Machine learning
III B.Tech II Semester CSE JNTUH
CS601PC

J.Kamal Vijetha
Assistant Professor
KMIT

Concept Learning

- Concept learning
  - supervised, eager learning
  - target problem: whether something belongs to the target concept or not
  - target function: $V: D \longrightarrow \{\text{true, false}\}$ where D is positive and negative examples of target function

- Underlying idea: problem inducing is Humans acquire general concepts from specific examples
  (e.g., concepts: living organism, beauty, computer, well-fitting-shoes)
  (note: each concept can be thought of as Boolean-valued function)

- **Concept learning is inferring a Boolean-valued function from training data**
  - ➢ **concept learning is the prototype binary classification**

8.Concept Learning Task – Training Examples for Enjoy Sport

- Concept learning is inferring a Boolean-valued function from training data
- concept learning is the prototype binary classification
- attribute Ex: **Enjoy water sport or not**
- **Task: *To predict enjoy sport*** on a arbitrary day based on other attributes

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

Representation of Hypotheses

Many possible representations

Here, ***h*** **is conjunction of constraints** on attributes .

Eg: Each constraint can be

- a specific value (e.g., *Water* = *Warm*)
- don't care (e.g., "*Water* =?") acceptable
- no value Acceptable(e.g., "Water=φ")

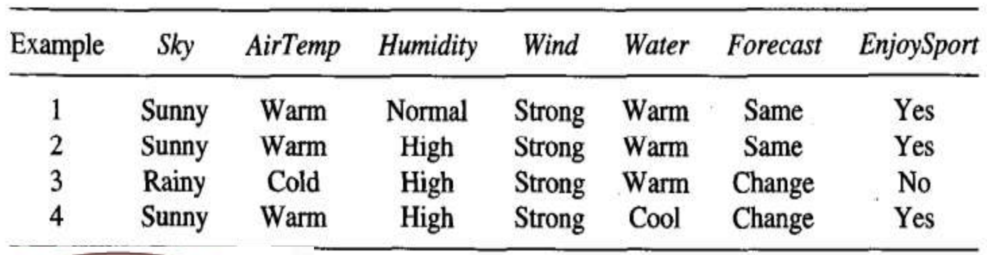
For example,

Sky	AirTemp	Humid	Wind	Water	Forecast
<*Sunny*	?	?	*Strong*	?	*Same*>

Notation

- Instances X: Possible days, each described by the attributes *Sky, AirTemp, Humidity, Wind, Water, Forecast*
- Target function *c*: *EnjoySport* : X → {0, 1}
- Hypotheses *H*: Conjunctions of literals. E.g.

  <?, *Cold*, *High*, ?, ?, ?>.



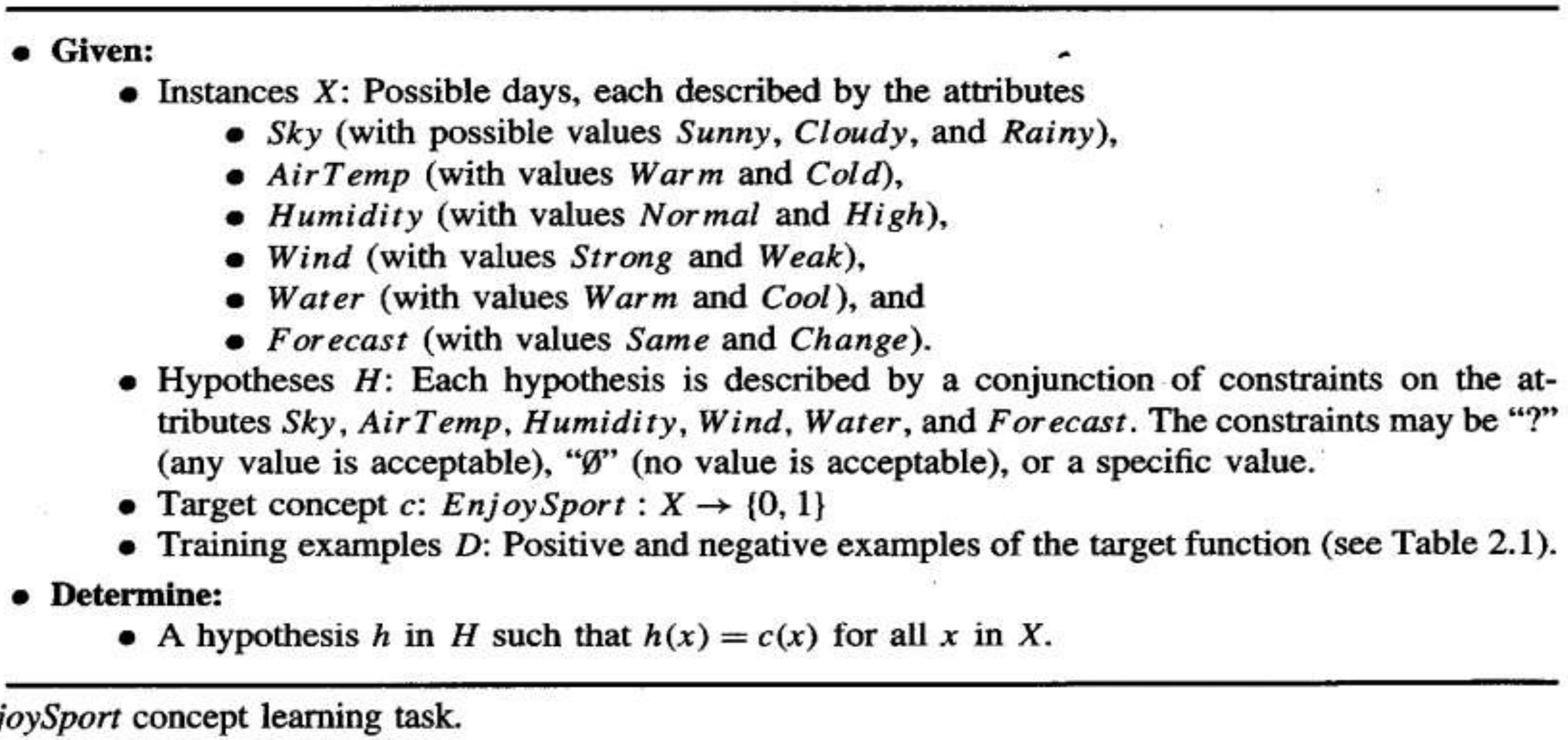
Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

Every day Tom his enjoy i.e., Only positive examples.

Most General Hypothesis: h = <?, ?, ?, ?, ?, ?>

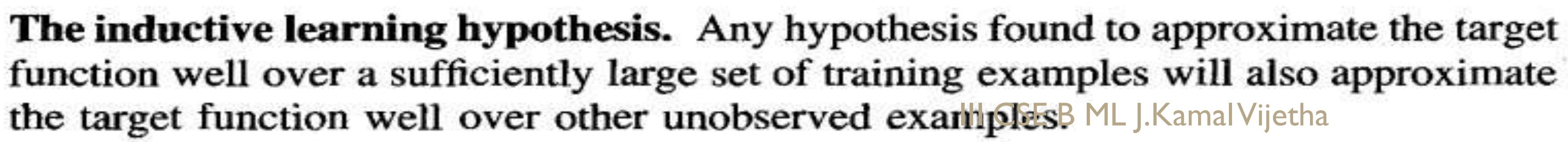
Most Specific Hypothesis: h = < Ø, Ø, Ø, Ø, Ø, Ø>

Task



- **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), "Ø" (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 (see Table 2.1).
- **Determine:**
  - A hypothesis *h* in *H* such that $h(x) = c(x)$ for all *x* in *X*.

The *EnjoySport* concept learning task.



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.

Summary of Concept Learning

- The problem of inducing **general functions from specific training examples** is central to learning .
- Acquiring the definition of a general category given a sample of positive and negative training examples of the category is concept learning.
- It can be formulated as a problem of searching through a predefined space of potential hypotheses for the hypothesis that best fits the training examples
- The problem of automatically inferring the general definition of some concept, given examples labeled as **members** or **non members** of the concept. This task is commonly referred to as concept learning.
- In general, each hypothesis h in H represents a boolean-valued function defined over X; that is, h : X → {0,1). **The goal of the learner is to find a hypothesis h such that h(x) = c(x) for all x in X.**

  In the current example, the target concept corresponds to the value of the attribute Enjoy Sport (i.e., **c(x) = 1** if EnjoySport = Yes, and **c(x) = 0** if EnjoySport = No).
- Usually H is determined by the human designer's choice of hypothesis representation. H to denote the set of all possible hypotheses.

9. 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.

•The goal of this **SEARCH is to find the hypothesis** which best fits the training examples.

- •*Example* ***learning Task*** *Enjoy sport :* ***Instance X and Hypotheses H***
- •*The attributes given as sky has 3 possible values and others have 5attributes have 2 possible values,* ***instance space*** *is 3.2.2.2.2.2=96-----→distinct instances*
- • *number of instances of hypothesis(?) are 5.4.4.4.4.4=5120--→syntactically instances.*
- •*For one or more φ is a empty set of instances classifies as negative ,*
  hypothesis is 1+(4.3.3.3.3.3)=973.------→ semantically distinct hypothesis
- •*Enjoy sport is a simple Task with* ***finite*** *hypothesis space ,*
- •*But practically we have a* ***infinite*** *hypotheses space*

• Most practical learning tasks involve much larger, sometimes infinite, hypothesis spaces.

•Learning as a **search problem**, study of learning ,different strategies for searching the hypothesis space.

•Particular interested in algorithms capable of efficiently searching very large or infinite hypothesis spaces, to find the hypotheses that best fit the training data.

