Inductive Logic Programming

Basic Approaches

By Chongbing Liu

Mar. 15, 2004
Outlines

- Introduction
- Basic Approaches
- A Simple Demo
- ILP In The Future
Introduction

- ILP and Machine Learning
- Problem Specification
- An Simple Example
- Different ILP Learners
- ILP Search Space
Learning concepts inductively and directly from given examples and background knowledge, with first-order logic as the only representation.
Introduction:

ILP and Machine Learning (2)

- is the intersection of inductive Machine Learning and Logic Programming
- inductively and directly learns concepts
- examples, background knowledge and target concepts are all represented in form of first-order logic
- more powerful than propositional learning (because ......)
Introduction: ILP Problem Specification

Given
- training examples $E$, $E = E^+ \cup E^-$
- background knowledge $B$, $B \not\models E$

Find
- target concepts $H$, which are complete, $B \cup H \not\models E^+$, and consistent, $B \cup H \cup E^- \not\models \bot$
- with respect to $B$ and $E$
Introduction:

An Simple Example

- Training Examples
  \[ E^+ = \{ \text{daughter}(\text{mary}, \text{ann}). \]
  \[ \quad \text{daughter}(\text{eve}, \text{tom}). \} \]
  \[ E^- = \{ \text{daughter}(\text{tom}, \text{ann}). \]
  \[ \quad \text{daughter}(\text{eve}, \text{ann}). \} \]

- Background Knowledge
  \[ \text{parent}(\text{ann}, \text{mary}). \]
  \[ \text{parent}(\text{ann}, \text{tom}). \]
  \[ \text{parent}(\text{tom}, \text{eve}). \]
  \[ \text{parent}(\text{tom}, \text{bob}). \]
  \[ \text{female}(\text{ann}). \]
  \[ \text{female}(\text{mary}). \]
  \[ \text{female}(\text{eve}). \]

- Concept learned:
  \[ \text{daughter}(X,Y) \leftarrow \text{female}(X), \text{parent}(Y,X) \]
Different ILP Learners

- Single concept learners and multiple concepts learners
- Batch learners and incremental learners
- Non-Interactive learners and interactive learners

Empirical ILP Learners:

non-interactive single concept batch learners
Introduction:

ILP Search Space (1)

structure

• ILP problem is a search problem
• ILP search space consists of all syntactically legal hypotheses (clauses) constructed from the predicates provided by the background knowledge
• Very big search space
Introduction:

ILP Search Space (2)

An Example

If target concept: \( \text{daughter}(X,Y) \),

background knowledge: \( \text{femal}(\text{tom}), \ldots, \text{parent}(\text{tom}, \text{bob}) \)

then the following hypotheses are in the search space:

\[
\begin{align*}
\text{d}(X,Y) & \quad \text{d}(X,Y) \leftarrow \text{f}(X), \text{p}(X,Y) \\
\text{d}(X,Y) \leftarrow \text{f}(X) & \quad \text{d}(X,Y) \leftarrow \text{f}(Y), \text{p}(X,Y) \\
\text{d}(X,Y) \leftarrow \text{f}(y) & \quad \text{d}(X,Y) \leftarrow \text{f}(X), \text{p}(Y,X) \\
\text{d}(X,Y) \leftarrow \text{p}(X,Y) & \quad \text{d}(X,Y) \leftarrow \text{f}(Y), \text{p}(Y,X) \\
\text{d}(X,Y) \leftarrow \text{p}(Y,X) & \quad \text{d}(X,Y) \leftarrow \text{f}(X), \text{p}(X,X) \\
\text{d}(X,Y) \leftarrow \text{p}(X,X) & \quad \text{d}(X,Y) \leftarrow \text{f}(Y), \text{p}(Y,Y) \\
\text{d}(X,Y) \leftarrow \text{p}(Y,Y) & \quad \text{d}(X,Y) \leftarrow \text{f}(X), \text{p}(X,Z)
\end{align*}
\]
Introduction:

