Concept Learning

Concept Learning

- Learning involves acquiring general concepts from specific training examples.
- Alternatively, each concept can be thought of as a Boolean-valued function defined over this larger set. (Example: A function defined over all animals, whose value is true for birds and false for other animals).
- Concept learning can be formulated as a problem of searching through a predefined space of potential hypotheses for the hypothesis that best fits the training 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 Aldo enjoys his favorite water sport."

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

Positive and negative training examples for the target concept ***EnjoySport***

The attribute ***EnjoySport*** indicates whether a Person 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 ?

What hypothesis representation is provided to the learner?

Let's consider a simple representation in which each hypothesis consists of a conjunction of constraints on the instance attributes.

Let 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 "Φ" 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 PERSON enjoys his favorite sport only on cold days with high humidity (independent of the values of the other attributes) is represented by the expression

(?, Cold, High, ?, ?, ?)

(?, Warm,?, ?, ?, ?)

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

(Φ , Φ, Φ, Φ, Φ, Φ)

Notation

- The set of items over which the concept is defined is called the set of ***instances***, denoted by X.

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***, denoted by c.

c can be any Boolean valued function defined over the instances X;

c : X→{O, 1}

The target concept corresponds to the value of the attribute ***EnjoySport***

- (i.e., c(x) = 1 if ***EnjoySport*** = Yes, and c(x) = 0 if ***EnjoySport*** = No).

- 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 non-members of the target concept.
- The ordered pair (x, c(x)) to describe the training example consisting of the instance x and its target concept value c(x).
- ***D*** to denote the set of available training examples

- The symbol ***H*** to denote the set of all possible hypotheses that the learner may consider regarding the identity of the target concept. Each hypothesis ***h*** in ***H*** represents a Boolean-valued function defined over X
  - **h : X→{O, 1}**

- The goal of the learner is to find a hypothesis h such that h(x) = c(x) for all x in ***X***.

- **Given:**
  - Instances X: Possible days, each described by the attributes
    - *Sky* (with possible values *Sunny*, *Cloudy*, and *Rainy*),
    - *AirTemp* (with values *Warm* and *Cold*),
    - *Humidity* (with values *Normal* and *High*),
    - *Wind* (with values *Strong* and *Weak*),
    - *Water* (with values *Warm* and *Cool*), and
    - *Forecast* (with values *Same* and *Change*).
  - Hypotheses H: Each hypothesis is described by a conjunction of constraints on the attributes *Sky*, *AirTemp*, *Humidity*, *Wind*, *Water*, and *Forecast*. The constraints may be "?" (any value is acceptable), "$\emptyset$" (no value is acceptable), or a specific value.
  - Target concept c: $EnjoySport : X \rightarrow \{0, 1\}$
  - Training examples D: Positive and negative examples of the target function (see Table 2.1).
- **Determine:**
  - A hypothesis h in H such that $h(x) = c(x)$ for all x in X.

TABLE The *EnjoySport* concept learning task.

Inductive Learning Hypothesis

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

Concept Learning as Search

- 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.

Example, the instances ***X*** and hypotheses ***H*** in the ***EnjoySport*** learning task.
The attribute ***Sky*** has three possible values, and ***AirTemp, Humidity, Wind, Water Forecast*** each have two possible values
The instance space X contains exactly

3.2.2.2.2.2 = 96 (Distinct instances)

• If we add the other two values tha attribute can take, then

The instance space X contains exactly

5.4.4.4.4.4 = 5120

(Syntactically distinct hypotheses within H.)

• Every hypothesis containing one or more " Φ" symbols represents the empty set of instances; that is, it classifies every instance as ***negative.***

Hence, the instance space X contains

1 + (4.3.3.3.3.3) = 973

(Semantically distinct hypotheses)

General-to-Specific Ordering of Hypotheses

- Many algorithms for concept learning organize the search through the hypothesis space by relying on a very useful structure that exists for any concept learning problem: a general-to-specific ordering of hypotheses.
- By taking advantage of this naturally occurring structure over the hypothesis space, we can design learning algorithms that exhaustively search even infinite hypothesis spaces without explicitly enumerating every hypothesis.

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

h_1 = (Sunny, ?, ?, Strong, ?, ?)

h_2 = (Sunny, ?, ?, ?, ?, ?)

