UNIT - V

Analytical Learning.

→ Inductive Learning: Learning Algorithms like neural N/wS, decision trees, Inductive logic programming etc. all require a good number of examples to be able to do good predictions.

dataset ⟶ Learning Algorithm ⟶ Hypothesis.

→ Dataset must be significantly large.

Ex:

→ Analytical learning:- we don't need a large dataset if besides taking examples as input, the learning algorithm can take prior knowledge.

dataset ⟶ learning Algorithm ⟶ hypothesis.

Prior knowledge ↗ learning Algorithm

→ Dataset does not need to be large.

Ex:- online chessgame.

→ Explanation - based learning:

9966772557
Kalyan reddy

Prior knowledge is used to reduce the size of the hypothesis space.

Hypothesis space. HS. → Prior knowledge. → Reduced HS.

→ It analyzes each example to infer which features are relevant and which ones are irrelevant.

Ex: learning to play chess.

→ Suppose we want to learn a concept like "what is a board position in which black will lose the queen in 'x' moves?".

→ chess is a Complex game. Each piece can occupy many positions. We would need many examples to learn this concept.

→ But humans Can learn these type of Concepts with very few examples. why?

→ Humans can analyze an example and use prior knowledge related to legal moves.

→ From there it can generalize with only few examples.

→ what is the prior knowledge involved in playing chess?

It is knowledge about the rules of chess:

→ Legal moves for the pieces

→ players alternate moves in games

→ To win you must Capture the opponent's King.

→ Inductive and Analytical learning:

Inductive learning	Analytical learning
→ i/p : HS, D	→ i/p : HS, D, B.
→ o/p : hypothesis h	→ o/p : hypothesis h.
→ h is Consistent with D	→ h is Consistent with D & B.

HS : Hypothesis space.

D : Training Set

B : Background knowledge (domain theory)

→ Analytical learning problem Example

Ex: Given : 1) Dataset where each instance is a pair of objects represented by the following predicates: Color, Volume, owner, Material, Density, On. 2)

→ Safe to Stack (x, y) is the target Concept.

→ Can we place x element on y ⓧ ⓨ or not.

→ **Hypothesis Space**: Set of Horn clause rules.
The head of each rule has the predicate Safetostack
The body of each rule is based on the instances and the predicates Less than, Equal, Greater and plus, minus and times.

Example: The following horn clause is in Hypothesis Space

Safetostack(x, y) ← Volume(x, vx) ^ volume(y, vy) ^ lessthan(vx, vy).

→ **Target Concept**: Safe to stack(x, y)

→ **Training examples**: A typical +ve example Safetostack(obj1, obj2) is given 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) ← ~fragile(y) [Not easily breakable]
Safe to stack(x, y) ← Lighter(x, y)
Lighter(x, y) ← weight(x, wx) ^ weight(y, wy) ^ Lessthan(wx, wy)

Fragile(x) ← Material(x, Glass)

Determine: Hypothesis 'H' consistent with both the training examples and the domain theory. V ③

Note: → The domain theory refers to predicates not contained in the examples.

→ ~~The domain theory refers to predicates not contained in the examples.~~

→ The domain theory is sufficient to prove the example is true.

→ Perfect Domain theories:

→ A domain theory is correct if each statement is true.

→ A domain theory is complete if it covers every positive example of the instance space (w.r.t a target concept and instance space).

→ A perfect domain theory is correct and complete.

→ Explanation Based Learning Algorithm:

→ we consider an algorithm that has the following properties:

→ It is a sequential covering algorithm considering the data incrementally.

→ for each positive example not covered by the current rules it forms a new rule by:

- Explain how training example satisfies target Concept, in terms of domain theory.
- Analyze the explanation to find the most general Conditions under which this explanation holds.
- Refine the current hypothesis by adding a new Horn clause rule to cover the example.

Ex:

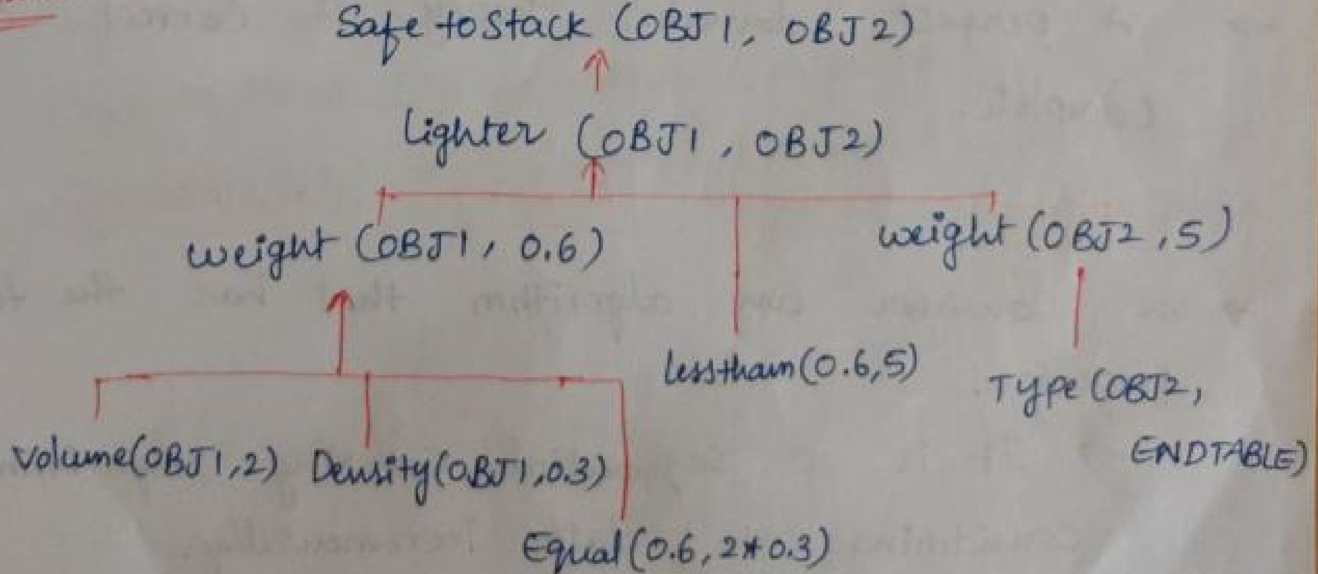
* PROLOG - EBG (Program logic - Explanation Based)

