Introduction to
Inductive Logic Programming
Lectures 1 and 2

Stephen Muggleton
Department of Computing
Imperial College, London

14th March, 2011
Overview

Lecture 1   Generalisation
Lecture 2   Refinement and Inverting Entailment

Suggested reading


Machine learning

Machine Learning is the study of computer programs that improve automatically through experience.


<table>
<thead>
<tr>
<th>Logical</th>
<th>Probabilistic</th>
<th>Mixed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decision trees</td>
<td>Neural nets</td>
<td>Bayes’ nets</td>
</tr>
<tr>
<td>Grammars</td>
<td>HMMs</td>
<td>SCFGs</td>
</tr>
<tr>
<td>Logic Programs</td>
<td>POMDPs</td>
<td>SLPs</td>
</tr>
</tbody>
</table>
**Inducing a model**

“4-helical up-and-down bundle”

<table>
<thead>
<tr>
<th>Positive(12)</th>
<th>Negative(12)</th>
</tr>
</thead>
</table>

The protein P has fold class “Four-helical up-and-down bundle” if it contains a long helix H1 at a secondary structure position between 1 and 3, and H1 is followed by a second helix H2.

[JMB, 2001; MLJ, 2001]
Inductive Logic Programming

**Background knowledge.** Protein sequence, partial grammar, domain constraints.

**Examples.** Molecules, annotated sentences.

**Hypothesis.** Explanation of molecular 3-D shape, new clauses in a grammar.
What is generalisation?

Statement A Daffy Duck can fly
Statement B All ducks can fly
Statement C Marek lives in London
Statement D Marek lives in England
Terms, atoms and literals

Function symbols  eg.  \( f, g \)
Predicate symbols  eg.  \( p, q \)
Constants  eg.  \( c, d \)
Variables  eg.  \( x, y, z \)
Terms  eg.  \( c, 3, x, f(c, g(x)) \)
Atoms  eg.  \( \forall x, y.p(x, f(3, y)) \)
  or  \( p(x, f(3, y)) \)
Literals  eg.  \( \neg p(x, f(3, y)), q(z, d) \)
## Clauses and Clausal Theories

<table>
<thead>
<tr>
<th>Concept</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clause - disjunction of literals</td>
<td>eg. $l_1 \lor \ldots \lor l_m$</td>
</tr>
<tr>
<td></td>
<td>or ${l_1, \ldots, l_m}$</td>
</tr>
<tr>
<td>Definite Clause - one positive literal</td>
<td>eg. $a_0 \lor \neg a_1 \ldots \lor \neg a_m$</td>
</tr>
<tr>
<td></td>
<td>or $a_0, \leftarrow a_1, \ldots, a_m$</td>
</tr>
<tr>
<td>Theory - conjunction of clauses</td>
<td>eg. $C_1 \land \ldots \land C_n$</td>
</tr>
<tr>
<td></td>
<td>or ${C_1, \ldots, C_n}$</td>
</tr>
<tr>
<td>Logic Program - conjunction of</td>
<td>eg. $C_1 \land \ldots \land C_n$</td>
</tr>
<tr>
<td>Horn clauses</td>
<td>or ${C_1, \ldots, C_n}$</td>
</tr>
</tbody>
</table>
Simple generalisation
Atom and Clause Subsumption

Given a substitution \( \theta = \{v_1/t_1, \ldots, v_n/t_n\} \) and formula \( F \). \( F\theta \) is formed by replacing every variable \( v_i \) in \( F \) by \( t_i \).

Atom \( A \) subsumes atom \( B \), \( A \succeq B \), iff there exists a substitution \( \theta \) such that \( A\theta = B \).

Clause \( C \) subsumes clause \( D \), \( C \succeq D \), iff there exists a substitution \( \theta \) such that \( C\theta \subseteq D \).
Generalisation example revisited

Daffy Duck can fly \( \text{can}\_\text{fly}(\text{daffy}) \)
All ducks can fly \( \text{can}\_\text{fly}(x) \)

\( \text{can}\_\text{fly}(x) \geq \text{can}\_\text{fly}(\text{daffy}) \)

\( \theta = \{x/\text{daffy}\} \)
Least general generalisation (lgg) [Plotkin/Reynolds]

Atom $A'$ is a common generalisation of atoms $A$ and $B$ iff $A' \succeq A$ and $A' \succeq B$.

$A'$ is a least general generalisation of atoms $A$ and $B$ iff all common generalisations of $A$ and $B$ subsume $A'$.