General-to-Specific Ordering

- Many concept learning algorithms utilize general-to-specific ordering of hypotheses
- To illustrate General-to-Specific Ordering:
- *Set of instances are classified positive*

$h_1 = \langle Sunny, ?, ?, Strong, ?, ? \rangle$

$h_2 = \langle Sunny, ?, ?, ?, ?, ? \rangle$

eg: *Both hypotheses are giving positive result then H2 is more general than H1.*

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

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

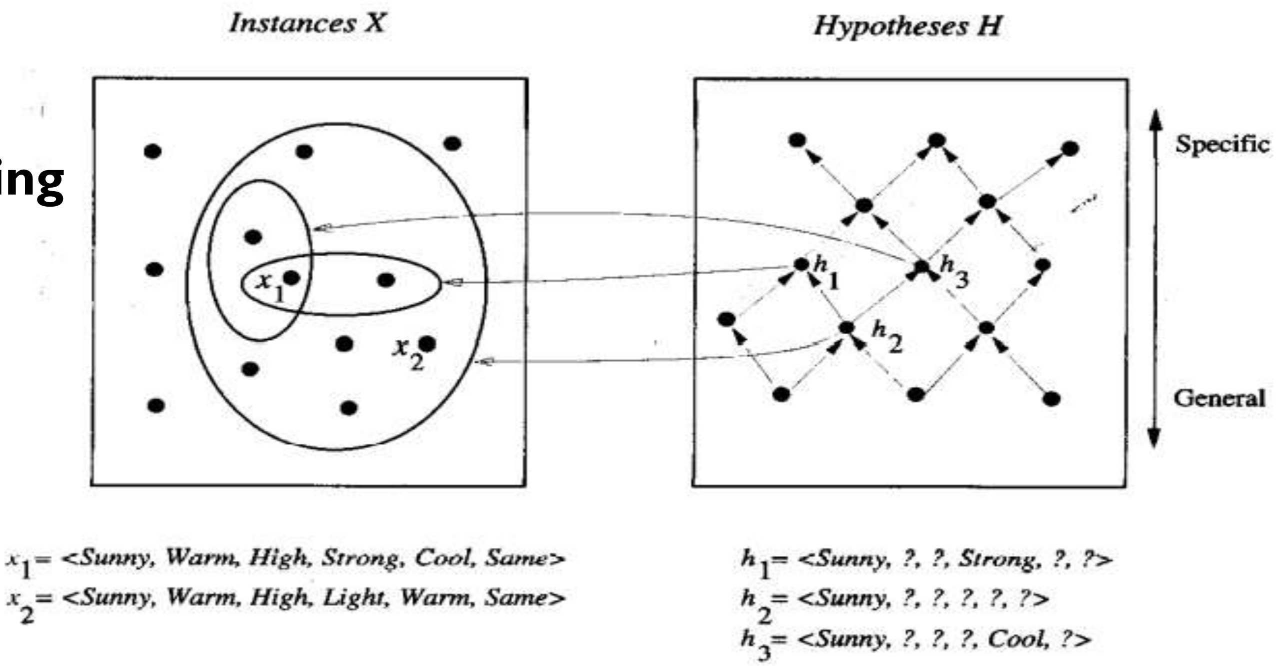
Example

3 Hypothesis

2 instances---- training

$\geq_g$

Is relation defines a partial order over the hypothesis space H



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, ?>

9. Concept Learning as Search

- Concept learning task:
  - target concept: Girls who Simon likes
  - target function: c: $D \rightarrow \{0, 1\}$
  - data $d \in D$: Girls, each described in terms of the following attributes
    - $a_1 \equiv$ *Hair* (possible values: blond, brown, black) + '?'
    - $a_2 \equiv$ *Body* (possible values: thin, normal, plump)
    - $a_3 \equiv$ *likesSimon* (possible values: yes, no) + '?'
    - $a_4 \equiv$ *Pose* (possible values: arrogant, natural, goofy)
    - $a_5 \equiv$ *Smile* (possible values: none, pleasant, toothy) + '?'
    - $a_6 \equiv$ *Smart* (possible values: yes, no) + '?'
  - target f-on representation: $h \equiv c'$: $\langle a_1, a_2, a_3, a_4, a_5, a_6 \rangle \rightarrow \{0, 1\}$
  - training examples D: positive and negative examples of target function c

instances

$|H| = 1 + 4 \cdot 4 \cdot 3 \cdot 4 \cdot 4 \cdot 3 = 2305$

$h \equiv \langle \varphi, \varphi, \varphi, \varphi, \varphi, \varphi \rangle$

error rate

- **Aim**: Find a hypothesis $h \in H$ such that $(\forall d \in D)\ h(d) - c(d) < \varepsilon \approx 0$, where H is the set of all possible hypotheses $h \equiv \langle a_1, a_2, a_3, a_4, a_5, a_6 \rangle$, where each a_k, $k = [1..6]$, may be '?' ($\equiv$ any value is acceptable), 'φ' ($\equiv$ no value is acceptable), or a specific value.

concept learning $\equiv$ searching through H

10.Find-S finding Maximally Specific Hypothesis

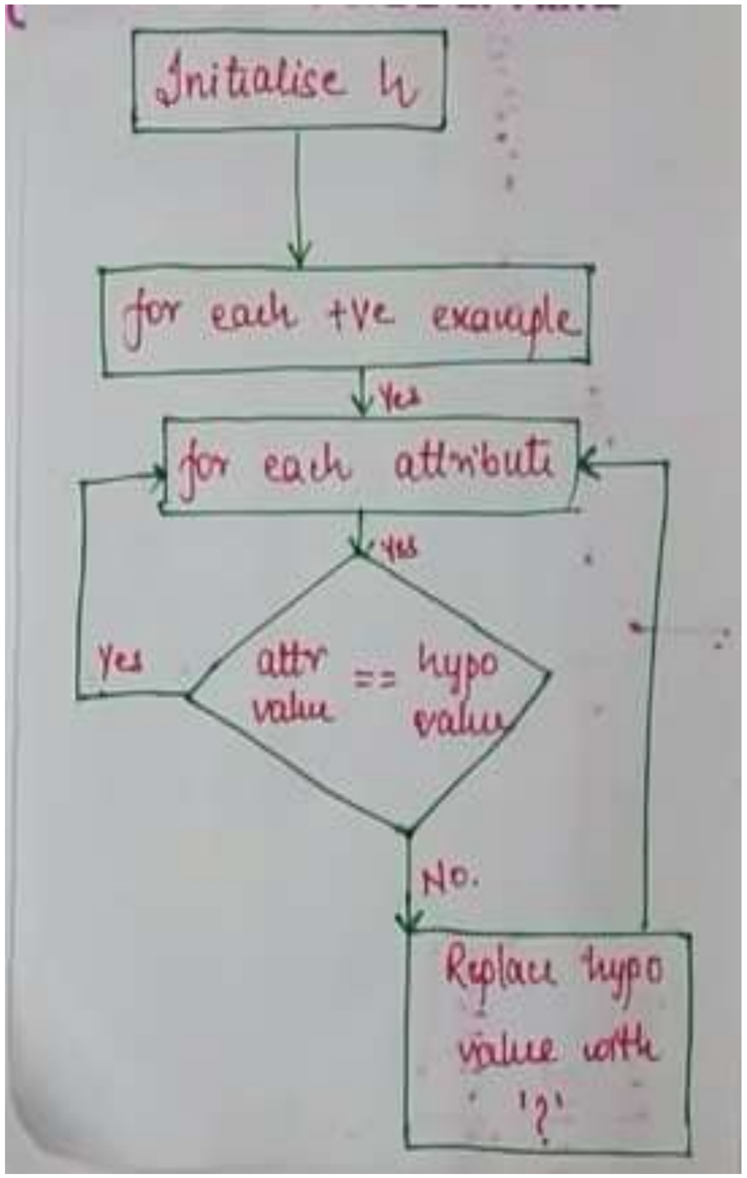
The more-general-than partial ordering to organize the search for a hypothesis consistent with the observed training examples?

One way is to begin with the most specific possible hypothesis in H, then generalize this hypothesis each time it fails to cover an observed positive training example.

(We say that a hypothesis "covers" a positive example if it correctly classifies the example as positive.)

