Machine Learning

Combining Inductive and Analytical Learning

Artificial Intelligence & Computer Vision Lab
School of Computer Science and Engineering
Seoul National University

Overview

- Motivation
- Inductive-Analytical Approaches to Learning
- KBANN
- TangentProp
- EBNN
- FOCL

Motivation

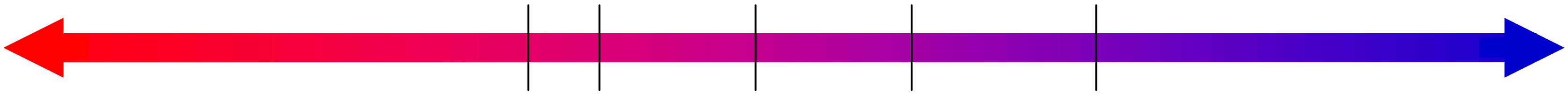
	Inductive Learning	**Analytical Learning**
Goal	Hypothesis fits data	Hypothesis fits domain theory
Justification	Statistical inference	Deductive inference
Advantages	Requires little prior knowledge	Learns from scarce data
Pitfalls	Scarce data, incorrect bias	Imperfect domain theory

The two approaches work well for different types of problem.

How to combine the two into a single algorithm
that captures the best aspects of both ?

Motivation (cont.)

Inductive learning **Analytical learning**



Plentiful data
No prior knowledge

Perfect prior knowledge
Scarce data

Most practical problems lie
somewhere between these two extremes

In analyzing a database of medical records…

In analyzing a stock market database…

Motivation (cont.)

- Desirable properties
  - 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 and 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
    - Tradeoff

$$\boxed{\operatorname{argmin}_{h \in H} \; K_D \, error_D(h) + K_B \, error_B(h)}$$

– $error_D(h)$: Proportion of examples from *D* that are misclassified by *h*

– $error_B(h)$: Probability that *h* will disagree with *B* on the classification of a randomly drawn instance

Inductive-Analytical Approaches to Learning (cont.)

- Learning methods as search algorithms
  - H : Hypothesis space
  - h_0 : Initial hypothesis
  - O : Set of search operators
  - G : Goal criterion

- Use prior knowledge to…
  - Derive an initial hypothesis h_0 from which to begin the search
    - **KBANN**
  - Alter the objective G of the hypothesis space search
    - **TangentProp, EBNN**
  - Alter the available search steps (operator O)
    - **FOCL**

KBANN

- Intuitively
  - Initialize the network using prior knowledge
  - If the domain theory is correct
    - The initial hypothesis will correctly classify all the training examples, no need to revise it.
  - If the initial hypothesis is found to imperfectly classify the training examples
    - Refine inductively to improve its fit to training examples

- c.f.) Purely inductive BACKPROPAGATION
  - Weights are typically initialized to small random values

Even if the domain theory is only approximately correct,
Better than random

"Initialize-the-hypothesis"

KBANN (cont.)

- 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

Create an artificial neural network that perfectly fits the domain theory

Analytical step

BACKPROPAGATION
To refine the initial network to fit the training examples

Inductive step

KBANN (cont.)

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

Domain theory:

$$\begin{array}{c} \text{Cup} \leftarrow \text{Stable, Liftable, OpenVessel} \\ \text{Stable} \leftarrow \text{BottomIsFlat} \\ \text{Liftable} \leftarrow \text{Graspable, Light} \\ \text{Graspable} \leftarrow \text{HasHandle} \\ \text{OpenVessel} \leftarrow \text{HasConcavity, ConcavityPointsUp} \end{array}$$

Training examples:

	Cups				Non-Cups					
BottomIsFlat	√	√	√	√	√	√	√			√
ConcavityPoints Up	√	√	√	√	√		√	√		
Expensive	√		√				√		√	
Fragile	√	√			√	√		√		√
HandleOnTop					√		√			
HandleOnSide	√			√					√	
HasConcavity	√	√	√	√	√		√	√	√	√
HasHandle	√			√	√		√		√	
Light	√	√	√	√	√	√	√		√	
MadeOfCeramic	√				√		√	√		
MadeOfPaper				√					√	
MadeOfStyrofoam		√	√			√				√

KBANN (cont.)

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

  – For each instance attribute, create a network input.

