Markov Logic Networks
An Introduction

Prepared by

Chris Winsor

6/9/2022

Agenda

- Motivation and Overview
- Declarative Representation and First Order Languages
- Probabilistic Representation and Graphs
- Markov Logic Networks
- Examples

About Me

PhD Student at University of Massachusetts, Lowell

- Statistical Relational Learning
- Probabilistic Graphical Models
- Markov Logic Networks

Consultant at www.cwinsorconsulting.com

- Proof-of-Concepts, Benchmarking, Algorithms

Software Engineer at Rapid Micro Biosystems in Lowell, MA

References

Richardson, Matthew, and Pedro Domingos. "Markov logic networks." Machine learning 62.1 (2006): 107-136.

Domingos, Pedro, and Daniel Lowd. "Markov logic: An interface layer for artificial intelligence." Synthesis lectures on artificial intelligence and machine learning 3.1 (2009): 1-155.

Judea Pearl. 1988. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Koller, Daphne, and Nir Friedman. Probabilistic graphical models: principles and techniques. MIT press, 2009.

Russell, Stuart, and Peter Norvig. "Artificial intelligence: a modern approach." (2002).

Neapolitan, Richard E. Learning bayesian networks. Vol. 38. Upper Saddle River: Pearson Prentice Hall, 2004.

Kolmogorov, A. N. "Foundations of the Theory of Probability, trans." Nathan Morrison (1956). Chelsea, New York (1933).

Genesereth, Michael R., and Nils J. Nilsson. Logical foundations of artificial intelligence. Morgan Kaufmann, 2012.

Friedman, Nir, et al. "Learning probabilistic relational models." IJCAI. Vol. 99. 1999.

URLs:

Alchemy: http://alchemy.cs.washington.edu

Coursera PGM: https://www.coursera.org/specializations/probabilistic-graphical-models

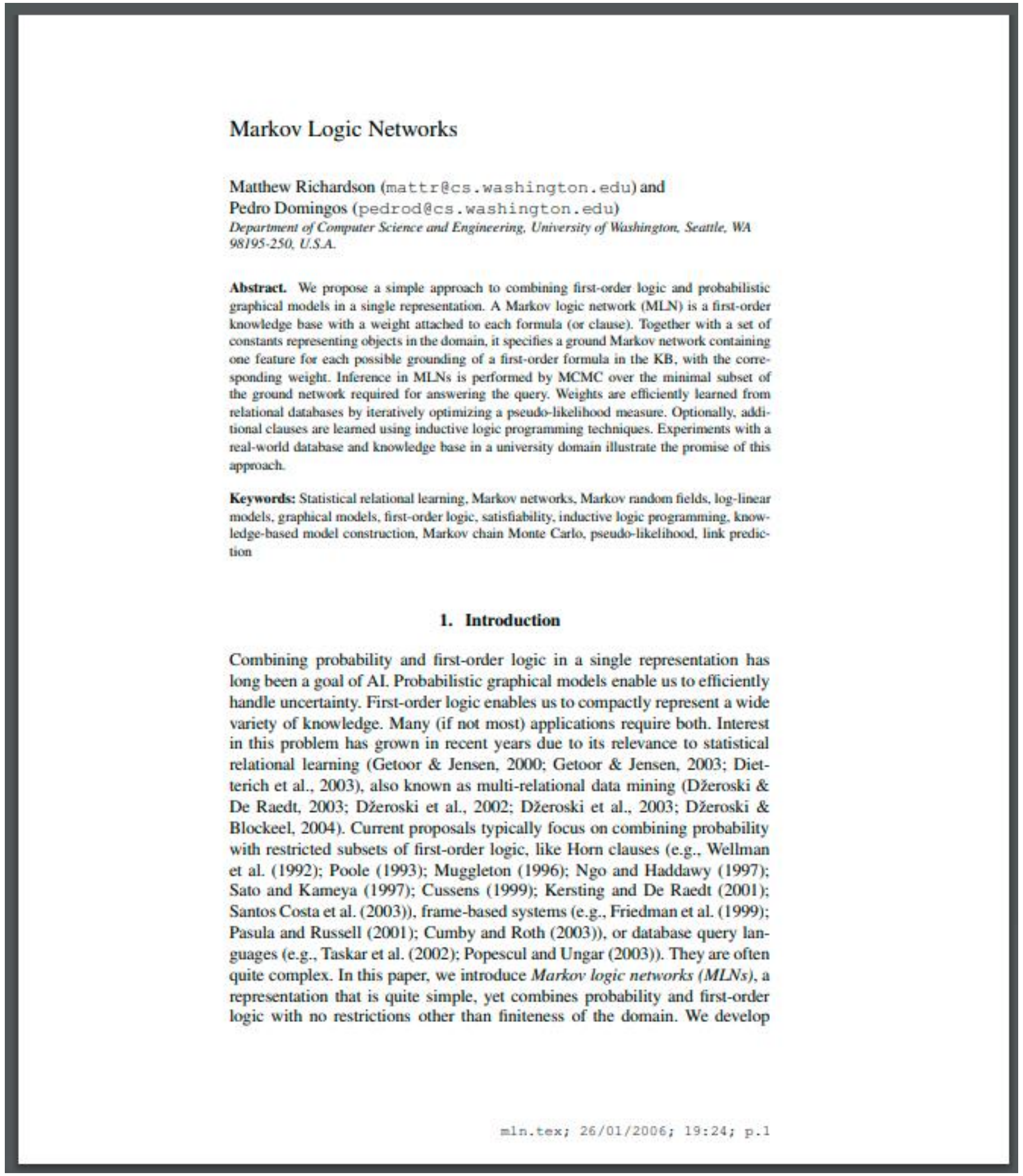
Stanford ML with Graphs: https://web.stanford.edu/class/cs224w/

Markov Logic Networks

What are they?

A probabilistic logic which applies the ideas of a Markov Network to first-order logic, enabling uncertain inference. (source: Wikipedia)

Research led by Pedro Domingos, Matt Richardson at University of Washington (first published in 2006)



Markov Logic Networks

Matthew Richardson (mattr@cs.washington.edu) and
Pedro Domingos (pedrod@cs.washington.edu)
Department of Computer Science and Engineering, University of Washington, Seattle, WA 98195-250, U.S.A.

Abstract. We propose a simple approach to combining first-order logic and probabilistic graphical models in a single representation. A Markov logic network (MLN) is a first-order knowledge base with a weight attached to each formula (or clause). Together with a set of constants representing objects in the domain, it specifies a ground Markov network containing one feature for each possible grounding of a first-order formula in the KB, with the corresponding weight. Inference in MLNs is performed by MCMC over the minimal subset of the ground network required for answering the query. Weights are efficiently learned from relational databases by iteratively optimizing a pseudo-likelihood measure. Optionally, additional clauses are learned using inductive logic programming techniques. Experiments with a real-world database and knowledge base in a university domain illustrate the promise of this approach.

Keywords: Statistical relational learning, Markov networks, Markov random fields, log-linear models, graphical models, first-order logic, satisfiability, inductive logic programming, knowledge-based model construction, Markov chain Monte Carlo, pseudo-likelihood, link prediction

1. Introduction

Combining probability and first-order logic in a single representation has long been a goal of AI. Probabilistic graphical models enable us to efficiently handle uncertainty. First-order logic enables us to compactly represent a wide variety of knowledge. Many (if not most) applications require both. Interest in this problem has grown in recent years due to its relevance to statistical relational learning (Getoor & Jensen, 2000; Getoor & Jensen, 2003; Dietterich et al., 2003), also known as multi-relational data mining (Džeroski & De Raedt, 2003; Džeroski et al., 2002; Džeroski et al., 2003; Džeroski & Blockeel, 2004). Current proposals typically focus on combining probability with restricted subsets of first-order logic, like Horn clauses (e.g., Wellman et al. (1992); Poole (1993); Muggleton (1996); Ngo and Haddawy (1997); Sato and Kameya (1997); Cussens (1999); Kersting and De Raedt (2001); Santos Costa et al. (2003)), frame-based systems (e.g., Friedman et al. (1999); Pasula and Russell (2001); Cumby and Roth (2003)), or database query languages (e.g., Taskar et al. (2002); Popescul and Ungar (2003)). They are often quite complex. In this paper, we introduce *Markov logic networks (MLNs)*, a representation that is quite simple, yet combines probability and first-order logic with no restrictions other than finiteness of the domain. We develop

Terminology and Motivation

Logic + Probability

First-order Language:

A formal language for representing logical (True/False) knowledge

Rules and algorithms for inference (deriving new facts from existing facts)

Probabilistic Graphical Model:

Graph for representing uncertainty and probabilistic relations

Algorithms for learning and inference in the face of uncertainty.