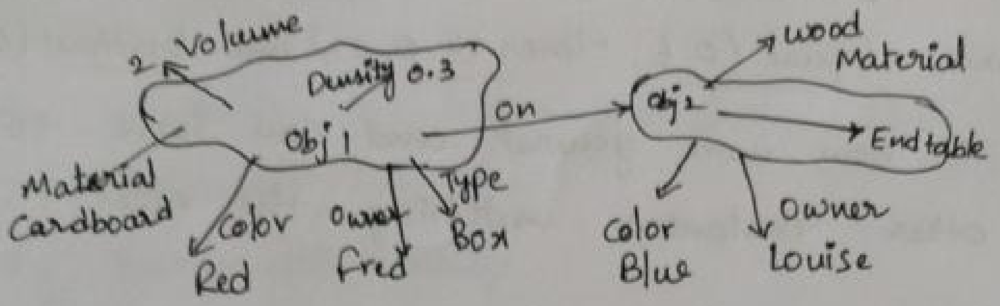
→ Learning a single Horn - Clause rule.

→ Remove the positive training examples covered by this rule until no further positive examples remain uncovered.

→ The explanation is a proof that the example belongs to the target (if the theory is perfect)

Explanation:





Explanation: There might be more than one explanation to the example. In that case one or all explanations may be used.

→ As explanation is obtained using a backward chaining search as is done by PROLOG. PROLOG-EBG stops when it finds the first proof.

Analyze: Many features appear in an example. Of them, how many are truly relavant?

→ we consider as relevant those features those features that show in the explanation.

Example:

Relevant feature : Density.

Irrelevant feature : Owner.

→ Taking the leafnodes of explanation and substituting variables x & y for obj1 & obj2:

Safetostack $(x, y) \leftarrow$ volume $(x, 2) \wedge$ Density $(x, 0.3) \wedge$ Type $(y,$ Endtable$)$

→ Remove features that are independent of x and y such as Equal (0.6, times (2, 0.3)) and lessthan (0.6,5)

→ The rule is now more general and can serve to explain other instances matching the rule.

Refine:

→ The current hypothesis is the set of Horn clauses that have been learned to this point.

→ By using sequential covering more rules are added, thus refining our hypothesis.

→ A new instance is negative if it is not covered by any rule.

Discovering New features: The PROLOG-EBG system described can formulate new features that are not in the training examples.

Example: Volume * Density > 5

(derived from the domain theory)

→ Similar to features represented by hidden neurons in ANN.

→ Difference:

1) ANN is a statistical method that requires many training examples to obtain the hidden neuron features

2) PROLOG-EBG use an analytical process to obtain new features from analysis of single training examples

Inductive Bias in Explanation - Based Learning:

→ what is the inductive bias of Explanation based learning?

The hypothesis h follows deductively from D & B.

D : data base

B : Background knowledge.

Bias : Prefer small sets of maximally general Horn clauses

logical statement ~~Concept~~ which Supports our target Concept.

→ Search Control knowledge

Problem : Learning to Speed up Search programs (Speed up learning)

Examples include : Playing chess. Scheduling and optimization problems

Problem formulation :

S : Set of possible search states

O : set of legal operators (transform one State to another State)

G : predicate over S indicating the goal states.

→ Prodigy

Prodigy is a planning System.

Input : State space S and operators O.

Output : A Sequence of operators that lead from the initial state to the final state.

Prodigy uses a means - end planner : we decompose goals into subgoals :

Problems with EBL :

→ The number of Control rules that must be learned is very large.

→ If the control rules are many, much time will be spent looking for the best rule.

Utility analysis is used to determine what rules to keep and what rules to forget.

→ Another problem with EBL is that it is sometimes difficult to create an explanation for the target concept.

→ For example, in chess learning a concept like :
" States for which operator A leads to a solution"
The Search here grows exponentially.

Summary! 1) Different from inductive learning, analytical learning looks for a hypothesis that fit the background knowledge and covers the training examples.

2) Explanation Based learning is one kind of analytical learning that divides into three steps :

a) Explain the target value for the current example.

b) Analyze the explanation (generalize)

c) Refine the hypothesis

3) PROLOG - EBG Constructs intermediate features after analyzing examples.

4) Explanation based learning can be used to find Search control rules.

5) Depend on a perfect domain theory.

Content

→ Motivation

→ Inductive - Analytical Approaches to learning

* the learning problem
* The Hypothesis Space search

→ Using Prior Knowledge to Initialise the Hypothesis

* KBANN

→ Using prior Knowledge to Alter the Search objective

* The Tangentprop Algorithm
* EBNN

→ Using prior Knowledge to Augment Search Operators

* ~~*~~ FOCL

Combining Inductive and Analytical learning:-

→ Motivation

→ pure Inductive methods formulate general hypothesis by recognising empherical regularities in the training examples..

* Advantage :
  1) Don't require explicit prior knowledge
  2) Learn regularities based solely on the training data.

* Disadvantage :
  1) fail when insufficient training data is given
  2) Can be mislead by the implicit inductive bias they must cope within the order to generalise beyond the observed data.

→ Pure Analytical Methods use prior knowledge to derive general hypothesis deductively.

* Advantage : Accurately generalise from a few training examples by using prior knowledge

* Disadvantage : Can be mislead when given incorrect or insufficient prior knowledge

→ Combination :- for the better accuracy on the generalisation when prior knowledge available and the reliance on the observed training data overcomes the shortcoming of the prior knowledge

	Inductive learning	Analytical learning
Goal :	Hypothesis fits the data	Hypothesis fits domain theory , and covers data.
Justification :	Satistical Inference	Logical Inference
Pitfalls :	Scarce data, incorrect bias	Imperfect domain theory
Advantages :	Requires little prior knowledge	Generalizes from scarce data.

* Difference :

→ Analytical : Logical Justification - The o/p hypothesis follows deductively from the domain theory and the training examples.

→ Inductive : Statistical Justification - The o/p hypothesis follows from the assumption that the set of training examples is sufficiently large and that it is representation of the underlying distribution of the examples.

→ The two approaches work well on different types of problems.

Inductive learning ⟷ Analytical learning

plentiful data — perfect prior knowledge

No prior knowledge — Scarce data.

