Relational Learning

www.cs.wisc.edu/~page/cs760/
Goals for the lecture

you should understand the following concepts

• rule-set learning
• relational learning
• the FOIL algorithm
Rule sets as a hypothesis space

- we can use propositional rule sets as a hypothesis space for a learning algorithm
- each rule is a conjunction of tests + a class that is implied (predicted) when the conjunction is satisfied

\[
\text{Class}=\text{yes} \leftarrow \text{Outlook}=\text{sunny} \land \text{Humidity} \leq 75\%
\]

\[
\text{Class}=\text{yes} \leftarrow \text{Outlook}=\text{overcast}
\]

\[
\text{Class}=\text{yes} \leftarrow \text{Outlook}=\text{rain} \land \text{Win} \leq 20
\]
Decision trees and rules

Any decision tree can be converted into an equivalent set of rules

Class=yes ← Outlook=sunny ∧ Humidity≤75%

Class=yes ← Outlook=overcast

Class=yes ← Outlook=rain ∧ Wind≤20
Decision trees and rules

a small set of rules can represent a large decision tree because of the replication problem

\[
Y = 1 \leftarrow X_1 \land X_2 \\
Y = 1 \leftarrow X_3 \land X_4 \\
Y = 1 \leftarrow X_5 \land X_6
\]
Rule learning

- rule sets can be learned by extracting them from decision trees (C4.5 has a module for this)
- there are also algorithms for learning rules directly, such as SLIPPER [Cohen & Singer, AAAI 1999]

- the rules we’ve considered so far are expressed in propositional logic – they’re not well suited to representing multiple entities and relationships among them
- let’s consider relational learning methods, which represent their hypotheses using a subset of first-order logic
Relational learning example

• suppose we want to learn the general concept of can-reach in a graph, given a set of training instances describing a particular graph

• 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.

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

\[
\text{can-reach}(X_1, X_2) \leftarrow \text{linked-to}(X_1, X_3) \land \text{can-reach}(X_3, X_2)
\]
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

[Finn et al., Machine Learning 1998]

a multi-relational representation for molecules
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 +/- 0.75;
- molecule A contains a hydrogen acceptor D;
- the distance between B and D in Conf is 8.5 +/- 0.75;
- the distance between C and D in Conf is 2.1 +/- 0.75;
- molecule A contains a hydrogen acceptor E;
- the distance between B and E in Conf is 4.9 +/- 0.75;
- the distance between C and E in Conf is 3.1 +/- 0.75;
- the distance between D and E in Conf is 3.8 +/- 0.75.
Relational representation

ACE_inhibitor(A) ← has_zinc_binding_site(A, B) \∧
    has_hydrogen_acceptor(A, C) \∧
    distance(B, C, 7.9, 0.75) \∧
    has_hydrogen_acceptor(A, D) \∧
    distance(B, D, 8.5, 0.75) \∧
    distance(C, D, 8.5, 0.75) \∧
    has_hydrogen_acceptor(A, E) \∧
    distance(B, E, 4.9, 0.75) \∧
    distance(C, E, 3.1, 0.75) \∧
    distance(D, E, 3.8, 0.75)

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

- has_zinc_binding_site
- has_hydrogen_acceptor
- zinc_binding_site_and_hydrogen_acceptor_distance
- 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
[Craven et al., ECML 1998]

• consider the task of classifying web pages according to their roles
• here is a learned rule for recognizing home pages for CS courses

\[
\text{course}(A) \leftarrow \\
\text{has-word}(A, \text{instructor}), \\
\neg \text{has-word}(A, \text{good}), \\
\text{link-from}(A, B), \\
\text{has-word}(B, \text{assign}), \\
\neg \text{link-from}(B, C)
\]

• test-set accuracy: 31 / 34
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>Gender</td>
</tr>
<tr>
<td>P1</td>
<td>M</td>
</tr>
<tr>
<td>P1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>labs</th>
<th>genetics</th>
</tr>
</thead>
<tbody>
<tr>
<td>PatientID</td>
<td>Date</td>
</tr>
<tr>
<td>P1</td>
<td>1/1/01</td>
</tr>
<tr>
<td>P1</td>
<td>1/9/01</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>
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

\[ \oplus: \langle 0,1 \rangle \langle 0,2 \rangle \langle 0,3 \rangle \langle 0,4 \rangle \langle 0,5 \rangle \langle 0,6 \rangle \langle 0,8 \rangle \langle 1,2 \rangle \langle 3,2 \rangle \langle 3,4 \rangle \langle 3,5 \rangle \langle 3,6 \rangle \langle 3,8 \rangle \langle 4,5 \rangle \langle 4,6 \rangle \langle 4,8 \rangle \langle 6,8 \rangle \langle 7,6 \rangle \langle 7,8 \rangle \]