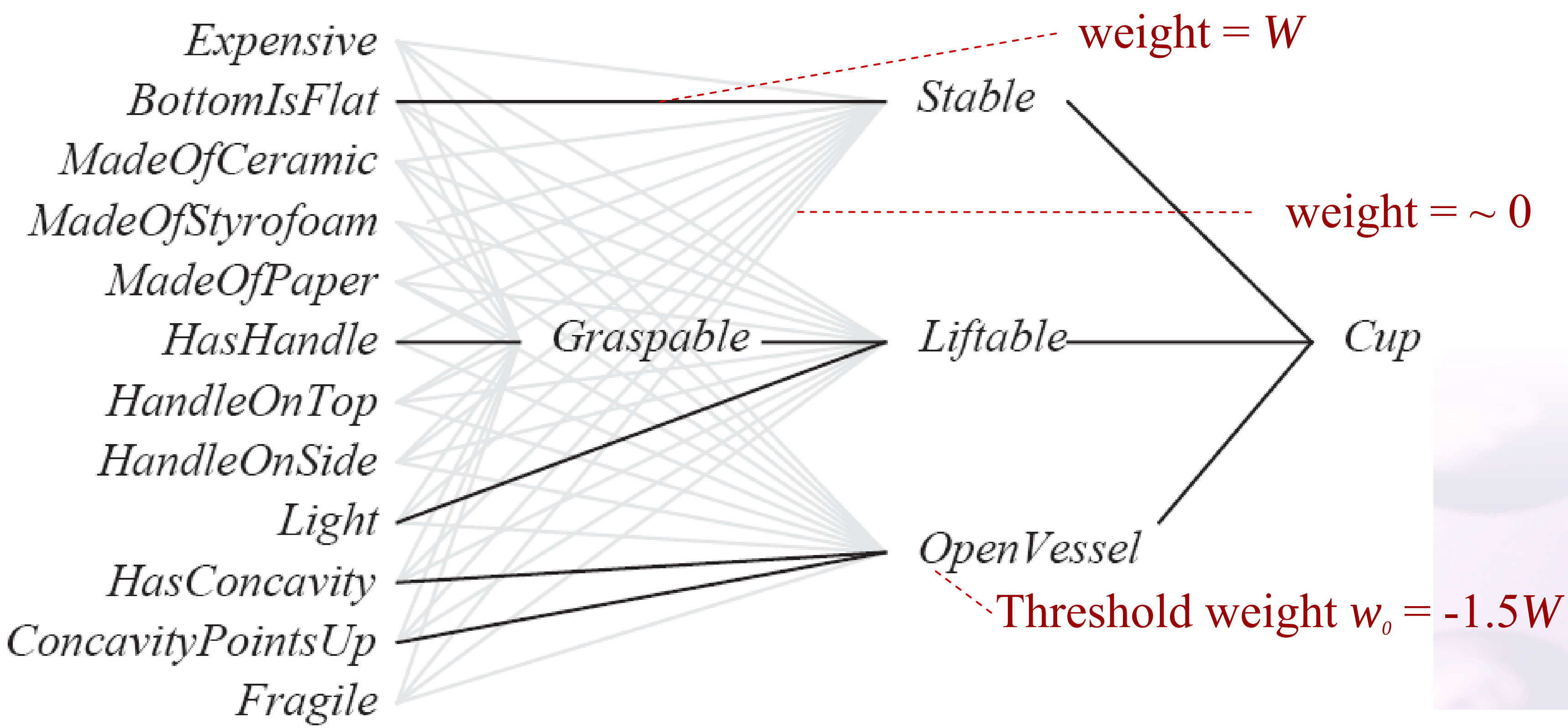
  – For each Horn clause in the *Domain_Theory*, create a network unit as follows:

    - 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 $-(n\text{-}0.5)W$, where n is the number of non-negated 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.

KBANN (cont.)

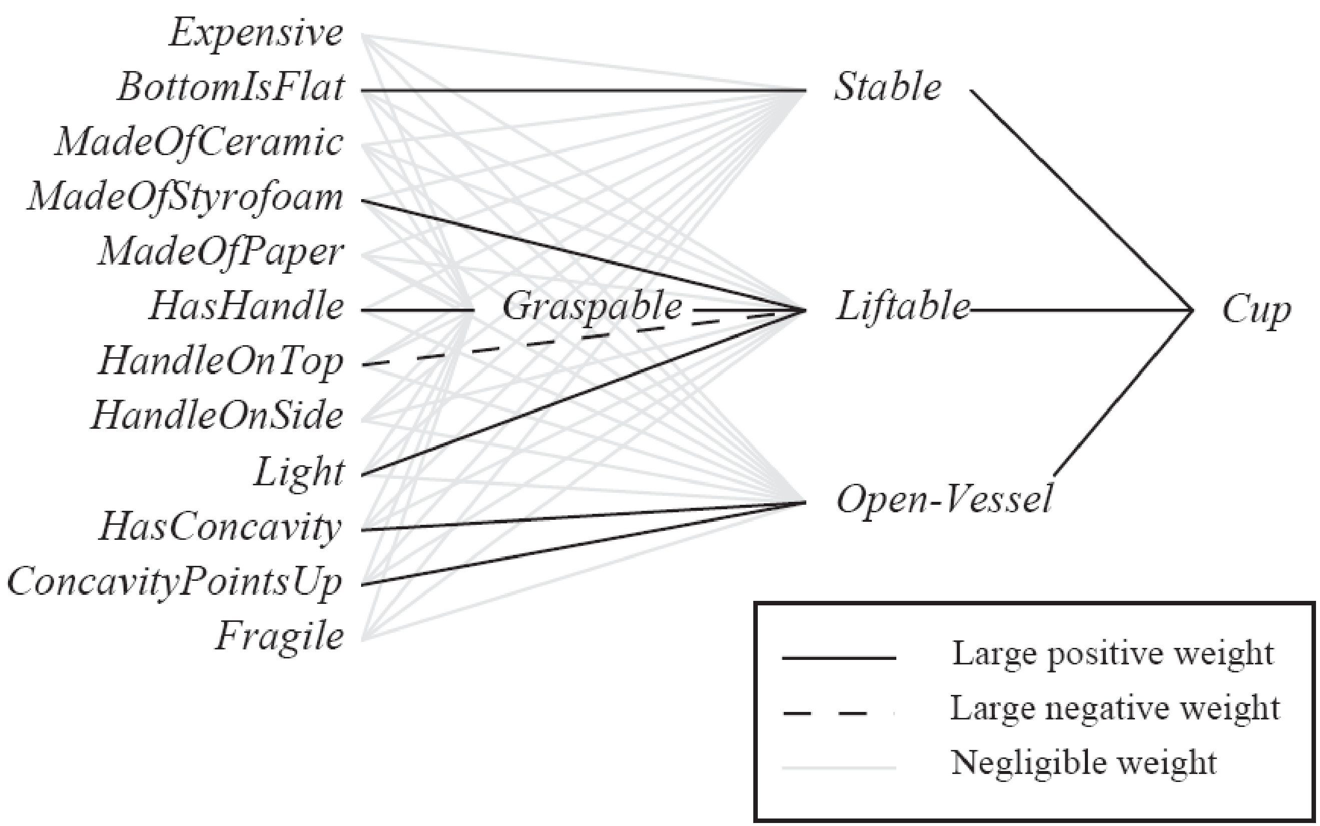
- A neural network equivalent to the domain theory
  - Created in the first stage of the KBANN
  - Sigmoid output value >= 0.5 is true, < 0.5 as false