→ The most practical learning problems lie somewhere between these two extremes. V ①

→ Ex:

* Analysing a database of medical records in order to learn "symptoms for which treatment x is more effective than treatment y"

* Analysing a stock market database in order to learn the target concept " Companies whose stock value will double over the next months"

→ Interested in systems that take prior knowledge as an explicit input to the learner

→ ~~Dom.~~ Goal: Domain Independent algorithms that employ explicitly input domain - dependent knowledge

→

* when no domain theory → It should learn at least as effectively as purely inductive methods

* when perfect domain theory → It should learn at least as effectively as pure Analytical methods.

* when an imperfect domain theory and imperfect traing data → 1) It should combine the two to outperform either purely Inductive / Purely Analytical methods.

2) It should accomodate an unknown level of errors in the training data.

3) It should accomodate an unknown level of error in the domain theory.

* Active current research ⇒ we do not yet have algorithms that satisfy all these constraints in a fully general fashion.

→ Inductive - Analytical Approaches to learn the learning problem.

→ Given :

* A set of training examples D, possibly containing errors.
* A domain theory B, possibly containing errors.
* A space of candidate hypothesis H.

→ Determine :

* A hypothesis that best fits the training examples and the domain theory.

→ what exactly does best fit mean?
Minimize some combined measures of the error of the hypothesis over the data and the domain theory.

→ Defining measures for the hypothesis error with respect to the data and with respect to the domain theory.

$$\text{argmin}_{h \in H} \left(k_D \; error_D(h) + k_B \; error_B(h) \right)$$

* $error_D(h)$: The proportion of examples from D that are misclassified by h
* $error_B(h)$: the probability that h will disagree with B on the classification of a randomly drawn instance.
* k_D , k_B : Constants

→ If we have a very poor theory and great deal of reliable data, it will be best to weight $error_D(h)$ more heavily.

→ Given a strong theory and small sample of very noisy data, the best results would be obtained by weighting $error_B(h)$ more heavily.

→ But the learner does not know the quality of D & B in advance ⇒ it is unclear how these two error components should be weighted.

Hypothesis Space Search

→ How can the domain theory and training data best be combined to constrain the search for an acceptable hypothesis?

→ Understand the range of possible approaches as searching through the space of alternate hypothesis

→ Notation :

H	hypothesis space
h_0	initial hypothesis
O	search operators
G	Goal criterian

→ Different methods for using prior knowledge to alter the search performed by inductive methods;

* Use prior knowledge to derive an initial hypothesis from which the search begins. A standard inductive method is applied then; KBANN

* Use prior knowledge to alter the objective of the hypothesis space.
  G is modified to require the output hypothesis fitting the domain theory as well as the training examples ; EBNN

* Use prior knowledge to alter the available search steps, O is altered by the domain theory; FOCL

~~* KBANN (Knowl~~

Using Prior knowledge to Initialize the Hypothesis

* KBANN (Knowledge - Based Artificial Neural Network; Shavlik Towel 1989) :

→ The Initial Network is Constructed for every possible instance, the classification assigned by the network is identical to the one assigned by the domain theory.

→ The BACKPROPAGATION algorithm is then employed to adjust the weights of this initial network as needed to fit the training example.

→ The input and output of KBANN are the following:

* Given :
  1) A set of training examples
  2) A domain theory consisting of non-recursive, propositional horn clauses.

* Determine :
  1) An artificial neural network that fits the training examples, based on the domain theory

→ KBANN (Domain theory, Training examples)

* Analytical step : Creates an initial network equivalent to the domain theory.
  → For each instance attribute, create a network input.

→ For each Horn clause in the Domain theory, Create a network unit :

* Connect the inputs of this unit to the attributes tested by the clause's antecedents
* For each positive antecedent of the clause, assign a weight of $(+w)$ to the Corresponding Sigmoid unit input
* For each Negative 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 $-(n-0.5)w$, where n is the number of Negative antecedents of the clause.

→ 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 :

→ Apply the backpropagation algorithm to adjust the initial network weights to fit the training examples

→ The Domain theory is not perfectly consistent with the training examples.

