CS601PC: MACHINE LEARNING
III Year B.Tech. CSE II-Sem.
UNIT- V

Analytical Learning-1- Introduction, learning with perfect domain theories: PROLOG-EBG, remarks on explanation-based learning, explanation-based learning of search control knowledge.

Analytical Learning-2- Using prior knowledge to alter the search objective, using prior knowledge to augment search operators.

Combining Inductive and Analytical Learning – Motivation, inductive-analytical approaches to learning, using prior knowledge to initialize the hypothesis.

TEXT BOOKS:

1. Machine Learning–Tom M.Mitchell-MGH (Page Nos. 307 to 364)

ANALYTICAL LEARNING-1

Introduction:

An analytical learning method is also called **explanation-based learning (EBL)**. In explanation-based learning, **prior knowledge** is used to analyze, or explain, how each observed training example satisfies the target concept. This explanation is then used to **distinguish the relevant features of the training example from the irrelevant**, so that examples can be generalized based on logical rather than statistical reasoning.

Learning algorithms accept explicit prior knowledge as an input, **in addition** to the input training data. Explanation-based learning is one such approach. It uses prior knowledge to analyze, or explain, each training example in order to infer which example features are relevant to the target function and which are irrelevant.

Explanation based learning uses **prior knowledge to reduce the complexity** of the hypothesis space to be searched, thereby reducing sample complexity and **improving generalization accuracy** of the learner.

Consider first example, **Knowledge about the legal rules of chess:**

knowledge of which moves are legal for the knight and other pieces, the fact that players must alternate moves in the game, and the fact that to win the game one player must capture his opponent's king. Note that given just this prior knowledge it is possible in principle (domain theory) to calculate the optimal chess move for any board position. That is, In analytical learning, the learner must output a hypothesis that is consistent with both the training data and the domain theory.

Difference between analytical and inductive learning methods:

- In inductive learning, the learner is given a hypothesis space H from which it must select an output hypothesis, and a set of training examples $D = \{\langle x_1, f(x_1)\rangle, \ldots \langle x_n, f(x_n)\rangle\}$ where $f(x_i)$ is the target value for the instance x_i. The desired output of the learner is a hypothesis h from H that is consistent with these training examples.
- In analytical learning, the input to the learner includes the same hypothesis space H and training examples D as for inductive learning. In addition, the learner is provided an additional input: A *domain theory* B consisting of background knowledge that can be used to explain observed training examples. The desired output of the learner is a hypothesis h from H that is consistent with both the training examples D and the domain theory B.

Consider Second example,

An instance space X in which each instance is a pair of physical objects. Each of the two physical objects in the instance is described by the predicates Color, Volume, Owner, Material, Type, and Density, and the relationship between the two objects is described by the predicate On.

Given this instance space, the task is to learn the target concept "pairs of physical objects, such that one can be stacked safely on the other," denoted by the predicate **SafeToStack(x,y)**. Learning this target concept might be useful, for example, to a robot system that has the task of storing various physical objects within a limited workspace. The full definition of this analytical learning task is given in Table 11.1.

Given:

- Instance space X: Each instance describes a pair of objects represented by the predicates $Type$, $Color$, $Volume$, $Owner$, $Material$, $Density$, and On.
- Hypothesis space H: Each hypothesis is a set of Horn clause rules. The head of each Horn clause is a literal containing the target predicate *SafeToStack*. The body of each Horn clause is a conjunction of literals based on the same predicates used to describe the instances, as well as the predicates $LessThan$, $Equal$, $GreaterThan$, and the functions $plus$, $minus$, and $times$. For example, the following Horn clause is in the hypothesis space:

$$SafeToStack(x, y) \leftarrow Volume(x, vx) \wedge Volume(y, vy) \wedge LessThan(vx, vy)$$

- Target concept: *SafeToStack(x,y)*
- Training Examples: A typical positive example, $SafeToStack(Obj1, Obj2)$, is shown below:

$On(Obj1, Obj2)$
$Type(Obj1, Box)$
$Type(Obj2, Endtable)$
$Color(Obj1, Red)$
$Color(Obj2, Blue)$
$Volume(Obj1, 2)$
$Owner(Obj1, Fred)$
$Owner(Obj2, Louise)$
$Density(Obj1, 0.3)$
$Material(Obj1, Cardboard)$
$Material(Obj2, Wood)$

