Rule Learning for Expert Systems

November 4, 2004
Lecture Notes on Advanced AI

Byoung-Tak Zhang

Biointelligence Laboratory
Computer Science and Engineering, Bioinformatics,
Brain Science, and Cognitive Science
Seoul National University

http://bi.snu.ac.kr/
**Expert Systems**

- **Expert Systems**
  - One of the most successful applications of AI reasoning technique using facts and rules
  - “AI Programs that achieve expert-level competence in solving problems by bringing to bear a body of knowledge [Feigenbaum, McCorduck & Nii 1988]”

- Expert systems vs. knowledge-based systems

- Rule-based expert systems
  - Often based on reasoning with propositional logic Horn clauses.
Expert Systems: Structure (1/2)

Structure of Expert Systems

♦ Knowledge Base
  ■ Consists of predicate-calculus facts and rules about subject at hand.

♦ Inference Engine
  ■ Consists of all the processes that manipulate the knowledge base to deduce information requested by the user.

♦ Explanation subsystem
  ■ Analyzes the structure of the reasoning performed by the system and explains it to the user.
Expert Systems: Structure (2/2)

- **Knowledge acquisition subsystem**
  - Checks the growing knowledge base for possible inconsistencies and incomplete information.

- **User interface**
  - Consists of some kind of natural language processing system or graphical user interfaces with menus.

- **“Knowledge engineer”**
  - Usually a computer scientist with AI training.
  - Works with an expert in the field of application in order to represent the relevant knowledge of the expert in a forms of that can be entered into the knowledge base.
Example: Loan Approval

Example: loan officer in a bank
“Decide whether or not to grant a personal loan to an individual.”

OK (The loan should be approved.)
COLLAT (The collateral for the loan is satisfactory.)
PYMT (The applicant is able to the loan payments.)
REP (The applicant has a good financial reputation.)
APP (The appraisal on the collateral is sufficiently greater than the loan amount.)
RATING (The applicant has a good credit rating.)
INC (The applicant's income exceeds his/her expenses.)
BAL (The applicant has an excellent balance sheet.)

Facts

Rules
1. COLLAT \& PYMT \& REP \(\Rightarrow\) OK
2. APP \(\Rightarrow\) COLLAT
3. RATING \(\Rightarrow\) REP
4. INC \(\Rightarrow\) PYMT
5. BAL \& REP \(\Rightarrow\) OK
**AND/OR Proof Tree**

- To prove OK
  - The inference engine searches for AND/OR proof tree using either **backward** or **forward chaining**.

- **AND/OR proof tree**
  - Root node: OK
  - Leaf node: facts
  - The root and leaves will be connected through the rules.

- Using the preceding rule in a **backward-chaining**
  - The user’s goal, to establish OK, can be done either by proving both BAL and REP or by proving each of COLLAT, PYMT, and REP.
  - Applying the other rules, as shown, results in other sets of nodes to be proved.
Backward Chaining

- By backward-chaining
Consulting system

- Attempt to answer a user’s query by asking questions about the truth of propositions that they might know about.
- Backward-chaining through the rule is used to get to askable questions.
- If a user were to “volunteer” information, bottom-up, forward chaining through the rules could be used in an attempt to connect to the proof tree already built.
- The ability to give explanations for a conclusion
  - Very important for acceptance of expert system advice.
- Proof tree
  - Used to guide the explanation-generation process.
Applications: MYCIN

- In many applications, the system has access only to uncertain rules, and the user not be able to answer questions with certainty.

- **MYCIN** [Shortliffe 1976]: Diagnose bacterial infections.

Rule 300
If : 1) The infection which requires therapy is meningitis, and 2) The patient does have evidence of serious skin or soft tissue infection, and 3) Organisms were not seen on the stain of the culture, and 4) The type of the infection is bacterial
Then: There is evidence that the organism (other than those seen on cultures or smears) which might be causing the infection is staphylococcus - coag - pos (.75); streptococcus - group - a (.5).
Applications: PROSPECTOR

♦ PROSPECTOR [Duda, Gaschnig & Hart 1979, Campbell, et al. 1982]
  ■ Reason about ore deposits.

If there is a pre-intrusive, thorough-going fault system, then there is (5, 0.7) a regional environment favorable for a porphyry copper deposit.

♦ The numbers (.75 and .5 in MYCIN, and 5, 0.7 in PROSPECTOR) are ways to represent the certainty or strength of a rule.
♦ The numbers are used by these systems in computing the certainty of conclusions.
Rule Learning

- Inductive rule learning
  - Creates new rules about a domain, not derivable from any previous rules.
  - Ex) Neural networks

- Deductive rule learning
  - Enhances the efficiency of a system’s performance by deducting additional rules from previously known domain rules and facts.
  - Ex) EBG (explanation-based generalization)
Learning Propositional Calculus Rules

- Train rules from given training set
  - Seek a set of rules that covers only positive instances
    - Positive instance: OK = 1
    - Negative instance: OK = 0
  - From training set, we desire to induce rules of the form
    \[ \alpha_1 \land \alpha_2 \land \cdots \alpha_n \Rightarrow OK \]
    where \( \alpha_i \in \{\text{APP, RATING, INC, BAL}\} \)
  - We can make some rule more specific by adding an atom to its antecedent to make it cover fewer instances.
    - Cover: If the antecedent of a rule has value True for an instance in the training set, we say that the rule covers that instance.
  - Adding a rule makes the system using these rules more general.
  - Searching for a set of rules can be computationally difficult.
    - Here, we use “greedy” method which is called separate and conquer.
Learning Propositional Calculus Rules (2)

