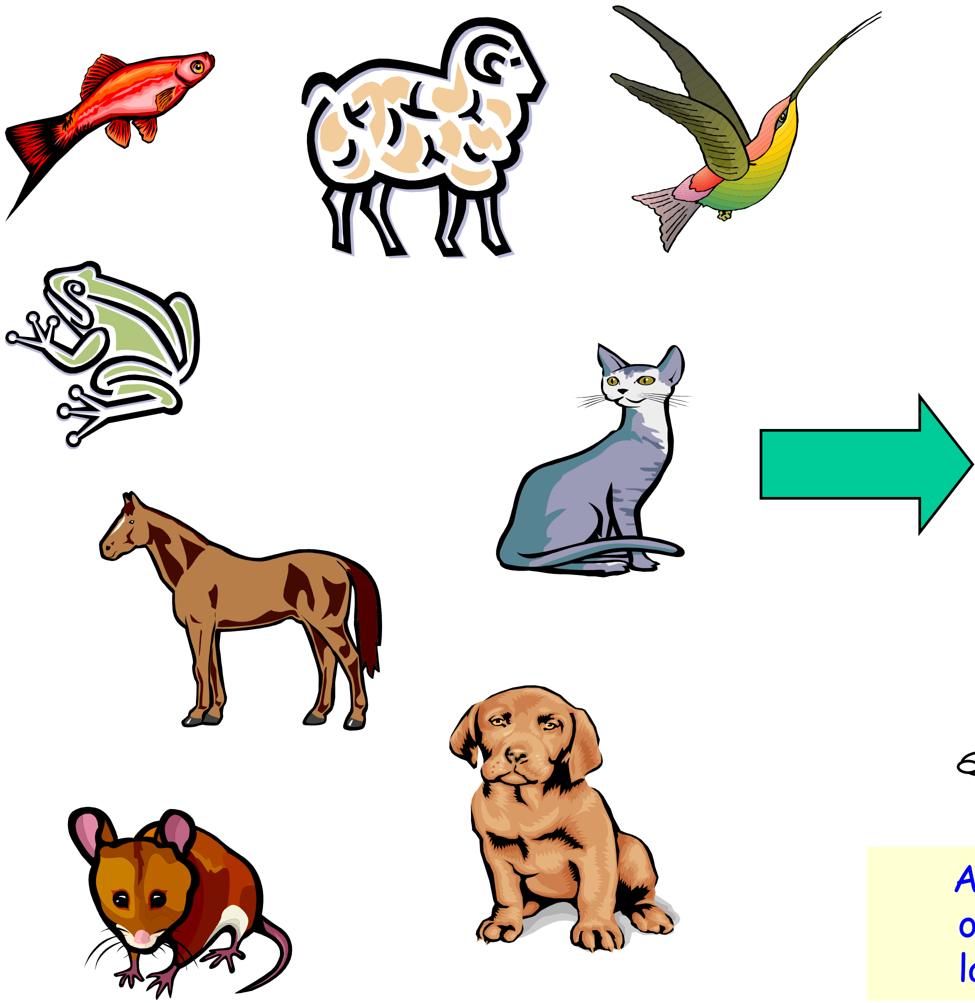
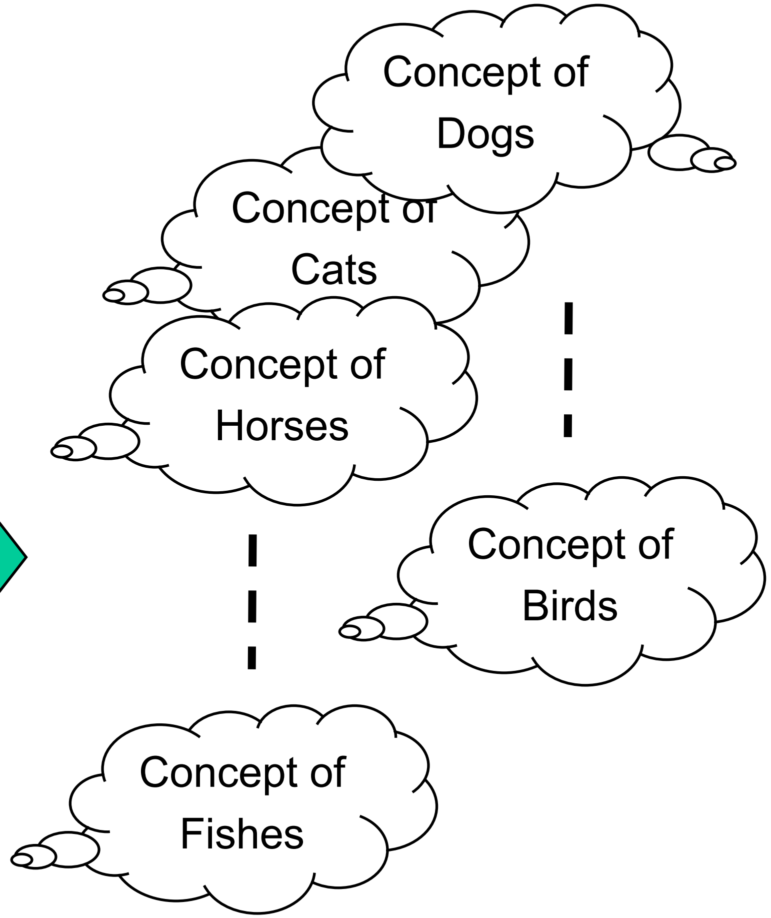
Concept Learning

Berlin Chen
Department of Computer Science & Information Engineering
National Taiwan Normal University

References:
1. Tom M. Mitchell, *Machine Learning* , Chapter 2
2. Tom M. Mitchell's teaching materials

What is a Concept ?



