Markov Logic Networks

For Knowledge Graphs
Overview

You will learn about:

- How we can use human-understandable rules (rather than e.g. latent features) for knowledge graphs
- A powerful formalism for combining rules and statistics for analysing knowledge graphs
Markov Networks and Knowledge Graphs
Different SRL models depending on whether random variables $y_{ijk}$ are considered to . . .

- . . . be conditionally independent given latent features associated with subject, object and relation type and additional parameters (latent feature models).
- . . . be conditionally independent given observed graph features and additional parameters (graph feature models).
- . . . have local interactions (Markov Networks).
A random variable $y_{ijk}$ can depend on any of the other $N_e \times N_e \times N_r - 1$ random variables.

→ Very hard to estimate the joint distribution $P(\mathbf{Y})$.

Graphical Models used to encode dependencies between random variables.

Each random variable is a node in such a graph.
Markov Networks

- There are different types of graphical models.
- **Bayesian Networks** represent causal relations.
  - E.g. Smoking $\rightarrow$ Cancer.
  - Bayesian networks are *probabilistic directed acyclic graphical models*.
- **Markov Networks** are useful for applications where relations between variables are symmetrical.
  - Markov networks represent correlations between variables.
  - E.g. ‘‘Bob is a democrat’’ – ‘‘Alice is a democrat’’.
  - When Bob and Alice are friends, these two variables depend on each other, but neither is the cause of the other.
- In knowledge graphs, we often have cyclic dependencies $\rightarrow$ cannot be modelling Bayes Networks, but require *undirected graphical models*. 
A *Markov network* is a set of random variables having a **Markov property** described by an **undirected graph**.

Example Markov network: edges represent dependency (A depends on B and D – B depends on A and D – D depends on A, B, and E – E depends on D and C – C depends on E)

---

1. Image taken from https://en.wikipedia.org/wiki/File:Markov_random_field_example.png
Definition (Cliques)

A clique is a complete subgraph in which every two vertices are connected to each other.

▷ Nodes in a clique are fully connected

Definition (Maximal Clique)

A clique $C$ is maximal if it is not possible to add any other nodes (from the graph) to it without it ceases to be a clique.
An Example of Clique

▷ This graph has five cliques of two nodes:
  - \( \{x_1, x_2\} \), \( \{x_1, x_3\} \), \( \{x_2, x_3\} \), \( \{x_2, x_4\} \), \( \{x_3, x_4\} \).

▷ It has two maximal cliques:
  - \( \{x_1, x_2, x_3\} \), \( \{x_2, x_3, x_4\} \)

▷ The set \( \{x_1, x_2, x_3, x_4\} \) is not a clique because of the missing link from \( x_1 \) to \( x_4 \).
Definition (Factor Potential)

Let $X$ be a set of random variables. We define a factor potential to be a function from values of $X$ to $\mathbb{R}^+$. 

Example for factor potential $\phi$:

<table>
<thead>
<tr>
<th>$X_1$ = Smoking</th>
<th>$X_2$ = Cancer</th>
<th>$\phi(X_1, X_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>4.5</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>4.5</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>2.7</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>4.5</td>
</tr>
</tbody>
</table>
**Factor Potential**

**Definition (Factor Potential)**

Let $X$ be a set of random variables. We define a factor potential to be a function from values of $X$ to $\mathbb{R}^+$. 

Example for factor potential $\phi$:

<table>
<thead>
<tr>
<th>$X_1$ = Cancer</th>
<th>$X_2$ = Asthma</th>
<th>$X_3$ = Cough</th>
<th>$\phi(X_1, X_2, X_3)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
<td>4.5</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>4.5</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
<td>2.7</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>4.5</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>1.5</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
<td>4.5</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td>2.7</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
<td>3.8</td>
</tr>
</tbody>
</table>
Definition of a Markov Network

Definition (Markov Network)

Let \( X = X_1, X_2, \ldots, X_n \) be a set of random variables and \( G \) a graph with vertices \( X \). Let \( C = C_1, C_2, \ldots, C_m \) be the set of all maximal cliques in \( G \). Let \( \phi_1, \phi_2, \ldots, \phi_m \) be factor potentials defined over \( C_1, C_2, \ldots, C_m \) respectively.