Motivation:

Combine power of FOL with flexibility of Probabilistic Representation

Declarative Knowledge

and First Order Language

Declarative Knowledge

Conceptualization

A conceptualization is the objects and relationships that define the domain of interest

- A set of Objects
- A set of Relations
- A set of Functions

Objects are things (Confucius, the sun, the number 2, justice)

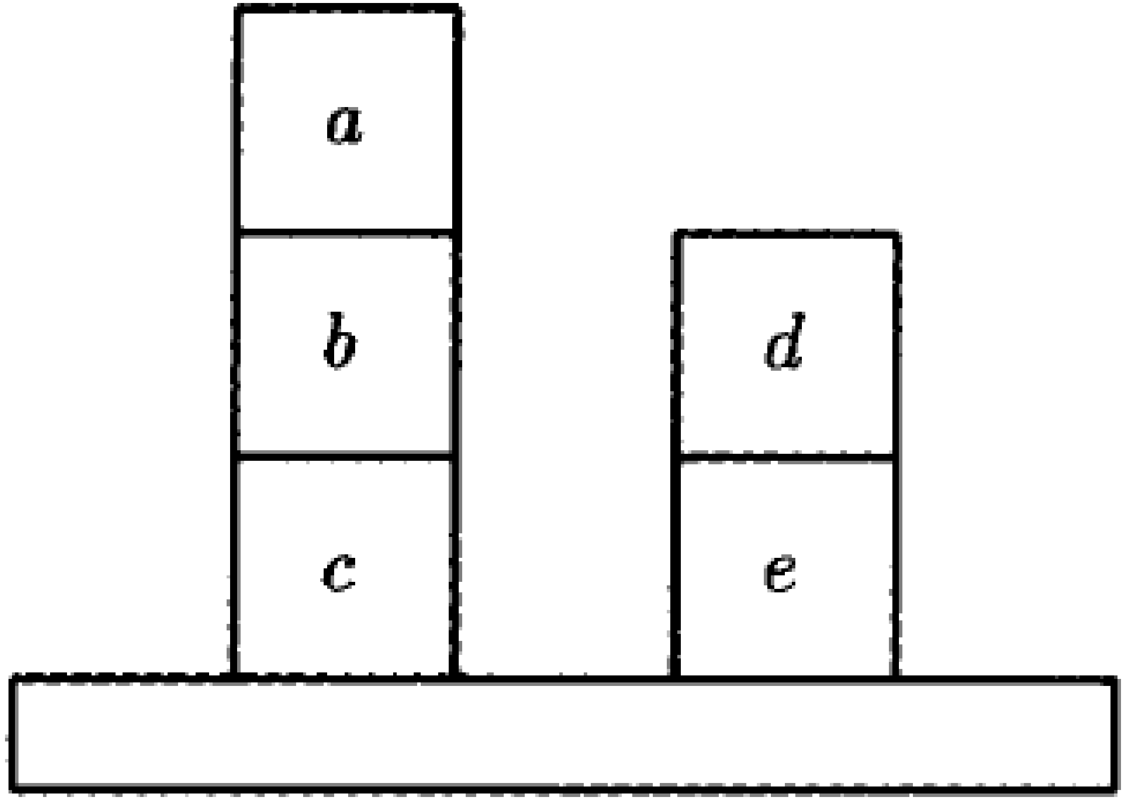
Relations take in Object/s and evaluate to True/False

Functions take in Object/s and evaluate to an Object

Formally a conceptualization is the triple: ({objects}, {relations}, {functions})

Reference: Genesereth and Nilsson

Example: Blocks World



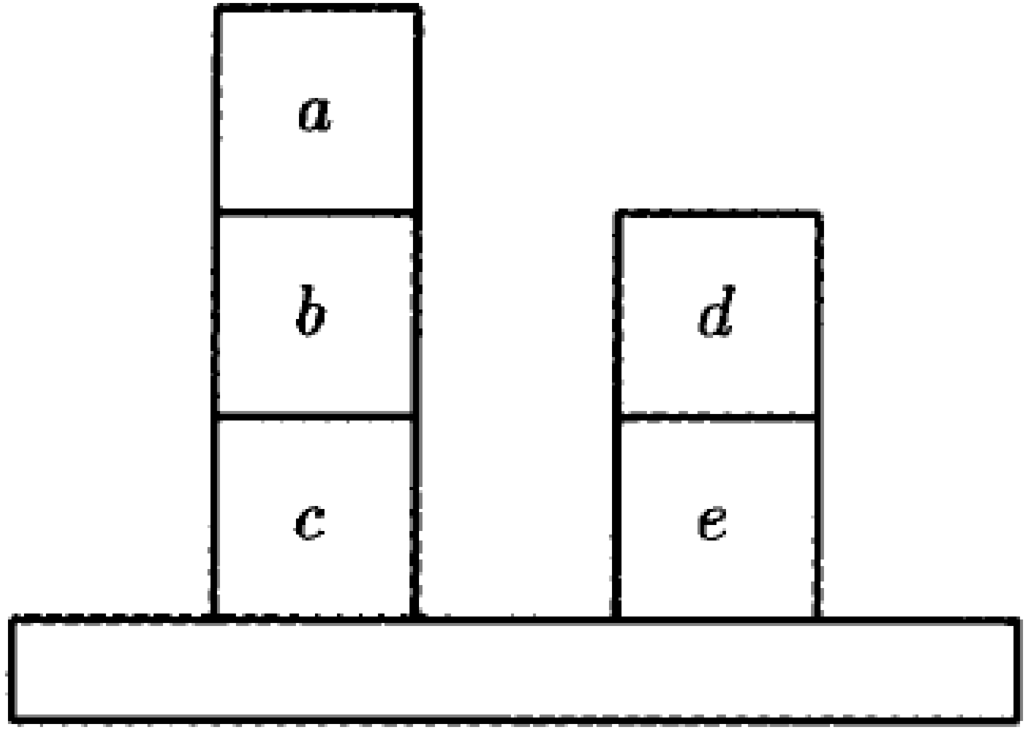
Reference: Genesereth and Nilsson

Example: Blocks World

Objects:

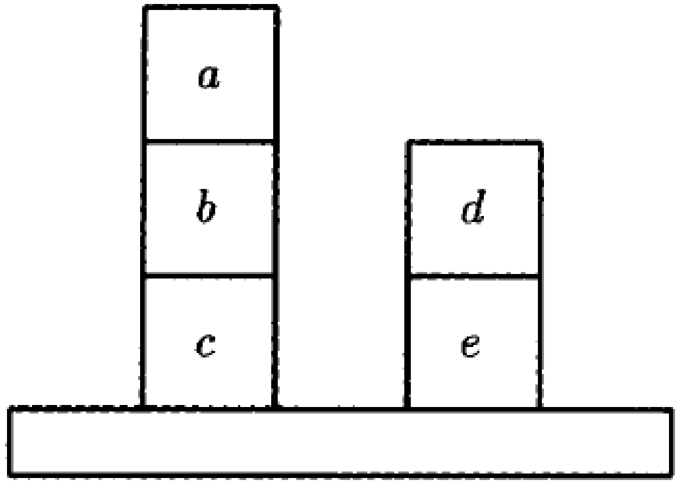
Relations:

Functions:



Reference: Genesereth and Nilsson

Example: Blocks World



Objects = { a, b, c, d, e }

Relations	Holds if
On(x, y)	x is directly on top of y
Above(x, y)	x is above y
Clear(x)	no blocks are above x
Table(x)	block x is directly on the table

Functions	Maps to
Hat(x)	the object directly above x (if any)

Conceptualization: $blocksworld = (\{a, b, c, d, e\}, \{hat\}, \{on, above, clear, table\})$

Reference: Genesereth and Nilsson

Predicate Calculus

Knowledge Expressed as Sentences

$$(\neg \mathtt{Age}(\mathtt{Confucius},\mathtt{100}))$$

logical operator relation constants sentence

Confucius is not 100 years old

$$(\mathtt{Location}(\mathtt{George},\mathtt{Home}) \Rightarrow \mathtt{Sick}(\mathtt{George}))$$

relation implication relation

If George is at home, then George is sick

Reference: Genesereth and Nilsson

Predicate Calculus

Logical Operators and Quantifiers…

$\neg$	Negation
$\wedge \;\; \vee$	Conjunction, Disjunction
$\Rightarrow$	Implication
$\Leftrightarrow$	Equivalence
$\forall$	Universal Quantifier (“for all…”)
$\exists$	Existential Quantifier (“there exists…”)

Reference: Genesereth and Nilsson

More examples

implication

$$(\forall x\ (Apple(x) \Rightarrow Red(x)))$$

universal quantifier

variable

All apples in the world are red

$$(\exists x\ (Apple(x) \wedge Red(x)))$$

existential quantifier

There exists at least one red apple in the world

Reference: Genesereth and Nilsson

Inference in a Declarative Representation
(an example)

We know that horses are faster than dogs

There is a greyhound that is faster than every rabbit

Harry is a horse and Ralph is a rabbit

Is Harry faster than Ralph?

