Learning in Planning

Opportunities and improvements along several dimensions:

- **Search Efficiency**: Learn control knowledge to guide the planner through its search space.
- **Domain Specification**: Learn the preconditions and effects of the planning actions.
- **Quality**: Learn control knowledge for high quality plans.
Choices... The Need for Learning!

• Inductive methods
  – Data-intensive
  – Extract a general description of a concept from many examples
• Deductive methods
  – Knowledge-intensive
  – Explain and analyze an example
  – Identify the explanation as the sufficient conditions for describing the concept
  – Generalize instantiated explanation to apply to other instances

Explanation-Based Generalization – EBG, (Mitchell ’80s)

Inputs:
• Target concept definition
• Training example
• Domain theory
• Operationality criterion

Output:
Generalization of the training example that is
• sufficient to describe the target concept, and
• satisfies the operationality criterion.
The SAFE-TO-STACK Example

Input:

• **target concept**: SAFE-TO-STACK(x, y)

• **training example**:

  - ON(OBJ1, OBJ2)
  - ISA(OBJ1, BOX)  ISA(OBJ2, ENDTABLE)
  - COLOR(OBJ1, RED)  COLOR(OBJ2, BLUE)
  - VOLUME(OBJ1, 1)  DENSITY(OBJ1, 0.1) ...

• **domain theory**:

  1. NOT(FRAGILE(y)) or LIGHTER(x, y) \(\rightarrow\) SAFE-TO-STACK(x, y)
  2. VOLUME(x, v) and DENSITY(x, d) \(\rightarrow\) WEIGHT(x, v*d)
  3. WEIGHT(x1, w1) and WEIGHT(x2, w2) and LESS(w1, w2)
     \(\rightarrow\) LIGHTER(x1, x2)
  4. ISA(x, ENDTABLE) \(\rightarrow\) WEIGHT(x, 5)
  5. LESS(0.1, 5) ...

• **operationality criterion**:

  learned description should be built of terms used to describe examples directly, or other “easily” evaluated, such as LESS.
The SAFE-TO-STACK Example

- Explain why \( \text{obj1} \) is SAFE-TO-STACK on \( \text{obj2} \).
  - Construct a proof.
  - Do Goal regression: regress target concept through proof structure.
  - Proof isolates relevant features.

Generating Operational Knowledge

- Generalize proof:
  - Sometimes simply replace constants by variables.
  - Prove that all identified relevant features are necessary in general (hard! -- may need a lot of “extra” knowledge, domain axioms).

Output:

\[
\text{VOLUME(x,v1) and DENSITY(x,d1) and ISA(y,ENDTABLE) and}
\text{ and LESS(v1*d1,5) \rightarrow SAFE-TO-STACK(x,y)}
\]
EBL: A Deductive Learning Method

Why are examples needed?
- Domain theory contains all the information: simply operationalize target concept.
- Examples focus on the relevant operationalizations: characterize only examples that actually occur.

Actual purpose of EBL:
- not to “learn” more about target concept,
- but to “re-express” target concept in a more operational manner (=efficiency).
- control learning.

EBL in PRODIGY (Minton 87)

Goal: -- improve the efficiency of the planner
   -- learn control rules.

Control rules:
- Apply at individual decisions.
- Antecedent matches the state of the planner at decision making time.
- Antecedent is operational -- planner can match its state using control rule language.
- Consequent selects, rejects or prefers particular alternatives.
Target Concepts

Identify the choices of the particular planner:

- Select goal $goal$
- Select operator $op$ for achieving $goal$
- Select bindings for operator $op$ and goal $goal$
- Decide subgoal if $op$ is applicable
- Decide apply $op$

Examples of Control Rules in PRODIGY

(CONTROL-RULE SELECT-OP-UNSTACK-FOR-HOLDING
  (if (and (current-goal (holding <x>))
           (true-in-state (on <x> <y>))))
  (then select operator UNSTACK))

(CONTROL-RULE SELECT-BINDINGS-UNSTACK-HOLDING
  (if (and (current-goal (holding <x>))
           (current-ops (UNSTACK))
           (true-in-state (on <x> <y>))))
  (then select bindings ((<ob> . <x>) (<underob> . <y>))))

(CONTROL-RULE SELECT-OP-PUTDOWN-FOR-ARMEMPTY
  (if (and (current-goal (arm-empty))
           (true-in-state (holding <ob>)))
  (then select operator PUT-DOWN))

(CONTROL-RULE SELECT-BINDINGS-PUTDOWN
  (if (and (current-ops (PUT-DOWN))
           (true-in-state (holding <x>))
  (then select bindings ((<ob> . <x>))))
Discussion

- Very successful in a variety of domains.
- Learned rules are applied as other rules, i.e. if their antecedent **totally** matches planning situation.
- Utility problem: The more rules learned, the slower the deliberation.
  - Matching cost (cost of utilization)
  - Frequency of application
  - Savings every time it is applied
  - Organization of learned rules!
- If EBL system is eager to learn provably correct, the explanation effort is really large, requiring a *complete* domain theory for generalization.
  - Incremental refinement of learned rules

**HAMLET: Deduction and Induction**
(Borrajo & Veloso 94)

- Extend the basic EBL approach developed for linear problem solving
  - Define new learning opportunities
  - Consider solution quality
- Reduce the explanation effort
  - No need to acquire extra domain knowledge
- Incrementally refine control knowledge
  - Converges towards an experience-supported correct set of rules
A Typical Search Tree

What are the learning opportunities?

