Concept Learning

Patricia J Riddle

Computer Science 367

Concept Learning

- Much of learning involves acquiring general concepts from specific training examples
- Each concept can be viewed as describing some subset of the objects or events defined over a larger set
- Alternatively each concept can be thought of as a boolean-valued function defined over this larger set
- Concept learning - inferring a boolean-valued function from training examples of its input and output

Concept Learning Example

- "Days on which my friend Aldo enjoys his favorite water sport"

Sky	Temp	Humid	Wind	Water	Forecast	Enjoy
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

Hypothesis Representation

- Conjunction of constraints on instance attributes

- Specifically, vector of six constraints with
  - ? - any value acceptable
  - Single required value (Warm)
  - 0 - no value acceptable

- Most general hypothesis <?,?,?,?,?,?>
- Most specific hypothesis <0,0,0,0,0,0>

Notation

- The set of items over which the concept is defined are called “instances” denoted by X.
- The “target concept” c: X -> {0,1}
- The “training examples” D:<x,c(x)>,
  - If c(x)=1 then positive example.
  - If c(x)=0 then negative example.

Notation II

- The problem faced by learner is to hypothesize or estimate c.
- H is the set of all possible hypotheses. H is determined by the human designers choice of hypothesis representation
- Each h: X->{0,1}
- Learners goal is to find h such that $h(x)=c(x) \forall x \in X$. (notice this is not $\forall d \in D$!!!)

Our Example

- Instances X
  - Sky (values Sunny, Cloudy, or Rainy)
  - Temp (values Warm or Cold)
  - Humidity (Normal or High)
  - Wind (Strong or Weak)
  - Water (Warm or Cool)
  - Forecast (Same or Change)
- Target Concept c: Enjoy: X -> {0,1}
- Training Examples D: see table
- Hypothesis H: conjunction of 6 constraints (?, 0, or value)

The Inductive 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 unobserved examples.

Concept Learning as Search

- Searching through a large space of hypotheses implicitly defined by the hypothesis representation (same for more general learning)

- The hypothesis representation defines the space of hypotheses the program can ever represent and therefore can ever learn

- For example, Sky has 3 possible values and Temp, Humidity, Wind, Water, and Forecast each have 2 possible values.

Size of Search Space

- X contains 3x2x2x2x2x2=96 distinct instances
- H contains 5x4x4x4x4x4=5120 ***syntactically*** distinct hypothesis. But notice any hypothesis containing one or more 0s represents the empty set of positive instances.
- Therefore H contains 1+4x3x3x3x3x3=973 ***semantically*** distinct hypothesis
- This is a very small finite hypothesis space. Most practical learning tasks have much larger or infinite hypothesis spaces.

General-to-Specific Ordering

- By taking advantage of naturally occurring structure, we can design learning algorithms that exhaustively search even infinite hypothesis spaces without explicitly enumerating every hypothesis

- For instance, general-to-specific ordering
  - h_1=<sunny,?,?,strong,?,?>
  - h_2=<sunny,?,?,?,?,?>