※Towell and Shavlik(1994), W=4.0

KBANN (cont.)

- Inductive step: Refine the initial network
  - Apply the BACKPROPAGATION algorithm to adjust the initial network weights to fit the *Training Examples*.



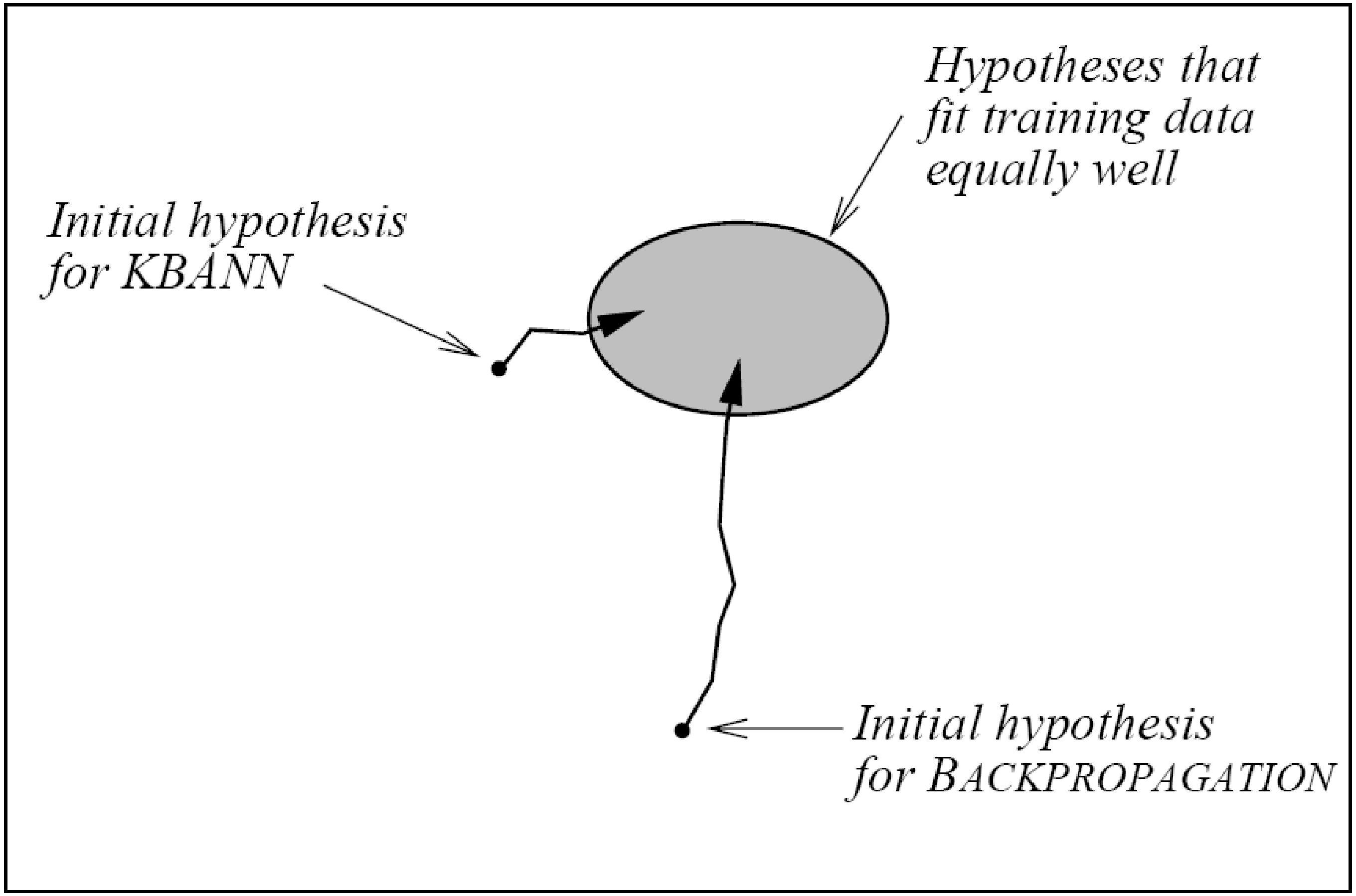
KBANN (cont.)