\( X \) is a Markov network if:

\[
P(X_1, X_2, \ldots, X_n) = \frac{1}{Z} \phi_1(C_1) \times \phi_2(C_2) \times \cdots \times \phi_m(C_m).
\]

\( Z \) is a normalization constant, called the partition function, defined as

\[
Z = \sum_{X_1, X_2, \ldots, X_n} \phi_1(C_1) \times \phi_2(C_2) \times \cdots \times \phi_m(C_m).
\]

(each \( C_i \) is a subset of \( \{X_1, X_2, \ldots, X_n\} \)).
Example

What are the maximal cliques?

\[ P(X) = \frac{1}{Z} \prod_{C_i \in C} \phi(C_i) \]

Dr. Hamed Shariat Yazdi, Prof. Jens Lehmann

Markov Logic Networks
Example

![Graph: Smoking, Cancer, Asthma, Cough]

\[ P(X) = \frac{1}{Z} \prod_{C_i \in C} \phi(C_i) \]

What are the maximal cliques?

\[ C_1 = \{\text{Smoking, Cancer}\}, \quad C_2 = \{\text{Cancer, Asthma, Cough}\} \]

What is the probability of the world \( w \) in which smoking=true, cancer=false, asthma=false, cough=true?

\[ p(w) = \frac{1}{Z} \phi_1(C_1) \phi_2(C_2) \]

→ in \( \phi_1 \) and \( \phi_2 \) read the corresponding values depending on whether variables are true / false (full example in exercise)
In an undirected graph $G$, being $A$, $B$ and $C$ disjoint subsets of nodes, if every path from $A$ to $B$ includes at least one node from $C$, then $C$ is said to separate $A$ from $B$ in $G$. 

Diagram:

- $A$, $B$, and $C$ as disjoint subsets of nodes in an undirected graph $G$. 
- Every path from $A$ to $B$ includes at least one node from $C$. 

$\implies$ Graph Separation
Definition (Global Markov Independence Property)

Let \( X \) be a Markov network and let \( X_A, X_B, X_C \subseteq X \). We say two subsets of \( X_A \) and \( X_B \) are conditionally independent given a separating subset \( X_C \), where every path from a node in \( X_A \) to a node in \( X_B \) passes through \( X_C \), and we denote it by \((X_A \perp X_B) \mid X_C\)
Definition (Hammersley-Clifford Theorem)

Let $X = \{X_1, X_2, \ldots, X_n\}$ be a set of random variables and $G$ a graph that has $X$ as vertices. If $X$ satisfies the global Markov independence property, then

$$P(X_1, X_2, \ldots, X_n) = \frac{1}{Z} \phi_1(C_1) \times \phi_2(C_2) \times \cdots \times \phi_m(C_m)$$

where $C = \{C_1, C_2, \ldots, C_m\}$ is the set of all maximal cliques in $G$ and $\phi_1, \phi_2, \ldots, \phi_m$ are factor potentials defined over $C_1, C_2, \ldots, C_m$ respectively.

In other terms, $X$ is a Markov network.

Global Markov Independence Property $\equiv$ Factorization Into Potentials.
Example

<table>
<thead>
<tr>
<th>Markov Network</th>
<th>Factorization</th>
<th>Independence</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x_1 )</td>
<td>( \varphi(\mathbf{x}_1) )</td>
<td>none</td>
</tr>
<tr>
<td>( x_2 )</td>
<td>( \varphi(\mathbf{x}_2) )</td>
<td>( \mathbf{x}_2 \perp \mathbf{x}_3</td>
</tr>
<tr>
<td>( x_3 )</td>
<td>( \varphi(\mathbf{x}_3) )</td>
<td>( x_3 \perp {x_1, x_2} )</td>
</tr>
</tbody>
</table>
Example