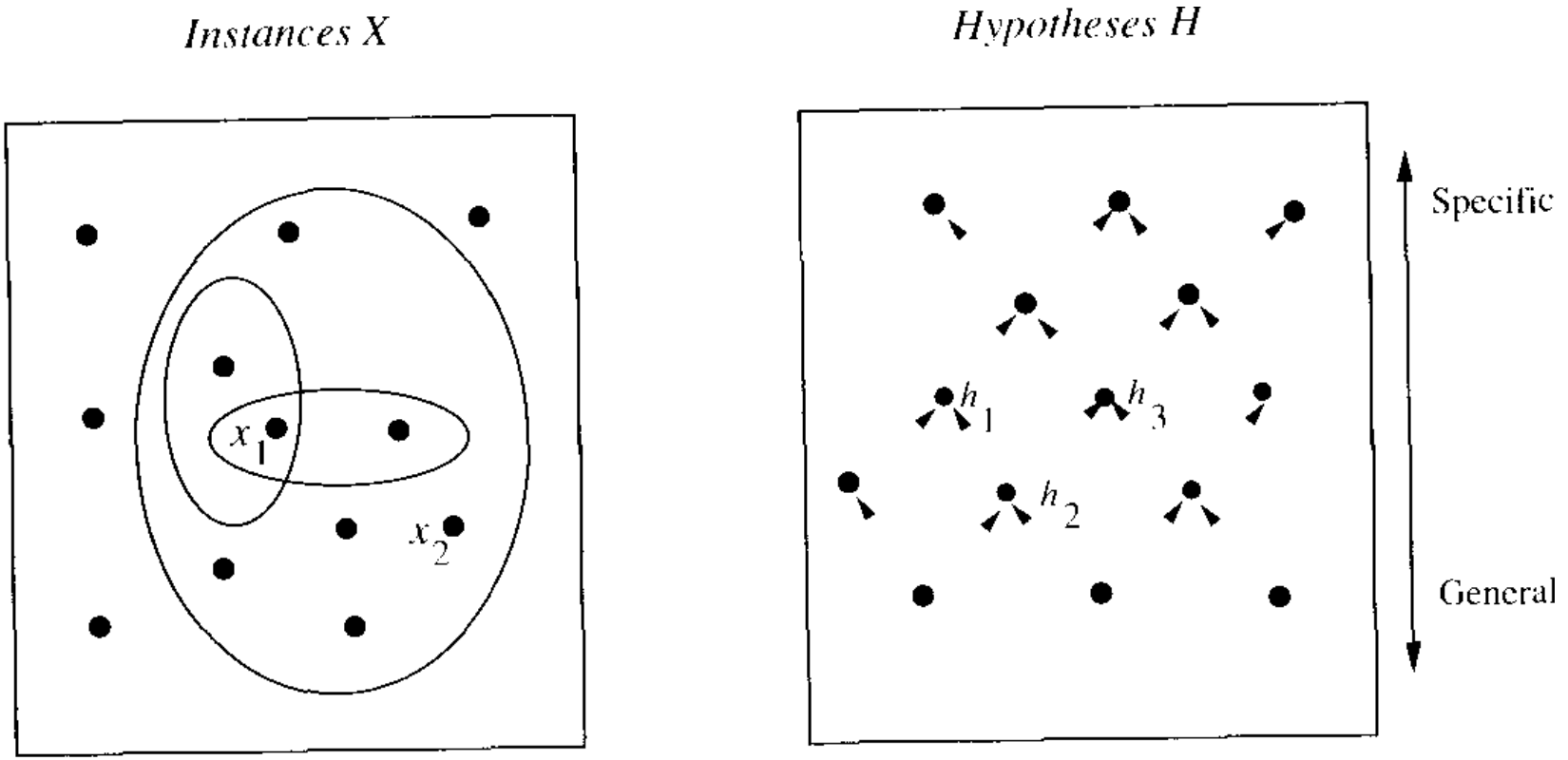
General-to-Specific Ordering II

- Any instance classified positive by h_1 will be classified positive by h_2, therefore h_2 is more general than h_1.

- 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 if and only if $(\forall x \in X)[(h_k(x)=1) \rightarrow (h_j(x)=1)]$

- More-general-than and more-specific-than are also useful

Hypothesis Search Space



x_1 = <*Sunny, Warm, High, Strong, Cool, Same*>

x_2 = <*Sunny, Warm, High, Light, Warm, Same*>

h_1 = <*Sunny, ?, ?, Strong, ?, ?*>

h_2 = <*Sunny, ?, ?, ?, ?, ?*>

h_3 = <*Sunny, ?, ?, ?, Cool, ?*>

Hypothesis Partial Ordering

- h_2 is more general than h_1
- h_2 is more general than h_3
- Neither h_1 nor h_3 is more general than the other

- More-general-than-or-equal-to defines a partial order over the hypothesis space H (reflexive, antisymmetric, and transitive)

Maximally Specific Hypothesis

- Begin with the most specific possible hypothesis in H, generalise this hypothesis each time it fails to cover an observed positive training example

  - h <- <0,0,0,0,0,0>
  - h <- <sunny,warm,normal,strong,warm,same>
  - h < - <sunny,warm,?,strong,warm,same>
  - h <- <sunny,warm,?,strong,?,?>

Maximally Specific Hypothesis II

- Find-S algorithm ignores negative examples

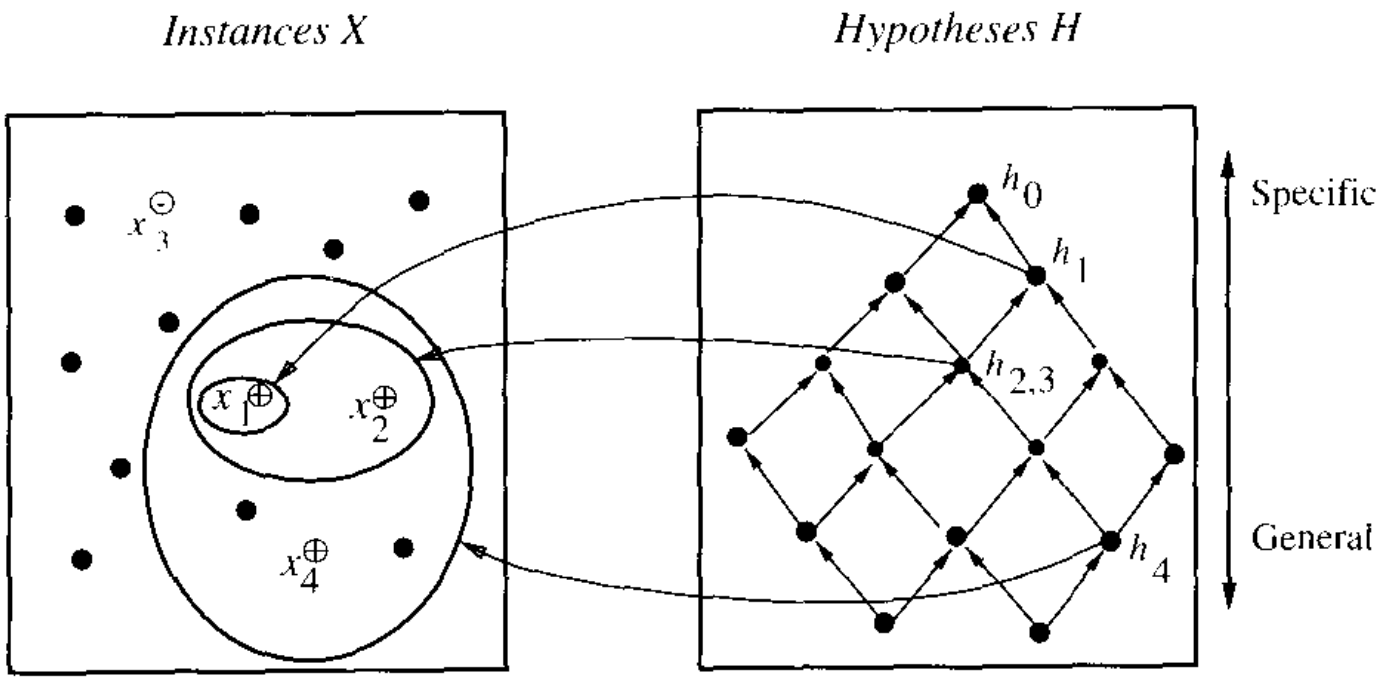
- If the hypothesis space H contains a hypothesis which describes the true target concept c & the training data contains no errors, then the current hypothesis h can never require a revision in response to a negative example - Big If