- Separate and conquer
  - First attempt to find a single rule that covers only positive instances
    - Start with a rule that covers all instances
    - Gradually make it more specific by adding atoms to its antecedent.
  - Gradually add rules until the entire set of rules covers all and only the positive instances.
  - Trained rules can be simplified using pruning.
    - Operations and noise-tolerant modifications help minimize the risk of overfitting.
Learning Propositional Calculus Rules (3)

- Example: loan officer in a bank
  - Start with the provisional rule $T \supset \text{OK}$.  
    - Which cover all instances.
  - Add an atom it cover fewer negative instances working toward covering only positive ones.
  - Decide, which item should we added?
    - From \{APP, RATING, INC, BAL\} by

$$r_\alpha = \frac{n^+_\alpha}{n_\alpha}$$

\(n_\alpha\): the total number of instance covered by the antecedent of the rule after the addition of \(\alpha\) to the antecedent.

\(n^+_\alpha\): the total number of positive instance covered by the antecedent of the rule after the addition of \(\alpha\) to the antecedent.
Select that $\alpha$ yielding the largest value of $r_\alpha$.

- $r_{\text{APP}} = \frac{3}{6} = 0.5$
- $r_{\text{RATING}} = \frac{4}{6} = 0.667$
- $r_{\text{INC}} = \frac{3}{6} = 0.5$
- $r_{\text{BAL}} = \frac{3}{4} = 0.75$

So, we select BAL, yielding the provisional rule.

**BAL \supset OK**

<table>
<thead>
<tr>
<th>Individual</th>
<th>APP</th>
<th>RATING</th>
<th>INC</th>
<th>BAL</th>
<th>OK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Table 17.1**

Bank Data
Learning Propositional Calculus Rules (4)

- Rule $\text{BAL} \Rightarrow \text{OK}$ covers the positive instances 3, 4, and 7, but also covers the negative instance 1.
  - So, select another atom to make this rule more specific.

- We have already decided that the first component in the antecedent is BAL, so we have to consider it.

$$r_{\text{APP}} = \frac{2}{3} = 0.667$$
$$r_{\text{RATING}} = \frac{3}{3} = 1.0$$
$$r_{\text{INC}} = \frac{2}{2} = 1.0$$

We select RATING because $r_{\text{RATING}}$ is based on a larger sample.

$\text{BAL} \land \text{RATING} \Rightarrow \text{OK}$
Learning Propositional Calculus Rules

We need more rules which cover positive instance 6. To learn the next rule, eliminate from the table all of the positive instances already covered by the first rule.

<table>
<thead>
<tr>
<th>Individual</th>
<th>APP</th>
<th>RATING</th>
<th>INC</th>
<th>BAL</th>
<th>OK</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 17.2

Reduced Data
Begin the process all over again with reduced table

- Start with the rule $T \supset OK$.

  - $r_{\text{APP}} = \frac{1}{4} = 0.25$
  - $r_{\text{RATING}} = \frac{0}{3} = 0.0$
  - $r_{\text{INC}} = \frac{1}{4} = 0.25$
  - $r_{\text{BAL}} = \frac{0}{1} = 0.0$

  APP=INC=0.25, arbitrarily select APP.

  APP $\supset OK$

  This rule covers negative instances 1, 8, and 9

  $\rightarrow$ we need another atom to the antecedent.

- Select RATING, and we get

  $r_{\text{RATING}} = \frac{1}{2} = 0.5$

  $r_{\text{INC}} = \frac{1}{2} = 0.5$

  $r_{\text{BAL}} = \frac{0}{1} = 0.0$

  APP $\land$ RATING $\supset OK$

  This rule covers negative example 9.

- Finally we get

  APP $\land$ RATING $\land$ INC $\supset OK$

  which covers only positive instances with first rule, so we are finished.
Pseudocode of this rule learning process.

- Generic Separate-and-conquer algorithm (GSCA)

\[ \Xi \text{ is the initial training set of instances of binary-valued features each} \]
\[ \text{labeled by the value of an atom, } \gamma \]
\[ \pi \text{ is a set of rules to be learned} \]
\[ \rho \text{ is one of the rules; it has } \gamma \text{ as its consequent and (the conjunction of atoms) } \Gamma \text{ as its antecedent} \]
\[ \alpha \text{ is an atom drawn from one of the features in } \Xi \]
Learning Propositional Calculus Rules (7)

GSCA

1. Initialize $\mathcal{E}_{\text{cur}} \leftarrow \mathcal{E}$.
2. Initialize $\pi \leftarrow$ empty set of rules.
3. repeat The outer loop adds rules until $\pi$ covers all (or most) of the positive instances.
4. Initialize $\Gamma \leftarrow T$.
5. Initialize $\rho \leftarrow \Gamma \cup \gamma$.
6. repeat The inner loop adds atoms to $\Gamma$ until $\rho$ covers only (or mainly) positive instances.
7. Select an atom $\alpha$ to add to $\Gamma$. This is a nondeterministic choice point that can be used for backtracking.
8. $\Gamma \leftarrow \Gamma \land \alpha$.
9. until $\rho$ covers only (or mainly) positive instances in $\mathcal{E}_{\text{cur}}$.
10. $\pi \leftarrow \pi, \rho$. We add the rule $\rho$ to the set of rules.
11. $\mathcal{E}_{\text{cur}} \leftarrow \mathcal{E}_{\text{cur}} -$ (the positive instances in $\mathcal{E}_{\text{cur}}$ covered by $\pi$).
12. until $\pi$ covers all (or most) of the positive instance in $\mathcal{E}$.