<table>
<thead>
<tr>
<th>Markov Network</th>
<th>Factorization</th>
<th>Independence</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$\phi(x_1)$</td>
<td>none</td>
</tr>
<tr>
<td>$x_2$</td>
<td>$\phi(x_2)$</td>
<td>none</td>
</tr>
<tr>
<td>$x_3$</td>
<td>$\phi(x_1, x_2)$</td>
<td>none</td>
</tr>
<tr>
<td></td>
<td>$\phi(x_1, x_3)$</td>
<td>none</td>
</tr>
<tr>
<td></td>
<td>$\phi(x_3)$</td>
<td>none</td>
</tr>
<tr>
<td></td>
<td>$x_2 \perp x_3 \mid x_1$</td>
<td>none</td>
</tr>
</tbody>
</table>
### Example

<table>
<thead>
<tr>
<th>Markov Network</th>
<th>Factorization</th>
<th>Independence</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$\phi(x_1, x_2, x_3)$</td>
<td>none</td>
</tr>
<tr>
<td>$x_2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_3$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Markov Logic Networks**
Example

<table>
<thead>
<tr>
<th>Markov Network</th>
<th>Factorization</th>
<th>Independence</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>$x_2$</td>
<td>$x_3$</td>
</tr>
<tr>
<td>$\phi(x_1) \phi(x_2) \phi(x_3)$</td>
<td>all independent</td>
<td></td>
</tr>
</tbody>
</table>
Example

<table>
<thead>
<tr>
<th>Markov Network</th>
<th>Factorization</th>
<th>Independence</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="" alt="Graph" /></td>
<td>$\phi(x_1, x_2, x_3)$</td>
<td>none</td>
</tr>
<tr>
<td><img src="" alt="Graph" /></td>
<td>$\phi(x_1)\phi(x_2)\phi(x_3)$</td>
<td>all independent</td>
</tr>
</tbody>
</table>
Example

<table>
<thead>
<tr>
<th>Markov Network</th>
<th>Factorization</th>
<th>Independence</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$ $x_2$</td>
<td>$\phi(x_1, x_2, x_3)$</td>
<td>none</td>
</tr>
<tr>
<td>$x_3$</td>
<td>$\phi(x_1)\phi(x_2)\phi(x_3)$</td>
<td>all independent</td>
</tr>
<tr>
<td></td>
<td>$\phi(x_1, x_2)\phi(x_1, x_3)$</td>
<td>$x_2 \perp x_3 \mid x_1$</td>
</tr>
</tbody>
</table>
Example

<table>
<thead>
<tr>
<th>Markov Network</th>
<th>Factorization</th>
<th>Independence</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Graph" /></td>
<td>$\phi(x_1, x_2, x_3)$</td>
<td>none</td>
</tr>
<tr>
<td><img src="image" alt="Graph" /></td>
<td>$\phi(x_1)\phi(x_2)\phi(x_3)$</td>
<td>all independent</td>
</tr>
<tr>
<td><img src="image" alt="Graph" /></td>
<td>$\phi(x_1, x_2)\phi(x_1, x_3)$</td>
<td>$x_2 \perp x_3 \mid x_1$</td>
</tr>
</tbody>
</table>
Example

<table>
<thead>
<tr>
<th>Markov Network</th>
<th>Factorization</th>
<th>Independence</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Diagram" /></td>
<td>$\phi(x_1, x_2, x_3)$</td>
<td>none</td>
</tr>
<tr>
<td><img src="image" alt="Diagram" /></td>
<td>$\phi(x_1)\phi(x_2)\phi(x_3)$</td>
<td>all independent</td>
</tr>
<tr>
<td><img src="image" alt="Diagram" /></td>
<td>$\phi(x_1, x_2)\phi(x_1, x_3)$</td>
<td>$x_2 \perp x_3 \mid x_1$</td>
</tr>
<tr>
<td><img src="image" alt="Diagram" /></td>
<td>$\phi(x_1, x_2)\phi(x_3)$</td>
<td>$x_3 \perp {x_1, x_2}$</td>
</tr>
</tbody>
</table>
For a factor potential \( \phi_i(C_i) \) we should look at the truth values contained in clique \( C_i \).

