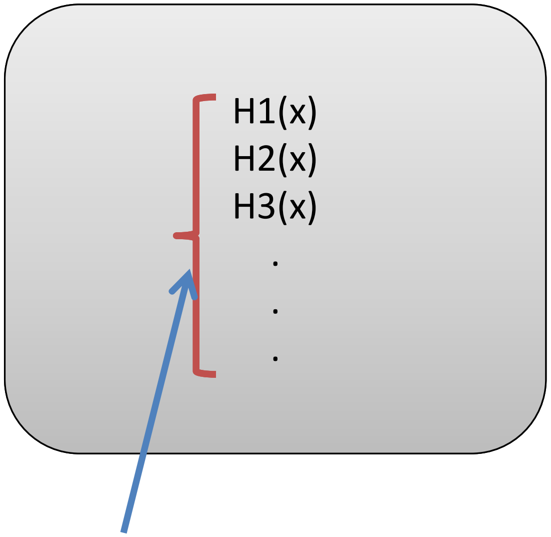
What is concept learning?

“Problem of searching through a predefined space of potential hypotheses for the hypothesis that best fits the training examples”



Space of all possible hypotheses

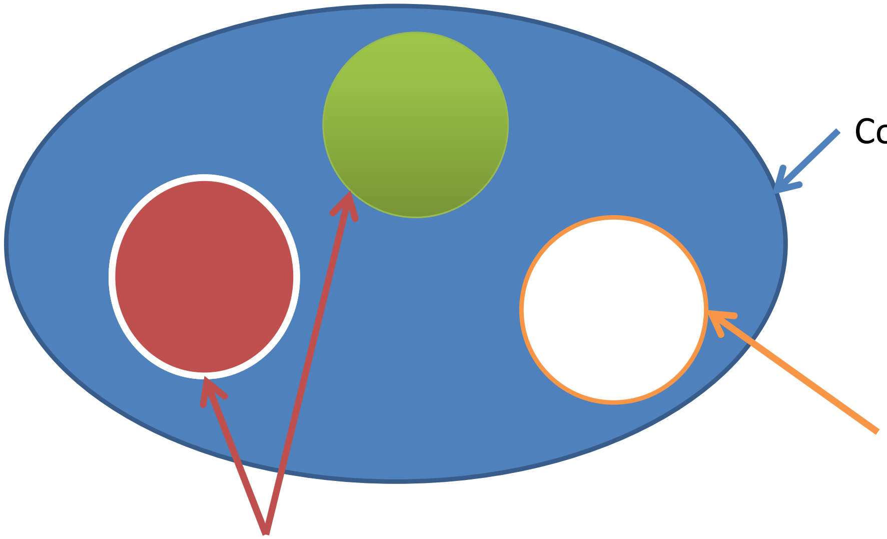
Concept Learning determines a hypothesis that best fits the training examples, by searching space of potential hypotheses

Introduction

- Assume a given domain, e.g. objects, animals, etc.
- A concept can be seen as a subset of the domain, e.g. birds⊆animals
- Task: acquire intentional concept description from training examples
- Generally we can’t look at all objects in the domain

What is concept

The set of features that differentiate one object from another, can be called a **concept**.



Concept Space

Target concept
to be learned

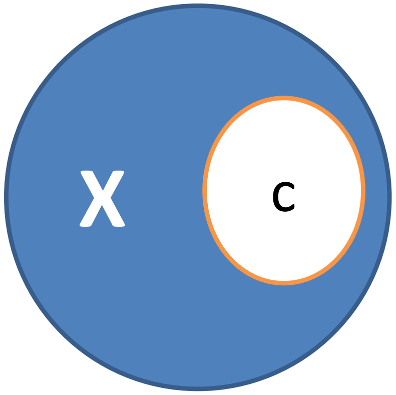
Other concepts

Boolean valued function is able to identity target concept over concept space

Target Concept:

The set of items/objects over which the concept is defined is called the **set of instances** and denoted by X.

The concept or function to be learned is called the **target concept** and denoted by c.



It can be seen as a **boolean valued function defined over X** and can be represented as:

c: X -> {0, 1}

The goal of concept learning is to find a hypothesis h which can identify all the objects in X so that:

h(x) = c(x) for all x in X

There are three necessary things for an algorithm which supports concept learning:
1. Training data (Past experiences to train our models)
2. Target Concept (Hypothesis to identify data objects)
3. Actual data objects (For testing the models)

The inductive learning is based on **formulating a generalized concept after observing a number of instances of examples of the concept**.

we can make our machine to learn from past data and make them intelligent to identify whether an object falls into a specific category of our interest or not.

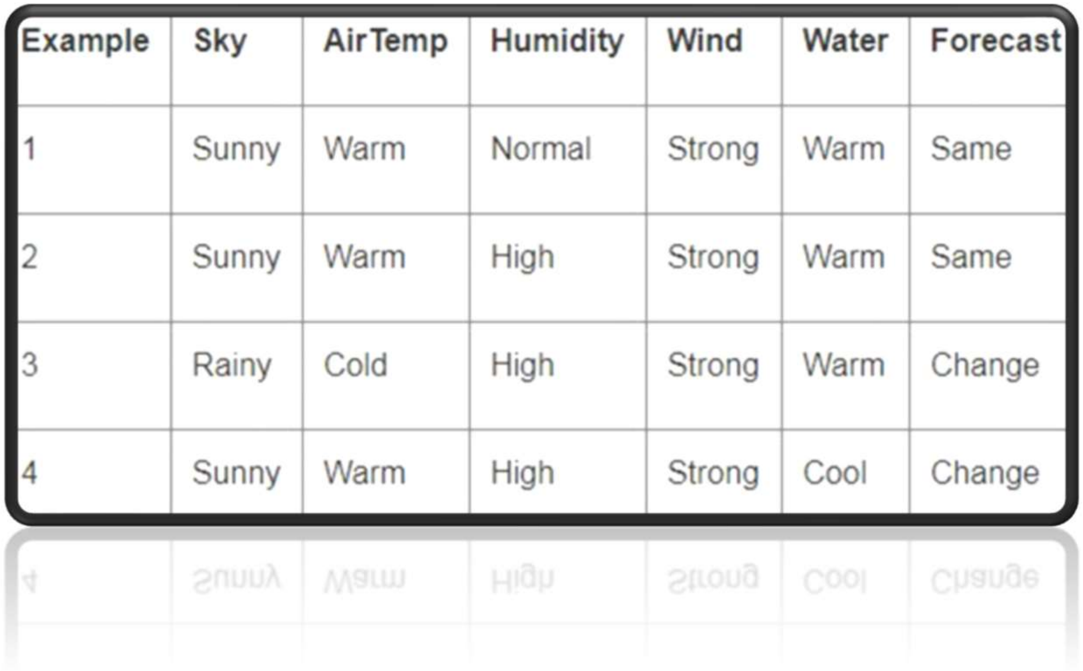
Machines can also learn from the concepts to identify whether an object belongs to a specific category or not by processing past/training data to find a hypothesis that best fits the training examples.

Assume the following:

Some attributes/features of the day can be:

Sky, Air Temperature, Humidity, Wind, Water, Forecast

X = set of instances



Example	Sky	AirTemp	Humidity	Wind	Water	Forecast
1	Sunny	Warm	Normal	Strong	Warm	Same
2	Sunny	Warm	High	Strong	Warm	Same
3	Rainy	Cold	High	Strong	Warm	Change
4	Sunny	Warm	High	Strong	Cool	Change