ILP Search Space (4)

Generality relations

• More-general-than
• More-specific-than
• No-more-general-than
• No-more-specific-than

(defined based on $\Theta$-subsumption)
Introduction:

ILP Search Space (5)

Operations

Specializations:
- Add literals into the clause body
- Apply substitutions to the clause

Generalizations:
- Remove literals from the clause body
- Apply inverse-substitutions to the clause
**Introduction:**

ILP Search Space (6)

Refinement graph

```
daught(X,Y) ←
X=Y
```

```
daught(X,Y) ←
female(X)
```

```
daught(X,Y) ←
parent(Y,X)
```

```
daught(X,Y) ←
parent(X,Z)
```

```
daught(X,Y) ←
female(X),
female(Y)
```

```
daught(X,Y) ←
female(X),
parent(Y,X)
```

Outlines

- Introduction
- ILP Approaches
- A Simple Demo
- ILP In The Future
Basic Approaches

- Top-down Approaches
- Bottom-up Approaches
- Hybrid Approaches
- Summary
Basic Approaches:

Top-down Approaches (1)

Overview

• Use specialization
• Keep a refinement graph
• Search from the most general clause down to less general clauses, i.e., top-down,

( potenially adding literals into the clause body)
Basic Approaches:

Top-down Approaches (2)

Generic Top-down Algorithm (for e.g. FOIL)

\[
\begin{align*}
E' & := E \\
H & := \emptyset \\
\text{repeat} & \\
\quad c & := T \leftarrow \\
\quad \text{repeat} & \\
\quad \quad c & := \text{refinement}(c) \\
\quad \text{until some criterion is satisfied} & \\
H & := H \cup \{c\} \\
B & := B \cup \{c\} \\
E' & := E' - \{ \text{positive examples covered by B} \} \\
\text{until some criterion is satisfied} & 
\end{align*}
\]
Basic Approaches:

Bottom-up Approaches (1)

Two main techniques:

- Relative Least General Generalization
- Inverse Resolution

*(all use inverse substitution in different ways)*
\textbf{Basic Approaches:}

\textbf{Bottom-up Approaches (2)}

\textbf{Relative Least General Generalization (rlgg)}

\begin{itemize}
  \item \textit{rlgg} is based on least general generalization \textit{lgg}
  \item \textit{lgg}(\textit{structure1,structure2}) is defined between any two structures, e.g., \textit{lgg}(parent(\textit{ann,mary}), parent(\textit{ann,tom})) = parent(\textit{ann,X})
  \item \textit{rlgg} is \textit{lgg} of two ground atoms \textit{A1} and \textit{A2} with respect to background knowledge \textit{K}, \textit{rlgg}(A1,A2) = lgg((A1\leftarrow K),(A2\leftarrow K))
\end{itemize}
Bottom-up Approaches (3)

.rlgg based algorithms (for e.g. Golem)

//assume background knowledge is a set of ground facts

\[ E' := E \]
\[ H := \emptyset \]

Repeat
   \[ E_p := \emptyset \]
   \[ c := \text{a clause which covers no examples} \]
   repeat
      \[ E_p := \text{randomly pick several pairs of examples from } E' - E_p \]
      compute \( rlgg \) of the pairs using \( rlgg(e_1,e_2) = lgg((e_1 \leftarrow B),(e_2 \leftarrow B)) \)
      compute \( rlggs \) of the \( rlggs \) obtained above and \( c \)
      \[ c := \text{choose the } rlgg \text{ with the greatest coverage} \]
      \[ E_p := E_p - \{ \text{those examples covered by } c \} \]
   until no more positive examples are covered by \( c \)

\[ H := H \cup \{c\} \]
\[ B := B \cup \{c\} \]
\[ E' := E' - \{ \text{positive examples covered by } H \text{ and } B \} \]

until some criterion is satisfied
Basic Approaches:

Bottom-up Approaches (4)