For example consider the graph \( G \) to be the figure on the right.

- We assume that all nodes have binary values of \( T=True, \ F=False \) and the state of \( G \) is \( A, B, D = T \) and \( E, C = F \).
- If \( C_i = \{D, E\} \), then \( \phi_i(C_i) = \phi(D = T, E = F) \), i.e. the (defined) value of the factor potential for the state \( (D = True, E = False) \) should be taken.
In our previous example if our binary variables indicate of having cardiovascular disease (CVD), then we can not just use $\phi(\text{Patient-1 has CVD} = \text{True})$ for every Patient-x, since every patient is different.

Also, we cannot write down $\phi(\text{Patient-x has CVD} = \text{True})$ for every possible Patient-x, since there are infinitely many patients.

We should include side information, or features, regarding each variable and each clique. For example, the medical checkup of Patient-x.

$\phi(\text{Patient-x has CVD} = \text{True})$ is not directly a function of Patient-x, but it is a function of the features of Patient-x.
Clique Features and Log-linear Model

- Each clique $C_i$ can be described by a vector of features $f_i$ (can be more straightforward than large potential tables).
- Feature functions $f_i$ represent clique potential:
  \[ \phi(C_i) = \exp\left( \sum_{j=1}^{k} \omega_{C_i[j]} f_i[j] \right) \]
- Example: feature $f_1$ is 1 if Smoking=True $\land$ Drinking=True and 0 otherwise, with a weight of 3.5.

Formulation as log-linear model:

\[
P(X) = \frac{1}{Z} \prod_{C_i \in C} \phi(C_i) = \frac{1}{Z} \prod_{C_i \in C} \exp\left( \sum_{j=1}^{k} \omega_{C_i[j]} f_i[j] \right)
\]

\[
= \frac{1}{Z} \exp\left( \sum_{i=1}^{m=|C|} \sum_{j=1}^{k} \omega_{C_i[j]} f_i[j] \right)
\]

where $f_i[j]$ is the $j$-th feature of clique $C_i$ and $\omega_{C_i[j]}$ is its weight.
Simple conversion:

- A potential edge in knowledge graph becomes a node in the Markov Network.
- All nodes are connected $\rightarrow$ calculating joint distribution not tractable.
Knowledge Graphs and Markov Networks

Simple conversion:

▷ A potential edge in knowledge graph becomes a node in the Markov Network.

▷ All nodes are connected → calculating joint distribution not tractable.

→ Use type constraints to reduce number of nodes.
→ Use template mechanism for conditional independencies.
Hard Type Constraints

Consider the following scenario:

- Two types of entities/constants: adults and children.
- Two relations: parentOf and marriedTo.
- Hard type constraints (to reduce nodes in Markov Network):
  
  - \( \forall x, y \) marriedTo \((x, y) = \Rightarrow \neg \text{parentOf}(y, x) \) (you cannot marry your child).
  
  - \( \forall x, y \) \( \neg \text{adult}(x) = \Rightarrow \neg \text{parentOf}(x, y) \) (only adults can be parents of children).
  
  - \( \forall x \) \( \neg \text{marriedTo}(x, x) \) (you cannot marry yourself).
  
  - \( \forall x \) \( \neg \text{parentOf}(x, x) \) (you are not your own parent).
Hard Type Constraints

Consider the following scenario:

- Two types of entities/ constants: adults and children.
- Two relations: parentOf and marriedTo.
- Hard type constraints (to reduce nodes in Markov Network):
  - $\forall x, y\ marriedTo(x, y) \implies \neg parentOf(y, x)$ (you cannot marry your child).
  - $\forall x, y\ \neg adult(x) \implies \neg parentOf(x, y)$ (only adults can be parents of children).
  - $\forall x\ \neg marriedTo(x, x)$ (you cannot marry yourself).
  - $\forall x\ \neg parentOf(x, x)$ (you are not your own parent).
Hard Type Constraints

- Red edges indicate potential \texttt{marriedTo} relation.
- Blue edges indicate potential \texttt{parentOf} relation.
First-Order Logic and Markov Logic Networks
First-Order Logic (in a nutshell)