Reference: Genesereth and Nilsson

Inference in a Declarative Representation

- Horses are faster than dogs
- There is a greyhound that is faster than every rabbit
- Harry is a horse and Ralph is a rabbit

$$\forall x \forall y\ \mathrm{Horse}(x) \land \mathrm{Dog}(y) \Rightarrow \mathrm{Faster}(x,y)$$

$$\exists y\ \mathrm{Greyhound}(y) \land (\forall z\ \mathrm{Rabbit}(z) \Rightarrow \mathrm{Faster}(y,z))$$

$$\forall y\ \mathrm{Greyhound}(y) \Rightarrow \mathrm{Dog}(y)$$

$$\forall x \forall y \forall z\ \mathrm{Faster}(x,y) \land \mathrm{Faster}(y,z) \Rightarrow \mathrm{Faster}(x,z)$$

$$\mathrm{Horse}(\mathrm{Harry})$$

$$\mathrm{Rabbit}(\mathrm{Ralph})$$

Is Harry faster than Ralph?

Derive → $\mathrm{Faster}(\mathrm{Harry},\mathrm{Ralph})$

Reference: Genesereth and Nilsson

and the answer is...

Harry is indeed faster than Ralph

Note:

- *Inference uses only standard rules of deduction – nothing specific to the domain.*

1.	$\forall x \forall y\ Horse(x) \wedge Dog(y) \Rightarrow Faster(x,y)$	Δ
2.	$\exists y\ Greyhound(y) \wedge (\forall z\ Rabbit(z) \Rightarrow Faster(y,z))$	Δ
3.	$\forall y\ Greyhound(y) \Rightarrow Dog(y)$	Δ
4.	$\forall x \forall y \forall z\ Faster(x,y) \wedge Faster(y,z) \Rightarrow Faster(x,z)$	Δ
5.	$Horse(Harry)$	Δ
6.	$Rabbit(Ralph)$	Δ
7.	$Greyhound(Greg) \wedge (\forall z\ Rabbit(z) \Rightarrow Faster(Greg,z))$	2, EI
8.	$Greyhound(Greg)$	7, AE
9.	$\forall z\ Rabbit(z) \Rightarrow Faster(Greg,z)$	7, AE
10.	$Rabbit(Ralph) \Rightarrow Faster(Greg,Ralph)$	9, UI
11.	$Faster(Greg,Ralph)$	10, 6, MP
12.	$Greyhound(Greg) \Rightarrow Dog(Greg)$	3, UI
13.	$Dog(Greg)$	12, 8, MP
14.	$Horse(Harry) \wedge Dog(Greg) \Rightarrow Faster(Harry,Greg)$	1, UI
15.	$Horse(Harry) \wedge Dog(Greg)$	5, 13, AI
16.	$Faster(Harry,Greg)$	14, 15, MP
17.	$Faster(Harry,Greg) \wedge Faster(Greg,Ralph) \Rightarrow Faster(Harry,Ralph)$	4, UI
18.	$Faster(Harry,Greg) \wedge Faster(Greg,Ralph)$	16, 11, AI
19.	$Faster(Harry,Ralph)$	17, 19, MP

Reference: Genesereth and Nilsson

but there's more…

Ask any question…

- Are all horses faster than all rabbits?
- Is there any horse that is slower than Greg?

Add knowledge, extend the domain…

- If turtles are slower than dogs, can we conclude all turtles are slower than rabbits?

Represent *partial knowledge*

- Waldo is faster than Ralph and slower than Greg. (We don't know what animal Waldo is, but we can add this partial knowledge to the Knowledgebase)

1.	$\forall x \forall y\ \mathtt{Horse}(x) \land \mathtt{Dog}(y) \Rightarrow \mathtt{Faster}(x,y)$	Δ
2.	$\exists y\ \mathtt{Greyhound}(y) \land (\forall z\ \mathtt{Rabbit}(z) \Rightarrow \mathtt{Faster}(y,z))$	Δ
3.	$\forall y\ \mathtt{Greyhound}(y) \Rightarrow \mathtt{Dog}(y)$	Δ
4.	$\forall x \forall y \forall z\ \mathtt{Faster}(x,y) \land \mathtt{Faster}(y,z) \Rightarrow \mathtt{Faster}(x,z)$	Δ
5.	$\mathtt{Horse}(\mathtt{Harry})$	Δ
6.	$\mathtt{Rabbit}(\mathtt{Ralph})$	Δ
7.	$\mathtt{Greyhound}(\mathtt{Greg}) \land (\forall z\ \mathtt{Rabbit}(z) \Rightarrow \mathtt{Faster}(\mathtt{Greg},z))$	2, EI
8.	$\mathtt{Greyhound}(\mathtt{Greg})$	7, AE
9.	$\forall z\ \mathtt{Rabbit}(z) \Rightarrow \mathtt{Faster}(\mathtt{Greg},z)$	7, AE
10.	$\mathtt{Rabbit}(\mathtt{Ralph}) \Rightarrow \mathtt{Faster}(\mathtt{Greg},\mathtt{Ralph})$	9, UI
11.	$\mathtt{Faster}(\mathtt{Greg},\mathtt{Ralph})$	10, 6, MP
12.	$\mathtt{Greyhound}(\mathtt{Greg}) \Rightarrow \mathtt{Dog}(\mathtt{Greg})$	3, UI
13.	$\mathtt{Dog}(\mathtt{Greg})$	12, 8, MP
14.	$\mathtt{Horse}(\mathtt{Harry}) \land \mathtt{Dog}(\mathtt{Greg}) \Rightarrow \mathtt{Faster}(\mathtt{Harry},\mathtt{Greg})$	1, UI
15.	$\mathtt{Horse}(\mathtt{Harry}) \land \mathtt{Dog}(\mathtt{Greg})$	5, 13, AI
16.	$\mathtt{Faster}(\mathtt{Harry},\mathtt{Greg})$	14, 15, MP
17.	$\mathtt{Faster}(\mathtt{Harry},\mathtt{Greg}) \land \mathtt{Faster}(\mathtt{Greg},\mathtt{Ralph}) \Rightarrow \mathtt{Faster}(\mathtt{Harry},\mathtt{Ralph})$	4, UI
18.	$\mathtt{Faster}(\mathtt{Harry},\mathtt{Greg}) \land \mathtt{Faster}(\mathtt{Greg},\mathtt{Ralph})$	16, 11, AI
19.	$\mathtt{Faster}(\mathtt{Harry},\mathtt{Ralph})$	17, 19, MP

Declarative Representation + Logical Reasoning

In Summary:

1. A Conceptualization establishes the domain of interest (objects, functions, relations)
2. Rules of inference are domain independent (no custom code)
3. New knowledge can be added to the model easily (as a new sentence)
4. Any domain-related question can be asked of the model – its not purpose-built