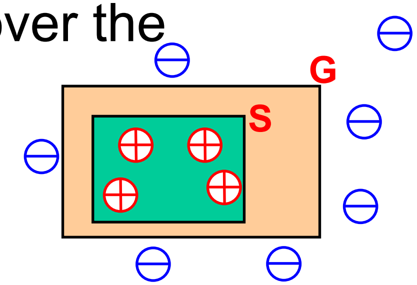


A concept describes a subset of objects or events defined over a larger set

Concept Learning

learning based on symbolic representations

- Acquire/Infer the definition of a general concept or category given a (labeled) sample of positive and negative training examples of the category
  - Each concept can be thought of as a Boolean-valued (true/false or yes/no) function
    - Approximate a Boolean-valued function from examples
  - 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
  - Take advantage of a naturally occurring structure over the hypothesis space
    - General-to-specific ordering of hypotheses



Training Examples for *EnjoySport*

- Concept to be learned
  - "Days on which Aldo enjoys his favorite water sport"

Attributes

Days

Sky	Temp	Humid	Wind	Water	Forecst	EnjoySpt
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

Concept to be learned

- Days (examples/instances) are represented by a set of attributes
- What is the general concept ?
  - The task is to learn to predict the value of *EnjoySport* for an arbitrary day based on the values of other attributes
  - Learn a (a set of) hypothesis representation(s) for the concept

Representing Hypotheses

- Many possible representations for hypotheses *h*
- Here *h* is conjunction of constraints on attributes
- Each constraint can be
  - A specific value (e.g., "*Water=Warm*")
  - Don't care (e.g., "*Water=?*")
  - No value acceptable (e.g., "*Water=Ø* ")

A hypothesis is
a vector of constraints

- For example

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

- Most general hypothesis

  < ? ? ? ? ? ? >

  All are positive examples

- Most specific hypothesis

  < Ø Ø Ø Ø Ø Ø >

  All are negative examples

Definition of Concept Learning Task

- Given
  - Instances X: possible days, each described by six attributes
    Sky, *AirTemp*, *Humidity*, *Wind*, *Water*, *Forecast*
    (*Sunny, Cloudy, Rainy*) (*Warm, Cold*) (*Normal, High*) (*Strong, Week*) (*Warm, Cool*) (*Same, Change*)
  - Target concept/function *c* : *EnjoySport* X → {0, 1}
    "No" "Yes"
  - Hypotheses *H* : Conjunctions of Literals. E.g.,

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

  - Training examples *D* : Positive and negative examples (members/nonmembers) of the target function (concept)

    $<x_1,c(x_1)>, <x_2,c(x_2)>, \ldots, <x_m,c(x_m)>$

    target concept value
- Determine
  - A hypothesis *h* in *H* (an approximate target function) such that
    $h(x)=c(x)$ for all x in D

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
  - Assumption of Inductive Learning
    - The best hypothesis regarding the unseen instances is the hypothesis that best fits the observed training data

Viewing Learning As a Search Problem (1/2)

- Concept learning can be viewed as the task of searching through a large space of hypotheses

Instance space *X*

Sky (*Sunny/Cloudy/Rainy*)
AirTemp (*Warm/Cold*)
Humidity (*Normal/High*)
Wind (*Strong/Weak*)
Water (*Warm/Cool*)
Forecast (*Same/Change*)

=> 3*2*2*2*2*2=96 instances

Hypothesis space *H*

5*4*4*4*4*4=5120 syntactically distinct hypotheses

Ø

1+4*3*3*3*3*3=973 semantically distinct hypotheses

Each hypothesis is represented as a conjunction of constraints

E.g.,
< Ø Warm Normal Strong Cool Same >
< Sunny Ø Normal Strong Cool Change>

Viewing Learning As a Search Problem (2/2)

- Study of learning algorithms that examine different strategies for searching the hypothesis space, e.g.,
  - *Rote Learner*
  - *Find-S* Algorithm
  - *List-Then-Eliminate* Algorithm
  - *Candidate Elimination* Algorithm

- How to exploit the naturally occurring structure in the hypothesis apace ?
  - Relations among hypotheses , e.g.,
    - General-to-Specific-Ordering

Rote Learner

- Learning corresponding simply to storing each observed training example in memory
- Subsequent instances are classified by looking them up in memory
  - If found, the stored classification is returned
  - Otherwise, the system refuses to classify the new instances

General-to-Specific-Ordering of Hypothesis (1/3)