\[ \ominus: \langle 0,0 \rangle \langle 0,7 \rangle \langle 1,0 \rangle \langle 1,1 \rangle \langle 1,3 \rangle \langle 1,4 \rangle \langle 1,5 \rangle \langle 1,6 \rangle \langle 1,7 \rangle \langle 1,8 \rangle \langle 2,0 \rangle \langle 2,1 \rangle \langle 2,2 \rangle \langle 2,3 \rangle \langle 2,4 \rangle \langle 2,5 \rangle \langle 2,6 \rangle \langle 2,7 \rangle \langle 2,8 \rangle \langle 3,0 \rangle \langle 3,1 \rangle \langle 3,3 \rangle \langle 3,7 \rangle \langle 4,0 \rangle \langle 4,1 \rangle \langle 4,2 \rangle \langle 4,3 \rangle \langle 4,4 \rangle \langle 4,7 \rangle \langle 5,0 \rangle \langle 5,1 \rangle \langle 5,2 \rangle \langle 5,3 \rangle \langle 5,4 \rangle \langle 5,5 \rangle \langle 5,6 \rangle \langle 5,7 \rangle \langle 5,8 \rangle \langle 6,0 \rangle \langle 6,1 \rangle \langle 6,2 \rangle \langle 6,3 \rangle \langle 6,4 \rangle \langle 6,5 \rangle \langle 6,6 \rangle \langle 6,7 \rangle \langle 7,0 \rangle \langle 7,1 \rangle \langle 7,2 \rangle \langle 7,3 \rangle \langle 7,4 \rangle \langle 7,5 \rangle \langle 7,7 \rangle \langle 8,0 \rangle \langle 8,1 \rangle \langle 8,2 \rangle \langle 8,3 \rangle \langle 8,4 \rangle \langle 8,5 \rangle \langle 8,6 \rangle \langle 8,7 \rangle \langle 8,8 \rangle \]

- extensionally defined background relations

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

FOIL uses a covering approach to learn a set of rules

\textsc{LearnRuleSet}(\text{set of tuples } T \text{ of target relation, background relations } B) \{
S = \{\}\n\text{repeat}
\quad R \leftarrow \textsc{LearnRule}(T, B)
\quad S \leftarrow S \cup R
\quad T \leftarrow T – \text{ positive tuples covered by } R
\text{ until there are no (few) positive tuples left in } T
\text{ return } S
\}
The FOIL algorithm for relational learning

\textsc{LearnRule}(set of tuples $T$ of target relation, background relations $B$)
{
    $R = \{ \}$
    repeat
        $L \leftarrow$ best literal, based on $T$ and $B$, to add to right-hand side of $R$
        $R \leftarrow R \cup L$
        $T \leftarrow$ new set of tuples that satisfy $L$
    until there are no (few) negative tuples left in $T$
    return $R$
}
Literals in FOIL

- Given the current rule \( R(X_1, X_2, \ldots X_k) \leftarrow L_1 \land L_2 \land \ldots \land L_n \)

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

\[ X_j \neq X_k \]

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

\[ \neg Q(V_1, V_2, \ldots V_a) \]  

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 \( a \) is a numeric constant

\[ X_j \leq a \]

\[ X_j > X_k \]

\[ X_j \leq X_k \]
• 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{itemize}
    \item and instances of the background relation \text{linked-to}
\end{itemize}

\[ \text{linked-to} = \{ \langle 0, 1 \rangle, \langle 0, 3 \rangle, \langle 1, 2 \rangle, \langle 3, 2 \rangle, \langle 3, 4 \rangle, \langle 4, 5 \rangle, \langle 4, 6 \rangle, \langle 6, 8 \rangle, \langle 7, 6 \rangle, \langle 7, 8 \rangle \} \]
Foil example

• 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

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

- 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\((X_1, X_3)\)

\[
\text{FOIL}\_\text{Gain}(L,R) = 9 \left( \log_2 \frac{18}{18 + 54} - \log_2 \frac{9}{9 + 62} \right) = 8.8
\]
Additional refinements of FOIL

• early stopping to prevent overfitting
• using $m$-estimates of rule precision to guide search [Džeroski & Bratko, *ILP* 1992]
• type constraints on variables
• *relational pathfinding* to guide search for binary target relations [Craven, Slattery & Nigam, *ECML* 1998]

\[
\text{between}(X, Y, Z) \leftarrow \text{less-than}(X, Y) \land \text{less-than}(Y, Z) \\
\text{between}(X, Y, Z) \leftarrow \text{less-than}(Z, Y) \land \text{less-than}(Y, X)
\]
Comments on relational learning

• enables learning with more expressive hypothesis spaces
• but this comes at the cost of having large hypothesis spaces
  • harder to search
  • easier to overfit
• can take advantage of background knowledge represented as extensional relations or logical clauses (rules)
• one branch of research in this area – inductive logic programming – focuses on learning hypotheses in a logic programming framework
• search can be top-down (like FOIL) or bottom-up
• many relational learning methods not well suited to handling noisy data, representing uncertainty
  • but in the next lecture we’ll discuss statistical relational learning methods which are designed to address these limitations