UNIT-V

Analytical Learning

Introduction

- Inductive learning methods, i.e. methods that generalize from observed training examples.
- The key practical limit on these inductive learners is that they perform poorly when insufficient data is available.
- One way is to develop learning algorithms that 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.
- These explanation helps in generalizing more accurately than inductive learning
- Explanation- based learning uses prior knowledge to reduce the complexity of the hypothesis space to be searched, thereby reducing space complexity and improving generalization accuracy of the learner.

Example 1:

Let us consider the task of learning to play chess. Here we are making our program to recognize the game position i.e. target concept as "chessboard positions in which black will lose its queen within two moves." Figure 1 shows the positive samples of training concept.

Now if we take inductive learning method to perform this task, it would be difficult because the chess board is fairly complex (32 pieces can be on any 64 square) and particular patterns i.e. to place the pieces in the relative positions (placing them exactly following game rules).So for all these we need to provide thousand of training examples similar to figure 1 to expect an inductively learned hypothesis to generalize correctly to new situations.

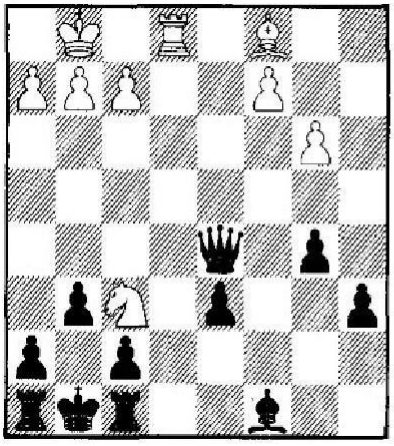


FIGURE 11.1
A positive example of the target concept "chess positions in which black will lose its queen within two moves." Note the white knight is simultaneously attacking both the black king and queen. Black must therefore move its king, enabling white to capture its queen.

Even after considering only the single example shown in Figure 1 , most would be willing to suggest a general hypothesis for the target concept, such as "board positions in which the black king and queen are simultaneously attacked," and would not even consider the (equally consistent) hypothesis "board positions in which four white pawns are still locations."So we can't generalize successfully with that one example.

in their original

Now why to consider training example as positive target concept? "Because white's knight is attacking both the king and queen, black must move out of check, thereby al- lowing the knightto capture the queen." They provide the information needed to rationally generalize from the details of the training example to a correct general hypothesis.

What knowledge is needed to learn chess? It is simply 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.

However, in practice this calculation can be frustratingly complex and despite the fact that we humans ourselves possess this complete, perfect knowledge of chess, we remain unable to play the game optimally.

Inductive and Analytical Learning Problems

- 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 = {(xl, f (x~)), . . . (x,, f (x,))} where f (xi) is the target value for the instance xi. 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.

To illustrate, in our chess example each instance xi would describe a particular chess position,and f (xi) would be True when xi is a position for which black will lose its queen within two
moves, and False otherwise. Now we define hypothesis space H to consist of sets of Horn clauses (if-then rules) where predicates used rules refer to the positions or relative positions of specific pieces on the board. The domain theory B would consist of a formalization of the rulesof chess.

Note in analytical learning, the learner must output a hypothesis that is consistent with both the training data and the domain theory.

Example2:

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 1.
SafeToStack

Here we chosen The example 2 is about Analytical Learning problem SafeToStack (x, y).

hypothesis space H which is set of hypothesis from first order if- then rules (i e. Horn Clause). The example Horn clause hypothesis shown in the table asserts that it is Safe To Stack any object on any object y, if the Volume of x is Less than the Volume of y. The Horn clause hypothesis can refer to any of the predicates used to describe the instances, as well as several additional predicates and functions. One such example is Safe To Stack(obj1, obj2) shown in table.

Here domain theory considered will explain certain pairs of objects can be safely stacked on one another (same as chess example it takes all the rules of the game). The domain theory shown in the table includes assertions such as "it is safe to stack x on y if y is not Fragile. Here the domain theory also uses subsequent theories i.e. predicators such as Lighter has more primitive attributes like weight, vol, etc which help sclassify. to generalize more accurately and the given is sufficient to

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.
- But our definition of completeness does not require that the domain theory be able to prove that negative examples do not satisfy the target concept.
- So we now with help of PROLOG-EBG explain definition of completeness includes full coverage of both positive and negative examples by the domain theory.

PROLOG-EBG Algorithm:

PROLOG-EBG is a sequential covering algorithm that considers the training data incrementally.