- Many concept learning algorithms organize the search through the hypothesis space by taking advantage of a naturally occurring structure over it
  - *"general-to-specific ordering"*

$$h_1 = <Sunny, ?, ?, Strong, ?, ?>$$
$$h_2 = <Sunny, ?, ?, ?, ?, ?>$$

Suppose that h_1 and h_2 classify positive examples

- *h_2 is more general than h_1*
  - h_2 imposes fewer constraints on instances
  - h_2 classify more positive instances than h_1 does

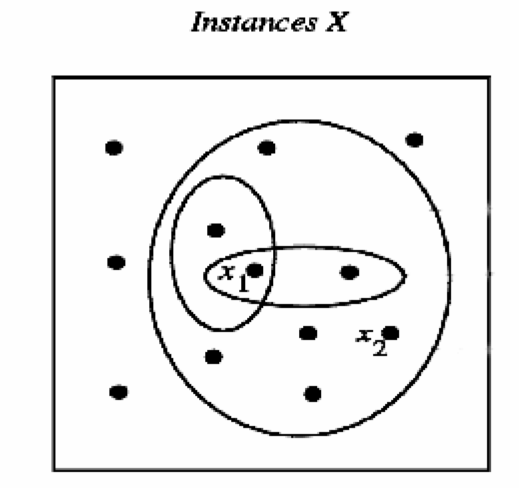
– A useful structure over the hypothesis space

General-to-Specific-Ordering of Hypothesis (2/3)

- *More-General-Than* Partial Ordering
  - Definition
    - Let h_j and h_k be Boolean-valued functions defined over X. Then h_j is *more general than* h_k ($h_j >_g h_k$) if and only if

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

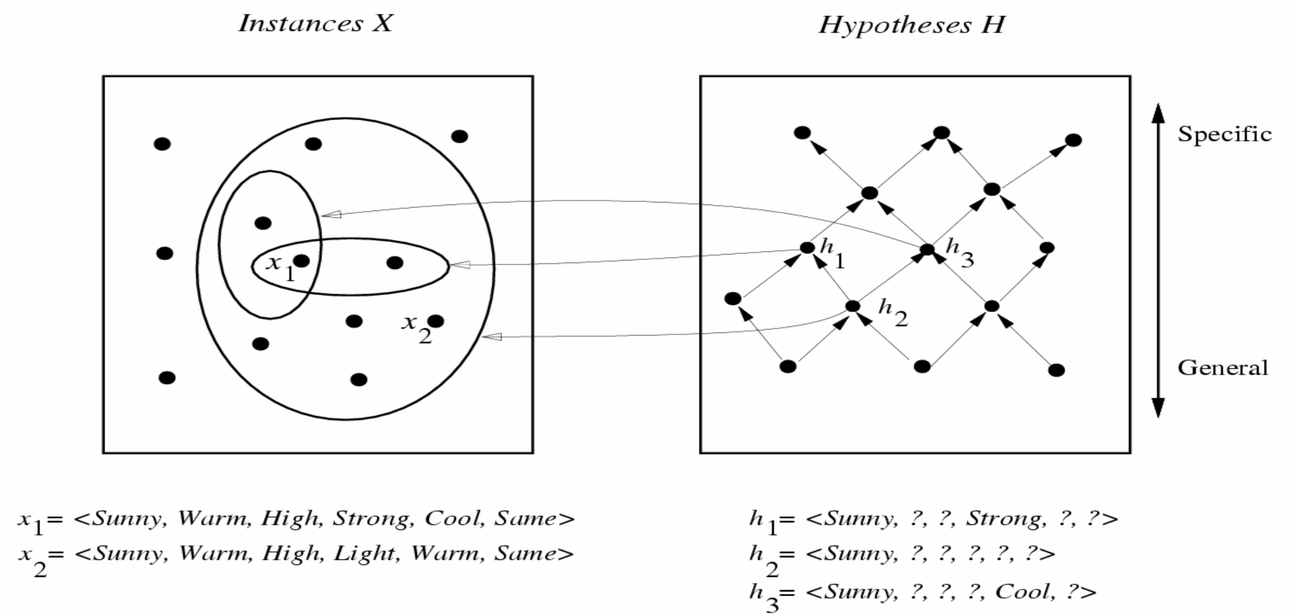
x satisfies h_k



  - We also can define the *more-specific-than* ordering

General-to-Specific Ordering of Hypotheses (3/3)

- An illustrative example



- Suppose instances are classified positive by h_1, h_2 , h_3
  - h_2 (imposing fewer constraints) are *more general than* h_1 and h_3
  - $h_1 \overset{?}{\longleftrightarrow} h_3$

partial order relation

- antisymmetric, transitive

$$h_a \geq_g h_b,\ h_b \geq_g h_c \ \Rightarrow\ h_a \geq_g h_c$$

Find-S Algorithm (1/3)

- Find a maximally specific hypothesis by using the *more-general-than* partial ordering to organize the search for a hypothesis consistent with the observed training examples

$$h \leftarrow \langle \phi, \phi, \phi, \phi, \phi, \phi \rangle$$

1. Initialize h to the most specific hypothesis in H
2. For each positive training instance x
   - For each attribute constraint a_i in h

