Introduction to *Statistical Relational Learning*

Pierre Lison,
Language Technology Group (LTG)
Department of Informatics

LT seminar
October 23 2012

Introduction

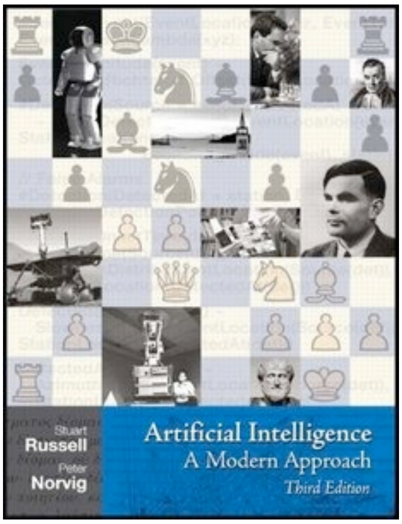
- Machine learning (ML) algorithms are now used in virtually any NLP system
- This talk will focus on the question of the *representation* used by these algorithms
- I'll describe a family of learning algorithms specifically designed for domains exhibiting *complex, relational structures*

Outline

- **Motivation**
- Markov Logic Networks
- Demonstration and examples
- Conclusion

Motivation

- 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: logic-based
reasoning

chap 13-18: probabilistic models

combination of the two?

Motivation

- The goal of ML algorithms is to learn a «good» function **F:I →O**
  - **I** is the space of possible inputs for the task
  - **O** is the space of possible outputs
- This function is learned from collected data, that can take various forms
  - Labelled or unlabelled data, reward signals, etc.
- What defines a «good» function depends on the performance metric we try to optimise

Motivation

- **Key question**: how do we represent the inputs and outputs of our problem?
- Typically, inputs are encoded as *feature vectors*, based on a fixed list of features
  - Both the features and their range are predefined
- Similarly, outputs are defined in terms of output classes or numerical range

Motivation

- Feature vectors are fine for many problems...
- But their expressivity is quite limited
  - Formal expressivity = propositional logic
- Many domains have a *relational structure* that cannot be easily encoded by feature vectors
  - Need to encode *objects* and *relations* between them
  - Need to express *generic facts* about these objects

Motivation

- Capturing relational structures requires a higher level of expressivity
- First-order logic gives us a *language* to describe such relational structures in a clear, concise fashion
  - *Terms* denote the entities of our domain
  - *Predicates* denote the attributes and relations of our domain, e.g. isGreen(x) or leftOf(x,y)

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

- Assume some prior domain knowledge written with 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

- Alternatively, you could try to estimate a classical statistical model
- Solves the problem of soft correlations
- But a standard statistical 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

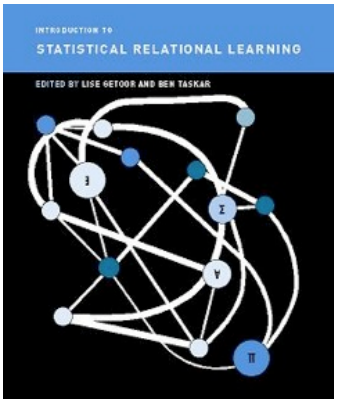
- Statistical relational learning (SRL):
  - Research subfield within AI / machine learning
  - deals with domains which exhibit *both* uncertainty and a complex relational structure
  - Alternative names: first-order probabilistic models, relational probabilistic models, etc.
  - Also related to *structured prediction* problems
  - 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 frameworks!
  - *Bayesian Logic, Markov Logic Networks, Probabilistic Relational Models, Relational Bayesian Networks, Stochastic Logic*, etc.
- Two main «strategies»:
  - Extensions of logic-based formalisms (e.g. inductive logic programming) to handle probabilities
  - Extensions of statistical models to capture relational structures



INTRODUCTION TO
STATISTICAL RELATIONAL LEARNING
EDITED BY LISE GETOOR AND BEN TASKAR

[L. Getoor and B. Taskar (2007). «*Introduction to Statistical Relational Learning*»]

