Outline

- Markov Logic Networks
- Alchemy
Markov Logic Networks

- Bayesian networks and Markov networks:
  - Model uncertainty
  - But propositional representation (e.g., we need one variable per object in the world)
- First-order logic:
  - First-order representation (e.g., quantifiers allow us to reason about several objects simultaneously)
  - But we can't deal with uncertainty
- Markov logic networks: combine Markov networks and first-order logic

Markov Logic

- A logical KB is a set of hard constraints on the set of possible worlds
- Let's make them soft constraints: when a world violates a formula, it becomes less probable, not impossible
- Give each formula a weight:
  (higher weight $\Rightarrow$ stronger constraint)

$$P(\text{world}) \propto e^{\sum \text{weights of formulas it satisfies}}$$
Markov Logic: Definition

- 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
- Together with a set of constants, it defines a Markov network with
  - One node for each grounding of each predicate in the MLN
  - One feature for each grounding of each formula \(F\) in the MLN, with the corresponding weight \(w\)

Example: Friends & Smokers

- Smoking causes cancer.
- Friends have similar smoking habits.
Example: Friends & Smokers

\[ \forall x \text{ Smokes}(x) \Rightarrow \text{Cancer}(x) \]
\[ \forall x, y \text{ Friends}(x, y) \Rightarrow (\text{Smokes}(x) \Leftrightarrow \text{Smokes}(y)) \]
Example: Friends & Smokers

1.5 \( \forall x \text{ Smokes}(x) \Rightarrow \text{Cancer}(x) \)
1.1 \( \forall x, y \text{ Friends}(x, y) \Rightarrow (\text{Smokes}(x) \Leftrightarrow \text{Smokes}(y)) \)

Two constants: Anna (A) and Bob (B)
Example: Friends & Smokers

∀x Smokes(x) ⇒ Cancer(x)
∀x, y Friends(x, y) ⇒ (Smokes(x) ⇔ Smokes(y))

Two constants: Anna (A) and Bob (B)

Friends(A,B)
Friends(A,A) Smokes(A) Smokes(B) Friends(B,B)
Cancer(A) Friends(B,A) Cancer(B)
Example: Friends & Smokers

\[ \forall x \ Smokes(x) \Rightarrow Cancer(x) \]
\[ \forall x, y \ Friends(x, y) \Rightarrow (Smokes(x) \Leftrightarrow Smokes(y)) \]

Two constants: Anna (A) and Bob (B)

Markov Logic Networks

- **MLN** is template for ground Markov nets
- Probability of a world \( x \):
  \[
P(x) = \frac{1}{Z} \exp \left( \sum_i w_i \eta_i(x) \right)
\]
  - Weight of formula \( i \)
  - No. of true groundings of formula \( i \) in \( x \)
- **Typed** variables and constants greatly reduce size of ground Markov net
Alchemy

- Open Source AI package
- http://alchemy.cs.washington.edu
- Implementation of Markov logic networks

- Problem specified in two files:
  - File1.mln (Markov logic network)
  - File2.db (database / data set)

- Learn weights and structure of MLN
- Inference queries

Markov Logic Encoding

- File.mln

- Two parts:
  - Declaration
    - Domain of each variable
    - Predicates
  - Formula
    - Pairs of weights with logical formula
Markov Logic Encoding

- **Example declaration**
  - Domain of each variable
    - person = \{Anna, Bob\}
  - Predicates:
    - Friends(person, person)
    - Smokes(person)
    - Cancer(person)

- **Example formula**
  - 8 Smokes(x) => Cancer(x)
  - 5 Friends(x, y) => (Smokes(x) <=> Smokes(y))

NB: by default, formulas are universally quantified in Alchemy

Dataset

- File.db

- **List of facts (ground atoms)**

- **Example:**
  - Friends(Anna, Bob)
  - Smokes(Anna)
  - Cancer(Bob)
Syntax

• Logical connective:
  - ! (not), ^ (and), v (or), => (implies), <=> (iff)

• Quantifiers:
  - forall (∀), exist (∃)
  - By default unquantified variables are universally quantified in Alchemy

• Operator precedence:
  - ! > ^ > v > => > <=> > forall = exist

Syntax

• Short hand for predicates
  - ! operator: indicates that the preceding variable has exactly one true grounding
  - Ex: HasPosition(x,y!): for each grounding of x, exactly one grounding of y satisfies HasPosition

• Short hand for multiple weights
  - + operator: indicates that a different weight should be learned for each grounding of the following variable
  - Ex: outcome(throw,+face): a different weight is learned for each grounding of face
Multinomial Distribution

Example: Throwing dice

Types:   throw = { 1, … , 20 }
         face = { 1, … , 6 }

Predicate: Outcome(throw, face)

Formulas: Outcome(t,f) ^ f!=f' => !Outcome(t,f').
          Exist f Outcome(t,f).