**Inverse Resolution**

**Resolution:**

*Given clause* $c_1$ *and* $c_2$, *derive resolvent* $c$

**Inverse resolution:**

*Given clause* $c_1$ *and resolvent* $c$, *derive another clause* $c_2$

**Two forms:**

- *propositional form*
- *first-order logic form*
Bottom-up Approaches (5)

Inverse Resolution

propositional form:

\[ \text{resolution ( find } L \text{ in } c1 \text{ and } \neg L \text{ in } c2 ) \]
\[ c1 \land c2 \vdash c \]
\[ c = (c1 - \{L\}) \cup (c2 - \{\neg L\}) \]

inverse resolution

\[ c2 = (c - (c1 - \{L\})) \cup \{\neg L\} \]
Bottom-up Approaches (6)

Inverse Resolution

propositional form: (an example)
Basic Approaches:

Bottom-up Approaches (7)

Inverse Resolution

first-order logic form:

resolution

\[
( \text{find } L_1 \text{ in } c_1 \text{ and } L_2 \text{ in } c_2 \text{ s.t. } L_1 \Theta_1 = L_2 \Theta_2 )
\]

\[
c_1 \wedge c_2 \models c
\]

\[
c = ( c_1 - \{L_1\} ) \Theta_1 \cup ( c_2 - \{L_2\} ) \Theta_2
\]

inverse resolution

\[
c - ( c_1 - \{L_1\} ) \Theta_1 = ( c_2 - \{L_2\} ) \Theta_2
\]

\[
L_2 = L_1 \Theta_1 \Theta_2^{-1}
\]

\[
c_2 = ( c - (c_1 - \{L_1\}) \Theta_1 ) \Theta_2^{-1} \cup \{ \top \} \leftarrow L_1 \Theta_1 \Theta_2^{-1}
\]
Basic Approaches:

Bottom-up Approaches (8)

Inverse Resolution

first-order logic form: (an example)

c1: father(shannon,tom)

c2: grandchild(bob,X) ∨ ¬father(X,tom)

\[\Theta_1 = \{\}\]

\[\Theta_2^{-1} = \{shannon/X\}\]

c: grandchild(bob,shannon)
Basic Approaches:

Bottom-up Approaches (9)

Inverse resolution based algorithms (for e.g. Cigol)

\[
E' := E \\
H := \emptyset \\
\text{while } E' \neq \emptyset \text{ do} \\
\quad e := \text{the next positive example} \\
\quad \text{invs := all the inverse resolutions of } e \text{ and } B \\
\quad c := \text{choose the one with the highest accuracy} \\
\quad H := H \cup \{c\} \\
\quad B := B \cup \{c\} \\
\quad E' := E' \setminus \{ \text{positive examples covered by } B \} \\
\]

generalization
Hybrid Approaches (1)

- Use most specific boundary (Progol)
- Use general and specific boundaries
- Translate into propositional learning problem
Basic Approaches:

Hybrid Approaches (2)

Use most specific boundary (Progol)

\[
E' := E \\
H := \emptyset \\
\text{while } E' \neq \emptyset \text{ do} \\
\quad e := \text{the next positive example} \\
\quad \bot := \text{the most specific clause from } e \text{ and } B \\
\quad c := \text{top-down search a best clause between } T_{\leftarrow} \text{ and } \bot \\
\quad H := H \cup \{c\} \\
\quad B := B \cup \{c\} \\
\quad E' := E' \setminus \{ \text{positive examples covered by } B \} 
\]
Basic Approaches:

Hybrid Approaches (3)

Use most specific boundary (Progol)

Consider examples one by one, using resolution, sounds bottom-up

Use resolution to construct a lower bound \( \bot \) from \( B \) and \( e \), instead of a hypothesis directly

Then search from the very top down \( \bot \) to find a clause which covers \( e \) and does not cover any of the negative examples, sounds top-down
Basic Approaches:

Hybrid Approaches (4)

Use most specific boundary (Progol)

\[ H := \emptyset \]

\[ H := H \cup \{c\} \]

true.