Ex: The cup learning task. (KBANN example) V ⓪

Domain theory: cup ← stable, liftable, open vessel

Stable ← Bottom is flat

Liftable ← Graspable, light

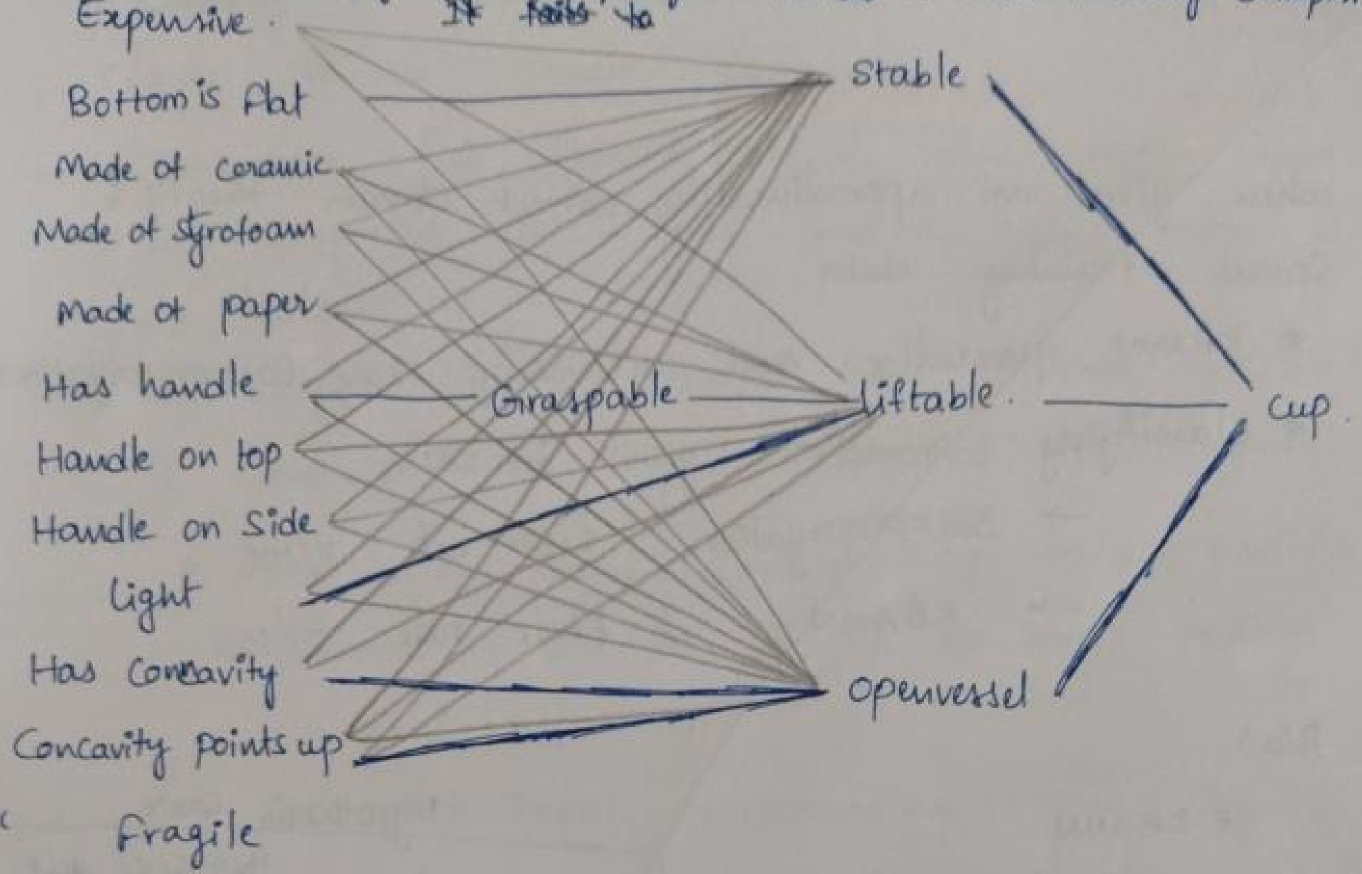
Graspable ← Has handle

open vessel ← Has concavity, Concavity points up.

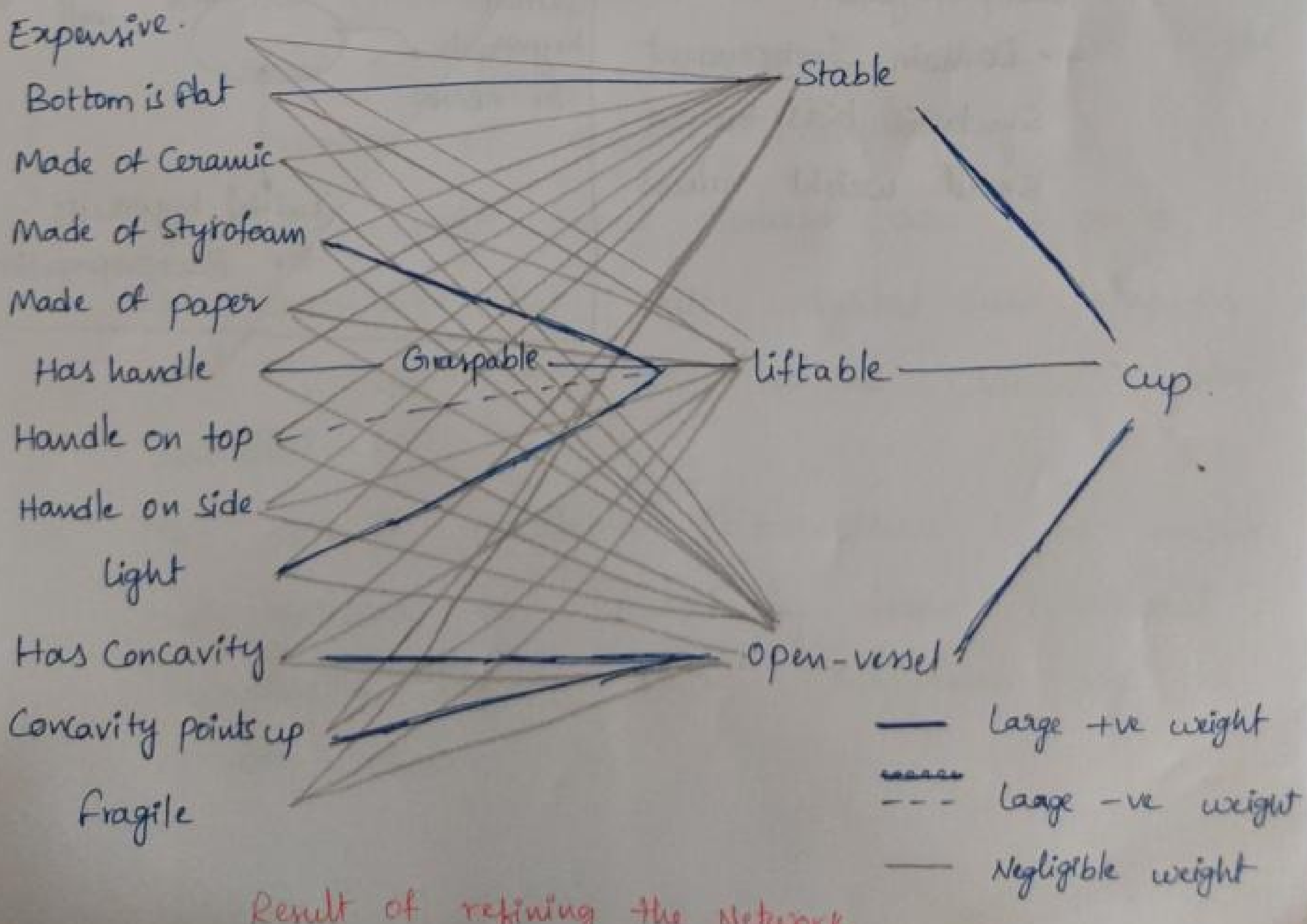
Training examples:

	Cups				Noncups					
Bottom is flat	✓	✓	✓	✓	✓	✓	✓			✓
Concavity points up	✓	✓	✓	✓	✓		✓	✓		
Expensive	✓		✓					✓	✓	
Fragile	✓	✓			✓	✓		✓		✓
Handles on top					✓			✓		
Handle on side	✓			✓					✓	
Has concavity	✓	✓	✓	✓	✓		✓	✓	✓	✓
Has handle	✓			✓	✓			✓	✓	
Light	✓	✓	✓	✓	✓	✓	✓		✓	
Made of ceramic	✓				✓		✓	✓		
Made of paper				✓					✓	
Made of styrofoam		✓	✓			✓				✓

→ The Domain theory is not perfectly Consistent with the training examples: V (ii)
~~It fails to~~



Neural Net Equivalent to Domain theory.



