Relational Learning

CS 760: Machine Learning
Spring 2018
Mark Craven and David Page

www.biostat.wisc.edu/~craven/cs760
Goals for the lecture

you should understand the following concepts

- relational learning
- the FOIL algorithm
- plate models
Relational learning example

consider the task of learning a *pharmacophore*: the substructure of a molecule that interacts with a target of interest

- instances for this task consist of interacting (+) and non-interacting molecules (-)

To represent each instance, we’d like to describe:

- the (variable # of) atoms in the molecule
- the possible conformations of the molecule
- the bonds among atoms
- distances among atoms
- etc.
Relational learning example

[Finne et al., *Machine Learning* 1998]

A multi-relational representation for molecules

<table>
<thead>
<tr>
<th>Molecule</th>
<th>Target_1</th>
<th>...</th>
<th>Target_n</th>
</tr>
</thead>
<tbody>
<tr>
<td>mol1</td>
<td>inactive</td>
<td></td>
<td>inactive</td>
</tr>
<tr>
<td>mol2</td>
<td>active</td>
<td></td>
<td>inactive</td>
</tr>
</tbody>
</table>

Molecular Bioactivity

<table>
<thead>
<tr>
<th>Molecule</th>
<th>Bond_ID</th>
<th>Atom_1_ID</th>
<th>Atom_2_ID</th>
<th>Bond_Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>mol1</td>
<td>bond1</td>
<td>a1</td>
<td>a2</td>
<td>aromatic</td>
</tr>
</tbody>
</table>

Bonds

<table>
<thead>
<tr>
<th>Molecule</th>
<th>Conformer</th>
<th>Atom_ID</th>
<th>Atom_Type</th>
<th>XCoordinate</th>
<th>YCoordinate</th>
<th>ZCoordinate</th>
</tr>
</thead>
<tbody>
<tr>
<td>mol1</td>
<td>confl1</td>
<td>a1</td>
<td>carbon</td>
<td>2.58</td>
<td>-1.23</td>
<td>0.69</td>
</tr>
</tbody>
</table>

3D Atom Locations
Relational learning example

[Finn et al., *Machine Learning* 1998]

a learned relational rule characterizing ACE inhibitors

Molecule A is an ACE inhibitor if for some conformer Conf of A:
- molecule A contains a zinc binding site B;
- molecule A contains a hydrogen acceptor C;
- the distance between B and C in Conf is 7.9 +/- .75;
- molecule A contains a hydrogen acceptor D;
- the distance between B and D in Conf is 8.5 +/- .75;
- the distance between C and D in Conf is 2.1 +/- .75;
- molecule A contains a hydrogen acceptor E;
- the distance between B and E in Conf is 4.9 +/- .75;
- the distance between C and E in Conf is 3.1 +/- .75;
- the distance between D and E in Conf is 3.8 +/- .75.
Relational representation

\[
\text{ACE_inhibitor}(A) \leftarrow \text{has_zinc_binding_site}(A, B) \land \\
\text{has_hydrogen_acceptor}(A, C) \land \\
\text{distance}(B, C, 7.9, 0.75) \land \\
\text{has_hydrogen_acceptor}(A, D) \land \\
\text{distance}(B, D, 8.5, 0.75) \land \\
\text{distance}(C, D, 8.5, 0.75) \land \\
\text{has_hydrogen_acceptor}(A, E) \land \\
\text{distance}(B, E, 4.9, 0.75) \land \\
\text{distance}(C, E, 3.1, 0.75) \land \\
\text{distance}(D, E, 3.8, 0.75)
\]

To learn an equivalent rule with a feature-vector learner, what features would we need to represent?