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 is satisfied by x
     Then do nothing
     Else replace a_i in h by the next more general constraint that is satisfied by x
3. Output hypothesis h

Partial Ordering



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 = <∅, ∅, ∅, ∅, ∅, ∅>

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

h_2 = <*Sunny Warm ? Strong Warm Same*>

h_3 = <*Sunny Warm ? Strong Warm Same*>

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

Questions Remain

- Has the learner converged?
- Why prefer the most specific hypothesis?
- Are training examples consistent?
- What if there are several maximally specific consistent hypothesis?

Version Spaces

- Output description of the set of all hypotheses consistent with the training examples
- Computed without explicit enumeration using more-general-than partial ordering
- A hypothesis h is consistent with a set of training examples D if and only if h(x)=c(x) for each example <x,c(x)> in D
- A version space denoted $VS_{H,D}$ with respect to hypothesis space H and training examples D is the subset of hypotheses from H consistent with the training examples in D.

List-then-Eliminate Algorithm

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 $VersionSpace$

Compact Representation for Version Spaces

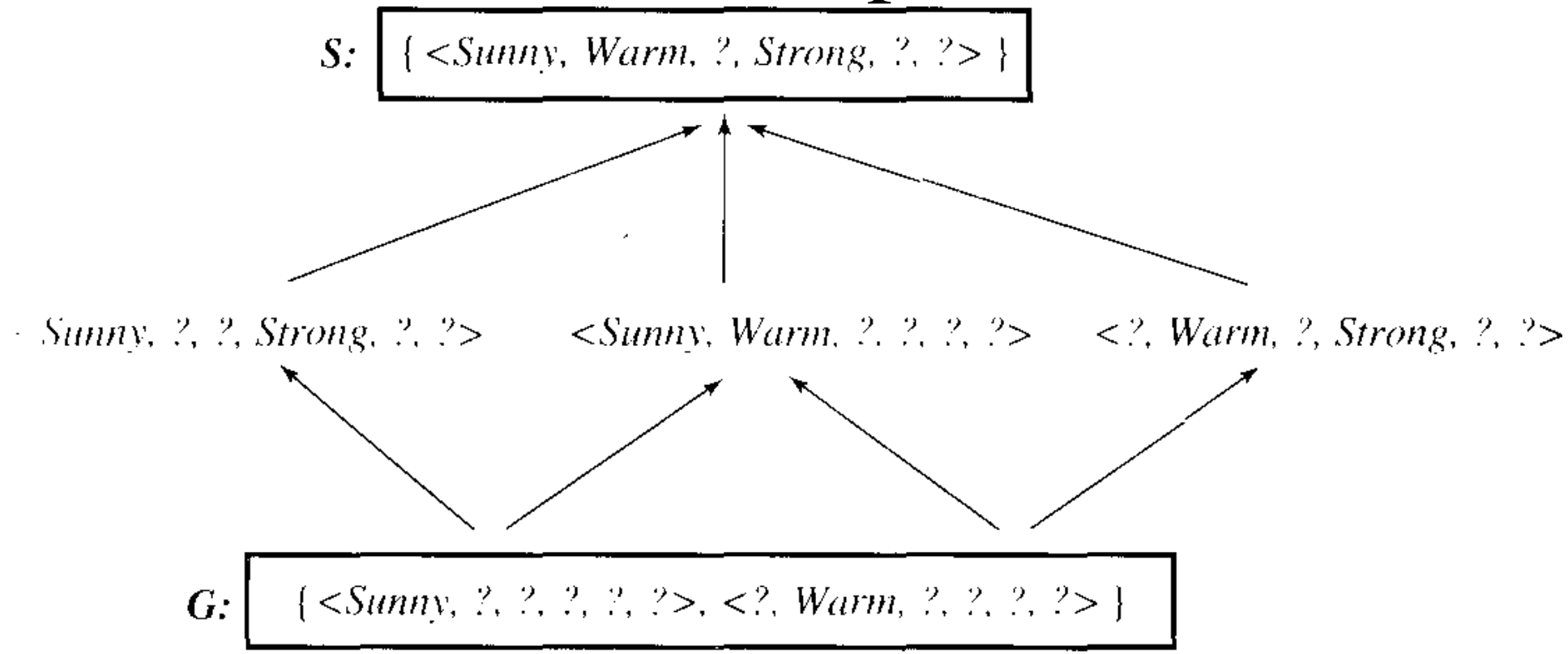


FIGURE 2.3

General Specific Boundaries

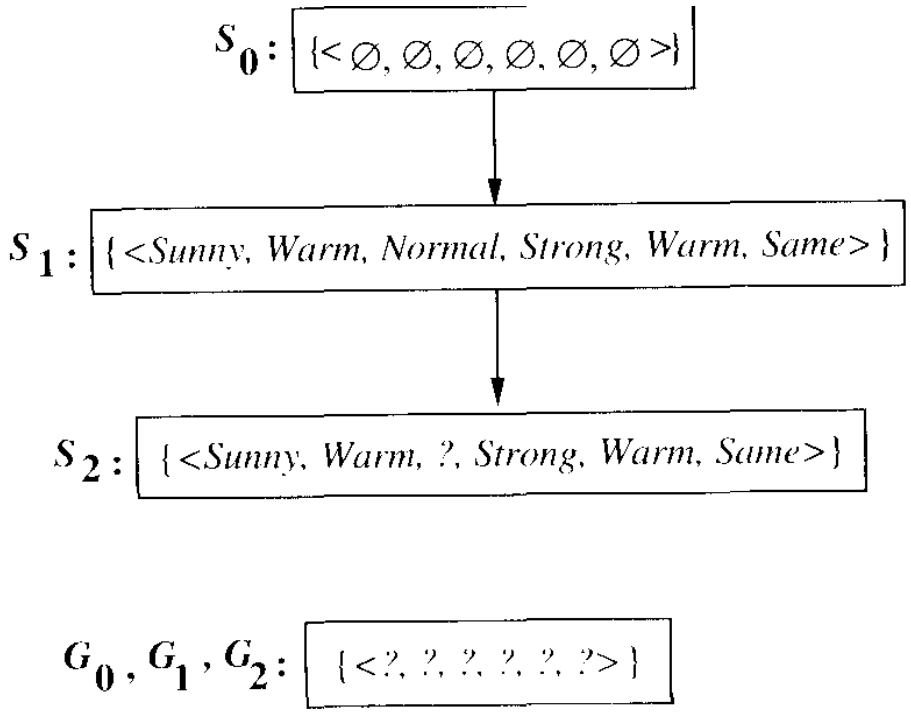
- 6 different hypotheses
- The **general boundary** G, with respect to hypothesis space H and training data D, is the set of maximally general members of H consistent with D.
- The **specific boundary** S, with respect to hypothesis space H and training data D, is the set of minimally general (I.e., maximally specific) members of H consistent with D.

Candidate Elimination Algorithm

Initialize G to the set of maximally general hypotheses in H
Initialize S to the set of 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
            - h is consistent with d, and some member of G is more general than h
        - Remove from S any hypothesis that is more general than another hypothesis in S
- 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
            - h is consistent with d, and some member of S is more specific than h
        - Remove from G any hypothesis that is less general than another hypothesis in G

Training Examples 1 & 2



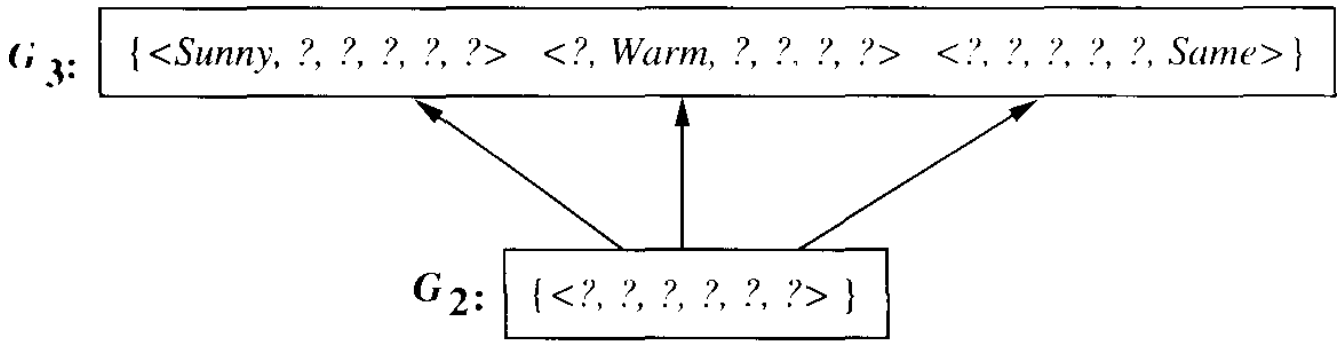
Training examples:

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

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

Training Example 3

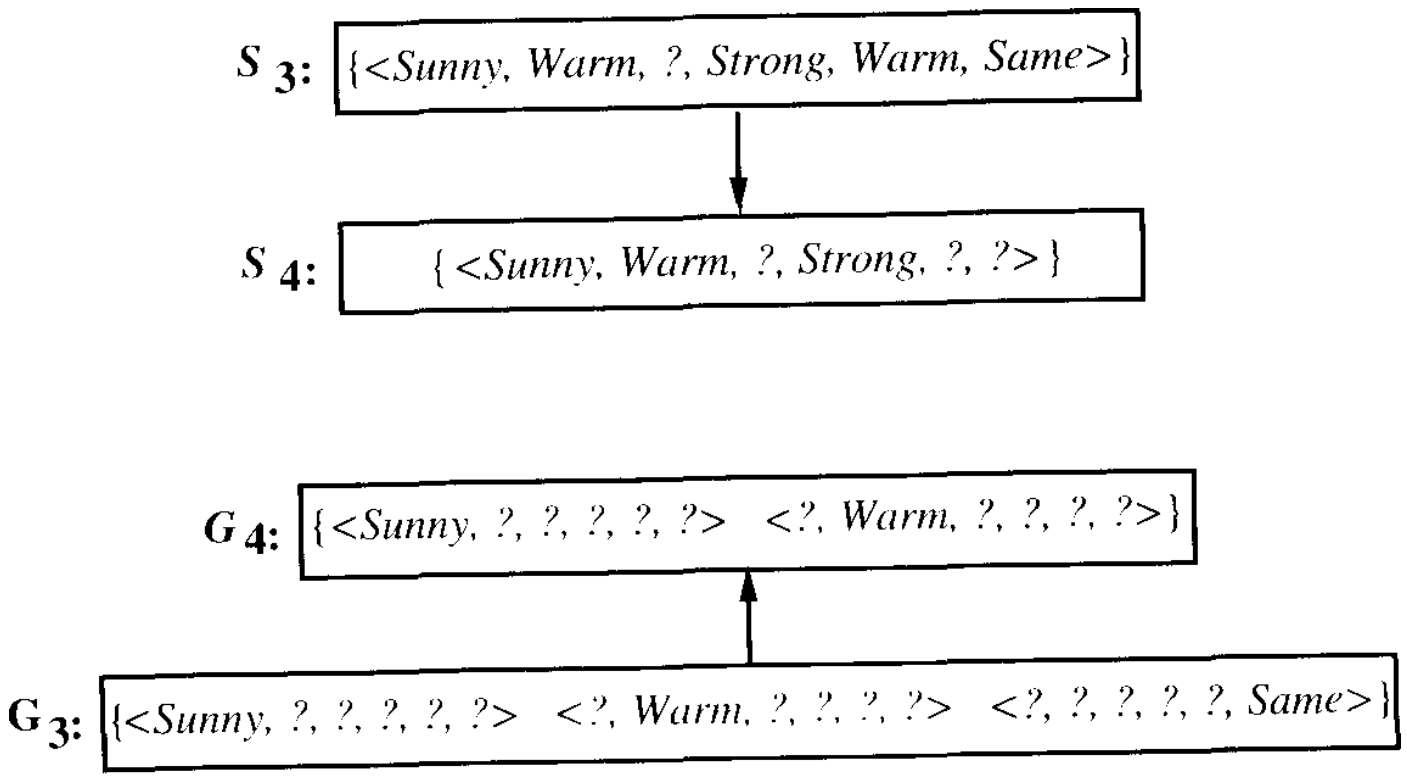
S_2, S_3: { <*Sunny, Warm, ?, Strong, Warm, Same*> }



Training Example:

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

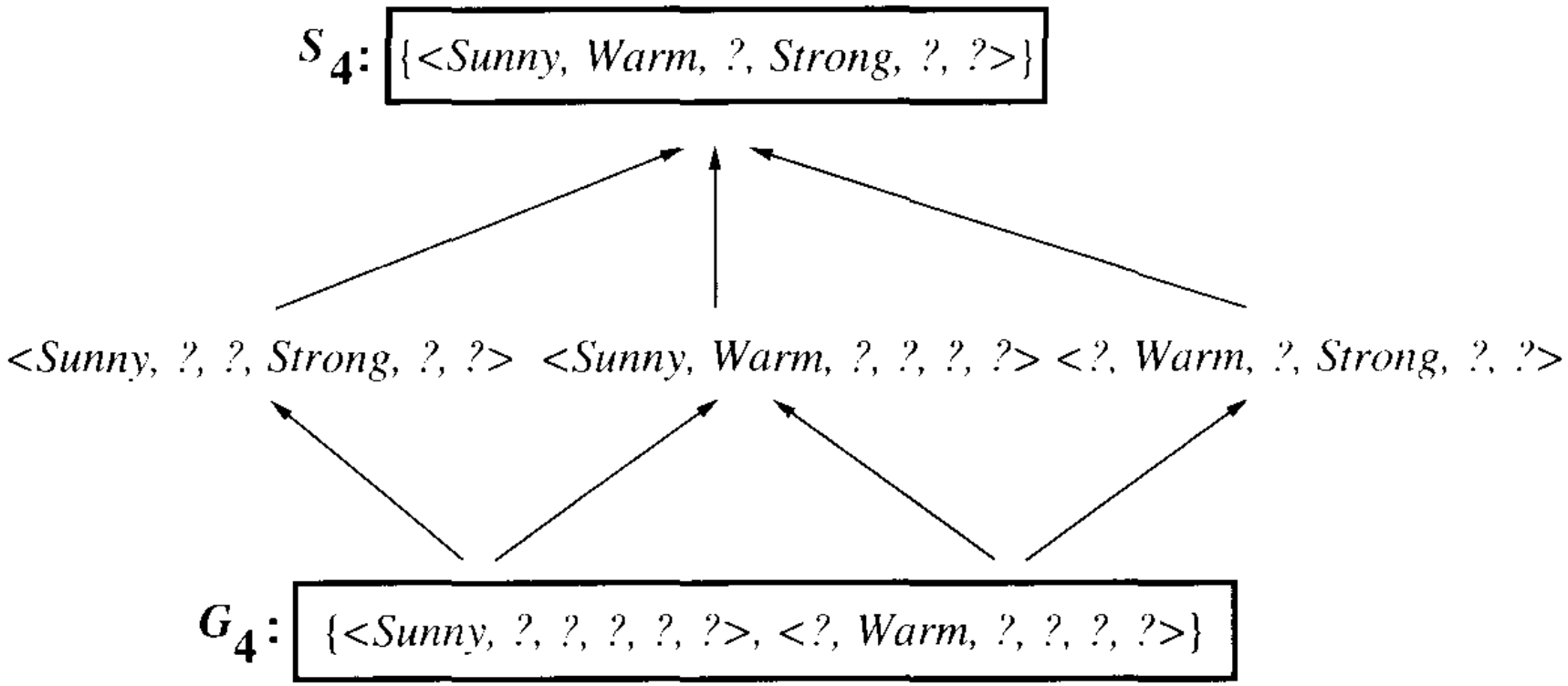
Training Example 4



Training Example:

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

Final Version Space



What if first instance is negative?

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

   - S_1=<0,0,0,0,0,0>
   - G_1={<Sunny,?,?,?,?,?>, <Cloudy,?,?,?,?,?>,
     <?,Warm,?,?,?,?>, <?,?,Normal,?,?,?>,
     <?,?,?,Light,?,?>, <?,?,?,?,Cool,?>,
     <?,?,?,?,?,Same>}

Singular S sets

- Why try to remove any hypothesis that is inconsistent from the S set?

- Caused by conjunctive representation

Version Spaces with Disjuncts

S_0=<0,0,0,0,0,0>

G_0=<?,?,?,?,?,?>

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

S_1=<Sunny,Warm,Normal,Strong,Warm,Same>

G_1=<?,?,?,?,?,?>

VS with Disjuncts II

2 <Sunny,Warm,High,Strong,Warm,Same>,
EnjoySport=Yes

S_2=<Sunny,Warm,Normal,Strong,Warm,Same>
∨ <Sunny,Warm,High,Strong,Warm,Same>

G_2=<?,?,?,?,?,?>

VS with Disjuncts III

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

S_3=<Sunny,Warm,Normal,Strong,Warm,Same> ∨
<Sunny,Warm,High,Strong,Warm,Same>

G_3={<Sunny,?,?,?,?,?> ∨ <?,Warm,?,?,?,?>,
<Sunny,?,?,?,?,?> ∨ <?,?,?,?,?,Same>,
<?,Warm,?,?,?,?> ∨ <?,?,?,?,?,Same>}

VS with Disjuncts IV

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

S_4={<Sunny,Warm,?,Strong,?,?> ∨
<Sunny,Warm,High,Strong,Warm,Same>,
<Sunny,Warm,Normal,Strong,Warm,Same> ∨
<Sunny,Warm,High,Strong,?,?>}

G_4={<Sunny,?,?,?,?,?> ∨ <?,Warm,?,?,?,?>,
<Sunny,?,?,?,?,?> ∨ <?,?,?,?,?,Same>,
<?,Warm,?,?,?,?> ∨ <?,?,?,?,?,Same>}