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*

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

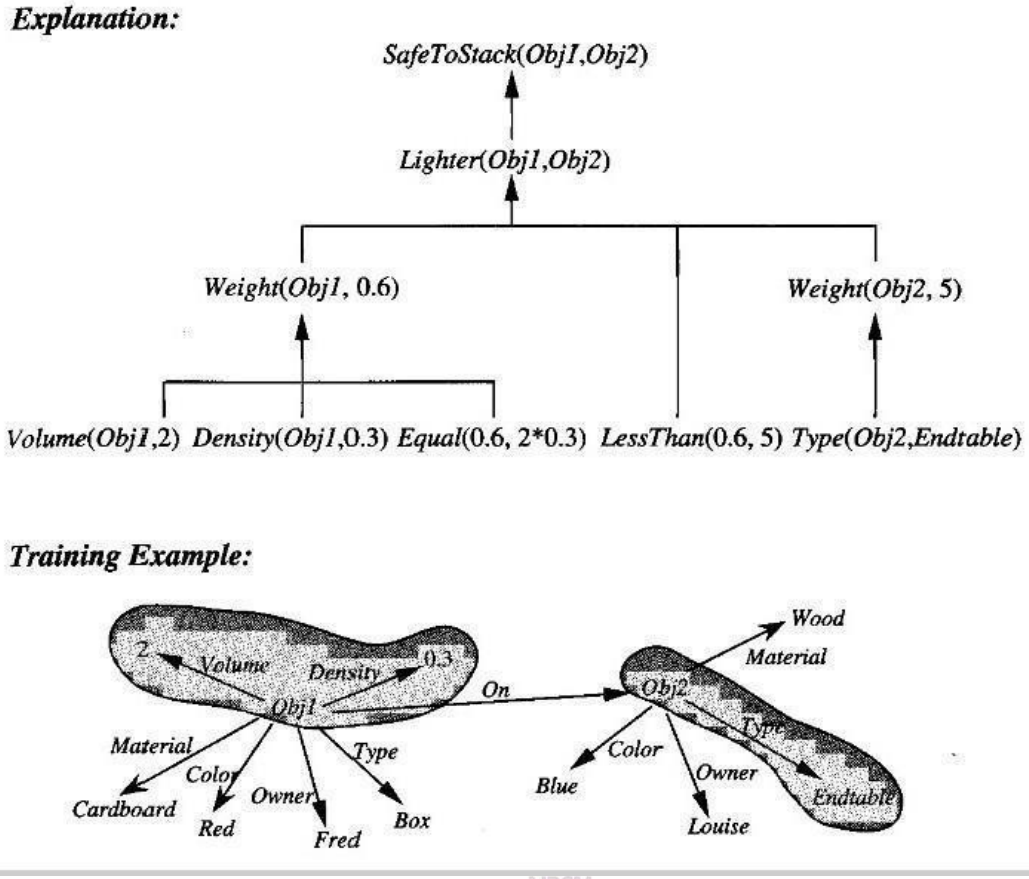


Fig 2. Explanation of training example

The bottom of this figure depicts in graphical form of +ve training example Sa eToStack(Objl , 0bj2) from Table 1. The top of the figure depicts the explanation constructed for this training example. Notice the explanation, or proof, states that it is SafeToStackObjl on 0bj2 because Objl is Lighter than Obj2. Furthermore, Objl is known to be Lighter, because its Weight can be inferred from its Density and Volume, and because the Weight of 0bj2 can be inferred from the default weight of an Endtable. The specific Horn clauses that underlie this explanation are shown in the domain theory of Table 1 . Notice that the explanation mentions only a small fraction ofthe known attributes of Objl and 0bj2 (i.e., those attributes corresponding to the shaded region in
the figure). While only a single explanation is possible for the training exa ple and domain
theory shown here, in general there may be multiple possible explanations. In such cases, any or all of the explanations may be used. In the case of PROLOG-EBG, the explanation is generated using a backward chaining search as performed by PROLOG. PROLOG, halts once it finds the first valid proof.

For example, the explanation of Figure 2 refers to the Density of Objl, but not to its Owner. Therefore, the hypothesis for SafeToStack(x,y) should include Density(x, 0.3), but not Owner(x,

Fred). By collecting just the features mentioned in the leaf nodes of the explanation in Figure 2 and substituting variables x and y for Objl and Obj2, we can form a general rule that is justified by the domain theory:

SafeToStack(x, y) ← Volume(x, 2) ^ Density(x, 0.3) ^ Type(y, Endtable)

The body of the above rule includes each leaf node in the proof tree, except for the leaf nodes
"Equal(0.6, times(2,0.3)" and "LessThan(0.6,5)." We omit these two because they are by
definition always satisfied, independent of x and y.

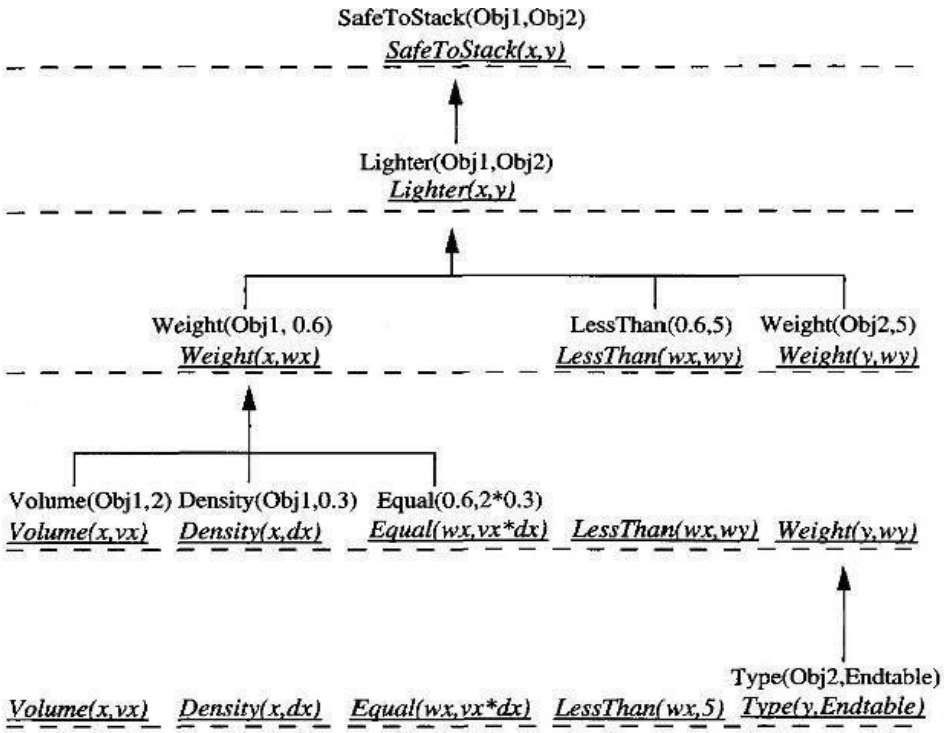
The above rule constitutes a significant generalization of the training example, because it omits many properties of the example (e.g., the Color of the two objects) that are irrelevant to the targetconcept. PROLOG- EBG computes the most general rule that can be justified by the explanation, by computing the weakest preimage of the explanation, defined as follows:

Definition: The **weakest preimage** of a conclusion C with respect to a proof P is the most general set of initial assertions A, such that A entails C according to P.

For example, the weakest preimage of the target concept SafeToStack(x,y), with respect to the explanation from Table 1, is given by the body of the following rule. This is the most generalrule that can be justified by the explanation of Figure 2:

$$SafeToStack(x, y) \leftarrow Volume(x, vx) \wedge Density(x, dx) \wedge Equal(wx, times(vx, dx)) \wedge LessThan(wx, 5) \wedge Type(y, Endtable)$$

Notice this more general rule does not require the specific values for Volume and Density that were required by the first rule. Instead, it states a more general constraint on the values of these attributes. The below figure depicts weakest preimage of SafeToStack.



The Weakest Preimage of target concept w.r.t explanation is produced by regression. It works iteratively through explanation first computing weakest preimage then weakest preimage of resulting expression and so on. It terminates when it has completed iterating all over steps inexplanation and yields weakest condition of target concept.

REMARKS ON EXPLANATION-BASED LEARNING

- Unlike inductive methods, PROLOG-EBG produces justified general hypotheses by using prior knowledge to analyze individual examples.
- The explanation of how the example satisfies the target concept determines which example attributes are relevant: those mentioned by the explanation.
- The further analysis of the explanation, regressing the target to determine its concept

weakest preimage with respect to the explanation, allows deriving more general
constraints on the values of the relevant features.

- 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.
- 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.

There are several related perspectives on explanation-based learning that help to understand itscapabilities and limitations.

- **EBL as theory-guided generalization of examples.** EBL uses its given domain theory to generalize rationally from examples, distinguishing the relevant ex- ample attributes from the irrelevant, thereby allowing it to avoid the bounds on sample complexity that apply to purely inductive learning.
- **EBL as example-guided reformulation of theories.** The PROLOG-EBG algorithm can be viewed as a method for reformulating the domain theory into a more operational formby creating rules that (a) follow deductively from the domain theory, and (b) classifythe observed training examples in a single inference step. Thus, the learned rules can be seen as a reformulation of the domain theory classifying instances of the target concept ina single inference step.
- **EBL as "just" restating what the learner already "knows. "** In one sense, the learner in our SafeToStack example begins with full knowledge of the Safe- ToStack concept.If its initial domain theory is sufficient to explain any observed training examples, then it is also sufficient to predict their classification in advance.

EXPLANATION-BASED LEARNING OF SEARCH CONTROL KNOWLEDGE

- The practical applicability of the PROLOG-EBG algorithm is restricted by its requirement that the domain theory be correct and complete.
- This EBL can be used in search programs(ex: chess game).
- One system that employs explanationbased learning is to implement search is PRODIGY.
- PRODIGY is domain independent planning system that accepts the problem in terms of state space S and operators O.
- It then solves the problem to find sequence of operators O that lead from initial state Si to state that reach goal G.

- PRODIGY divides thesolutions to final one.problem into sub problem and solves them and combines all

    - For example, one target concept is "the set of states in which subgoal A should be solved before subgoal B." An example of a rule learned by PRODIGY for this target concept in a simple block-stacking problem domain is

IF One subgoal to be solved is $On(x, y)$, and
One subgoal to be solved is $On(y, z)$
THEN Solve the subgoal $On(y, z)$ before $On(x, y)$

The goal of block-staking problem is to stack the blocks so that they spell the word "universal." PRODIGY would decompose this problem into several subgoals to be achieved. Notice the above rule matches the subgoalsOn(U, N) and On(N, I), and recommends solving the subproblem On(N, I) before solving On(U, N). The justification for this rule (and the explanationused by PRODIGY to learn the rule) is that if we solve the subgoals in the reverse sequence, we will encounter a conflict in which we must undo the solution to the On(U, N) subgoal in order to achieve the other subgoal On(N, I).

PRODIGY learns by first encountering such a conflict, then explaining to itself the reason for this conflict and creating a rule such as the one above.

The net effect is that PRODIGY uses domain-independent knowledge about possible subgoal conflicts, together with domain-specific knowledge of specific operators (e.g., the fact that the robot can pick up only one block at a time), to learn useful domain-specific planning rules such as the one illustrated above.

USING PRIOR KNOWLEDGE TO ALTER THE SEARCH OBJECTIVE

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

EBNN Algorithm

The EBNN (Explanation-Based Neural Network learning) algorithm (Mitchell and Thrun 1993a;Thrun 1996) 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. (how to select mue).

