

## Aims

This lecture aims to develop your understanding of representing and searching hypothesis spaces for *concept learning*. Following it you should be able to:

- define a representation for concepts
- define a hypothesis space in terms of generality ordering on concepts
- describe an algorithm to search a hypothesis space
- express the framework of version spaces
- describe an algorithm to search a hypothesis space using the framework of version spaces
- explain the role of inductive bias in concept learning

COMP9417: March 11, 2009

Fundamentals of Concept Learning: Slide 1

09s1: COMP9417 Machine Learning and Data Mining

# Fundamentals of Concept Learning

March 11, 2009

**Acknowledgement:** Material derived from slides for the book  
Machine Learning, Tom Mitchell, McGraw-Hill, 1997  
<http://www-2.cs.cmu.edu/~tom/mlbook.html>

## Overview

**Concept Learning** inferring a Boolean-valued function from training examples of its input and output.

- Learning from examples
- General-to-specific ordering over hypotheses
- Version spaces and candidate elimination algorithm
- Picking new examples
- The need for inductive bias

Note: simple approach assuming no noise, illustrates key concepts

## Training Examples for *EnjoySport*

Sky	Temp	Humid	Wind	Water	Forecst	EnjoySpt
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

What is the general concept?

## Representing Hypotheses

Many possible representations ...

Here,  $h$  is a conjunction of constraints on attributes.

Each constraint can be:

- a specific value (e.g.,  $Water = Warm$ )
- don't care (e.g., " $Water = ?$ ")
- no value allowed (e.g., " $Water = \emptyset$ ")

For example,

Sky    AirTemp    Humid    Wind    Water    Forecast  
 $\langle Sunny, \quad ? \quad ? \quad Strong, \quad ? \quad Same \rangle$

## The Prototypical Concept Learning Task

- **Given:**

- Instances  $X$ : Possible days, each described by the attributes

Attribute	Values
Sky	Sunny, Cloudy, Rainy
AirTemp	Warm, Cold
Humid	Normal, High
Wind	Strong, Weak
Water	Warm, Cool
Forecast	Same, Change

## The Prototypical Concept Learning Task

- Target function  $c: EnjoySport : X \rightarrow \{0, 1\}$
- Hypotheses  $H$ : Conjunctions of literals. E.g.

$\langle ?, Cold, High, ?, ?, ? \rangle$ .

- Training examples  $D$ : Positive and negative examples of the target function

$\langle x_1, c(x_1) \rangle, \dots, \langle x_m, c(x_m) \rangle$

- **Determine:** A hypothesis  $h$  in  $H$  such that  $h(x) = c(x)$  for all  $x$  in  $D$  (usually called the *target 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.

## Concept Learning as Search

**Question:** What can be learned ?

**Answer:** (only) what is in the *hypothesis space*

How big is the hypothesis space for *EnjoySport* ?

Instance space

$$\begin{aligned} \text{Sky} \times \text{AirTemp} \times \dots \times \text{Forecast} &= 3 \times 2 \times 2 \times 2 \times 2 \times 2 \\ &= 96 \end{aligned}$$