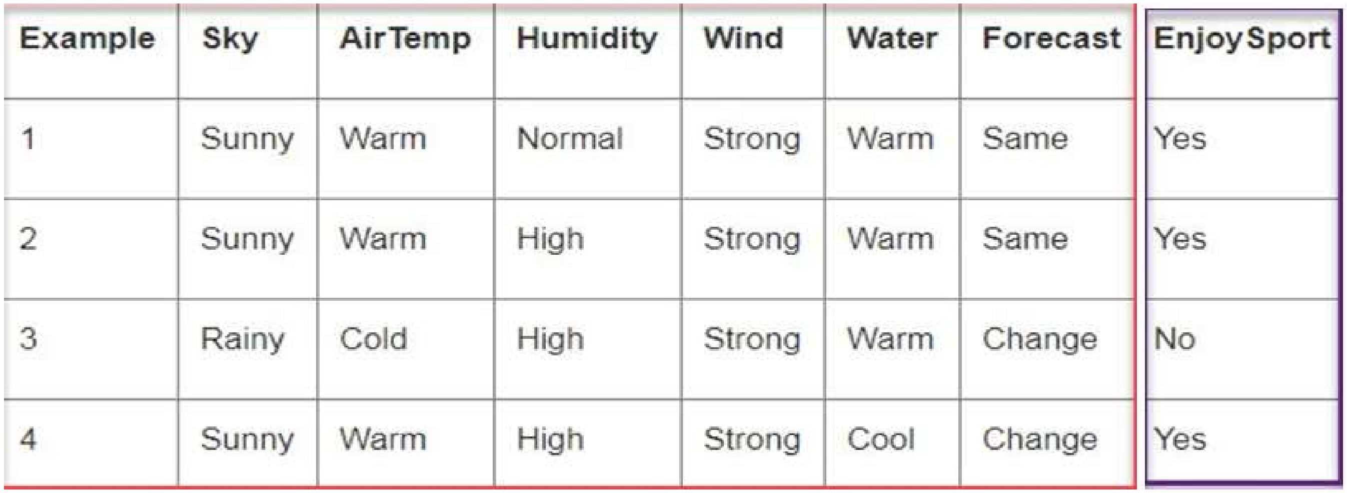
Many concepts can defined over the X.
For example, the concepts can be
- Days on which my friend Rama enjoys his favorite water sport
- Days on which my friend Rama will not go outside of his house.
- etc

Target concept —

-the concept or function to be learned
-denoted by c
-a **boolean valued function defined over X**
-represented as c: X → {0, 1}.

e.g., c = Days on which Rama will enjoy sports

To indicate whether Rama enjoys sports on that day, one more attribute EnjoySport is included in dataset as shown below



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

X, set of instances

Our target concept is EnjoySport.
It is defined as EnjoySport : X -> {0,1}

Our goal is to predict EnjoySport for given arbitrary day with new sample values for attributes of day, based on previous learning(from training examples)

H denotes set of all possible hypotheses that computer can consider regarding the identify of target concept.

Our goal is to determine hypothesis from H, to identify target concept, such that $h(x)=c(x)=1$

Learning can be viewed as a task of searching this space

Different learning algorithms search this space in different ways

Representation of Hypothesis:

Each hypothesis is represented by vector of constraints as follows:
<attribute-1, attribute-2, …, attribute-n>

e.g., For task EnjoySports, hypothesis is a vector of six constraints as follows
<Sky, AirTemp, Humidity, Wind, Water, Forecast>

Each attribute has one of these three posibilities:

-a specfic value (e.g., W ater = W arm)
– don’t care (e.g., “W ater =?”)
– no value allowed (e.g.,“Water= Ø”)

Examples of hypotheses:

Hypothesis	Target Concept
<?, Cold, High, ?, ?, ?>	Rama will enjoy sports on cold days with high humidity
<?, ?, ?, ?, ?, ?>	Every day is good day for enjoying sports
<0, 0, 0, 0, 0, 0>	No day is good day for enjoying sports

Most General/Most Specific Hypothesis

- Most general hypothesis: (?, ?, ?, ?, ?)

- Most specific hypothesis: (Ø, Ø, Ø, Ø, Ø)

EnjoySport Concept Learning Task

X — 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.

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

(x, c(x)) — When learning the target concept, the learner is presented by a set of training examples, each consisting of an instance x from X, along with its target concept value c(x). Instances for which c(x) = 1 are called positive examples and instances for which c(x) = 0 are called negative examples. 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).

D — We use the symbol D to denote the set of available training examples.

H — 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.

h(x) — In general, each hypothesis h in H represents a Boolean-valued function defined over X; that is, $h : X \rightarrow \{0, 1\}$. 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*** : set of all possible days, each described by the attributes
  - Sky – (values: Sunny, Cloudy, Rainy)
  - AirTemp – (values: Warm, Cold)
  - Humidity – (values: Normal, High)
  - Wind – (values: Strong, Weak)
  - Water – (values: Warm, Cold)
  - Forecast – (values: Same, Change)
- ***Target Concept (Function) c*** : EnjoySport : X → {0,1}
- ***Hypotheses H*** : Each hypothesis is described by a conjunction of constraints on the attributes.
- ***Training Examples D*** : positive and negative examples of the target function

Determine

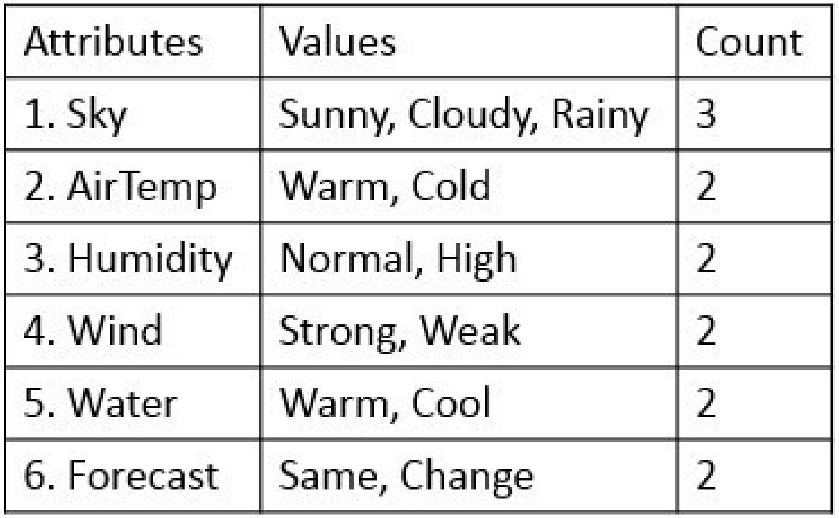
- A hypothesis h in H such that h(x) = c(x) for all x in D.

The Inductive Learning Hypothesis

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

Hypothesis Space

Sky has 3 possible values, and other 5 attributes have 2 possible values, as follows



Attributes	Values	Count
1. Sky	Sunny, Cloudy, Rainy	3
2. AirTemp	Warm, Cold	2
3. Humidity	Normal, High	2
4. Wind	Strong, Weak	2
5. Water	Warm, Cool	2
6. Forecast	Same, Change	2

$$Distinct\ Observations\ in\ X = 3.2.2.2.2.2 = 96$$

$$Distinct\ hypothesis\ in\ H = 5.4.4.4.4.4 = 5120$$

In the hypothesis representation of EnjoySports, value of each attribute could be either "?" or "0" other than defined values. So the hypothesis space H has 5120 distinct hypothesis.

The number of combinations: 5×4×4×4×4×4 = 5120 syntactically distinct hypotheses. They are syntactically distinct but not semantically.

For example, the below 2 hypothesis says the same but they look different.

h1 = <Sky=0, Temp=warm, Humidity=?, Wind=strong, Water=warm, Forecast=same >

h2 = <Sky=sunny, Temp=warm, Humidity=?, Wind=strong, Water=0, Forecast=same>

Neither of these hypotheses accept any “day”, so semantically the same.

All such hypothesis having same semantic is counted as 1. So we can have total number of combinations as below.

1 (hypothesis with one or more 0)
+
4×3×3×3×3×3 (add ? to each attribute)
=
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 general-to-specific ordering of hypotheses.

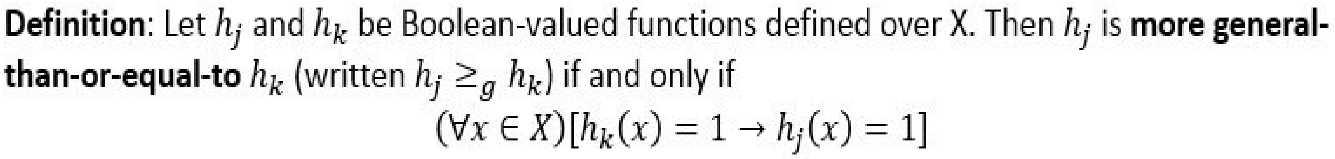
Consider two hypotheses
h1 = (Sunny, ?, ?, Strong, ?, ?)
h2 = (Sunny, ?, ?, ?, ?, ?)

Now consider the sets of instances that are classified positive by hl and by h2.
– Because h2 imposes fewer constraints on the instance, it classifies more instances as positive.
– In fact, any instance classified positive by hl will also be classified positive by h2.
– Therefore, we say that h2 is more general than hl.
-**In reverse, we can also say that, h1 is more specific than h2.**

More-General-Than Relation

For any instance x in X and hypothesis h in H, we say that x satisfies h if and only if h(x) = 1.