- Benefits of KBANN
  - Generalizes more accurately than BACKPROPAGATION
    - When given an approximately correct domain theory
    - When training data is scarce
  - Initialize-the-hypothesis
    - Outperform purely inductive systems in several practical problems
      - Molecular genetics problem (1990)
        - » KBANN: Error rate of 4/106
        - » Standard BACKPROPATATION: Error rate of 8/106
        - » Variant of KBANN(1993) by Fu: Error rate of 2/106

- Limitations of KBANN
  - Accommodate only propositional domain theories
    - Collection of variable-free Horn clauses
  - Misled when given highly inaccurate domain theories
    - Worse than BACKPROPAGATION

KBANN (cont.)

- Hypothesis space search in KBANN



TangentProp

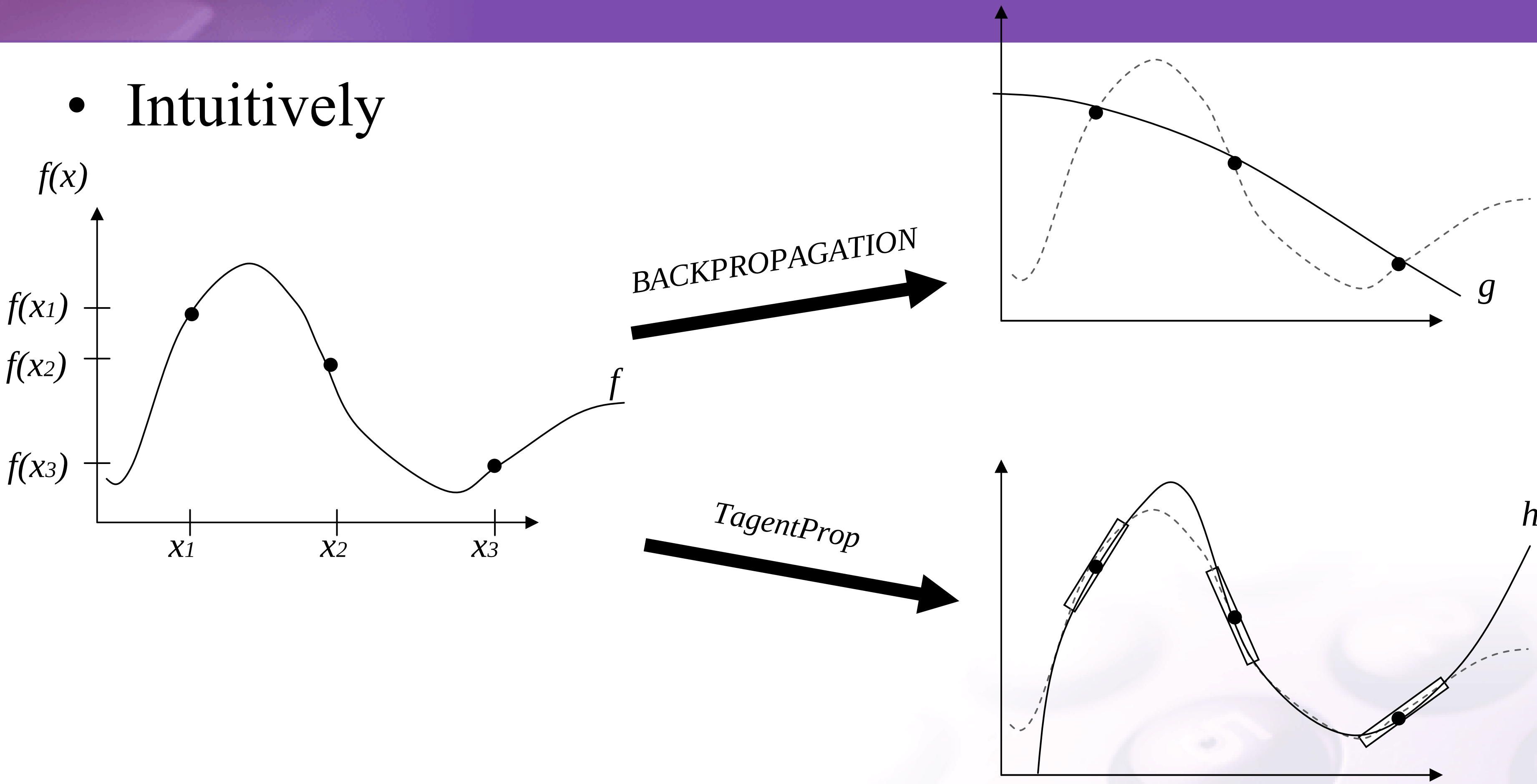
- Prior knowledge
  - Derivatives of the target function
- Trains a neural network to fit both
  - Training values
  - Training derivatives
- TagentProp & EBNN
  - Outperform purely inductive methods
    - Character and object recognition
    - Robot perception and control tasks

TangentProp (cont.)