false.
Hybrid Approaches (5)

Maintain an upper bound and a lower bound

E’ := E
Bound_{upper} = true ← .
Bound_{lower} = false ← true .

while E’ ≠ ∅ do
  e := the next positive example
  if e is positive
    Bound_{lower} = generalize(Bound_{lower}) which covers e but none of E’
  if e is negative
    Bound_{upper} = specialize(Bound_{upper}) which does not cover e

E’ := E’ – {e}

H := { those hypotheses in between Bound_{upper} and Bound_{lower} }

Basic Approaches:
Hybrid Approaches (6)

Maintain a upper bound and a lower bound

- Positive examples
  raise the lower bound up
- Negative examples
  push the upper bound down
Basic Approaches:

Hybrid Approaches (7)

Use upper and lower boundaries

true .

false .

positive e

H

negative e

H

......

H
Basic Approaches:

Summary

**Top-down approaches:**
- perform specialization operations
- search the refinement graph top-down in brute-force, unique starting point
- successor hypotheses generated based only on the syntax of the current hypothesis representation, independent of the coming data
- generate-and-test fashion
- the impact of noisy data is minimized
- batch mode: all examples are considered simultaneously

**Bottom-up approaches:**
- perform generalization operations
- search is guided bottom-up by inverse resolution, multiple starting points
- hypotheses generated based on analysis of an individual example
- example-driven fashion
- more easily misled by noisy data
- incremental mode: examples are considered one at a time
Outlines

- Introduction
- ILP Approaches
- A Simple Demo
- ILP In The Future
Sample Problem

Target concept:

\[ \text{daughter}(X,Y) \]

Background Knowledge:

\[ \text{parent}(\text{ann}, \text{mary}). \quad \text{female}(\text{ann}). \]
\[ \text{parent}(\text{ann}, \text{tom}). \quad \text{female}(\text{mary}). \]
\[ \text{parent}(\text{tom}, \text{eve}). \quad \text{female}(\text{eve}). \]
\[ \text{parent}(\text{tom}, \text{ian}). \]

Examples:

\[ \% \text{ Positive examples } \]
\[ \text{daughter}(\text{mary}, \text{ann}). \]
\[ \text{daughter}(\text{eve}, \text{tom}). \]

\[ \% \text{ Negative examples } \]
\[ \text{daughter}(\text{tom}, \text{ann}). \]
\[ \text{daughter}(\text{eve}, \text{ann}). \]
CProgol Version 4.4
[Testing for contradictions]
[No contradictions found]

[Generalising daughter(mary,ann).]
[Most specific clause is]

daughter(A,B) :- parent(B,A), female(A), female(B).
A Simple Demo:

Progol Output (2)

[Learning daughter/2 from positive examples]
[C:-9993,8,10000,0 daughter(A,B).]
[C:-9994,8,10000,0 daughter(A,B) :- parent(B,A).]
[C:-9994,8,10000,0 daughter(A,B) :- female(A).]
[C:-19996,4,10000,0 daughter(A,B) :- female(B).]
[C:3,8,2,0 daughter(A,B) :- parent(B,A), female(A).]
[C:-19998,4,10000,0 daughter(A,B) :- parent(B,A), female(B).]
[C:-19998,4,10000,0 daughter(A,B) :- female(A), female(B).]
[7 explored search nodes]
f=3,p=8,n=2,h=0
A Simple Demo:

Progol Output (3)

[Result of search is]

daughter(A,B) :- parent(B,A), female(A).

[2 redundant clauses retracted]

daughter(A,B) :- parent(B,A), female(A).

[Total number of clauses = 1]

[Time taken 0.00s]
Outlines

- Introduction
- ILP Approaches
- A Simple Demo
- ILP In The Future
ILP In The Future

- Novel search methods
- Incorporation of explicit probabilities
- Special-purpose reasoners
- Parallel implementations (*PILP*)
- Enhanced human interaction

*(handle huge data sets in the future)*
Thank You!