## Concept Learning as Search

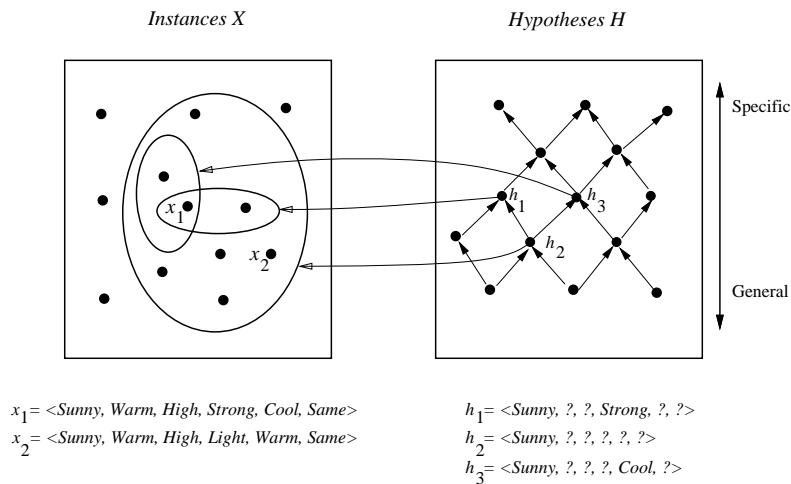
Hypothesis space

$$\begin{aligned} \text{Sky} \times \text{AirTemp} \times \dots \times \text{Forecast} &= 5 \times 4 \times 4 \times 4 \times 4 \times 4 \\ &= 5120 \\ (\text{semantically distinct* only}) &= 1 + (4 \times 3 \times 3 \times 3 \times 3 \times 3) \\ &= 973 \end{aligned}$$

\* any hypothesis with an  $\emptyset$  constraint covers no instances, hence all are semantically equivalent.

The learning problem  $\equiv$  searching a hypothesis space. How ?

## Instances, Hypotheses, and More-General-Than



## A generality order on hypotheses

**Definition:** Let  $h_j$  and  $h_k$  be Boolean-valued functions defined over instances  $X$ . Then  $h_j$  is **more-general-than-or-equal-to**  $h_k$  (written  $h_j \geq_g h_k$ ) if and only if

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

Intuitively,  $h_j$  is **more-general-than-or-equal-to**  $h_k$  if any instance satisfying  $h_k$  also satisfies  $h_j$ .

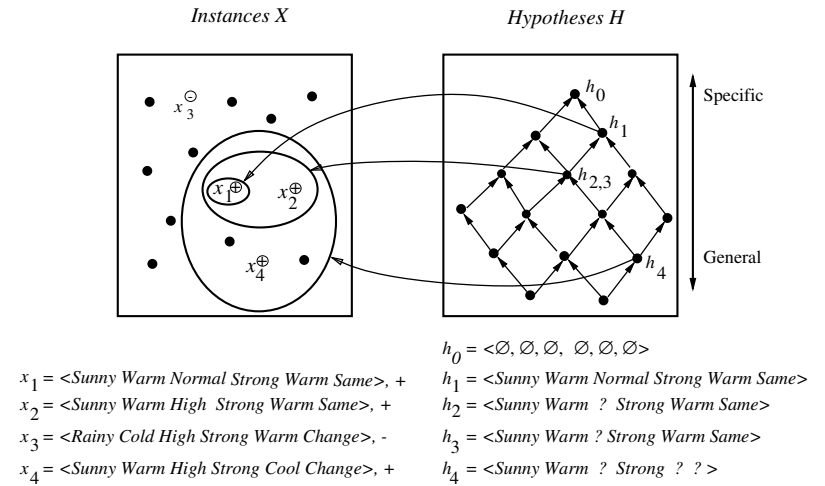
$h_j$  is (strictly) **more-general-than**  $h_k$  (written  $h_j >_g h_k$ ) if and only if  $(h_j \geq_g h_k) \wedge (h_k \not\geq_g h_j)$ .

$h_j$  is **more-specific-than**  $h_k$  when  $h_k$  is **more-general-than**  $h_j$ .

## The Find-S Algorithm

1. Initialize  $h$  to the most specific hypothesis in  $H$
2. For each positive training instance  $x$ 
  - For each attribute constraint  $a_i$  in  $h$ 
    - If the constraint  $a_i$  in  $h$  is satisfied by  $x$ 
      - Then do nothing
      - Else replace  $a_i$  in  $h$  by the next more general constraint that is satisfied by  $x$

## Hypothesis Space Search by Find-S



## Find-S - does it work ?

**Assume:** a hypothesis  $h_c \in H$  describes target function  $c$ , and training data is error-free.

By definition,  $h_c$  is consistent with all positive training examples and can never cover a negative example.

For each  $h$  generated by Find-S,  $h_c$  is **more\_general\_than\_or\_equal\_to**  $h$ .

So  $h$  can never cover a negative example.