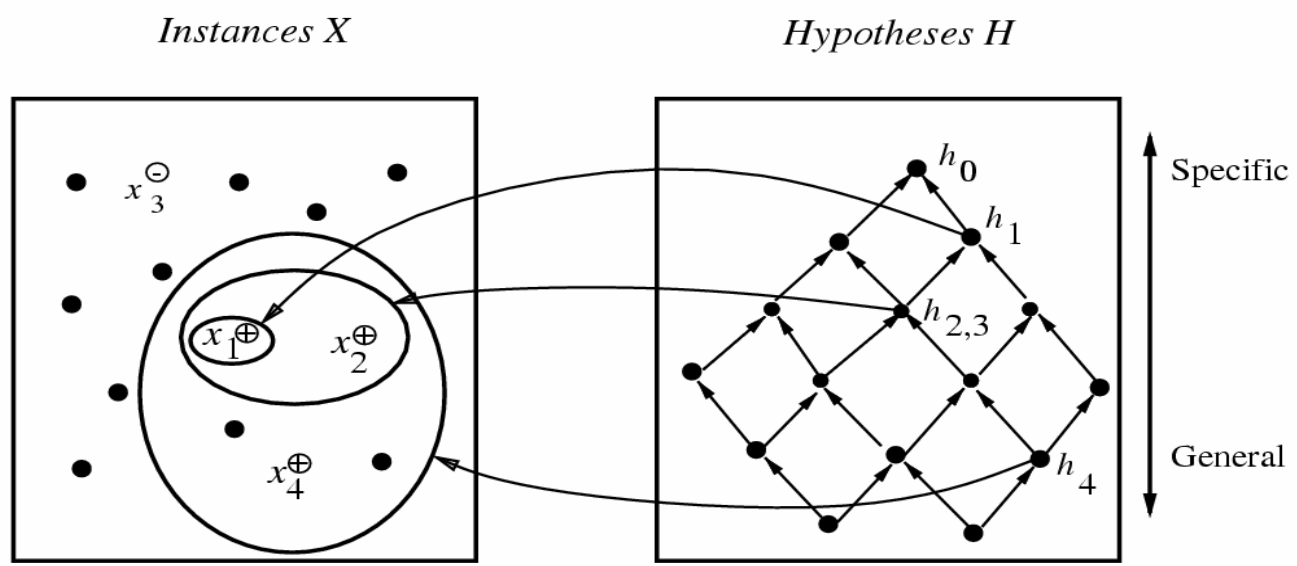
     If the constraint a_i in h is satisfied by x

     Then do nothing

     Else replace a_i in h by the next more general constraint that is satisfied by x
3. Output hypothesis h

Find-S Algorithm (2/3)

- Hypothesis Space Search by ***Find-S***



h_0 = <∅, ∅, ∅, ∅, ∅, ∅>

x_1 = <*Sunny Warm Normal Strong Warm Same*>, + h_1 = <*Sunny Warm Normal Strong Warm Same*>

x_2 = <*Sunny Warm High Strong Warm Same*>, + h_2 = <*Sunny Warm ? Strong Warm Same*>

x_3 = <*Rainy Cold High Strong Warm Change*>, - h_3 = <*Sunny Warm ? Strong Warm Same*>   no change!

x_4 = <*Sunny Warm High Strong Cool Change*>, + h_4 = <*Sunny Warm ? Strong ? ?* >

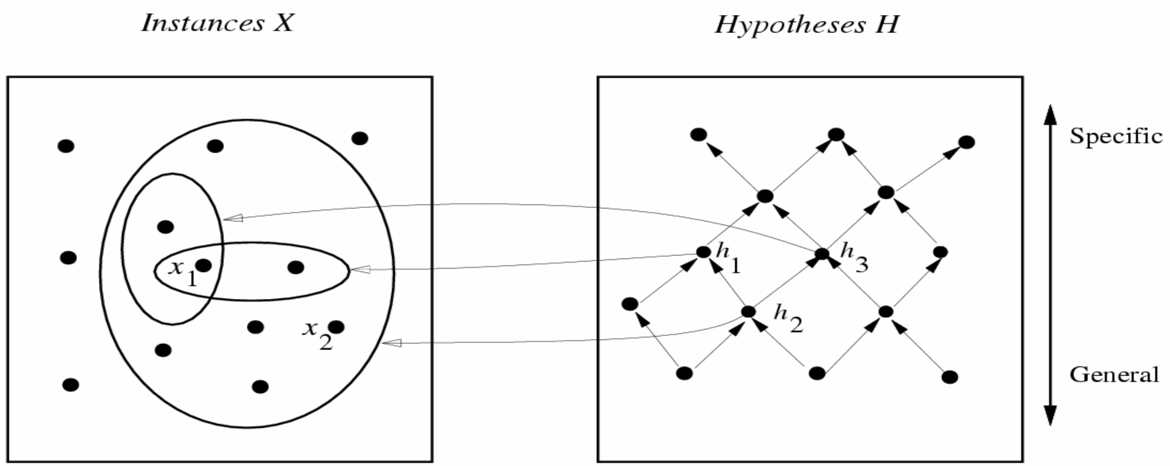
  – Substitute a “?” in place of any attribute value in *h* that is not satisfied by the new example

Find-S Algorithm (3/3)

- Why *F-S* never check a negative example ?
  - The hypothesis *h* found by it is the most specific one in *H*
  - Assume the target concept *c* is also in *H* which will cover both the training and unseen positive examples
    - *c* is more general than *h*

can be represented as
a conjunction of attributes

  - Because the target concept will not cover the negative examples, thus neither will the hypothesis *h*



Complaints about *Find-S*

- Can not tell whether it has learned concept
  (Output only one. Many other consistent hypotheses may exist!)

