Machine Learning

Chapter 2

CONCEPT LEARNING AND THE GENERAL-TO-SPECIFIC ORDERING

Dr.T.ADILAKSHMI

INTRODUCTION

- Much of learning involves acquiring general concepts from specific training examples. People, for example, continually learn general concepts or categories such as "bird," "car,".
- Each such concept can be viewed as describing some subset of objects or events defined over a larger set e.g., the subset of animals that constitute birds.
- Alternatively, each concept can be thought of as a boolean-valued function defined over this larger set
- e.g., a function defined over all animals, whose value is true for birds and false for other animals.

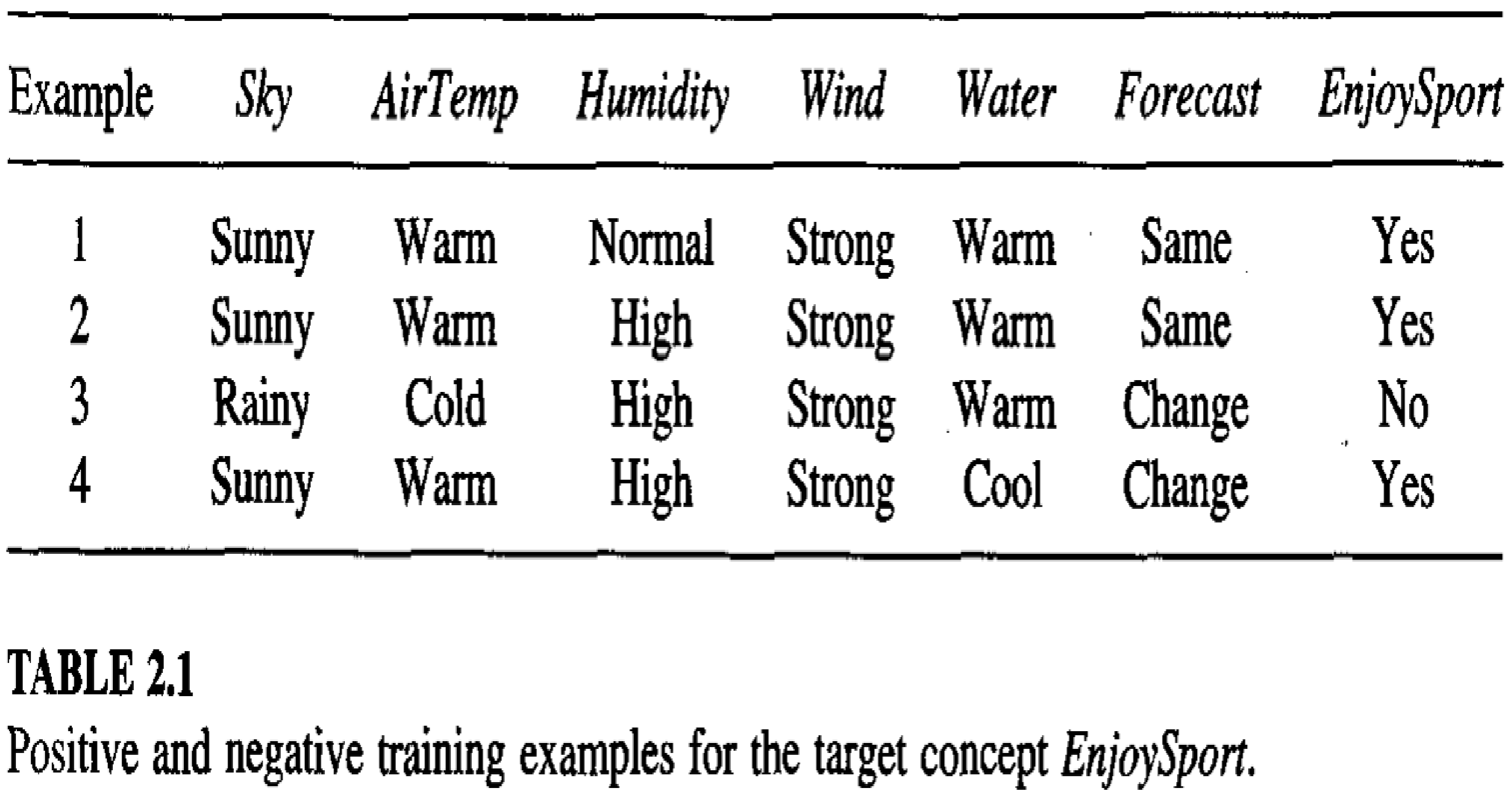
INTRODUCTION

- In this chapter we consider the problem of automatically inferring the general definition of some concept, given examples labeled as members or nonmembers of the concept. This task is commonly referred to as ***concept learning*** or approximating a boolean-valued function from examples.
- **Concept learning:** Inferring a boolean-valued function from training examples of its input and output.

A CONCEPT LEARNING TASK

- consider the example task of learning the target concept "days on which my friend Aldo enjoys his favorite water sport.“
- Table 2.1 describes a set of example days, each represented by a set of attributes.
- The attribute EnjoySport indicates whether or not Aldo enjoys his favorite water sport on this day.
- The task is to learn to predict the value of EnjoySport for an arbitrary day, based on the values of its other attributes.

Example of Concept Learning: EnjoySport



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

TABLE 2.1
Positive and negative training examples for the target concept *EnjoySport*.

- Let us begin by considering a simple representation in which each hypothesis consists of a conjunction of constraints on the instance attributes.
- each hypothesis be 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
  - ➢ indicate by a "?' that any value is acceptable for this attribute,
  - ➢ specify a single required value (e.g., Warm) for the attribute, or
  - ➢ indicate by a "0" that no value is acceptable.
- If some instance x satisfies all the constraints of hypothesis h, then h classifies x as a positive example (h(x) = 1)
- The hypothesis that Aldo enjoys his favorite sport only on cold days with high humidity (independent of the values of the other attributes) is represented by the expression

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

- The most general hypothesis-that every day is a positive example-is represented by

  (?, ?, ?, ?, ?, ?)
- The most specific possible hypothesis-that no day is a positive example-is represented by

  (0, 0, 0, 0, 0, 0)
- the EnjoySport concept learning task requires learning the set of days for which EnjoySport = yes, describing this set by a conjunction of constraints over the instance attributes.
- In general, any concept learning task can be described by the set of instances over which the target function is defined, the target function, the set of candidate hypotheses considered by the learner, and the set of available training examples

- **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), "∅" (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.

TABLE 2.2
The *EnjoySport* concept learning task.

Notation

- The set of items over which the concept is defined is called the set of instances, which we denote by X
- In the current example, X is the set of all possible days, each represented by the attributes Sky, AirTemp, Humidity, Wind, Water, and Forecast.
- The concept or function to be learned is called the **target concept,** which we denote by c. In general, c can be any boolean-valued function defined over the instances X; that is, c : X -> {0,1}.
- In the current example, the target concept corresponds to the value of the attribute EnjoySport c(x) = 1 if EnjoySport = Yes, and c(x) = 0 if EnjoySport = No

- When learning the target concept, the learner is presented a set of training examples, each consisting of an instance x from X, along with its target concept value c(x) (e.g., the training examples in Table 2.1).
- Instances for which c(x) = 1 are called positive examples, or members of the target concept. Instances for which C(X) = 0 are called negative examples, or nonmembers of the target concept.
- We will often write the ordered pair (x, c(x)) to describe the training example consisting of the instance x and its target concept value c(x).
- We use the symbol D (training and testing) to denote the set of available training examples

- Given a set of training examples of the target concept c, the problem faced by the learner is to hypothesize, or estimate, c.
- We use the symbol H to denote the set of all possible hypotheses that the learner may consider regarding the identity of the target concept.
- 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.**

Inductive Learning Hypothesis

- learning task is to determine a hypothesis h identical to the target concept c over the entire set of instances X, the only information available about c is its value over the training examples.
- inductive learning algorithms can at best guarantee that the output hypothesis fits the target concept over the training data.
- our assumption is that the best hypothesis regarding unseen instances is the hypothesis that best fits the observed training data. This is the fundamental assumption of inductive learning

- **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 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 that best fits the training examples.

CONCEPT LEARNING AS SEARCH

- Consider, for example, the instances X and hypotheses H in the EnjoySport learning task.
- Given that the attribute Sky has three possible values, and that AirTemp, Humidity, Wind, Water, and Forecast each have two possible values, the instance space X contains exactly 3.2.2.2.2.2 = 96 distinct instances.

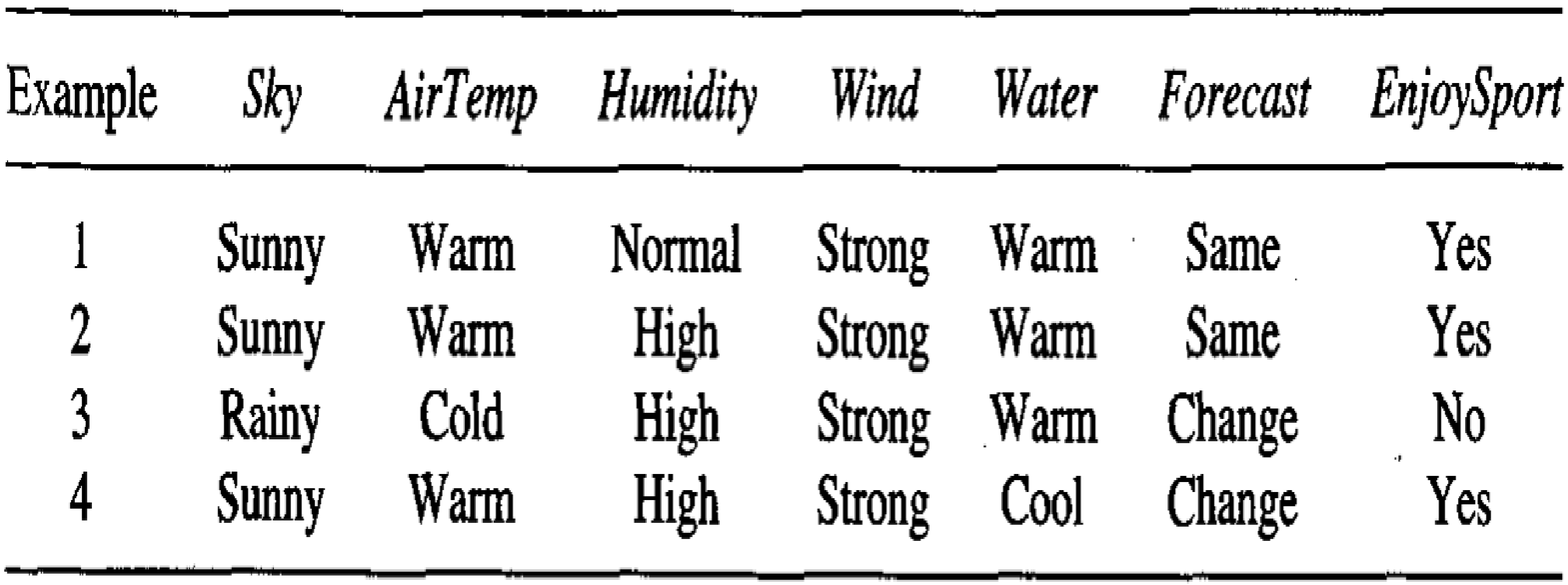
General-to-Specific Ordering of Hypotheses