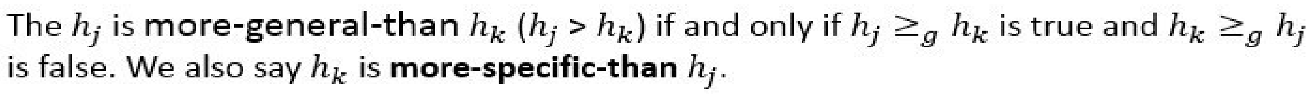
More-General-Than-Or-Equal Relation:



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

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

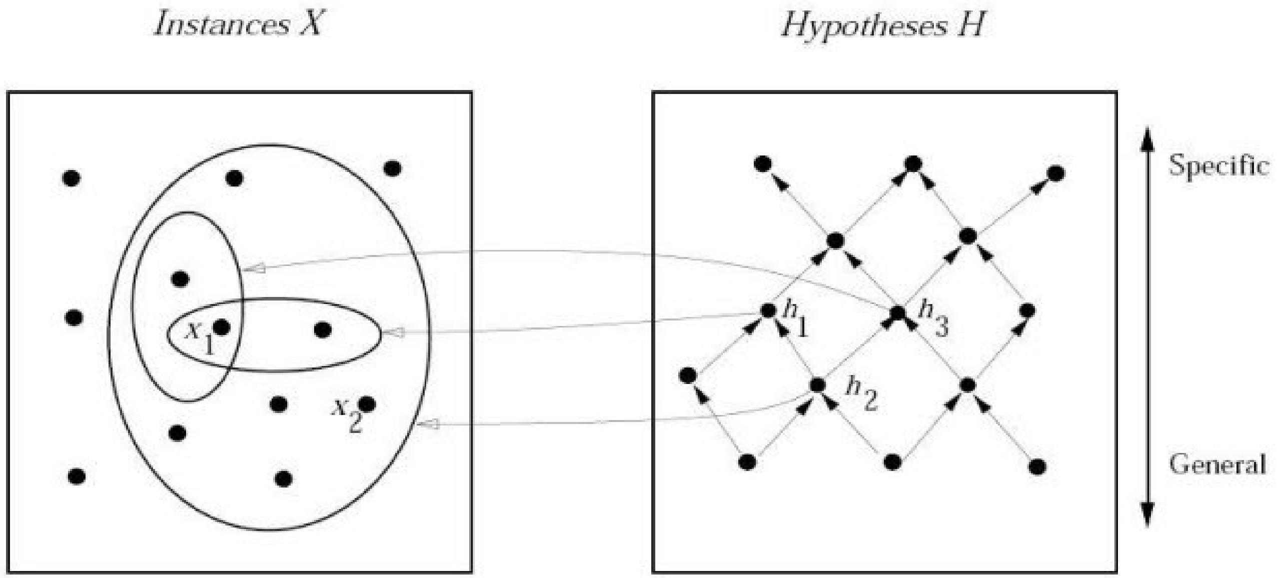
more-specific-than:



The h_j is **more-general-than** h_k ($h_j > h_k$) if and only if $h_j \geq_g h_k$ is true and $h_k \geq_g h_j$ is false. We also say h_k is **more-specific-than** h_j.

≥ does not depend on the concept to be learned
- It defines a partial order over the set of hypotheses
- strictly-more-general than: >
- more-specific-than ≤

more-general-than-equal-to or more-general-than



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, ?>

In the above example,
there are 2 instances- x1 and x2, and 3 hypothesis - h1, h2 and h3.
h1 classifies — x1, h2 classifies — x1 and x2, and h3 classifies — x1.
This indicates, h2 is more-general-than h1 and h3.
h2 > h1 and h2 > h3
But there is no more-general relation between h1 and h3

Two most popular approaches to find a suitable hypothesis, they are:

1.Find-S Algorithm

2.List-Then-Eliminate Algorithm

3.Candidate-Elimination Algorithm

Find-s:
finding a maximally specific hypothesis

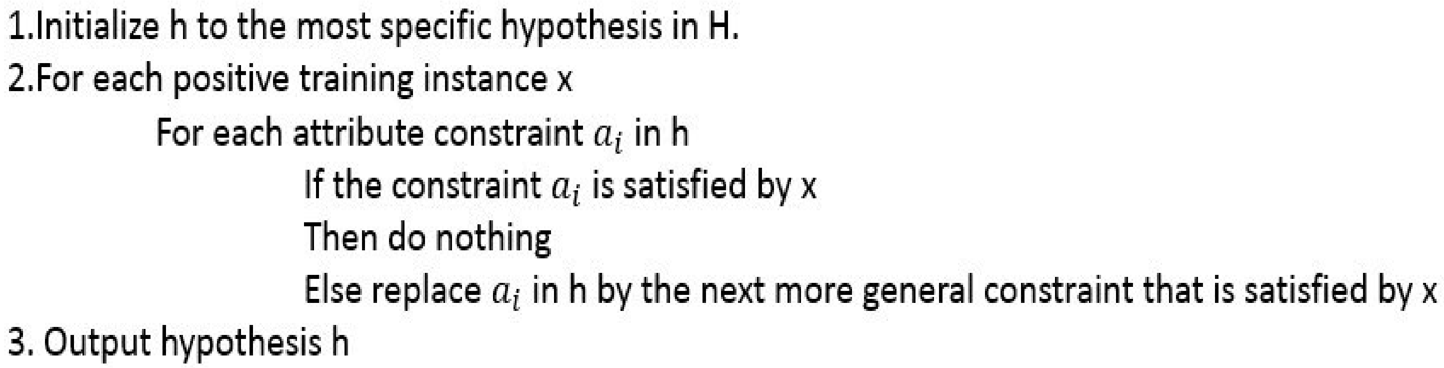
FIND-S Algorithm starts from the most specific hypothesis and generalize it by considering only positive examples.

- FIND-S algorithm ignores negative examples.

FIND-S algorithm finds the most specific hypothesis within H that is consistent with the positive training examples.

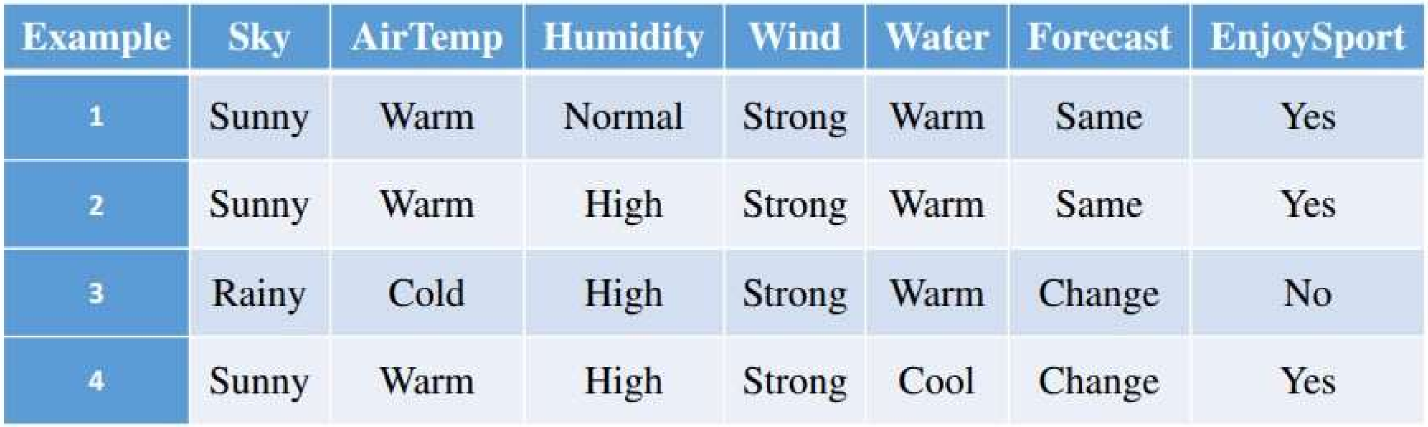
The final hypothesis will also be consistent with negative examples if the correct target concept is in H, and the training examples are correct.