1.	$\forall x \forall y\ \mathtt{Horse}(x) \land \mathtt{Dog}(y) \Rightarrow \mathtt{Faster}(x,y)$	Δ
2.	$\exists y\ \mathtt{Greyhound}(y) \land (\forall z\ \mathtt{Rabbit}(z) \Rightarrow \mathtt{Faster}(y,z))$	Δ
3.	$\forall y\ \mathtt{Greyhound}(y) \Rightarrow \mathtt{Dog}(y)$	Δ
4.	$\forall x \forall y \forall z\ \mathtt{Faster}(x,y) \land \mathtt{Faster}(y,z) \Rightarrow \mathtt{Faster}(x,z)$	Δ
5.	$\mathtt{Horse}(\mathtt{Harry})$	Δ
6.	$\mathtt{Rabbit}(\mathtt{Ralph})$	Δ
7.	$\mathtt{Greyhound}(\mathtt{Greg}) \land (\forall z\ \mathtt{Rabbit}(z) \Rightarrow \mathtt{Faster}(\mathtt{Greg},z))$	2, EI
8.	$\mathtt{Greyhound}(\mathtt{Greg})$	7, AE
9.	$\forall z\ \mathtt{Rabbit}(z) \Rightarrow \mathtt{Faster}(\mathtt{Greg},z)$	7, AE
10.	$\mathtt{Rabbit}(\mathtt{Ralph}) \Rightarrow \mathtt{Faster}(\mathtt{Greg},\mathtt{Ralph})$	9, UI
11.	$\mathtt{Faster}(\mathtt{Greg},\mathtt{Ralph})$	10, 6, MP
12.	$\mathtt{Greyhound}(\mathtt{Greg}) \Rightarrow \mathtt{Dog}(\mathtt{Greg})$	3, UI
13.	$\mathtt{Dog}(\mathtt{Greg})$	12, 8, MP
14.	$\mathtt{Horse}(\mathtt{Harry}) \land \mathtt{Dog}(\mathtt{Greg}) \Rightarrow \mathtt{Faster}(\mathtt{Harry},\mathtt{Greg})$	1, UI
15.	$\mathtt{Horse}(\mathtt{Harry}) \land \mathtt{Dog}(\mathtt{Greg})$	5, 13, AI
16.	$\mathtt{Faster}(\mathtt{Harry},\mathtt{Greg})$	14, 15, MP
17.	$\mathtt{Faster}(\mathtt{Harry},\mathtt{Greg}) \land \mathtt{Faster}(\mathtt{Greg},\mathtt{Ralph}) \Rightarrow \mathtt{Faster}(\mathtt{Harry},\mathtt{Ralph})$	4, UI
18.	$\mathtt{Faster}(\mathtt{Harry},\mathtt{Greg}) \land \mathtt{Faster}(\mathtt{Greg},\mathtt{Ralph})$	16, 11, AI
19.	$\mathtt{Faster}(\mathtt{Harry},\mathtt{Ralph})$	17, 19, MP

Declarative vs Procedural Representation

Task	Procedural Code	Declarative / FOL
Deriving facts from other facts	Domain-specific procedure	Entirely domain independent
Adding new knowledge	Implementation dependent	Add sentence to KB
Adding partial knowledge (stoplight #4 is not green)	Re-code	Add sentence to KB

Reference: Russell, Norvig Chapter 8

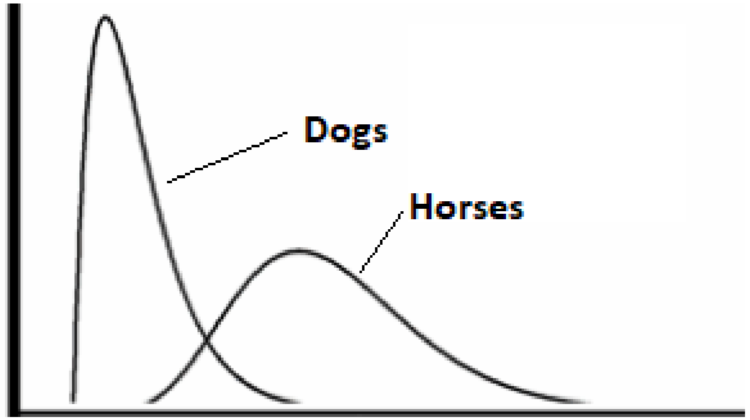
The problem w/ FOL

Sensors are unreliable...

Data is messy...

A rule may not hold for all cases.

One bad observation can spoil a model




```
Smokes(x) => Cancer(x)
```

Probabilistic Representation

Probabilistic Graphical Models (PGMs)

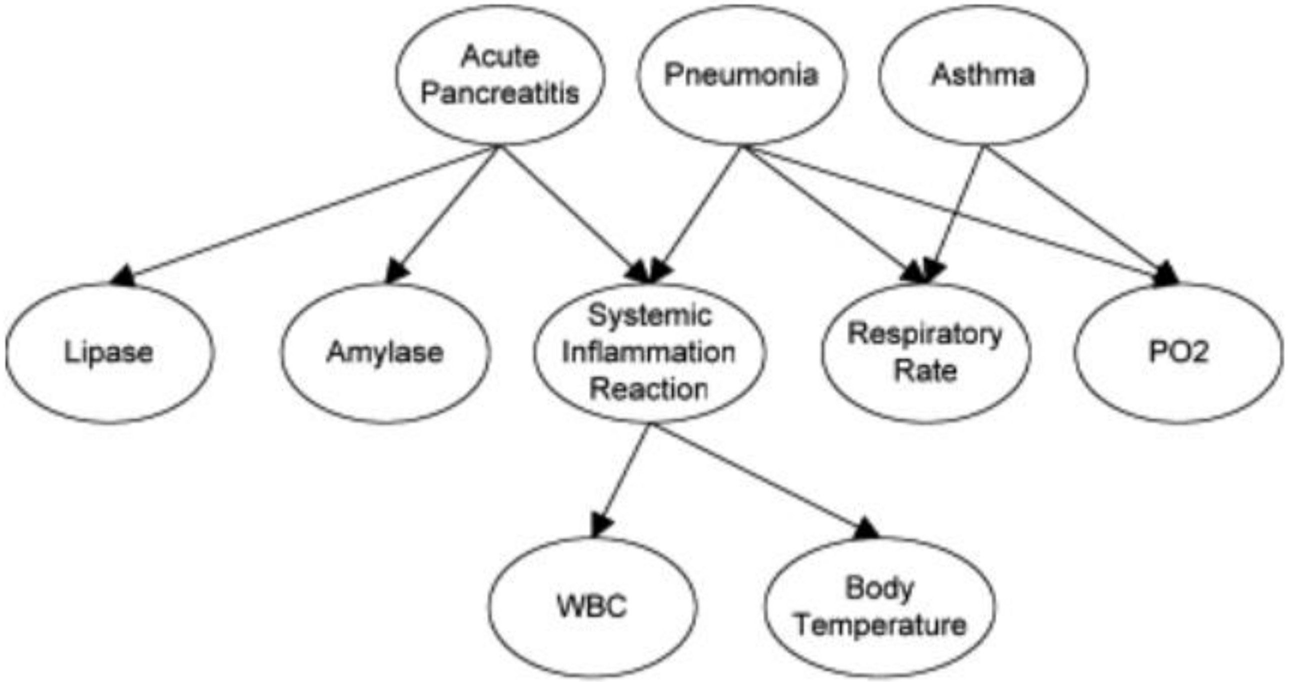
A PGM is a graph data structure consisting of Nodes, Edges

$$G = \{V, E\}$$

Graphs may be:

- Directed, undirected
- Cyclic, acyclic
- Have attributes on nodes, edges or groups of nodes

Graphs succinctly represent many real-world problems



Medical Diagnosis (directed, acyclic)

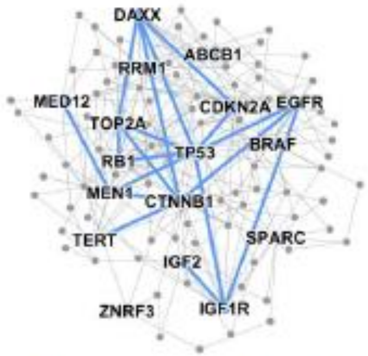
Jau-Huei Lin, Peter J. Haug, Exploiting missing clinical data in Bayesian network modeling for predicting medical problems, Journal of Biomedical Informatics, Volume 41, Issue 1, 2008, Pages 1-14, ISSN 1532-0464, https://doi.org/10.1016/j.jbi.2007.06.001.