- To illustrate the general-to-specific ordering, consider the two hypotheses

  h1 = (Sunny, ?, ?, Strong, ?, ?)

  h2 = (Sunny, ?, ?, ?, ?, ?)
- Now consider the sets of instances that are classified positive by h1 and by h2. Because h2 imposes fewer constraints on the instance, it classifies more instances as positive.
- In fact, any instance classified positive by h1 will also be classified positive by h2. Therefore, we say that h2 is more general than h1.
- Any instance x in X and hypothesis h in H, we say that x satisfies h if and only if h(x) = 1.

Example of Concept Learning: EnjoySport



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

TABLE 2.1
Positive and negative training examples for the target concept *EnjoySport*.

- Definition: Let hj and hk be boolean-valued functions defined over X. Then hj is more-general-than-or-equal-to hk (written hj $>=_g$ hk) if and only if

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

We will also find it useful to consider cases where one hypothesis is strictly more general than the other. Therefore, we will say that h_j is (strictly) *more_general_than*

h_k (written $h_j >_g h_k$) if and only if $(h_j \geq_g h_k) \wedge (h_k \not\geq_g h_j)$. Finally, we will sometimes find the inverse useful and will say that h_j is *more_specific_than* h_k when h_k is *more_general_than* h_j.

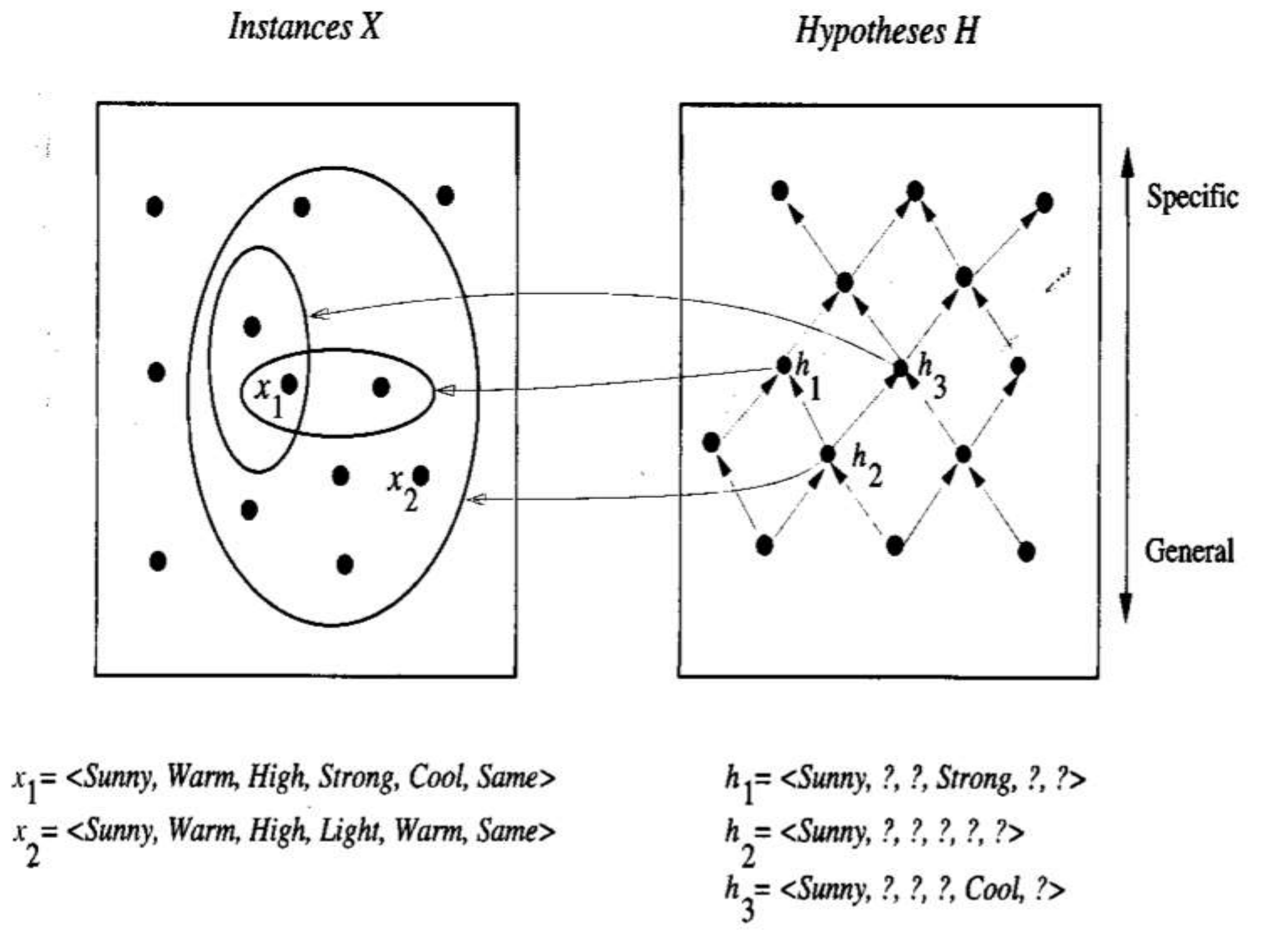
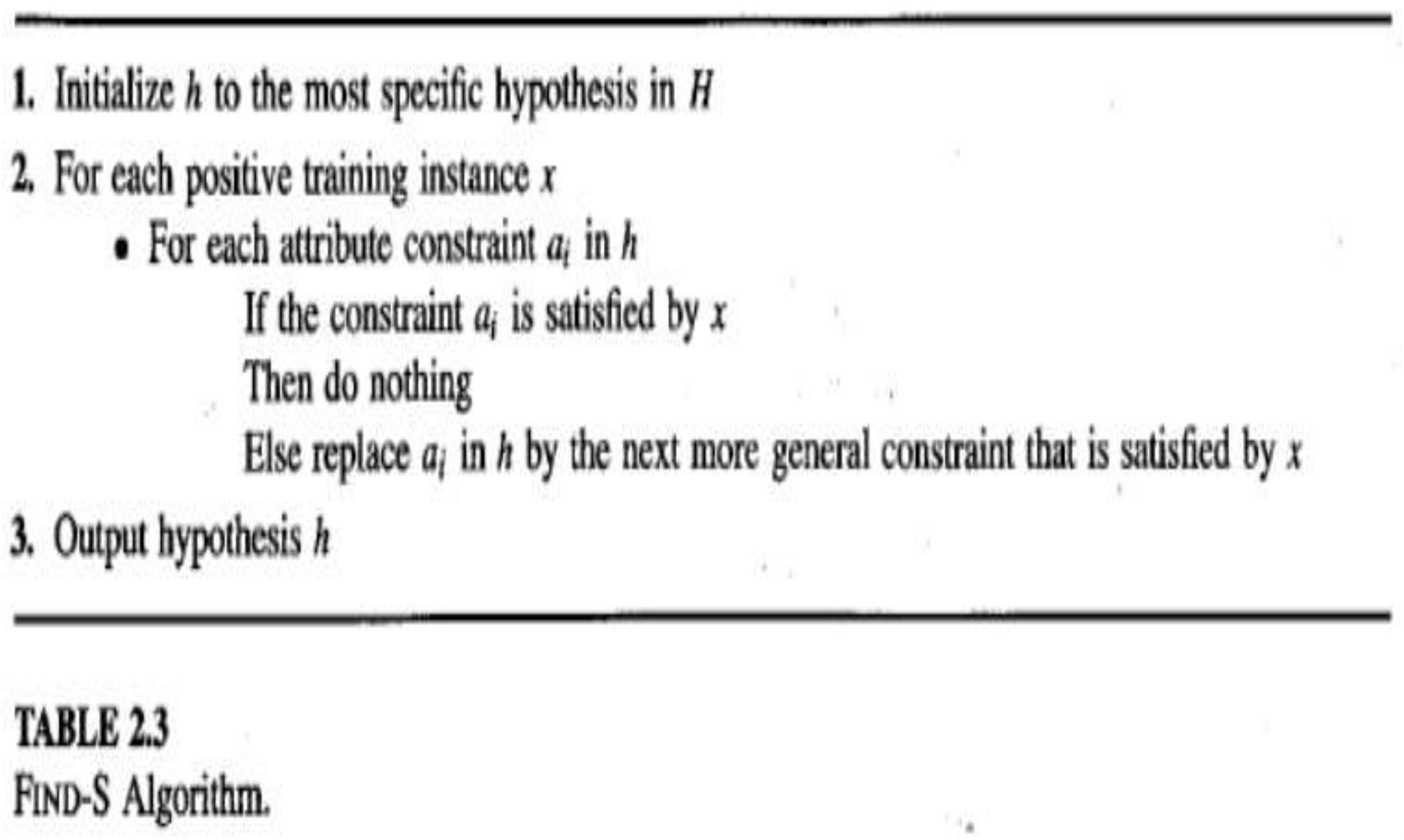

Instances X
Hypotheses H
Specific
General
x_1
x_2
h_1
h_2
h_3
x_1= <Sunny, Warm, High, Strong, Cool, Same>
x_2= <Sunny, Warm, High, Light, Warm, Same>
h_1= <Sunny, ?, ?, Strong, ?, ?>
h_2= <Sunny, ?, ?, ?, ?, ?>
h_3= <Sunny, ?, ?, ?, Cool, ?>

FIGURE 2.1 Instances, hypotheses, and the more-general-than relation. The box on the left represents the set X of all instances, the box on the right the set H of all hypotheses.