- Picks a maximally specific *h* (why?)
  (Find a most specific hypothesis consistent with the training data)

- Can not tell when training data inconsistent
  - What if there are noises or errors contained in training examples

- Depending on *H*, there might be several !

Consistence of Hypotheses

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

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

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

Version Space

Mitchell 1977

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

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

  - A subspace of hypotheses
  - Contain all plausible versions (描述) of the target concepts

List-Then-Eliminate Algorithm

1. *VersionSpace* $\leftarrow$ a list containing all hypotheses in *H*

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

   remove from *VersionSpace* any hypothesis *h* for which

   $h(x) \neq c(x)$

   – i.e., eliminate hypotheses inconsistent with any training examples
   – The *VersionSpace* shrinks as more examples are observed

3. Output the list of hypotheses in *VersionSpace*

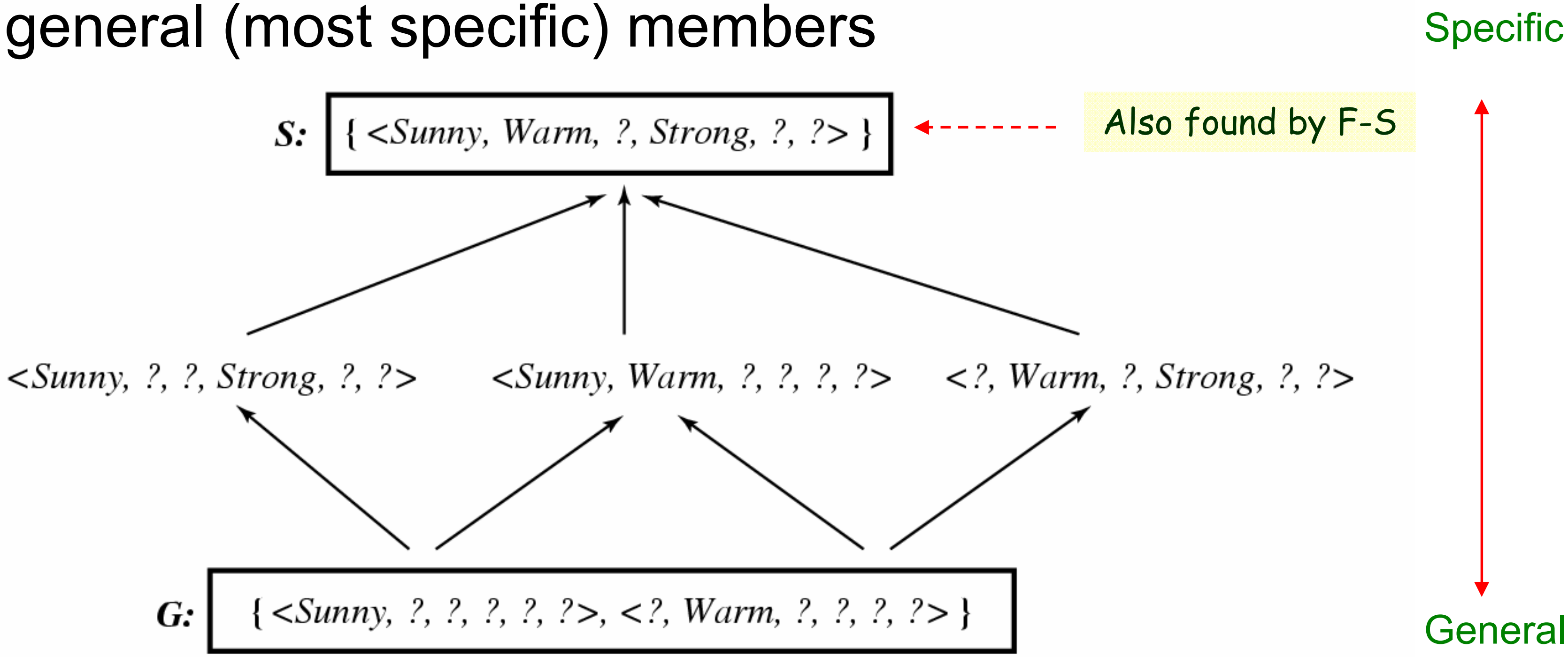
Drawbacks of *List-Then-Eliminate*

- The algorithm requires exhaustively enumerating all hypotheses in *H*
  - An unrealistic approach ! (full search)

- If insufficient (training) data is available, the algorithm will output a huge set of hypotheses consistent with the observed data

Example Version Space

- Employ a much more compact representation of the version space in terms of its most general and least general (most specific) members



Sky	Temp	Humid	Wind	Water	Forecst	EnjoySpt
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

Arrows mean more-general-than relations

Representing Version Space

- The **General boundary** *G,* of version space $VS_{H,D}$ is the set of its maximally general members