Result of refining the Network.

→ Creating a Sigmoid threshold unit for each Horn clause in the domain theory.

→ Convention : A Sigmoid output value greater than 0.5 is interpreted as 'true' and a value below 0.5 as 'false'

→ the weight of the sigmoid unit is then set so that it computes the logical AND of its inputs :

* For each input corresponding to a positive example , set the weight to some positive constant 'w'.
* for each input corresponding to a Negative example, set the weight to '$-w$'.
* the threshold weight of the unit w_0 is then set to $-(n-0.5)w$. where n is the number of negative examples.

→ when the unit input values are 1 or 0, this assures that their weighted sum plus w_0 will be positive ⟺ all the clause's are satisfied.

→ The role of additional connections is to enable the network to inductively learn additional dependencies beyond those suggested by the domain theory.

Remarks

→ KBANN analytically creates a network equivalent to the given domain theory, then inductively refines this initial hypothesis to better fit the training data.

→ In doing so, it modifies the network weights as needed to overcome inconsistencies between the domain theory and the observed data.

→ Benefit : Generalises more accurately than BACKPROPAGATION when given an approximately correct domain theory especially when training data is scarce.

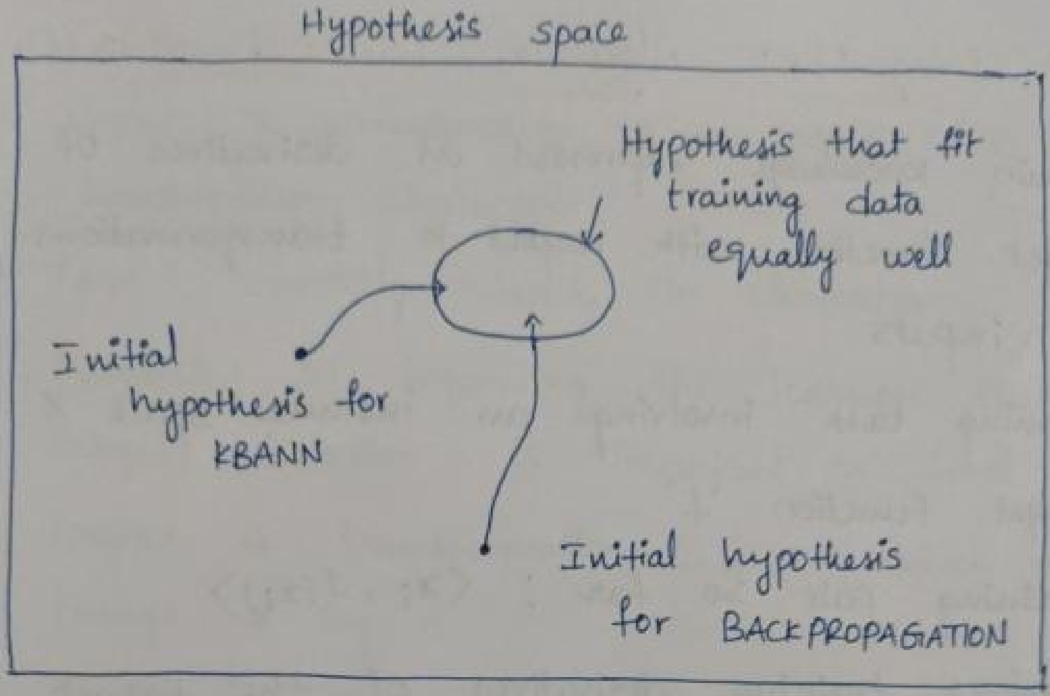
→ Limitation :

* It can accomodate only propositional domain theories

* Mislead when a highly inaccurate domain theory is given, so that its generalisation accuracy can deteriorate below the level of BACKPROPAGATION

→ Application on Practical problems :

* Molecular genetics problem:
Recognise DNA segments called Promoter regions.



→ **Using Prior Knowledge to Alter the Search Objective.**

→ Using prior knowledge to incorporate it into the error criterion minimised by the gradient descent, so that the network must fit a combined function of the training data and the domain theory.

→ Prior knowledge is given in the form of known derivatives of the target function.

→ Certain types of prior knowledge can be expressed quite naturally in this form

→ Example:

* Training neural networks to recognise handwritten characters.
* Derivatives: The identity of the character is independent

of small translations and rotations of the image.

→ The TangentProp Algorithm (Simard 1992)

→ Domain knowledge expressed as derivatives of the target function with respect to transformations of its inputs.

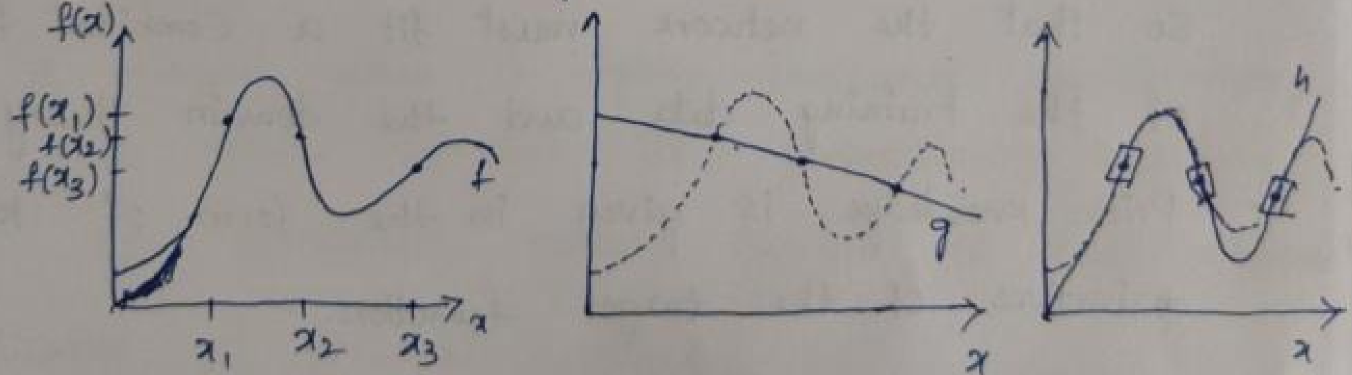
→ Learning task involving an instance space 'x' and target function 'f'

→ Training pair so far : $\langle x_i, f(x_i) \rangle$

→ Various training derivatives of the target function are also provided is described by a single real value

$$\left\langle x_i, f(x_i), \left.\frac{\partial f(x)}{\partial x}\right|_{x_i} \right\rangle$$

→ Example: Learn target function 'f'



→ Tangentprop can accept training derivatives with respect to various transformations

→

→ Example : Learning to recognise handwritten characters

* Input : x corresponding to an image of a single handwritten character
* Task : Correctly classify the character
* Interested in informing the learner that the target function is invariant to small rotations.
* Define a transformation $s(\alpha, x)$ which rotates the image x by α
* Rotational invariance

If $$\frac{\partial f(s(\alpha, x))}{\partial \alpha} = 0$$, then it is invariant to rotation.