- hypothesis h2 is more general than hl because every instance that satisfies hl also satisfies h2.
- Similarly, h2 is more general than h3.
- Note that neither hl nor h3 is more general than the other; although the instances satisfied by these two hypotheses intersect, neither set subsumes the other

FIND-S: FINDING A MAXIMALLY SPECIFIC HYPOTHESIS



```
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 $h$
```

TABLE 2.3
FIND-S Algorithm.

FIND-S: FINDING A MAXIMALLY SPECIFIC HYPOTHESIS

- To illustrate this algorithm, assume the learner is given the sequence of training examples from Table 2.1 for the EnjoySport task.
- The first step of FIND-S is to initialize h to the most specific hypothesis in H

$$h \leftarrow (0, 0, 0, 0, 0, 0)$$

1. Sunny Warm Normal Strong Warm Same Yes

- In particular, none of the "0" constraints in h are satisfied by this example, so each is replaced by the next more general constraint that fits the example; namely, the attribute values for this training example.

  h <-(Sunny, Warm, Normal, Strong, Warm, Same)
- This h is still very specific; it asserts that all instances are negative except for the single positive training example we have observed.

  2 Sunny Warm High Strong Warm Same Yes
- The second training example (also positive in this case) forces the algorithm to further generalize h, this time substituting a "?' in place of any attribute value in h that is not satisfied by the new example.

- The refined hypothesis in this case is

  h <-(Sunny, Warm, ?, Strong, Warm, Same)

- the FIND-S algorithm simply ignores every negative example

  3. Rainy Cold High Strong Warm Change No

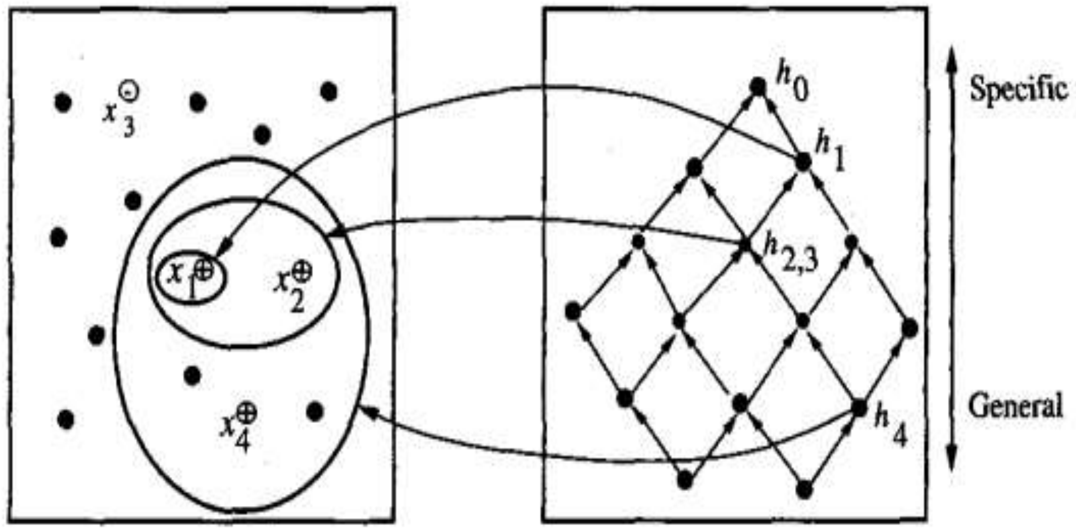
- But the target concept c will never cover a negative example, thus neither will h (by the definition of more-general than).
- Therefore, no revision to h will be required in response to any negative example.
- To complete our trace of FIND-S, the fourth (positive) example leads to a further generalization of h

  4. Sunny Warm High Strong Cool Change Yes

  h <-(Sunny, Warm, ?, Strong, ?, ?)

- The search moves from hypothesis to hypothesis, searching from the most specific to progressively more general hypotheses along one chain of the partial ordering.
- At each step, the hypothesis is generalized only as far as necessary to cover the new positive example. Therefore, at each stage the hypothesis is the most specific hypothesis consistent with the training examples observed up to this point (hence the name FIND-S).

Example of Find-S



	h_0 = <Ø, Ø, Ø, Ø, Ø, Ø>
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 ? ? >*

FIGURE 2.2
The hypothesis space search performed by FIND-S. The search begins (h_0) with the most specific hypothesis in H, then considers increasingly general hypotheses (h_1 through h_4) as mandated by the training examples. In the instance space diagram, positive training examples are denoted by "+," negative by "–," and instances that have not been presented as training examples are denoted by a solid circle.

FIND-S

- 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.**
- Its final hypothesis will also be consistent with the negative examples provided the correct target concept is contained in H, and provided the training examples are correct.
- However, there are several questions still left unanswered by this learning algorithm,

FIND-S

- Has the learner converged to the correct target concept? Although FIND-S will find a hypothesis consistent with the training data, it has no way to determine whether it has found the ***only hypothesis in H consistent with*** the data or whether there are many other consistent hypotheses as well
- Why prefer the most specific hypothesis? In case there are multiple hypotheses consistent with the training examples, FIND-S will find the most specific. It is unclear whether we should prefer this hypothesis over, say, the most general, or some other hypothesis of intermediate generality.

- Are the training examples consistent? In most practical learning problems there is some chance that the training examples will contain at least some errors or noise. Such inconsistent sets of training examples can severely mislead FIND-S, given the fact that it ignores negative examples.
- What if there are several maximally specific consistent hypotheses? In the hypothesis language H for the ***EnjoySport task, there is always a unique,*** most specific hypothesis consistent with any set of positive examples.

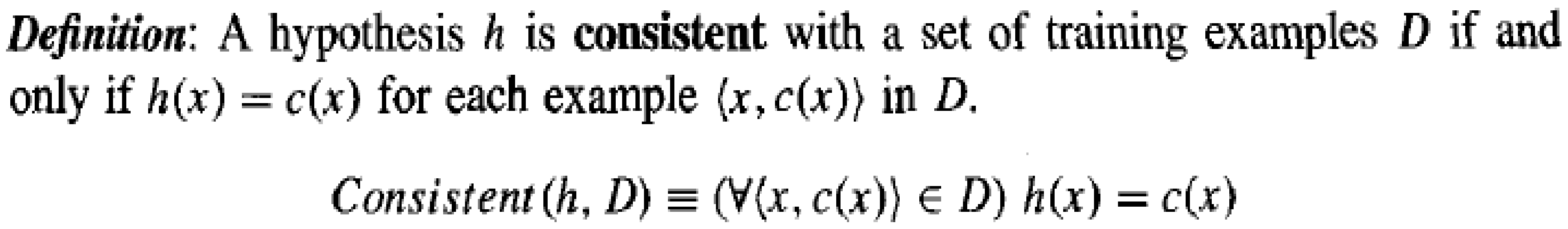
VERSION SPACES AND THE CANDIDATE-ELIMINATIO ALGORITHM

- This section describes a second approach to concept learning, the CANDIDATE ELIMINATIO algorithm
- That addresses several of the limitations of FIND-S.
- although FIND-S outputs a hypothesis from H, that is consistent with the training examples, this is just one of many hypotheses from H that might fit the training data equally well.
- The key idea in the CANDIDATE-ELIMINATlON Algo is to output a description of the set of ***all hypotheses consistent with the training examples.***

- This is accomplished by using the *more-general-than* partial ordering
- to maintain a compact representation of the set of consistent hypotheses and to incrementally refine this representation as each new training example is encountered

Representation

- The CANDIDATE-ELIMINATION algorithm finds all describable hypotheses that are consistent with the observed training examples



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

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

An example *x is said to* ***satisfy hypothesis h when h(x) = 1,*** regardless of whether x is a positive or negative example of the target concept

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

Version Space

- **Definition:** The **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.

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

The LIST-THEN-ELIMINATE Algorithm

- The LIST-THEN-ELIMINATE algorithm first initializes the version space to contain all hypotheses in H, then eliminates any hypothesis found inconsistent with any training example.
- The version space of candidate hypotheses thus shrinks as more examples are observed, until ideally just one hypothesis remains that is consistent with all the observed examples
- If insufficient data is available to narrow the version space to a single hypothesis, then the algorithm can output the entire set of hypotheses consistent with the observed data.
- In principle, the LIST-THEN-ELIMINATE algorithm can be applied whenever the hypothesis space H is finite. It has many advantages, including the fact that it is guaranteed to output all hypotheses consistent with the training data.

