Symbolic AI

Andre Freitas

Photo by Vasilyev Alexandr
Acknowledgements

• These slides were based on the slides of:
  – Anoop & Hector, Inductive Logic Programming (for Dummies).
  – Gabor Melli, Scribe Notes on FOIL and Inverted Deduction.
  – CS 5751 Machine Learning, Chapter 10 Learning Sets of Rules.
This Lecture

• Introduction to Inductive Logic Programming

• FOIL
Linear Classifier

Flach 2001
Decision Trees
Decision Trees

- If $X \leq x_1$ then ◇
- Else if $Y \leq y_1$ then ■
- Else if $Y \leq y_2$ then ▲
- Else ●
ILP: Objective

Given a dataset:

• Positive examples ($E+$) and optionally negative examples ($E-$).
• Additional knowledge about the problem/application domain (Background Knowledge $B$).
• Set of constraints to make the learning process more efficient ($C$).

Goal of an ILP system is to find a set of hypothesis that:

• Explains (covers) the positive examples – Completeness.
• Are consistent with the negative examples – Consistency.
Generalisation & Specialisation

• **Generalising** a concept involves enlarging its extension in order to cover a given instance or subsume another concept.

• **Specialising** a concept involves restricting its extension in order to avoid covering a given instance or subsuming another concept.
First-order Representations

- **Propositional** representations:
  - data case is *fixed-size vector of values*
  - features are those given in the dataset

- **First-order** representations:
  - data case is *flexible-size, structured object*
  - sequence, set, graph
  - hierarchical: e.g. set of sequences
  - features need to be *selected* from potentially infinite set

Flach 2001
Deductive Vs Inductive Reasoning

\[ T \cup B \rightarrow E \text{ (deduce)} \]

\[ E \cup B \rightarrow T \text{ (induce)} \]

- \( T \)
  - parent(X, Y) :- mother(X, Y).
  - parent(X, Y) :- father(X, Y).

- \( B \)
  - mother(mary, vinni).
  - mother(mary, andre).
  - father(carrey, vinni).
  - father(carry, andre).

- \( E \)
  - parent(mary, vinni).
  - parent(mary, andre).
  - parent(carrey, vinni).
  - parent(carrey, andre).

- \( T \) (induce)
  - parent(X, Y) :- mother(X, Y).
  - parent(X, Y) :- father(X, Y).
Relational Pattern

IF Customer(C1,Age1,Income1,TotSpent1,BigSpender1)
    AND MarriedTo(C1,C2)
    AND Customer(C2,Age2,Income2,TotSpent2,BigSpender2)
    AND Income2 ≥ 10000
THEN BigSpender1 = Yes

big_spender(C1,Age1,Income1,TotSpent1) ←
    married_to(C1,C2) ^
    customer(C2,Age2,Income2,TotSpent2,BigSpender2) ^
    Income2 ≥ 10000
Discover the rule that describes whether a person has a granddaughter.
Propositional Learner with simple data transformation

• One of the first challenges that a propositional learner would encounter with this dataset is that the dataset is not structured as a set of fixed length-vectors of attribute-value pairs. This situation is typically resolved by JOINing the relations.

<table>
<thead>
<tr>
<th>Predictors</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>Father</td>
<td>Has Gdough</td>
</tr>
<tr>
<td>Bob</td>
<td>FALSE</td>
</tr>
<tr>
<td>Sharon</td>
<td>TRUE</td>
</tr>
<tr>
<td>Victor</td>
<td>TRUE</td>
</tr>
<tr>
<td>Bob</td>
<td>FALSE</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Propositional Learner with simple data transformation

• A propositional learner would not locate a predictive model for this dataset.
• It would not be able to state that Sharon is Victor’s granddaughter.
• At best it may discover that a child’s gender has some influence on the likelihood that that child is a parent, or even a parent to a female child.
Propositional Leaner with complex data transformation

• The algorithm cannot make the connection in one observation (*Bob as a father*) and another (*Bob as child*).

• A common way to enable a propositional learner to produce a predictive model on this data is to transform the data so that the required relations appear as attributes in the data.
• This transformation is sometimes referred to as ‘flattening’ the data.

