Probabilistic Graphical Models

Course 2 : Markov Logic Networks

Hoel Le Capitaine
Academic year 2017-2018

Outline

Introduction

Information and Knowledge – 1988

Databases
SQL
Datalog

Free text
Information retrieval
NLP

Knowledge bases
FOL

Structured ⟷ Unstructured

Information

Motivation

Information and Knowledge – today

Databases
SQL
Datalog

Web services
SOAP
WSDL

Hypertext
HTML

Free text
Information retrieval
NLP

Semis-struc. info
XML

Knowledge bases
FOL

Deep Web

Semantic web
RDF
OWL

Information extraction

Sensor Data

Structured ⟷ Unstructured

Information

Needed

We need languages that can handle

- Structured information
- Unstructured information
- Any variation or combination of them

We need efficient algorithms for them

- Inference
- Machine learning

Background

Unifies first-order logic and probabilistic graphical models

- First-order logic handles structured information

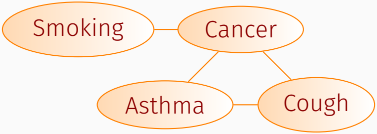
Unifies first-order logic and probabilistic graphical models

- First-order logic handles structured information
- Probability handles unstructured information

Unifies first-order logic and probabilistic graphical models

- First-order logic handles structured information
- Probability handles unstructured information
- No separation between the two
- Builds on previous work (BN, PRM, ...)

Undirected graphical models



Potential functions

- potential functions defined on **cliques**
- joint distribution of the MN :

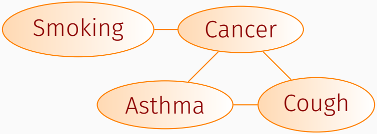
$$P(X = x) = \frac{1}{Z} \prod_k \phi_k(x_{\{k\}})$$

- $x_{\{k\}}$ is the state of the kth clique
- Z is the partition function

$$Z = \sum_x \prod_k \phi_k(x_{\{k\}})$$

Smoking	Cancer	$\phi(S, C)$
false	false	4.5
false	true	4.5
true	false	2.7
true	true	4.5

Undirected graphical models

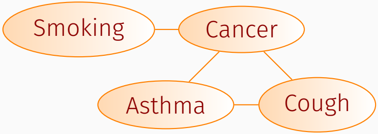


Log-linear models

- each clique replaced by an exponentiated weighted sum of (binary) features of the state

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

Undirected graphical models



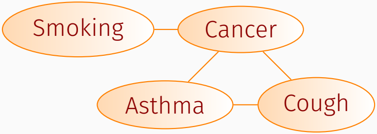
Log-linear models

- each clique replaced by an exponentiated weighted sum of (binary) features of the state

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_j w_j \, f_j(x) \right)$$

- weight of feature j

Undirected graphical models



Log-linear models

- each clique replaced by an exponentiated weighted sum of (binary) features of the state

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

- weight of feature j
- feature j

Some recalls

- constants, variables, functions, predicates
  Sarah, x, MotherOf(x), Friends(x,y)
- grouding : replace all variables by constants
  Friends(Sarah, Bernard)
- World (model, interpretation) : assignment of truth valuser to all ground predicates

Markov Logic

Markov logic

From logic to probability

- a logical knowledge base is a set of **hard constraints** on the set of possible worlds
- transform them into **soft constraints** : if a world violates a formula, it becomes less probable, not impossible
- each formula is associated to a **weight**, corresponding to the strength of the constraint

$$P(world) \approx \exp\left(\sum \text{weights of satisfied formulas}\right)$$

Definition

A **Markov Logic Network (MLN)** is a set of pairs (F, w) where F is a 1^{st}-order logic formula, and w is a real value.
It defines a Markov network with

- one node for each grounding of each predicate
- one feature for each grounding of each formula F, with corresponding weight w

A first KB

- smoking causes cancer

Example : friends and smokers

A first KB

- smoking causes cancer $\forall x\ Smokes(x) \rightarrow Cancer(x)$

A first KB

- smoking causes cancer $\forall x\ Smokes(x) \rightarrow Cancer(x)$
- if 2 people are friends, either both smokes or neither does

A first KB

- smoking causes cancer $\forall x\ Smokes(x) \rightarrow Cancer(x)$
- if 2 people are friends, either both smokes or neither does $\forall x \forall y\ Friends(x, y) \rightarrow \Big(Smokes(x) \Leftrightarrow Smokes(y)\Big)$

A first KB

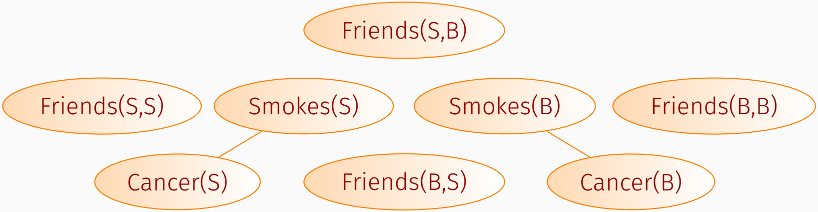
- **smoking causes cancer**

  $\forall x \; Smokes(x) \rightarrow Cancer(x)$ $w = 1.5$

- if 2 people are friends, either both smokes or neither does

  $\forall x \forall y \; Friends(x, y) \rightarrow \big(Smokes(x) \Leftrightarrow Smokes(y)\big)$ $w = 1.1$