VS with Disjuncts V

5 <Sunny,Warm,Normal,Strong,Cool,Change>, EnjoySport=No

S_5 = <Sunny,Warm,Normal,Strong,Warm,Same> ∨ <Sunny,Warm,High,Strong,?,?>

G_5 ={<?,Warm,?,?,?,Same> ∨ <?,?,?,?,?,Same>,
<?,Warm,?,?,Warm,?> ∨ <?,?,?,?,?,Same>,
<?,Warm,High,?,?,?> ∨<?,?,?,?,?,Same>,
<Sunny,?,High,?,?,?> ∨ <?,?,?,?,?,Same>,
<Sunny,?,?,?,Warm,?> ∨ <?,?,?,?,?,Same>,
<Sunny,?,?,?,?,Same> ∨ <?,?,?,?,?,Same>,
<Sunny,?,High,?,?,?> ∨ <?,Warm,High,?,?,?>,
<Sunny,?,?,?,Warm,?> ∨ <?,Warm,High,?,?,?>, <Sunny, ? High,?,?,?> ∨ <?,Warm,?,?,Warm,?>}

Properties of Candidate-Elimination Algorithm

- Independent of the order in which the training data is presented
- S and G boundaries move monotonically closer to each other
- Will converge if
  1. There are no errors in the training examples
  2. There is some hypothesis in H that correctly describes the target concept
- Can determine when sufficient training examples have been observed to converge, S and G are identical
- Can detect errors or bad representation by convergence to the empty version space

Requesting Training Examples

<Sunny,Warm,Normal,Light,Warm,Same>

- Generate instances that satisfy 1/2 the hypotheses
- Correct target concept found in $\lceil \log_2 |VS| \rceil$ experiments

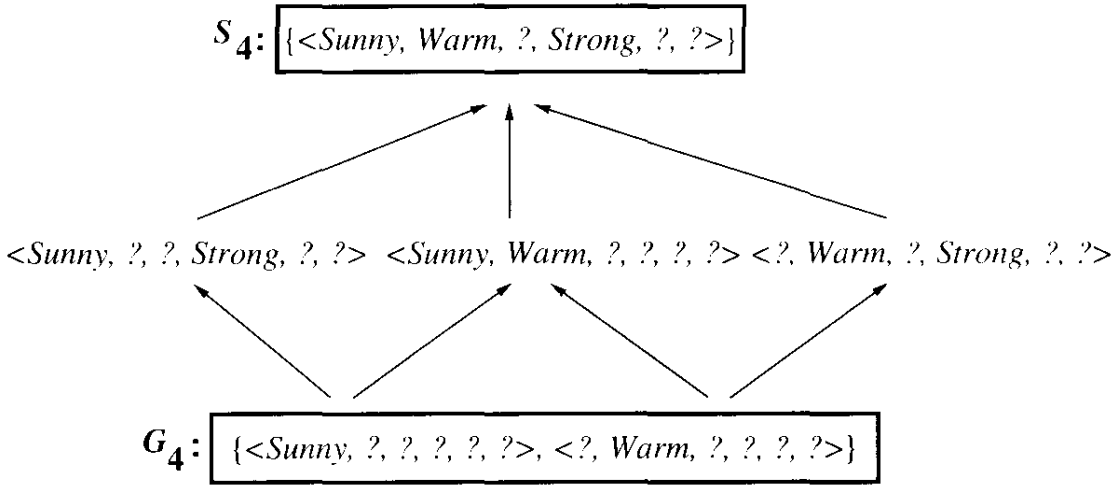
- This is not always possible!

Partially Learned Concepts

- What if run out of training data before convergence?
- Can still classify new data!!

- New instance will be classified as positive by all the hypotheses if and only if the instance satisfies every member of S
- New instance will be classified as negative by all the hypotheses if and only if the instance satisfies none of the members of G
- Can use voting if not equally split

Classifying with Partially Learned Concepts



Instance	Sky	Temp	Humidity	Wind	Water	Forecast	EnjoySport
A	Sunny	Warm	Normal	Strong	Cool	Change	?
B	Rainy	Cold	Normal	Light	Warm	Same	?
C	Sunny	Warm	Normal	Light	Warm	Same	?
D	Sunny	Cold	Normal	Strong	Warm	Same	?

Inductive Bias

- What if the target concept is not in the hypothesis space?
- Use a hypothesis space that includes every possible hypothesis!!!
- Does the size of this space influence the ability to generalize to unobserved instances?
- Does it influence the number of training examples that must be observed?

An Unbiased Learner

- Can't represent "Sky = Sunny or Sky = Cloudy"
- Provide a hypothesis space capable of representing ***every teachable concept*** - power set of X - set of all subsets
- Instance space = 96, power set = $2^{96} \approx 10^{28}$
- Can allow arbitrary disjunctions

- Now **completely unable to generalise** beyond the observed examples
- Can't even use voting - unobserved instance always divide space in half

Futility of Bias-Free Learning

- A learner that makes no *a priori* assumptions regarding the identity of the target concept has no rational basis for classifying any unseen instances. (just a database - rote learning)