- Domain Theory B:

$SafeToStack(x, y) \leftarrow \neg Fragile(y)$
$SafeToStack(x, y) \leftarrow Lighter(x, y)$
$Lighter(x, y) \leftarrow Weight(x, wx) \wedge Weight(y, wy) \wedge LessThan(wx, wy)$
$Weight(x, w) \leftarrow Volume(x, v) \wedge Density(x, d) \wedge Equal(w, times(v, d))$
$Weight(x, 5) \leftarrow Type(x, Endtable)$
$Fragile(x) \leftarrow Material(x, Glass)$
...

Determine:

- A hypothesis from H consistent with the training examples and domain theory.

TABLE 11.1
An analytical learning problem: *SafeToStack(x,y)*.

LEARNING WITH PERFECT DOMAIN THEORIES: PROLOG-EBG

We consider explanation-based learning from **domain theories** that are **perfect**, that is, **domain theories that are correct and complete**. A domain theory is said to be **correct** if each of its assertions is a **truthful statement** about the world. A domain theory is said to be **complete** with respect to a **given target concept** and instance space, if the domain theory covers **every positive example** in the instance space. Put another way, it is complete if every instance that satisfies the target concept can be proven by the domain theory to satisfy it. Completeness does not require that the domain theory be able to prove that negative examples do not satisfy the target concept. So, Completeness includes full coverage of both positive and negative examples by the domain theory.

An algorithm called PROLOG-EBG, is representative of explanation-based learning algorithms and it is a sequential covering algorithm.

PROLOG-EBG operates by learning a single **Horn clause rule**, removing the positive training examples covered by this rule, then iterating this process on the remaining positive examples until no further positive examples remain uncovered. When given a complete and correct domain theory, PROLOG-EBG is guaranteed to output a hypothesis (set of rules) that is itself correct and that covers the observed positive training examples. For any set of training examples, the hypothesis output by **PROLOG-EBG constitutes a set of logically sufficient conditions for the target concept, according to the domain theory**.

A clause with at most one positive (unnegated) literal is called a Horn Clause.

Deductive Database: A type of database which can make conclusions based on sets of well-defined rules, stored in database.

For each new positive training example that is not yet covered by a learned Horn clause, it forms a new Horn clause by:

(1) explaining the new positive training example,
(2) analyzing this explanation to determine an appropriate generalization, and
(3) refining the current hypothesis by adding a new Horn clause rule to cover this positive example, as well as other similar instances.

(1) explaining the new positive training example:

PROLOG-EBG(*TargetConcept*, *TrainingExamples*, *DomainTheory*)

- *LearnedRules* ← {}
- *Pos* ← the positive examples from *TrainingExamples*
- for each *PositiveExample* in *Pos* that is not covered by *LearnedRules*, do
  1. *Explain:*
     - *Explanation* ← an explanation (proof) in terms of the *DomainTheory* that *PositiveExample* satisfies the *TargetConcept*
  2. *Analyze:*
     - *SufficientConditions* ← the most general set of features of *PositiveExample* sufficient to satisfy the *TargetConcept* according to the *Explanation*.
  3. *Refine:*
     - *LearnedRules* ← *LearnedRules* + *NewHornClause*, where *NewHornClause* is of the form

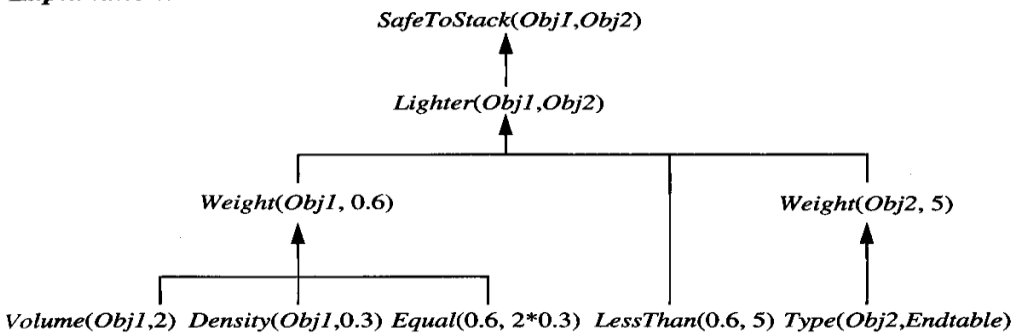
$$TargetConcept \leftarrow SufficientConditions$$