→ Question : How are such training derivatives used by Tangentprop to constrain the weights of neural networks?

→ Answer: The training derivatives are incorporated into the error function that is minimised by gradient descent:

$$E = \sum_i \left(f(x_i) - \hat{f}(x_i) \right)^2$$

* x_i → i^{th} training instance
* f → True target function
* $\hat{f}$ → The function represented by the learned neural networks.

→ Here, Additional term is added to the error function to penalise the discrepancies between the training derivatives and the actual derivatives of the learned neural network.

→ Each tranyformation must be of the form $s_j(\alpha, x)$

α → Continuous parameter

s_j → Diffentiable

$s_j(0, x) = x$.

→ The modified error function

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

where

μ → constant provided by the user to determine the relative importance of fitting training values Vs fitting training derivatives.

→ AN ILLUSTRATIVE EXAMPLE

→ Comparing the generalisation accuracy of Tangentprop and purely inductive Backpropagation

→ Task: Label imaging Containing a Single digit between 0 and 9

→ Training: Varying size of the Set

→ Test: 160 example.

→ Prior knowledge given to TangentProp:

The fact that the classification of the digit is invariant of vertical and horizontal translation of the image

Training set size.	Percent errors on test set Tangent prop	Backpropagation
10	34	48
20	17	33
40	7	18
80	4	10
160	0	3
320	0	0

Remarks:

→ Tangentprop uses prior knowledge in the form of derivatives of the desired target function with respect to the transformations of its input.

→ It combines this prior knowledge with the observed training data, by minimising an objective function that measures both the network's error with respect to the training example values and its error w.r.t the desired derivation.

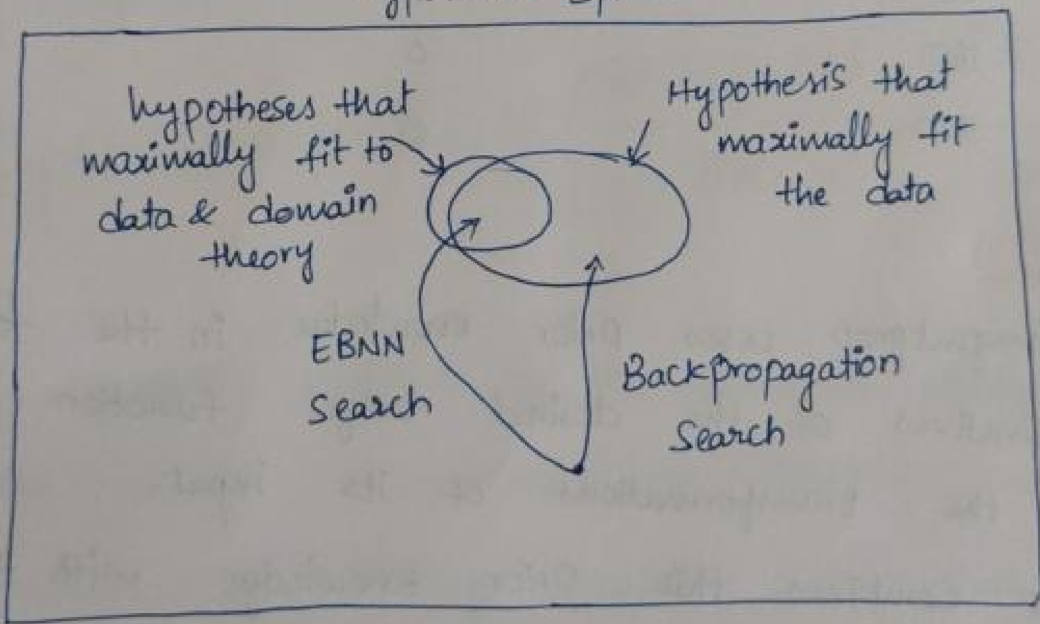
→ The value μ determines the degree to which the network will fit one or the other of these two components in the total error.

→ Disadvantage: Not robust to the errors of the prior knowledge.

* Algorithm will ~~fit~~ attempt to fit incorrect derivatives ⇒ ignoring the prior knowledge would have led to better generalisation ⇒ select μ.

→ EBNN ~~sel~~ automatically selects μ on an example-by-example basis.

Hypothesis space



The EBNN Algorithm (Mitchel and Thurn 1993) V (16)

→ EBNN Computes the training derivatives by itself.

→ These are Calculated by explaining each training example in terms of the given domain theory, then extracting training derivatives from this explanation.

→ EBNN addresses the issue of how to weigh the relative importance of inductive and analytical components of learning.

→ The value is chosen independently for each training example, based on heuristic that considers how accurately the domain theory predicts the training value for this training example.