- \text{has_zinc_binding_site}
- \text{has_hydrogen_acceptor}
- \text{zinc_binding_site_and_hydrogen_acceptor_distance}
- \text{hydrogen_acceptor_hydrogen_acceptor_distance}

... can easily encode distance between a pair of atoms; but this pharmacophore has 4 important atoms with 6 relevant distances among them
Relational learning example
[Page et al., AAAI 2012]

- Data from electronic health records (EHRs) is being used to learn models for risk assessment, adverse event detection, etc.
- A patient’s record is described by multiple tables in a relational DB

<table>
<thead>
<tr>
<th>demographics</th>
<th>diagnoses</th>
</tr>
</thead>
<tbody>
<tr>
<td>PatientID</td>
<td>PatientID</td>
</tr>
<tr>
<td>Gender</td>
<td>Date</td>
</tr>
<tr>
<td>Birthdate</td>
<td>Physician</td>
</tr>
<tr>
<td>P1</td>
<td>P1</td>
</tr>
<tr>
<td>M</td>
<td>1/1/01</td>
</tr>
<tr>
<td>3/22/63</td>
<td>Smith</td>
</tr>
<tr>
<td></td>
<td>Jones</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>labs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PatientID</td>
</tr>
<tr>
<td>P1</td>
</tr>
<tr>
<td>P1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>genetics</th>
</tr>
</thead>
<tbody>
<tr>
<td>PatientID</td>
</tr>
<tr>
<td>P1</td>
</tr>
<tr>
<td>P2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>drugs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PatientID</td>
</tr>
<tr>
<td>P1</td>
</tr>
</tbody>
</table>
Relational learning example

• suppose we want to learn the general concept of \( \text{can-reach} \) in a graph, given a set of training instances describing a particular graph.

\[
\begin{align*}
\text{\( \oplus \):} & \quad \langle 0,1 \rangle \quad \langle 0,2 \rangle \quad \langle 0,3 \rangle \quad \langle 0,4 \rangle \quad \langle 0,5 \rangle \quad \langle 0,6 \rangle \quad \langle 0,8 \rangle \quad \langle 1,2 \rangle \quad \langle 3,2 \rangle \quad \langle 3,4 \rangle \\
& \quad \langle 3,5 \rangle \quad \langle 3,6 \rangle \quad \langle 3,8 \rangle \quad \langle 4,5 \rangle \quad \langle 4,6 \rangle \quad \langle 4,8 \rangle \quad \langle 6,8 \rangle \quad \langle 7,6 \rangle \quad \langle 7,8 \rangle \\
\text{\( \ominus \):} & \quad \langle 0,0 \rangle \quad \langle 0,7 \rangle \quad \langle 1,0 \rangle \quad \langle 1,1 \rangle \quad \langle 1,3 \rangle \quad \langle 1,4 \rangle \quad \langle 1,5 \rangle \quad \langle 1,6 \rangle \quad \langle 1,7 \rangle \quad \langle 1,8 \rangle \\
& \quad \langle 2,0 \rangle \quad \langle 2,1 \rangle \quad \langle 2,2 \rangle \quad \langle 2,3 \rangle \quad \langle 2,4 \rangle \quad \langle 2,5 \rangle \quad \langle 2,6 \rangle \quad \langle 2,7 \rangle \quad \langle 2,8 \rangle \quad \langle 3,0 \rangle \\
& \quad \langle 3,1 \rangle \quad \langle 3,3 \rangle \quad \langle 3,7 \rangle \quad \langle 4,0 \rangle \quad \langle 4,1 \rangle \quad \langle 4,2 \rangle \quad \langle 4,3 \rangle \quad \langle 4,4 \rangle \quad \langle 4,7 \rangle \quad \langle 5,0 \rangle \\
& \quad \langle 5,1 \rangle \quad \langle 5,2 \rangle \quad \langle 5,3 \rangle \quad \langle 5,4 \rangle \quad \langle 5,5 \rangle \quad \langle 5,6 \rangle \quad \langle 5,7 \rangle \quad \langle 5,8 \rangle \quad \langle 6,0 \rangle \quad \langle 6,1 \rangle \\
& \quad \langle 6,2 \rangle \quad \langle 6,3 \rangle \quad \langle 6,4 \rangle \quad \langle 6,5 \rangle \quad \langle 6,6 \rangle \quad \langle 6,7 \rangle \quad \langle 7,0 \rangle \quad \langle 7,1 \rangle \quad \langle 7,2 \rangle \quad \langle 7,3 \rangle \\
& \quad \langle 7,4 \rangle \quad \langle 7,5 \rangle \quad \langle 7,7 \rangle \quad \langle 8,0 \rangle \quad \langle 8,1 \rangle \quad \langle 8,2 \rangle \quad \langle 8,3 \rangle \quad \langle 8,4 \rangle \quad \langle 8,5 \rangle \quad \langle 8,6 \rangle \\
& \quad \langle 8,7 \rangle \quad \langle 8,8 \rangle 
\end{align*}
\]