- Return *LearnedRules*

TABLE 11.2
The explanation-based learning algorithm PROLOG-EBG. For each positive example that is not yet covered by the set of learned Horn clauses (*LearnedRules*), a new Horn clause is created. This new Horn clause is created by (1) explaining the training example in terms of the domain theory, (2) analyzing this explanation to determine the relevant features of the example, then (3) constructing a new Horn clause that concludes the target concept when this set of features is satisfied.

(2) analyzing this explanation to determine an appropriate generalization

Explanation:



Training Example:

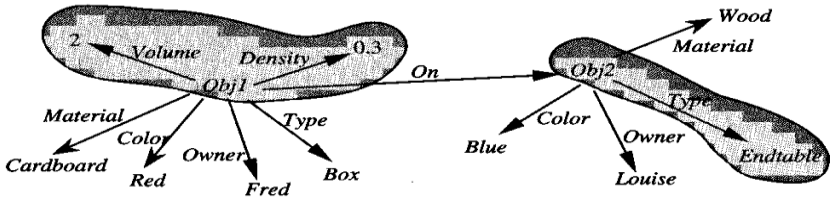


FIGURE 11.2
Explanation of a training example. The network at the bottom depicts graphically the training example *SafeToStack*(*Obj1*, *Obj2*) described in Table 11.1. The top portion of the figure depicts the explanation of how this example satisfies the target concept, *SafeToStack*. The shaded region of the training example indicates the example attributes used in the explanation. The other, irrelevant, example attributes will be dropped from the generalized hypothesis formed from this analysis.

(3) Refining the current hypothesis by adding a new Horn clause rule to cover this positive example, as well as other similar instances.

The current hypothesis at each stage consists of the set of Horn clauses. At each stage, the sequential covering algorithm picks a new positive example that is not yet covered by the current Horn clauses, explains this new example, and formulates a new rule according to the procedure described above.

Only positive examples are covered in the algorithm as we have defined it, and the learned set of Horn clause rules predicts only positive examples. A new instance is classified as negative if the current rules fail to predict that it is positive. This is in keeping with the standard negation-as-failure approach used in Horn clause inference systems such as PROLOG.

REMARKS ON EXPLANATION-BASED LEARNING: (Properties of Explanation Based Learning Algorithm)

1. Unlike inductive methods, PROLOG-EBG produces justified general hypotheses by using prior knowledge to analyze individual examples.
2. The explanation of how the example satisfies the target concept determines which example attributes are relevant: those mentioned by the explanation.
3. The further analysis of the explanation, regressing the target concept to determine its weakest preimage with respect to the explanation, allows deriving more general constraints on the values of the relevant features.
4. Each learned Horn clause corresponds to a sufficient condition for satisfying the target concept. The set of learned Horn clauses covers the positive training examples encountered by the learner, as well as other instances that share the same explanations.
5. The generality of the learned Horn clauses will depend on the formulation of the domain theory and on the sequence in which training examples are considered.
6. PROLOG-EBG implicitly assumes that the domain theory is correct and complete. If the domain theory is incorrect or incomplete, the resulting learned concept may also be incorrect.

EXPLANATION-BASED LEARNING OF SEARCH CONTROL KNOWLEDGE:

The largest scale attempts to apply explanation-based learning have addressed the problem of learning to control search, or what is sometimes called "speedup" learning. For example, playing games such as chess involves searching through a vast space of possible moves and board positions to find the best move. Many practical scheduling and optimization problems are easily formulated as large search problems, in which the task is to find some move toward the goal state. In such problems the definitions of the legal search operators, together with the definition of the search objective, provide a complete and correct domain theory for learning search control knowledge.

First we have to formulate the problem of learning search control and then we can apply explanation based learning.