- Consider the sets of instances that are classified positive by h_1 and by h_2.
- h_2 imposes fewer constraints on the instance, it classifies more instances as positive. So, any instance classified positive by h_1 will also be classified positive by h_2. Therefore, h_2 is more general than h_1.

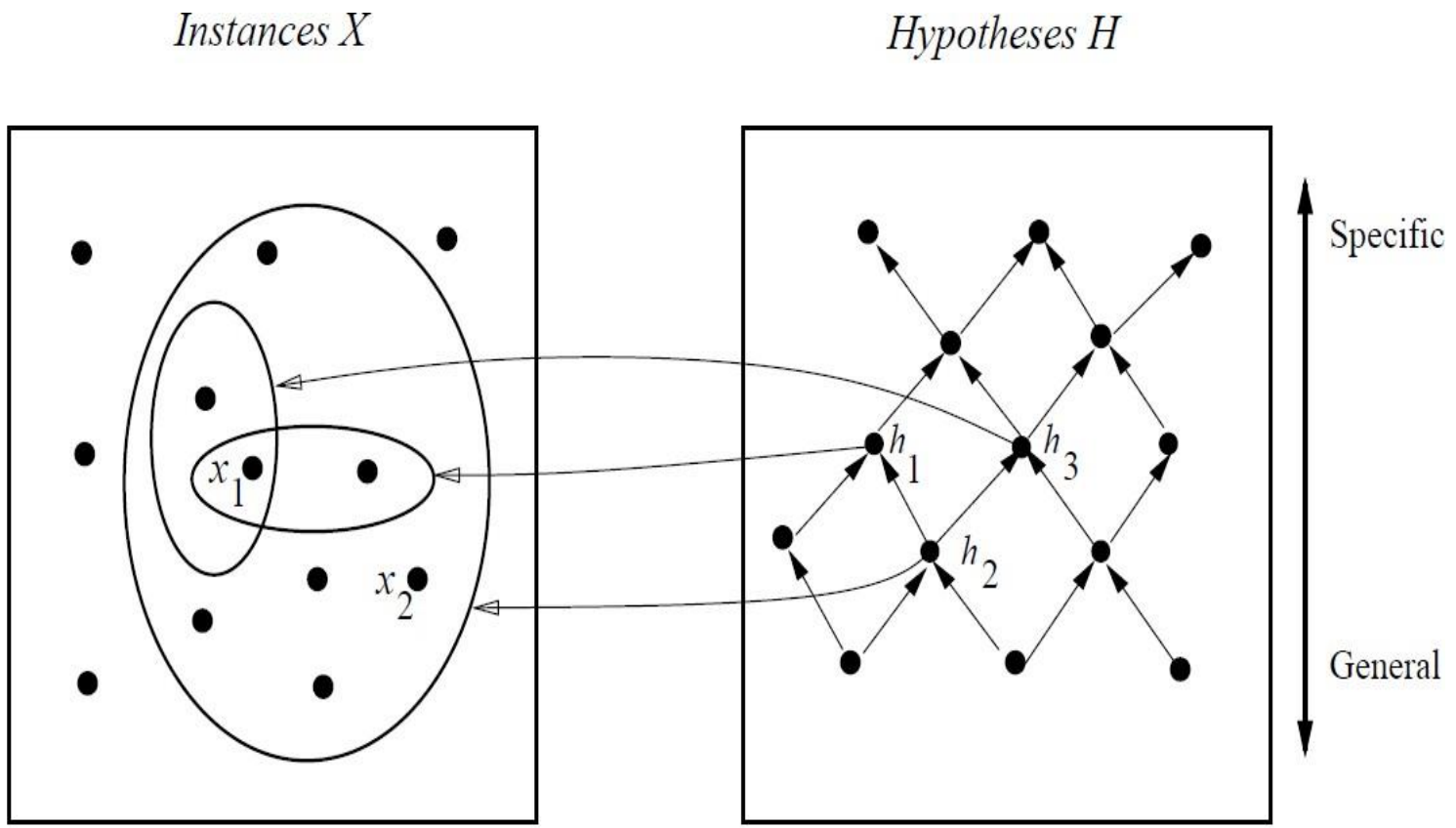
- Given hypotheses ***h_j*** and ***h_k***,

h_j is more-general-than or- equal to ***h_k*** if and only if any instance that satisfies ***h_k*** also satisfies ***h_j***