• how would you represent this task to a learner?
Relational learning example

- a relational representation, such as first-order logic, can capture this concept succinctly and in a general way

\[
\text{can-reaching}(X_1, X_2) \leftarrow \text{linked-to}(X_1, X_2)
\]

\[
\text{can-reaching}(X_1, X_2) \leftarrow \text{linked-to}(X_1, X_3) \land \text{can-reaching}(X_3, X_2)
\]
The FOIL algorithm for relational learning
[Quinlan, *Machine Learning* 1990]

given:
- tuples (instances) of a target relation
- *extensionally* represented background relations

do:
- learn a set of rules that (mostly) cover the positive tuples of the target relation, but not the negative tuples
Input to FOIL

- instances of target relation

\[
\oplus: \{ (0,1), (0,2), (0,3), (0,4), (0,5), (0,6), (0,8), (1,2), (3,2), (3,4), \\
(3,5), (3,6), (3,8), (4,5), (4,6), (4,8), (6,8), (7,6), (7,8) \}
\]

\[
\ominus: \{ (0,0), (0,7), (1,0), (1,1), (1,3), (1,4), (1,5), (1,6), (1,7), (1,8), \\
(2,0), (2,1), (2,2), (2,3), (2,4), (2,5), (2,6), (2,7), (2,8), (3,0), \\
(3,1), (3,3), (3,7), (4,0), (4,1), (4,2), (4,3), (4,4), (4,7), (5,0), \\
(5,1), (5,2), (5,3), (5,4), (5,5), (5,6), (5,7), (5,7), (6,0), (6,1), \\
(6,2), (6,3), (6,4), (6,5), (6,6), (6,7), (7,0), (7,1), (7,2), (7,3), \\
(7,4), (7,5), (7,7), (8,0), (8,1), (8,2), (8,3), (8,4), (8,5), (8,6), \\
(8,7), (8,8) \}
\]

- extensionally defined background relations

\[
\text{linked-to} = \{ (0,1), (0,3), (1,2), (3,2), (3,4), \\
(4,5), (4,6), (6,8), (7,6), (7,8) \}
\]
The FOIL algorithm for relational learning

FOIL uses a covering approach to learn a set of rules

\text{LEARNRULESET}(\text{set of tuples } T \text{ of target relation, background relations } B) \{
S = \{ \}
\text{repeat}
\hspace{1em} R \leftarrow \text{LEARNRULE}(T, B)
\hspace{1em} S \leftarrow S \cup R
\hspace{1em} T \leftarrow T \setminus \text{positive tuples covered by } R
\hspace{1em} \text{until there are no (few) positive tuples left in } T
\text{return } S
\}

The FOIL algorithm for relational learning

**LEARNRULE**(set of tuples $T$ of target relation, background relations $B$)

\[
\begin{align*}
R &= \{ \} \\
\text{repeat} & \\
L &\leftarrow \text{best literal, based on } T \text{ and } B, \text{ to add to right-hand side of } R \\
R &\leftarrow R \cup L \\
T &\leftarrow \text{new set of tuples that satisfy } L \\
\text{until there are no (few) negative tuples left in } T \\
\text{return } R
\end{align*}
\]
Literals in FOIL

- Given the current rule $R(X_1, X_2, ... X_k) \leftarrow L_1 \land L_2 \land ... \land L_n$
- FOIL considers adding several types of literals

- $X_j = X_k$
  - both $X_j$ and $X_k$ either appear in the LHS of the rule, or were introduced by a previous literal

- $X_j \neq X_k$

- $Q(V_1, V_2, ... V_a)$
  - at least one of the $V_i$'s has to be in the LHS of the rule, or was introduced by a previous literal

