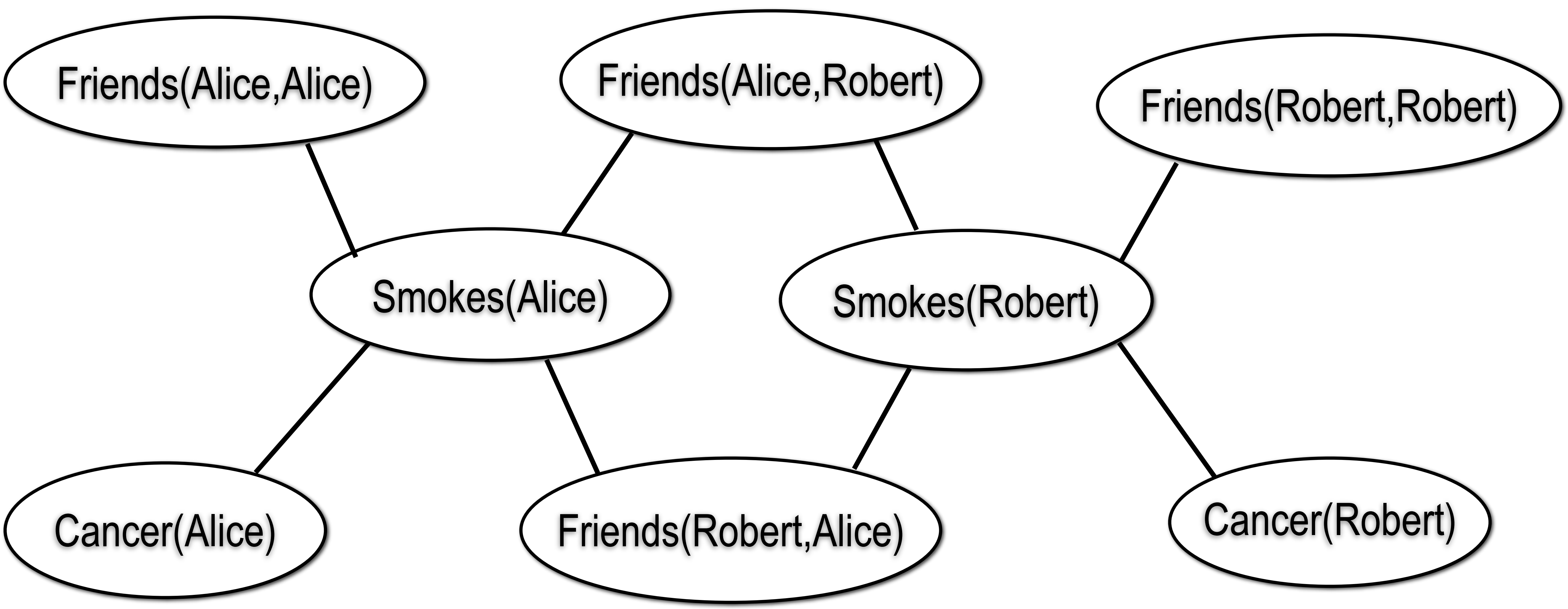
- This is called a *log-linear* model

Ground Markov Network

- Assume a Markov Logic Network *L* together with a set of constants $C=\{c_1 \ldots c_n\}$
- We can then construct the ground Markov network $M_{L,C}$ as follows:
  - For each predicate grounding over C, there is a node in $M_{L,C}$, with values true/false
  - For each formula F_i, there is a feature $f_i(x)$ for each possible grounding of F_i over C. The value of $f_i(x)$ is 1 if F_i is true in *x*, and false otherwise. The weight associated with $f_i(x)$ corresponds to the weight w_i of the formula

Construction example

- Two constants: *Alice* and *Robert*
- L = $\begin{array}{c|l} 8.2 & \forall x : smokes(x) \Rightarrow cancer(x) \\ 1.7 & \forall x, y : friends(x,y) \Rightarrow (smokes(x) \Leftrightarrow smokes(y)) \end{array}$



Probabilistic inference

- Once the ground Markov Network is constructed, it can be directly used for inference given some evidence
  - *But*: ground network can grow exponentially in the number of constants!
- Use approximate inference techniques to ensure the problem remains tractable
  - Monte Carlo sampling, (lifted) belief propagation, simulated tempering, weighted MaxSAT etc.

Learning in MLNs

- Can we learn MLNs from data?
  - *parameter learning*: assume the formulae are given, but not the weights
  - *structure learning*: try to learn both the formulae and the weights (much harder)
- Several learning algorithms available
  - both generative and discriminative models
  - Usually some variant of gradient descent on the weights

Applications of MLNs

- In Natural Language Processing:
  - Information extraction
  - Semantic parsing
  - Coreference resolution
- Outside NLP:
  - Social network analysis
  - Cognitive robotics
  - Bioinformatics

Live demonstration

- Live demonstration of *Alchemy*, an open source for inference and learning with Markov Logic Networks

Conclusions

- First conclusion: *smoking is bad for you*!
- We have also seen that Statistical Relational Learning allows us to capture domains which are both *complex* and *uncertain*
- Unification of logic and probability theory
- Various tools for efficient inference & learning
- Important applications for NLP