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

$$E = \sum_i \left[(f(x_i) - \hat{f}(x_i))^2 + \mu \sum_j \left(\frac{\partial f(s_j(\alpha, x_i))}{\partial \alpha} - \frac{\partial \hat{f}(s_j(\alpha, x_i))}{\partial \alpha} \right)^2_{\alpha=0} \right]$$

Fig 4. Modified error function from tangent prop

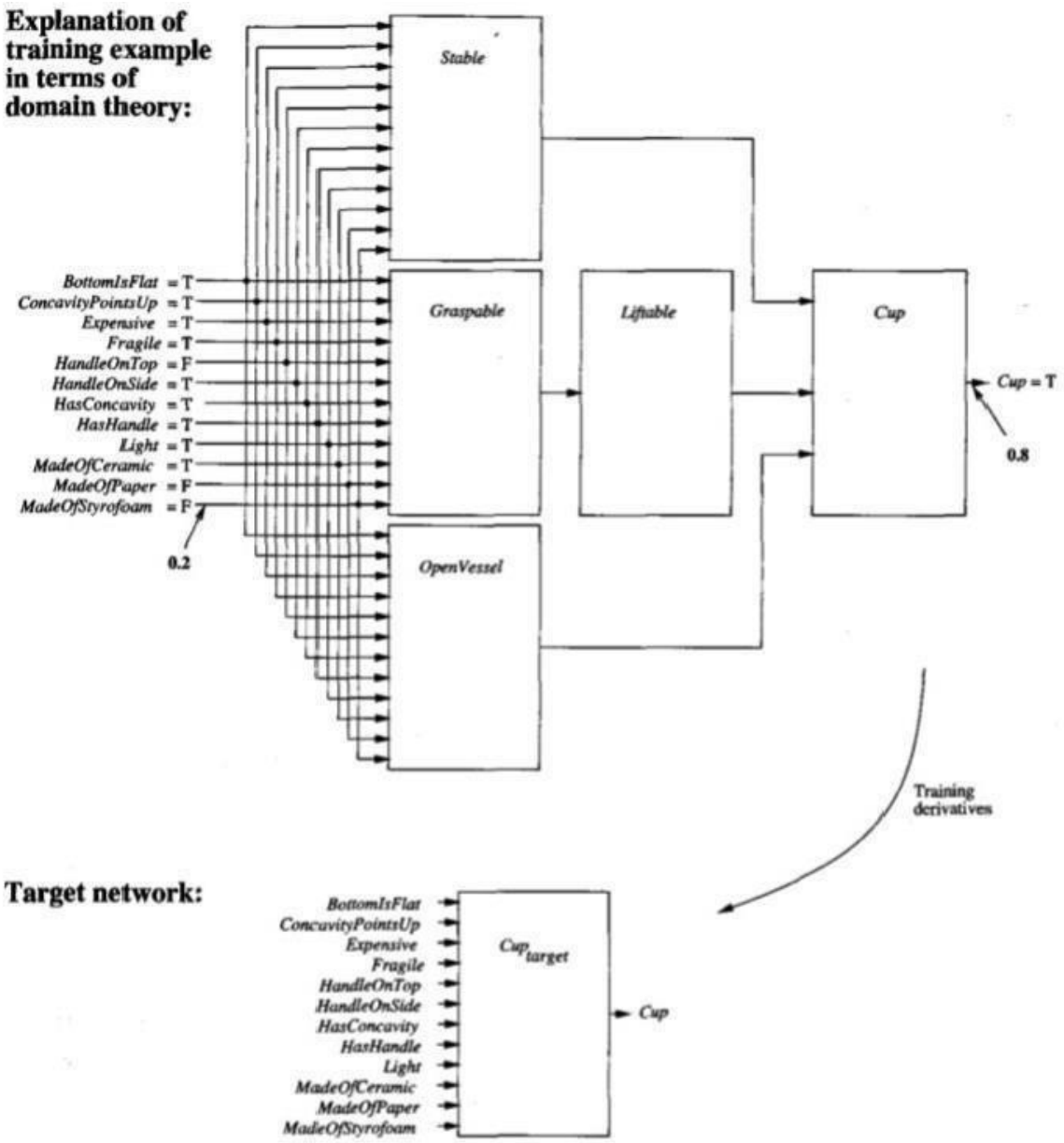
algorithm.value of μ is chosen independently for each training

example.

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.

To illustrate the type of domain theory used by EBNN, consider Figure . The top portion of this figure depicts an EBNN domain theory for the target function Cup, with each rectangular block representing a distinct neural network in the domain theory. Notice in this example there is one network for each of the Horn clauses in the symbolic domain theory of Table 1. For example, the network labeled Graspable takes as input the description of an instance and produces as output a value indicating whether the object is graspable (EBNN typically repre- sents true propositions by the value 0.8 and false propositions by the value 0.2). This network is analogous to the Horn

clause for Graspable given in Table 1. Some networks take the outputs of other networks as their inputs (e.g., the right- most network labelled Cup takes its inputs from the outputs of the Stable, Liftable, and OpenVessel networks). Thus, the networks that make up the domain theory can be chained together to infer the target function value for the input instance, just as Horn clauses might be chained together for this purpose. In general, these domain theory networks may be provided to the learner by some external source, or they may be the result of previous learning by the same system. EBNN makes use of these domain theory networks to learn the newtarget function. It does not alter the domain theory networks during this process.