- $\neg Q(V_1, V_2, ... V_a)$

where $Q$ is a background relation
Literals in FOIL (continued)

\[ X_j = c \]

where \( c \) is a constant

\[ X_j \neq c \]

\[ X_j > a \]

where \( X_j \) and \( X_k \) are numeric variables and \( a \)

\[ X_j \leq a \]

is a numeric constant

\[ X_j > X_k \]

\[ X_j \leq X_k \]
• suppose we want to learn rules for the target relation \( \text{can-reach}(X_1, X_2) \)
• we’re given instances of the target relation from the following graph

\[
\begin{array}{c}
\text{linked-to} = \{ \langle 0,1 \rangle, \langle 0,3 \rangle, \langle 1,2 \rangle, \langle 3,2 \rangle, \langle 3,4 \rangle, \\
\langle 4,5 \rangle, \langle 4,6 \rangle, \langle 6,8 \rangle, \langle 7,6 \rangle, \langle 7,8 \rangle \}
\end{array}
\]
the first rule learned covers 10 of the positive instances
  $\text{can-reach}(X_1, X_2) \leftarrow \text{linked-to}(X_1, X_2)$

the second rule learned covers the other 9 positive instances
  $\text{can-reach}(X_1, X_2) \leftarrow \text{linked-to}(X_1, X_3) \land \text{can-reach}(X_3, X_2)$

note that these rules generalize to other graphs
Evaluating literals in FOIL

- **FOIL** evaluates the addition of a literal $L$ to a rule $R$ by

  \[ FOIL\_Gain(L,R) = t \left( \log_2 \frac{p_1}{p_1 + n_1} - \log_2 \frac{p_0}{p_0 + n_0} \right) \]

  where

  - $p_0 = \#$ of positive tuples covered by $R$
  - $n_0 = \#$ of negative tuples covered by $R$
  - $p_1 = \#$ of positive tuples covered by $R \land L$
  - $n_1 = \#$ of negative tuples covered by $R \land L$
  - $t = \#$ of positive of tuples of $R$ also covered by $R \land L$

- like information gain, but takes into account
  - we want to cover positives, not just get a more “pure” set of tuples
  - the size of the tuple set grows as we add new variables
Evaluating literals in FOIL

\[
\text{FOIL\_Gain}(L, R) = t \left( \text{Info}(R_0) - \text{Info}(R_1) \right)
\]

- where \( R_0 \) represents the rule without \( L \) and \( R_1 \) is the rule with \( L \) added
- \( \text{Info}(R_i) \) is the number of bits required to encode a positive in the set of tuples covered by \( R_i \)

\[
\text{Info}(R_i) = -\log_2 \left( \frac{p_i}{p_i + n_i} \right)
\]
Recall this example

- **Definition of can-reach:**

\[
\text{can-reach}(X_1, X_2) \leftarrow \text{linked-to}(X_1, X_2)
\]

\[
\text{can-reach}(X_1, X_2) \leftarrow \text{linked-to}(X_1, X_3) \land \text{can-reach}(X_3, X_2)
\]
consider the first step in learning the second clause

can-reach(\(X_1, X_2\)) ←

can-reach(\(X_1, X_2\)) ← linked-to(\(X_1, X_3\))

\[
\text{\textsc{foil}} \text{ \textit{gain}}(L, R) = 9 \left( \log_2 \frac{18}{18 + 54} - \log_2 \frac{9}{9 + 62} \right) = 8.8
\]
Alternative: Plates

A rectangle labeled “N” denotes N copies of the Bayes net inside. Typically arcs go into rectangles, but we relax to allow outgoing next…

https://upload.wikimedia.org/wikipedia/commons/d/d3/Latent_Dirich...
Alternative: Probabilistic Relational Models (PRMs)
PRM Dependency Structure for the University Domain

Edges from one class to another are routed through slot-chains.

- **Professor**
  - **Teaching-Ability**
  - **Popularity**

- **Course**
  - **Rating**
  - **Difficulty**

- **Registration**
  - **Satisfaction**
  - **Grade**

- **Student**
  - **Intelligence**
  - **Ranking**

Edges correspond to probabilistic dependency for objects in that class.

Edges from one class to another are routed through slot-chains.
The student’s ranking depends on the average of his grades.

A course rating depends on the average satisfaction of students in the course.