- Training examples
  - Up to now: $\langle x_i, f(x_i) \rangle$
  - In TangentProp: $\langle x_i, f(x_i), \frac{\partial f(x)}{\partial x} | x_i \rangle$
    - Assumes various training derivatives of the target function are also provided

TangentProp (cont.)

- Intuitively



The learner has a better chance to correctly generalize from the sparse training data

TangentProp (cont.)

- Accept training derivatives with respect to various transformations of the input *x*
  - Learning to recognize handwritten characters
    - Input *x*: An image containing a single handwritten character
    - Task: Correctly classify the character
    - Prior knowledge
      - "The target function is invariant to small rotations of the character within the image"
      - $s(\alpha, x)$: Rotates the image *x* by α degrees

$$\frac{\partial f(s(\alpha, x_i))}{\partial \alpha} = 0$$

TangentProp (cont.)

- c.f.) BACKPROPAGATION
  - Performs gradient descent to attempt to minimize the sum of squared errors

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

- TargentProp
  - Accept multiple transformations
    - Each transformation must be of the form $s_j(\alpha, x)$
      - α : Continuous parameter
      - s_j : Differentiable, $s_j(0,x) = x$

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

- μ : Constant
  - » Relative importance of fitting training values / training derivatives

TangentProp (cont.)

- Recognizing handwritten characters (1992)
  - Images containing a single digit 0 ~ 9
  - Prior knowledge
    - Classification of a character is invariant of vertical and horizontal translation

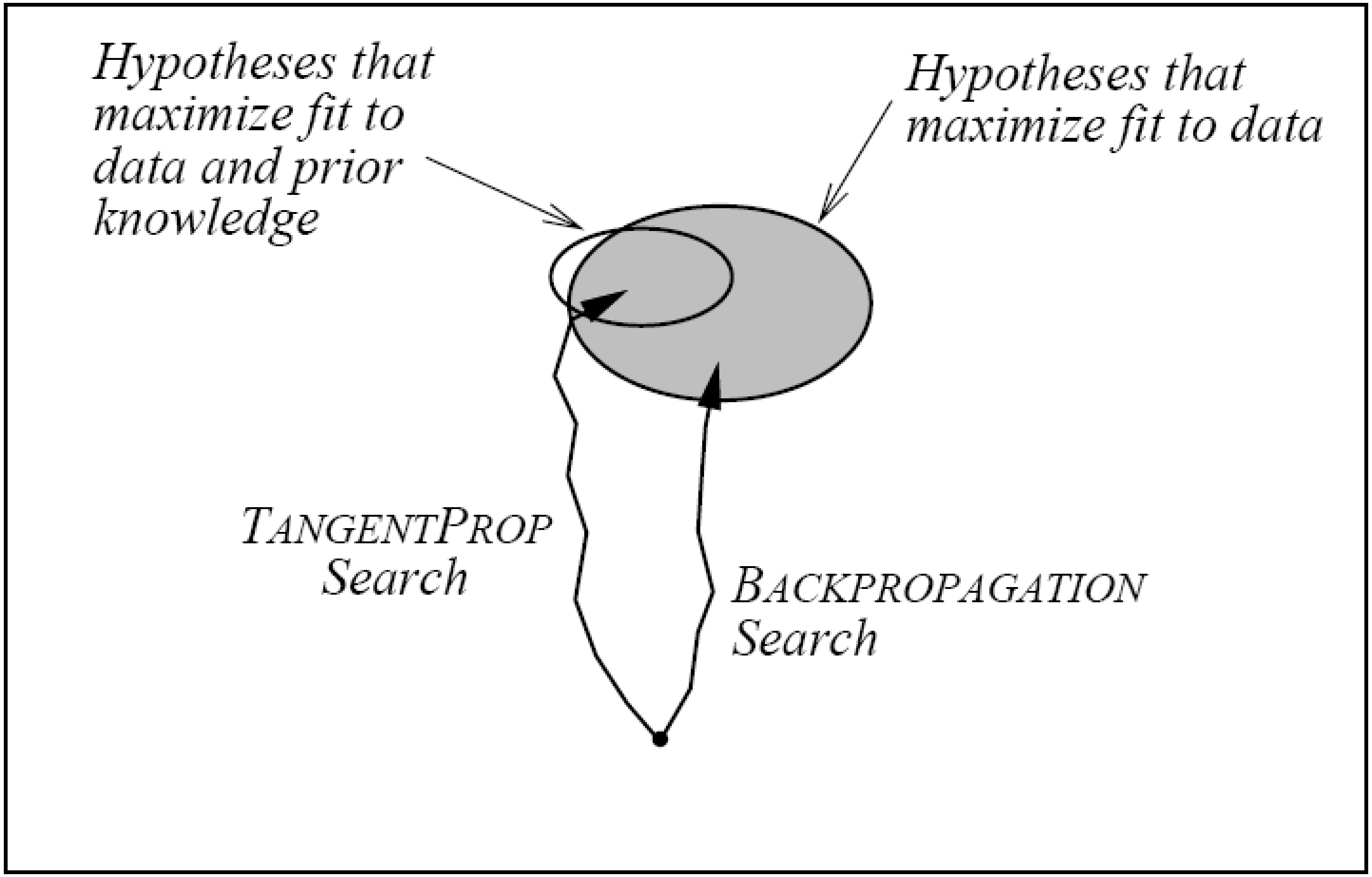
Training set size	Percent error on test set	
	TangentProp	BACKPROPAGATION
10	34	48
20	17	33
40	7	18
80	4	10
160	0	3
320	0	0