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 $h_j \geq h_k$) if and only if

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

- In the figure, the box on the left represents the set X of all instances, the box on the right the set H of all hypotheses.
- Each hypothesis corresponds to some subset of X-the subset of instances that it classifies positive.
- The arrows connecting hypotheses represent the ***more_general_than*** relation, with the arrow pointing toward the less general hypothesis.
- Note the subset of instances characterized by h_2 subsumes the subset characterized by h_1, hence h_2 is more - general– than h_1



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, ?>

FIND-S: Finding a Maximally Specific Hypothesis

- **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*

To illustrate this algorithm, assume the learner is given the sequence of training examples from the *EnjoySport* task

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

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

$$\boldsymbol{H} \leftarrow (\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$$

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

•Observing the first training example, it is clear that our hypothesis is too specific. In particular, none of the "Ø" constraints in h are satisfied by this example, so each is replaced by the next more general constraint that fits the example

- ***h_1 = <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

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

- The second training example 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

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

- *x3 = <Rainy, Cold, High, Strong, Warm, Change>, -*

- Upon encounteringthe third training the algorithm makes no change to h. The FIND-S algorithm simply ignores every negative example.

- ***h3 = < Sunny,Warm ? Strong, Warm, Same>***

- *x4 = <Sunny, Warm, High, Strong, Cool, Change>, +*

- The fourth example leads to a further generalization of h

- ***h4 = < Sunny, Warm, ?, Strong, ?,? >***

Training example 1

Example	Time of day	Weather	Temp	Company	Humidity	Wind	Play Tennis
1	Morning	Sunny	Warm	Yes	Mild	Strong	Yes
2	Evening	Rainy	Cold	No	Mild	Normal	No
3	Morning	Sunny	Moderate	Yes	Normal	Normal	Yes
4	Evening	Sunny	Cold	Yes	High	Strong	Yes

Training Example 2

Example	Toothed	Legs	Species
1	True	True	Mammal
2	True	False	Reptile
3	False	True	Mammal
4	False	True	Reptile

Training Example 3

Example	Body Temp	Gives Birth	Four Legged	Hibernates	Class label
1	Warm	Yes	Yes	Yes	Yes
2	Warm	Yes	Yes	No	Yes
3	Warm	No	No	No	No
4	Cold	No	No	No	No
5	Cold	Yes	No	No	No
6	Warm	Yes	No	Yes	Yes
7	Warm	Yes	No	No	Yes

Training Example 4

Training Example	N (running nose)	C (coughing)	R (reddened skin)	F (fever)	**Classification**
d_1	+	+	+	−	positive (ill)
d_2	+	+	−	−	positive (ill)
d_3	−	−	+	+	positive (ill)
d_4	+	−	−	−	negative (healthy)
d_5	−	−	−	−	negative (healthy)
d_6	−	+	+	−	negative (healthy)

Find-S: Key properties

- FIND-S is guaranteed to output the most specific hypothesis within H that is consistent with the positive training examples
- FIND-S algorithm's final hypothesis will also be consistent with the negative examples provided
  - the correct target concept is contained in H,
  - the training examples are correct.
- Yet there are several which are unanswered

Unanswered Questions by Find-S

1. Has the learner converged to the correct target concept?
   - It is not clear whether the hypothesis found is the only hypothesis consistent with the data
2. 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
3. Are the training examples consistent?
   - the training examples may 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. We would prefer an algorithm that could at least detect when the training data is inconsistent and, preferably, accommodate such errors.
4. What if there are several maximally specific consistent hypotheses?
   - there can be several maximally specific hypotheses consistent with the data.

Version Spaces and the Candidate Elimination Algorithm

Consistent Hypothesis

Definition: 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.***

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

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

1. **<?, Warm, ?, Strong, ?, ?>**
2. **<?, ?, ?, ?, ?, ?>**
3. **<Sunny, ?, ?, Strong, ?, ?>**
4. **<Sunny, Warm, ?, Strong, ?, ?>**

Version Space

A representation of the set of all hypotheses which are consistent with D

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

1. VersionSpace c *a* list containing every hypothesis in H

2. For each training example, ***(x, c(x))***

    remove from ***VersionSpace*** any hypothesis ***h*** for which ***h(x) ≠ c(x)***

3. Output the list of hypotheses in ***VersionSpace***

- **List-Then-Eliminate** works in principle, so long as version space is finite.
- However, since it requires exhaustive enumeration of all hypotheses in practice it is not feasible.