A student may take multiple courses.
CPDs in PRMs

Professor
- Teaching-Ability
- Popularity

Course
- Rating
- Difficulty

Registration
- Satisfaction
- Grade

Student
- Intelligence
- Ranking

<table>
<thead>
<tr>
<th>D.I</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>h,h</td>
<td>0.5</td>
<td>0.4</td>
<td>0.1</td>
</tr>
<tr>
<td>h,l</td>
<td>0.1</td>
<td>0.5</td>
<td>0.4</td>
</tr>
<tr>
<td>l,h</td>
<td>0.8</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>l,l</td>
<td>0.3</td>
<td>0.6</td>
<td>0.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>avg</th>
<th>l</th>
<th>m</th>
<th>h</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.1</td>
<td>0.2</td>
<td>0.7</td>
</tr>
<tr>
<td>B</td>
<td>0.2</td>
<td>0.4</td>
<td>0.4</td>
</tr>
<tr>
<td>C</td>
<td>0.6</td>
<td>0.3</td>
<td>0.1</td>
</tr>
</tbody>
</table>
Alternative: Markov logic

- A logical knowledge base 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 → stronger constraint).

\[
P(\text{world}) \propto \exp\left(\sum \text{weights of formulas it satisfies}\right)
\]
MLN 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 \(w\)
MLN example: friends & smokers

Smoking causes cancer.
Friends have similar smoking habits.
MLN example: friends & smokers

\[ \forall x \ Smokes(x) \Rightarrow Cancer(x) \]
\[ \forall x, y \ Friends(x, y) \Rightarrow (Smokes(x) \Leftrightarrow Smokes(y)) \]
MLN example: friends & smokers

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.5</td>
<td>$\forall x \ Smokes(x) \Rightarrow Cancer(x)$</td>
</tr>
<tr>
<td>1.1</td>
<td>$\forall x, y \ Friends(x, y) \Rightarrow (\ Smokes(x) \Leftrightarrow Smokes(y) )$</td>
</tr>
</tbody>
</table>
MLN example: friends & smokers

Two constants: **Anna** (A) and **Bob** (B)
MLN example: friends & smokers

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

Two constants: **Anna** (A) and **Bob** (B)
MLN example: friends & smokers

Two constants: **Anna** (A) and **Bob** (B)
MLN 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)
MLN example: friends & 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)
Markov logic networks

- a MLN is a **template** for ground Markov nets
  - the logic determines the form of the cliques
  - but if we had one more constant (say, **Larry**), we’d get a different Markov net

- we can determine the probability of a world $v$ (assignment of truth values to ground predicates) by

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

- weight of formula
- # of true groundings of formula $i$ in $v$
Example translated to Markov Network

• **Facts:**
  - `smokes(Mary)`
  - `smokes(Joe)`
  - `friend(Joe, Mary)`

• **Rules:**
  - `Friend(x, y) ∧ Smokes(x) → Smokes(y)`

Graph:
- `friend(Joe, Mary)`
- `smokes(Mary)`
- `smokes(Joe)`
- `friend(Mary, Joe)`
- `other facts, like cancer(Mary)`
- `other facts, like cancer(Joe)`
Computing weights

Consider the effect of rule R: \( \text{Friends}(x,y) \land \text{Smokes}(x) \rightarrow \text{Smokes}(y) \)

<table>
<thead>
<tr>
<th>\text{smokes(Mary)}</th>
<th>\neg\text{smokes(Mary)}</th>
<th>\text{smokes(Joe)}</th>
<th>\neg\text{smokes(Joe)}</th>
</tr>
</thead>
<tbody>
<tr>
<td>smokes(Joe)</td>
<td>\neg\text{smokes(Joe)}</td>
<td>\text{smokes(Joe)}</td>
<td>\neg\text{smokes(Joe)}</td>
</tr>
<tr>
<td>( e^{1.1} )</td>
<td>1</td>
<td>( e^{1.1} )</td>
<td>( e^{1.1} )</td>
</tr>
<tr>
<td>( e^{1.1} )</td>
<td>( e^{1.1} )</td>
<td>( e^{1.1} )</td>
<td>( e^{1.1} )</td>
</tr>
</tbody>
</table>