Explanation of training example in terms of domain theory:
Stable
Graspable
Liftable
Cup
OpenVessel
BottomIsFlat = T
ConcavityPointsUp = T
Expensive = T
Fragile = T
HandleOnTop = F
HandleOnSide = T
HasConcavity = T
HasHandle = T
Light = T
MadeOfCeramic = T
MadeOfPaper = F
MadeOfStyrofoam = F
0.2
Cup = T
0.8
Training derivatives
Target network:
BottomIsFlat
ConcavityPointsUp
Expensive
Fragile
HandleOnTop
HandleOnSide
HasConcavity
HasHandle
Light
MadeOfCeramic
MadeOfPaper
MadeOfStyrofoam
Cup target
Cup

The goal of EBNN is to learn a new neural network to describe the target function. We will refer to this new network as the target network. In the example of Figure, the target network Cup,,,,,, shown at the bottom of the figure takes as input the description of an arbitrary instance and outputs a value indicating whether the object is a Cup.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

USING PRIOR KNOWLEDGE TO AUGMENT SEARCH OPERATORS

In this section we consider a third way of using prior knowledge to alter the hypothesis space search: using it to alter the set of operators that define legal steps in the search through the hypothesis space. This approach is followed by systems such as FOCL

The FOCL Algorithm

- FOCL is an extension of the purely inductive FOIL system.It also employees sequential covering algorithm (generic to specific search)
- Both FOIL and FOCL learn a set of first-order Horn clauses to cover the observed training examples
- Difference is FOCL considers Domain Theory.

The solid edges in the search tree of Figure 6 show the general-to-specific search steps considered in a typical search by FOIL. The dashed edge in the search tree of Figure 6 denotes an additional candidate specialization that is considered by FOCL and based on the domain theory.

To describe operation FOCL operation, we must know about operational and non operational literals .operational literals are the 12 attributes describing the training sample where asnon operational are intermediate feature that occurs in domain theory.

For example in fig 6 ,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).

Domain theory:

$$Cup \leftarrow Stable, Liftable, OpenVessel$$
$$Stable \leftarrow BottomIsFlat$$
$$Liftable \leftarrow Graspable, Light$$
$$Graspable \leftarrow HasHandle$$
$$OpenVessel \leftarrow HasConcavity, ConcavityPointsUp$$

Training examples:

	Cups				Non-Cups					
BottomIsFlat	✓	✓	✓	✓	✓	✓	✓			✓
ConcavityPointsUp	✓	✓	✓	✓	✓		✓	✓		
Expensive	✓		✓				✓		✓	
Fragile	✓	✓			✓	✓		✓		✓
HandleOnTop					✓		✓			
HandleOnSide	✓			✓					✓	
HasConcavity	✓	✓	✓	✓	✓		✓	✓	✓	✓
HasHandle	✓			✓	✓		✓		✓	
Light	✓	✓	✓	✓	✓	✓	✓		✓	
MadeOfCeramic	✓				✓		✓	✓		
MadeOfPaper				✓					✓	
MadeOfStyrofoam		✓	✓			✓				✓

Fig 5. Cup target concept (Training examples and domain theory)