SRL approaches

- I'll focus here on **Markov Logic Networks** (MLNs)
  - framework that combines first-order logic and (undirected) graphical models
  - Quite popular in NLP applications
  - Tries to «unify» many previous SRL approaches
  - Practical software toolkits are available

[M. Richardson and P. Domingos (2006). «Markov logic networks». *Machine Learning*]

[P. Domingos and Daniel Lowd (2009), «*Markov Logic: An Interface Layer for Artificial Intelligence*»]

Outline

- Motivation
- **Markov Logic Networks**
- Demonstration and examples
- Conclusion

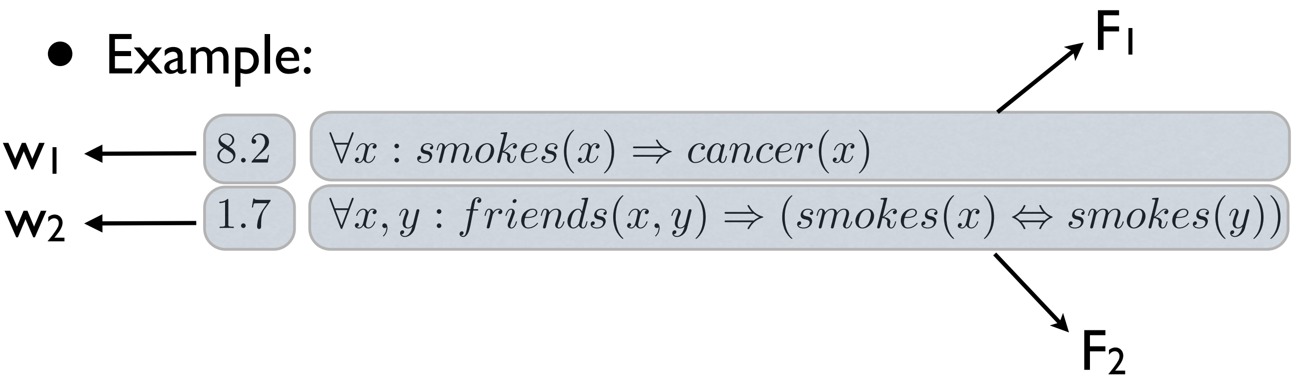
Logic?

> «... My friend Kit Fine has compared the position of the linguist or artificial intelligencer who turns to logic for this purpose to that of a man in need of trousers who goes to a tailor, only to be told that tailors only make jackets, and that in fact only jackets are necessary, for it is easy to show that jackets are topologically equivalent to trousers. Such is the authority of logicians that many otherwise decorous persons have found themselves in the position of trying to use jackets as trousers. When they have complained that jackets don't seem to work very well for the purpose [...], the response has often been impatient.
>
> Sometimes the users have been led to give up on logic entirely and to go off and invent their own knowledge representations. [...]
>
> *This is a shame, because in the end one's trousers are best made by tailors, and logicians are (or ought to be) the right people to make knowledge representations.»*

[M. Steedman. (2005) «*The Productions of Time: Temporality and Causality in Linguistic Semantics*»]

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:



w_1 ← 8.2 $\forall x : smokes(x) \Rightarrow cancer(x)$ → F_1

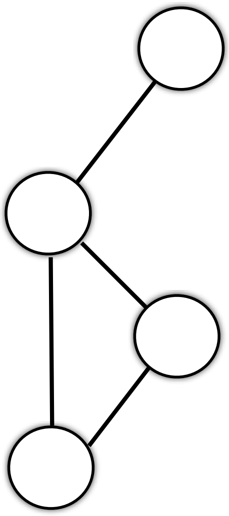
w_2 ← 1.7 $\forall x, y : friends(x,y) \Rightarrow (smokes(x) \Leftrightarrow smokes(y))$ → F_2

Reasoning with MLNs

- 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*), an equivalent Markov Network can be constructed, and used for inference
- This ground Markov Network can then be used for practical inference tasks
  - I.e. to compute the probability of *cancer(Alice)*