FIND-S Algorithm



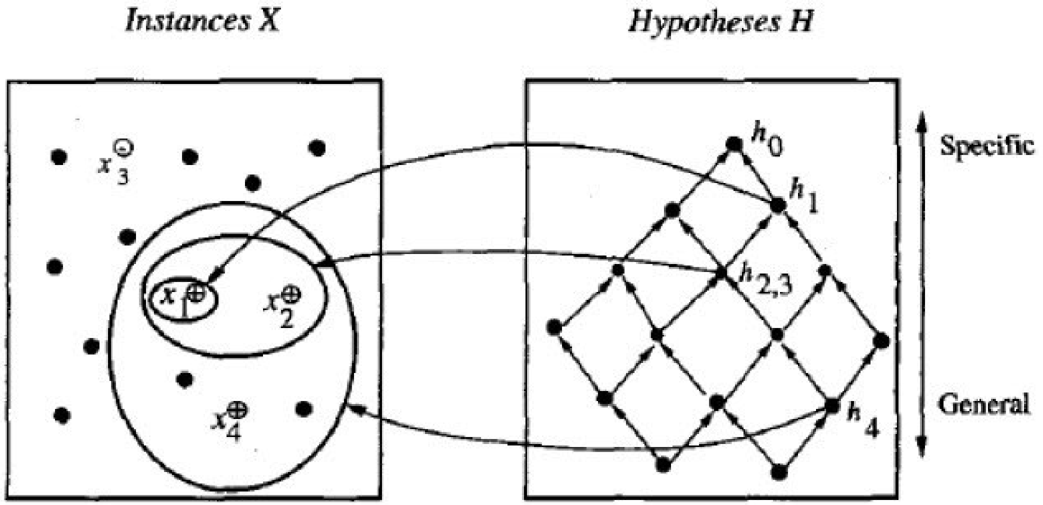
```
1.Initialize h to the most specific hypothesis in H.
2.For each positive training instance x
          For each attribute constraint $a_i$ in h
                    If the constraint $a_i$ is satisfied by x
                    Then do nothing
                    Else replace $a_i$ in h by the next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

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 Find-S algorithm for the above example, can be illustrated with the below figure.



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

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

x_3 = *<Rainy Cold High Strong Warm Change>*, -

x_4 = *<Sunny Warm High Strong Cool Change>*, +

h_0 = <∅, ∅, ∅, ∅, ∅, ∅>

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

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

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

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

The key property of the FIND-S algorithm:

•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, and provided the training examples are correct.

Unanswered questions by FIND-S

There are several questions still left unanswered, such as:

1.Has FIND-S 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 (i.e., the correct target concept), or whether there are many other consistent hypotheses as well.
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 ?. 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.
4.What if there are several maximally specific consistent hypotheses?. There can be several maximally specific hypotheses consistent with the data. Find S finds only one.

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.

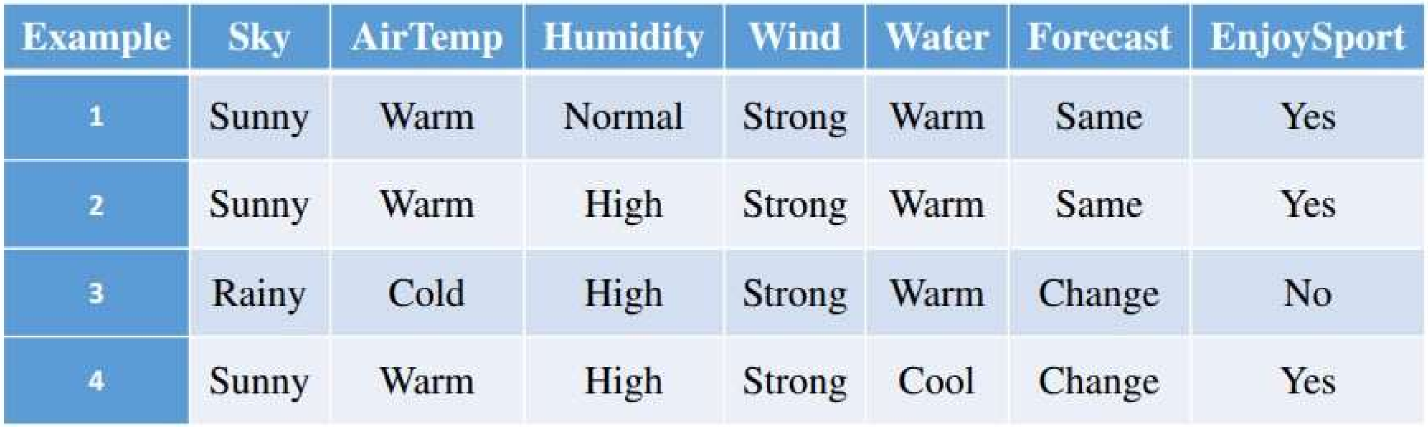
$$Consistent\ (h, D) \equiv (\forall \langle x, c(x)\rangle \in D)\ h(x) = c(x))$$

difference between definitions of consistent and satisfies

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.

An example x is said to consistent with hypothesis h iff h(x) = c(x)

The set of training examples D are below.



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

Assume
hypothesis h = **< Sunny, Warm, ?, Strong, ?, ?>**

Now for each example (x, c(x)) in D, we will evaluate h(x)

1.(<Sunny, Warm, Normal, Strong, Warm, Same>,<yes>) → h(x)=c(x)
2.(<Sunny, Warm, High, Strong, Warm, Same>,<yes>) → h(x)=c(x)
3.(<Rainy, Cold, High, Strong, Warm, Change>,<No>) → h(x)=c(x)
4.(<Sunny, Warm, High, Strong, Cool, Change>,<yes>) → h(x)=c(x)

Hence, hypothesis h is consistent with a set of training examples D

Lets say, we have a hypothesis h2 = < ?, Warm, ?, Strong, ?, ?>,
is this hypothesis consistent with set of training example D ?

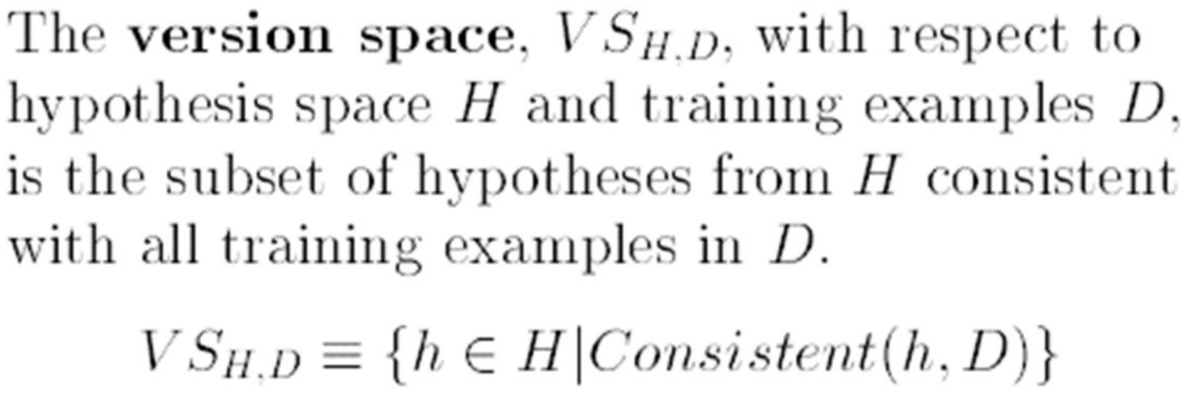
All the training examples hold h(x) = c(x). So hypothesis h2 is consistent with D.

Assume a hypothesis h1 = < ?, ?, ?, Strong, ?, ?>,
is this hypothesis consistent with set of training example D ?

In case of training example (3), h(x) != c(x). So hypothesis h1 is not consistent with D.

Version Spaces

The Candidate-Elimination algorithm represents the set of all hypotheses consistent with the observed training examples.



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.

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

In above example, we have two hypothesis from H and they are consistent with D.

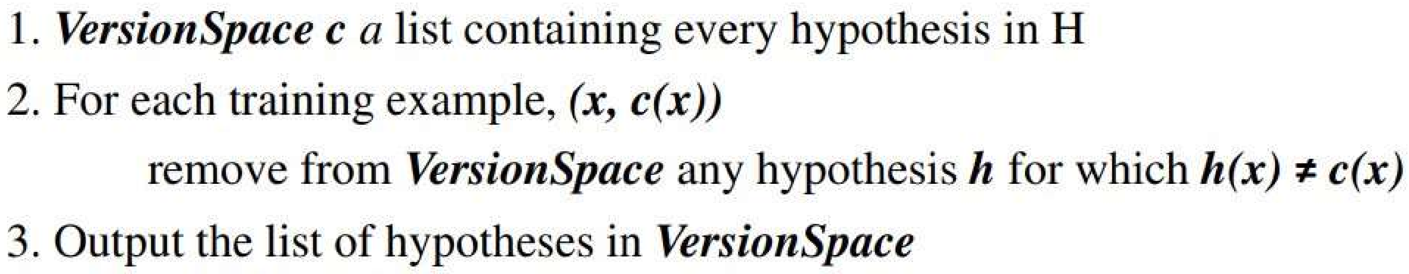
h1=< Sunny, Warm, ?, Strong, ?, ?> and h2=< ?, Warm, ?, Strong, ?, ?>

So this set of hypothesis { h1, h2} is called a Version Space.

List-Then-Eliminate Algorithm

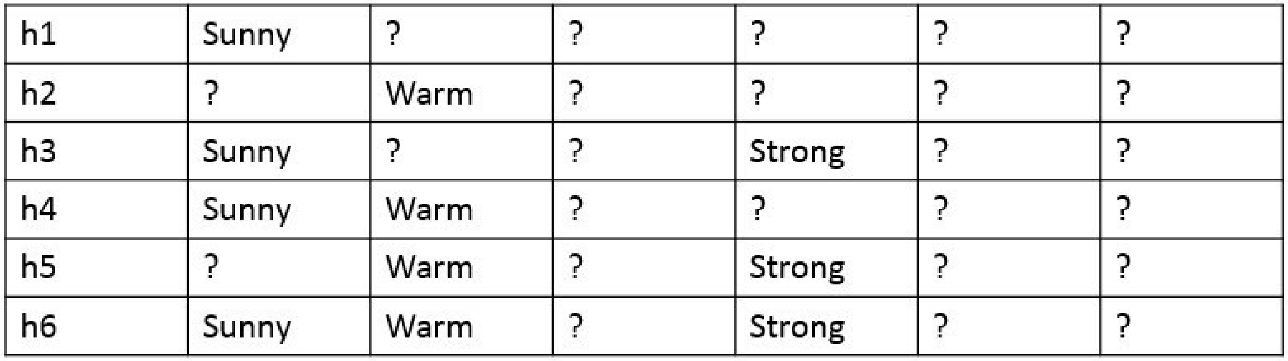
List-Then-Eliminate algorithm initializes the version space to contain all hypotheses in H, then eliminates any hypothesis found inconsistent with any training example.

The algorithm is as follows :