- **Constants**: symbols representing objects in the domain, e.g. John, Mary, ...
- **Variables**: symbols which take objects in the domain as values, e.g. $x, y, \ldots$
- **Functions**: symbols which mapping tuples of objects to objects, e.g. BandOf(.), ActedIn(.), ParentOf(.), etc.
- **Predicates**: symbols representing relations among objects or object attributes. Predicates are Boolean-valued functions i.e. they return True or False. e.g. Singer(.), SangTogether(.,.), etc.
- **Literal**: a predicate or its negation.
- **Clause**: disjunction of literals.
First-Order Logic (in a nutshell)

- **Connectives**: ¬ (negation), ∧ (conjunction), ∨ (disjunction), ⇒ (implication), ⇔ (equivalence).
- **Quantifiers**: ∃ (existential quantifier), ∀ (universal quantifier).
- **Formulas**: (roughly speaking) is a combination of constants, variables, functions and predicates using connectors and quantifiers.
- **Grounding**: replacing all variables by constants in a formula, e.g. \( \text{Smokes}(x) \Rightarrow \text{Cancer}(x) \Leftrightarrow \text{Smokes}(\text{John}) \Rightarrow \text{Cancer}(\text{John}) \).

In FOL we need to
- specify a domain of interest (e.g. humans, real numbers etc.).
- assign a formula to constants, function symbols and predicate symbols in the domain of interest.
First-Order Logic (in a nutshell)

▷ Interpretation: an Interpretation is\(^2\) a pair \(I = \langle D, A \rangle\) where:
   - \(D\) is a non empty set called domain of \(I\).
   - \(A\) is a function that maps:
     ▷ every constant symbol \(c\) into an element \(c^A \in D\);
     ▷ every n-ary function symbol \(f\) into a function \(f^A : D^n \rightarrow D\);
     ▷ every n-ary predicate symbol \(P\) into a n-ary relation \(P^A : D^n \rightarrow \{\text{True}, \text{False}\}\).

▷ Model: an interpretation is called a model (of a set of formulas), when all the formulas are true in the interpretation.

▷ World: is the assignment of truth values to all ground predicates.

First-Order Logic: Interpretation Example

$$\forall x \exists y P(x, y)$$

As an example\(^3\), we can interpret the above formula as:

- **\(D\) is the set of human beings.**
  - \(P^A(a, b) = \text{True} \iff b \text{ is father of } a\).
  - i.e. all human beings have a father.

- **\(D\) is the set of human beings.**
  - \(P^A(a, b) = \text{True} \iff b \text{ is mother of } a\).
  - i.e. all human beings have a mother.

- **\(D\) is the set of natural numbers.**
  - \(P^A(a, b) = \text{True} \iff a < b\).
  - i.e. for every natural number there is a greater one.

---

Markov Logic Network: Intuition

- A FOL Knowledge Base is a set of clauses.
- A logical KB is a set of hard constraints on the set of possible worlds.
- World = assignment of truth values to all ground predicates e.g. marriedTo(Anna,Bob) is assigned to a truth value.
- Lets make them soft constraints: When a world violates a formula, it becomes less probable, not impossible.
A FOL Knowledge Base is a set of clauses.

A logical KB is a set of hard constraints on the set of possible worlds.

World = assignment of truth values to all ground predicates
e.g. marriedTo(Anna,Bob) is assigned to a truth value.

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 ⇒ stronger constraint)

\[ P(X = x) \propto \exp \left( \sum \text{weights of formulas that are true in } x \right). \]

Probability of a world \( x \) is proportional to the exponentiated sum of the weights of the formulas, which satisfy it.
Example: Friends and Smokers

Smoking causes cancer.
Friends have similar smoking habits.