Markov Networks in a nutshell

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

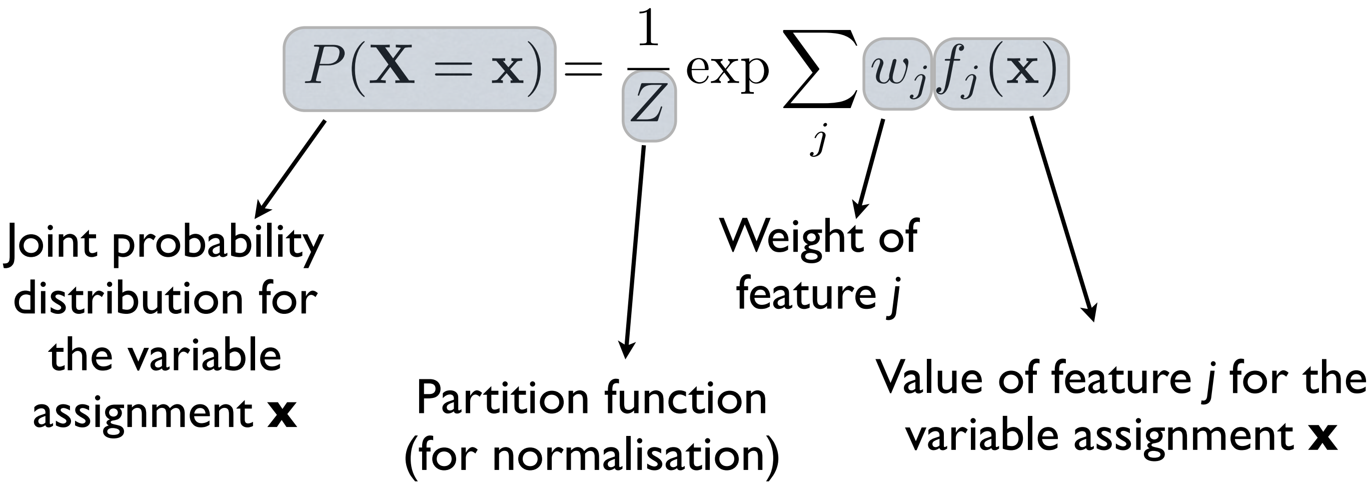


$$P(\mathbf{X} = \mathbf{x}) = \frac{1}{Z} \prod_k \phi_k(\mathbf{x}_{\{k\}})$$

where *k* is a clique, ϕ_k its potential function and Z a normalisation factor

Markov Networks in a nutshell

- The potential function ϕ$_k$ can be rewritten as an exponentiated weighted sum over feature functions
- The distribution then becomes a log-linear model:



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

Joint probability distribution for the variable assignment **x**

Partition function (for normalisation)

Weight of feature *j*

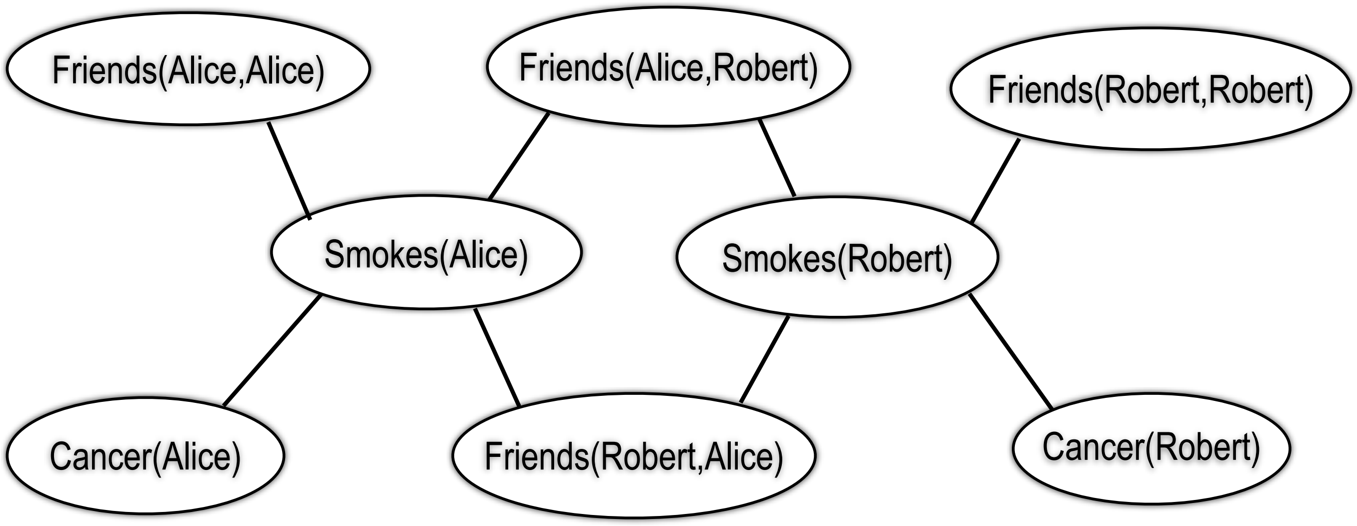
Value of feature *j* for the variable assignment **x**