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

- The **Specific boundary** *S,* of version space $VS_{H,D}$ is the set of its maximally specific members

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

- Every member of the version space lies between these boundaries

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

  - Version Space Representation Theorem

Candidate Elimination Algorithm (1/3)

Mitchell 1979

- G $\leftarrow$ maximally general hypotheses in H

$$G_0 \leftarrow \{\langle ?, ?, ?, ?, ?, ? \rangle\}$$

Should be specialized

- S $\leftarrow$ maximally specific hypotheses in H

$$S_0 \leftarrow \{\langle \phi, \phi, \phi, \phi, \phi, \phi \rangle\}$$

Should be generalized

Candidate Elimination Algorithm (2/3)

- 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*
        (i.e., partial-ordering relations exist)

positive training examples force the S boundary become increasing general

Candidate Elimination Algorithm (3/3)

– 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*
    (i.e., partial-ordering relations exist)

negative training examples force the *G* boundary become increasing specific

Example Trace (1/5)

S_0: {<Ø, Ø, Ø, Ø, Ø, Ø>}

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

Example Trace (2/5)

S_0: {<∅, ∅, ∅, ∅, ∅, ∅>}

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

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

G_0, G_1, G_2: {<*?, ?, ?, ?, ?, ?*>}

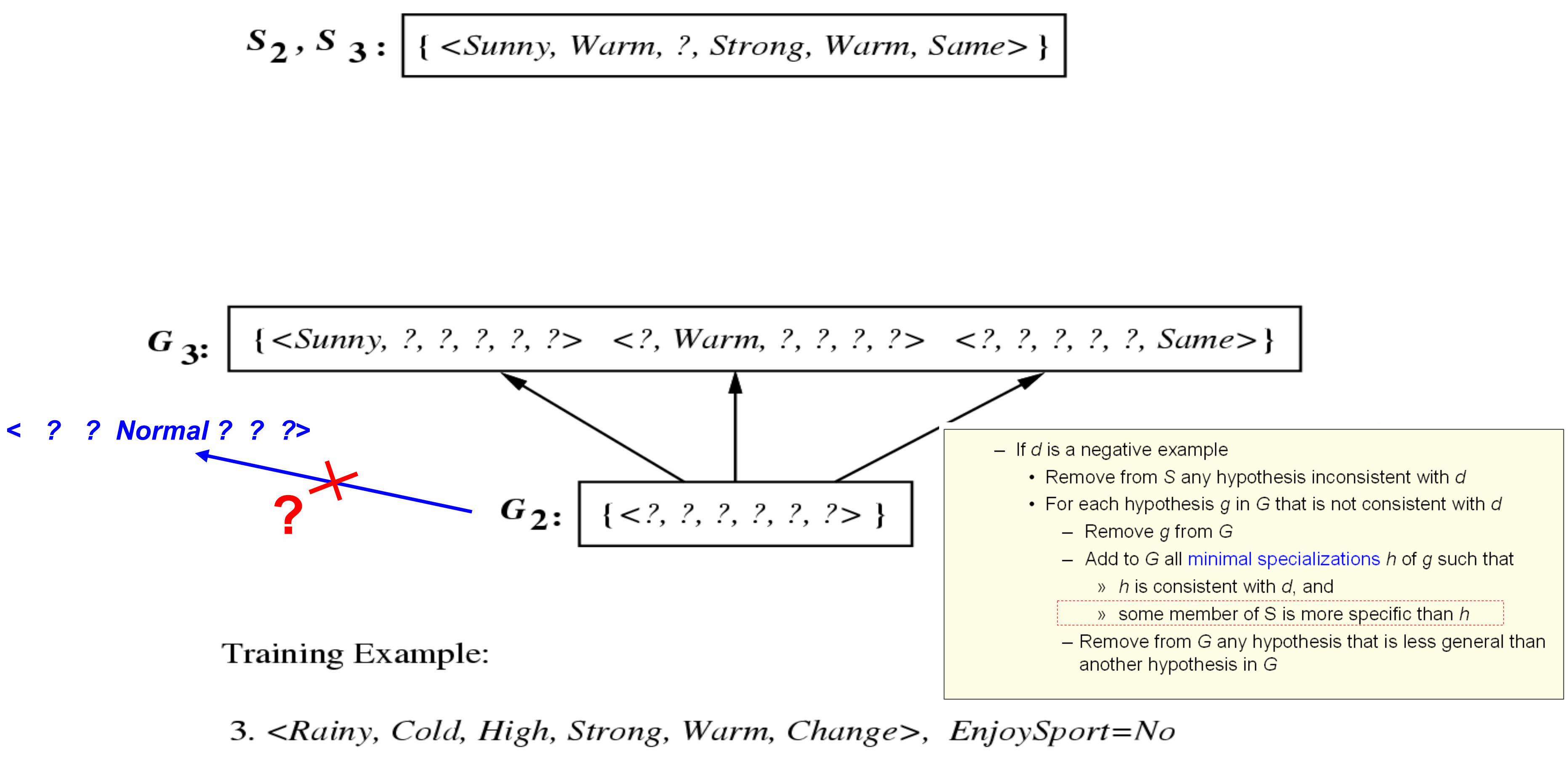
- For each training example *d*, do
  - If *d* is a positive example
    - Remove from *G* any hypothesis inconsistent with *d*
    - For each hypothesis *s* in *S* that is not consistent with *d*
      - Remove *s* from *S*
      - Add to *S* all minimal generalizations *h* of *s* such that
        - » *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* (i.e., partial-ordering relations exist)

Training examples:

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

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

Example Trace (3/5)



- G_2 has six ways to be minimally specified
  - Why <*?,? ,Normal,?,?,?* > etc. do not exist in G_3 ?