## Complaints about Find-S

- Can't tell whether it has learned concept
  - learned hypothesis may not be the only consistent hypothesis
- Can't tell when training data inconsistent
  - cannot handle noisy data
- Picks a maximally specific  $h$  (why?)
  - might require maximally general  $h$

## Version Spaces

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

$$\text{Consistent}(h, D) \equiv (\forall \langle x, c(x) \rangle \in D) h(x) = c(x)$$

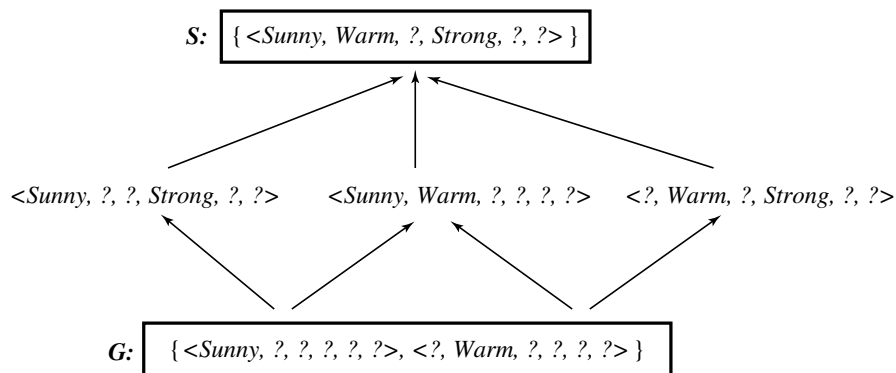
The **version space**,  $VS_{H,D}$ , with respect to hypothesis space  $H$  and training examples  $D$ , is the subset of hypotheses from  $H$  consistent with all training examples in  $D$ .

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

## The List-Then-Eliminate Algorithm

1.  $VersionSpace \leftarrow$  a list containing every hypothesis in  $H$
2. For each training example,  $\langle x, c(x) \rangle$   
remove from  $VersionSpace$  any hypothesis  $h$  for which  $h(x) \neq c(x)$
3. Output the list of hypotheses in  $VersionSpace$

## Example Version Space



## Representing Version Spaces

The **General boundary**,  $G$ , of version space  $VS_{H,D}$  is the set of its maximally general members

The **Specific boundary**,  $S$ , of version space  $VS_{H,D}$  is the set of its 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 to  $y$

## The Candidate Elimination Algorithm

$G \leftarrow$  maximally general hypotheses in  $H$

$S \leftarrow$  maximally specific hypotheses in  $H$

For each training example  $d$ , do

- If  $d$  is a positive example
  - Remove from  $G$  any hypothesis 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
      1.  $h$  is consistent with  $d$ , and
      2. some member of  $G$  is more general than  $h$
    - \* Remove from  $S$  any hypothesis that is more general than another hypothesis in  $S$

### Example Trace

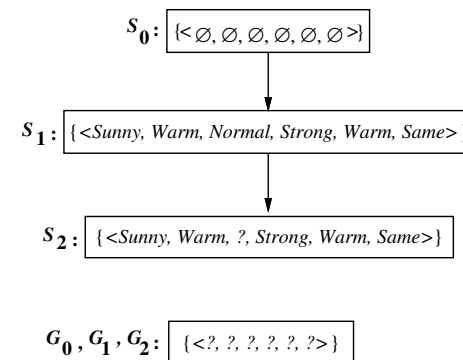
$S_0$ : {< $\emptyset$ ,  $\emptyset$ ,  $\emptyset$ ,  $\emptyset$ ,  $\emptyset$ ,  $\emptyset$ >}

$G_0$ : {<?, ?, ?, ?, ?, ?>}

## The Candidate Elimination Algorithm

- If  $d$  is a negative example
  - Remove from  $S$  any hypothesis 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
      1.  $h$  is consistent with  $d$ , and
      2. some member of  $S$  is more specific than  $h$
    - \* Remove from  $G$  any hypothesis that is less general than another hypothesis in  $G$

### Example Trace

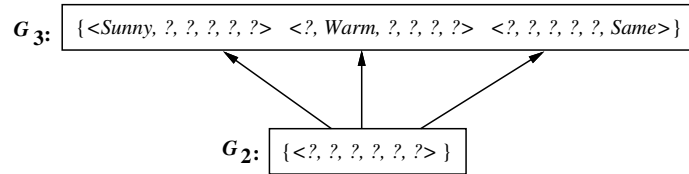


Training examples:

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

### Example Trace

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



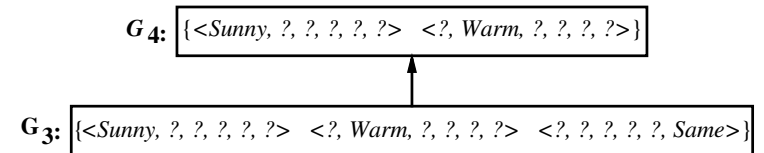
Training Example:

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

### Example Trace

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

$S_4$ : { <Sunny, Warm, ?, Strong, ?, ?> }



Training Example:

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

### Example Trace

$S_4$ : { <Sunny, Warm, ?, Strong, ?, ?> }

<Sunny, ?, ?, Strong, ?, ?> <Sunny, Warm, ?, ?, ?, ?> <?, Warm, ?, Strong, ?, ?>

$G_4$ : { <Sunny, ?, ?, ?, ?, ?>, <?, Warm, ?, ?, ?, ?> }

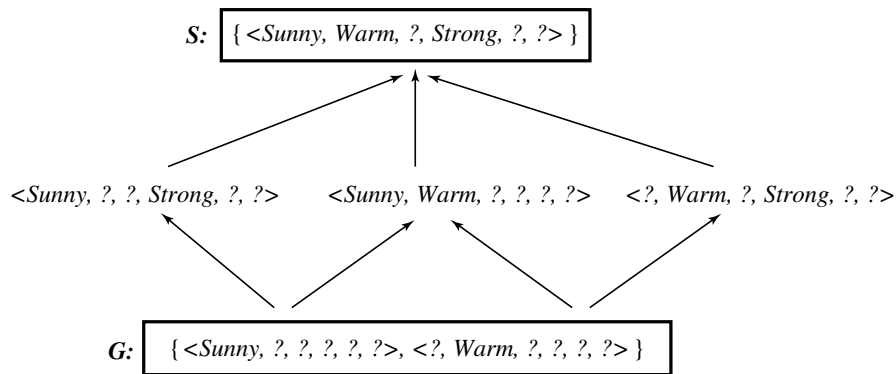
### Which Training Example Is Best To Choose Next ?

$S$ : { <Sunny, Warm, ?, Strong, ?, ?> }

<Sunny, ?, ?, Strong, ?, ?> <Sunny, Warm, ?, ?, ?, ?> <?, Warm, ?, Strong, ?, ?>

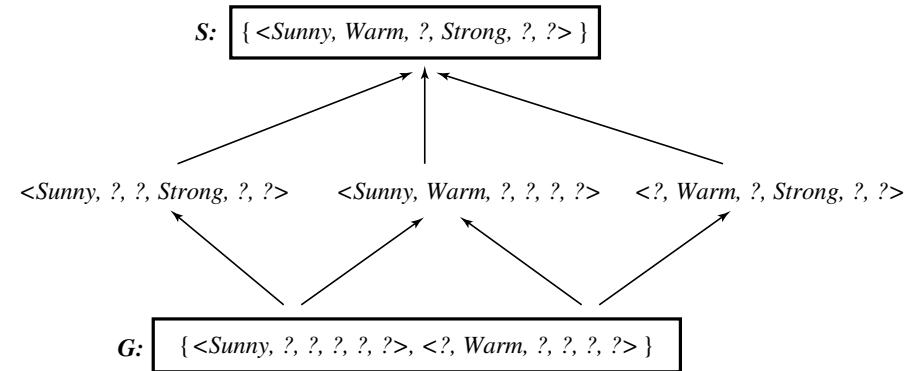
$G$ : { <Sunny, ?, ?, ?, ?, ?>, <?, Warm, ?, ?, ?, ?> }

