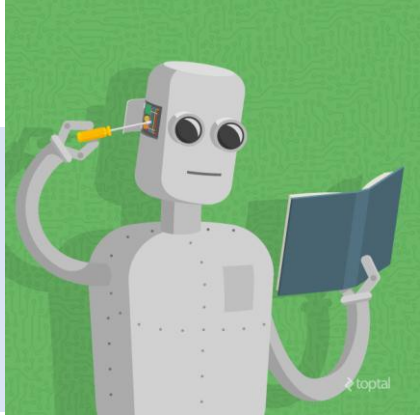
Machine Learning (15CS73)



Text Book :

Tom M. Mitchell, Machine Learning, India Edition 2013, McGraw Hill

Lecture Notes :https://thyagumachinelearning.blogspot.com/

ಡಾ|| ತ್ಯಾಗರಾಜು ಜಿ.ಎಸ್

Head Dept of CSE

SDMIT Ujire

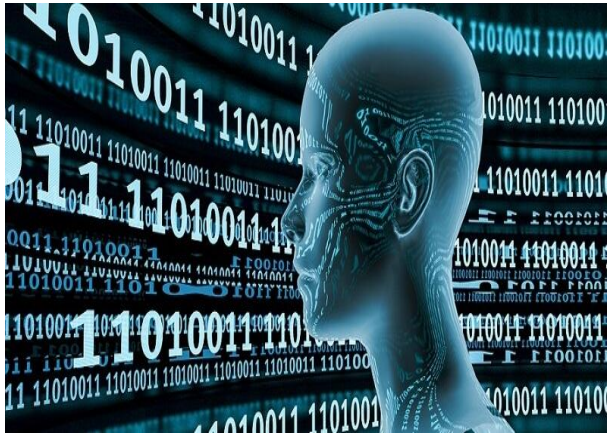
Modules

- **Module1 :** Well posed learning problems, Designing a Learning system, Perspective and Issues in Machine Learning. Concept Learning: Concept learning task, Concept learning as search, Find-S algorithm, Version space, Candidate Elimination algorithm, Inductive Bias.
- **Module2**: **Decision tree representation**, Appropriate problems for decision tree learning, Basic decision tree learning algorithm, hypothesis space search in decision tree learning, Inductive bias in decision tree learning, Issues in decision tree learning
- **Module3: Artificial Neural Networks**: Introduction, Neural Network representation, Appropriate problems, Perceptron's, Backpropagation algorithm.
- **Module4**: Introduction, **Bayes theorem**, Bayes theorem and concept learning, ML and LS error hypothesis, ML for predicting probabilities, MDL principle, Naive Bayes classifier, Bayesian belief networks, EM algorithm
- **Module5:** Motivation, **Estimating hypothesis accuracy**, Basics of sampling theorem, General approach for deriving confidence intervals, Difference in error of two hypothesis, Comparing learning algorithms. **Instance Based Learning**: Introduction, k-nearest neighbor learning, locally weighted regression, radial basis function, cased-based reasoning, **Reinforcement Learning**: Introduction, Learning Task, Q Learning .

Module 1 Chapter 2

Text Book : Machine Learning by Tom M Mitchell (Chapter 1)

Dr Thyagaraju G S
Head Dept of CSE
SDMIT Ujire



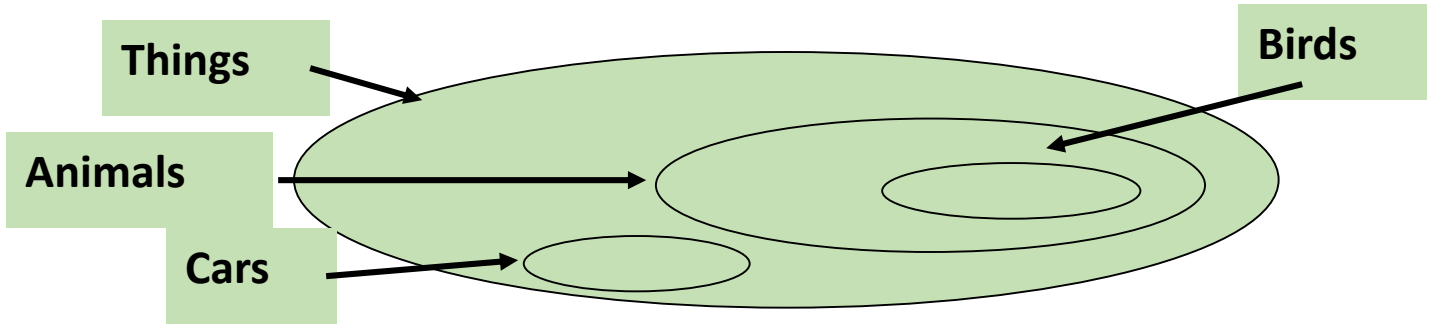
Syllabus

2. Concept Learning :

1. Concept Learning Task
2. Concept Learning as Search
3. Find S – Algorithm
4. Version Space
5. Candidate Elimination Algorithm
6. Inductive Bias

2.1 What is a Concept?

- A **Concept** is a subset of objects or events defined over a larger set [Example: The concept of a *bird* is the subset of all objects (i.e., the set of all *things* or all *animals*) that belong to the category of bird.]



- Alternatively, a concept is a boolean-valued function defined over this larger set [**Example:** a function defined over all *animals* whose value is *true* for birds and *false* for every other animal].

2.1 What is Concept-Learning?

- Given a set of examples labeled as members or non-members of a concept, *concept-learning* consists of automatically inferring the general definition of this concept.
- In other words, *concept-learning* consists of approximating a boolean-valued function from training examples of its input and output.

2.1 A CONCEPT LEARNING TASK

- Consider the example task of learning the target concept "*days on which some Player enjoys his favorite water sport*.“
- Table describes a set of example days, each represented by a set of attributes.
- The attribute EnjoySport indicates whether or not player enjoys his favorite water sport on this day.