HAMLET’s Architecture

HAMLET

Quality Measure

Learned Control Knowledge

Bounded Explanation Module

Inductive Module

Refinement module

Training problems

Domain

PRODIGY

ST ST’

L L’

L’’

ST ST’
HAMLET’s Algorithm

Let $L$ refer to the set of learned control rules.
Let $ST$, $ST'$ refer to search trees.
Let $P$ be a problem to be solved.
Let $Q$ be a quality measure.
Initially $L$ is empty.
For all $P$ in training problems
$$ST = \text{Result of solving } P \text{ without any rules.}$$
$$ST' = \text{Result of solving } P \text{ with current set of rules } L.$$ 
If positive-examples-p($ST$, $ST'$, $Q$)
Then $L' = \text{Bounded-Explanation}(ST, ST', Q)$
     $L'' = \text{Induce}(L, L')$
If negative-examples-p($ST$, $ST'$, $Q$)
Then $L = \text{Refine}(ST, ST', L'')$

Induction Module

- Why induction?
  - Bounded explanation generates possibly over-specific rules
- Inductive operators
  - Deletion of rules that subsume others
  - Intersection of preconditions. $state$
  - Refinement of subgoaling dependencies. $prior\ goal$
  - Relaxing the subgoaling dependencies. $prior\ goal$
  - Refinement of the set of interacting goals. $other\ goals$
  - Find common superclass. $type\ of\ object$
Rule Learned by HAMLET

(\text{control-rule select-bind-fly-airplane-1})
\text{(if (current-operator fly-airplane)}
\text{(current-goal (at-airplane \text{<plane1> <airport3>}))}
\text{(true-in-state (at-airplane \text{<plane1> <airport2>}))}
\text{(true-in-state (at-object \text{<package4> <airport1>}))}
\text{(other-goals \text{((at-object \text{<package4> <airport3>})))))}
\text{(then select bindings (\text{((<plane> . <plane1>)}}}
\text{(\text{<loc-from> . <airport1>}))}
\text{(\text{<loc-to> . <airport3>})))}

Inducing Over Two Rules

- Old rule:
\text{(control-rule select-unload-airplane-1)}
\text{(if (current-goal (at-object \text{<object1> <airport2>}))}
\text{(true-in-state (at-airplane \text{<plane4> <airport3>}))}
\text{(true-in-state (at-object \text{<object1> <airport3>}))}
\text{(then select operators unload-airplane))}

- New rule:
\text{(control-rule select-unload-airplane-2)}
\text{(if (current-goal (at-object \text{<object1> <airport2>}))}
\text{(true-in-state (at-airplane \text{<plane4> <airport5>}))}
\text{(true-in-state (at-object \text{<object1> <airport3>}))}
\text{(then select operators unload-airplane))}

- Induced rule:
\text{(control-rule induced-select-unload-airplane-3)}
\text{(if (current-goal (at-object \text{<object1> <airport2>}))}
\text{(true-in-state (at-object \text{<object1> <airport3>}))}
\text{(then select operators unload-airplane))}
Refining

• Why refinement?
  – HAMLET may produce over-general rules

• Negative examples: occasions in which control rules
  have been applied and should have not

Overgeneralization

• Induced rule
  (control-rule induced-select-unload-airplane-3
   (if (current-goal (at-object <object1> <airport2>))
    (true-in-state (at-object <object1> <airport3>)))
   (then select operators unload-airplane))

• New rule
  (control-rule induced-select-unload-airplane-4
   (if (current-goal (at-object <object1> <airport2>)))
    (true-in-state (inside-airplane <object1> <plane4>)))
   (then select operators unload-airplane))

• Overgeneral rule
  (control-rule induced-select-unload-airplane-5
   (if (current-goal (at-object <object1> <airport2>)))
   (then select operators unload-airplane))
Empirical Results

<table>
<thead>
<tr>
<th>Goals</th>
<th>Problems</th>
<th>Unsolvable problems</th>
<th>Better solutions</th>
<th>Solution length</th>
<th>Nodes explored</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>without rules</td>
<td>with rules</td>
<td>without rules</td>
<td>with rules</td>
</tr>
<tr>
<td>1</td>
<td>100</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>15</td>
<td>6</td>
<td>0</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>100</td>
<td>44</td>
<td>18</td>
<td>1</td>
<td>33</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>68</td>
<td>32</td>
<td>1</td>
<td>24</td>
</tr>
<tr>
<td>20</td>
<td>75</td>
<td>62</td>
<td>36</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
<td>49</td>
<td>40</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>totals</td>
<td></td>
<td>525</td>
<td>243</td>
<td>122</td>
<td>2</td>
</tr>
<tr>
<td>%</td>
<td></td>
<td>46.3%</td>
<td>25.1%</td>
<td>0.7%</td>
<td>36.9%</td>
</tr>
</tbody>
</table>

Summary – EBL in Planning

- Long-term goal of automating planning efficiency.
- Knowledge in domain theory is not usually effective.
- Explain examples to produce operational control knowledge for decisions.
- Provably correct explanations that generalize to new situations are hard to learn.
- Difficult goal and operator choice interactions can be learned through a combined deductive and inductive approach.
- User’s quality metrics can be cast in the learned knowledge.