Consider a general search problem where
S is the set of possible search states,
O is a set of legal search operators that transform one search state into another,
G is a predicate defined over S that indicates which states are goal states.

The problem in general is to find a sequence of operators that will transform an arbitrary initial state s_i to some final state s_f that satisfies the goal predicate G.

PRODIGY is a domain-independent planning system that accepts the definition of a problem domain in terms of the state space S and operators O. It then solves problems of the form "find a sequence of operators that leads from initial state s_i to a state that satisfies goal predicate G." PRODIGY use s a means-ends planner that decomposes problems into subgoals, solves them, then combines their solutions into a solution for the full problem.

The net effect is that PRODIGY uses domain-independent knowledge about possible subgoal conflicts, together with domain-specific knowledge of specific operators, to learn useful domain-specific planning rules.

SOAR supports a broad variety of problem-solving strategies that subsumes PRODIGY'S means-ends planning strategy. SOAR uses a variant of explanation-based learning called **chunking** to extract the general conditions under which the same explanation applies. SOAR has been applied in a great number of problem domains and has also been proposed as a psychologically plausible model of human learning processes

PRODIGY and SOAR demonstrate that explanation-based learning methods can be successfully applied to acquire search control knowledge in a variety of problem domains.

For example, in a series of robot block-stacking problems, PRODIGY encountered 328 opportunities for learning a new rule, but chose to exploit only 69 of these, and eventually reduced the learned rules to a set of 19, once low-utility rules were eliminated.

COMBINING INDUCTIVE AND ANALYTICAL LEARNING

MOTIVATION:

Purely inductive learning methods (decision tree, Back propagation) formulate general hypotheses over the training examples. Purely analytical methods (PROLOG-EBG) use prior knowledge to derive general hypotheses.

The method that combine inductive and analytical mechanisms to obtain the benefits of both approaches: better generalization accuracy when prior knowledge is available and reliance on observed training data to overcome shortcomings in prior knowledge.

Purely analytical learning methods offer the advantage of generalizing more accurately from less data by using prior knowledge to guide learning. However, they can be misled when given incorrect or insufficient prior knowledge. Purely inductive methods offer the advantage that they require no explicit prior knowledge and learn regularities based solely on the training data. However, they can fail when given insufficient training data, and can be misled by the implicit inductive bias they must adopt in order to generalize beyond the observed data.

	Inductive learning	Analytical learning
Goal:	Hypothesis fits data	Hypothesis fits domain theory
Justification:	Statistical inference	Deductive inference
Advantages:	Requires little prior knowledge	Learns from scarce data
Pitfalls:	Scarce data, incorrect bias	Imperfect domain theory

TABLE 12.1
Comparison of purely analytical and purely inductive learning.

FIGURE 12.1
A spectrum of learning tasks. At the left extreme, no prior knowledge is available, and purely inductive learning methods with high sample complexity are therefore necessary. At the rightmost extreme, a perfect domain theory is available, enabling the use of purely analytical methods such as PROLOG-EBG. Most practical problems lie somewhere between these two extremes.

INDUCTIVE-ANALYTICAL APPROACHES TO LEARNING:

The Learning Problem

Given:

- A set of training examples D, possibly containing errors
- A domain theory B, possibly containing errors
- A space of candidate hypotheses H

Determine:

- A hypothesis that best fits the training examples and domain theory

Solution:

We can define measures of hypothesis error with respect to the data and with respect to the domain theory.

$$\underset{h \in H}{\operatorname{argmin}} \; k_D error_D(h) + k_B error_B(h)$$

$error_D(h)$ is defined to be the proportion of **examples D** that are misclassified by h.

$error_B(h)$ is defined with respect to a **domain theory B** that h will disagree with B on the classification of a randomly drawn instance.

The learning problem is to minimize some combined measure of the error of the hypothesis over the data and the domain theory.

Three different methods explored for using prior knowledge to alter the **search** performed by purely inductive methods.