Graphs

Event Graphs

Image credit: SalientNetworks

Computer Networks



Disease Pathways

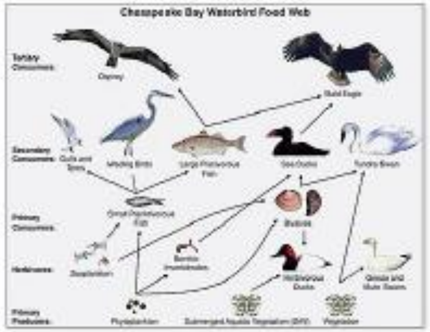


Image credit: Wikipedia

Food Webs

Image credit: Pinterest

Particle Networks

Image credit: visitlondon.com

Underground Networks

Graphs

Image credit: Medium

Social Networks

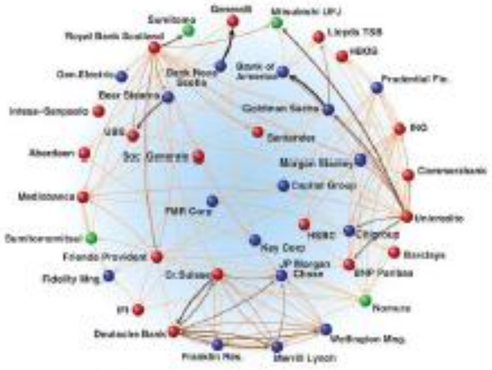
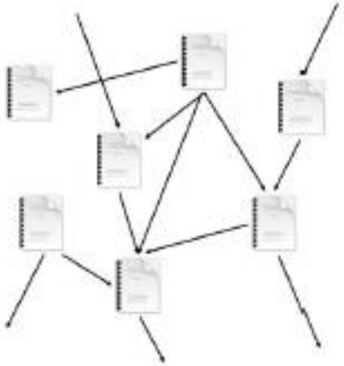


Image credit: Science

Economic Networks

Image credit: Lumen Learning

Communication Networks



Citation Networks

Image credit: Missoula Current News

Internet

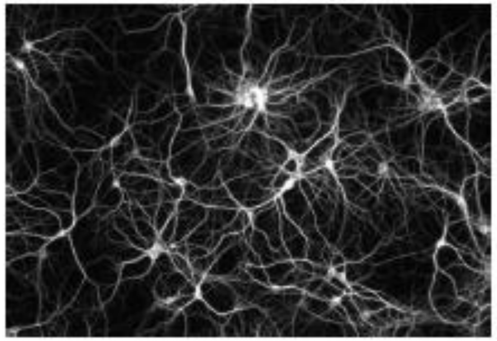


Image credit: The Conversation

Networks of Neurons

Graphs

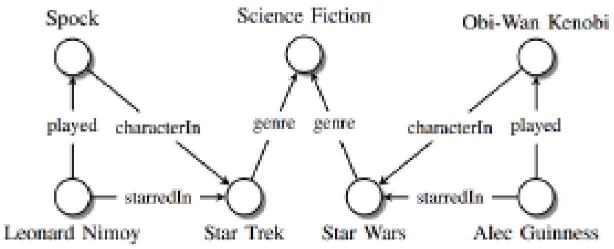


Image credit: Maximilian Nickel et al

Knowledge Graphs

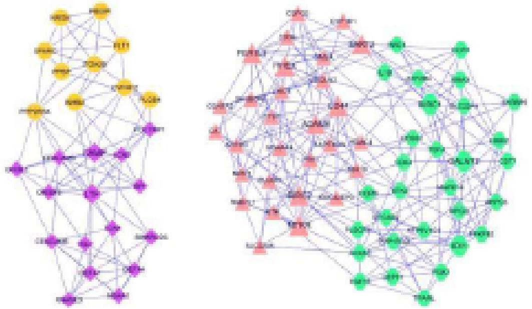


Image credit: ese.wustl.edu

Regulatory Networks

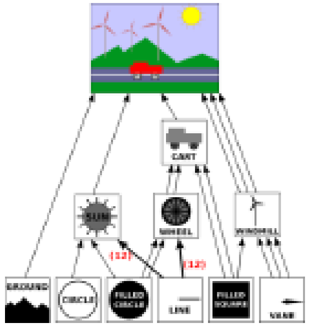


Image credit: math.hws.edu

Scene Graphs

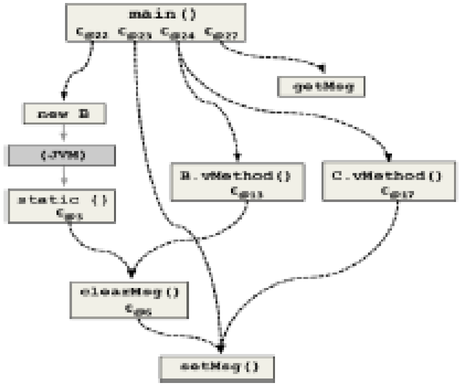


Image credit: ResearchGate

Code Graphs

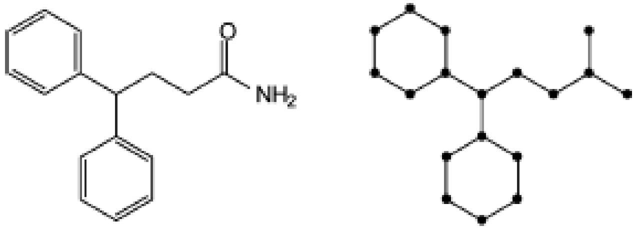


Image credit: MDPI

Molecules

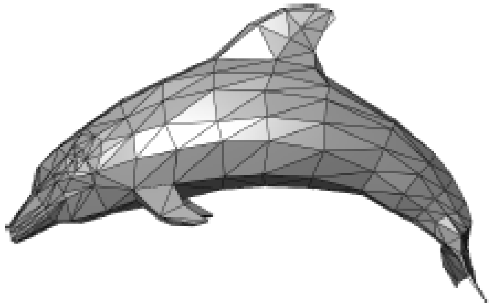


Image credit: Wikipedia

3D Shapes

Jure Leskovec, Stanford CS224W: Machine Learning with Graphs
https://web.stanford.edu/class/cs224w/slides/01-intro.pdf

Markov Network
A Graph representing a Joint Probability Distribution

A Markov Network is:

- An Undirected Cyclic Graph $G = \{V, E\}$
- A set of potential functions ϕ_k - one for each “clique” (set of fully-connected nodes)

With above constraints the graph represents a Joint Probability Distribution

- Each node in the graph is a random variable
- Each potential function expresses the relation between the random variables in the clique

The representation is complete, consistent (Pearl 1988)

Judea Pearl

Clique

a fully connected subgraph

In the example:

12 cliques of 1

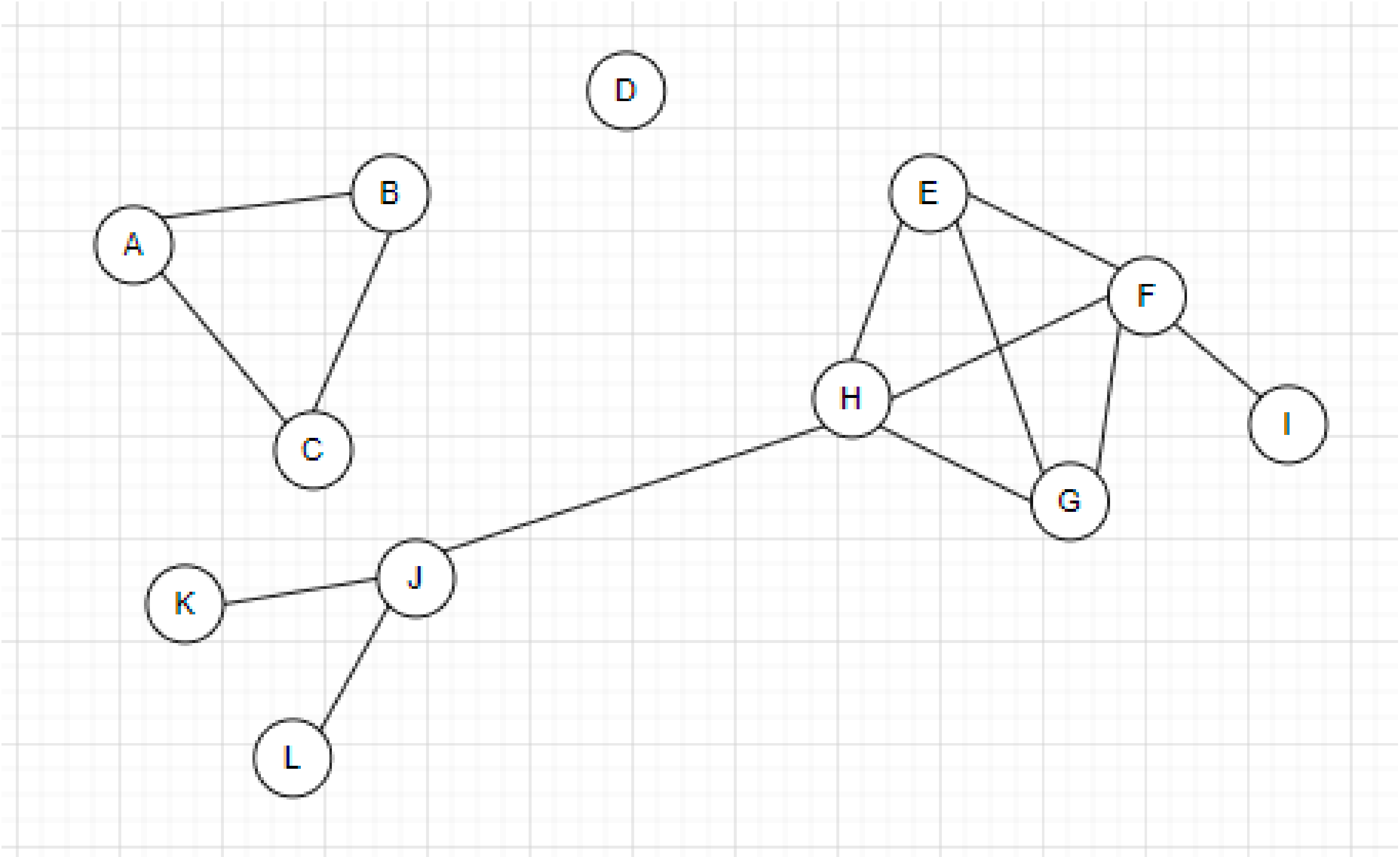
13 cliques of 2

5 cliques of 3

1 clique of 4

Represent using 31 ϕ

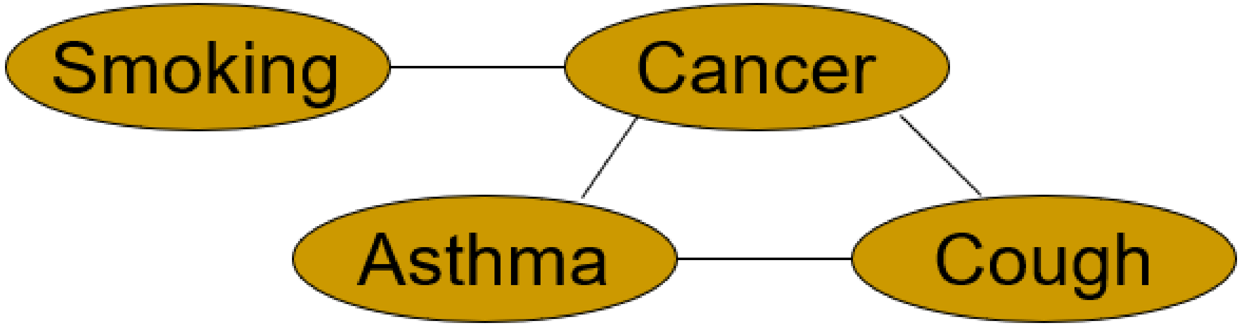
$$P(\,) = \frac{1}{Z} \prod_{n \in (1..31)} \phi_n$$



$\{\phi\}$ is a complete and consistent expression of relationships

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

http://alchemy.cs.washington.edu/alchemy1.html

Features of Markov Networks

Markov network is based on probability theory - peer accepted in scientific communities and regulatory institutions

Structure of an MLN graph can have intuitive meaning - help in understanding the domain.

“Probabilistic models are liberating. Instead of a rigid formalism that needs to enumerate every possibility and exception we can sweep these under the probabilistic rug as something unusual happened” (1)

(1) Coller, Friedman 2009

Markov *Logic* Networks

Combine FOL + Markov Network

MLN "softens" the constraints of the FOL. Each formula has a 'weight'. The fewer constraints a world violates the more probable it is.

DEFINITION 4.1. *A Markov logic network L is a set of pairs (F_i, w_i), where F_i is a formula in first-order logic and w_i is a real number. Together with a finite set of constants $C = \{c_1, c_2, \ldots, c_{|C|}\}$, it defines a Markov network $M_{L,C}$ (Equations 1 and 2) as follows:*

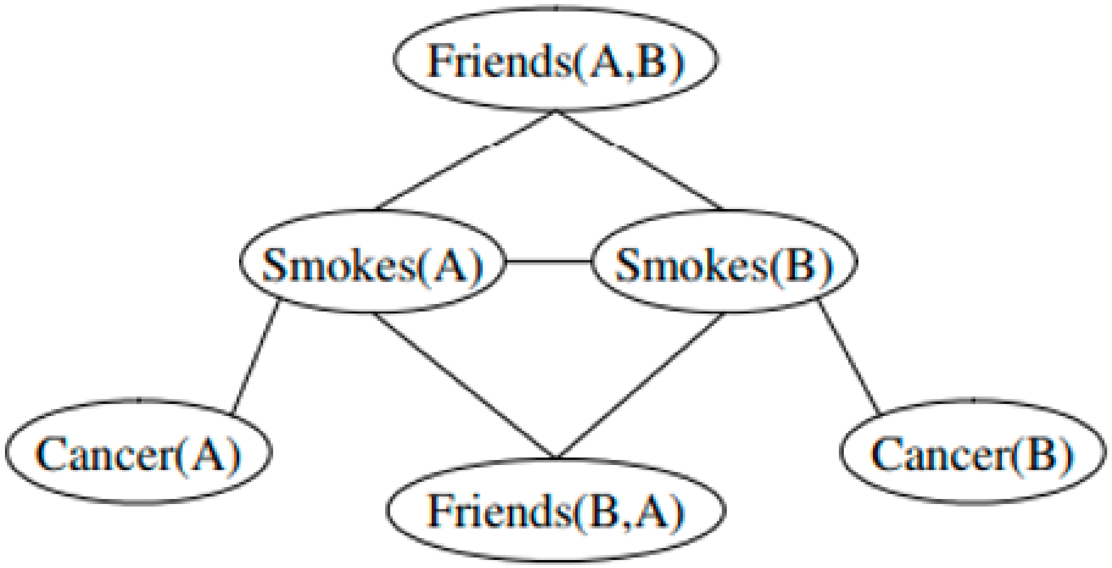
1. *$M_{L,C}$ contains one binary node for each possible grounding of each predicate appearing in L. The value of the node is 1 if the ground atom is true, and 0 otherwise.*
2. *$M_{L,C}$ contains one feature for each possible grounding of each formula F_i in L. The value of this feature is 1 if the ground formula is true, and 0 otherwise. The weight of the feature is the w_i associated with F_i in L.*

Richardson, M., Domingos, P. Markov logic networks. *Mach Learn* **62,** 107–136 (2006). https://doi.org/10.1007/s10994-006-5833-1

https://homes.cs.washington.edu/~pedrod/papers/mlj05.pdf

Example

English	First-Order Logic	Weight
Smoking causes cancer.	$\forall x\ \mathtt{Sm}(x) \Rightarrow \mathtt{Ca}(x)$	1.5
If two people are friends, either both smoke or neither does.	$\forall x \forall y\ \mathtt{Fr}(x, y) \Rightarrow (\mathtt{Sm}(x) \Leftrightarrow \mathtt{Sm}(y))$	1.1



Ground Markov network obtained by applying the formulas the constants Anna(A) and Bob(B).

https://homes.cs.washington.edu/~pedrod/papers/mlj05.pdf

Exercises

Alchemy

A tool for MLN research by University of Washington (Domingos et al.)