```
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 $h(x) \neq c(x)$
3. Output the list of hypotheses in VersionSpace
```

For the above EnjoySport training examples D, we can output the below list of hypothesis which are consistent with D. In other words, the below list of hypothesis is a version space.



h1	Sunny	?	?	?	?	?
h2	?	Warm	?	?	?	?
h3	Sunny	?	?	Strong	?	?
h4	Sunny	Warm	?	?	?	?
h5	?	Warm	?	Strong	?	?
h6	Sunny	Warm	?	Strong	?	?

In the list of hypothesis, there are two extremes representing general (h1 and h2) and specific (h6) hypothesis.

Lets define these 2 extremes as general boundary G and specific boundary S.

Compact Representation of Version Spaces

A version space can be represented with its general and specific boundary sets.

Definition — G
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.

Definition — S
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.

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.

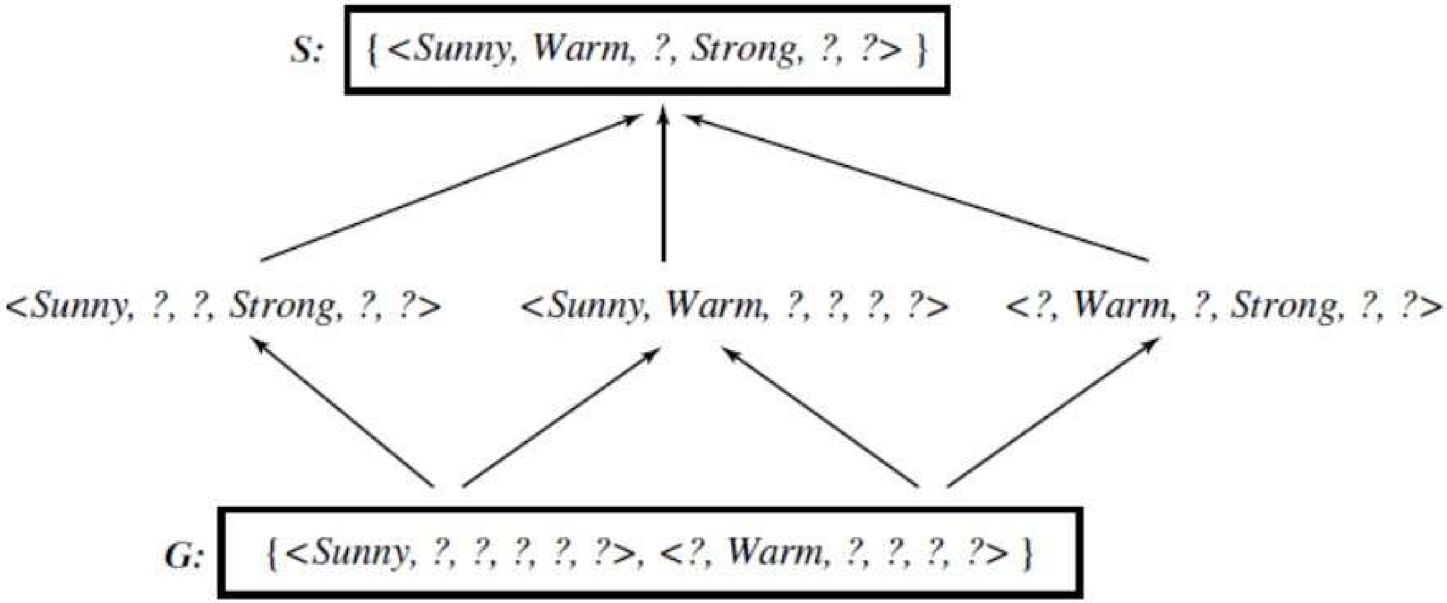
Every member of the version space lies between these boundaries

$$VS_{H,D}=\{h \in H \,|(\exists s \in S)\ (\exists g \in G)\ (g \geq_g h \geq_g s)\}$$

where x ≥y means x is more general or equal to y

Example Version Space

The below figure shows the version space for EnjoySport concept learning including both general and specific boundary sets.



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 as

G0 🡐 { <?, ?, ?, ?, ?, ?> }

and initializing the S boundary set to contain the most specific hypothesis as

S0 🡐 { <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.

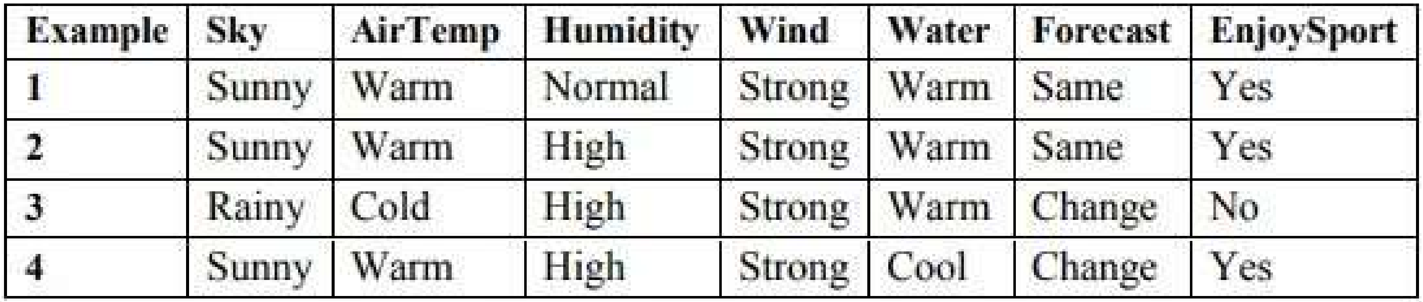
•After all examples have been processed, the computed version space contains all the hypotheses consistent with these examples and only these hypotheses.

Candidate-Elimination Algorithm

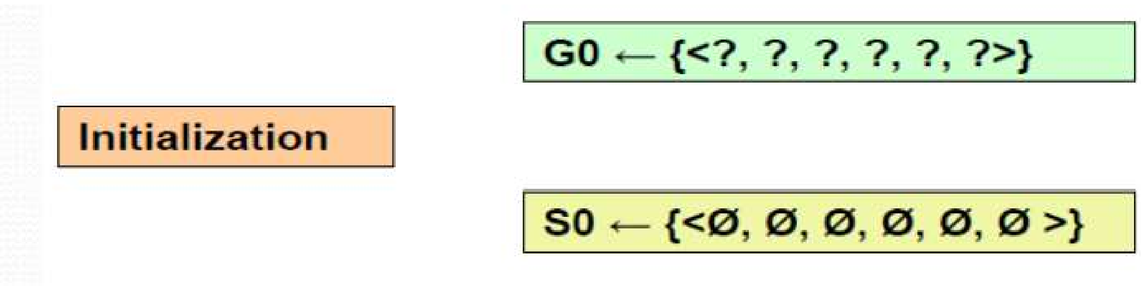
```
Initialize G to the set of maximally general hypotheses in H
Initialize S to the set of maximally specific hypotheses in H
For each training example d, do
• If d is a positive example
    • Remove from G any hypothesis inconsistent with d
    • For each hypothesis s in S that is not consistent with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