<table>
<thead>
<tr>
<th>Father</th>
<th>Child</th>
<th>Child is Fem.</th>
<th>Child's Child</th>
<th>C's C is Fem.</th>
<th>Has GrandDaughter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bob</td>
<td>Sharon</td>
<td>TRUE</td>
<td>NULL</td>
<td>NULL</td>
<td>FALSE</td>
</tr>
<tr>
<td>Victor</td>
<td>Bob</td>
<td>FALSE</td>
<td>Sharon</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

• Now the search for a rule is trivial. A decision tree would locate the pattern:

```
IF Child’s Child is Female = TRUE
    THEN HasGrandDaughter = TRUE.
ELSE HasGrandDaughter = FALSE
```
Propositional Sequential Covering

• A covering algorithm, in the context of propositional learning systems, is an algorithm that develops a cover for the set of positive examples.
  – that is, a set of hypotheses that account for all the positive examples but none of the negative examples.

• Sequential covering: it learns one rule at a time and repeat this process to gradually cover the full set of positive examples.
Iterate to Learn Multiple Rules

• Select seed from positive examples to build bottom clause.

• Get some rule “If A \land B then P”. Now throw away all positive examples that were covered by this rule.

• Repeat until there are no more positive examples.
Propositional Sequential Covering

1. Start with an empty **Cover**
2. Use **Learn-One-Rule** to find the best hypothesis.
3. If the Just-Learnt-Rule satisfies the threshold then
   - Put Just-Learnt-Rule to the **Cover**.
   - Remove examples covered by Just-Learnt-Rule.
   - Go to step 2.
4. Sort the **Cover** according to its performance over examples.
5. Return: **Cover**.
### Example

<table>
<thead>
<tr>
<th>Id</th>
<th>Size</th>
<th>Colour</th>
<th>Shape</th>
<th>Weight</th>
<th>Expensive</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Big</td>
<td>Red</td>
<td>Square</td>
<td>Heavy</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>Small</td>
<td>Blue</td>
<td>Triangle</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>Small</td>
<td>Blue</td>
<td>Square</td>
<td>Light</td>
<td>No</td>
</tr>
<tr>
<td>4</td>
<td>Big</td>
<td>Green</td>
<td>Triangle</td>
<td>Heavy</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>Big</td>
<td>Blue</td>
<td>Square</td>
<td>Light</td>
<td>No</td>
</tr>
<tr>
<td>6</td>
<td>Big</td>
<td>Green</td>
<td>Square</td>
<td>Heavy</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>Small</td>
<td>Red</td>
<td>Triangle</td>
<td>Light</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Expensive = Yes if:

- Colour = Red.  
- Or (Colour = Green & Shape = Square).  
- Or (Colour = Blue & Shape = Triangle).  

(covers example 1,7) 
(covers example 6) 
(covers example 2)
Complex

• A complex is a conjunction of attribute-value specifications. It forms the condition part in a rule, like "if condition then predict class".

<table>
<thead>
<tr>
<th>Size=Big</th>
<th>Size=Small</th>
<th>Colour=Red</th>
</tr>
</thead>
<tbody>
<tr>
<td>Colour=Green</td>
<td>Colour=Blue</td>
<td>Shape=Square</td>
</tr>
<tr>
<td>Shape=Triangle</td>
<td>Weight=Light</td>
<td>Weight=Heavy</td>
</tr>
</tbody>
</table>

• Specialising a complex is making a conjunction of the complex with one more attribute-value pair. For example:

Colour=Green & Shape=Square  (specialising Colour=Green or Shape=Square)
Colour=Blue & Weight=Heavy   (specialising Colour=Blue or Weight=Heavy)
Learn-One-Rule using Beam Search

1. Initialize a set of most general complexes.
2. Evaluate performances of those complexes over the example set.
   – Count how many positive and negative examples it covers.
   – Evaluate their performances.
3. Sort complexes according to their performances.
4. If the best complex satisfies some threshold, form the hypothesis and return.
5. Otherwise, pick $k$ best performing complexes for the next generation.
6. Specializing all $k$ complexes in the set to find new set of less general complexes.
7. Go to step 2.

The number $k$ is the beam factor of the search, meaning the maximum number of complexes to be specialized.
Example

R=Yes

A=A2
B=B2

A=A2
C=C2

A=A1
D=D1

B=B1
General to Specific Beam Search
Example

• In the first step, 2 best complexes are found, namely A=A1 and B=B2.