Ground Markov Network

- Assume a Markov Logic Network *L* together with a set of constants *C*={c$_1$...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) is1 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

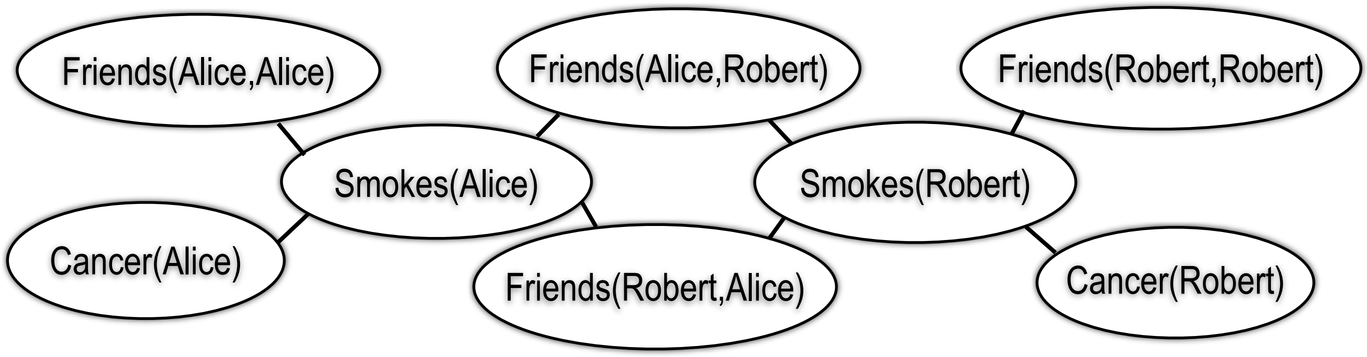
Construction example

- Two constants: *Alice* and *Robert*
- L = $8.2 \quad \forall x : smokes(x) \Rightarrow cancer(x)$
  $1.7 \quad \forall x, y : friends(x,y) \Rightarrow (smokes(x) \Leftrightarrow smokes(y))$



Construction example

- Two constants: *Alice* and *Robert*
- L = $8.2 \quad \forall x : smokes(x) \Rightarrow cancer(x)$
  $1.7 \quad \forall x, y : friends(x,y) \Rightarrow (smokes(x) \Leftrightarrow smokes(y))$



First formula has 2 possible groundings

Second formula has 4 possible groundings

→ model has 6 features

Construction example

- Two constants: *Alice* and *Robert*
- L = $8.2 \quad \forall x : smokes(x) \Rightarrow cancer(x)$
  $1.7 \quad \forall x, y : friends(x, y) \Rightarrow (smokes(x) \Leftrightarrow smokes(y))$