A. USING PRIOR KNOWLEDGE TO INITIALIZE THE HYPOTHESIS:

A.1. KBANN (Knowledge-Based Artificial Neural Network) algorithm

In this approach the domain theory B is used to construct an initial hypothesis h_0 that is consistent with B. A standard inductive method is then applied, starting with the initial hypothesis ho. For example, the KBANN system learns artificial neural networks in this way. It uses prior knowledge to design the interconnections and weights for an initial network, so that this initial network is perfectly consistent with the given domain theory. This initial network hypothesis is then refined inductively using the BACKPROPAGATlON aIgorithm and available data. **Beginning the search at a hypothesis consistent with the domain theory makes it more likely that the final output hypothesis will better fit this theory**.

ANALYTICAL LEARNING-2

B. USING PRIOR KNOWLEDGE TO ALTER THE SEARCH OBJECTIVE:

B.1. The TANGENTPROP Algorithm

B.2.The EBNN (Explanation-Based Neural Network learning) algorithm

In this approach, the goal criterion G is modified to require that the output hypothesis fits the domain theory as well as the training examples. For example, the EBNN system learns neural networks in this way. Whereas inductive learning of neural networks performs gradient descent search to minimize the squared error of the network over the training data, EBNN performs gradient descent to optimize a different criterion. This modified criterion includes an additional term that measures the error of the learned network relative to the domain theory.

C. USING PRIOR KNOWLEDGE TO AUGMENT SEARCH OPERATORS:

C.1. The FOCL Algorithm

In this approach, the set of search operators O is altered by the domain theory. For example, the FOCL system learns sets of Horn clauses in this way. It is based on the inductive system FOIL, which conducts a greedy search through the space of possible Horn clauses, at each step revising its current hypothesis by adding a single new literal. FOCL uses the domain theory to expand the set of alternatives available when revising the hypothesis, allowing the addition of

multiple literals in a single search step when warranted by the domain theory. In this way, FOCL allows single-step moves through the hypothesis space that would correspond to many steps using the original inductive algorithm. These "macro-moves" can dramatically alter the course of the search, so that the final hypothesis found consistent with the data is different from the one that would be found using only the inductive search steps.

DETAILED ANALYSIS OF THREE METHODS:

A.1. KBANN (Knowledge-Based Artificial Neural Network) algorithm

---

KBANN(*Domain_Theory*, *Training_Examples*)
Domain_Theory: Set of propositional, nonrecursive Horn clauses.
Training_Examples: Set of ⟨*input output*⟩ *pairs of the target function.*

Analytical step: Create an initial network equivalent to the domain theory.

1. For each instance attribute create a network input.
2. For each Horn clause in the *Domain_Theory*, create a network unit as follows:
   - Connect the inputs of this unit to the attributes tested by the clause antecedents.
   - For each non-negated antecedent of the clause, assign a weight of W to the corresponding sigmoid unit input.
   - For each negated antecedent of the clause, assign a weight of $-W$ to the corresponding sigmoid unit input.
   - Set the threshold weight w_0 for this unit to $-(n - .5)W$, where n is the number of non-negated antecedents of the clause.
3. Add additional connections among the network units, connecting each network unit at depth i from the input layer to all network units at depth $i + 1$. Assign random near-zero weights to these additional connections.

Inductive step: Refine the initial network.

4. Apply the BACKPROPAGATION algorithm to adjust the initial network weights to fit the *Training_Examples*.

TABLE 12.2
The KBANN algorithm. The domain theory is translated into an equivalent neural network (steps 1–3), which is inductively refined using the BACKPROPAGATION algorithm (step 4). A typical value for the constant W is 4.0.

Given:

- A set of training examples
- A domain theory consisting of nonrecursive, propositional Horn clauses

Determine:

- An artificial neural network that fits the training examples, biased by the domain theory.

The two stages of the KBANN algorithm are first to create an artificial neural network that perfectly fits the domain theory and second to use the BACKPROPAGATION algorithm to refine this initial network to fit the training examples.

B.1. The TANGENTPROP Algorithm

The approach begins the gradient descent search with a hypothesis that perfectly fits the domain theory, then perturbs this hypothesis as needed to maximize the fit to the training data.

An alternative way of using prior knowledge is to incorporate it into the Error criterion minimized by gradient descent, so that the network must fit a combined function of the training data and domain theory.

TANGENTPROP algorithm accommodates domain knowledge expressed as derivatives of the target function with respect to transformations of its inputs.