To be more precise about how the partial ordering is used, consider the FIND-S algorithm defined below

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*
3. Output hypothesis *h*



Hypothesis Space Search by Find-S

The first step of FIND- S is to initialize h to the most specific hypothesis in H

$$h \leftarrow \langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle$$

Step 2: each is replaced by the next more general constraint that fits:

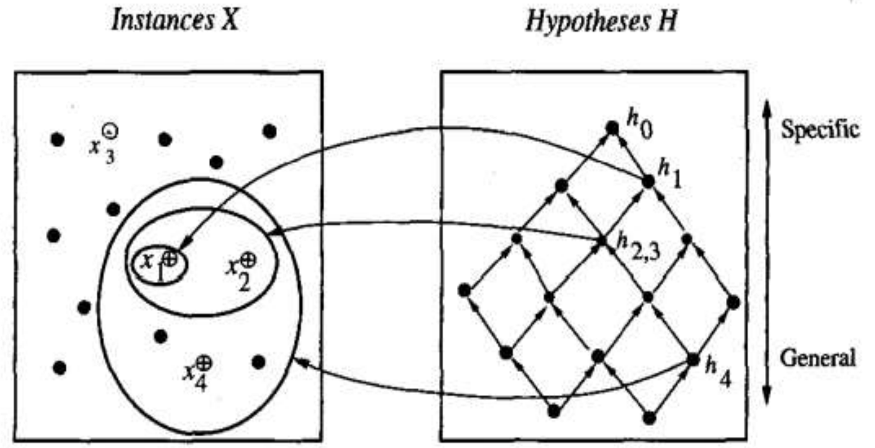
$$h \leftarrow \langle Sunny, Warm, Normal, Strong, Warm, Same \rangle$$

Step 3:The refined hypothesis in this case is

$$h \leftarrow \langle Sunny, Warm, ?, Strong, Warm, Same \rangle$$

Step 4: trace of FIND-S, the fourth (positive)

$$h \leftarrow \langle Sunny, Warm, ?, Strong, ?, ? \rangle$$



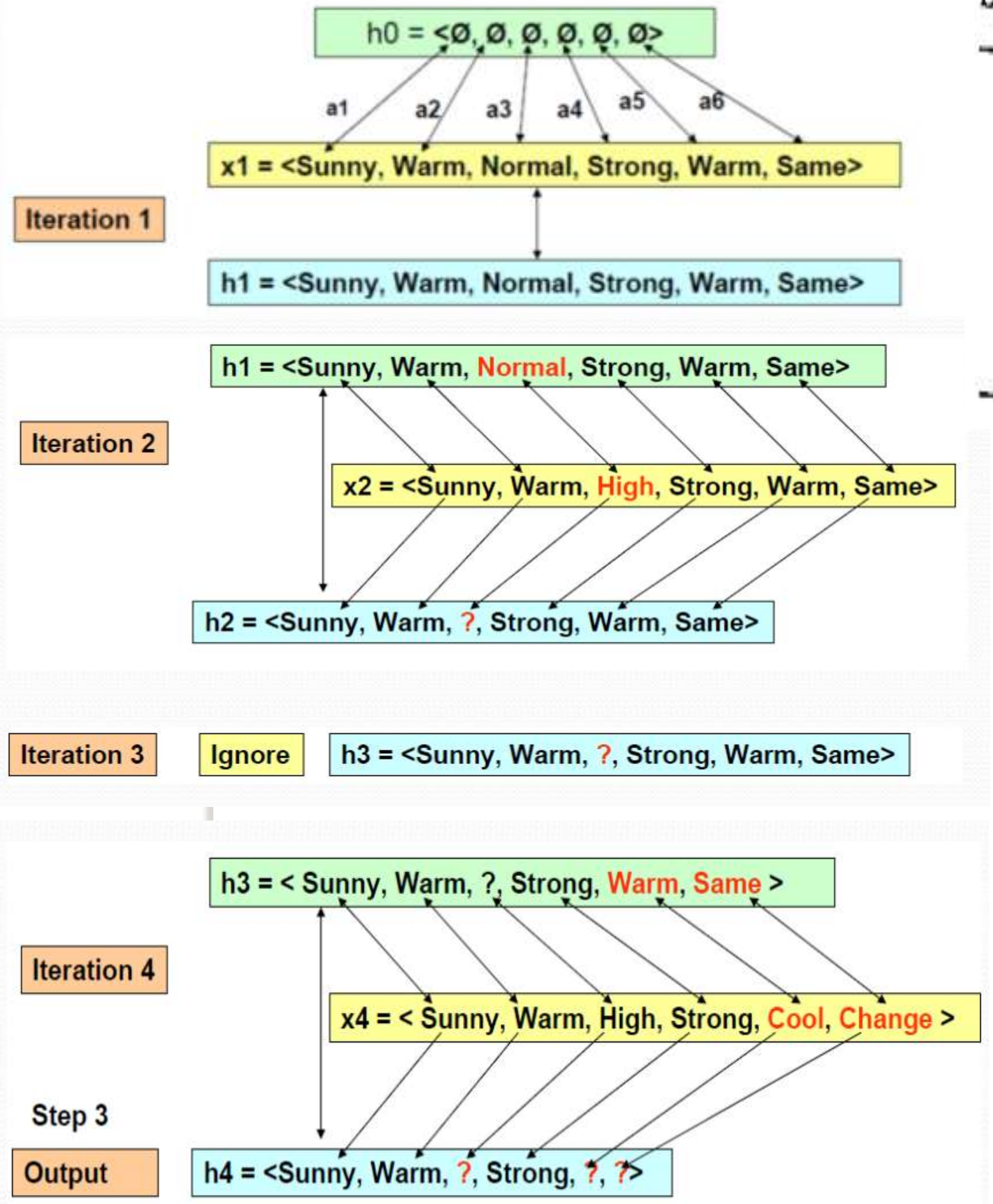
	h_0 = <∅, ∅, ∅, ∅, ∅, ∅>
x_1 = <Sunny Warm Normal Strong Warm Same>, +	h_1 = <Sunny Warm Normal Strong Warm Same>
x_2 = <Sunny Warm High Strong Warm Same>, +	h_2 = <Sunny Warm ? Strong Warm Same>
x_3 = <Rainy Cold High Strong Warm Change>, -	h_3 = <Sunny Warm ? Strong Warm Same>
x_4 = <Sunny Warm High Strong Cool Change>, +	h_4 = <Sunny Warm ? Strong ? ? >

The key property of the FIND-S algorithm is that for hypothesis spaces described by conjunctions of attribute constraints (such as H for the EnjoySport task),

FIND-S is guaranteed to output the most specific hypothesis within H that is consistent with the positive training examples.

•Has the learner converged to the correct target concept?
•Why prefer the most specific hypothesis?
•Are the training examples consistent?

FIND-S :


h0 = <Ø, Ø, Ø, Ø, Ø, Ø>
a1
a2
a3
a4
a5
a6
x1 = <Sunny, Warm, Normal, Strong, Warm, Same>
Iteration 1
h1 = <Sunny, Warm, Normal, Strong, Warm, Same>
h1 = <Sunny, Warm, Normal, Strong, Warm, Same>
Iteration 2
x2 = <Sunny, Warm, High, Strong, Warm, Same>
h2 = <Sunny, Warm, ?, Strong, Warm, Same>
Iteration 3
Ignore
h3 = <Sunny, Warm, ?, Strong, Warm, Same>
h3 = < Sunny, Warm, ?, Strong, Warm, Same >
Iteration 4
x4 = < Sunny, Warm, High, Strong, Cool, Change >
Step 3
Output
h4 = <Sunny, Warm, ?, Strong, ?, ?>

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

Complaints about Find-S

- Can’t tell whether it has learned concept
- Can’t tell when training data inconsistent
- Picks a maximally specific *h* (why?)
- Depending on *H*, there might be several!

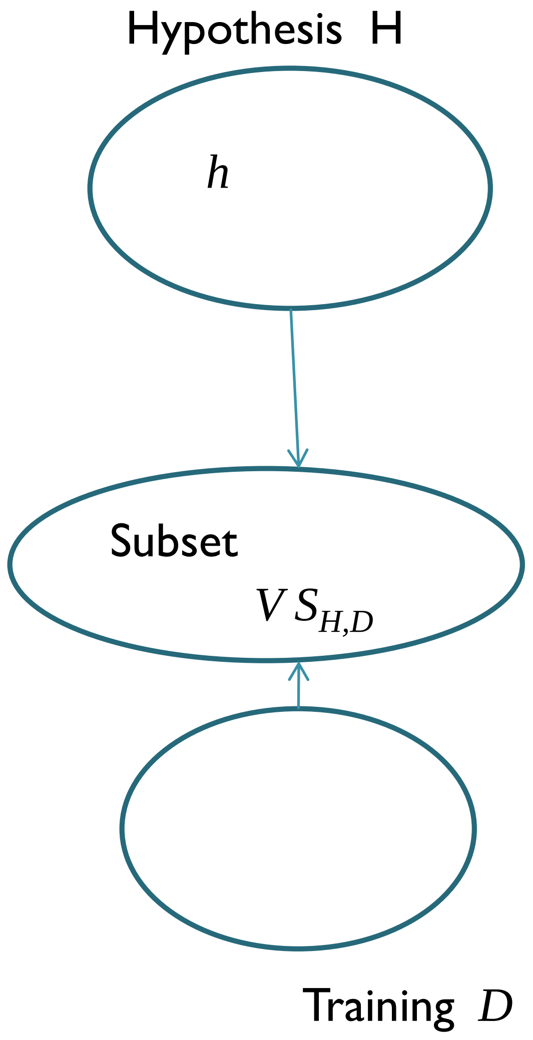
11. Version Spaces

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

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

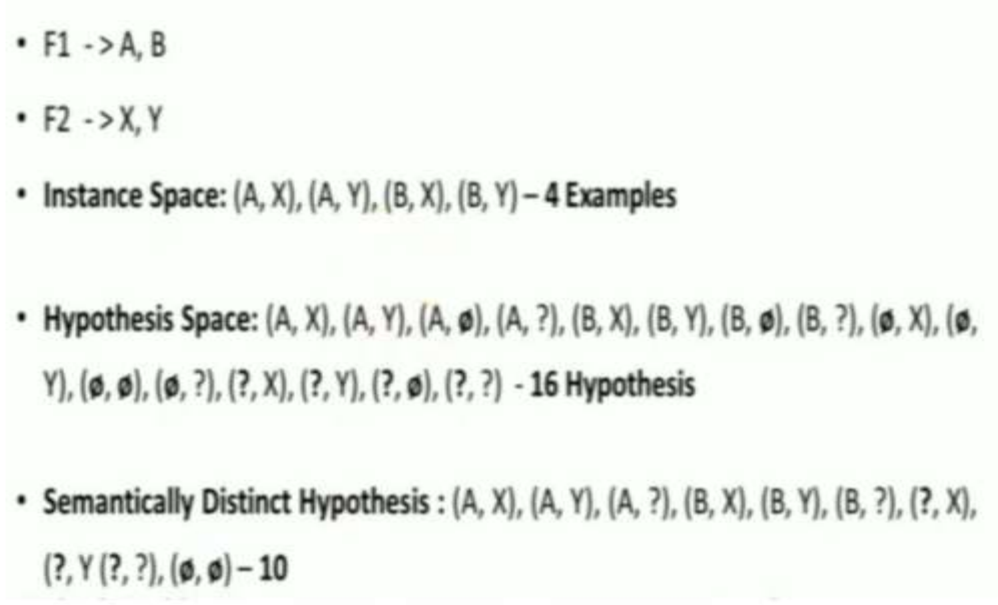
- The **version space**, $VS_{H,D}$, with respect to hypothesis space *H* and training examples *D*, is the subset of hypotheses from *H* consistent with all training examples in *D*.