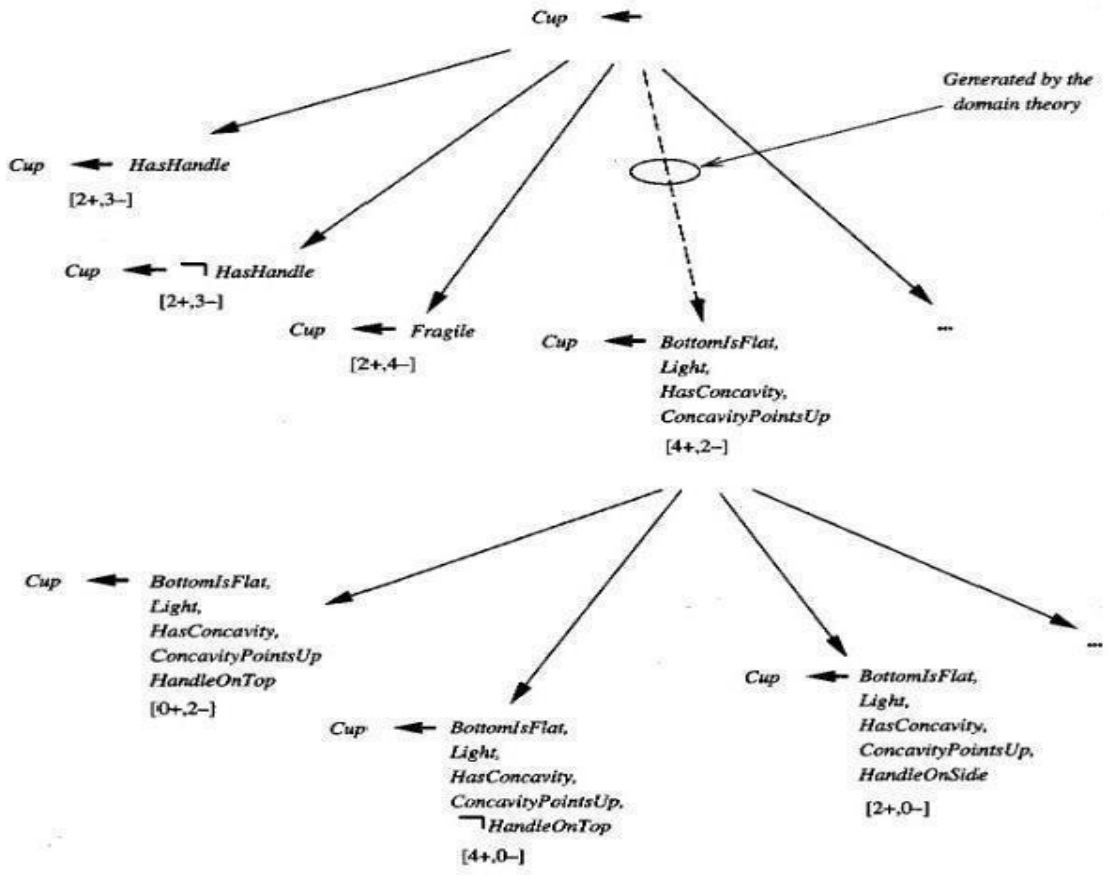


Fig 6. Hypothesis space search in foil

FOCL expands its current hypothesis h using the following two operators: ,

1. For each operational literal that is not part of h, create a specialization of h by adding this single literal to the preconditions. This is also the method used by FOIL to generate candidate successors. he solid arrows in Figure 6 denote this type of specialization.

2. Create an operational, logically sufficient condition for the target concept according to the domain theory. Add this set of literals to the current preconditions of h. Finally, prune the preconditions of h by removing any literals that are unnecessary according to the training data. The dashed arrow in Figure 6 denotes this type of specialization.

- FOCL first selects one domain theory clause whose post condidtion (head) matches the target concept. If there are more such clauses then it selects whose preconditions have highest information.
- For example in the above figure **Cup ←Stable, Liftable, Openvessel**
- Now each non operational literal is replaced with its sufficient i.e. instead of Stable we replace BottomIsFlat similarly we do for all… this process is unfolding
- Then it looks like **BottomIsFlat , HasHandle, Light, HasConcavity , ConcavityPointsUp**
- As a final step in generating the candidate specialization, this sufficient condition is pruned. For each literal in the expression, the literal is removed unless its removal reduces classification accuracy over the training examples. Pruning (removing) the literal **HasHandle**results in improved performance.
    - **BottomZsFlat , Light, HasConcavity , ConcavityPointsUp**

  this hypothesis is the result of the search step shown by the dashed arrow in Figure
- Once candidate specializations of the current hypothesis have been gener- ated, using both of the two operations above, the candidate with highest information gain is selected.

FOCL learns Horn clauses of the form **c ←0i ^ 0b ^ 0f**

where c is the target concept, **0i** is an initial conjunction of operational literals added one at a time by the first syntactic operator, **0b** is a conjunction of operational literals added in a single step based on the domain theory, and **0f** is a final conjunction of operational literals added one ata time by the first syntactic operator.

REINFORCEMENT LEARNING

Each time the agent performs an action in its environment, a trainer may provide a reward or penalty to indicate the desirability of the resulting state. For example, when training an agent to play a game the trainer might provide a positive reward when the game is won, negative reward when it is lost, and zero reward in all other states. The task of the agent is to learn from this

Learning Techniques

Combing Inductive and Analytical Learning:

Motivation:

- two paradigms for machine learning: inductive learning and analytical learning.
- 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.

  Eg: PROLOG-EBG, seek general hypotheses that fit prior knowledge while covering theobserved data.
- 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.

  Eg : decision tree induction and neural network BACKPROPAGATION, seek generalhypotheses that fit the observed training data.
- Combining them offers the possibility of more powerful learning methods.

Differnces between Inductive Learning and Analytical Learning

Inductive Learning	**Analytical Learning**
These methods seek general hypotheses that fit the observed training data.	These methods seek general hypotheses that fit prior knowledge while covering the observed data.
These offer the advantage that they require no explicit prior knowledge and learn regularities based solely on the training data	These offer the advantage of generalizing more accurately from less data by using prior knowledge to guide learning.
The output hypothesis follows from statistical arguments that the training sample is	The output hypothesis follows deductively from the domain theory and training

sufficiently large that it is probably representative of the underlying distribution of example	examples.
The disadvantage is 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	The disadvantage is they can be misled when given incorrect or insufficient prior knowledge.
These provide statistically justified hypotheses	These provide logically justified hypotheses.
Inductive methods are Decision tree ,Backpropagation	Analytical methods are PROLOG-EBG

- The two approaches work well for different types of problems. By combining them we can hope to devise a more general learning approach that covers a more broad range of learning tasks. Fig1,a spectrum of learning problems that varies by the availability of prior knowledge and training data. At one extreme, a large volume of training data is available, but no prior knowledge. At the other extreme, strong prior knowledge is available, but little training data. Most practical learning problemslie somewhere between these two extremes of the spectrum.

Fig 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. Mostpractical problems lie somewhere between these two extremes

Some specific properties we would like from such a learning method include:

- Given no domain theory, it should learn at least as effectively as purely inductive methods.
- Given a perfect domain theory, it should learn at least as effectively as purely analytical methods.
- Given an imperfect domain theory and imperfect training data, it should combine the two to outperform either purely inductive or purely analytical methods.
- It should accommodate an unknown level of error in the training data.
- It should accommodate an unknown level of error in the domain theory.

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

Which hypothesis to consider?

→ One which fits training data well

→ One which fits domain theory well

errorD(h) is defined to be the proportion of examples from D that are misclassified by h. Let usdefine the error errorB(h) of h with respect to a domain theory B to be the probability that h will
disagree with B on the classification of a randomly drawn instance. We can attempt to characterize the desired output hypothesis in terms of these errors.

We require hypothesis that could minimize some combined measures of hypothesis such as

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

At first instance it satisfies, it is not clear what values to assign to k_D and k_B to specify the relative importance of fitting the data versus fitting the theory.