### Which Training Example To Choose Next ?



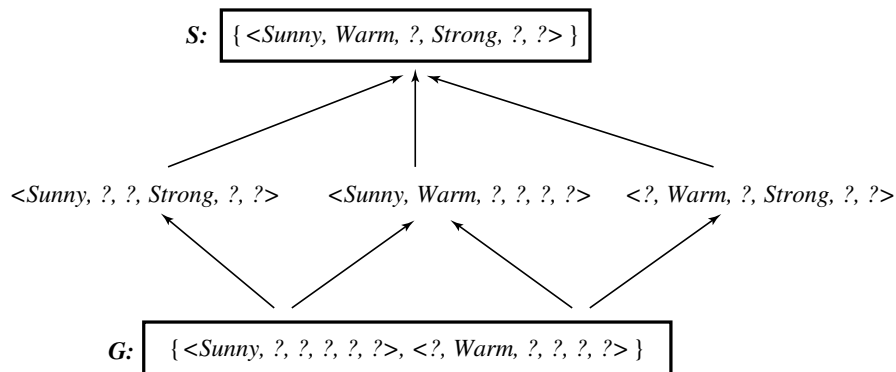
*<Sunny Warm Normal Light Warm Same>*

### How Should New Instances Be Classified ?



- < Sunny Warm Normal Strong Cool Change >*
- < Rainy Cold Normal Light Warm Same >*
- < Sunny Warm Normal Light Warm Same >*

### How Should New Instances Be Classified ?



- < Sunny Warm Normal Strong Cool Change >* (6 + /0-)
- < Rainy Cold Normal Light Warm Same >* (0 + /6-)
- < Sunny Warm Normal Light Warm Same >* (3 + /3-)

### What Justifies this Inductive Leap ?

- + *<Sunny Warm Normal Strong Cool Change>*
- + *<Sunny Warm Normal Light Warm Same>*

*S : <Sunny Warm Normal ? ? ?>*

Why believe we can classify this unseen instance ?

*<Sunny Warm Normal Strong Warm Same>*

## An UNBiased Learner

Idea: Choose  $H$  that expresses every teachable concept (i.e.  $H$  is the power set of  $X$ )

Consider  $H' =$  disjunctions, conjunctions, negations over previous  $H$ .  
E.g.

$\langle \text{Sunny Warm Normal } ? ? ? \rangle \vee \neg \langle ? ? ? ? ? \text{ Change} \rangle$

What are  $S, G$  in this case?

$S \leftarrow$

$G \leftarrow$

## Inductive Bias

Consider

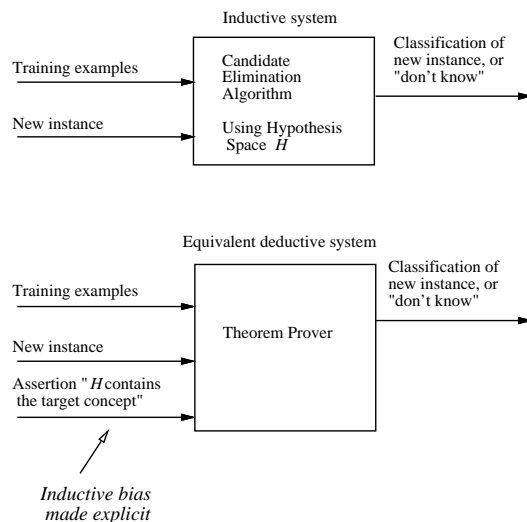
- concept learning algorithm  $L$
- instances  $X$ , target concept  $c$
- training examples  $D_c = \{ \langle x, c(x) \rangle \}$
- let  $L(x_i, D_c)$  denote the classification assigned to the instance  $x_i$  by  $L$  after training on data  $D_c$ .

**Definition:** The **inductive bias** of  $L$  is any minimal set of assertions  $B$  such that for any target concept  $c$  and corresponding training examples  $D_c$

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

where  $A \vdash B$  means  $A$  logically entails  $B$

## Inductive Systems and Equivalent Deductive Systems



## Three Learners with Different Biases

1. *Rote learner*: Store examples, Classify  $x$  iff it matches previously observed example.
2. *Version space candidate elimination algorithm*
3. *Find-S*

## Summary Points

1. Concept learning as search through  $H$
2. General-to-specific ordering over  $H$
3. Version space candidate elimination algorithm
4.  $S$  and  $G$  boundaries characterize learner's uncertainty
5. Learner can generate useful queries
6. Inductive leaps possible only if learner is biased
7. Inductive learners can be modelled by equivalent deductive systems

[Suggested reading: Mitchell, Chapter 2]