TangentProp (cont.)

- The behavior of algorithm is sensitive to μ
- Not robust to errors in the prior knowledge
  - Degree of error in the training derivatives is unlikely to be known in advance

TangentProp (cont.)

- Hypothesis space search in TangentProp



EBNN (Explanation-Based Neural Network Learning)

- EBNN
  - Automatically selects values for μ on an example-by-example basis in order to address the possibility of incorrect prior knowledge
- Using the prior knowledge to alter the search objective
- Builds on TangentProp
  - Compute <u>training derivatives</u> itself for each examples
  - "How to <u>weight the relative importance</u> of the inductive and analytic components of learning"
    $\rightarrow$ Determined by itself

EBNN (cont.)

- Given
  - Training example : <x_i, $f(x_i)$>
  - Domain theory : represented as a set of previously trained neural networks
- Determine
  - A new neural network that approximates the target function f
  - This learned network is trained to fit both the training examples and training derivatives of f extracted from the domain theory

EBNN (cont.)

- Algorithm

Create a feedforward Network

Initialize : small random weights

↓

Determines training derivatives

Predict the value $A(x_i)$ of the target function for training example x_i using domain theory network

↓

Analyze the weights and activation of the domain theory networks

↓

Extract derivatives of $A(x_i)$

↓

Train the target network

Error function

EBNN (cont.)

Error function

$$E = \sum_i \left[\underbrace{(f(x_i) - \hat{f}(x_i))^2}_{} + \mu_i \sum_j \left(\frac{\partial A(x)}{\partial x^j} - \frac{\partial \hat{f}(x)}{\partial x^j} \right)^2_{(x=x_i)} \right] \quad \text{where } \mu_i \equiv 1 - \frac{|A(x_i) - f(x_i)|}{c}$$

<u>Inductive constraint</u>
that the hypothesis must fit
the training data

<u>Analytical constraint</u>
that the hypothesis must fit
the training derivatives