Positive and Negative Examples for the target concept Enjoy Sports

Days	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	YEs

Training Examples:

Instance	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
x1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
x2	Sunny	Warm	High	Strong	Warm	Same	Yes
x3	Rainy	Cold	High	Strong	Warm	Change	No
x4	Sunny	Warm	High	Strong	Cool	Change	YEs

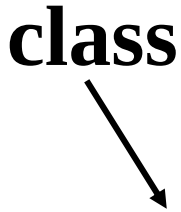
Given Concept

- **Concept:** Good Days for Water Sports (values: Yes, No)

Attributes /Features

- **Sky** (values: Sunny, Cloudy, Rainy)
- **AirTemp** (values: Warm, Cold)
- **Humidity** (values: Normal, High)
- **Wind** (values: Strong, Weak)
- **Water** (Warm, Cool)
- **Forecast** (values: Same, Change)

Example of Training Data Point



x1 <Sunny, Warm, High, Strong, Warm, Same, **Yes**>

Task

- The task is to learn to predict the value of EnjoySport for an arbitrary day ,based on the values of its other attributes.

Hypothesis Representation

- Each hypothesis is a vector of **six constraints** , specifying the values of the six attributes
  - **<Sky , AirTemp ,Humidity ,Wind ,Water and Forecast>.**
- For Each attribute the hypothesis will either :
  - **? :**any value is acceptable by this attribute
  - Specify a single required value (**eg. Warm**) for the attribute or
  - **0**:indicate that no value is acceptable.

Example : < ? ,Cold ,High ,?,?,? >

- Player enjoys his favorite sport only on cold days with high humidity is represented by the expression

Instance	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
x	?	Cold	High	?	?	?	Yes

Example : <?, ?, ?, ?, ?, ?>

- **The most general hypothesis-that every day is a good day for water sports, positive example**

Instance	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
x	?	?	?	?	?	?	Yes

Example : <0,0,0,0,0,0>

- **The most specific possible hypothesis-that no day is a positive example**

Instance	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
x	?	?	?	?	?	?	Yes

Notation

- **Given:**
  - Instances X: Possible days, each described by the attributes
    - *Sky* (with possible values *Sunny*, *Cloudy*, and *Rainy*),
    - *AirTemp* (with values *Warm* and *Cold*),
    - *Humidity* (with values *Normal* and *High*),
    - *Wind* (with values *Strong* and *Weak*),
    - *Water* (with values *Warm* and *Cool*), and
    - *Forecast* (with values *Same* and *Change*).
  - Hypotheses H: Each hypothesis is described by a conjunction of constraints on the attributes *Sky*, *AirTemp*, *Humidity*, *Wind*, *Water*, and *Forecast*. The constraints may be "?" (any value is acceptable), "$\emptyset$" (no value is acceptable), or a specific value.
  - Target concept c: $EnjoySport : X \rightarrow \{0, 1\}$
  - Training examples D: Positive and negative examples of the target function
- **Determine:**
  - A hypothesis h in H such that $h(x) = c(x)$ for all x in X.

The *EnjoySport* concept learning task.

Number of Instances, Concepts, Hypotheses

- Sky: Sunny, Cloudy, Rainy
- AirTemp: Warm, Cold
- Humidity: Normal, High
- Wind: Strong, Weak
- Water: Warm, Cold
- Forecast: Same, Change

#distinct instances : 3*2*2*2*2*2 = 96

#distinct concepts : 2^{96}

#syntactically distinct hypotheses : 5*4*4*4*4*4=5120

#semantically distinct hypotheses : 1+4*3*3*3*3*3=973

The Inductive Learning Hypothesis

The inductive learning hypothesis : Any hypothesis found to approximate the target function well over a **sufficiently large set** of training examples will also approximate the target function well over other unobserved examples.

2.2 CONCEPT LEARNING AS SEARCH

- Concept Learning can be viewed as the task of searching through a large space of hypotheses implicitly defined by the hypothesis representation.
- Selecting a Hypothesis Representation is an important step since it restricts (or *biases*) the space that can be searched. [For example, the hypothesis “If the air temperature is cold **or** the humidity high then it is a good day for water sports” cannot be expressed in our chosen representation.]

General to Specific Ordering of Hypotheses

- **Definition:** Let *hj* and *hk* be boolean-valued functions defined over X.

  Then *hj* is **more-general-than-or-equal-to** *hk* iff For all x in X, [(*hk*(x) = 1) --> (*hj*(x)=1)]

- **Example:**
  - *h1* = <Sunny,?,?,Strong,?,?>
  - *h2* = <Sunny,?,?,?,?,?>

  Every instance that are classified as positive by *h1* will also be classified as positive by *h2* in our example data set. Therefore *h2* is more general than *h1*.

- We also use the ideas of **"strictly"-more-general-than,** and **more-specific-than** (illustration [Mitchell, p. 25])

Definition: Let h_j and h_k be boolean-valued functions defined over X. Then h_j is **more_general_than_or_equal_to** h_k (written $h_j \geq_g h_k$) if and only if

