Chapter 11

Explanation-based Learning

11.1 Introduction

The inductive learning algorithms discussed in Chapters 1-4 generalize on the basis of regularities in training data. These algorithms are often referred to as *similarity based*, i.e. generalization is primarily determined by the syntactical structure of the training examples. The use of domain knowledge is limited to specifying the syntax of the hypothesis language and exploring the hierarchy of the attribute values.

Typically a learning system which uses domain knowledge is expected to have some ability to solve problems. Then the point of learning is to improve the system's knowledge or system's performance using that knowledge. This task could be seen as *knowledge reformulation* or *theory revision*.

Explanation-based learning (EBL) uses a domain theory to construct an explanation of the training example, usually a proof that the example logically follows from the theory. Using this proof the system filters the noise, selects only the relevant to the proof aspects of the domain theory, and organizes the training data into a systematic structure. This makes the system more efficient in later attempts to deal with the same or similar examples.

11.2 Basic concepts of EBL

1. *Target concept*. The task of the learning system is to find an effective definition of this concept. Depending on the specific application the target concept could be a classification, theorem to be proven, a plan for achieving goal, or heuristic to make a problem solver more efficient.

2. *Training example*. This is an instance of the target concept.

3. *Domain theory*. Usually this is a set of rules and facts representing domain knowledge. They are used to explain how the training example is an instance of the target concept.

4. *Operationality criteria*. Some means to specify the form of the concept definition.

11.3 Example

Consider the following *domain theory* in the form of Prolog facts and rules.

```
on(object1,object2).

isa(object1,box).
isa(object2,table).
isa(object3,box).

color(object1,red).
color(object2,blue).

volume(object1,1).
volume(object3,6).

density(object1,2).
density(object3,2).


safe_to_stack(X,Y) :- lighter(X,Y).

lighter(O1,O2) :- weight(O1,W1), weight(O2,W2), W1 < W2.

weight(O,W) :- volume(O,V), density(O,D), W is V * D.
weight(O,5) :- isa(O,table).
```

The *operational criterion* defines the predicates which can be used to construct the concept definition. Those include the facts and arithmetic predicates such as ">", "<" and "*is*" (actually this set can be extended to all Prolog built-in predicates).

Depending on the training example - an instance of a $safe\_to\_stack$ predicate, the following definitions of the *target concept* can be obtained:

- For training example $safe\_to\_stack(object1, object3)$ the concept definition is as follows:

```
safe_to_stack(A,B) :-
   volume(A,C),
   density(A,D),
   E is C * D,
   volume(B,F),
   density(B,G),
   H is D * G,
   E < H.
```

- For training example $safe\_to\_stack(object1, object2)$ the concept definition is as follows:

```
safe_to_stack(A,B) :-
   volume(A,C),
   density(A,D),
   E is C * D,
   isa(B,table),
   E < 5
```

- For training example $safe\_to\_stack(object2, object3)$ the concept definition is as follows:

```
safe_to_stack(A,B) :-
   isa(A,table),
   volume(B,C),
   density(B,D),
   E is C * D,
   5 < E
```

The process of building the target concept definition is accomplished by generalization of the proof tree for the training example. This process is called *goal regression*. In the particular case described above, the goal regression is simply variabalization (replacing same constants with same variables) of the leaves of the Prolog proof tree and then using that leaves as goals in the body of the target concept.

11.4 Discussion

In the form discussed here EBL can be seen as *partial evaluation*. In terms of Prolog this technique is called some times *unfolding*, i.e. replacing some body literals with the bodies of the clauses they match, following the order in which Prolog reduces the goals. Hence in its pure form EBL doesn't learn anything new, i.e. all the rules inferred belong to the *deductive closure* of the domain theory. This means that these rules can be inferred from the theory without using the training example at all. The role of the training example is only to focus the theorem prover on relevant aspects of the problem domain. Therefore EBL is often viewed as a form of *speed up learning* or *knowledge reformulation*.

Consequently EBL can be viewed not as a form of generalization, but rather as *specialization*, because the rule produced is more specific than a theory itself (it is applicable only to one example).

All these objections however do not undermine the EBL approach as a Machine Learning one. There are three responses to them:

1. There are small and well defined theories, however practically inapplicable. For example, consider the game of chess. The rules of chess combined with an ability to perform unlimited look-ahead on the board states will allow a system to play well. Unfortunately this approach is not practically useful. An EBL system given well chosen training examples will not add anything new to the rules of chess playing, but will learn actually some heuristics to apply these rules, which might be practically useful.

2. An interesting application of the EBL techniques is to incomplete or incorrect theories. In such cases an incomplete (with some failed branches) or incorrect proof can indicate the deficiency of the theory and give information how to refine or complete it.

3. An integration of EBL and other similarity based approaches could be fruitful. This can be used to refine the domain theory by building explanations of successful and failed examples and passing them as positive and negative examples correspondingly to an inductive learning system.