The 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

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*

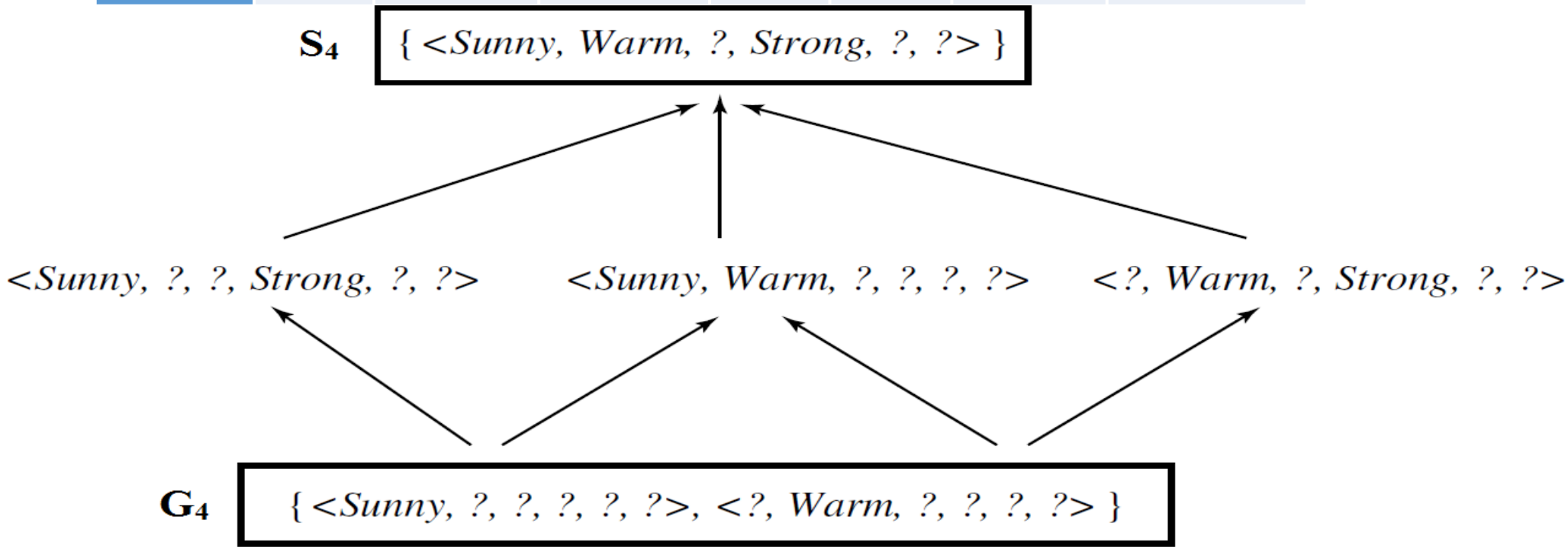
$$G \equiv \{g \in H \mid Consistent(g, D) \wedge (\neg \exists g' \in H)[(g' >_g g) \wedge Consistent(g', D)]\}$$

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*.

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

Version Space :*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



Remarks on Version Space and Candidate Elimination

1. Will the CANDIDATE-ELIMINATION Algorithm Converge to the Correct Hypothesis?

- Yes/No based on following constraints
  - there are no errors in the training examples
  - there is some hypothesis in H that correctly describes the target concept
- The target concept is exactly learned when the S and *G* boundary sets converge to a single, identical, hypothesis
- If training data contain errors
  - the algorithm is certain to remove the correct target concept from the version space
  - And possibly the S and G boundary sets eventually converge to an empty version space
- A similar symptom will appear when the training examples are correct, but the target concept cannot be described in the hypothesis representation

Example	Toothed	Legs	Species
1	True	True	Mammal (Y)
2	True	False	Reptile (N)
3	False	True	Mammal (Y)
4	False	True	Reptile (N)