First formula has 2 possible groundings
Second formula has 4 possible groundings → model has 6 features

w_1= 8.2	$f_1(\mathbf{x})$ = 1 if *smokes(A)* ⇒ *cancer(A)*, 0 otherwise
w_2= 8.2	$f_2(\mathbf{x})$ = 1 if *smokes(R)* ⇒ *cancer(R)*, 0 otherwise
w_3= 1.7	$f_3(\mathbf{x})$ = 1 if *friends(A,R)⇒(smokes(A)⇔smokes(R))*, 0 otherwise
w_4= 1.7	$f_4(\mathbf{x})$ = 1 if *friends(A,A)⇒(smokes(A)⇔smokes(A))*, 0 otherwise
w_5= 1.7	$f_5(\mathbf{x})$ = 1 if *friends(R,A)⇒(smokes(R)⇔smokes(A))*, 0 otherwise
w_6= 1.7	$f_6(\mathbf{x})$ = 1 if *friends(R,R)⇒(smokes(R)⇔smokes(R))*, 0 otherwise

Probabilistic inference

- Once the ground Markov Network is constructed, it can be directly used for inference given some evidence
- For instance, compute the probability:

  P(cancer(R)|¬smokes(R), friends(R,A), smokes(A))

- Exact inference is intractable for all but the smallest domains

Probabilistic inference

- Fortunately, several algorithms for *approximate inference* are available for MLNs
- Often extensions of existing algorithms
  - Weighted version of MaxSAT, modified MCMC, etc.
- Most algorithms perform ground inference, but *lifted reference* is also possible
  - Akin to *resolution* in first-order logic

[P.Singla and P.Domingos (2008), «Lifted first-order belief propagation», *AAAI*]

Learning in MLNs

- Markov Logic Networks can be learned 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)
- MLNs have been successfully applied to both supervised and unsupervised learning

Learning in MLNs

- The easiest form of learn is to estimate the weights of known formulae
  - *Generative* weight learning seeks to maximise the pseudo-log-likelihood of the dataset
  - *Discriminative* weight learning seeks to maximise the conditional log-likelihood
- Typically done via some kind of *gradient descent* on the weights

Learning in MLNs

- It is also possible to do *structure learning*, where the algorithm tries to learn both the formulae and their weights
  - Akin to Inductive Logic Programming
- Finally, MLNs have also been applied to *unsupervised learning*, notably for reference resolution

Expressive power of MLNs

- Markov Logic Networks subsumes:
  - First-order logic
  - Probabilistic Graphical Models, either directed (Bayesian Networks) or undirected (Markov Networks)
  - Hidden Markov Models
  - Logistic regression (MaxEnt)
  - Probabilistic Context-Free Grammars
- But modelling not always trivial

[D. Jain (2011), «Knowledge Engineering with Markov Logic Networks: A Review», in *DKB*]

Extensions of MLNs

- Encoding of continuous variables and infinite domains with MLNs
- Relational *decision theory:* Adding *utility* values to certain predicates
- Recursive Random fields

[P. Singla and P. Domingos (2007). «Markov logic in infinite domains», *UAI*]

[Daniel Lowd and Pedro Domingos (2007), «Recursive Random Fields», IJCAI'07]

Outline

- Motivation
- Markov Logic Networks
- **Demonstration and examples**
- Conclusion

Live demonstration

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

http://alchemy.cs.washington.edu

(other toolkits are also available)

Applications of MLNs

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

Text classification

```
page = {1, ..., max}
word = { ... }
topic = { ... }
```

Term declaration

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

Predicate declaration

Rules with weights to estimate from data

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

If topics mutually exclusive: `Topic(page,topic!)`

Unsupervised reference resolution

[H. Poon and P. Domingos (2008). «Joint unsupervised coreference resolution with Markov logic»., *EMNLP*]

```
Head(mention, string)
Type(mention, type)
MentionOf(mention, entity)
Apposition(mention,mention)

MentionOf(+m,+e)
Type(+m,+t)
Head(+m,+h) ^ MentionOf(+m,+e)

MentionOf(a,e) ^ MentionOf(b,e)
   => (Type(a,t) <=> Type(b,t))

Apposition(a,b)
   => (MentionOf(a,e) <=> MentionOf(b,e))
```

enforce agreement

appositions are more likely to co-refer

Outline

- Motivation
- Markov Logic Networks
- Demonstration and examples
- **Conclusion**

Conclusions

- 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 algorithms for efficient inference & learning with MLNs
- Important applications for NLP