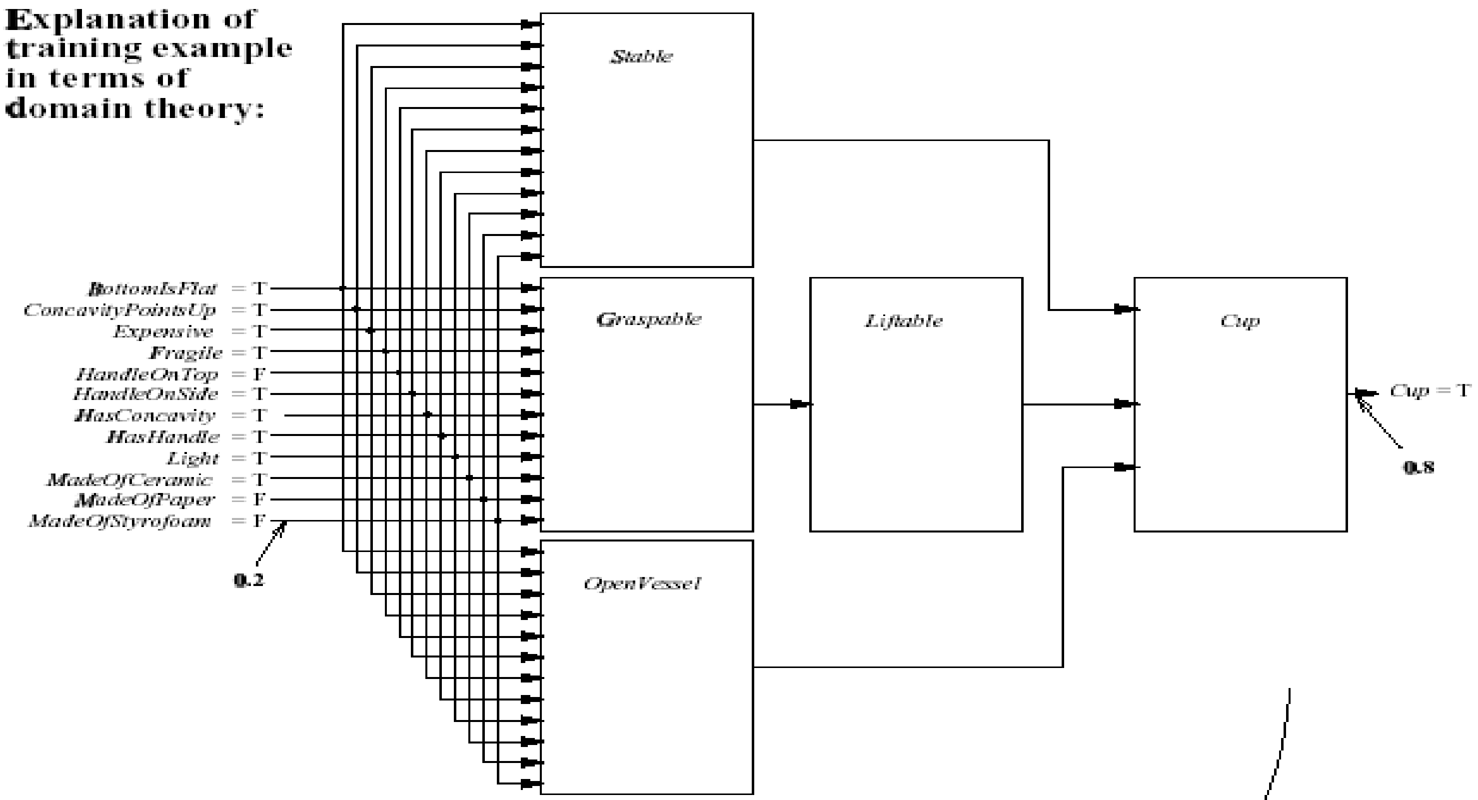
x_i : The i th training instance

$A(x)$: The domain theory prediction for input x

x^j : The j th component of the vector x

c : A normalizing constant ($0 \le \mu_j \le 1$, for all i)

EBNN (cont.)

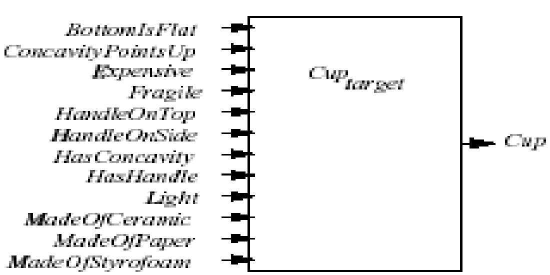


Training derivatives:

$$\left[\frac{\partial Cup}{\partial BottomIsFlat}, \frac{\partial Cup}{\partial ConcavityPointsUp}, \ldots \frac{\partial Cup}{\partial MadeOfStyrofoam}\right]_{x=x_i}$$

Target network:

Learns target network by invoking TangentProp algorithm



EBNN (cont.)

- Remarks
  - Domain theory

    : Expressed as a set of previously learned neural networks

  - Training derivative

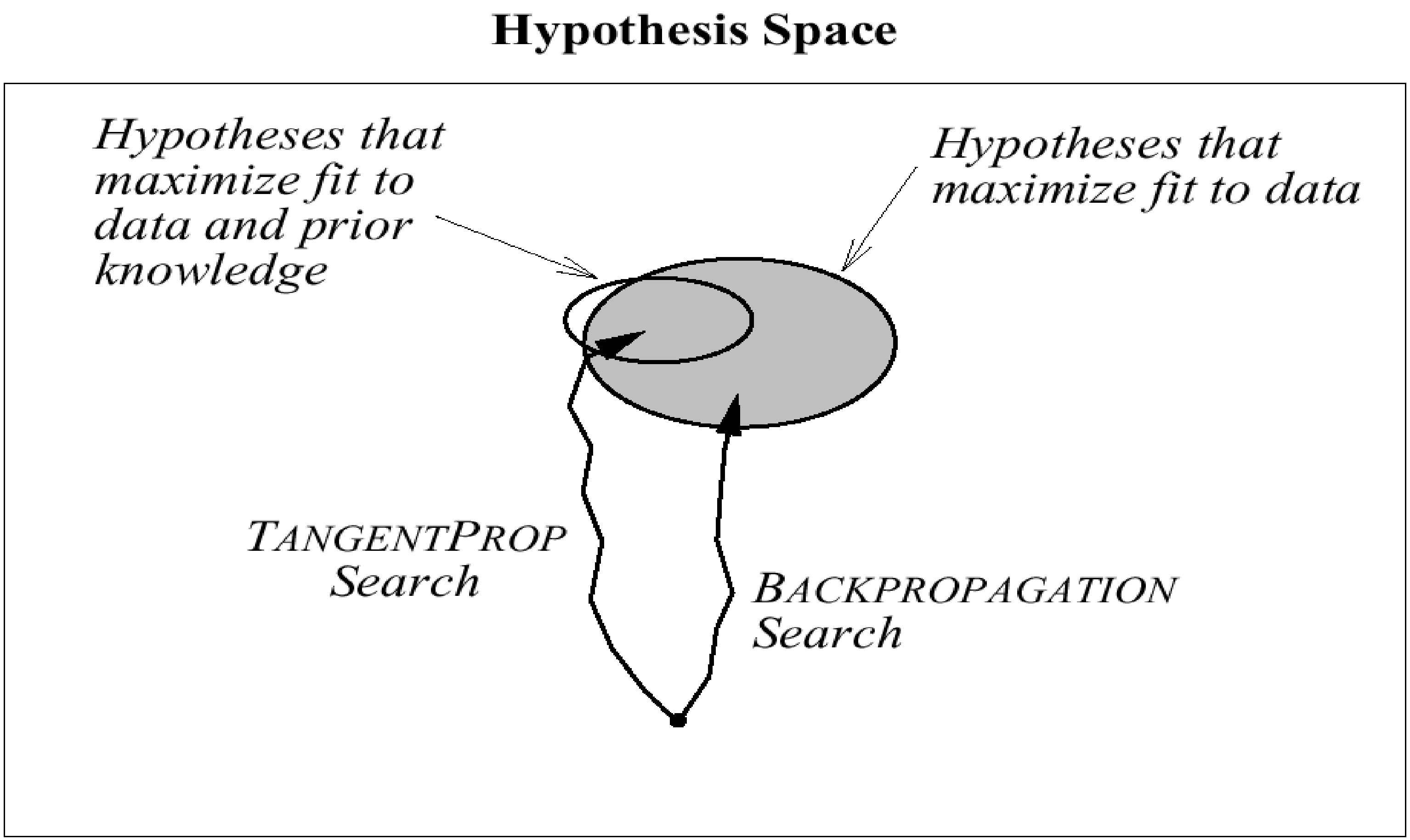
    : How the target function value is influenced by a small change to attribute value