- Here are the training examples D



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



Initialization

G0 ← {<?, ?, ?, ?, ?, ?>}

S0 ← {<Ø, Ø, Ø, Ø, Ø, Ø >}

G0 ← {<?, ?, ?, ?, ?, ?>}

S0 ← {<Ø, Ø, Ø, Ø, Ø, Ø >}

Iteration 1

x1 = <Sunny, Warm, Normal, Strong, Warm, Same>

G1 ← {<?, ?, ?, ?, ?, ?>}

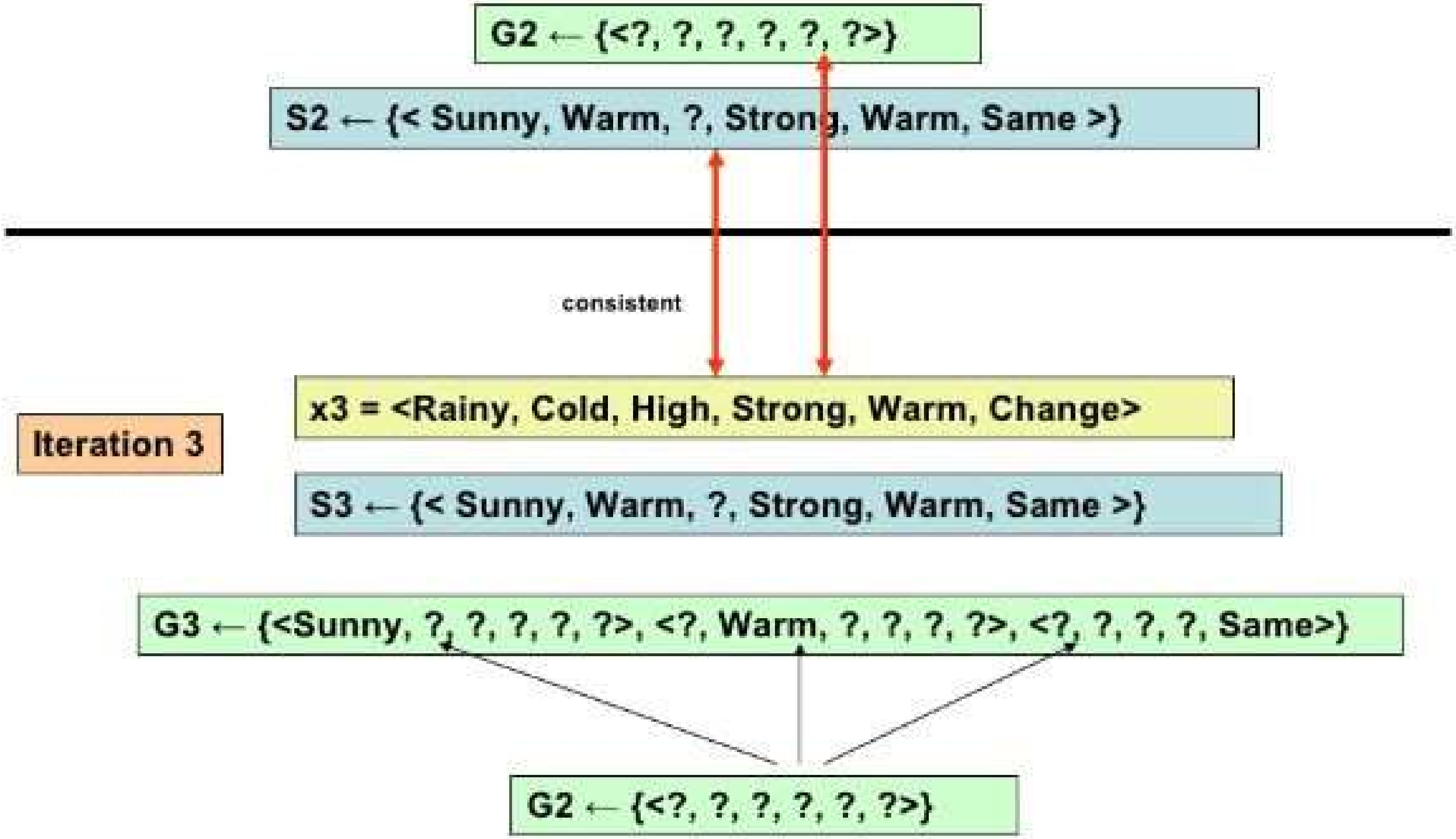
S1 ← {< Sunny, Warm, Normal, Strong, Warm, Same >}