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

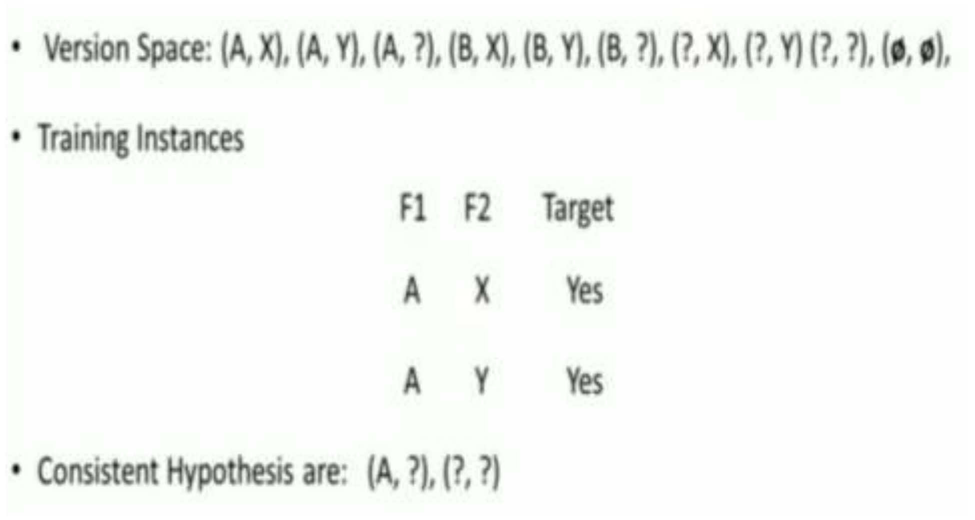


The List-Then-Eliminate Algorithm:

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*



- F1 ->A, B
- F2 ->X, Y
- **Instance Space:** (A, X), (A, Y), (B, X), (B, Y) – **4 Examples**
- **Hypothesis Space:** (A, X), (A, Y), (A, ø), (A, ?), (B, X), (B, Y), (B, ø), (B, ?), (ø, X), (ø, Y), (ø, ø), (ø, ?), (?, X), (?, Y), (?, ø), (?, ?) - **16 Hypothesis**
- **Semantically Distinct Hypothesis :** (A, X), (A, Y), (A, ?), (B, X), (B, Y), (B, ?), (?, X), (?, Y (?, ?), (ø, ø) – **10**



- Version Space: (A, X), (A, Y), (A, ?), (B, X), (B, Y), (B, ?), (?, X), (?, Y) (?, ?), (ø, ø),
- Training Instances

F1	F2	Target
A	X	Yes
A	Y	Yes

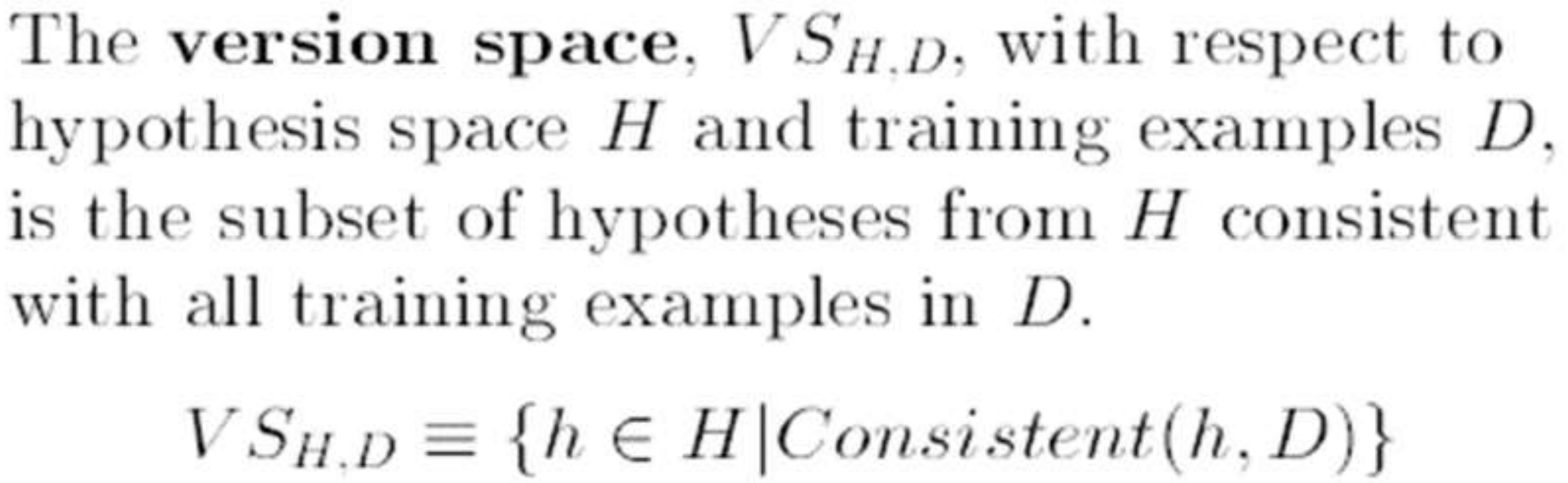
- Consistent Hypothesis are: (A, ?), (?, ?)

Two problems :

1. The Hypothesis space must be finite --- listing is hard ---- lot of time of removing
2. Enumeration of all the hypothesis ,rather inefficient.

Version Spaces

- The Candidate-Elimination algorithm represents the set of all hypotheses consistent with the observed training examples.
- with respect to the hypothesis space H and the training examples D, because it contains all plausible versions of the target concept.This subset of all hypotheses is called the ***version space*** t.



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.

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

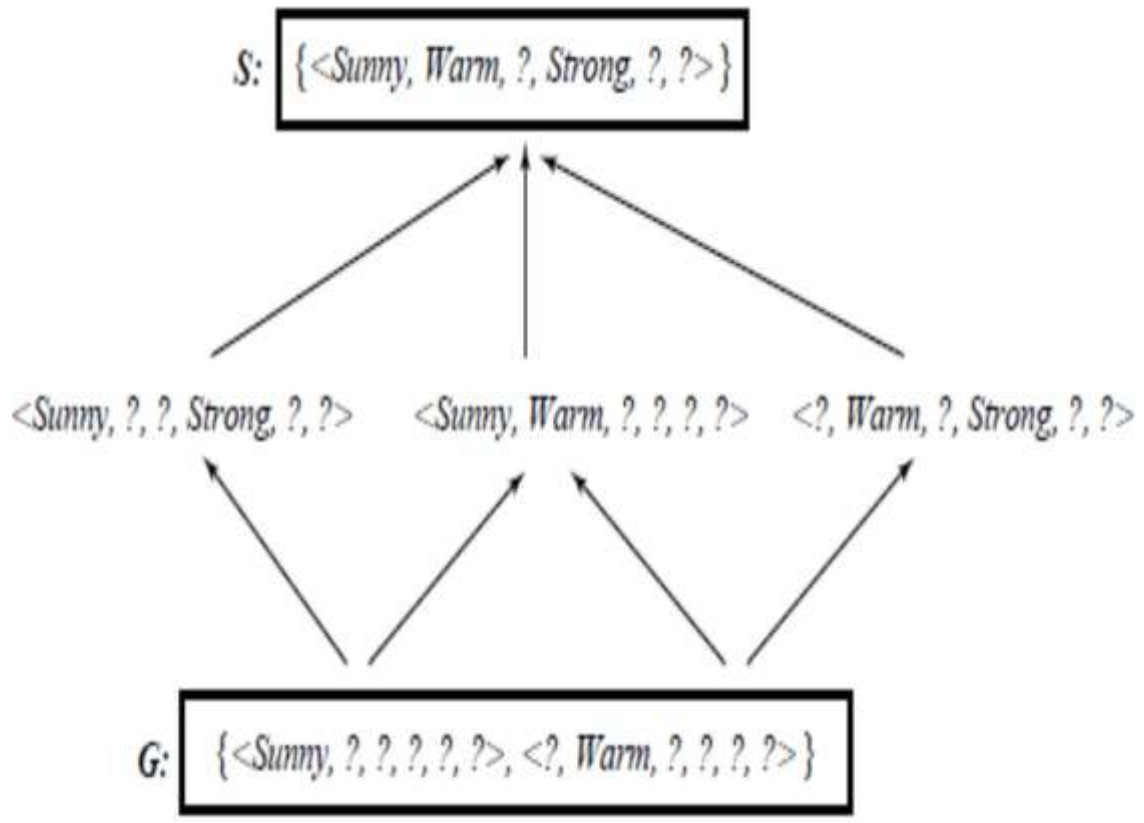
Compact Representation of Version Spaces

- A version space can be represented with its *general* and *specific boundary sets*.
- The Candidate-Elimination algorithm represents the version space by storing only its most general members G and its most specific members S.
- Given only these two sets S and G, it is possible to enumerate all members of a version space by generating hypotheses that lie between these two sets in general-to-specific partial ordering over hypotheses.
- 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.