Two constants : Sarah (S) and Bernard (B)



Example : friends and smokers

A first KB

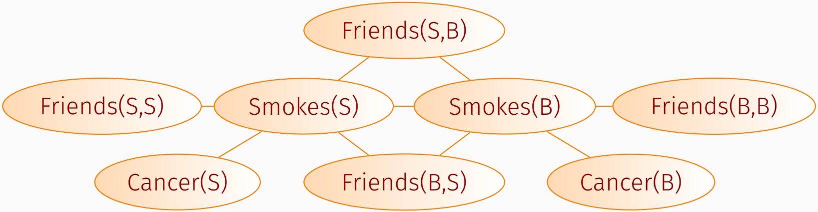
- smoking causes cancer

  $\forall x \; Smokes(x) \rightarrow Cancer(x)$ $w = 1.5$

- **if 2 people are friends, either both smokes or neither does**

  $\forall x \forall y \; Friends(x,y) \rightarrow \Big(Smokes(x) \Leftrightarrow Smokes(y)\Big)$ $w = 1.1$

Two constants : Sarah (S) and Bernard (B)



- MLN is a template for ground Markov networks
- probability of a world X

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

  $n_i(X)$: number of true groundings of formula i in X
- **typed** variables and constants reduce size of the net
- functions, existential quantifiers
- both infinite and continuous domains

Coming back to first order logic

- infinite weights exactly correspond to first-order logic
- satisfiable KB, positive weights : Satisfying assignments = Modes of distribution
- Markov logic allows contradictions between formulas

Inference

MAP inference

- we want to find the most likely state of world, given our observation $\operatorname{argmax}_y P(y|x)$

MAP inference

- we want to find the most likely state of world, given our observation $\mathbf{argmax}_y\, P(y|x)$
  - y : query
  - x : evidence

MAP inference

- we want to find the most likely state of world, given our observation $\mathbf{argmax}_y\, P(y|x)$
  - y : query
  - x : evidence
- using definition, $\mathbf{argmax}_y\, \frac{1}{Z_x} \exp\left(\sum_i w_i n_i(x, y)\right)$

MAP inference

- we want to find the most likely state of world, given our observation $\mathbf{argmax}_y\, P(y|x)$
  - y : query
  - x : evidence
- using definition, $\mathbf{argmax}_y\, \frac{1}{Z_x} \exp\left(\sum_i w_i n_i(x, y)\right)$
- due to indep. and monotonocity, $\mathbf{argmax}_y\, \sum_i w_i n_i(x, y)$
- reduced to a weighted maxSAT problem (eg walkSAT)
- may be faster than logical inference !

walkSAT Algorithm (1996)

```
for i ←1 to max-tries do
    solution = random truth assignment
    for j ← 1 to max-flips do
        if all clauses satisfied then
            return solution
        c ← random unsatisfied clause
        with probability p
            flip a random variable in c
        else
            flip variable in c that maximizes number of satisfied clauses
return failure
```

maxwalkSAT Algorithm (1997)

```
for i ←1 to max-tries do
    solution = random truth assignment
    for j ← 1 to max-flips do
        if ∑ weights of satisfied clauses > t then
            return solution
        c ← random unsatisfied clause
        with probability p
            flip a random variable in c
        else
            flip variable in c that maximizes ∑ weights of satisfied clauses
return failure, best solution found
```

Ok .. but

- memory explosion ...
- if there are n constants, and highest clause arity is c, ground network requires $O(n^c)$ memory
- we need to exploit sparseness : LazySAT (2006)

Learning

Learning

- Data is a relational database
- Closed world assumption (if not: EM)
- Learning parameters (weights)
  - Generatively
  - Discriminatively
- Learning structure (formulas)

Generative weight learning

- based on maximum likelihood
- uses gradient ascent (or more complicated ones)
- no local maxima $\Delta_{w_i} \log P_w(x) = n_i(x) - E_w[n_i(x)]$
    - $E_w[n_i(x)]$ expected number of true groundings according to the model
- but requires inference at each step of the optimization (slow)

Pseudo-likelihood

- Likelihood of each variable given its neighbors in the data

$$PL(x) = \prod_i P(x_i | N(x_i))$$

- 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

Discriminative weight learning

- maximize conditional likelihood of query y given observation x

$$\Delta_{w_i} \log P_w(y|x) = n_i(x, y) - E_w[n_i(x, y)]$$

- approximation of expectation counts by counts in MAP state of y given x (voted perceptron)

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 (2005–2007)