• None of them satisfy the threshold, then the next level complexes are expanded and found 2 best complexes, eg. A=A1 & D=D1 and B=B2 & A=A2.

• The procedure keeps going until we find a complex that satisfies the threshold.
Entropy Evaluation Function

• The evaluation is based on the entropy of the set covered by that complex. Here is an example of a hypothesis covering 8 positive and 2 negative examples.

\[
p1 = P(\text{positive}) = \frac{8}{2+8} = 0.8;
\]
\[
p2 = P(\text{negative}) = \frac{2}{2+8} = 0.2;
\]

\[
\text{Entropy} = -p1 \log(p1) - p2 \log(p2) = 0.72.
\]

• In this function, the smaller the entropy is, the better the complex.

• In other words, the accuracy function can be defined as \((1-\text{Entropy})\).
The FOIL Algorithm

- The FOIL algorithm is a supervised learning algorithm that produces rules in first-order logic.
- The algorithm is a generalization of the SEQUENTIAL-COVERING and LEARN-ONE-RULE algorithms.
- The main modification is that search can also specialize on predicates with variables.
- The resulting rules differ from Horn clauses in two ways:
  - Negated symbols are allowed within the body.
  - FOIL’s rules will not include function symbols.
Back to the Example
fathers $x$ are somewhat likely to be grandfathers

<table>
<thead>
<tr>
<th>IF any</th>
<th>THEN $GD(x,y)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>POS 5% 50</td>
<td></td>
</tr>
<tr>
<td>NEG 95% 950</td>
<td></td>
</tr>
</tbody>
</table>

no female grandfathers in the data

<table>
<thead>
<tr>
<th>IF Female ($x$)</th>
<th>THEN $GD(x,y)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>POS 0% 0</td>
<td></td>
</tr>
<tr>
<td>NEG 100% 425</td>
<td></td>
</tr>
</tbody>
</table>

fathers $x$ of a father $z$ have 50% chance of having a granddaughter

<table>
<thead>
<tr>
<th>IF Father ($z,x$) ^ Father($y,z$)</th>
<th>THEN $GD(x,y)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>POS 50% 50</td>
<td></td>
</tr>
<tr>
<td>NEG 50% 50</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>IF Father ($z,x$) ^ Father($y,z$) ^ Female($y$)</th>
<th>THEN $GD(x,y)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>POS 100% 50</td>
<td></td>
</tr>
<tr>
<td>NEG 0% 0</td>
<td></td>
</tr>
</tbody>
</table>
FOIL

\[ \text{FOIL}(Target\_predicate,\text{Predicates},\text{Examples}) \]
\[ Pos \leftarrow \text{positive Examples} \]
\[ Neg \leftarrow \text{negative Examples} \]
while \( Pos \) do (Learn a New Rule)
\[ \text{NewRule} \leftarrow \text{most general rule possible} \]
\[ \text{NegExamplesCovered} \leftarrow Neg \]
while \( \text{NegExamplesCovered} \) do
\[ \text{Add a new literal to specialize NewRule} \]
1. \( \text{Candidate\_literals} \leftarrow \text{generate candidates} \)
2. \( \text{Best\_literal} \leftarrow \underset{L\in \text{candidate\_literal}}{\text{argmax}} \text{FOIL\_GAIN}(L,\text{NewRule}) \)
3. Add \( \text{Best\_literal} \) to \( \text{NewRule} \) preconditions
4. \( \text{NegExamplesCovered} \leftarrow \text{subset of } \text{NegExamplesCovered} \text{ that satisfies } \text{NewRule} \text{ preconditions} \)
\[ \text{Learned\_rules} \leftarrow \text{Learned\_rules} + \text{NewRule} \]
\[ Pos \leftarrow Pos - \{\text{members of } Pos \text{ covered by } \text{NewRule}\} \]
Return \( \text{Learned\_rules} \)
The FOIL Algorithm

• The *outer loop adds new rules* to the output until no more positive examples are covered.

• The *inner loop searches for the next best rule* by incremental specialization.