Iteration 2

x2 = <Sunny, Warm, High, Strong, Warm, Same>

G2 ← {<?, ?, ?, ?, ?, ?>}

S2 ← {< Sunny, Warm, ?, Strong, Warm, Same >}



G2 ← {<?, ?, ?, ?, ?, ?>}

S2 ← {< Sunny, Warm, ?, Strong, Warm, Same >}

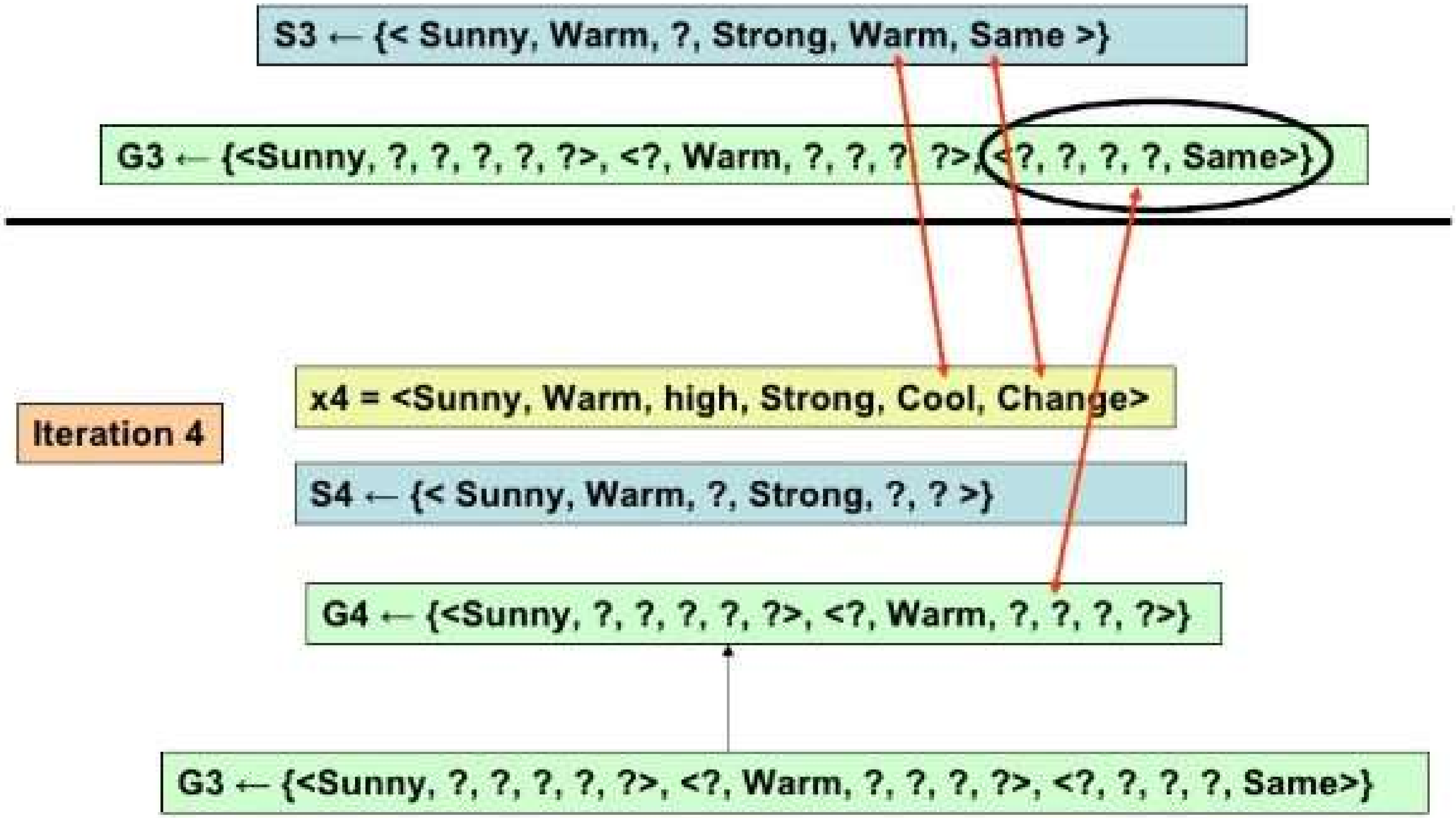
Iteration 3

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

S3 ← {< Sunny, Warm, ?, Strong, Warm, Same >}

G3 ← {<Sunny, ?, ?, ?, ?, ?>, <?, Warm, ?, ?, ?, ?>, <?, ?, ?, ?, Same>}

G2 ← {<?, ?, ?, ?, ?, ?>}



S3 ← {< Sunny, Warm, ?, Strong, Warm, Same >}

G3 ← {<Sunny, ?, ?, ?, ?, ?>, <?, Warm, ?, ?, ?, ?>, <?, ?, ?, ?, Same>}

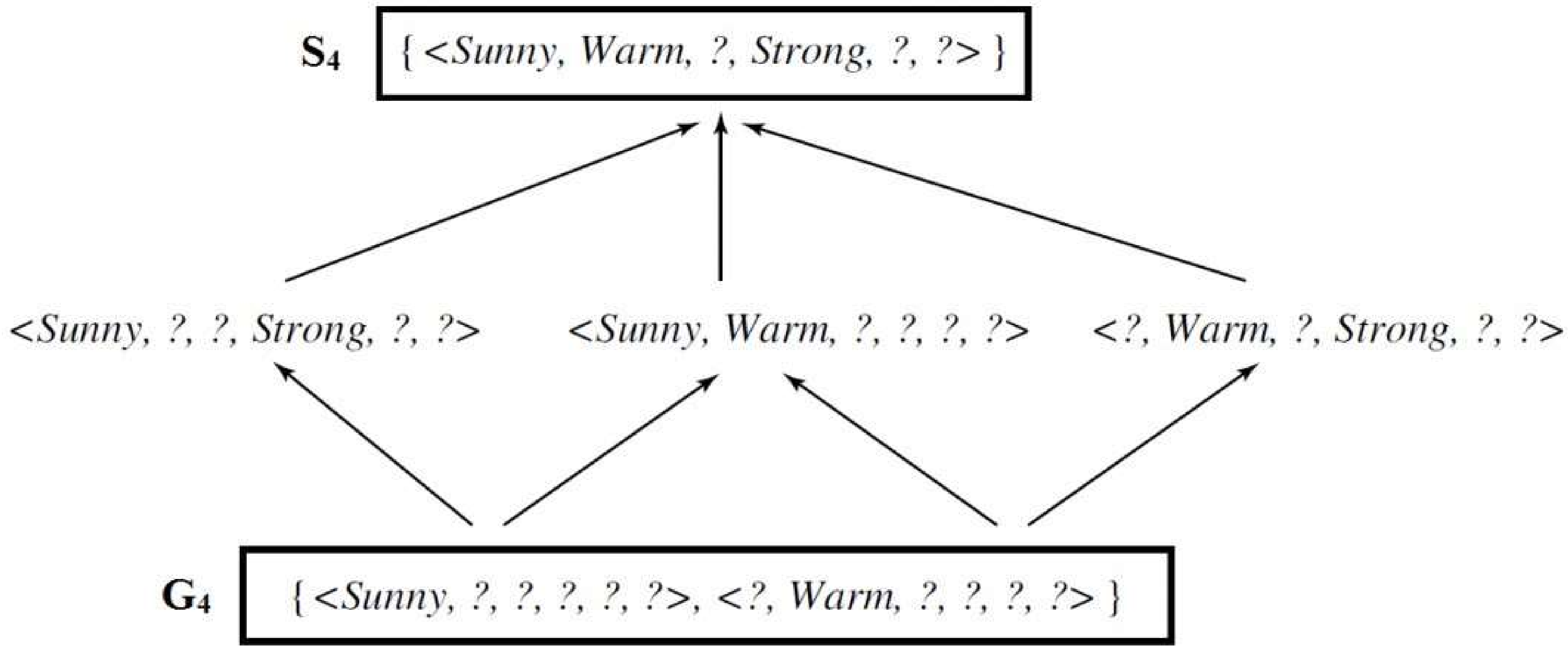
Iteration 4

x4 = <Sunny, Warm, high, Strong, Cool, Change>

S4 ← {< Sunny, Warm, ?, Strong, ?, ? >}

G4 ← {<Sunny, ?, ?, ?, ?, ?>, <?, Warm, ?, ?, ?, ?>}

G3 ← {<Sunny, ?, ?, ?, ?, ?>, <?, Warm, ?, ?, ?, ?>, <?, ?, ?, ?, Same>}


S4
{ <Sunny, Warm, ?, Strong, ?, ?> }
<Sunny, ?, ?, Strong, ?, ?>
<Sunny, Warm, ?, ?, ?, ?>
<?, Warm, ?, Strong, ?, ?>
G4
{ <Sunny, ?, ?, ?, ?, ?>, <?, Warm, ?, ?, ?, ?> }

Remarks on Version Space and Candidate elimination algorithm

Will the Candidate elimination algorithm Converge to the Correct Hypothesis ?

The version space learned by the Candidate elimination algorithm will converge toward the hypothesis that correctly describes the target concept, provided
(1) there are no errors in the training examples, and
(2) there is some hypothesis in H that correctly describes the target concept.

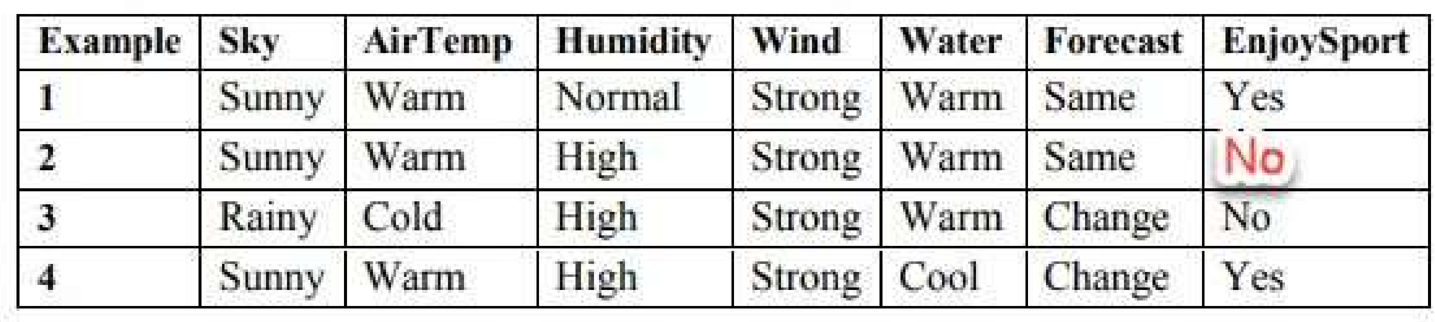
What will happen if the training data contains errors ?.

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.