Version Space used CEA



Specific Hypothesis

version space
Intermediate Hypothesis

General Hypothesis

- A version space with its general and specific boundary sets.
- The version space includes all six hypotheses shown here, but can be represented more simply by S and ***G.***---→***FIND S***

12.Candidate-Elimination Algorithm

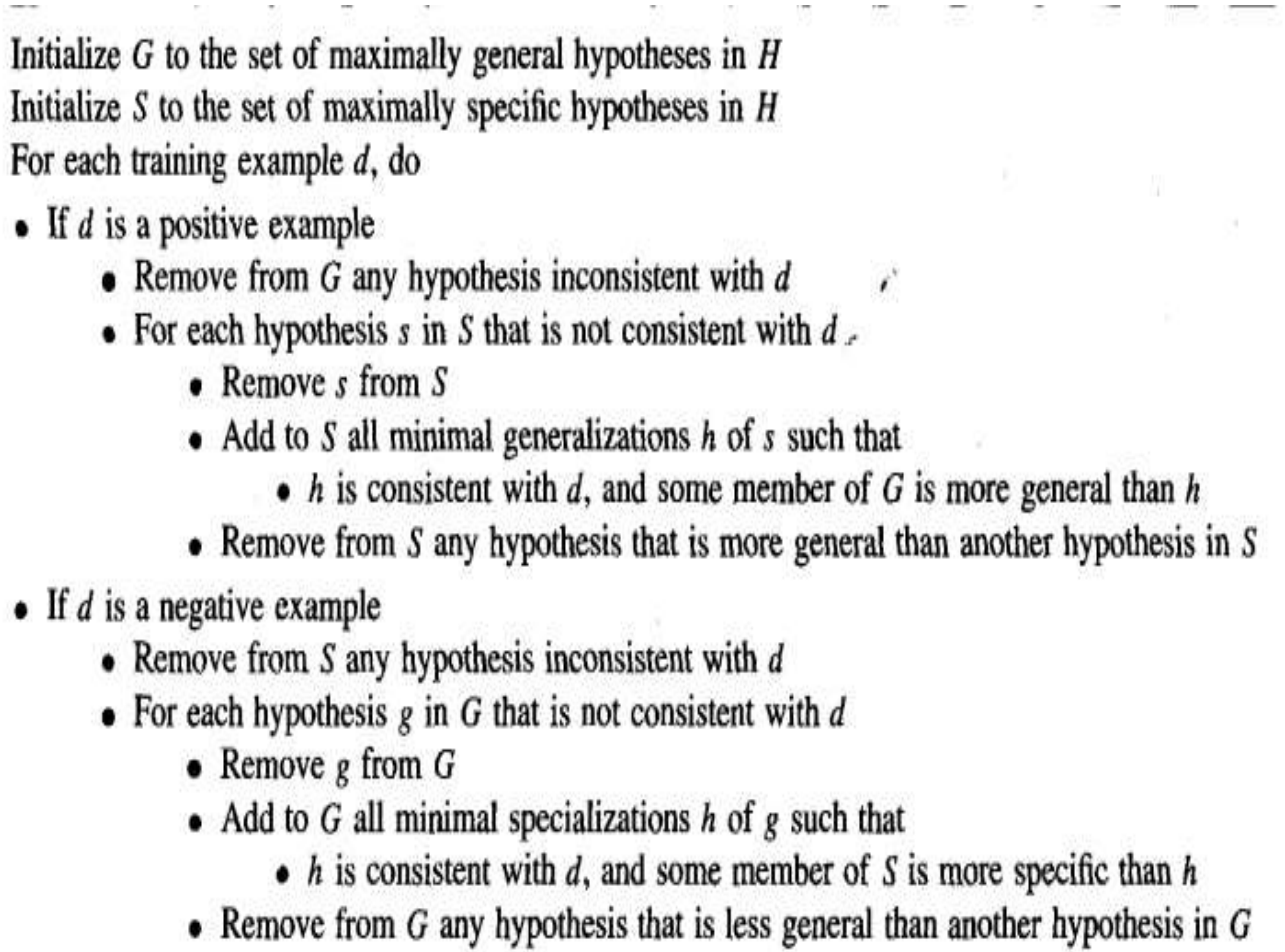
- The Candidate-Elimination algorithm computes the version space containing all hypotheses from H that are consistent with an observed sequence of training examples.
- It begins by initializing the version space to the set of all hypotheses in H; that is, by initializing the G boundary set to contain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

  $$G0 \longleftarrow \{ <?, ?, ?, ?, ?, ?> \}$$

  and initializing the S boundary set to contain the most specific

  hypothesis $S0 \longleftarrow \{ <0, 0, 0, 0, 0, 0> \}$
- These two boundary sets delimit the entire hypothesis space, because every other hypothesis in H is both more general than S0 and more specific than G0.
- As each training example is considered, the S and G boundary sets are generalized and specialized, respectively, to eliminate from the version space any hypotheses found inconsistent with the new training example.
- After all examples have been processed, the computed version space contains all the hypotheses consistent with these examples and only these hypotheses.