$$(\forall x \in X)[(h_k(x) = 1) \rightarrow (h_j(x) = 1)]$$

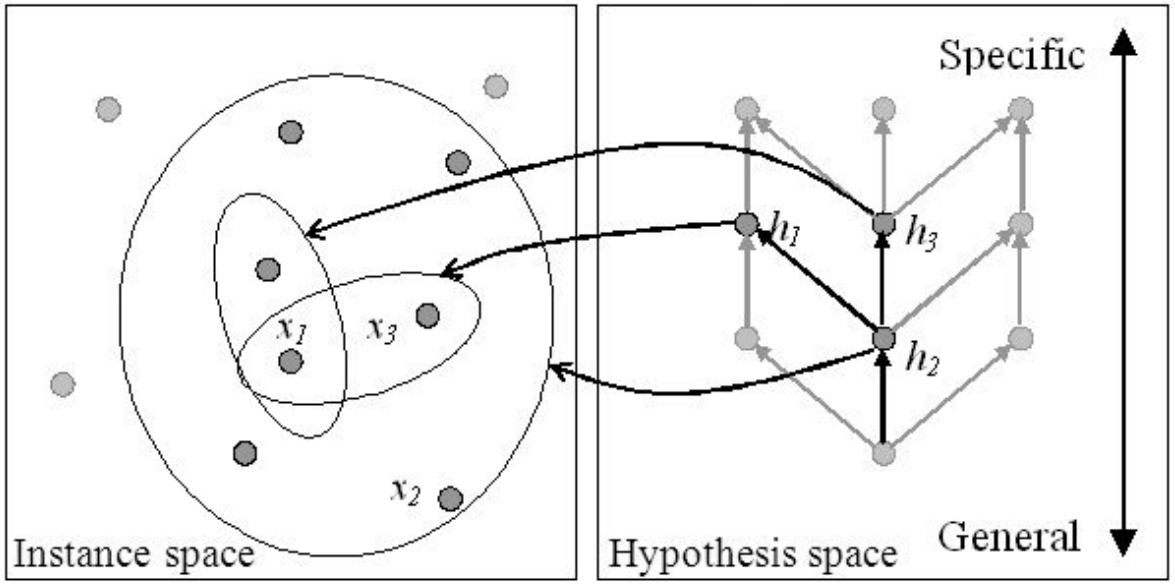
The Structure of *H*

x = instance
h = hypothesis
d = training example
c = target function

Many algorithms for concept learning organize the search by using a useful structure in the hypothesis space: **general-to-specific ordering**!

An example:

h_1 = <*Sunny, ?, ?, Strong, ?, ?*>
h_2 = <*Sunny, ?, ?, ?, ?, ?*>
h_3 = <*Sunny, ?, ?, ?, Cool, ?*>



Because h_2 imposes fewer constraints, it will cover more instances x in X than both h_1 and h_3.

2.3 Find-S, a Maximally Specific Hypothesis Learning Algorithm

1. Initialize **h** to the most specific hypothesis in **H**
2. **For** each positive training instance **x**
   - **For** each attribute constraint $\mathbf{a_i}$ in **h**

     If the constraint $\mathbf{a_i}$ in **h** is satisfied by **x** then do nothing

     else replace $\mathbf{a_i}$ in h by the next more general constraint that is satisfied by **x**
3. Output hypothesis **h**

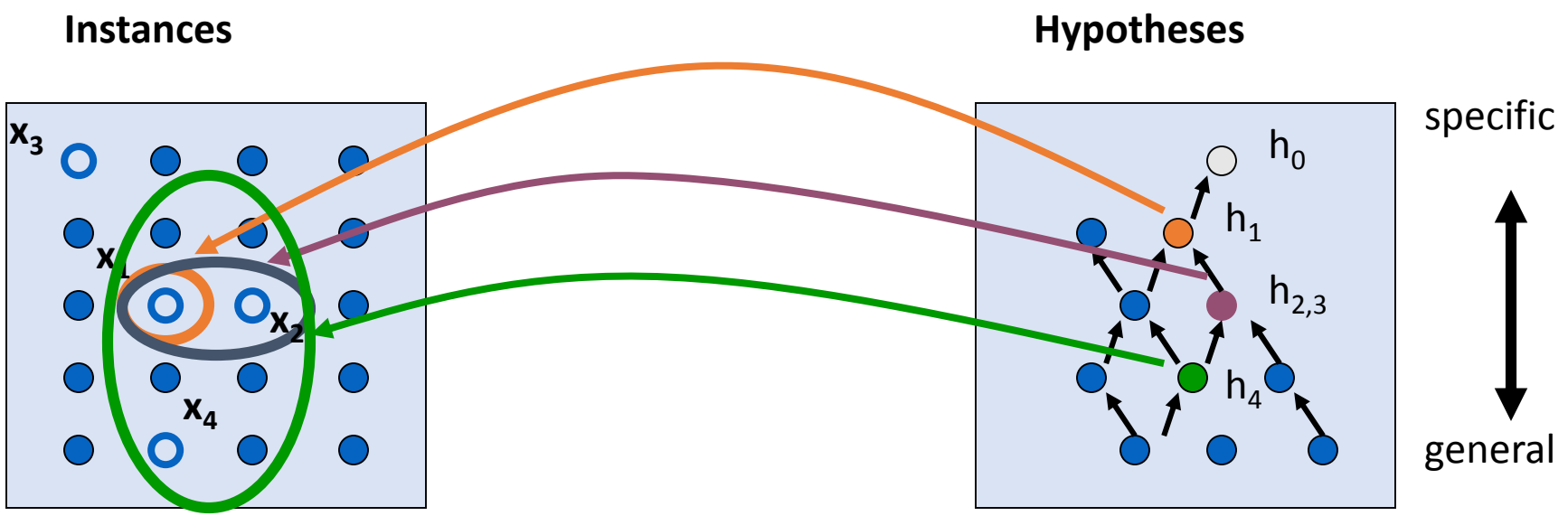
Find S Algorithm

Instance/ Hypothesis	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
ho	**Ø**	**Ø**	**Ø**	**Ø**	**Ø**	**Ø**	**NO**
x1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
h1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
x2	Sunny	Warm	High	Strong	Warm	Same	Yes
h2	Sunny	Warm	?	Strong	Warm	Same	Yes
x3	Rainy	Cold	High	Strong	Warm	Change	No
h3	Sunny	Warm	?	Strong	Warm	Same	Yes
x4	Sunny	Warm	High	Strong	Cool	Change	YEs
h4	Sunny	Warm	?	Strong	?	?	YES

Find S Algorithm :Hypothesis Set

Instance/ Hypothesis	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
Most Specific	**Ø**	**Ø**	**Ø**	**Ø**	**Ø**	**Ø**	**NO**
h1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
h2	Sunny	Warm	?	Strong	Warm	Same	Yes
h3	Sunny	Warm	?	Strong	Warm	Same	Yes
h4	Sunny	Warm	?	Strong	?	?	Yes
Most General	?	?	?	?	?	?	Yes

Hypothesis Space Search by Find-S



x_1=<Sunny,Warm,Normal,Strong,Warm,Same>+

x_2=<Sunny,Warm,High,Strong,Warm,Same> +

x_3=<Rainy,Cold,High,Strong,Warm,Change> -

x_4=<Sunny,Warm,High,Strong,Cool,Change> +

h_0 = < Ø, Ø, Ø, Ø, Ø, Ø,>

h_1=< Sunny,Warm,Normal,Strong,Warm,Same>

$h_{2,3}$=< Sunny,Warm,?,Strong,Warm,Same>

h_4=< Sunny,Warm,?, Strong,?,?>

Properties of Find-S

- Hypothesis space described by conjunctions of attributes
- Find-S will output the most specific hypothesis within H that is consistent with the positive training examples
- The output hypothesis will also be consistent with the negative examples, provided the target concept is contained in H.