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 $VersionSpace$
```

TABLE 2.4

The List-Then-Eliminate algorithm.

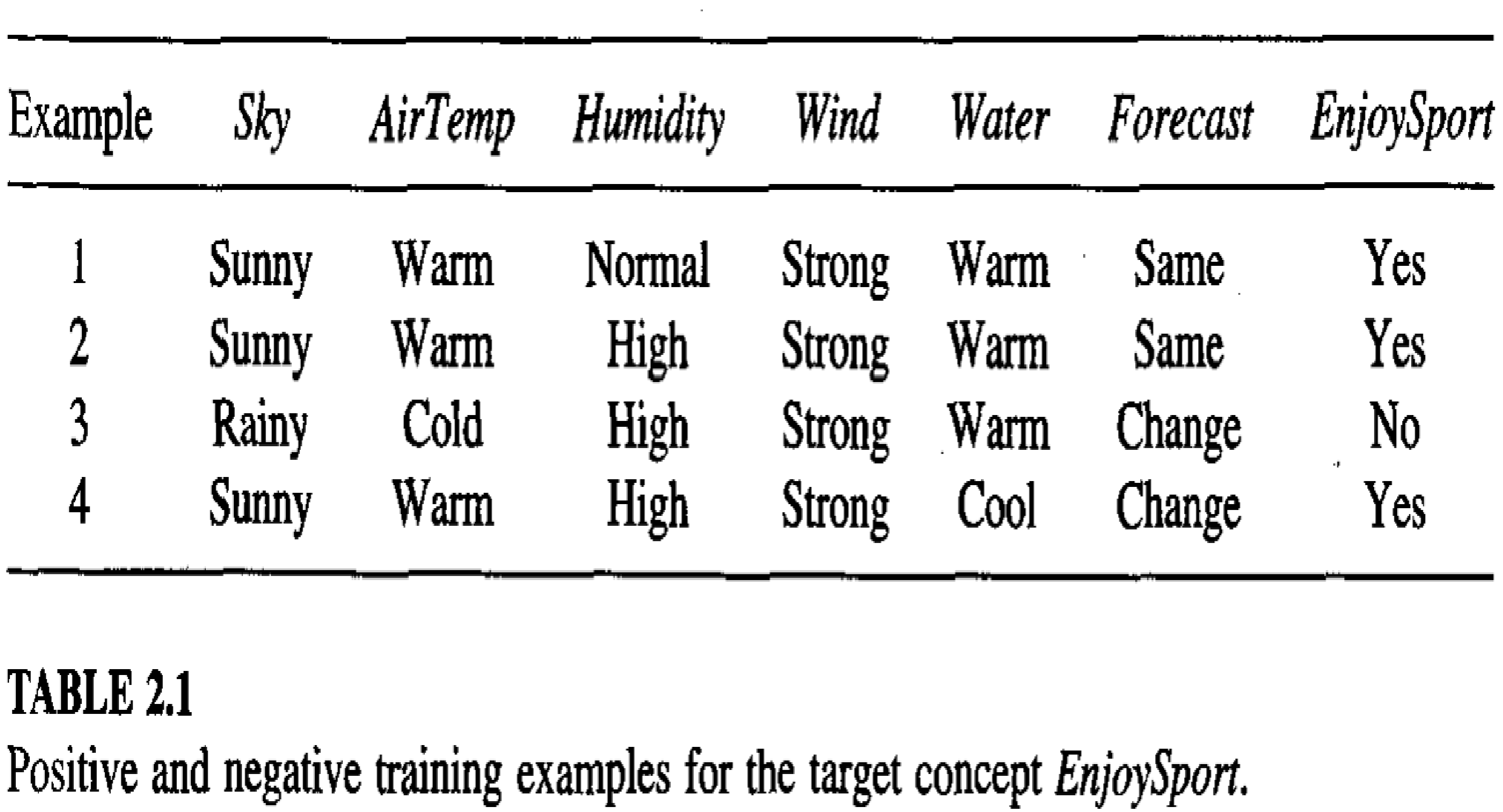
A More Compact Representation for Version Spaces

- The CANDIDATE-ELIMINATION algorithm works on the same principle as the above LIST-THEN-ELIMINATE algorithm.
- However, it employs a much more compact representation of the version space.
- the version space is represented by its most general and least general members.
- These members form general and specific boundary sets that delimit the version space within the partially ordered hypothesis space.

- To illustrate this representation for version spaces, consider the En*joysport* concept learning problem
- Recall that given the four training examples from Table 2.1, FIND-S outputs the hypothesis

  ***h* = (Sunny, Warm, ?, Strong, ?, ?)**
- this is just one of six different hypotheses from H that are consistent with these training examples.
- All six hypotheses are shown in Figure 2.3. They constitute the version space relative to this set of data and this hypothesis representation.
- The arrows among these six hypotheses in Figure 2.3 indicate instances of the *more-general-than* relation.

Example of Concept Learning: EnjoySport



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

TABLE 2.1
Positive and negative training examples for the target concept *EnjoySport*.

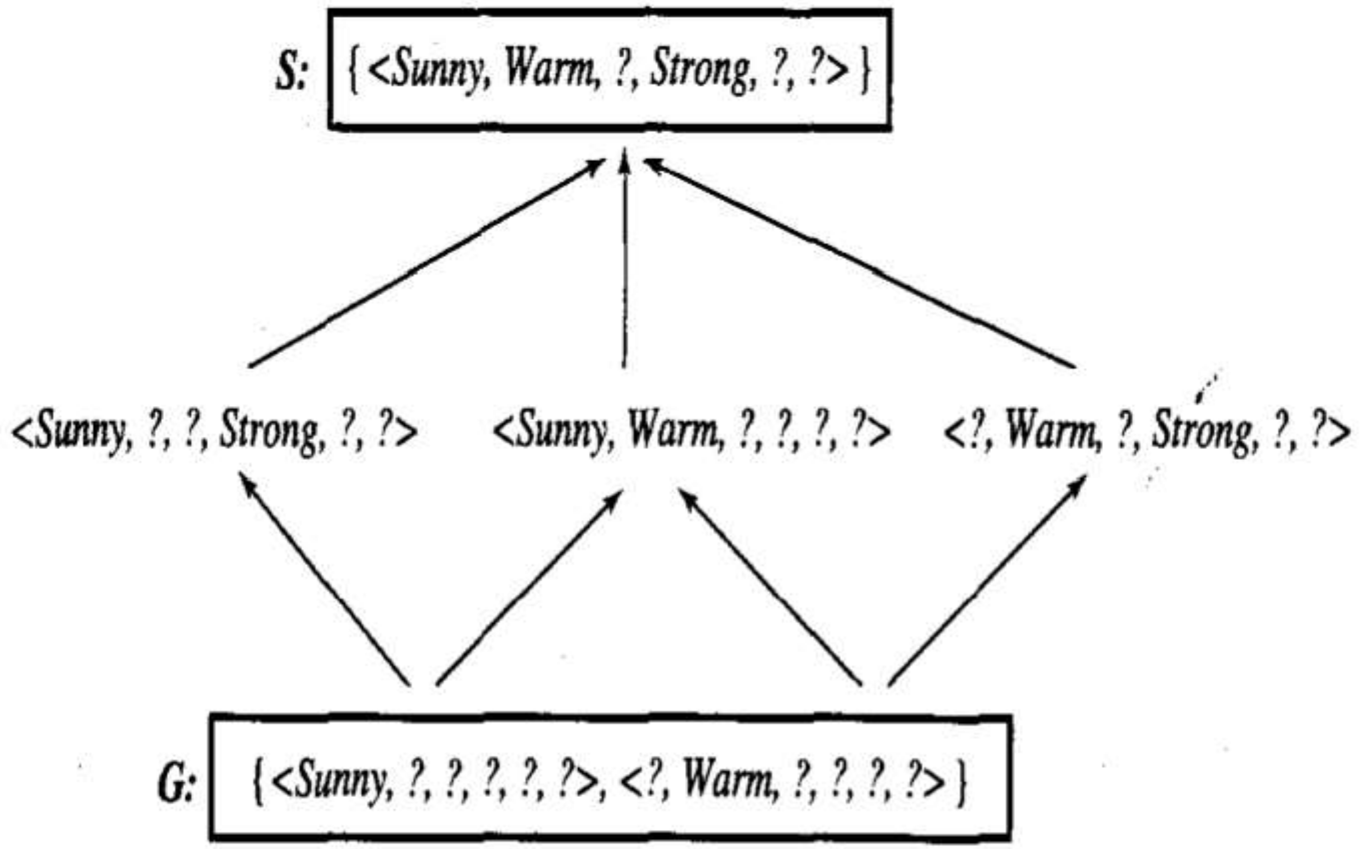


FIGURE 2.3
A version space with its general and specific boundary sets. The version space includes all six hypotheses shown here, but can be represented more simply by S and G. Arrows indicate instances of the *more_general_than* relation. This is the version space for the *EnjoySport* concept learning problem and training examples described in Table 2.1.

- The CE algorithm represents the version space by storing only its most general members (labeled G in Figure 2.3) and its most specific (labeled *S in the figure).*
- *Given only these* two sets *S and G, it is possible to enumerate all members of the version space* as needed by generating the hypotheses that lie between these two sets in the general-to-specific partial ordering over hypotheses.

Boundary sets G and S represent the version space

- **Definition:** 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.

$$G \equiv \{g \in H | Consistent(g, D) \wedge (\neg \exists g' \in H)[(g' >_g g) \wedge Consistent(g', D)]\}$$

- **Definition:** 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.

$$S \equiv \{s \in H | Consistent(s, D) \wedge (\neg \exists s' \in H)[(s >_g s') \wedge Consistent(s', D)]\}$$

- **version space is precisely the set of hypotheses contained in *G, plus those contained in S, plus* those that lie between *G and S in the partially ordered hypothesis space***

CANDIDATE-ELIMINATION Learning 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

$$S_0 \leftarrow \{(0,0,0,0,0,0)\}$$

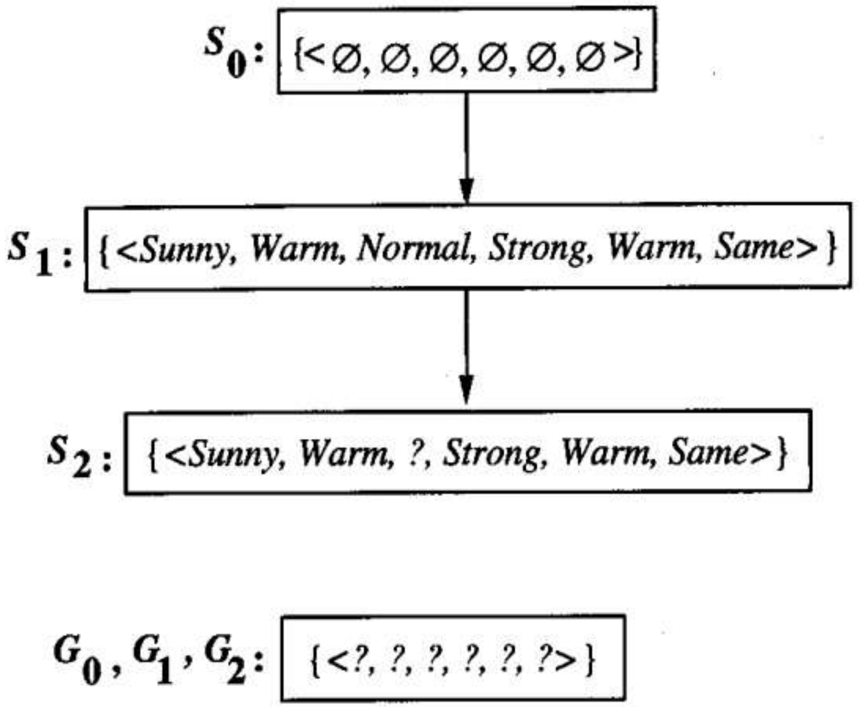
```
Initialize G to the set of maximally general hypotheses in H
Initialize S to the set of maximally specific hypotheses in H
For each training example d, do
• If d is a positive example
        • Remove from G any hypothesis inconsistent with d
        • For each hypothesis s in S that is not consistent 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
```

TABLE 2.5

CANDIDATE-ELIMINATION algorithm using version spaces. Notice the duality in how positive and negative examples influence S and G.

An Illustrative Example

- The boundary sets are first initialized to G_0 and S_0, the most general and most specific hypotheses in H, respectively.
- When the first training example is presented (a positive example in this case), the Candidate Elimination algorithm checks the S boundary and finds that it is overly specific-it fails to cover the positive example.
- The boundary is therefore revised by moving it to the least more general hypothesis that covers this new example. This revised boundary is shown as S_1.