• The outer loop corresponds to the SEQUENTIAL-CONVERING algorithm, the inner to FIND-A-RULE
Specialising Rules in FOIL

Learning rule: $P(x_1, x_2, \ldots, x_k) \leftarrow L_1 \ldots L_n$

Candidate specializations add new literal of form:

- $Q(v_1, \ldots, v_r)$, where at least one of the $v_i$ in the created literal must already exist as a variable in the rule
- $\text{Equal}(x_j, x_k)$, where $x_j$ and $x_k$ are variables already present in the rule
- The negation of either of the above forms of literals
Information Gain in FOIL

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

Where

- \( L \) is the candidate literal to add to rule \( R \)
- \( p_0 \) = number of positive bindings of \( R \)
- \( n_0 \) = number of negative bindings of \( R \)
- \( p_1 \) = number of positive bindings of \( R + L \)
- \( n_1 \) = number of negative bindings of \( R + L \)
- \( t \) is the number of positive bindings of \( R \) also covered by \( R + L \)

Note
- \( -\log_2 \frac{p_0}{p_0 + n_0} \) is optimal number of bits to indicate the class of a positive binding covered by \( R \)
Applications
First Order Rule for Classifying Web Pages

From (Slattery, 1997)

course(A) ←
   has-word(A,instructor),
   NOT has-word(A,good),
   link-from(A,B)
   has-word(B,assignment),
   NOT link-from(B,C)

Train: 31/31, Test 31/34
Early diagnosis of rheumatic diseases

• Sample CN2 rule for an 8-class problem:

IF Sex = male AND Age > 46 AND
  Number_of_painful_joints > 3 AND
  Skin_manifestations = psoriasis

Flach 2001
Application

A molecular compound is carcinogenic if:

1. it tests positive in the Salmonella assay; or
2. it tests positive for sex-linked recessive lethal mutation in Drosophila; or
3. it tests negative for chromosome aberration; or
4. it has a carbon in a six-membered aromatic ring with a partial charge of -0.13; or
5. it has a primary amine group and no secondary or tertiary amines; or
6. it has an aromatic (or resonant) hydrogen with partial charge $\geq 0.168$; or
7. it has an hydroxy oxygen with a partial charge $\geq -0.616$ and an aromatic (or resonant) hydrogen; or
8. it has a bromine; or
9. it has a tetrahedral carbon with a partial charge $\leq -0.144$ and tests positive on Progol’s mutagenicity rules.
Final Considerations
Why ILP is not just Decision Trees.

• Language is First-Order Logic
  – Natural representation for multi-relational settings
  – Thus, a natural representation for full databases

• Not restricted to the classification task.
• So then, what is ILP?
Efficiency Issues

• Representational Aspects
• Search
• Evaluation
• Sharing computations
• Memory-wise scalability
Representational Aspects

• Example:
  – Student(string sname, string major, string minor)
  – Course(string cname, string prof, string cred)
  – Enrolled(string sname, string cname)

• In a natural join of these tables there is a one-to-one correspondence between join result and the Enrolled table.

• Data mining tasks on the Enrolled table are really propositional.
Representational Aspects

• Three settings for data mining:
  – Find patterns within individuals represented as tuples (single table, propositional)
    • eg. Which minor is chosen with what major
  – Find patterns within individuals represented as sets of tuples (each individual ‘induces’ a sub-database)
    • Multiple tables, restricted to some individual
    • eg. Student X taking course A, usually takes course B
  – Find patterns within the whole database
    • Multiple tables
Evaluation

• Evaluating a clause: get some measure of coverage
  – Match each example to the clause:
    • Run multiple logical queries.

  – Query optimization methods from DB community
    • Rel. Algebra operator reordering
    • BUT: queries for DB are set oriented (bottom-up), queries in PROLOG find a single solution (top-down).
Sharing Computations

• Materialization of features
• Propositionalization
• Pre-compute some statistics
  – Joint distribution over attributes of a table
  – Query selectivity
• Store proofs, reuse when evaluating new clauses
Summary

• Rules: easy to understand
  – Sequential covering algorithm
  – generate one rule at a time
  – general to specific - add antecedents
  – specific to general - delete antecedents

• First order logic and covering
  – how to connect variables
  – FOIL
Recommended Reading

QuickFOIL: Scalable Inductive Logic Programming

Qiang Zeng  
University of Wisconsin–Madison  
qzeng@cs.wisc.edu

Jignesh M. Patel  
University of Wisconsin–Madison  
jignesh@cs.wisc.edu

David Page  
University of Wisconsin–Madison  
page@biostat.wisc.edu