Complaints about Find-S

- Can’t tell if the learner has converged to the target concept, in the sense that it is unable to determine whether it has found the ***only*** **hypothesis consistent** with the training examples.
- Can’t tell **when training data is inconsistent**, as it ignores negative training examples.
- Why prefer the most specific hypothesis?
- What if there are multiple maximally specific hypothesis?

2.4 Version Spaces

- A hypothesis h is **consistent** with a set of training examples D of target concept if and only if h(x)=c(x) for each training example <x,c(x)> in D.

  Consistent(h,D) := $\forall$<x,c(x)>$\in$D h(x)=c(x)

- The **version space**, $VS_{H,D}$, with respect to hypothesis space **H**, and training set **D**, is the subset of hypotheses from **H** consistent with all training examples:

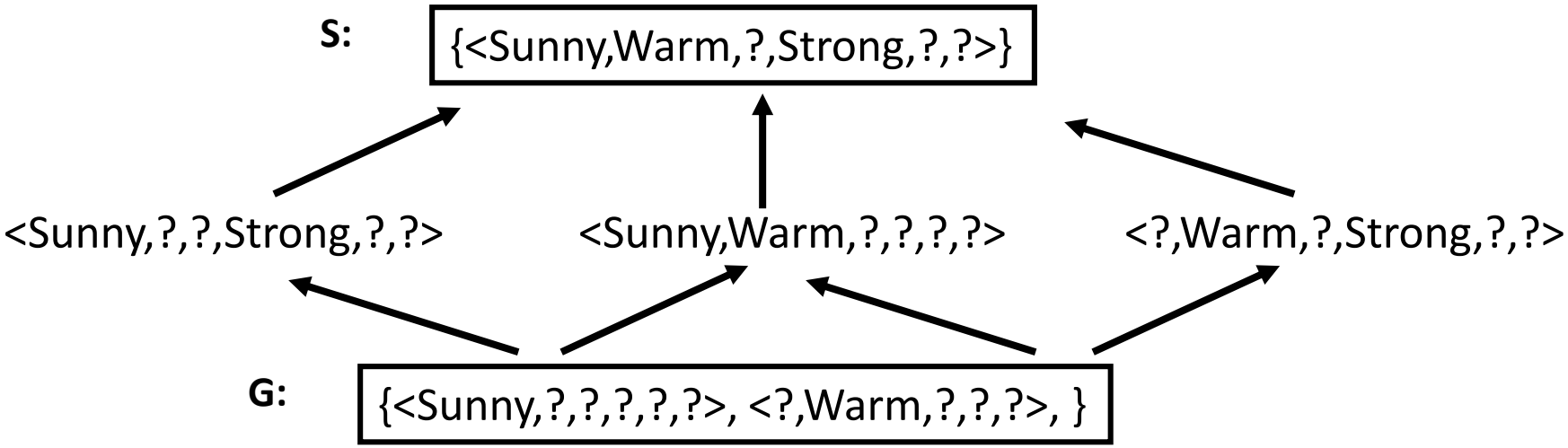
  $VS_{H,D}$ = {h $\in$ H | Consistent(h,D) }

List-Then Eliminate Algorithm

1. *VersionSpace* ← a list containing every hypothesis in H
2. For each training example <x,c(x)>

   remove from *VersionSpace* any hypothesis that is inconsistent with the training example $h(x) \neq c(x)$
3. Output the list of hypotheses in *VersionSpace*

Example Version Space



x_1 = <Sunny Warm Normal Strong Warm Same> +
x_2 = <Sunny Warm High Strong Warm Same> +
x_3 = <Rainy Cold High Strong Warm Change> -
x_4 = <Sunny Warm High Strong Cool Change> +

Representing Version Spaces

- The **general boundary**, G, of version space $VS_{H,D}$ is the set of maximally general members.
- The **specific boundary**, S, of version space $VS_{H,D}$ is the set of maximally specific members.
- Every member of the version space lies between these boundaries

$$VS_{H,D} = \{h \in H | (\exists s \in S) (\exists g \in G) (g \geq h \geq s)$$

where $x \geq y$ means x is more general or equal than y

2.6 Candidate Elimination Algorithm

- The CANDIDATE-ELIMINATION algorithm computes the version space containing all hypotheses from H that are consistent with an observed sequence of training examples.
- It begins by initializing the version space to the set of all hypotheses in H; that is, by initializing the G boundary set to contain the most general hypothesis in H

$$G_0 \leftarrow \{(?, ?, ?, ?, ?, ?)\}$$

- and initializing the S boundary set to contain the most specific (least general) hypothesis

$$S_0 \leftarrow \{(\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)\}$$

2.5 Candidate Elimination Algorithm

$G \leftarrow$ maximally general hypotheses in H

S $\leftarrow$ maximally specific hypotheses in H

For each training example d=<x,c(x)>

Case 1 : If d is a positive example

Remove from G any hypothesis that is inconsistent with d

For each hypothesis s in S that is not consistent with d

- *Remove s from S.*
- *Add to S all minimal generalizations h of s such that*
  - *h consistent with d*
  - *Some member of G is more general than h*
- *Remove from S any hypothesis that is more general than another hypothesis in S*

Case 2: If d is a negative example

Remove from S any hypothesis that is inconsistent with d

For each hypothesis g in G that is not consistent with d

- *remove g from G.*
- *Add to G all minimal specializations h of g such that*
  - *h consistent with d*
  - *Some member of S is more specific than h*
- *Remove from G any hypothesis that is less general than another hypothesis in G*

An Illustrative Example

Figure traces the CANDIDATE-ELIMINATION algorithm applied to the first two training examples from table

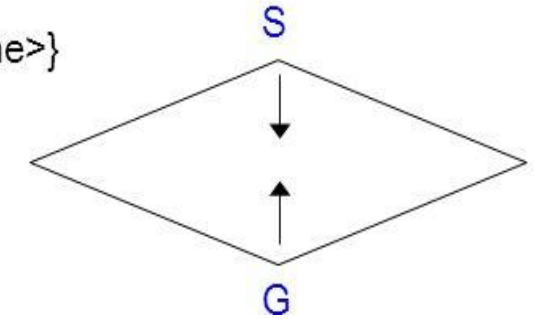
Candidate-Elimination Algorithm

Example	Sky	AirTemp	Humidity	Wind	Water	Forecast	EnjoySport
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