∀x Smokes(x) ⇒ Cancer(x)
∀x, y Friends(x,y) ⇒ (Smokes(x) ⇔ Smokes(y))
Example: Friends and Smokers

\[ \forall x \text{ Smokes}(x) \Rightarrow \text{Cancer}(x) \]

\[ \forall x, y \text{ Friends}(x,y) \Rightarrow (\text{Smokes}(x) \iff \text{Smokes}(y)) \]
Example: Friends and Smokers

1.5) ∀x Smokes(x) ⇒ Cancer(x)
1.1) ∀x, y Friends(x,y) ⇒ (Smokes(x) ⇔ Smokes(y))
Example: Friends and Smokers

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

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

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

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

1.5) \( \forall x \) Smokes\( (x) \) \( \Rightarrow \) Cancer\( (x) \)
1.1) \( \forall x, y \) Friends\( (x,y) \) \( \Rightarrow \) (Smokes\( (x) \) \( \iff \) Smokes\( (y) \))

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

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

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

1.5) $\forall x \ Smokes(x) \Rightarrow Cancer(x)$
1.1) $\forall x, y \ Friends(x,y) \Rightarrow (Smokes(x) \iff Smokes(y))$

Two constants: Anna (A) and Bob (B)
A Markov Logic Network (MLN) $L$ is a set of pairs $(F_i, \omega_i)$ where

- $F_i$ is a formula in first-order logic.
- $\omega_i \in \mathbb{R}$ is a real number representing how “true” the formula is.

Together with a set of constants $C$, $L$ defines a Markov network $M_{L,C}$ (using the log linear representation) as follows:

- $M_{L,C}$ contains a boolean node for each basic atom. There is an edge between two nodes of $M_{L,C}$ iff the corresponding ground predicates appear together in at least one grounding of one formula in $L$.
- $M_{L,C}$ contains a feature for each possible instantiation of every formula $(F_j, \omega_j) \in L$. For a possible world $x$, the value of that feature is 1 if it is satisfied in $x$, and 0 if not.
MLN is template for ground Markov nets.

Probability of a world $x$:

$$P(X = x) = \frac{1}{Z} \exp \left( \sum_{i} \omega_i n_i(x) \right)$$

$\omega_i$: weight of formula $i$.

$n_i$: number of true groundings of formula $i$ in $x$.

Formula is generally true $\iff$ all groundings are true.

Typed variables and constants greatly reduce size of ground Markov net.
Counting True Groundings

- Soft constraint/MLN formula:
  \[ \forall x, y, z \ parentOf(x, z) \land parentOf(y, z) \implies marriedTo(x, y) \]

- True grounding:
  \[ parentOf(Anna, Charles), parentOf(Bob, Charles), marriedTo(Anna, Bob) \]

- Possible world = KG:
How to use MLNs for knowledge graphs?

Typical workflow for applying MLNs to a knowledge graph:

- Obtain domain knowledge / domain expert.
- Create MLN formulas (alternatively: learn formulas).
- Perform weight learning of MLN formulas with respect to knowledge graph.
- Perform inference tasks.
Most common inference tasks:

- Estimating the most probable state of the world:

\[
\arg\max_y P(y|x)
\]

where \( y \) is the query and \( x \) is the evidence.

- Conditional probabilities for a formula given a knowledge graph.

MLN acts as a template for a Markov network \( \rightarrow \) Markov Network Inferencing Methods can be applied.

Often much faster is to combine probabilistic methods with ideas from logical inference (not covered here).
Inference in MLNs: MAP Problem

\[
\arg \max_y P(y|x) = \arg \max_y \frac{1}{\mathcal{Z}_x} \exp \left( \sum_i \omega_i n_i(x, y) \right) = \arg \max_y \sum_i \omega_i n_i(x, y)
\]

in which \( n_i \) is the number of true groundings of formula \( i \) in \( x \) extended by \( y \).