Example Trace (4/5)

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

↓

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

- 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* (i.e., partial-ordering relations exist)

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

↑

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

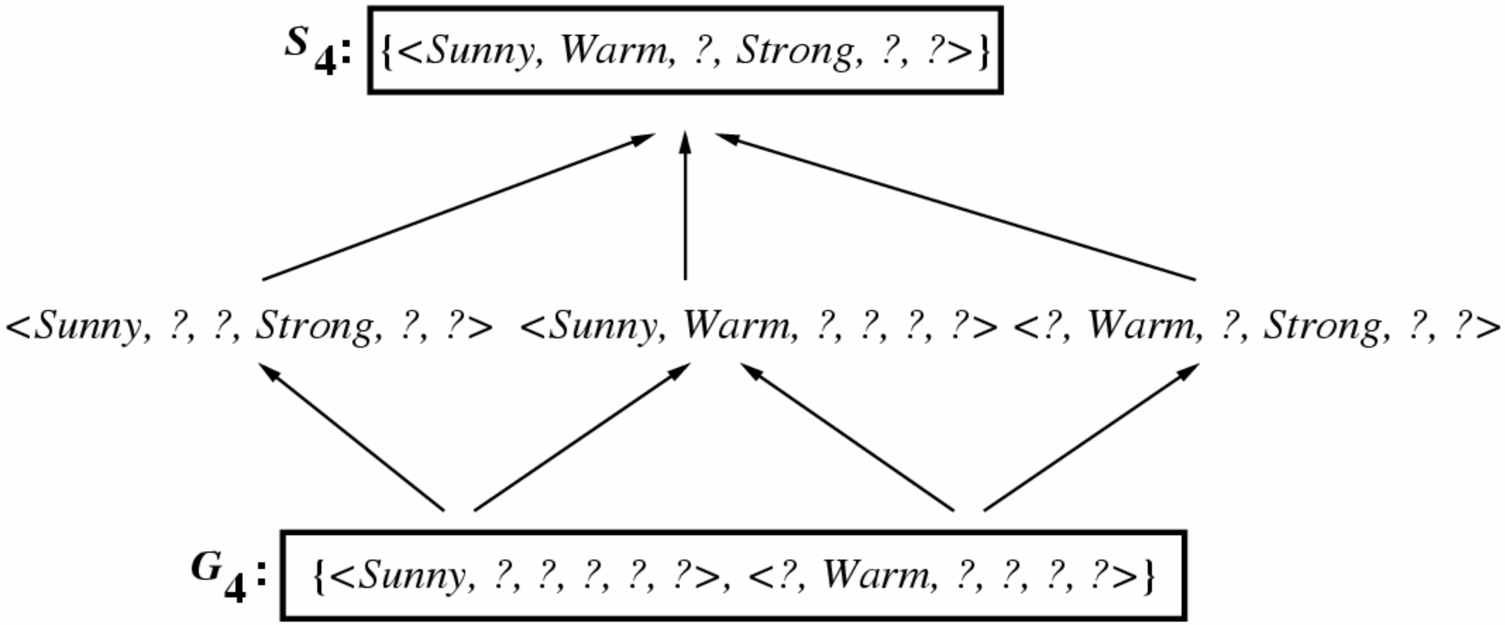
can not be specialized nor generalized?

Training Example:

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

- Notice that,
  - *S* is a summary of the previously positive examples
  - *G* is a summary of the previously negative examples

Example Trace (5/5)