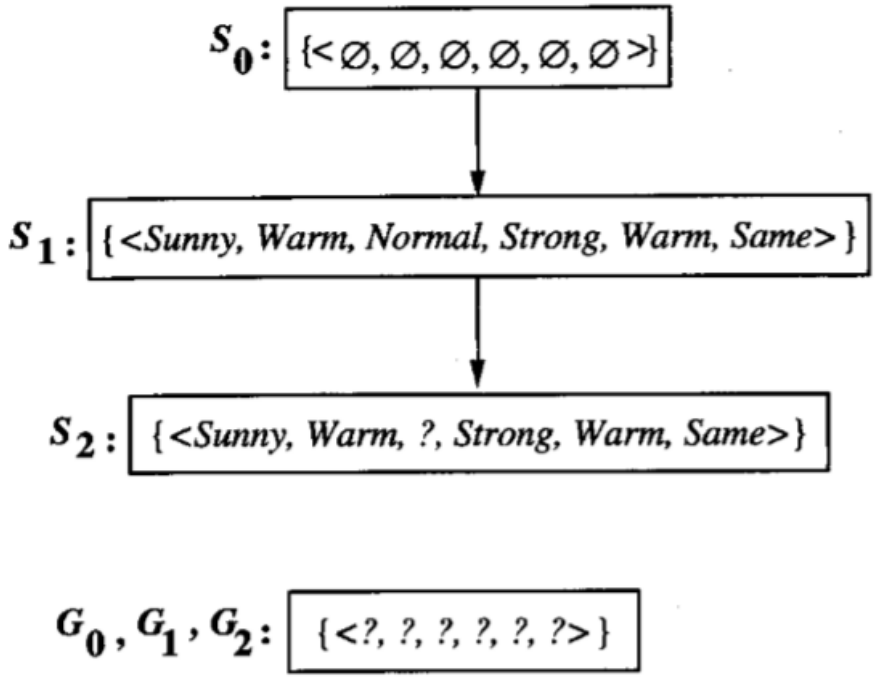
S_0 = {<∅, ∅, ∅, ∅, ∅, ∅>}
G_0 = {<?, ?, ?, ?, ?, ?>}

S_1 = {<Sunny, Warm, Normal, Strong, Warm, Same>}
G_1 = {<?, ?, ?, ?, ?, ?>}

S_2 = {<Sunny, Warm, ?, Strong, Warm, Same>}
G_2 = {<?, ?, ?, ?, ?, ?>}



Trace1



Training examples:

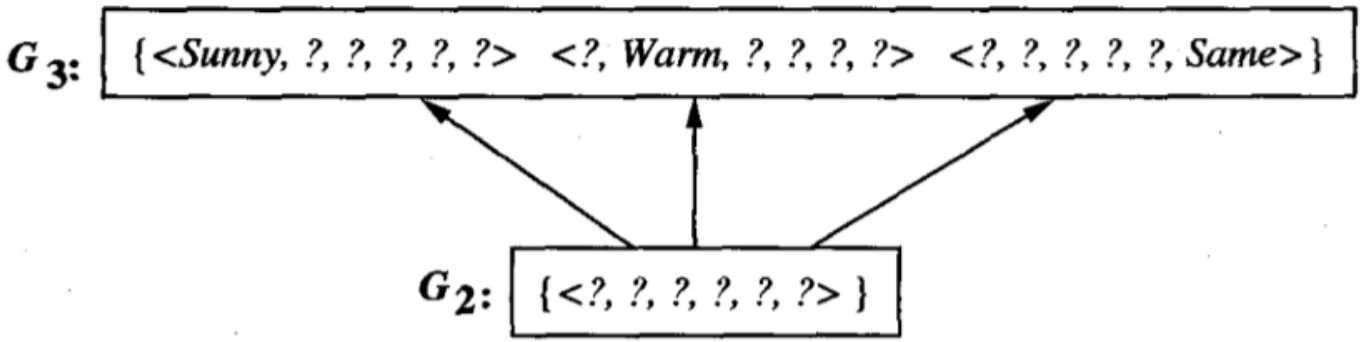
1. <Sunny, Warm, Normal, Strong, Warm, Same>, Enjoy Sport = Yes

2. <Sunny, Warm, High, Strong, Warm, Same>, Enjoy Sport = Yes

CANDIDATE-ELIMINATION Trace 1. S_0 and G_0 are the initial boundary sets corresponding to the most specific and most general hypotheses. Training examples 1 and 2 force the S boundary to become more general, as in the FIND-S algorithm. They have no effect on the G boundary.