- MAP Problem: finding the most probable state of the world \( y \) given facts / knowledge graph \( x \) (also called evidence).
- Reduces to finding the truth assignment that maximizes the sum of weights of satisfied clauses.
- Any weighted satisfiability solver can be used.
- NP hard.
“What is the probability that formula $F_1$ holds given that $F_2$ does?”

$$P(F_1|F_2, M_L, C) = \frac{P(F_1 \land F_2|M_L, C)}{P(F_2|M_L, C)}$$

$$= \frac{\sum_{x \in X_{F_1} \cap X_{F_2}} P(X = x|M_L, C)}{\sum_{x \in X_{F_2}} P(X = x|M_L, C)}$$

in which:

$F_1$: query (e.g. “Smokes(x) $\implies$ Cancer(x)”?),

$F_2$: evidence (e.g. all facts in the knowledge graph),

$M_L, C$: Markov logic network,

$X_{F_i}$: set of worlds where $F_i$ holds.

Methods for efficiently computing this not part of the lecture.
Learning in MLNs

- **Weight learning**: learn weights of formulas / facts
  - Given the knowledge graph, we want to estimate weights of Markov logic formulas (e.g. verify whether the data supports the claim that smoking causes cancer).
  - Generative weight learning: maximise likelihood of observed knowledge graph. Recall that in generative learning, we learn the joint probability distribution $P(X, Y)$.
  - Discriminative learning setting: divide between evidence facts and queries facts. Recall that in discriminative learning, we learn the conditional probability distribution $P(Y|X)$.

- **Structure learning**: learn structure of formulas.
  - Not part of this lecture.

---

Derivative of the log-likelihood of a formula with respect to its weight is given by:

$$\frac{\partial}{\partial \omega} \log P_\omega(X = x) = n_i(x) - \sum_{x'} P_\omega(X = x') n_i(x')$$

- $n_i(x)$: true groundings of the $i$-th formula in the knowledge graph.
- $x'$: all possible knowledge graphs.
- $i$-th component of the gradient is the difference between the number of true groundings in the knowledge graph and its expectation according to the model.
Question: How the previous derivative is computed?

Note that, likelihood and log-likelihood of a world \( x \) (or a knowledge graph) are respectively as follows:

\[
P_\omega(X = x) = \frac{1}{Z} \exp \left( \sum_j \omega_j n_j(x) \right),
\]

\[
\log P_\omega(X = x) = \sum_j \omega_j n_j(x) - \log Z.
\]

Recall that \( Z \) normalizes the above likelihood, so:

\[
Z = \sum_{x'} \exp \left( \sum_j \omega_j n_j(x') \right)
\]

where \( x' \) is a possible world (or a knowledge graph).
Therefore we have:

\[
\frac{\partial}{\partial \omega_i} \log P_\omega(X = x) = \frac{\partial}{\partial \omega_i} \left( \sum_j \omega_j n_j(x) - \log Z \right)
\]

\[
= n_i(x) - \frac{1}{Z} \frac{\partial}{\partial \omega_i} Z
\]

\[
= n_i(x) - \frac{1}{Z} \sum_{x'} \frac{\partial}{\partial \omega_i} \exp \left( \sum_j \omega_j n_j(x') \right)
\]

\[
= n_i(x) - \frac{1}{Z} \sum_{x'} \exp \left( \sum_j \omega_j n_j(x') \right) n_i(x')
\]

\[
= n_i(x) - \sum_{x'} P_\omega(X = x') n_i(x')
\]

◮ Note: the derivative of \( \omega_j n_j(x) \) with respect to \( \omega_i \) is zero for \( j \neq i \).
Markov networks can handle the case that we consider all random variables for facts in the knowledge graph to be potentially statistically dependant.

Inference in Markov networks intractable → need to restrict considered nodes and edges by constraints.

Markov Logic provides template for Markov Networks.

MLN Inference can be used to find most probably worlds and answer queries.

MLN Learning can be used to find weights and rules.
Markov Logic Networks use First-Order Logic for those constraints

- **Advantages:**
  - Unification of logics and statistics - one of the premier approaches to powerful AI in general.
  - Rules are human readable (in contrast to latent features).

- **Disadvantages:**
  - Weights for MLN formulas are not very intuitive (no probabilistic interpretation).
  - Scalability problem as result of potentially dense Markov network - many approaches to tackle this.


Abdeslam Boularias, Course 16:198:520, Introduction To Artificial Intelligence, Lecture 9, Markov Networks.