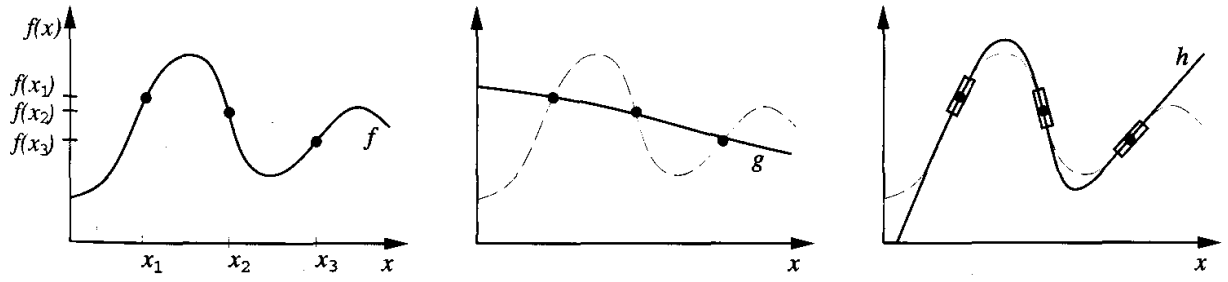


FIGURE 12.5
Fitting values and derivatives with TANGENTPROP. Let f be the target function for which three examples $\langle x_1, f(x_1)\rangle$, $\langle x_2, f(x_2)\rangle$, and $\langle x_3, f(x_3)\rangle$ are known. Based on these points the learner might generate the hypothesis g. If the derivatives are also known, the learner can generalize more accurately h.

Training set size	Percent error on test set	
	TANGENTPROP	BACKPROPAGATION
10	34	48
20	17	33
40	7	18
80	4	10
160	0	3
320	0	0

TABLE 12.4
Generalization accuracy for TANGENTPROP and BACKPROPAGATION, for handwritten digit recognition. TANGENTPROP generalizes more accurately due to its prior knowledge that the identity of the digit is invariant of translation. These results are from Simard et al. (1992).

It is interesting to compare the search through hypothesis space performed by TANGENTPROP, KBANN, and BACKPROPAGATION. TANGENTPROP incorporates prior knowledge to influence the hypothesis search by altering the objective function to be minimized by gradient descent. This corresponds to altering the goal of the hypothesis space search.

B.2.The EBNN (Explanation-Based Neural Network learning) algorithm

EBNN Algorithm builds on the TANGENTPROP algorithm in two significant ways.

First, instead of relying on the user to provide training derivatives, EBNN computes training derivatives itself for each observed training example. These training derivatives are calculated by explaining each training example in terms of a given domain theory, then extracting training derivatives from this explanation.

Second, EBNN addresses the issue of how to weight the relative importance of the inductive and analytical components of learning.

The inputs to EBNN include

(1) a set of training examples of the form (xi, f (xi)) with no training derivatives provided, and

(2) a domain theory analogous to that used in explanation-based learning and in KBANN, but represented by a set of previously trained neural networks rather than a set of Horn clauses.

The output of EBNN is a new neural network that approximates the target function f.

This learned network is trained to fit both the training examples $(x_i, f(x_i))$ and training derivatives of f extracted from the domain theory. Fitting the training examples $(x_i, f(x_i))$ constitutes the inductive component of learning, whereas fitting the training derivatives extracted from the domain theory provides the analytical component.

The EBNN algorithm uses a domain theory expressed as a set of previously learned neural networks, together with a set of training examples, to train its output hypothesis (the target network). For each training example EBNN uses its domain theory to explain the example, then extracts training derivatives from this explanation.

EBNN provides TANGENTPROP with derivatives that it calculates from the domain theory. To see how EBNN calculates these training derivatives, consider again Figure 12.7.