Trace 2

S_2, S_3: { <Sunny, Warm, ?, Strong, Warm, Same> }

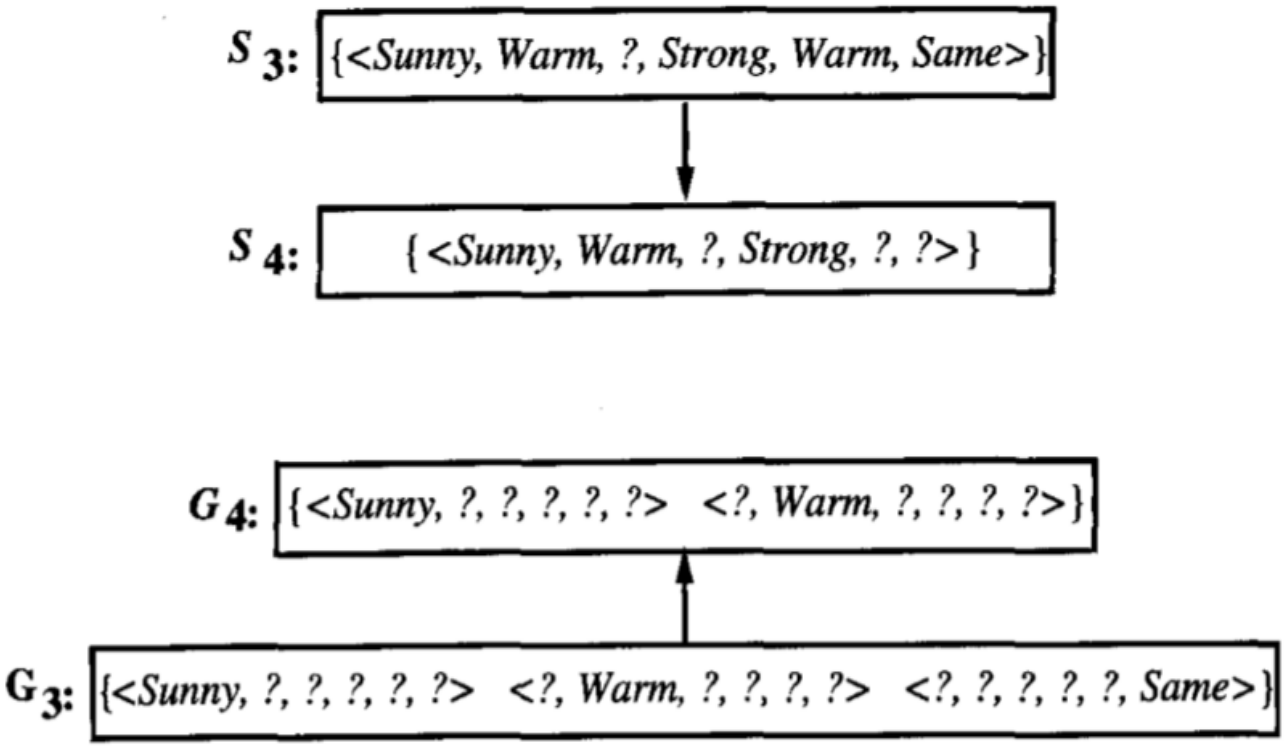


Training Example:

3. <Rainy, Cold, High, Strong, Warm, Change>, EnjoySport=No

CANDIDATE-ELIMINATION Trace 2. Training example 3 is a negative example that forces the G_2 boundary to be specialized to G_3. Note several alternative maximally general hypotheses are included in G_3.

Trace3

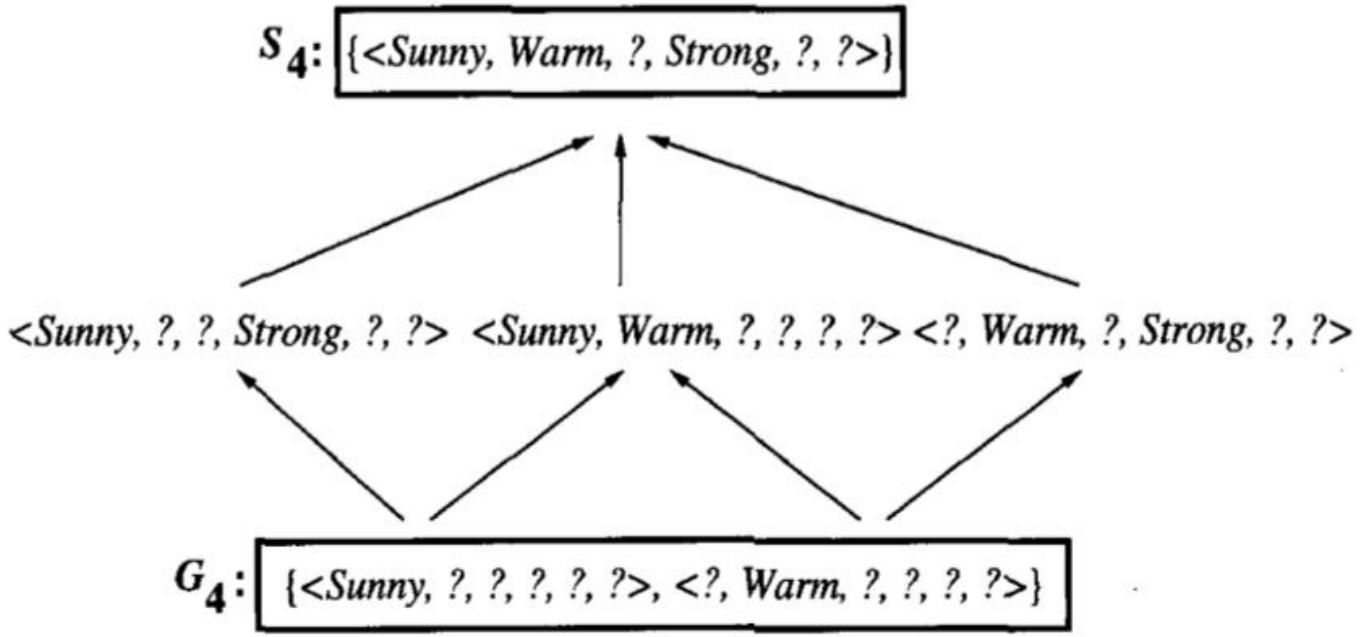


Training Example:

4.<*Sunny, Warm, High, Strong, Cool, Change*>, *EnjoySport = Yes*

CANDIDATE-ELIMINATION Trace 3. The positive training example generalizes the S boundary, from S_3 to S_4. One member of G_3 must also be deleted, because it is no longer more general than the S_4 boundary.

Final Version Space



The final version space for the *EnjoySport* concept learning problem and training examples described earlier.

REMARKS ON VERSION SPACES AND CANDIDATE-ELIMI

1. Will the CANDIDATE-ELIMINATION Algorithm Converge to the Correct Hypothesis?
2. What Training Example Should the Learner Request Next?
3. How Can Partially Learned Concepts Be Used?

2.6 Inductive Bias

Inductive Bias I: A Biased Hypothesis Space

Database:

Day	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*WaterSport*
1	Sunny	Warm	Normal	Strong	Cool	Change	Yes
2	Cloudy	Warm	Normal	Strong	Cool	Change	Yes
3	Rainy	Warm	Normal	Strong	Cool	Change	No

class

- Given our previous choice of the hypothesis space representation, **no hypothesis is consistent with the above database**: we have ***BIASED*** the learner to consider only conjunctive hypotheses

Inductive Bias II: An Unbiased Learner

- Idea: Choose H that expresses every teachable concept, that means H is the set of all possible subsets of X called the power set P(X)
- $|X|=96$, $|P(X)|=2^{96} \sim 10^{28}$ distinct concepts
- H = disjunctions, conjunctions, negations
  - e.g. <Sunny Warm Normal ? ? ?> v <? ? ? ? ? Change>
- H surely contains the target concept.