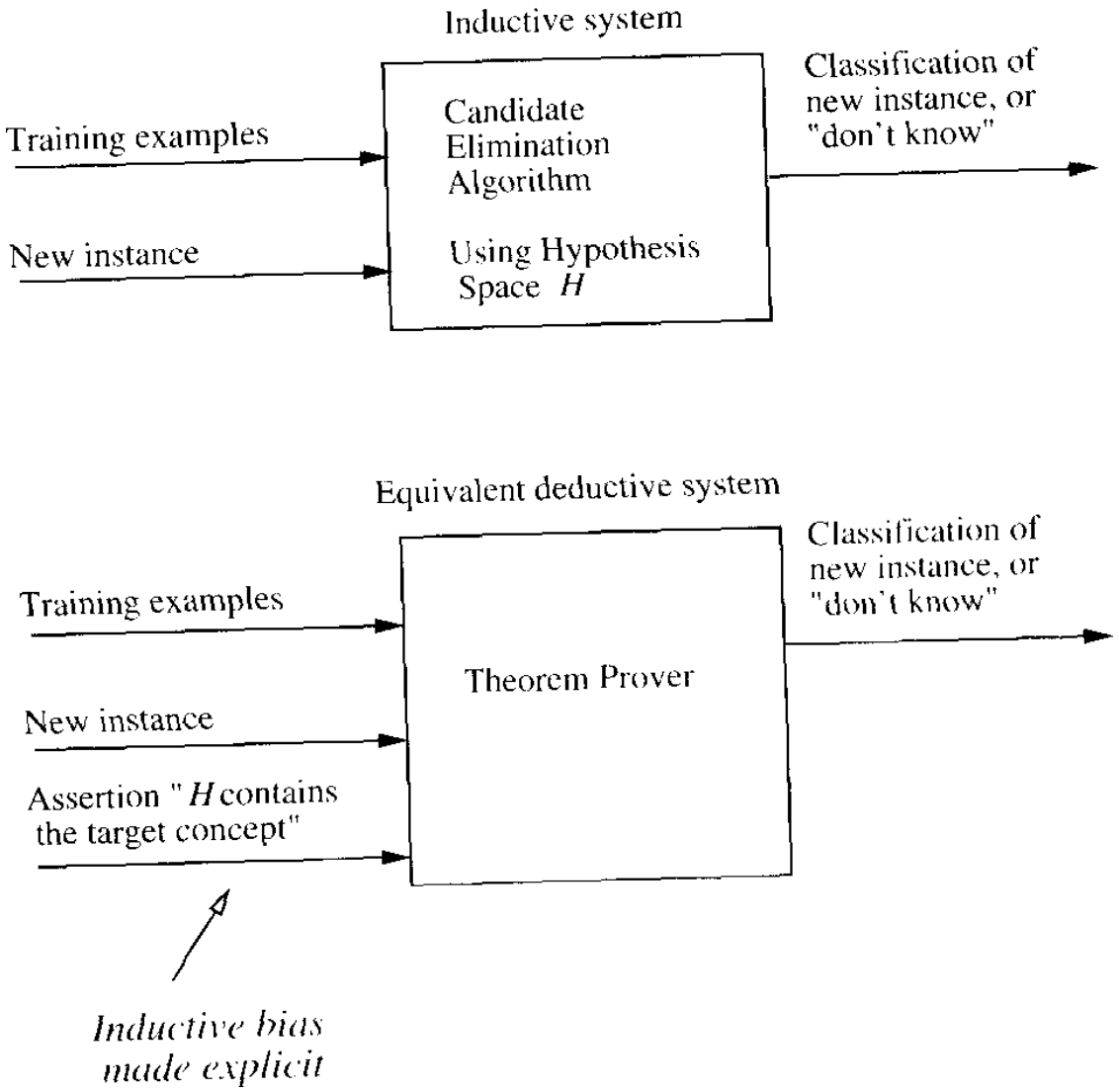
- Candidate-Elimination algorithm was able to generalise beyond the observed training examples because it was biased by the implicit assumption that the target could be represented as a conjunction of target values

Inductive Bias

- Consider a concept learning algorithm L for the set of instances X.
  - Let c be an arbitrary concept defined over X, and let D_c = {<x,c(x)>} be an arbitrary set of training examples of c.
  - Let $L(x_i,D_c)$ denote the classification assigned to the instance x_i by L after training on the data D_c.
- 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 $(\forall x_i \in X)[(B \wedge D_c \wedge x_i) \vdash L(x_i,D_c)]$

- Inductive bias of the Candidate-Elimination algorithm: The target concept c is contained in the given hypothesis space H.

Inductive versus Deductive



Inductive Biases of Algorithms

- Rote Learner - no inductive bias
- Candidate Elimination - the target concept can be represented in its hypothesis space - can classify some instances that the Rote Learner will not.
- Find-S - in addition that all instances are negative instances until the opposite is entailed by its other knowledge.

- More strongly biased methods make more inductive leaps - Is this good or bad??

Summary

- Concept learning can be seen as search.
- General-to-Specific partial ordering of hypotheses can be used to organize search
- Find-S and Candidate-Elimination algorithms
- Inductive learning algorithms are able to classify unseen examples only because of their implicit inductive bias for selecting one consistent hypothesis over another
- An unbiased learner cannot make inductive leaps to classify unseen examples.