- No update of the G boundary is needed in response to this training example because Go correctly covers this example. When the second training example (also positive) is observed, it has a similar effect of generalizing S further to S_2, leaving G again unchanged (i.e., $G_2 = G_1 = G_0$)


S_0: {<∅, ∅, ∅, ∅, ∅, ∅>}
S_1: {<*Sunny, Warm, Normal, Strong, Warm, Same*>}
S_2: {<*Sunny, Warm, ?, Strong, Warm, Same*>}
G_0, G_1, G_2: {<*?, ?, ?, ?, ?, ?*>}

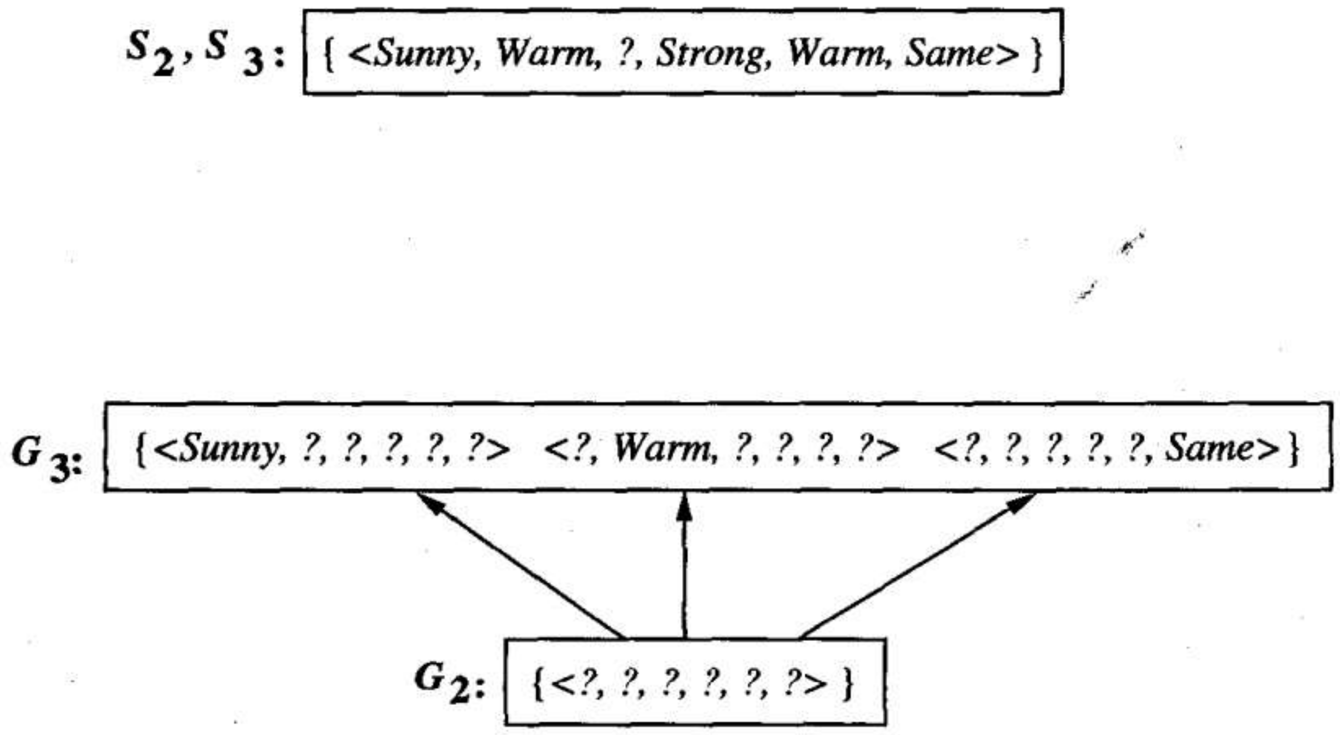
Training examples:

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

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

FIGURE 2.4
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.

- positive training examples may force the **S** boundary of the version space to become increasingly general.
- Negative training examples play the complimentary role of forcing the **G** boundary to become increasingly specific.
- Consider the third training example, shown in Figure 2.5. This negative example reveals that the **G** boundary of the version space is overly general; that is,
- the hypothesis in **G** incorrectly predicts that this new example is a positive example.
- The hypothesis in the **G** boundary must therefore be specialized until it correctly classifies this new negative example.
- As shown in Figure *2.5,* there are several alternative minimally more specific hypotheses. All of these become members of the new **G3** boundary set.



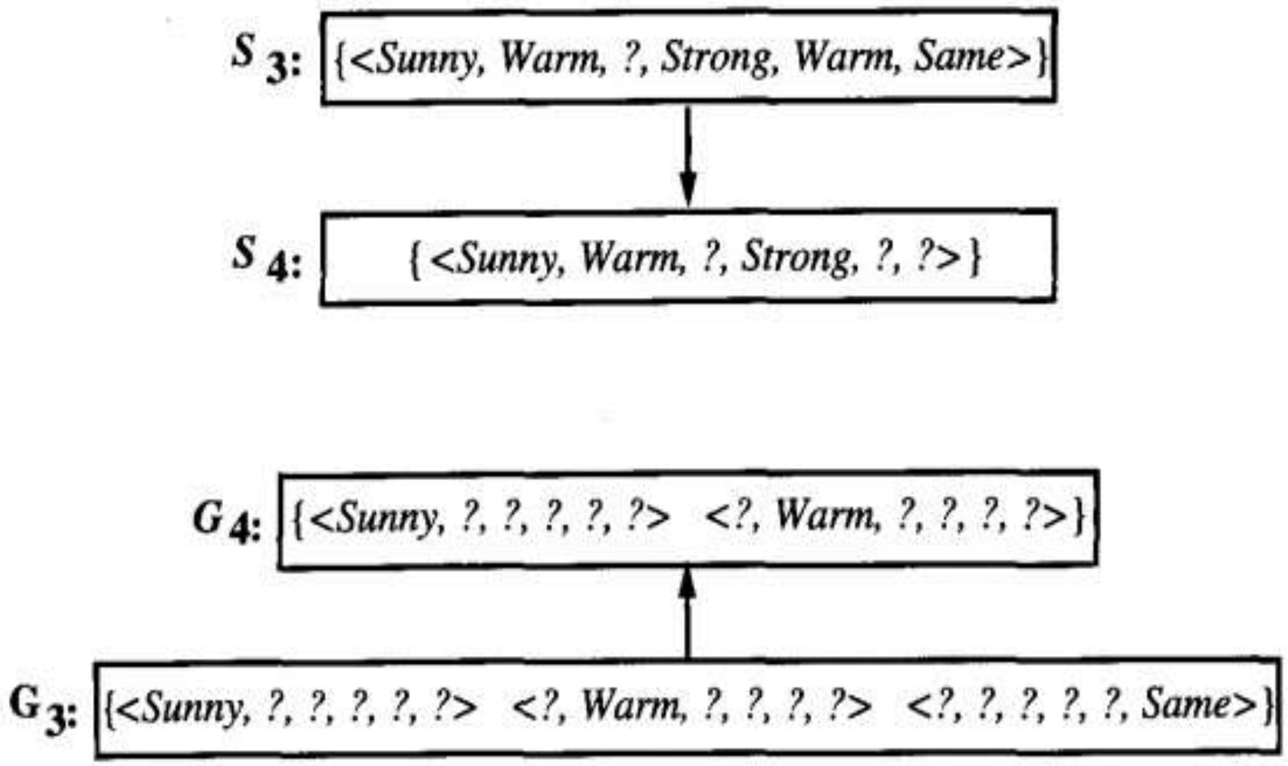
Training Example:

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

FIGURE 2.5
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.

- Given that there are six attributes that could be specified to specialize G2, why are there only three new hypotheses in G3? For example, the hypothesis h =(?,?,Normal,?,?, ?) is a minimal specialization of G2 that correctly labels the new example as a negative example, but it is not included in G3.
- The reason this hypothesis is excluded, it is inconsistent with the previously encountered positive examples.
- The algorithm determines this simply by noting that h is not more general than the current specific boundary, S2.

- The fourth training example, as shown in Figure 2.6, further generalizes the S boundary of the version space. It also results in removing one member of the G boundary, because this member fails to cover the new positive example.
- 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.
- The entire version space, including those hypotheses bounded by S4 and G4, is shown in Figure 2.7. This learned version space is independent of the sequence in which the training examples are presented.



Training Example:

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

FIGURE 2.6
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.

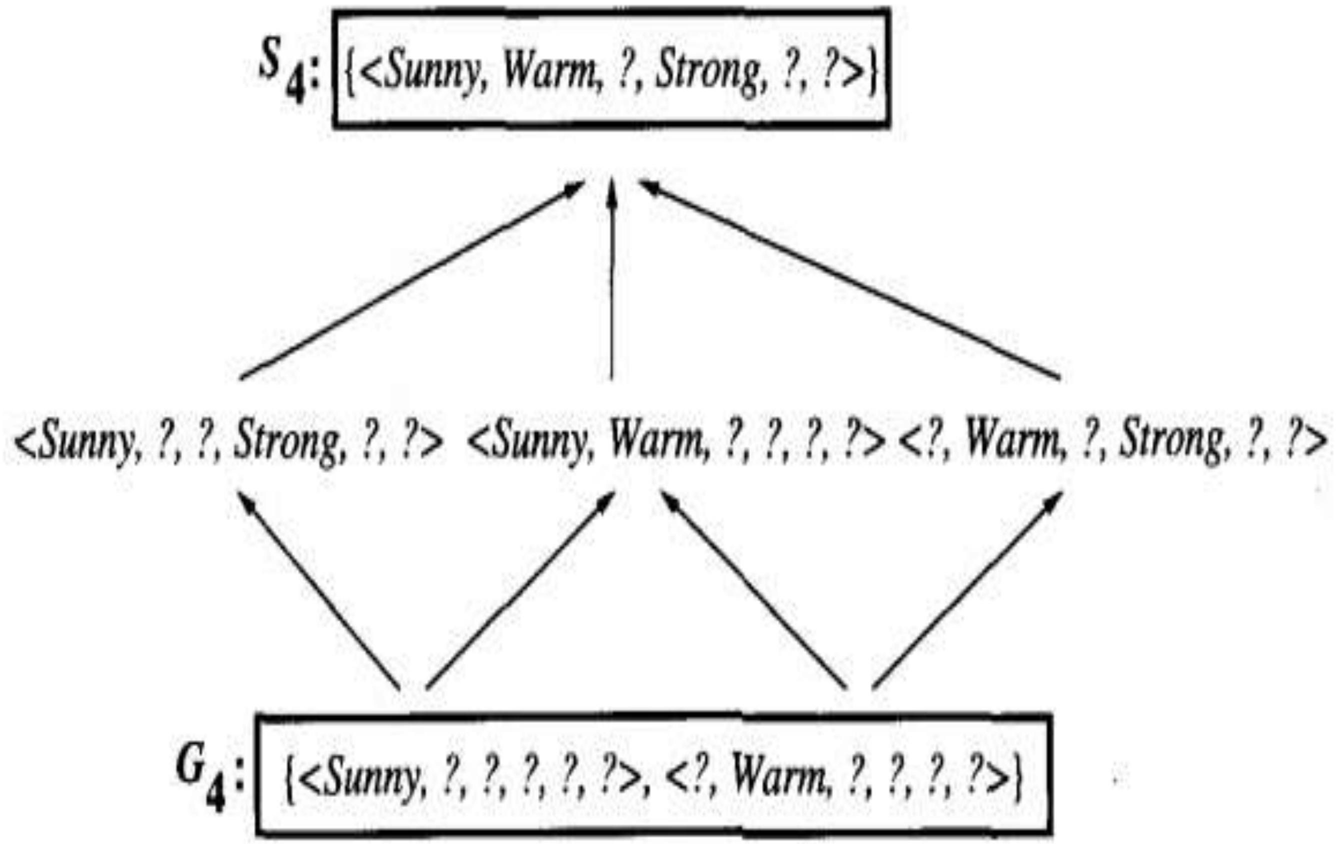
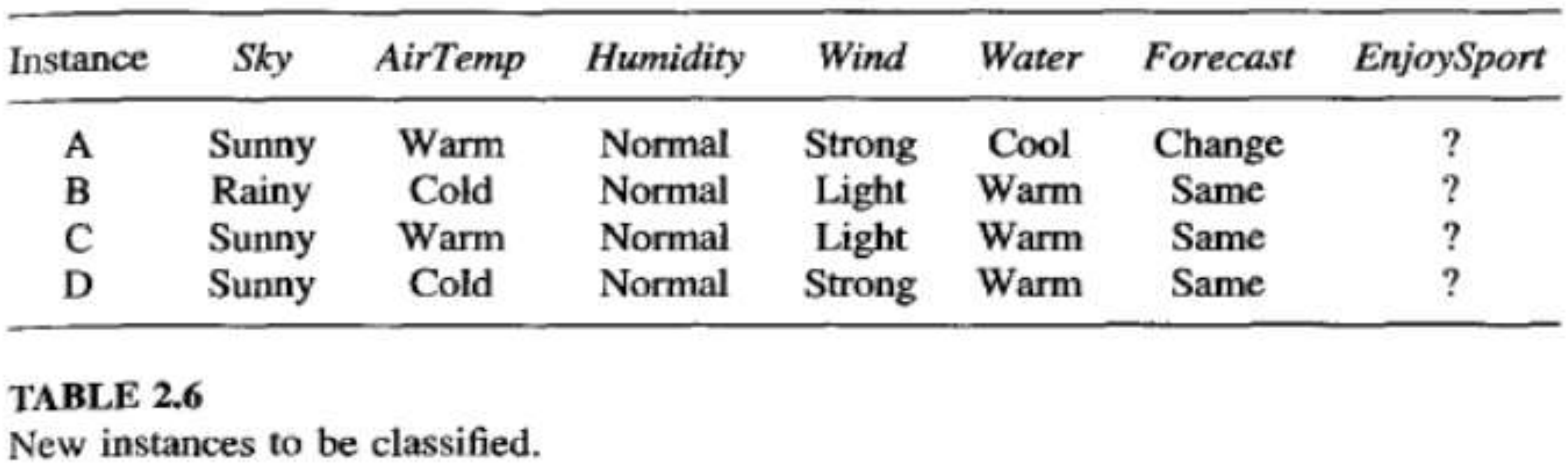


FIGURE 2.7
The final version space for the *EnjoySport* concept learning problem and training examples described earlier.

How Can Partially Learned Concepts Be Used?



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	?

TABLE 2.6
New instances to be classified.

- A was not among the training examples, it is classified as a positive instance by ***every hypothesis in the current version space***
- 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

- Instance C presents a different situation. Half of the version space hypotheses classify it 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,
- one approach we could take would be to output the majority vote, perhaps with a confidence rating indicating how close the vote was.

INDUCTIVE BIAS

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.

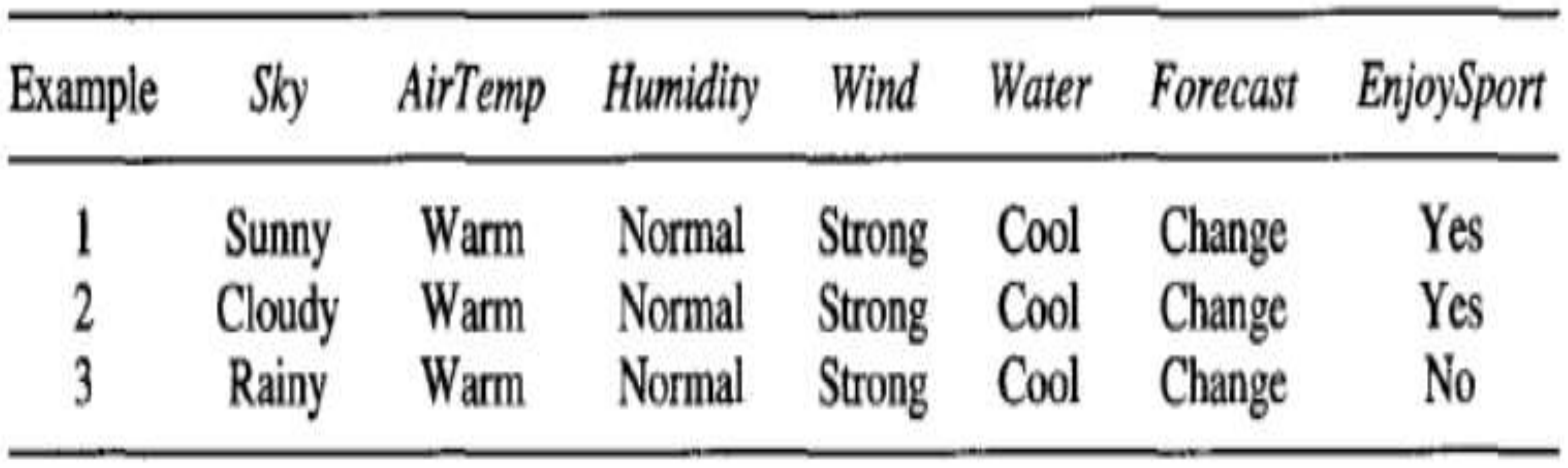
- What if the target concept is not contained in the hypothesis space?
- Can we avoid this difficulty by using a hypothesis space that includes every possible hypothesis?
- How does the size of this hypothesis space influence the ability of the algorithm to generalize to unobserved instances?
- How does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

These are fundamental questions for inductive inference in general.