Inductive Bias II: An Unbiased Learner

- In order to solve the problem caused by the bias of the hypothesis space, we can remove this bias and allow the hypotheses to represent every possible subset of instances. The previous database could then be expressed as: <Sunny, ?,?,?,?,?> v <Cloudy,?,?,?,?,?,?>
- ***<u>However, such an unbiased learner is not able to generalize beyond the observed examples!!!!</u>*** All the non-observed examples will be well-classified by half the hypotheses of the version space and misclassified by the other half.

Inductive Bias II: An Unbiased Learner

What are S and G in this case?

Assume positive examples (x_1, x_2, x_3) and negative examples (x_4, x_5)

S : { (x_1 v x_2 v x_3) } G : { ¬ (x_4 v x_5) }

The only examples that are classified are the training examples themselves. In other words in order to learn the target concept one would have to present every single instance in X as a training example.

Each unobserved instance will be classified positive by precisely half the hypothesis in VS and negative by the other half.

Inductive Bias III: The Futility of Bias-Free Learning

- **<u>Fundamental Property of Inductive Learning</u>** A learner that makes no a priori assumptions regarding the identity of the target concept has no rational basis for classifying any unseen instances.
- **<u>We constantly have recourse to inductive biases</u>** *Example:* we all know that the sun will rise tomorrow. Although we cannot *deduce* that it will do so based on the fact that it rose today, yesterday, the day before, etc., we do take this **leap of faith** or use this **inductive bias**, naturally!

Inductive Bias

Consider:

- Concept learning algorithm L
- Instances X, target concept c
- Training examples D_c={<x,c(x)>}
- Let L(x_i,D_c) denote the classification assigned to instance x_i by L after training on D_c.

Definition:

The inductive bias of L is any minimal set of assertions B such that for any target concept c and corresponding training data D_c

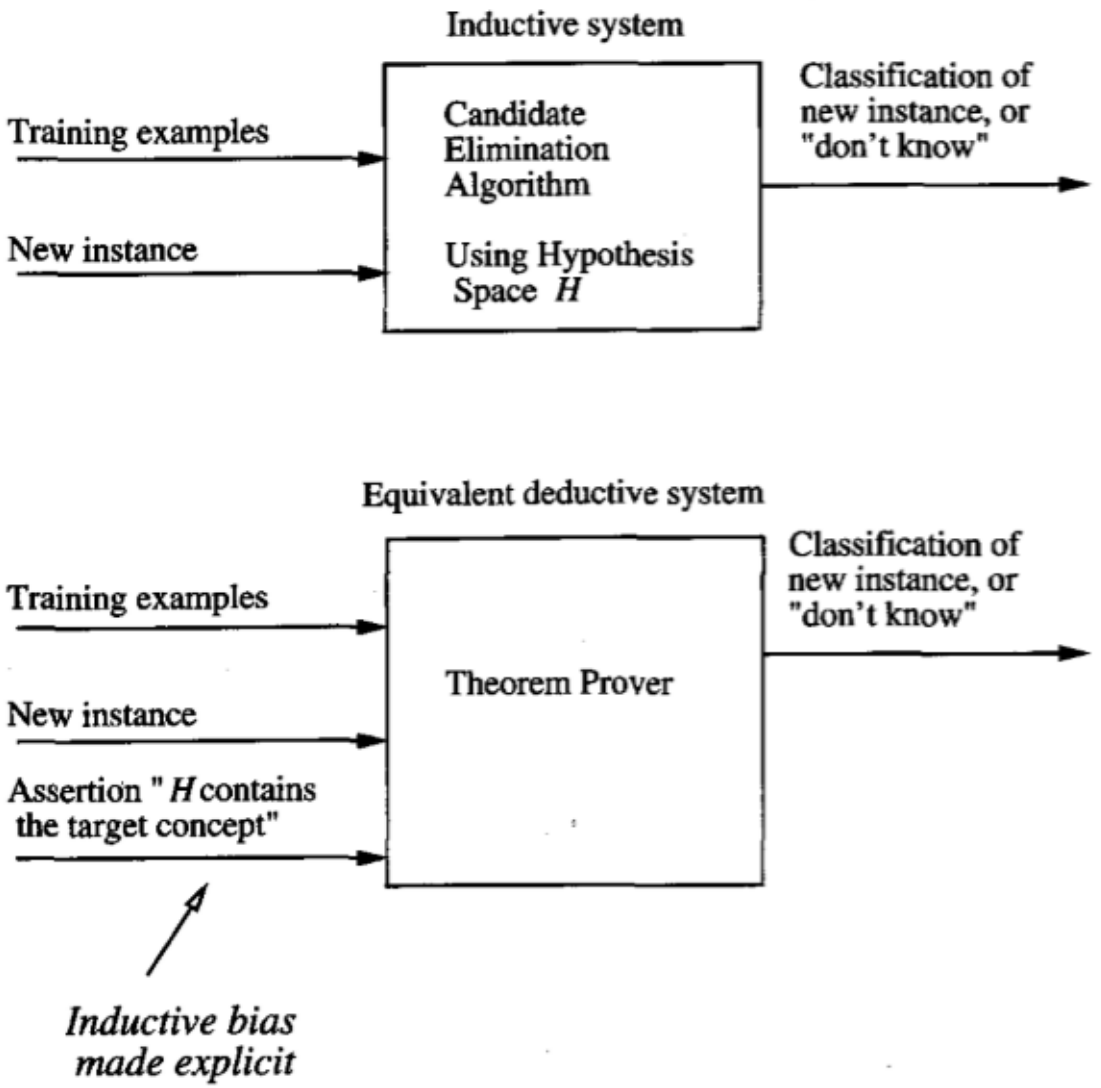
$(\forall x_i \in X)[B \wedge D_c \wedge x_i] \vdash L(x_i, D_c)$

Where A |-- B means that A logically entails B.