If we have poor theory and great deal of data the error w.r.t D weight more heavily and if we have strong theory and noisy data the error w.r.t B weight more heavily.so the learner doesn't know about training data and domain theory to unclear these components.

So to weight these we use Bayes theorem. Bayes theorem describes how to compute the posterior probability P(h/D) of hypothesis h given observed training data D.Bayes theorem computes this posterior probability based on the observed data D, together with prior knowledge in the form of P(h), P(D), and P(D/h).we can think of P(h), P(D), and P(D/h) as a form of background knowledge or domain theory.Here we should choose hypothesis whose posterior probability is high. If P(h), P(D), and P(D/h) these are not perfectly known then Bayes theorem alone does not prescribe how to combine them with the observed data. Then, we have to assume prior probabilistic values for P(h), P(D), and P(D/h).

Hypothesis space search:

We can characterize most learning methods as search algorithms by describing the hypothesis space H they search, the initial hypothesis ho at which they begin their search, the set of search operators 0 that define individual search steps, and the goal criterion G that specifies the search objective.

three different methods are:

- **Use prior knowledge to derive an initial hypothesis from which to begin the search**. In this approach the domain theory B is used to construct an initial hypothesis ho that is consistent with B. A standard inductive method is then applied, starting with the initial hypothesis ho.
- **Use prior knowledge to alter the objective of the hypothesis space search.** In this approach, the goal criterion G is modified to require that the out- put hypothesis fits the domain theory as well as the training examples.
- **Use prior knowledge to alter the available search steps.** In this approach, the set of search operators 0 is altered by the domain theory.

USING PRIOR KNOWLEDGE TO INITIALIZE THE HYPOTHESIS

One approach to using prior knowledge is to initialize the hypothesis to perfectly fit the domain theory, then inductively refine this initial hypothesis as needed to fit the training data. This approach is used by the **KBANN** (Knowledge-Based Artificial Neural

Network) algorithm to learn artificial neural networks.

In KBANN, initial network is first constructed for every instance, the classification assigned by the network is identical to that assigned by the domain theory. Backpropagation algorithm is employed to adjust the weights of initial network as needed to fit training examples.

If the initial hypothesis is found to imperfectly classify the training examples, then it will be refined inductively to improve its fit to the training examples (Backpropagation algorithm). If thedomain theory is correct, the initial hypothesis will correctly classify all the training examples.

The intuition behind KBANN is that even if the domain theory is only approximately correct, initializing the network to fit this domain theory will give a better starting approximation to the target function than initializing the network to random initial weights.

The KBANN Algorithm

It first initializes the hypothesis approach to using domain theories.It assumes a domain theory represented by a set of propositional, nonrecursive Horn clauses.

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 torefine this initial network to fit the training examples

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*.

EXAMPLE:

Here each instance describes a physical object in terms of the material from which it is made,whether it is light, etc. The task is to learn the target concept Cup defined over such physical
objects. The domain theory defines a Cup as an object that is Stable, Liftable, and an OpenVessel. The domain theory also defines each of these three attributes in terms of moreprimitive attributes and all those attributes describe the instances.

Table 1. describes a set of training examples and a do- main theory for the Cup target concept

Domain theory:

$$Cup \leftarrow Stable, Liftable, OpenVessel$$
$$Stable \leftarrow BottomIsFlat$$
$$Liftable \leftarrow Graspable, Light$$
$$Graspable \leftarrow HasHandle$$
$$OpenVessel \leftarrow HasConcavity, ConcavityPointsUp$$

Training examples:

	Cups				*Non-Cups*					
BottomIsFlat	√	√	√	√	√	√	√			√
ConcavityPointsUp	√	√	√	√	√		√	√		
Expensive	√		√				√		√	
Fragile	√	√			√	√		√		√
HandleOnTop					√		√			
HandleOnSide	√			√					√	
HasConcavity	√	√	√	√	√		√	√	√	√
HasHandle	√			√	√		√		√	
Light	√	√	√	√	√	√	√		√	
MadeOfCeramic	√				√		√	√		
MadeOfPaper				√					√	
MadeOfStyrofoam		√	√			√				√

Table 1. The Cup Learning Task

Here the domain theory is inconsistent because the domain theory fails to classify two and three training examples. KBANN uses the domain theory and training examples together to learn the target concept more accurately than it could from either alone.

1. In First stage, Initial network is constructed consistent with domain theory
2. KBANN follows the convention that a sigmoid output value greater than 0.5 is

interpreted as True and a value below 0.5 as False.

3. Each unit is therefore constructed so that its output will be greater than 0.5 just in those cases where the corresponding Horn clause applies.
4. for each input corresponding to a non-negated antecedent,the weight is set to some positive constant W. For each input corresponding to a negated antecedent, the weight is set to - W.
5. The threshold weight of the unit, wo is then set to -(n- .5) W, where n is the number of non-negated antecedents.
   When i/p values are 1 or 0 then weightedsum+ w0 will be +ve , if all antecedents are satisfied.
6. Each sigmoid unit input is connected to the appropriate network input or to the output of another sigmoid unit, to mirror the graph of dependencies among the corresponding attributes in the domain theory. As a final step many additional inputs are added to eachthreshold unit, with their weights set approximately to zero.