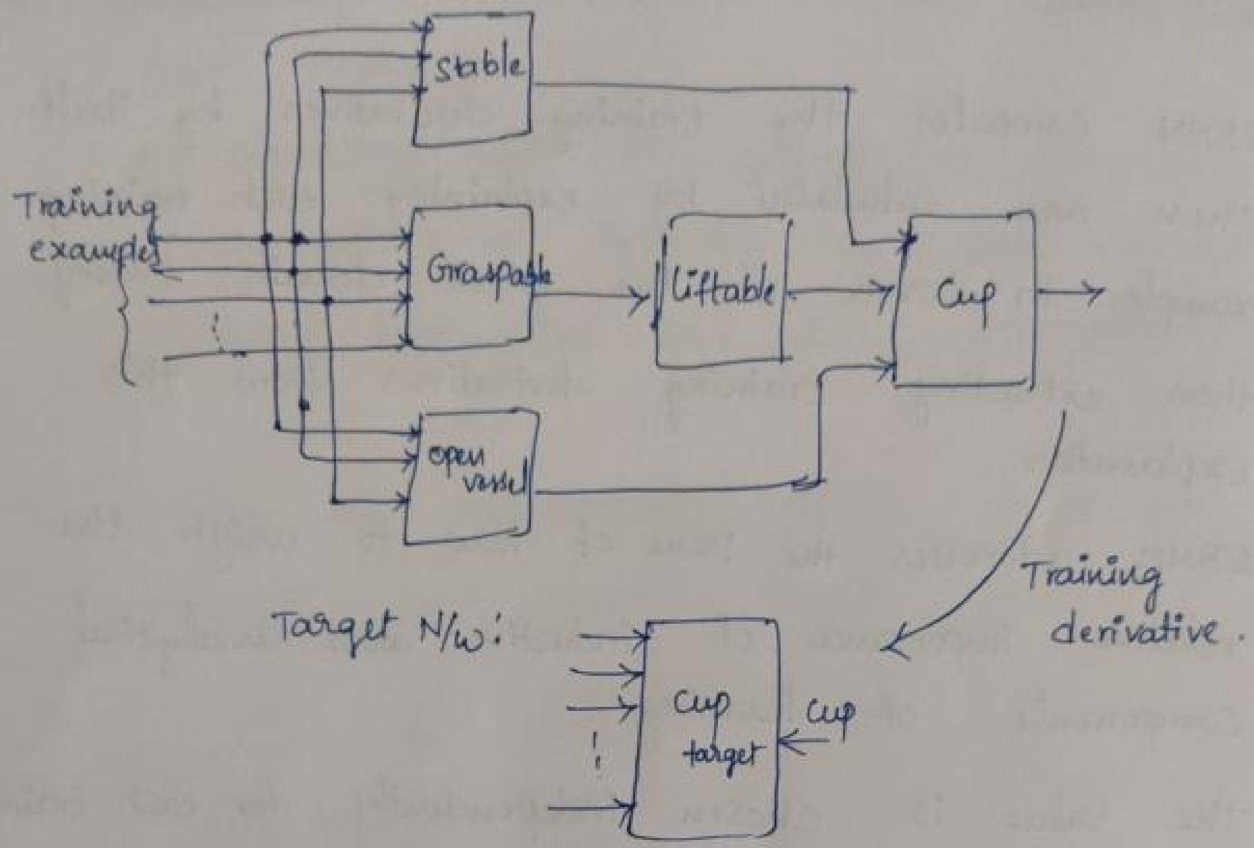
→ Input:

* set of training examples $\langle x_i, f(x_i) \rangle$ with no training derivatives provided.
* Domain theory analogous to that used in explanation - based learning.

→ Output:

* New neural N/ω that approximates the target function f

→ N/ω is trained to fit both training examples and training objectives f



→ Each rectangular block represents a distinct neural N/w in the domain theory

→ The N/w Graspable takes as input the description of an instance and produces as o/p a value indicating whether the object is graspable.

→ Some N/ws take the o/p s of other N/ws as their input.

→ Goal: Learn a new neural N/w to describe the target function (target N/w)

→ EBNN learns a target N/w by involving Tangentprop.

→ EBNN passes the training values $\langle x_i, f(x_i) \rangle$ to tangentprop and provides it with derivatives

Calculated from the domain theory.

→ EBNN Calculates the derivatives w.r.t each feature of the i/p instance.

→ Example : x_i is described by MadeofStyrofoam = 0.2(False) + the domain theory prediction is that cup = 0.8
EBNN Calculates the partial derivative of this prediction w.r.t each instance feature yielding the set of derivatives.

$$\left[\frac{\partial cup}{\partial Bottom\ is\ flat}, \frac{\partial cup}{\partial Concavity\ points\ up}, ---- \frac{\partial cup}{\partial Made\ of\ Styrofoam} \right]_{x=x_i}$$

→ This set of derivations is the gradient of the domain theory prediction function with respect to the i/p sequence

→ General case where the target function has multiple o/p units, the gradient is Computed for each of these o/ps ⇒ the matrix of the gradients called the Jacobian of the target function.

→ Importance of the training derivatives :

* · If the feature Expensive is irrelevant ⇒ $\frac{\partial cup}{\partial Expensive}$ has the value 0.

* Large +/- derivatives Correspond to the assertion that the feature is highly relevant to determine the target value

Algorithm :- → Given D & B create a new fully connected feedforward N/w to represent the target function.

→ N/w is initialised with small random weights.

→ For each $\langle x_i, f(x_i) \rangle$ determine the corresponding training derivatives.

* Use the B to predict the value of the target function for instance x_i : $A(x_i)$

* the weights and the actions of the domain theory network are analysed to extract the derivatives of $A(x_i)$ w.r.t each of the components.

→ Use a minor variant of TangentProp to train the target N/w to fit the error.

$$E = \left[(f(x_i) - \tilde{f}(x_i))^2 + \mu_i \sum_j \left(\frac{\partial A(x)}{\partial x_j} - \frac{\partial \tilde{f}(x)}{\partial x_j} \right)^2_{(x = x_i)} \right]$$

$$\text{where} \quad \mu_i = 1 - \frac{|A(x_i) - f(x_i)|}{c}$$

x_j → j^{th} component of the vector x , $0 \leq \mu_i \leq 1$

The coefficient 'c' is a normalising constant whose value is chosen to assure that for all 'i'

Remarks:

→ EBNN uses a domain theory expressed as set of previously learned neural Networks together with a set of training examples to train its output hypothesis (target N/ω)

→ For each training example EBNN uses its domain theory to explain the example then extracts training derivatives from this explanation.

→ For each attribute of the instance a training derivative is computed that describes how the target function value is influenced by a small change to this value according to the domain theory.

→ Fitting the derivatives constrains the learned N/ω to fit the dependencies given by the domain theory, while fitting the training values constrains it to fit the observed data itself.

→ The weight placed on fitting the derivatives is determined independently for each training example, based on how accurately the domain theory predicts the training value for this example.

→ Prolog-EBG Vs EBNN

→ In Prolog - EBG the explanation is constructed by B which consists of Horn clauses and the target hypothesis is refined by calculating the weakest preimage.