S0=< Ø, Ø>
S1=<T,T>
S2=<T,T>
S3=<?,T>
S4=< >
S4 = G4 = { }
G4= ~~< T,?> <?, F>~~
G3=<?,T>
G2= ~~<F,?>~~ <?,T>
G1=<?,?>
G0=<?,?>

Example	Toothed	Legs	Species
1	True	True	Mammal
2	True	False	Reptile
3	False	True	Mammal
4	False	**False**	Reptile

S0=< Ø, Ø>
S1=<T,T>
S2=<T,T>
S3=<?,T>
S4= <?,T>

G4= <?,T>
G3= <?,T>
G2= ~~<F,?>~~ <?,T>
G1=<?,?>
G0=<?,?>

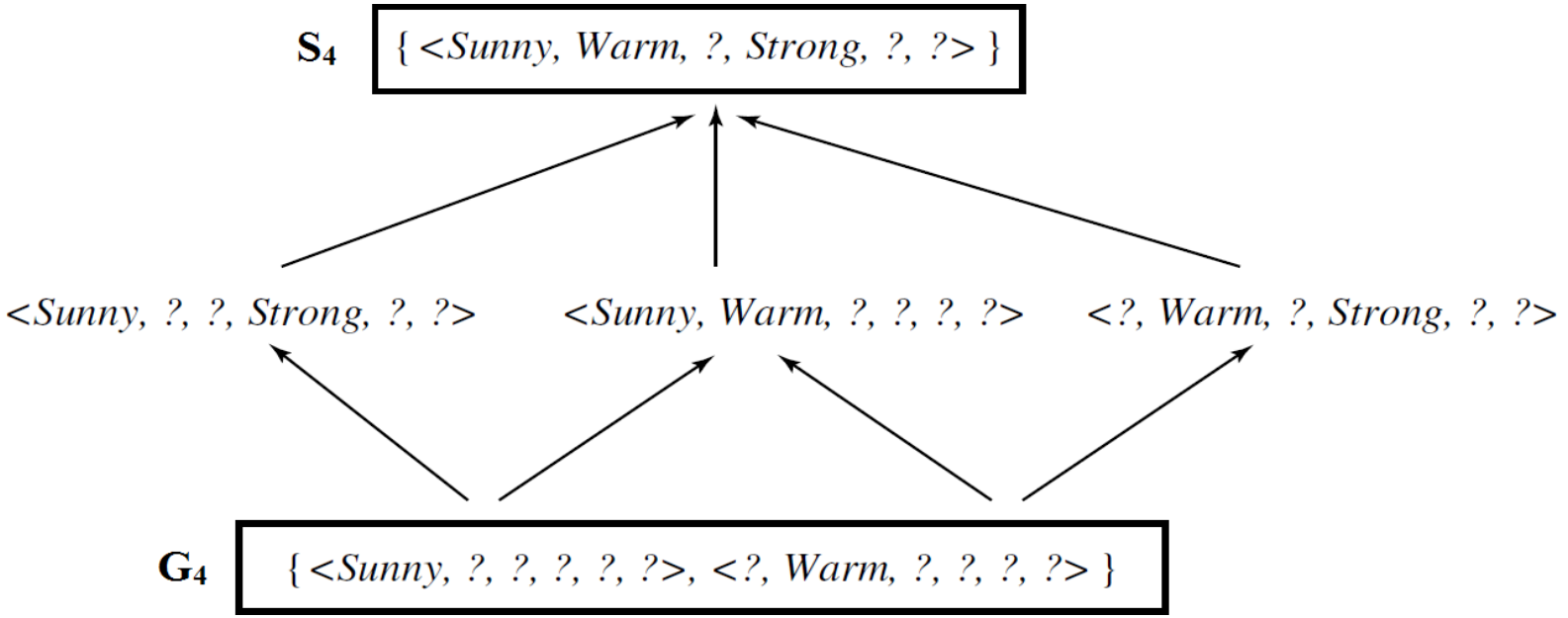
Remarks on Version Space and Candidate Elimination contd..