This is the only setting that does not satisfy R. Thus, it is given value 1, while the others are given value \( \exp(\text{weight}(R)) \).
Probability of a world in an MLN

$$v = \begin{bmatrix}
\text{Friends}(A,A) = T \\
\text{Friends}(A,B) = T \\
\text{Friends}(B,A) = T \\
\text{Friends}(B,B) = T \\
\text{Smokes}(A) = F \\
\text{Smokes}(B) = T \\
\text{Cancer}(A) = F \\
\text{Cancer}(B) = F 
\end{bmatrix}$$

$$\forall x \, \text{Smokes}(x) \Rightarrow \text{Cancer}(x)$$

$$\begin{array}{ll}
x = A & T \\
x = B & F \\
n_1(v) = 1
\end{array}$$

$$\forall x, y \, \text{Friends}(x,y) \Rightarrow (\text{Smokes}(x) \Leftrightarrow \text{Smokes}(y))$$

$$\begin{array}{llll}
x = A, y = A & T \\
x = A, y = B & F \\
x = B, y = A & F \\
x = B, y = B & T \\
n_2(v) = 2
\end{array}$$

# of true groundings of formula 1 in $v$
Probability of a world in an MLN

\[ v = \begin{cases} 
\text{Friends}(A,A) = T \\
\text{Friends}(A,B) = T \\
\text{Friends}(B,A) = T \\
\text{Friends}(B,B) = T \\
\text{Smokes}(A) = F \\
\text{Smokes}(B) = T \\
\text{Cancer}(A) = F \\
\text{Cancer}(B) = F 
\end{cases} \]

\[
P(v) = \frac{1}{Z} \exp \left( \sum_i w_i n_i(v) \right)
\]

\[
= \frac{1}{Z} \exp \left( 1.5(1) + 1.1(2) \right)
\]
Three MLN tasks

• inference: can use the toolbox of inference methods developed for ordinary Markov networks
  – Markov chain Monte Carlo methods, including persistent contrastive divergence (PCD) where we iterate MCMC and gradient ascent steps (don’t wait for MCMC to converge)
  – belief propagation
  – variational methods
    in tandem with weighted SAT solver
    (e.g., MaxWalkSAT [Kautz et al., 1997] )

• parameter learning

• structure learning: can use ordinary relational learning methods like FOIL or other ILP algorithms to learn new formula
MLN learning tasks

- the input to the learning process is a relational database of ground atoms

<table>
<thead>
<tr>
<th>Friends$(x, y)$</th>
<th>Smokes$(x)$</th>
<th>Cancer$(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Anna, Anna</td>
<td>Bob</td>
<td>Bob</td>
</tr>
<tr>
<td>Anna, Bob</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bob, Anna</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bob, Bob</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- the closed world assumption is used to infer the truth values of atoms not present in the DB
Parameter learning

- parameters (weights on formulas) can be learned using gradient ascent

\[
\frac{\partial}{\partial \omega_i} \log P_w(v) = n_i(v) - E_w[n_i(v)]
\]

- approximation methods may be needed to estimate both terms

# of times clause \( i \) is true in data

Expected # times clause \( i \) is true according to MLN
MLN experiment

- testbed: a DB describing Univ. of Washington CS department
  - 12 predicates
    - Professor(person)
    - Student(person)
    - Area(x, area)
    - AuthorOf(publication, person)
    - AdvisedBy(person, person)
    - etc.
  - 2707 constants
    - publication (342)
    - person (442)
    - course (176)
    - project (153)
    - etc.
MLN experiment

- obtained knowledge base by having four subjects provide a set of formulas in first-order logic describing the domain

- the formulas in the KB represent statements such as
  - students are not professors
  - each student has at most one advisor
  - if a student is an author of a paper, so is her advisor
  - at most one author of a given publication is a professor
  - etc.

- note that the KB is not consistent
Learning to predict the AdvisedBy\((x, y)\) relation

- MLN w/ original KB
- MLN w/ KB + ILP learned rules
- KB alone
- KB + ILP learned rules
- ILP learned rules
- naïve Bayes
- Bayes net learner