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
```

Candidate-Elimination Algorithm - Example

Concept : Days on which person enjoy the sport

Trace0:

S_0: {<∅, ∅, ∅, ∅, ∅, ∅>}

Trace

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

Training examples:

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

- *S0* and *G0* 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.
- They have no effect on the *G* boundary

Trace 1:

Training examples:

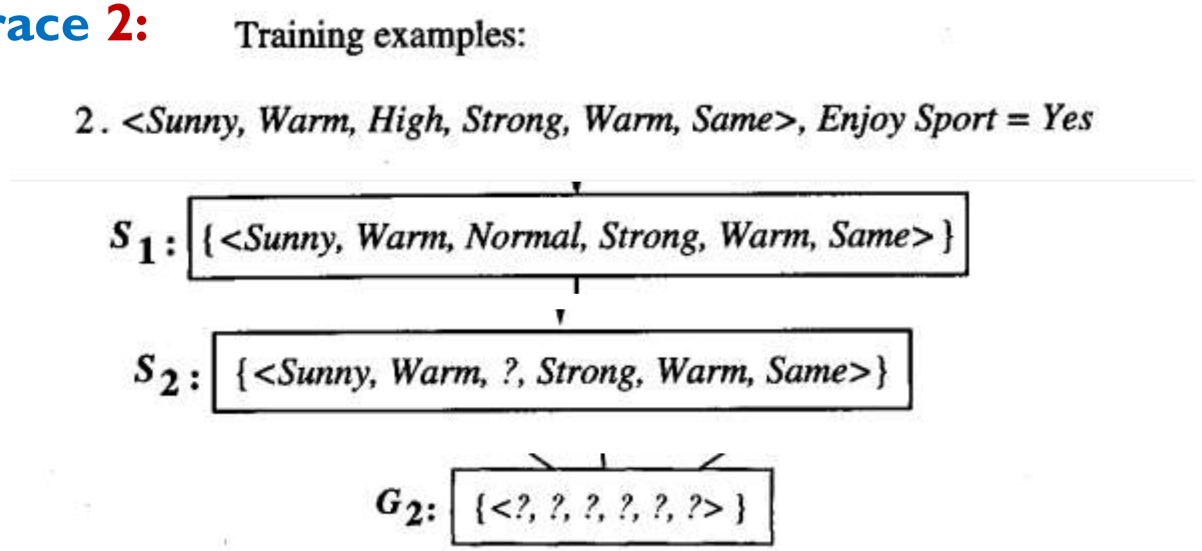
1. <*Sunny, Warm, Normal, Strong, Warm, Same*>, *Enjoy Sport = Yes*

S_1: {<*Sunny, Warm, Normal, Strong, Warm, Same*>}

G_1: {<?, ?, ?, ?, ?, ?>}

"+" S--→ Modify ?

Trace 2:



Training examples:

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

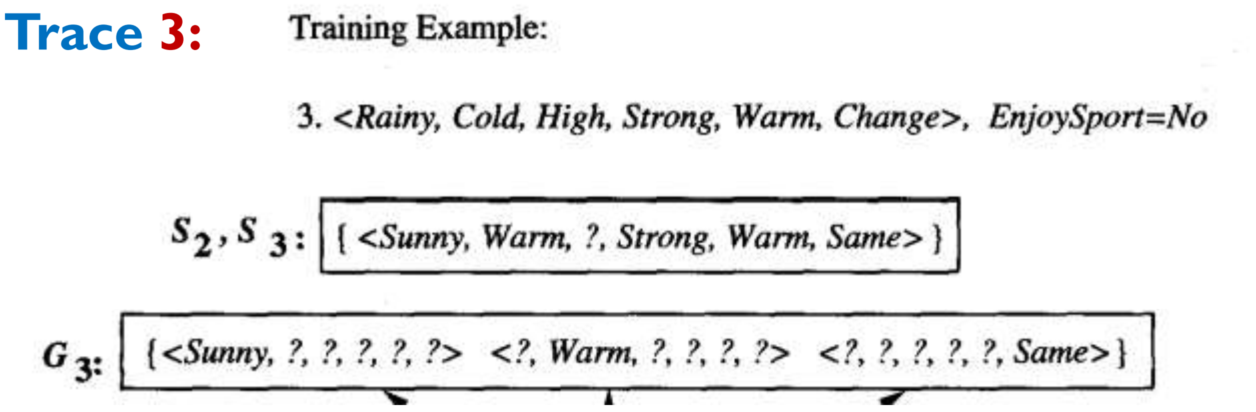
S_1: {<*Sunny, Warm, Normal, Strong, Warm, Same*>}

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

G_2: {<?, ?, ?, ?, ?, ?>}

"+" S--→ Modify ?

Trace 3:



Training Example:

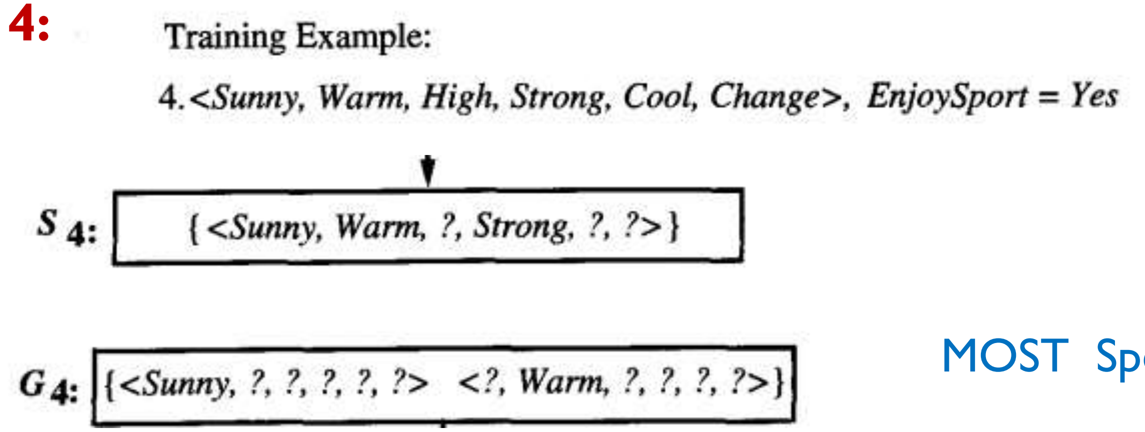
3. <*Rainy, Cold, High, Strong, Warm, Change*>, *EnjoySport=No*

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

G_3: {<*Sunny*, ?, ?, ?, ?, ?> <?, *Warm*, ?, ?, ?, ?> <?, ?, ?, ?, ?, *Same*>}

"_" G--→ Modify ?

Trace 4:



Training Example:

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

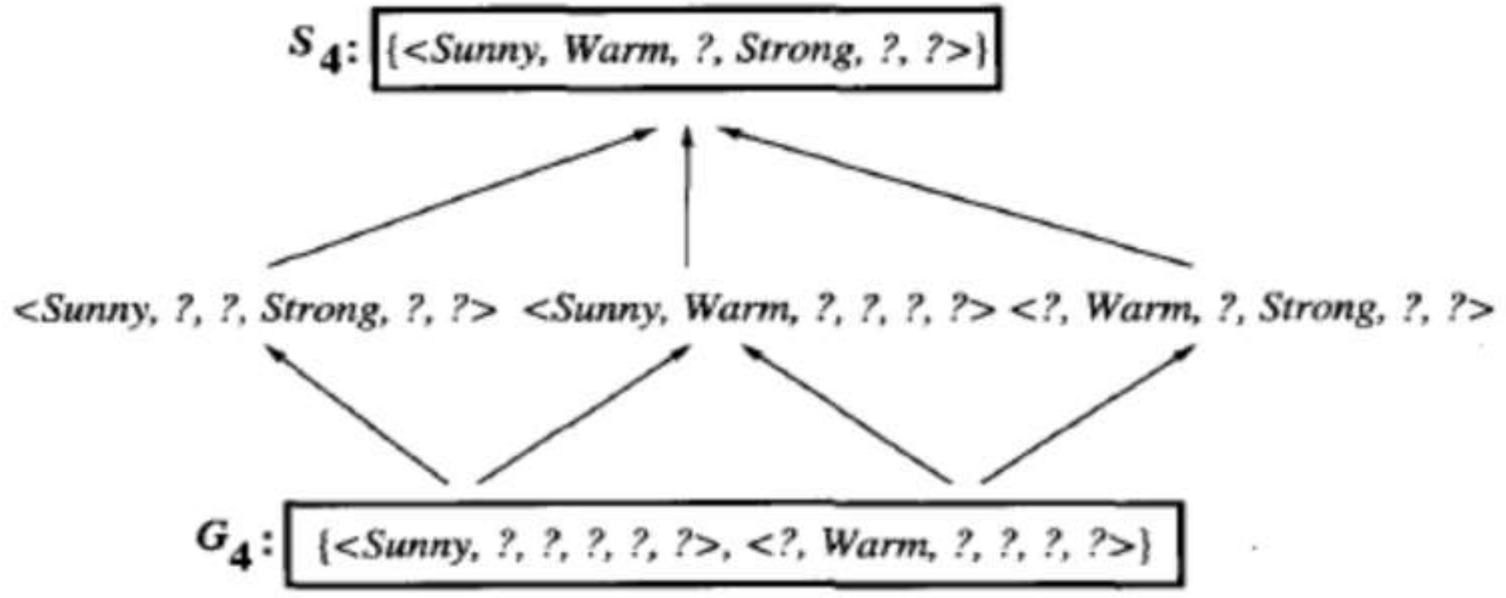
S_4: {<*Sunny, Warm, ?, Strong, ?, ?*>}

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

"+" S--→ Modify ?

MOST Specific & General HYPOTHESIS

Candidate-Elimination Algorithm ----Final Version Space



- After processing these four examples, the boundary sets S4 and G4 delimit the version space of all hypotheses consistent with the set of incrementally observed training examples.
- This learned version space is independent of the sequence in which the training examples are presented (because in the end it contains all hypotheses consistent with the set of examples).
- As further traindata is encountered, the S and G boundaries will move monotonically closer to each other, delimiting a smaller and smaller version space of candidate hypotheses.

13. Remarks on Version Space & Candidate Elimination Algorithm

a. Will Candidate-Elimination Alg.. Converge to Correct Hypothesis?

- The version space learned by the Candidate-Elimination Algorithm will converge toward the hypothesis that correctly describes the target concept, provided
  - There are no errors in the training examples, and
  - there is some hypothesis in H that correctly describes the target concept.
- What will happen if the training data contains errors?
  - The algorithm removes the correct target concept from the version space.
  - S and G boundary sets eventually converge to an empty version space if sufficient additional training data is available.
  - Such an empty version space indicates that there is no hypothesis in H consistent with all observed training examples.
- A similar symptom will appear when the training examples are correct, but the target concept cannot be described in the hypothesis representation.
  - e.g., if the target concept is a disjunction of feature attributes and the hypothesis space supports only conjunctive descriptions

b.What Training Example Should the Learner Request Next?

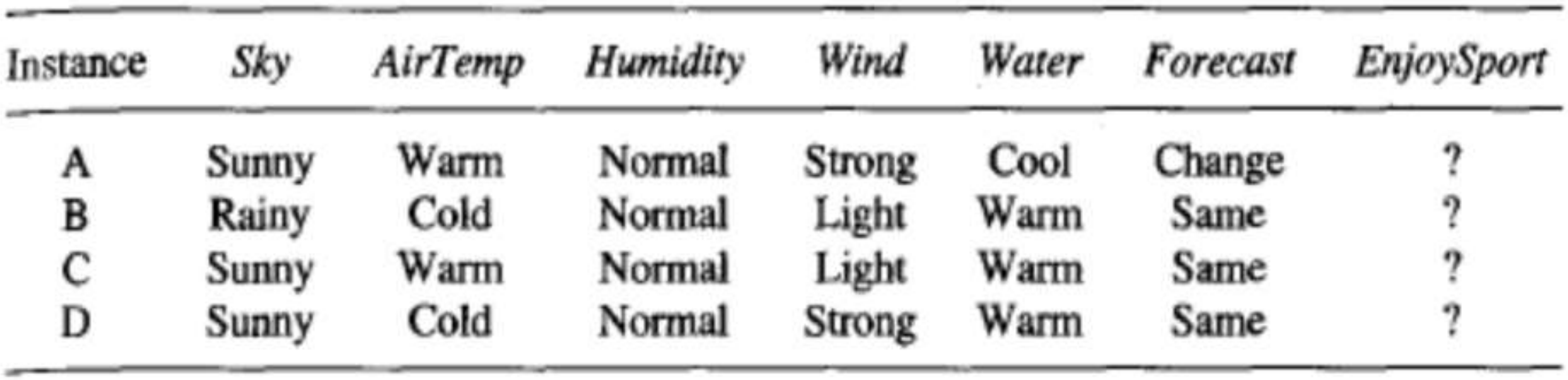
- We have assumed that training examples are provided to the learner by some external teacher.
- Suppose instead that the learner is allowed to conduct experiments in which it chooses the next instance, then obtains the correct classification for this instance from an external oracle (e.g., nature or a teacher).
  - This scenario covers situations in which the learner may conduct experiments in nature or in which a teacher is available to provide the correct classification.
  - We use the term query to refer to such instances constructed by the learner, which are then classified by an external oracle.
- Considering the version space learned from the four training examples of the EnjoySport concept.
  - What would be a good query for the learner to pose at this point?
  - What is a good query strategy in general?

C.What Training Example Should the Learner Request Next?

- The learner should attempt to discriminate among the alternative competing hypotheses in its current version space.
  - Therefore, it should choose an instance that would be classified positive by some of these hypotheses, but negative by others.
  - One such instance is <Sunny, Warm, Normal, Light, Warm, Same>
  - This instance satisfies three of the six hypotheses in the current version space.
  - If the trainer classifies this instance as a positive example, the S boundary of the version space can then be generalized.
  - Alternatively, if the trainer indicates that this is a negative example, the G boundary can then be specialized.
- In general, the optimal query strategy for a concept learner is to generate instances that satisfy exactly half the hypotheses in the current version space.
- When this is possible, the size of the version space is reduced by half with each new example, and the correct target concept can therefore be found with only $\lceil \log_2 |VS| \rceil$ experiments.

d.How Can Partially Learned Concepts Be Used?

- Even though the learned version space still contains multiple hypotheses, indicating that the target concept has not yet been fully learned, it is possible to classify certain examples with the same degree of confidence as if the target concept had been uniquely identified.
- Let us assume that the followings are new instances to be classified:



Instance	*Sky*	*AirTemp*	*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	?

- ***Instance A*** was is classified as a positive instance by every hypothesis in the current version space.
- Because the hypotheses in the version space unanimously agree that this is a positive instance, the learner can classify instance A as positive with the same confidence it would have if it had already converged to the single, correct target concept.
- Regardless of which hypothesis in the version space is eventually found to be the correct target concept, it is already clear that it will classify instance A as a positive example.
- Notice furthermore that we need not enumerate every hypothesis in the version space in order to test whether each classifies the instance as positive.
  - This condition will be met if and only if the instance satisfies every member of S.
  - The reason is that every other hypothesis in the version space is at least as general as some member of S.