Intended for:

- Tutorials and Learning about MLN
- Experimenting with Algorithms
  - *Structure Learning*
  - *Weight learning*
  - *Inference*

Version 1.0 ← Initial version

Version 2.0 ← Lifted Inference

Light ← "Tractable MLN"

Example 1: Social Network

English	First-Order Logic
Smoking causes cancer.	$\forall x\ \mathtt{Sm}(x) \Rightarrow \mathtt{Ca}(x)$
If two people are friends, either both smoke or neither does.	$\forall x \forall y\ \mathtt{Fr}(x, y) \Rightarrow (\mathtt{Sm}(x) \Leftrightarrow \mathtt{Sm}(y))$

Domingos, Lowd, 2009 Appendix A

Elicitation from
Knowledge Expert

.mln file establishes the model

predicates, functions, first-order formulas

.mln file (smoking.mln)

```
// Evidence
Friends(person, person)

// Some evidence, some query
Smokes(person)

// Query
Cancer(person)

// Rules
// If you smoke, you get cancer
1.5 Smokes(x) => Cancer(x)

// People with friends who smoke, also smoke
// and those with friends who don't smoke, don't smoke
0.8 Friends(x, y) => (Smokes(x) <=> Smokes(y))
```

Predicates

First-order Formulas

.db file grounds the model

smoking-train.db

```
Friends(Anna, Bob)
Friends(Bob, Anna)
Friends(Anna, Edward)
Friends(Edward, Anna)
Friends(Anna, Frank)
Friends(Frank, Anna)
Friends(Bob, Chris)
Friends(Chris, Bob)
Friends(Chris, Daniel)
Friends(Daniel, Chris)
Friends(Edward, Frank)
Friends(Frank, Edward)
Friends(Gary, Helen)
Friends(Helen, Gary)
Friends(Gary, Anna)
Friends(Anna, Gary)

Smokes(Anna)
Smokes(Edward)
Smokes(Frank)
Smokes(Gary)

Cancer(Anna)
Cancer(Edward)
```

Ground atoms

Weight learning

input .mln

output .mln

training.db

non-evidence predicates

```
$ learnwts -d -i smoking.mln -o smoking-out.mln -t smoking-train.db -ne Smokes,Cancer
```

smoking-out.mln

updated weights

first-order formula converted to CNF form

weights for non-evidence predicates

```
//predicate declarations
Cancer(person)
Friends(person,person)
Smokes(person)

// 1.3597   Smokes(x) => Cancer(x)
1.3597   !Smokes(a1) v Cancer(a1)

// 0.946642   Friends(x,y) => (Smokes(x) <=> Smokes(y))
0.473321   !Friends(a1,a2) v Smokes(a1) v !Smokes(a2)
0.473321   !Friends(a1,a2) v Smokes(a2) v !Smokes(a1)

// 0         Friends(a1,a2)
0        Friends(a1,a2)

// 0.798775   Smokes(a1)
0.798775   Smokes(a1)

// -1.57104   Cancer(a1)
-1.57104   Cancer(a1)
```

Inference

input .mln
evidence
output
query predicate

```
$ infer -i smoking-out.mln -e smoking-test.db -r smoking.results -q Smokes -ms -maxSteps 20000
```

smoking-test.db

```
Friends(Ivan, John)
Friends(John, Ivan)
Friends(Katherine, Lars)
Friends(Lars, Katherine)
Friends(Michael, Nick)
Friends(Nick, Michael)
Friends(Ivan, Michael)
Friends(Michael, Ivan)

Smokes(Ivan)
Smokes(Nick)
```

smoking.results

```
Smokes(John) 0.689081
Smokes(Katherine) 0.467253
Smokes(Lars) 0.458004
Smokes(Michael) 0.853215
```

Inference

new query

```
$ infer -i smoking-out.mln -e smoking-test.db -r smoking.results -q Cancer -ms -maxSteps 20000
```

smoking.results

```
Cancer(Ivan) 0.446605
Cancer(John) 0.166883
Cancer(Katherine) 0.170583
Cancer(Lars) 0.166883
Cancer(Michael) 0.172583
Cancer(Nick) 0.452205
```

Key Points (example 1)

No Data Required. Elicitation from Knowledge Expert is a valid and good starting point.

Model is transferrable. Structure from and weights from group (Anna, Bob, Chris) is applied to the (Katherine, Ivan, John) group.

Query is not pre-defined – can query on any predicate

Structure succinct – expressed in just a few lines

Example 2: University Students
Learning Structure from Data

The previous model came from Knowledge Expert. If we want structure from data...

"University" database is:

- Students, Professors, "AdvisedBy" relationship.
- Publication Title (identifies student and professor)
- Faculty members have "position", student has "phase"

Start with empty .mln

Example 2: University Students

The Data

univ-train.db

```
professor(Ada)
professor(Alan)
professor(Alex)
professor(Alice)
professor(Andy)
student(Bart)
student(Becca)
student(Betty)
student(Bill)
student(Bob)
student(Carl)
student(Carol)
student(Cathy)
student(Charles)
student(Claire)
advisedBy(Bart, Ada)
advisedBy(Becca, Ada)
advisedBy(Betty, Alan)
advisedBy(Bill, Alan)
advisedBy(Bob, Alex)
advisedBy(Carl, Alex)
advisedBy(Carol, Alice)
advisedBy(Cathy, Alice)
advisedBy(Charles, Andy)
advisedBy(Claire, Andy)
```

```
publication(Title1, Bart)
publication(Title1, Ada)
publication(Title2, Becca)
publication(Title2, Ada)
publication(Title3, Betty)
publication(Title3, Alan)
publication(Title4, Bill)
publication(Title4, Alan)
publication(Title5, Bob)
publication(Title5, Alex)
publication(Title6, Carl)
publication(Title6, Alex)
publication(Title7, Carol)
publication(Title7, Alice)
publication(Title8, Cathy)
publication(Title8, Alice)
publication(Title9, Charles)
publication(Title9, Andy)
publication(Title10, Claire)
publication(Title10, Andy)
```

```
inPhase(Bart, Pre_Quals)
inPhase(Becca, Pre_Quals)
inPhase(Betty, Post_Quals)
inPhase(Bill, Post_Quals)
inPhase(Bob, Post_Quals)
inPhase(Carl, Post_Quals)
inPhase(Carol, Post_Quals)
inPhase(Cathy, Post_Quals)
inPhase(Charles, Post_Quals)
inPhase(Claire, Post_Quals)
hasPosition(Ada, Faculty)
hasPosition(Alan, Faculty)
hasPosition(Alex, Faculty)
hasPosition(Alice, Faculty)
hasPosition(Andy, Faculty_emeritus)
```

Example 2: University Students
The Data

univ-empty.mln

```
//predicate declaration
professor(person)
student(person)
advisedBy(person, person)
publication(title, person)
inPhase(person, phase)
hasPosition(person, position)
```

learn structure input output data

```
$ learnstruct -i univ-empty.mln -o univ-empty-out.mln -t univ-train.db
```

Example 2: University Students

The Structure

univ-empty-out.mln

```
//predicate declarations
hasPosition(person,position)
advisedBy(person,person)
professor(person)
publication(title,person)
inPhase(person,phase)
student(person)

-6.26304   inPhase(a1,a2)
-10.4154   hasPosition(a1,a2)
-6.26298   student(a1)
-6.70464   advisedBy(a1,a1)
4.49645    publication(a1,a2)
6.26298    professor(a1)
3.16703    advisedBy(a1,a2)
8.60628    !advisedBy(a1,a2) v publication(a3,a2) v !publication(a3,a1)
2.78642    !publication(a1,a3) v !publication(a2,a3) v hasPosition(a3,a4) v hasPosition(a3,a5) v a1 = a2
-1.87409   professor(a1) v !student(a1) v advisedBy(a2,a1) v inPhase(a1,a3) v !inPhase(a1,a4) v hasPosition(a1,a5)
```

Key Points (example 2)

Structure from Data

Model expressed in CNF form – marginally human readable

Example 3: Logistic Regression

Maps attributes (numeric or ordinal) to nominal (categorical).