Atoms $A$ and $B$ are compatible iff they have the same predicate symbol and sign.
### lgg example

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>lgg(A, B)</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>can_fly(daffy)</code></td>
<td><code>can_fly(donald)</code></td>
<td><code>can_fly(x)</code></td>
</tr>
<tr>
<td><code>conn(n1,n1)</code></td>
<td><code>conn(n2,n2)</code></td>
<td><code>conn(x,x)</code></td>
</tr>
</tbody>
</table>
Generalisation - harder example

C Marek lives in London    | lives(marek,london)
D Marek lives in England  | lives(marek,england)

Background knowledge

lives(x,england) ← lives(x,london)
Interpretations

Ground formula: Formula containing no variables
Herbrand Universe, $U$: Set of all ground terms constructed from given predicate, function symbols and constants
Herbrand Base: Set of all ground atoms over Herbrand Universe
Interpretation: Subset of Herbrand Base Atoms assigned True
Models and Entailment

Interpretation $M$ is a model of formula (atom/literal/clause/theory) $F$ iff $F$ evaluates to True given $M$. To evaluate $F$ for each variable $x$ replace $\forall x.P(x)$ throughout by $\bigwedge_{t \in U} P(t)$ and then each atom in $M$ by True and apply logical connective truth tables throughout.

$$F \models G$$ iff every model of $F$ is a model of $G$
Generalisation as entailment

Entailment

$C$ more general than $D$ iff $C \models D$

Relative Entailment

$C$ more general than $D$ wrt $B$ iff $B, C \models D$
ILP general logical setting

B  Background Knowledge - Logic Program
E  Examples - Set of ground unit clauses
H  Hypothesis - Logic Program

Given $B, E$ find $H$ such that

$$B, H \models E$$
Search and refinement

Given $B, E$ find $H$ such that

$$B, H \models E$$

$Q$ : Algorithmically how do we find $H$ given $B, E$?

$A$ : Search space of clauses from simple to complex (general to specific) or complex to simple (specific to general). This process is called Clause Refinement.
Refinement operator [Shapiro] $\rho$

Predicate symbols $P$, Function symbols $F$

Clause $D \in \rho(C)$ iff one of the following.

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>$C \cup {p(v_1, \ldots, v_m)}$</th>
<th>$p_m \in P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>${l_1, \ldots, l_n}$</td>
<td>${l_1, \ldots, l'_i, \ldots, l_n}$</td>
<td>$l'_i = l_i{u/v}$</td>
</tr>
<tr>
<td>2.</td>
<td>${l_1, \ldots, l_i, \ldots, l_n}$</td>
<td>${l_1, \ldots, l'_i, \ldots, l_n}$</td>
<td>$t = f(v_1, \ldots, v_m)$ $f_m \in F$</td>
</tr>
<tr>
<td>3.</td>
<td>${l_1, \ldots, l_i, \ldots, l_n}$</td>
<td>${l_1, \ldots, l'_i, \ldots, l_n}$</td>
<td>$l'_i = l_i{u/t}$</td>
</tr>
</tbody>
</table>


Refinement graph $\rho$

$$\Box$$

$\text{lives}(U,V) \leftarrow$

$$\text{lives}(U,V) \leftarrow \text{lives}(W,X)$$

$\text{lives}(U,U) \leftarrow \text{lives}(\text{marek},V) \leftarrow$$
Theorem. There is no refinement operator $\rho$ which has all the following properties.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finite</td>
<td>$\forall C. \</td>
</tr>
<tr>
<td>Proper</td>
<td>$\forall C, D. D \in \rho(C)$ only if $C \succ D$</td>
</tr>
<tr>
<td>Complete</td>
<td>$\forall C \exists n. C \in \rho^n(\square)$</td>
</tr>
</tbody>
</table>
Unconstrained search space

\[ H \geq H' \text{ iff } H \models H' \]
Constrained search space

\[ C \geq D \text{ iff } \exists \theta. C\theta \subseteq D \]
Inverse resolution (first-order)

\[ C (\, (+) \, \) \]
\[ \theta \]
\[ D \]
\[ \theta' \]

\[ C' (\, (-) \, \) \]
Inverting entailment

\[ B \land H \models E \]
\[ B \land \overline{E} \models \overline{H} \]
\[ B \land \overline{E} \models \bot \models \overline{H} \]
\[ H \models \bot \]
## Inverting Entailment Examples (1)

<table>
<thead>
<tr>
<th>B</th>
<th>E</th>
<th>⊥</th>
</tr>
</thead>
<tbody>
<tr>
<td>anim(X) ← pet(X). pet(X) ← dog(X).</td>
<td>nice(X) ← dog(X).</td>
<td>nice(X) ← dog(X), pet(X), anim(X).</td>
</tr>
<tr>
<td>hasbeak(X) ← bird(X). bird(X) ← vulture(X).</td>
<td>hasbeak(tweety).</td>
<td>hasbeak(tweety); bird(tweety); vulture(tweety).</td>
</tr>
</tbody>
</table>
## Inverting Entailment Examples (2)

<table>
<thead>
<tr>
<th>B</th>
<th>E</th>
<th>⊥</th>
</tr>
</thead>
</table>
| white(swan1). | ← black(swan1). | ← black(swan1),  
white(swan1). |
| sentence([],[]). | sentence([a,a,a],[[]]). | sentence([a,a,a],[[]]) ←  
sentence([],[]). |