  - By our definition of more-general-than, if the new instance satisfies all members of S it must also satisfy each of these more general hypotheses.

- ***Instance B*** is classified as a negative instance by every hypothesis in the version space.
  - This instance can therefore be safely classified as negative, given the partially learned concept.
  - An efficient test for this condition is that the instance satisfies none of the members of G.
- Half of the version space hypotheses classify ***Instance C*** as positive and half classify it as negative.
  - Thus, the learner cannot classify this example with confidence until further training examples are available.
- ***Instance D*** is classified as positive by two of the version space hypotheses and negative by the other four hypotheses.
  - In this case we have less confidence in the classification than in the unambiguous cases of instances A and B.
  - Still, the vote is in favor of a negative classification, and one approach we could take would be to output the majority vote, perhaps with a confidence rating indicating how close the vote was.

13. Remarks on Version Space & Candidate Elimination Algorithm

a. Will Candidate-Elimination Alg.. Converge to Correct Hypothesis?

- The VS learned by the CEA will converge toward the hypothesis that correctly describes the target concept, provided
  - There are no errors in the training examples, and
  - there is some hypothesis in H that correctly describes the target concept.
- What will happen if the training data contains errors?
  - The algorithm removes the correct target concept from the version space.
  - S and G boundary sets eventually converge to an empty version space if sufficient additional training data is available.
  - Such an empty version space indicates that there is no hypothesis in H consistent with all observed training examples.
- A similar symptom will appear when the training examples are correct, but the target concept cannot be described in the hypothesis representation.
  - e.g., if the target concept is a disjunction of feature attributes and the hypothesis space supports only conjunctive descriptions

b.What Training Example Should the Learner Request Next?

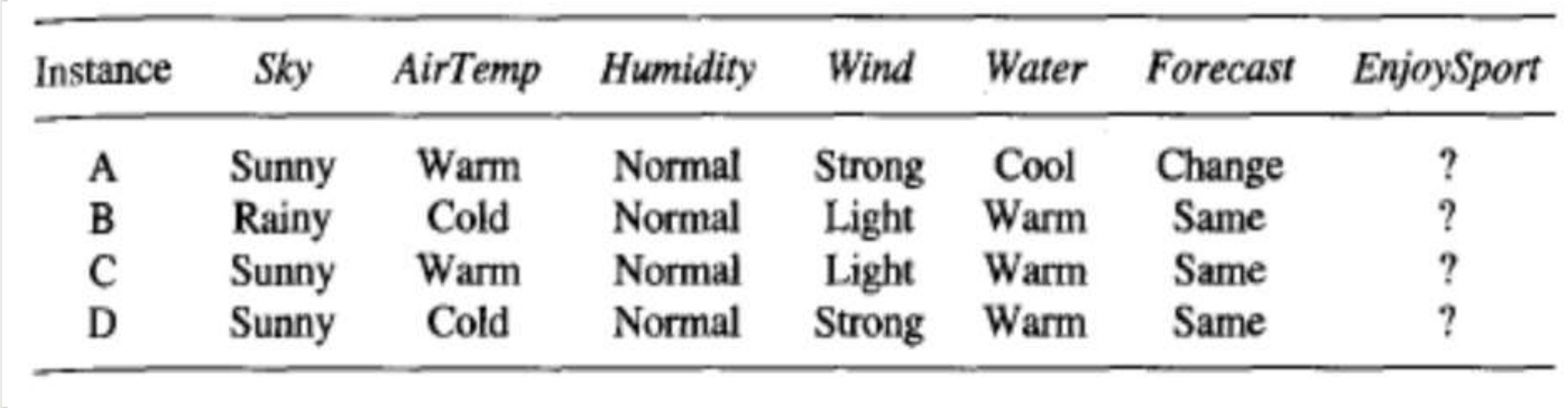
- Assumed that training examples are provided to the learner by some external teacher.
- Suppose instead that the learner is allowed to conduct experiments in which it chooses the next instance, then obtains the correct classification for this instance from an external oracle (e.g., nature or a teacher).
  - This scenario covers situations in which the learner may conduct experiments in nature or in which a teacher is available to provide the correct classification.
  - We use the term query to refer to such instances constructed by the learner, which are then classified by an external oracle.
- Considering the VS learned from the 4 training examples of the EnjoySport concept.
  - What would be a good query for the learner to pose at this point?
  - What is a good query strategy in general?

C.What Training Example Should the Learner Request Next?

- The learner should attempt to discriminate among the alternative competing hypotheses in its current VS.
  - it should choose an instance that would be classified positive by some of these hypotheses, but negative by others.
  - One such instance is <Sunny, Warm, Normal, Light, Warm, Same>
  - This instance satisfies three of the six hypotheses in the current VS.
  - If the trainer classifies this instance as a positive example, the *S boundary* of the VS can then be *generalized*.
  - Alternatively, if the trainer indicates that this is a *negative example*, the **G boundary** of VS can then be *specifized*.
- In general, the optimal query strategy for a concept learner is to generate instances that satisfy exactly half the hypotheses in the current version space.
- When this is possible, the size of the version space is reduced by half with each new example, and the correct target concept can therefore be found with only $\lceil \log_2 |VS| \rceil$ experiments.

d. How Can Partially Learned Concepts Be Used?

- Even though the learned version space still contains multiple hypotheses, indicating that the target concept has not yet been fully learned, it is possible to classify certain examples with the same degree of confidence as if the target concept had been uniquely identified.
- Let us assume that the followings are new instances to be classified:



Instance	*Sky*	*AirTemp*	*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	?

- ***Instance A*** was is classified as a positive instance by every hypothesis in the current version space.
- Because the hypotheses in the version space unanimously agree that this is a positive instance, the learner can classify instance A as positive with the same confidence it would have if it had already converged to the single, correct target concept.
- Regardless of which hypothesis in the version space is eventually found to be the correct target concept, it is already clear that it will classify instance A as a positive example.
- Notice furthermore that we need not enumerate every hypothesis in the version space in order to test whether each classifies the instance as positive.
  - This condition will be met if and only if the instance satisfies every member of S.
  - The reason is that every other hypothesis in the version space is at least as general as some member of S.
  - By our definition of more-general-than, if the new instance satisfies all members of S it must also satisfy each of these more general hypotheses.

- ***Instance B*** is classified as a negative instance by every hypothesis in the version space.
  - This instance can therefore be safely classified as negative, given the partially learned concept.
  - An efficient test for this condition is that the instance satisfies none of the members of G.
- Half of the version space hypotheses classify ***instance C*** as positive and half classify it as negative.
  - Thus, the learner cannot classify this example with confidence until further training examples are available.
- ***Instance D*** is classified as positive by two of the version space hypotheses and negative by the other four hypotheses.
  - In this case we have less confidence in the classification than in the unambiguous cases of instances A and B.
  - Still, the vote is in favor of a negative classification, and one approach we could take would be to output the majority vote, perhaps with a confidence rating indicating how close the vote was.

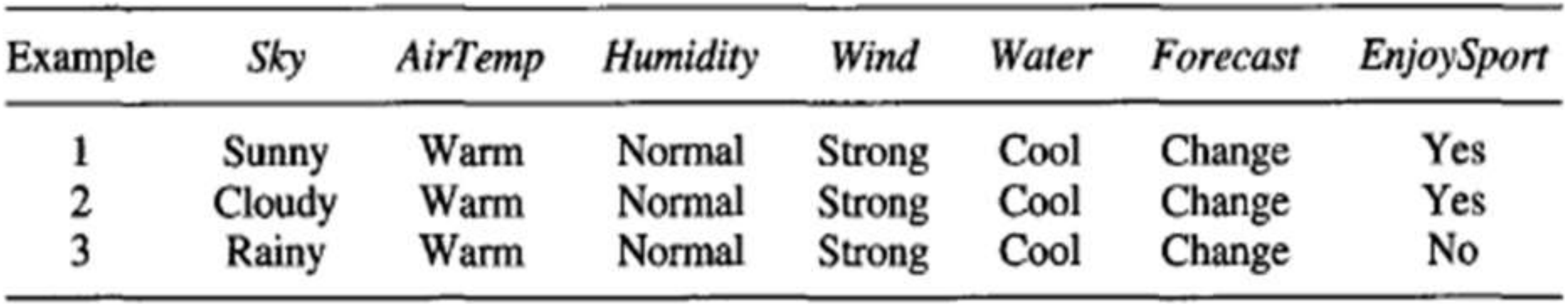