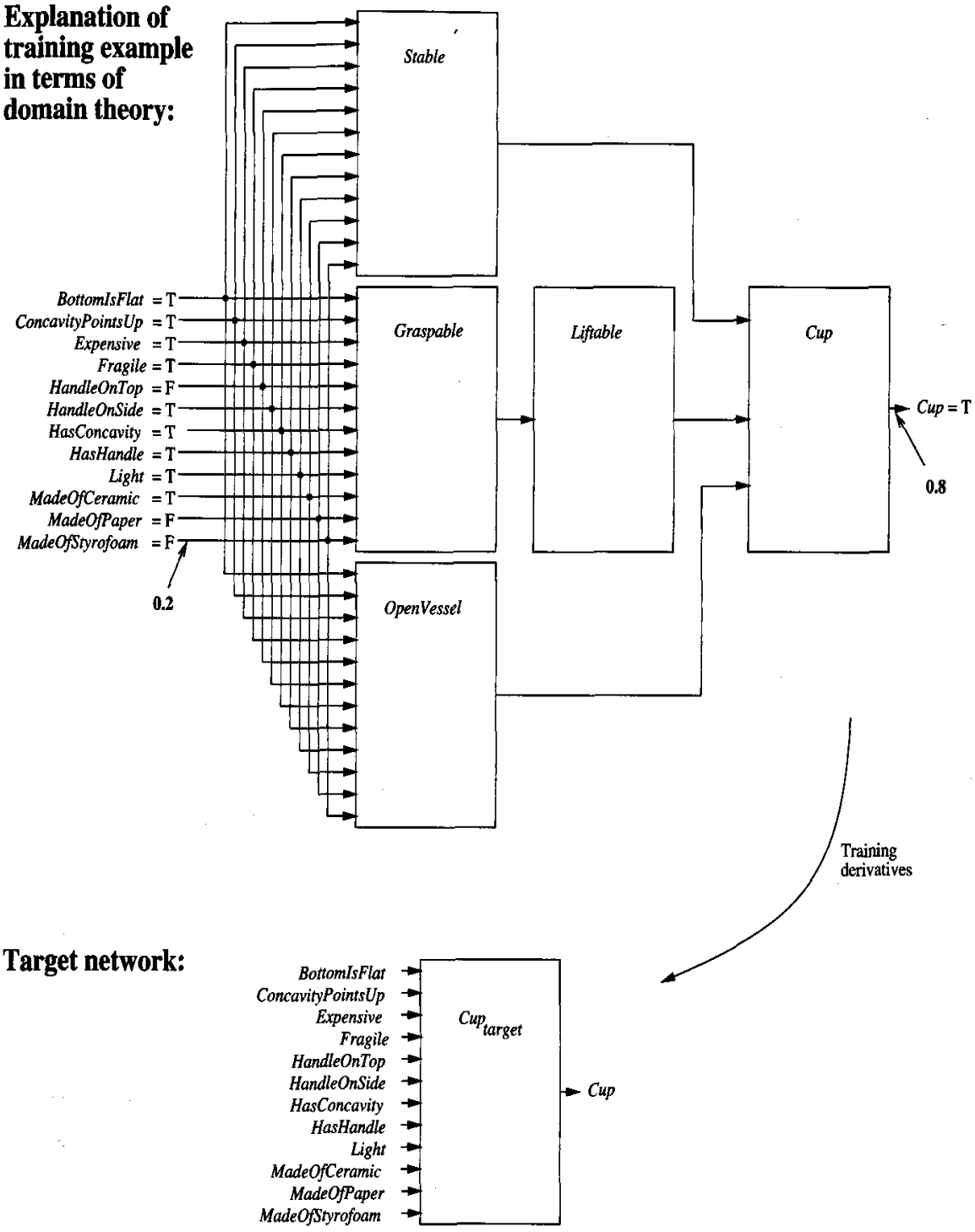


FIGURE 12.7
Explanation of a training example in EBNN. The explanation consists of a prediction of the target function value by the domain theory networks (top). Training derivatives are extracted from this explanation in order to train the separate target network (bottom). Each rectangular block represents a distinct multilayer neural network.

C.1. The FOCL Algorithm

FOCL is an extension of the purely inductive FOIL system. Both FOIL and FOCL learn a set of first-order Horn clauses to cover the observed training examples. Both systems employ a sequential covering algorithm that learns a single Horn clause, removes the positive examples covered by this new Horn clause, and then iterates this procedure over the remaining training examples.