A Biased Hypothesis Space

- Suppose we wish to assure that the hypothesis space contains the unknown target concept. solution is to enrich the hypothesis space to include ***every possible*** hypothesis.

- To illustrate, consider again the ***EnjoySport*** example in which 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."***
- given the following three training examples of this disjunctive hypothesis, our algorithm would find that there are zero hypotheses in the version space



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

- the most specific hypothesis consistent with the first two examples in the given hypothesis space H is
  S2 : (?, ***Warm, Normal, Strong, Cool, Change)***
- although it is the maximally specific hypothesis from H that is consistent with the first two examples, it incorrectly covers the third (negative) training example.
- The problem is that we have **biased the learner to consider only conjunctive hypotheses.** In this case we require a more expressive hypothesis space.

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
- that is, it is capable of representing every possible subset of the instances X.
- In general, the set of all subsets of a set X is called ***the power set*** of X.
- In the ***EnjoySport*** learning task, for example, the size of the instance space X of days described by the six available attributes is 96.
- How many possible concepts can be defined over this set of instances? how large is the power set of X?
- Thus, there are $\mathbf{2^{96}}$ or 10^{28} approximately distinct target concepts that could be defined over this instance space and that our learner might be called upon to learn.

An Unbiased Learner

- Let us reformulate the Enjoysport learning task in an unbiased way by defining a new hypothesis space H' that can represent every subset of instances; let H' correspond to the power set of X.
- One way to define such an H' is to allow arbitrary disjunctions, conjunctions, and negations of our earlier hypotheses.
- For instance, the target concept "Sky = Sunny or Sky = Cloudy" could then be described as

$$\langle Sunny, ?, ?, ?, ?, ?\rangle \vee \langle Cloudy, ?, ?, ?, ?, ?\rangle$$

- Given this hypothesis space, we can use the CANDIDATE-ELIMINATION algorithm without worrying that the target concept might not be expressible.
- However, while this hypothesis space eliminates any problems of expressibility,
- it unfortunately raises a new, equally difficult problem:

An Unbiased Learner

- **our concept learning algorithm is now completely unable to generalize beyond the observed examples!**
- To see why, if we present three positive examples (x1, x2, x3) and two negative examples (x4, x5) to the learner.
- the S boundary of the version space will contain the hypothesis which is just the disjunction of the positive examples

$$S : \{(x_1 \vee x_2 \vee x_3)\}$$

- Similarly, the G boundary will consist of the hypothesis that rules out only the observed negative examples.

$$G : \{\neg(x_4 \vee x_5)\}$$

- The problem here is that with this representation, the S boundary will always be simply the disjunction of the observed positive examples, while the G boundary will always be the negated disjunction of the observed negative examples.
- Therefore, the only examples that will be unambiguously classified by S and G are the observed training examples themselves. In order to converge to a single, final target concept, we will have to present every single instance in X as a training example

- inductive learning requires some form of prior assumptions, or inductive bias,
- The key idea we wish to capture here is the policy by which the learner generalizes beyond the observed training data, to infer the classification of new instances.

The Futility of Bias-Free Learning

- The key idea we wish to capture here is the policy by which the learner generalizes beyond the observed training data, to infer the classification of new instances.
- Consider the general setting in which an arbitrary learning algorithm L is provided an arbitrary set of training data Dc = {(x, c(x))} of some arbitrary target concept c. After training, L is asked to classify a new instance xi.