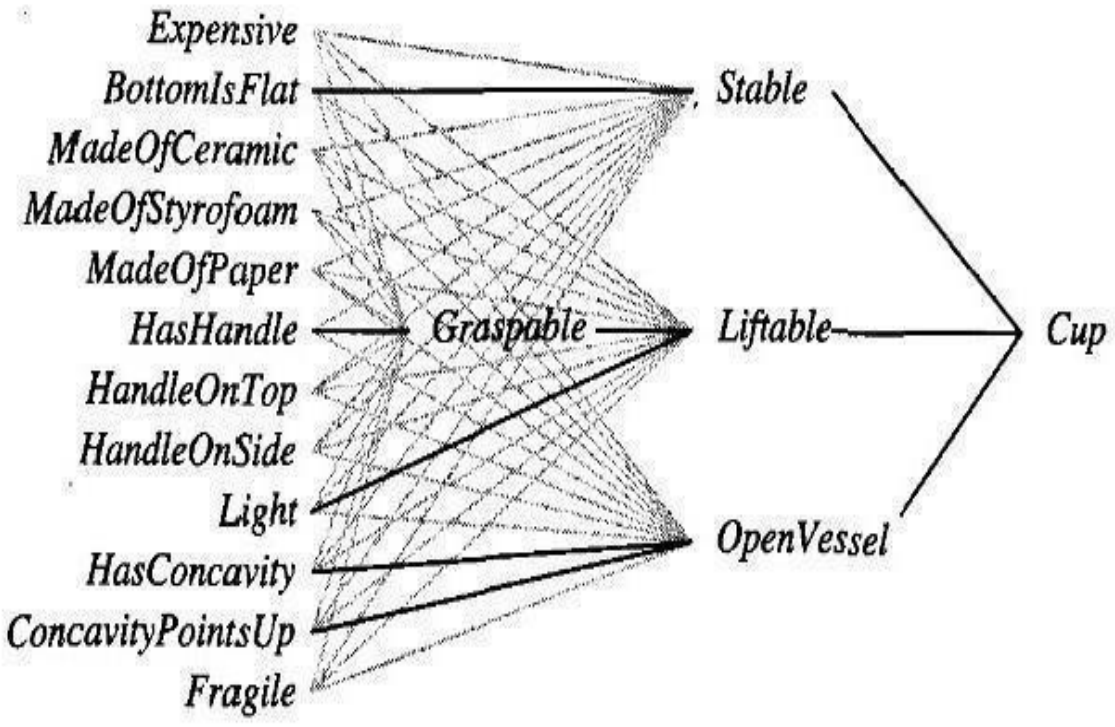


Fig 2. A Neural network equivalent to domain theory

The solid lines in the network of Figure 2 indicate unit inputs with weights of W, whereas thelightly shaded lines indicate connections with initial weights near zero.

7. The second stage of KBANN uses the training examples and BACKPROPAGATION thealgorithm to refine the initial network weights, if the intial

network is not consistent with theory. If consistent no need of back propagation.

8. But our example is not consistent so we perform back propagation

Figure 3, with dark solid lines indicating the largest positive weights, dashed lines indicating thelargest negative weights, and light linesindicating negligible weights.

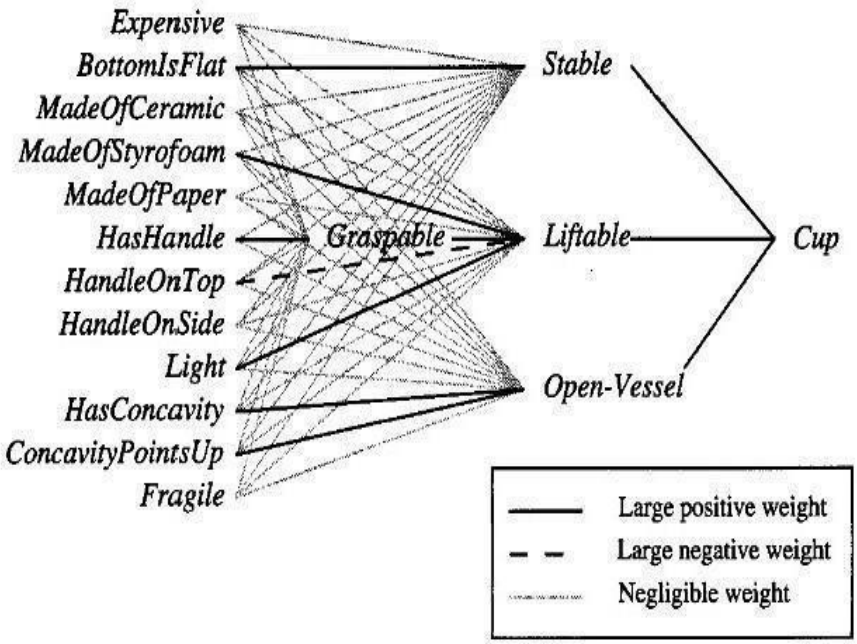


FIGURE 12.3
Result of inductively refining the initial network. KBANN uses the training examples to modify the network weights derived from the domain theory. Notice the new dependency of *Liftable* on *MadeOfStyrofoam* and *HandleOnTop*.

Fig 3. Result of inductively refined neural network.

REMARKS:

- The chief benefit of KBANN over purely inductive BACKPROPAGATION is that it typically generalizes more accurately than BACKPROPAGATION when given an approximately correct domain theory, especially when training data is scarce.
- Limitations of KBANN include the fact that it can accommodate only propositional

domain theories; that is, collections of variable-free Horn clauses. It is also possible for

KBANN to be misled when given highly inaccurate domain theories, so that its

generalization accuracy can deteriorate below the level of BACKPROPAGATION