FOIL generates each candidate specialization by adding a single new literal to the clause preconditions. FOCL uses this same method for producing candidate specializations, but also generates additional specializations based on the domain theory. The solid edges in the search tree of Figure 12.8 show the general-to-specific search steps considered in a typical search by FOIL. The dashed edge in the search tree of Figure 12.8 denotes an additional candidate specialization that is considered by FOCL and based on the domain theory.

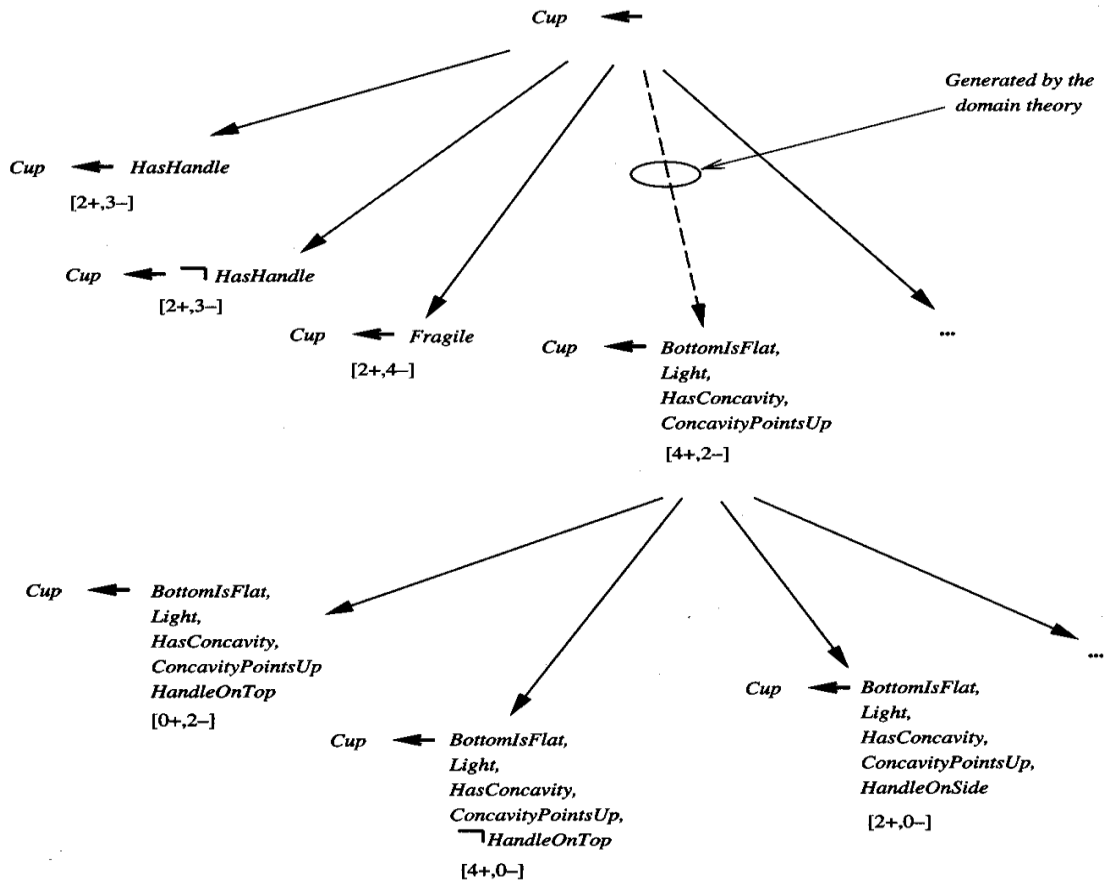


FIGURE 12.8
Hypothesis space search in FOCL. To learn a single rule, FOCL searches from general to increasingly specific hypotheses. Two kinds of operators generate specializations of the current hypothesis. One kind adds a single new literal (solid lines in the figure). A second kind of operator specializes the rule by adding a set of literals that constitute logically sufficient conditions for the target concept, according to the domain theory (dashed lines in the figure). FOCL selects among all these candidate specializations, based on their performance over the data. Therefore, imperfect domain theories will impact the hypothesis only if the evidence supports the theory. This example is based on the same training data and domain theory as the earlier KBANN example.