2. What Training Example Should the Learner Request Next?

- It is assumed that training examples are provided to the learner, 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
- The learner may conduct experiments in nature (e.g., build new bridges and allow nature to classify them as stable or unstable), or in which a teacher is available to provide the correct classification (e.g., propose a new bridge and allow the teacher to suggest whether or not it will be stable).
- The term ***query is used*** to refer to such instances constructed by the learner, which are then classified by an external oracle(teacher).
- 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.

For example

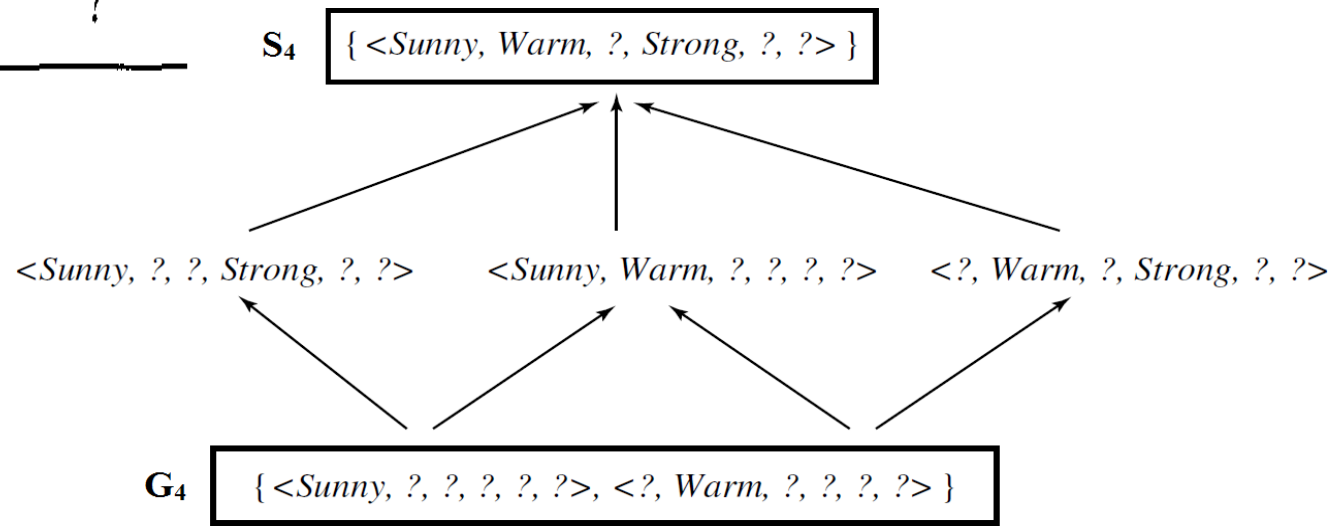
- The learner should attempt to discriminate among the alternative competing hypotheses in its current version space. Therefore, it should choose an instance that would be classified positive by some of these hypotheses, but negative by others.
- Suppose the trainer will classify the following instance as positive than only 3 out 6 hypothesis will be consistent
- ***<Sunny, Warm, Normal, Light, Warm, Same>***



Remarks on Version Space and Candidate Elimination contd..

- **How Can Partially Learned Concepts Be Used?**
  - A learner is partially learned if it contains multiple hypotheses.
  - But still it is possible to classify certain examples with the same degree of confidence as if the target concept had been uniquely identified.

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	?



Summary

- Learning involves acquiring general concepts from specific training examples.
- Concept learning can be cast as a problem of searching through a large predefined space of potential hypotheses.
- The general-to-specific partial ordering of hypotheses provides a useful structure for organizing the search through the hypothesis space.
- The FINDS algorithm utilizes this general-to-specific ordering, performing a specific-to-general search through the hypothesis space to find the most specific hypothesis consistent with the training examples
- The CANDIDATE-ELIMINATION algorithm utilizes this general-to-specific ordering to compute the version space (the set of all hypotheses consistent with the training data) by incrementally computing the sets of maximally specific (S) and maximally general (G) hypotheses.
- The target concept is exactly learned when the S and G boundary sets converge to a single, identical, hypothesis. Otherwise, the S and G sets delimit the entire set of hypotheses consistent with the data, they provide the learner with a description of its uncertainty regarding the exact identity of the target concept.
- Version spaces-based CANDIDATE-ELIMINATION algorithm is not robust to noisy data or to situations in which the unknown target concept is not expressible in the provided hypothesis space.