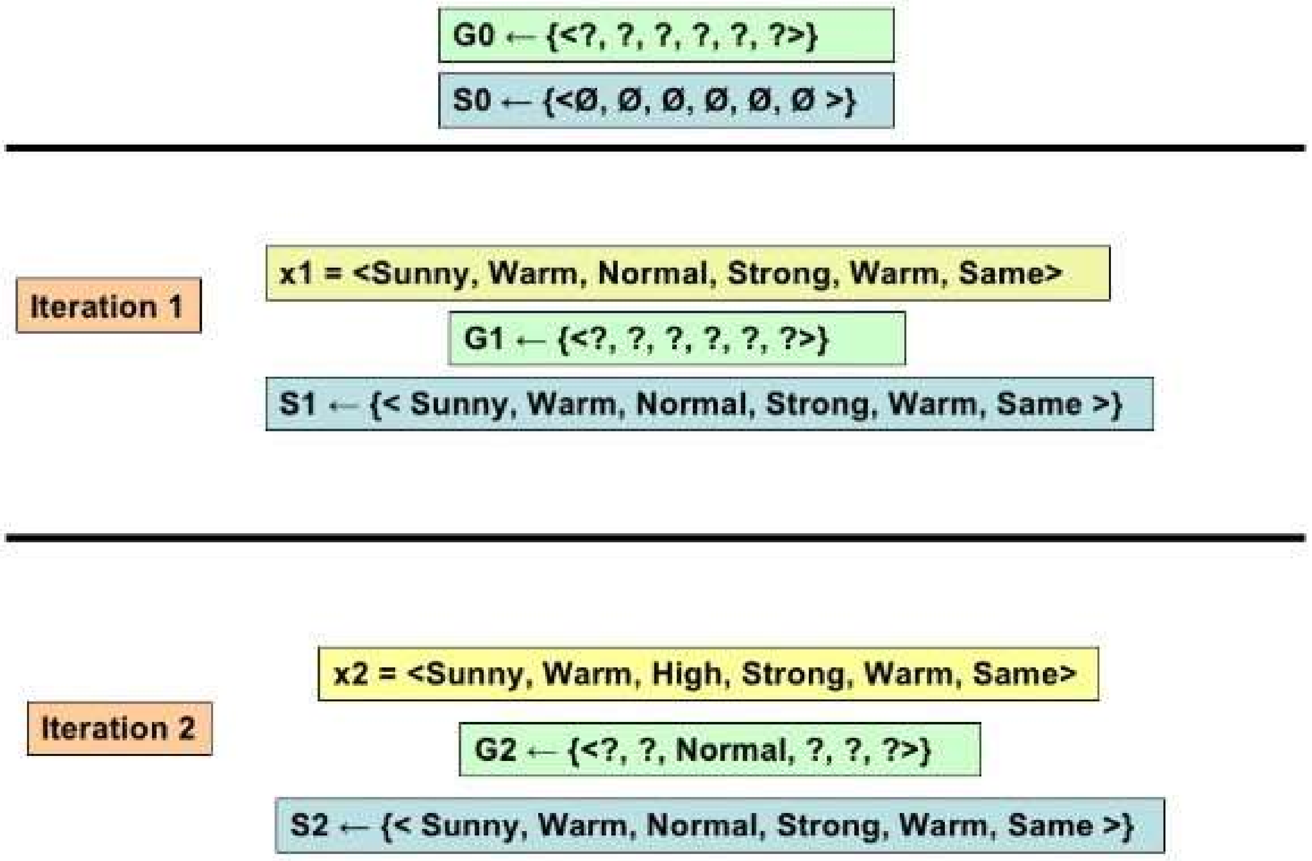
What will happen if the training data contains errors ?.

Suppose, for example, that the second training example above is incorrectly presented as a negative example instead of a positive example.



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

Lets run the candidate elimination algorithm on this data and see the result.



G0 ← {<?, ?, ?, ?, ?, ?>}

S0 ← {<Ø, Ø, Ø, Ø, Ø, Ø >}

Iteration 1

x1 = <Sunny, Warm, Normal, Strong, Warm, Same>

G1 ← {<?, ?, ?, ?, ?, ?>}

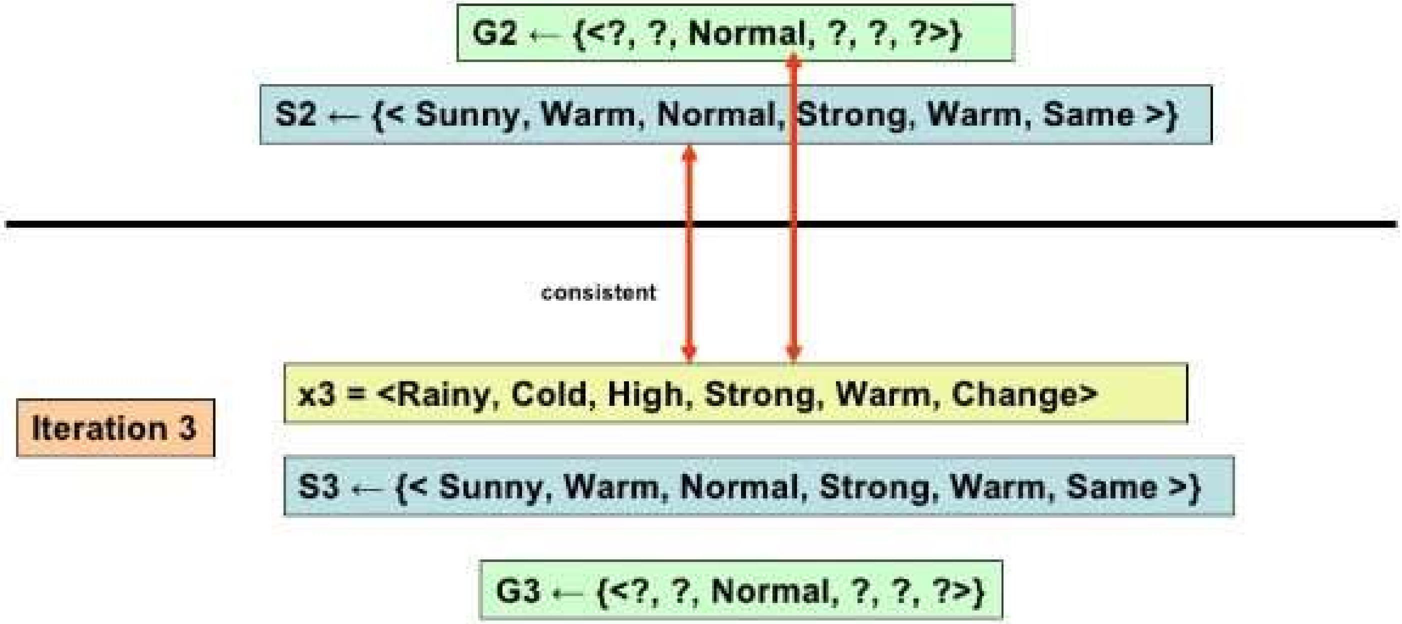
S1 ← {< Sunny, Warm, Normal, Strong, Warm, Same >}

Iteration 2

x2 = <Sunny, Warm, High, Strong, Warm, Same>

G2 ← {<?, ?, Normal, ?, ?, ?>}

S2 ← {< Sunny, Warm, Normal, Strong, Warm, Same >}



G2 ← {<?, ?, Normal, ?, ?, ?>}

S2 ← {< Sunny, Warm, Normal, Strong, Warm, Same >}

consistent

Iteration 3

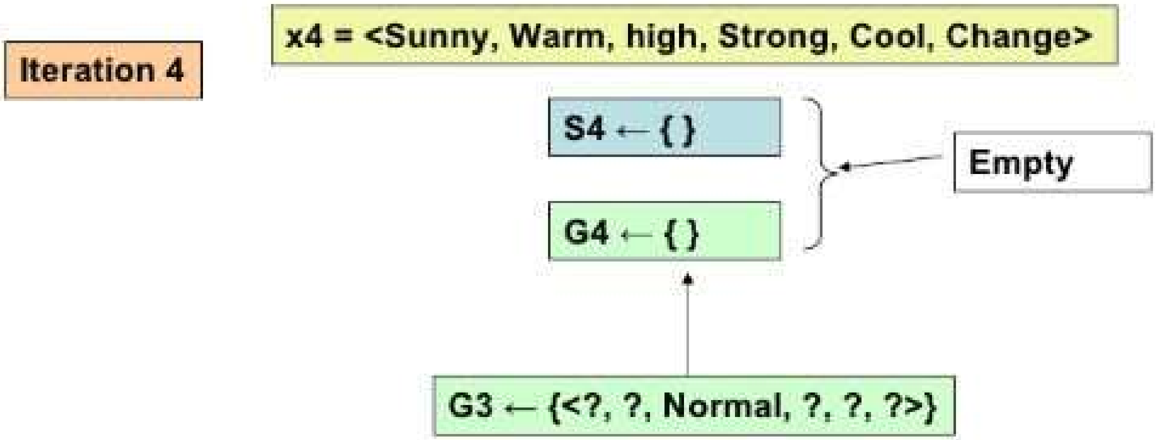
x3 = <Rainy, Cold, High, Strong, Warm, Change>

S3 ← {< Sunny, Warm, Normal, Strong, Warm, Same >}

G3 ← {<?, ?, Normal, ?, ?, ?>}

S3 ← {< Sunny, Warm, Normal, Strong, Warm, Same >}

G3 ← {<?, ?, Normal, ?, ?, ?>}



Iteration 4

x4 = <Sunny, Warm, high, Strong, Cool, Change>

S4 ← { }

G4 ← { }

Empty

G3 ← {<?, ?, Normal, ?, ?, ?>}

After processing all the training examples, the algorithm removes the correct target concept from the version space.

Find-S	Candidate Elimination
Find a hypothesis consistent with training data.	Find a compact representation of all hypothesis consistent with training data
Only consider the most specific one.	Consider all possible consistent hypothesis.
Ignore No instances.	Consider both Yes and No instances