  - μ_i

    : Determined independently for each training example, based on how accurately the domain theory predicts the training value for example

EBNN (cont.)

- Hypothesis Space Search in EBNN



EBNN (cont.)

- EBNN vs. PROLOG-EBG

	EBNN	PROLOG-EBG
Explanation	Training derivatives	Weakest preimage
Domain theory	Neural network	Horn clause
	Imperfect	Perfect
Size	Learns a fixed size neural network	Learns a growing set of Horn clause

FOCL

- Using prior knowledge to augment search operators
- Extension of the purely FOIL

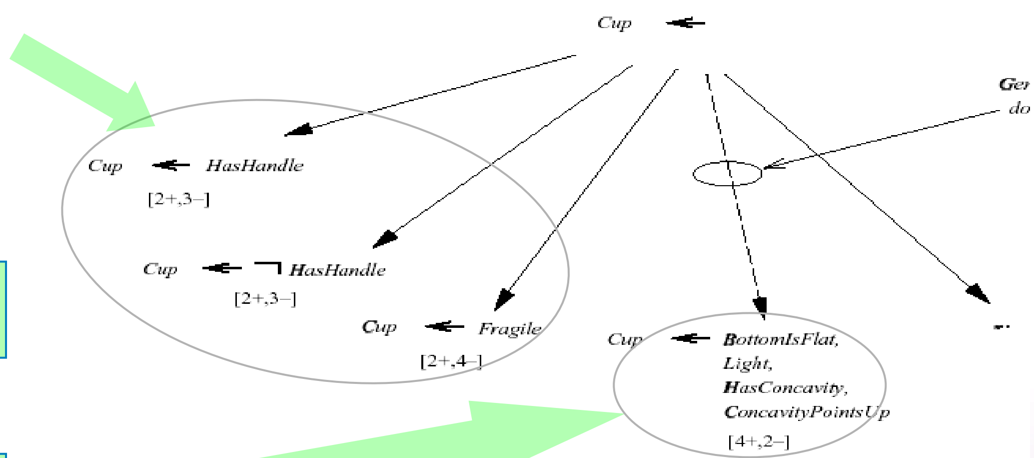
	FOCL	FOIL
Generating candidate specialization	FOIL + Additional specializations based on the domain theory	Add a single new literal to the clause precondition
	Learn a set of first-order Horn clauses Sequential covering algorithm	

FOCL (cont.)

- Operational
  - If a literal is allowed to be used in describing an output hypothesis
- Nonoperational
  - If a literal occur only as intermediate features in the domain theory

FOCL (cont.)

- Algorithm
  - Generating candidate specializations



Selects one of the domain theory clause

Nonoperational literal is replaced

Prune the preconditions of h
unless pruning reduces classification accuracy
over training examples

1. *Cup ← Stable, Liftable, Openvessel*
2. *BottomIsFlat , HasHandle, Light, HasConcavity, ConcavityPointsUp*
3. Remove *HasHandle*
   Cup ← BottomIsFlat , Light, HasConcavity, ConcavityPointsUp

FOCL (cont.)

- Remarks
  - Horn clause of the form

$$\boxed{C \leftarrow O_i \wedge O_b \wedge O_f}$$

O_i : An initial conjunction of operational literals
(added one at a time by the first syntactic operator)
O_b : A conjunction of operational literals
(added in a single step based on the domain theory)
O_f : A final conjunction of operational literals
(added one at a time by the first syntactic operator)

  - Uses both a syntactic generation of candidate specialization and a domain theory driven generation of candidate specialization at each step

FOCL (cont.)

- Hypothesis space