→ EBNN constructs an analogous explanation but it is based on B consisting of a neural N/ω.

→ Difference :

* EBNN accomodates imperfect domain theories
* Prolog - EBG learns a growing set of Horn clauses whereas EBNN learns a fixed size neural N/w
  ⇒ difficulty in learning Horn clauses is that the cost of classifying a new instance grows as the learning proceeds and new clauses are added
* Disadvantage of EBNN : It may be unable to represent sufficiently complex functions where as a growing set of Horn clauses can represent increasingly complex functions.

Using Prior Knowledge to Augment Search operators

→ 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.

→ FOCL (Pazzani, Kibler 1992)

The FOCL Algorithm

→ ~~FOIL~~ FOCL learn sets of first-order Horn clauses to cover the observed training examples.

→ They employ a sequential covering algorithm that learns a single horn clause, removes the +ve example covered by this new Horn clause and then iterates this procedure over the remaining training examples

→ A new clause is created by performing general - to - specific search, begining with the most general one

→ Several candidate specializations of current clause are then generated and the specialization with the greatest information gain relative to the training examples is chosen

→ Difference : The way of how the candidates are specialised.

→ Def : A literal is operational if it is allowed to be used in describing an o/p hypothesis ⟺ nonoperational : occurs only in B.

→ Ex: In cup only 12 attributes are allowed as operational

→ At each point in its general-to-specific search, 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 precondition.
2. Create an operational, logically sufficient condition for the target concept according to B

   Add this set of literals to the current preconditions of h

   Prune the preconditions of h by removing any literal that is unnecessary according to the training data.

In detail:

→ Select one clause from B whose head matches the target concept. If there are several, select the clause whose body has the highest information gain relative to the training examples of the target concept.

→ Each nonoperational literal in these sufficient conditions replaced again using B and substituting clause preconditions for clause post conditions. This process of "unfolding" the B continues until the sufficient conditions have been restated in terms of operational literals.

→ the sufficient condition is pruned.

Ex:

→ cup ← Stable, liftable, open vessel

stable ← Bottom is Flat

liftable ← Graspable, Light

Graspable ← Has handle

openvessel ← Has Concavity, Concavity points up

Cups.

Bottom is Flat	X	X	X	X	X	X
Concavity points up	X	X	X	X	X	
Expensive	X		X			
Fragile	X	X			X	X
Handle on top					X	
Handle on Side	X			X		
Has Concavity	X	X	X	X	X	
Has handle	X			X	X	
light	X	X	X	X	X	X

Cup ←

—

Generated by domain theory

Cup ← Has handle
[2+, 3-]

Cup ← ~ Has handle
[2+, 3-]

Cup ← Fragile
[2+, 4-]

Cup ← Bottom is flat
Light
Has Concavity
Concavity points up
[4+, 2-]

—

Cup ← Bottom is flat
Light
Has Concavity
Concavity points up
Handle on top
[0+, 2-]

Cup ← Bottom is Flat
Light
Has concavity
Connectivity points up
~ Handle on top
[4+, 0-]

Cup ← B. is flat
Light
Has Concavi.
⋮
[2+, 0-]

Remarks:

→ FOCL uses the domain theory to increase the number of candidate specializations ~~considerations~~ considered at each step of the search for a single Horn clause.

→ FOCL uses both a syntactic generation of candidate specializations and domain theory driven generation of candidate specialization at each step of search.

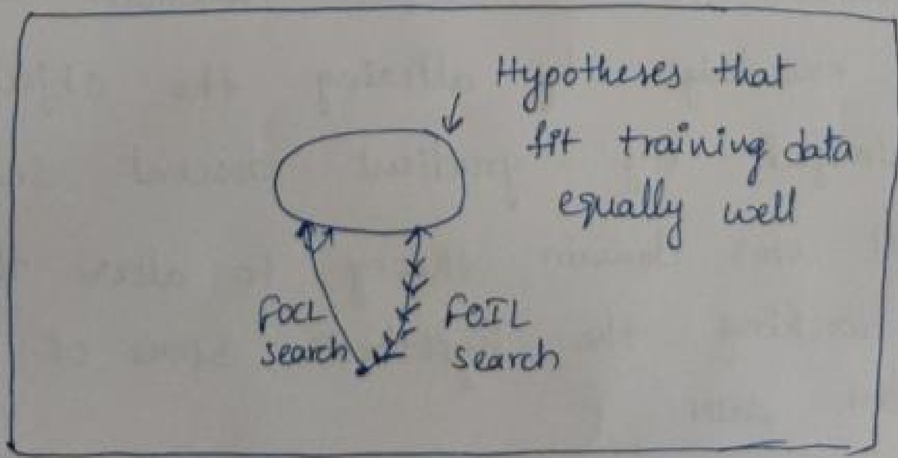
→ Example 1: Legal chessboard position:

* 60 training example (30 legal & 30 illegal endgame board positions)
* FOIL 86% over an independent set of examples
* FOCL has domain theory with 76% accuracy

It produced a hypothesis with generalisation accuracy 94%

→ Example 2: Telephone Network problem.

Hypothesis space



Summary:

→ Approximate prior knowledge, or domain theories are available in many practical learning problems.

⇒ Purely Inductive learning method cannot use it

⇒ Purely analytical learning method can be used only if the domain theory is correct and complete

→ Combination:

The domain theory can affect the hypothesis space search:

* Create the initial hypothesis in search, KBANN
* Alter the objective of the search, EBNN
* Expand the set of search operators that generate revisions to the current hypothesis, TangentProp, FOCL

→ KBANN uses domain theory encoded as proportional rules to analytically construct an ANN then inductively refines it with BACKPROPAGATION

→ Tangentprop uses prior knowledge represented by desired derivatives of the target function. It incorporates this knowledge by altering the objective function minimized by gradient descent search.

→ EBNN uses domain theory to alter the objective in searching the hypothesis space of possible weights for an ANN.

It uses a domain theory consisting of a previously

V (22)

learned neural network to perform a neural N/w analogous to symbolic explanation - based learning

→ FOCL uses domain theory to expand the set of candidates considered at each step in the search. It uses an approximate domain theory represented by first order Horn clauses to learn a set of Horn clauses that approximate the target function. It employs a sequential covering algorithm learning each Horn clause by a general - to - specific search.