- Let L(xi, Dc) denote the classification (e.g., positive or negative) that L assigns to xi after learning from the training data Dc. We can describe this inductive inference step performed by L as follows

$$(D_c \wedge x_i) \succ L(x_i, D_c)$$

where the notation $y \succ z$ indicates that z is inductively inferred from y.

- For example, if we take L to be the CANDIDATE-ELIMINATION algorithm, D, to be the training data from Table 2.1, and xi to be the fist instance from Table 2.6,
- then the inductive inference performed in this case concludes that L(xi, Dc) = (EnjoySport = yes).
- Because L is an inductive learning algorithm, the result L(xi, Dc) that it infers will not in general be provably correct

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	?

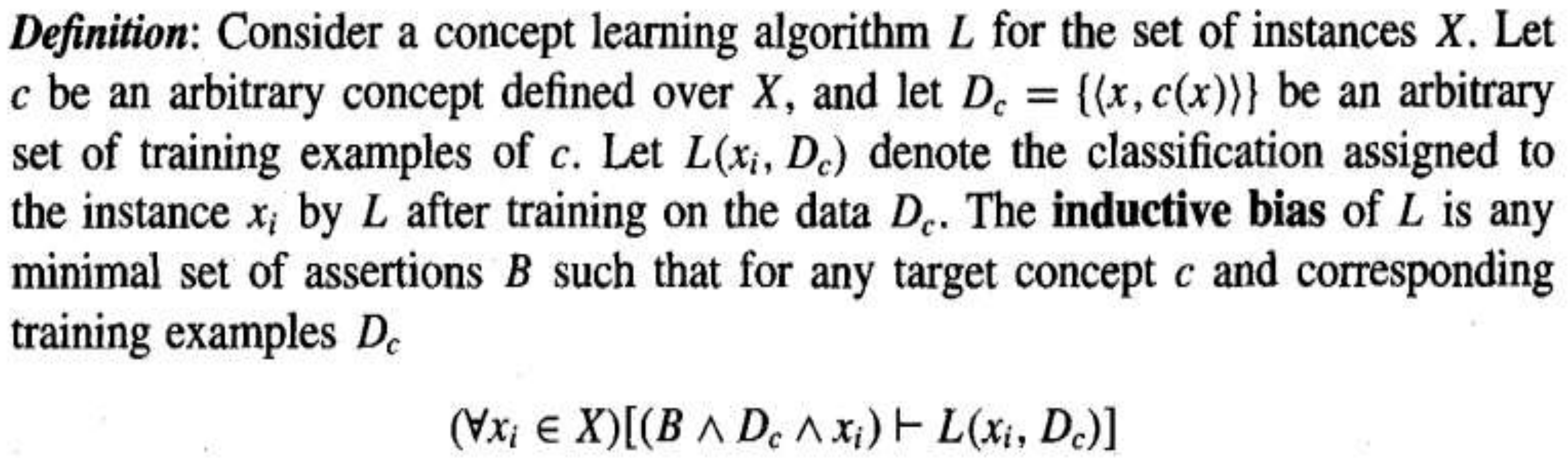
TABLE 2.6
New instances to be classified.

- However, what additional assumptions could be added to $D_c \wedge x_i$ so that L(xi, D,) would follow deductively.

- More precisely, we define inductive bias of L to be the set of assumptions B such that for all new instances xi

$$(B \wedge D_c \wedge x_i) \vdash L(x_i, D_c)$$

where the notation $y \vdash z$ indicates that z follows deductively from y (i.e., that z is provable from y). Thus, we define the inductive bias of a learner as the set of additional assumptions B sufficient to justify its inductive inferences as deductive inferences.



Definition: Consider a concept learning algorithm L for the set of instances X. Let c be an arbitrary concept defined over X, and let $D_c = \{\langle x, c(x)\rangle\}$ 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

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

- To see why the classification L(xi, Dc) follows deductively from B = {c є H), together with the data D_c and description of the instance xi, consider the following argument.
- First, if we assume c є H then it follows deductively that c є $VS_{H,Dc}$. This follows from c є H, from the definition of the version space $VS_{H,Dc}$ as the set of all hypotheses in H that are consistent with Dc, and from our definition of Dc = {(x, c(x))} as training data consistent with the target concept c.
- Second, we defined the classification L(xi, Dc) to be the unanimous vote of all hypotheses in the version space. Thus, if L outputs the classification L(xi, Dc), it must be the case the every hypothesis in $VS_{H,Dc}$ also produces this classification, including the hypothesis c є $VS_{H,Dc}$.

c(xi) = L(xi, Dc)

- To summarize, the CANDIDATE-ELIMINATION algorithm defined in this fashion can be characterized by the following bias.

  Inductive bias of CANDIDATE-ELIMINATION algorithm: The target concept c is contained in the given hypothesis space H.
- Figure summarizes the situation schematically.
- The inductive CANDIDATE-ELIMINATION algorithm at the top of the figure takes two inputs: the training examples and a new instance to be classified.
- At the bottom of the figure, a deductive theorem prover is given these same two inputs plus the assertion "H contains the target concept." These two systems will in principle produce identical outputs for every possible input set of training examples and every possible new instance in X.

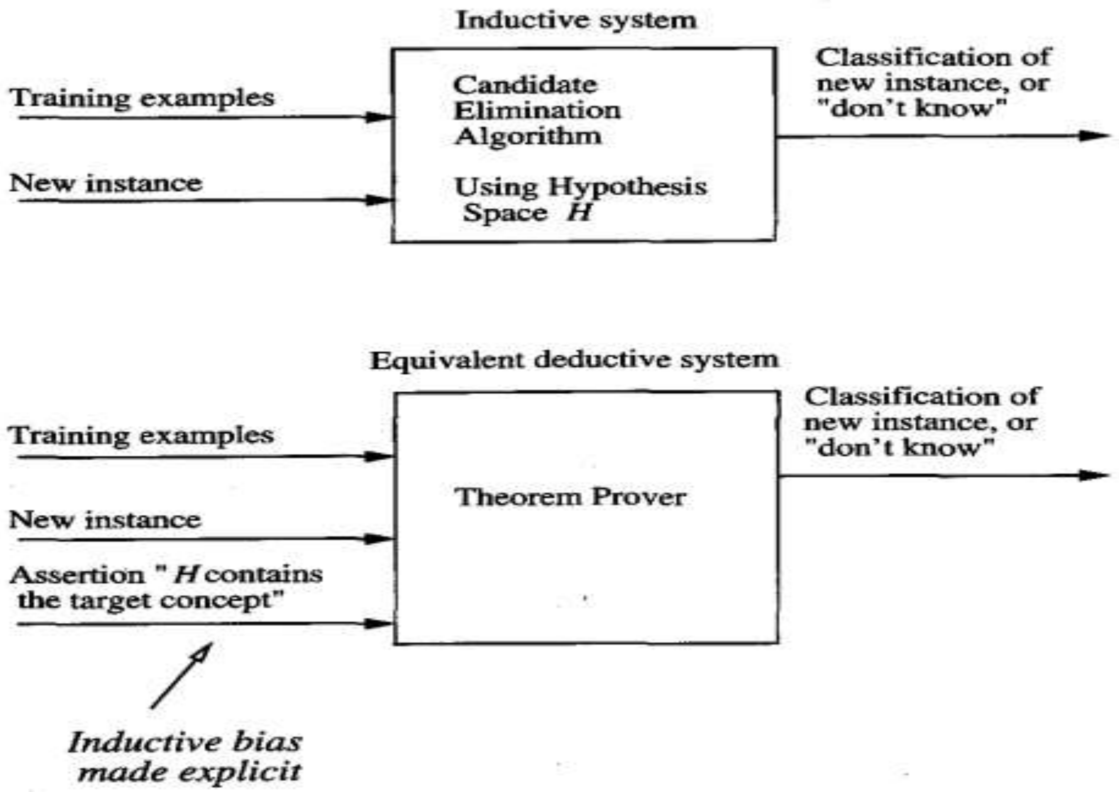

Inductive system
Training examples
New instance
Candidate Elimination Algorithm
Using Hypothesis Space H
Classification of new instance, or "don't know"
Equivalent deductive system
Training examples
New instance
Assertion " H contains the target concept"
Theorem Prover
Classification of new instance,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

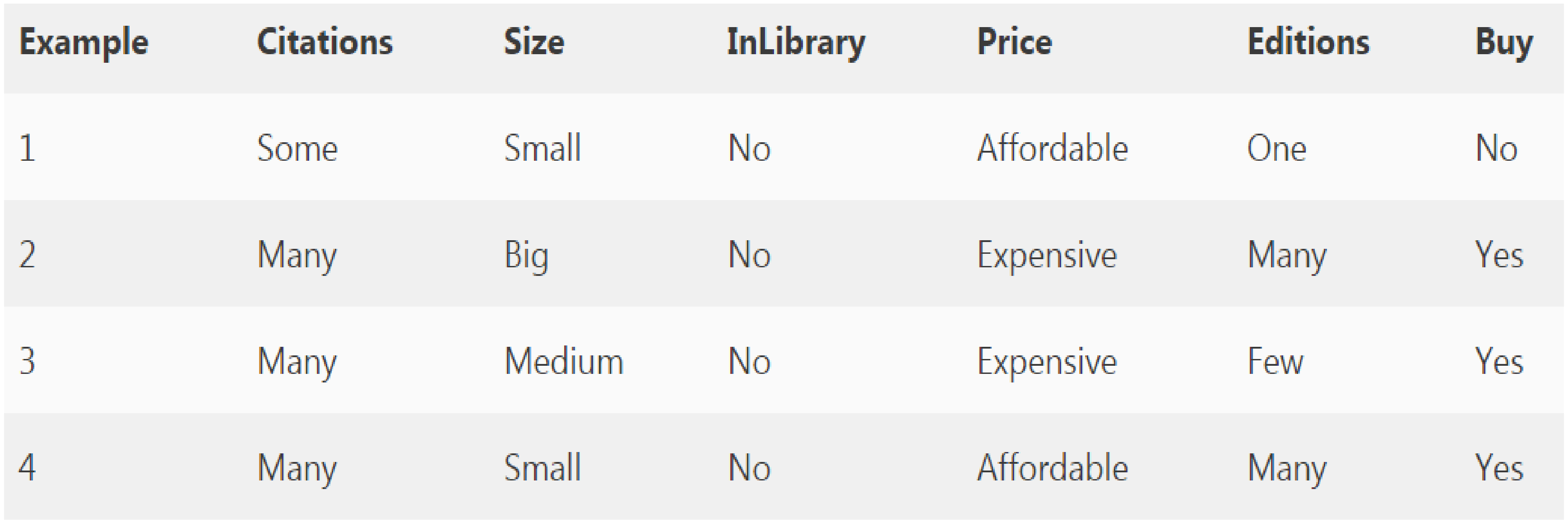
FIGURE :

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.

- Consider, for example, the following three learning algorithms, which are listed from weakest to strongest bias.

1. **ROTE-LEARNER**: Learning corresponds simply to storing each observed training example in memory. Subsequent instances are classified by looking them up in memory. If the instance is found in memory, the stored classification is returned. Otherwise, the system refuses to classify the new instance.