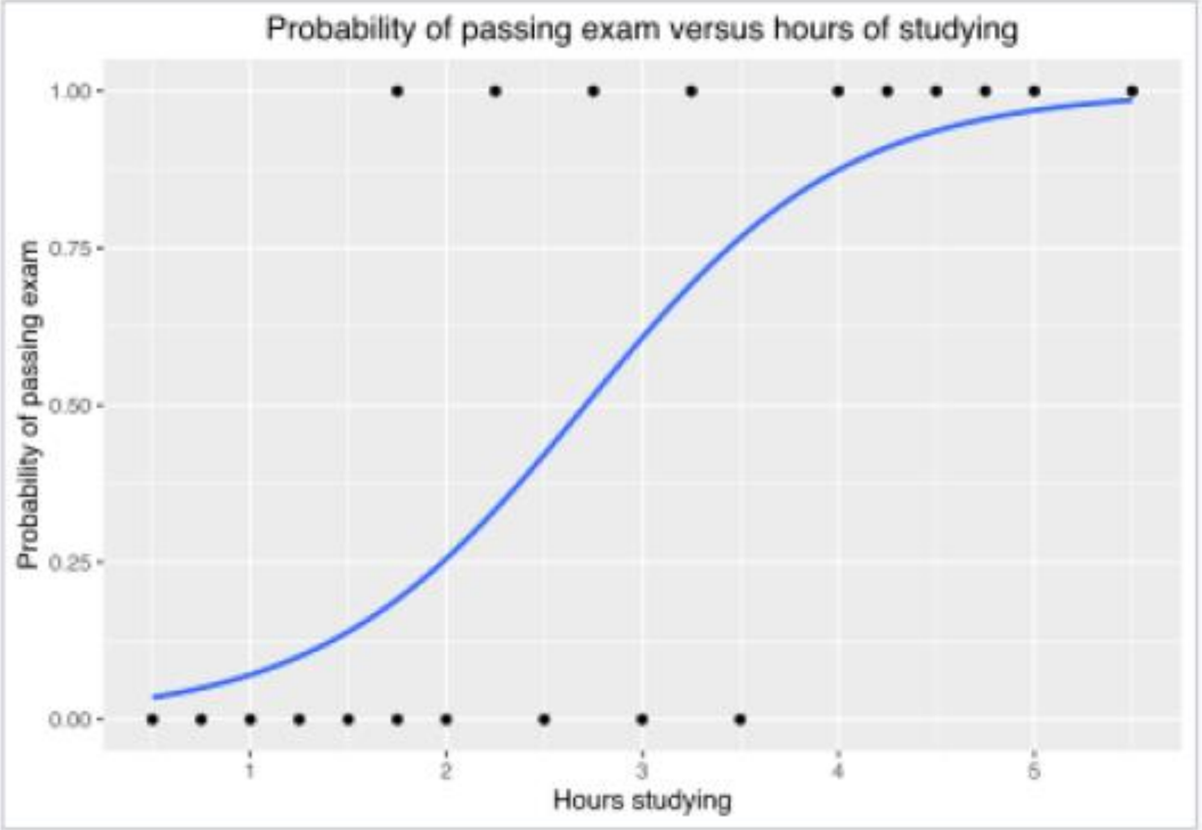
Output sums to 1.0 and is between 0 and 1.0

The logistic function is of the form:

$$p(x) = \frac{1}{1 + e^{-(x-\mu)/s}}$$

where μ is a location parameter (the midpoint of the curve, where $p(\mu) = 1/2$) and s is a scale parameter.

Source: wikipedia



Graph of a logistic regression curve fitted to the (x_m, y_m) data. The curve shows the probability of passing an exam versus hours studying.

Example 3: UCI "voting-records" Dataset

Yea/Nay votes for 232 congresspersons on 16 topics

Class to be determined is "Republican" or "Democrat"

Each vote is a binary predictor

```
republican,n,y,n,y,y,y,n,n,n,y,?,y,y,y,n,y
republican,n,y,n,y,y,y,n,n,n,n,n,y,y,y,n,?
democrat,?,y,y,?,y,y,n,n,n,n,y,n,y,y,n,n
democrat,n,y,y,n,?,y,n,n,n,n,y,n,y,n,n,y
democrat,y,y,y,n,y,y,n,n,n,n,y,?,y,y,y,y
…
```

```
Number of Instances: 435 (267 democrats, 168 republicans)

Number of Attributes: 16 + class name = 17 (all Boolean valued)

Attribute Information:
 1. Class Name: 2 (democrat, republican)
 2. handicapped-infants: 2 (y,n)
 3. water-project-cost-sharing: 2 (y,n)
 4. adoption-of-the-budget-resolution: 2 (y,n)
 5. physician-fee-freeze: 2 (y,n)
 6. el-salvador-aid: 2 (y,n)
 7. religious-groups-in-schools: 2 (y,n)
 8. anti-satellite-test-ban: 2 (y,n)
 9. aid-to-nicaraguan-contras: 2 (y,n)
10. mx-missile: 2 (y,n)
11. immigration: 2 (y,n)
12. synfuels-corporation-cutback: 2 (y,n)
13. education-spending: 2 (y,n)
14. superfund-right-to-sue: 2 (y,n)
15. crime: 2 (y,n)
16. duty-free-exports: 2 (y,n)
17. export-administration-act-south-africa: 2 (y,n)
```

Example 3:

Step 1: Attributes become Grounding Predicates

As received: standard table format

```
$ more house-votes-84.data
republican,n,y,n,y,y,y,n,n,n,y,?,y,y,y,n,y
republican,n,y,n,y,y,y,n,n,n,n,n,y,y,y,n,?
democrat,?,y,y,?,y,y,n,n,n,n,y,n,y,y,n,n
democrat,n,y,y,n,?,y,n,n,n,n,y,n,y,n,n,y
democrat,y,y,y,n,y,y,n,n,n,n,y,?,y,y,y,y
…
```

In MLN each vote becomes a grounding predicate

Congressperson 190 voted against WaterProjectCostSharing

```
$
$ grep 190 ../voting-train.db
!Democrat(190)
HandicappedInfants(190)
!WaterProjectCostSharing(190)
AdoptionOfTheBudgetResolution(190)
PhysicianFeeFreeze(190)
!ElSalvadorAid(190)
…
```

common_sm_ui

Example 3:
Learn Weights

```
$
$ learnwts -g -i voting.mln -o voting-gen.mln -t voting-train.db -ne Democrat
...
Computing counts took 0.07 secs
L-BFGS-B is finding optimal weights......
num iterations        = 40
time taken            = 0.05 secs
pseudo-log-likelihood = -0.0302705
Total time = 0.17 secs
```

Example 3:
Inference

```
$
$
infer -ms -i voting-disc.mln -r voting.result -e voting-test.db -q Democrat
cat voting.result
$ cat voting.result
Democrat(191) 0.0090491
Democrat(192) 0.0020498
Democrat(193) 0.99795
Democrat(194) 0.99895
Democrat(195) 0.223028
Democrat(196) 0.283022
Democrat(197) 0.936956
Democrat(198) 0.989951
Democrat(199) 0.0520448
Democrat(200) 0.0340466
Democrat(201) 0.0440456
Democrat(202) 0.0080492
Democrat(203) 0.0120488
Democrat(204) 0.99995
Democrat(205) 0.346015
Democrat(206) 0.927957
Democrat(207) 0.99995
Democrat(208) 0.124038
```

Key Points (example 3)

Common machine learning design patterns can be implemented in MLN using recipe.

Example (# from tutorial)	Dataset	Purpose	Predicates and Functions	Ground-ings	Run time
1 – Social Network	Smoking	Basic learnstruct, learnweights, Infer	5 hand crafted	22	15 sec
2 – Student/Advisor	University	Structure and weights from data	From Data		
3 – Logistic Regression	Voting	Logistic Regression (numeric/ordinal to nominal)	5 hand crafted	22	6 sec
5.1 – Text Classification	WebKB	Classification by bag-of-words	2	6100	22 min
6 – Entity Resolution	CORA	Matching items across image frames, documents	10 basic 28 sentence	56000	19 hours learning 21 hours total
7 – Hidden Markov Model	Traffic	Sequential pattern (toy dataset)	7	32	1hr 40
9.1 - NLP	Dogs and cats	Grammar and lexicon	20	230	6 min
10 – Bayes Net	Alarm	Bayesian/PGM to MLN	37	n/a	n/a
11 - Hybrid	Robot range	Numeric attributes	5	295	

In Summary

Markov Logic Networks combine the logic of a First Order Language with the probabilistic representation of a Markov Network.

Structure can be established from data or knowledge expert

Knowledgebase is flexible - can be easily updated or extended

Allows representing partial knowledge

Inference from rules engine

Inference accepts any question (not purpose-built)

Succinctly represents complicated relationships

Thank You