15. InductiveBias - Fundamental Questions for Inductive Inference

The Candidate-Elimination Algorithm will converge toward the true target concept provided it is given accurate training examples and provided its initial hypothesis space contains the target concept.

- What if the target concept is not contained in the hypothesis space?
- Can we avoid this difficulty by using a hypothesis space that includes every possible hypothesis?
- How does the size of this hypothesis space influence the ability of the algorithm to generalize to unobserved instances?
- How does the size of the hypothesis space influence the number of training examples that must be observed?

Inductive Bias - A Biased Hypothesis Space

- In EnjoySport example, we restricted the hypothesis space to include only conjunctions of attribute values.
  - Because of this restriction, the hypothesis space is unable to represent even simple disjunctive target concepts such as "Sky = Sunny or Sky = Cloudy."



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

- From first two examples $\longrightarrow$ S2 : <?, Warm, Normal, Strong, Cool, Change>
  - This is inconsistent with third examples, and there are no hypotheses consistent with these three examples

PROBLEM: We have biased the learner to consider only conjunctive hypotheses.

$\longrightarrow$ We require a more expressive hypothesis space.

Inductive Bias - An Unbiased Learner

- The obvious solution to the problem of assuring that the target concept is in the hypothesis space H is to provide a hypothesis space capable of representing every teachable concept.
  - Every possible subset of the instances X $\longrightarrow$ ***the power set of X.***

- What is the size of the hypothesis space H (the power set of X) ?
  - In EnjoySport, the size of the instance space X is 96.
  - The size of the power set of X is $2^{|X|}$ $\longrightarrow$ The size of H is 2^{96}
  - Our conjunctive hypothesis space is able to represent only 973of these hypotheses.

    $\longrightarrow$ a very biased hypothesis space

Inductive Bias - An Unbiased Learner : Problem

- Let the hypothesis space H to be the power set of X.
  - A hypothesis can be represented with disjunctions, conjunctions, and negations of our earlier hypotheses.
  - The target concept "Sky = Sunny or Sky = Cloudy" could then be described as

    <Sunny, ?, ?, ?, ?, ?> ∨ <Cloudy, ?, ?, ?, ?, ?>

NEW PROBLEM: our concept learning algorithm is now completely unable to generalize beyond the observed examples.

- three positive examples (xl,x2,x3) and two negative examples (x4,x5) to the learner.
- S : { x1 ∨ x2 ∨ x3 } and G : { ¬ (x4 ∨ x5) } □ NO GENERALIZATION
- Therefore, the only examples that will be unambiguously classified by S and G are the observed training examples themselves.

Inductive Bias – Futility of Bias Free Learning

Fundamental Property of Inductive Inference

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

*This learning task is representative of a large class of tasks for which the legal moves that define some large search space are known a **priori**, but for which the best search strategy is not known.*

- ***Inductive Leap**: A learner should be able to generalize training data using prior assumptions in order to classify unseen instances.*

- The generalization is known as **inductive leap** and our prior assumptions are the **inductive bias** of the learner.

- Inductive Bias (prior assumptions) of Candidate-Elimination Algorithm is that the target concept can be represented by a conjunction of attribute values, the target concept is contained in the hypothesis space and training examples are correct.

Inductive Bias – Formal Definition

Inductive Bias:

Consider a concept learning algorithm ***L*** for the set of instances ***X***. Let ***c*** be an arbitrary concept defined over ***X***, and

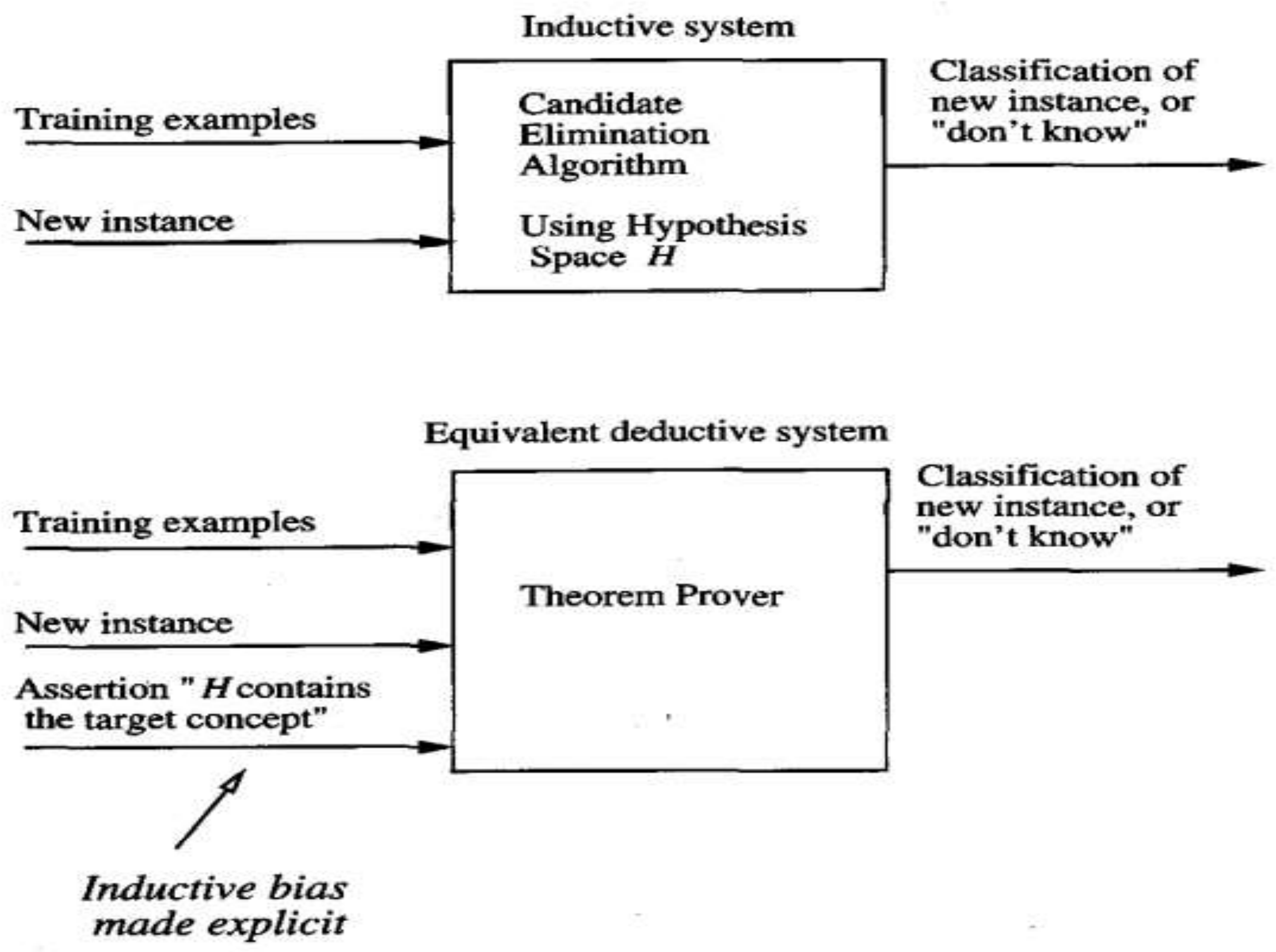
let ***D***D_c **= {<*x* , *c(x)*>}** be an arbitrary set of training examples of ***c***.

Let ***L(xi, Dc)*** denote the classification assigned to the instance ***xi*** by ***L*** after training on the data ***Dc*****.**

The **inductive bias** of ***L*** is any minimal set of assertions ***B*** such that for any target concept ***c*** and corresponding training examples ***Dc*** the following formula holds.

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

Modelling Inductive systems by equivalent deductive systems


Inductive system
Training examples
New instance
Candidate Elimination Algorithm
Using Hypothesis Space H
Classification of new instance, or "don't know"
Equivalent deductive system
Training examples
New instance
Assertion "H contains the target concept"
Theorem Prover
Classification of new instance, or "don't know"
Inductive bias made explicit

Definitions to Remember

"LEARNING,“ : The class of programs that improve through experience E.

DESIGNING A LEARNING SYSTEM : 1.Choosing the Training Experience
2. Choosing the Target Function
3. Representation for the Target Function (Algorithm)
4. Function Approximation Algorithm(LMS ,weights ,Accuracy)
5.The Final Design

Concept learning can be formulated as a *problem of searching through a predefined space* of potential hypotheses for the hypothesis (h ,H) that best fits the training examples.

Concept learning: Inferring a boolean-valued function from training examples of its input and output.

Each hypothesis h consists of a conjunction of constraints on the instance attributes.

"?' that any value is acceptable for this attribute
specify a single required value (e.g.,Warm) for the attribute,
“ φ” or "0" that no value is acceptable.

most general hypothesis -- every day is positive eg ----→ <?, ?, ?, ?, ?, ?>

most specific possible hypothesis-that no day is a positive eg -----→ <φ, φ ,φ, φ ,φ, φ>

➢target concept ----- c

➢set of instances, X ----- all possible days----- all rows in the dataset

➢Particular instance x

➢c(x) = 1 ------ positive examples

➢ c(x) = 0 -------- Negative examples

➢Set of all possible Hypotheses H

➢h each hypothesis h , a boolean-valued function

➢h : X → {0, 1}

➢ordered pair <x, c(x)> to describe the training example **<x, h(x)>**

➢D set of available training examples or experience

➢d is each training example or experience

➢determine a hypothesis h identical to the target concept c h(x) = c(x) is **inductive learning**

➢distinct instances(possible attributes number products all instances X

➢syntactically distinct hypotheses (attributes ? & φ)

➢semantically distinct hypotheses is only--→ 1 + (attributes & ?)

➢The subset of hypotheses from **H consistent** with the training examples D is Version Space

$\geq_g$