2. **CANDIDATE-ELIMINATION algorithm**: New instances are classified only in the case where all members of the current version space agree on the classification. Otherwise, the system refuses to classify the new instance.
3. **FIND-S**: finds the most specific hypothesis consistent with the training examples. It then uses this hypothesis to classify all subsequent instances.

Example 1



Example	Citations	Size	InLibrary	Price	Editions	Buy
1	Some	Small	No	Affordable	One	No
2	Many	Big	No	Expensive	Many	Yes
3	Many	Medium	No	Expensive	Few	Yes
4	Many	Small	No	Affordable	Many	Yes

S0: (0, 0, 0, 0, 0) Most Specific Boundary

G0: (?, ?, ?, ?, ?) Most Generic Boundary

- The first example is negative, the hypothesis at the specific boundary is consistent, hence we retain it, and the hypothesis at the generic boundary is inconsistent hence we write all consistent hypotheses by removing one “?” at a time.
- { <Some,Small,No,Affordable,One> No}
- S1: (0, 0, 0, 0, 0)
- G1: (Many,?,?,?, ?) (?, Big,?,?,?) (?,Medium,?,?,?) (?,?,?,Exp,?) (?,?,?,?,Few)

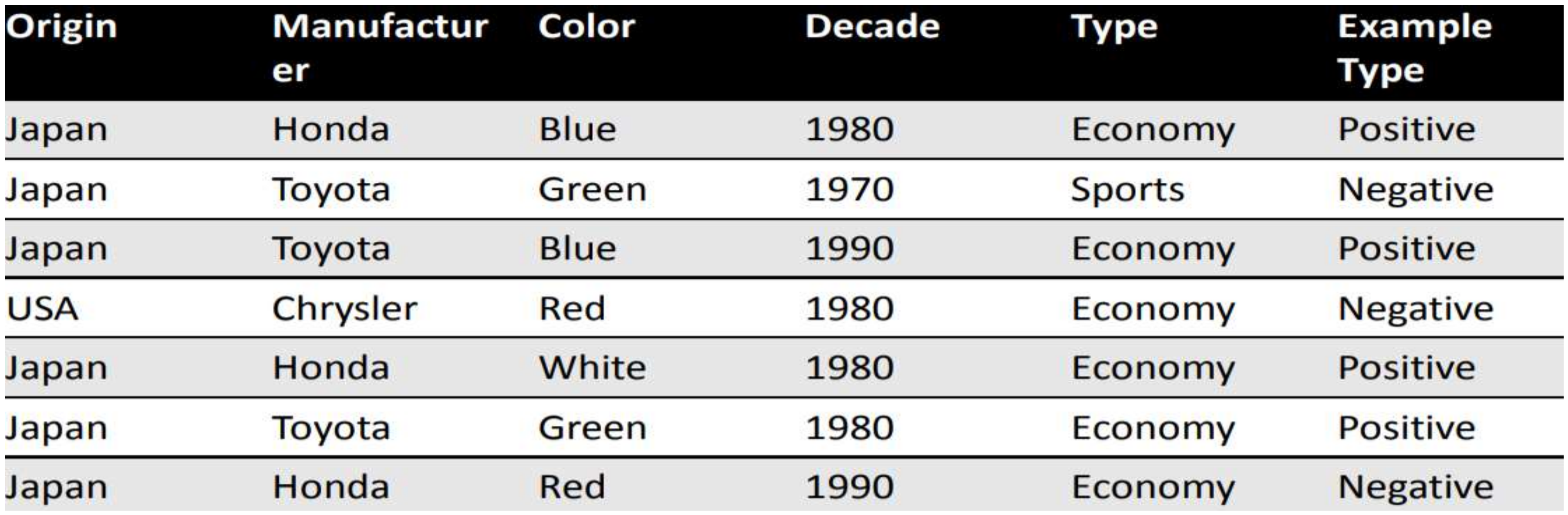
- S2: (Many, Big, No, Exp, Many)
- G2: (Many,?,?,?, ?) (?, Big,?,?,?) (?,?,?,Exp,?) (?,?,?,?,Many)
- The third example is positive, the hypothesis at the specific boundary is inconsistent, hence we extend the specific boundary, and the consistent hypothesis at the generic boundary is retained and inconsistent hypotheses are removed from the generic boundary.

- S3: (Many, ?, No, Exp, ?)
- G3: (Many,?,?,?,?) (?,?,?,exp,?)

- S4: (Many, ?, No, ?, ?)
- G4: (Many,?,?,?,?)

(Many, ?, No, ?, ?) (Many, ?, ?, ?, ?)

Example 2



Origin	Manufacturer	Color	Decade	Type	Example Type
Japan	Honda	Blue	1980	Economy	Positive
Japan	Toyota	Green	1970	Sports	Negative
Japan	Toyota	Blue	1990	Economy	Positive
USA	Chrysler	Red	1980	Economy	Negative
Japan	Honda	White	1980	Economy	Positive
Japan	Toyota	Green	1980	Economy	Positive
Japan	Honda	Red	1990	Economy	Negative

G = { (?, ?, ?, ?, ?) }

S = { (Japan, Honda, Blue, 1980, Economy) }

G = { (?, Honda, ?, ?, ?), (?, ?, Blue, ?, ?), (?, ?, ?, 1980, ?), (?, ?, ?, ?, Economy) }
S = { (Japan, Honda, Blue, 1980, Economy) }

G = { (?, ?, Blue, ?, ?), (?, ?, ?, ?, Economy) }
S = { (Japan, ?, Blue, ?, Economy) }

Positive Example 1
(Japan, Honda, Blue, 1980, Economy)

- Initialize G to a singleton set that includes everything.

  G = { (?, ?, ?, ?, ?) }

- Initialize S to a singleton set that includes the first positive example.

  S = { (Japan, Honda, Blue, 1980, Economy) }

Negative Example 2
(Japan, Toyota, Green, 1970, Sports)

- Specialize G to exclude the negative example.

G = { (?, Honda, ?, ?, ?), (?, ?, Blue, ?, ?), (?, ?, ?, 1980, ?), (?, ?, ?, ?, Economy) }

S = { (Japan, Honda, Blue, 1980, Economy) }

Positive Example 3 (Japan, Toyota, Blue, 1990, Economy)

- Prune G to exclude descriptions inconsistent with the positive example.

G = { (?, ?, Blue, ?, ?), (?, ?, ?, ?, Economy) }

Generalize S to include the positive example.

S = { (Japan, ?, Blue, ?, Economy) }

4 Negative Example (USA, Chrysler, Red, 1980, Economy)

- Specialize G to exclude the negative example (but stay consistent with S)

G = { (?, ?, Blue, ?, ?), (Japan, ?, ?, ?, Economy) }

S = { (Japan, ?, Blue, ?, Economy) }

5 Positive Example (Japan, Honda, White, 1980, Economy)

- Prune G to exclude descriptions inconsistent with positive example.

  G = { (Japan, ?, ?, ?, Economy) }

- Generalize S to include positive example.

  S = { (Japan, ?, ?, ?, Economy) }

6 Positive Example: (Japan, Toyota, Green, 1980, Economy)

- New example is consistent with version-space, so no change is made.

G = { (Japan, ?, ?, ?, Economy) }

S = { (Japan, ?, ?, ?, Economy) }

7 Negative Example: (Japan, Honda, Red, 1990, Economy)

- Example is inconsistent with the version-space. G cannot be specialized. S cannot be generalized.
- The version space collapses.
- Conclusion: No conjunctive hypothesis is consistent with the data set.