Too cumbersome!

Multinomial Distrib.: ! Notation

Example: Throwing dice

Types:   throw = { 1, … , 20 }
         face = { 1, … , 6 }

Predicate: Outcome(throw, face!)

Formulas:

Semantics: Arguments without “!” determine args with “!”.
Only one face possible for each throw.
**Multinomial Distrib.: + Notation**

**Example:** Throwing biased dice

**Types:**
- \( \text{throw} = \{ 1, \ldots, 20 \} \)
- \( \text{face} = \{ 1, \ldots, 6 \} \)

**Predicate:** \( \text{Outcome}(\text{throw,face!}) \)

**Formulas:** \( \text{Outcome}(t,+f) \)

**Semantics:** Learn weight for each grounding of args with “+”.

---

**Text Classification**

\( \text{page} = \{ 1, \ldots, n \} \)
\( \text{word} = \{ \ldots \} \)
\( \text{topic} = \{ \ldots \} \)

\( \text{Topic}(\text{page,topic!}) \)
\( \text{HasWord}(\text{page,word}) \)
\( \text{Links}(\text{page,page}) \)

\( \text{HasWord}(p,+w) \Rightarrow \text{Topic}(p,+t) \)
\( \text{Topic}(p,t) \land \text{Links}(p,p') \Rightarrow \text{Topic}(p',t) \)
Information Retrieval

\[
\text{InQuery}(\text{word})
\]
\[
\text{HasWord}(\text{page}, \text{word})
\]
\[
\text{Relevant}(\text{page})
\]
\[
\text{Links}(\text{page}, \text{page})
\]
\[
\text{InQuery}(+w) \land \text{HasWord}(p, +w) \implies \text{Relevant}(p)
\]
\[
\text{Relevant}(p) \land \text{Links}(p, p') \implies \text{Relevant}(p')
\]


Record deduplication

Problem: Given database, find duplicate records

\[
\text{HasToken}(\text{token}, \text{field}, \text{record})
\]
\[
\text{SameField}(\text{field}, \text{record}, \text{record})
\]
\[
\text{SameRecord}(\text{record}, \text{record})
\]
\[
\text{HasToken}(+t, +f, r) \land \text{HasToken}(+t, +f, r')
\]
\[
\implies \text{SameField}(f, r, r')
\]
\[
\text{SameField}(+f, r, r') \implies \text{SameRecord}(r, r')
\]
\[
\text{SameRecord}(r, r') \land \text{SameRecord}(r', r'')
\]
\[
\implies \text{SameRecord}(r, r'')
\]

Record resolution

Can also resolve fields:

\[ \text{HasToken(token, field, record)} \]
\[ \text{SameField(field, record, record)} \]
\[ \text{SameRecord(record, record)} \]

\[ \text{HasToken}(+t, +f, r) \land \text{HasToken}(+t, +f, r') \Rightarrow \text{SameField}(f, r, r') \]
\[ \text{SameField}(+f, r, r') \iff \text{SameRecord}(r, r') \]
\[ \text{SameRecord}(r, r') \land \text{SameRecord}(r', r'') \Rightarrow \text{SameRecord}(r, r'') \]
\[ \text{SameField}(f, r, r') \land \text{SameField}(f, r', r'') \Rightarrow \text{SameField}(f, r, r'') \]


Information Extraction

- **Problem:** Extract database from text or semi-structured sources
- **Example:** Extract database of publications from citation list(s) (the “CiteSeer problem”)
- **Two steps:**
  - **Segmentation:**
    Use HMM to assign tokens to fields
  - **Record resolution:**
    Use logistic regression and transitivity
Information Extraction

Token(token, position, citation)
InField(position, field, citation)
SameField(field, citation, citation)
SameCit(citation, citation)

Token(+t,i,c) => InField(i,+f,c)
InField(i,+f,c) <=> InField(i+1,+f,c)
f \neq f' => (!InField(i,+f,c) v !InField(i,+f',c))

Token(+t,i,c) ^ InField(i,+f,c) ^ Token(+t,i',c')
^ InField(i',+f,c') => SameField(+f,c,c')
SameField(+f,c,c') <=> SameCit(c,c')
SameField(f,c,c') ^ SameField(f,c',c'') => SameField(f,c,c'')
SameCit(c,c') ^ SameCit(c',c'') => SameCit(c,c'')