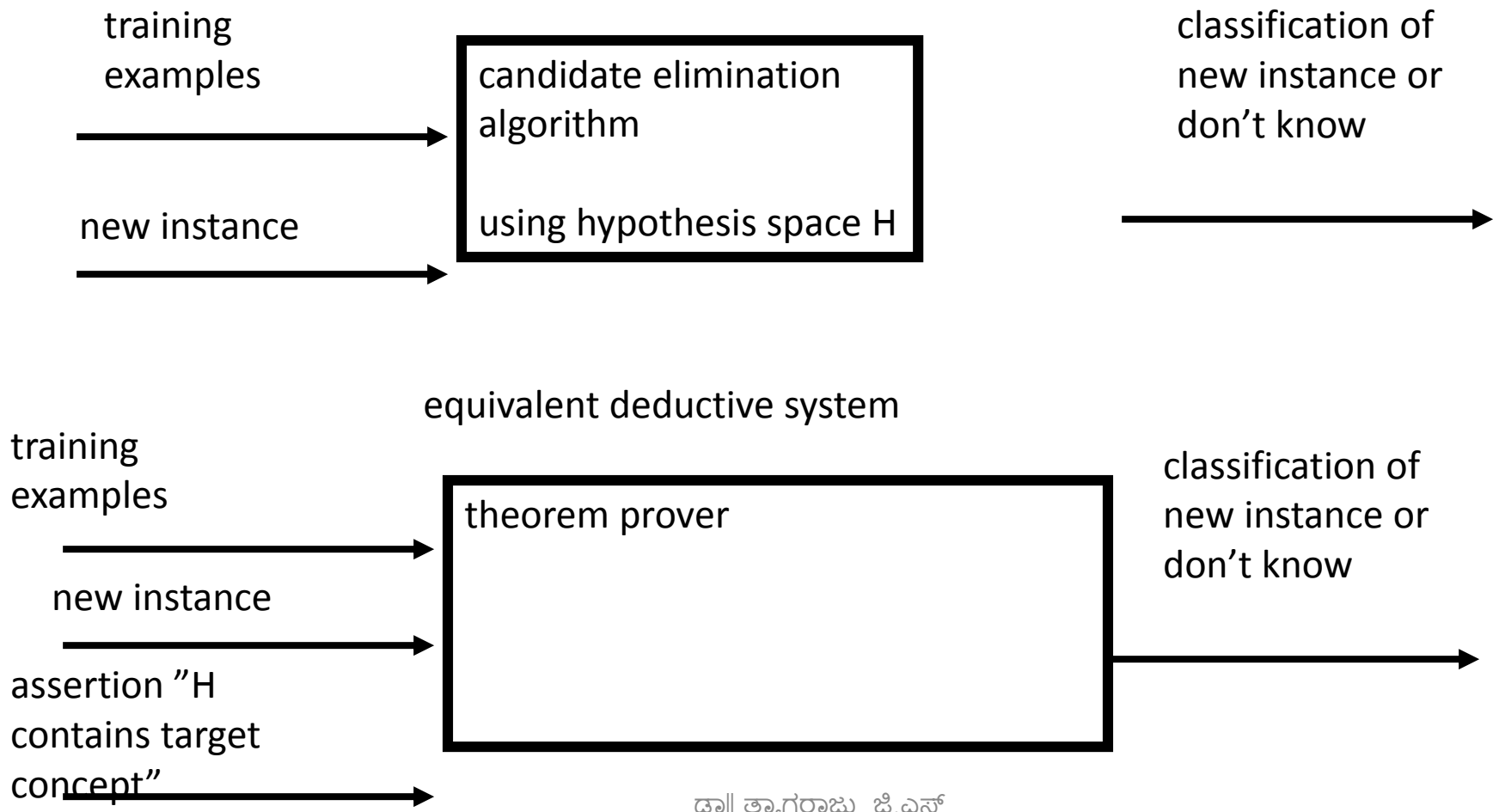
<u>Inductive bias of CANDIDATE-ELIMINATION algorithm.</u> The target concept c is contained in the given hypothesis space H.

Inductive Systems and Equivalent Deductive Systems

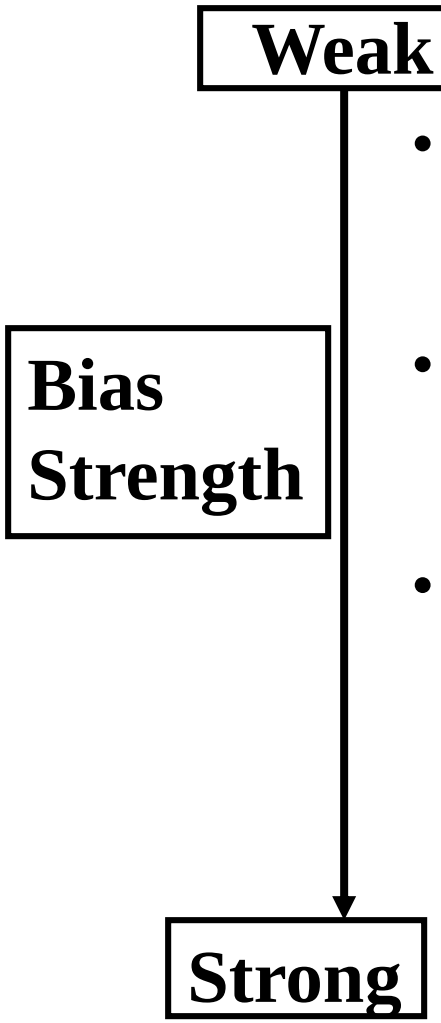
- Modeling inductive systems by equivalent deductive systems. The input-output behavior of the CANDIDATE-ELIMINATION Algorithm using a hypothesis space H is identical to that of a deductive theorem prover utilizing the assertion "*H contains the target concept*." This assertion is therefore called the inductive bias of the CANDIDATE-ELIMINATION Algorithm . Characterizing inductive systems by their inductive bias allows modeling them by their equivalent deductive systems. This provides a way to compare inductive systems according to their policies for generalizing beyond the observed training data.



Inductive Systems and Equivalent Deductive Systems



Ranking Inductive Learners according to their Biases



- **Rote-Learner:** This system simply memorizes the training data and their classification--- No generalization is involved.
- **Candidate-Elimination:** New instances are classified only if all the hypotheses in the version space agree on the classification
- **Find-S:** New instances are classified using the most specific hypothesis consistent with the training data