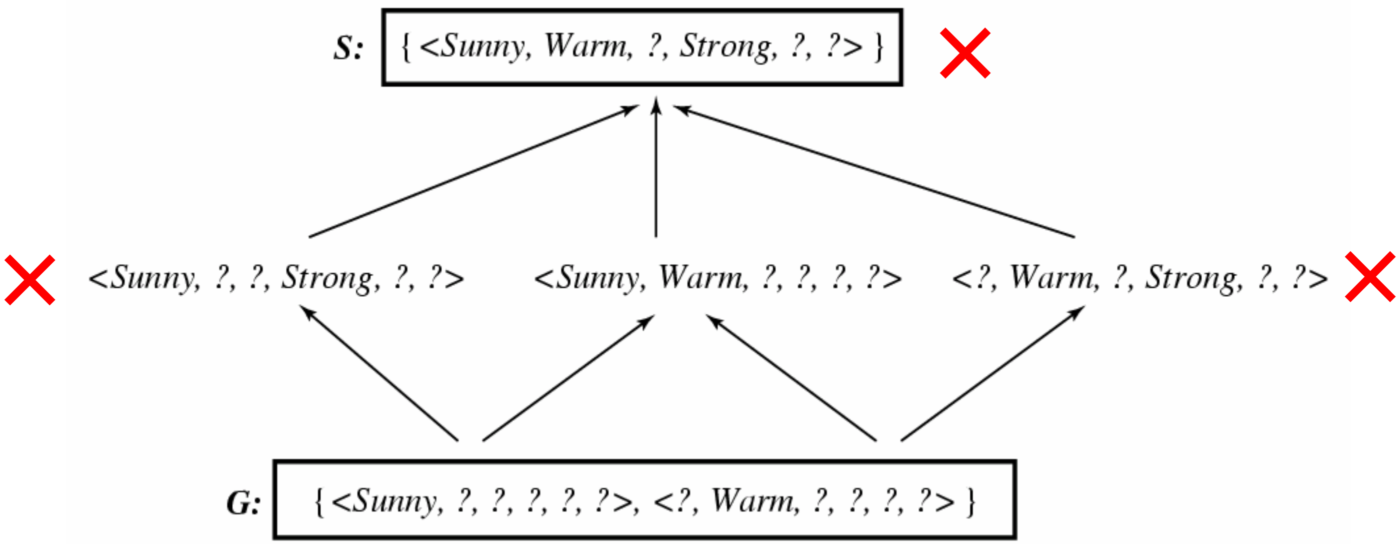


- *S* and *G* boundaries move monotonically closer to each other, delimiting a smaller and smaller version space

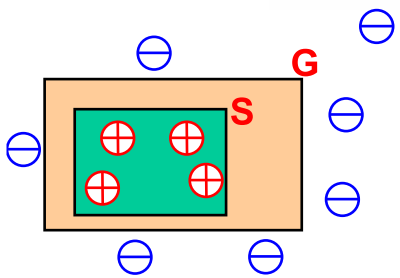
What Next Training Example

- Learner can generate useful queries
  - Discriminate among the alternatives competing hypotheses in the current version space

Specific



General

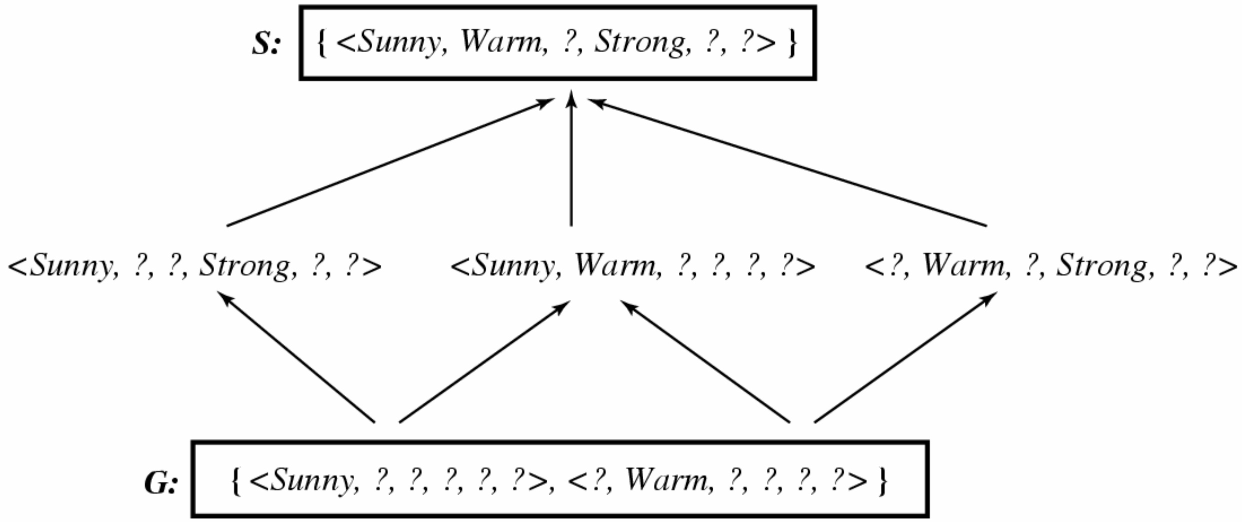


If a positive hypothesis is posed:

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

What if it is a negative one ?

How Should These Be Classified ?



Instance	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*	
A	Sunny	Warm	Normal	Strong	Cool	Change	?	Yes ?
B	Rainy	Cold	Normal	Light	Warm	Same	?	No ?
C	Sunny	Warm	Normal	Light	Warm	Same	?	?
D	Sunny	Cold	Normal	Strong	Warm	Same	?	?

majority vote?

Biased Hypothesis Space

- Biased hypothesis space
  - Restrict the hypothesis space to include only conjunctions of attribute values
  - I.e., bias the learner to consider only conjunctive ypothesis
- Cannot represent disjunctive target concepts

"*Sky=Sunny* or *Sky=Cloud* "

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

After the first two examples learned:

<?, Warm, Normal, Strong, Cool, Change>

Summary Points

- Concept learning as search through *H*
- General-to-specific ordering over *H*
- Version space candidate elimination algorithm
  - *S* and *G* boundaries characterize learners uncertainty
- Learner can generate useful queries