Concept Learning

Waiting outside the house to get an autograph.

Which days does he come out to enjoy sports?

- Sky condition
- Humidity
- Temperature
- Wind
- Water
- Forecast

- Attributes of a day: takes on values

Learning Task

- We want to make a hypothesis about the day on which SRK comes out..
  - in the form of a boolean function on the attributes of the day.

- Find the right hypothesis/function from historical data

Training Examples for EnjoySport

	Sky	Temp	Humid	Wind	Water	Forecst		EnjoySpt
c(	Sunny	Warm	Normal	Strong	Warm	Same	)=1	Yes
c(	Sunny	Warm	High	Strong	Warm	Same	)=1	Yes
c(	Rainy	Cold	High	Strong	Warm	Change	)=0	No
c(	Sunny	Warm	High	Strong	Cool	Change	)=1	Yes

- Negative and positive learning examples
- Concept learning:

  c is the target concept

  - Deriving a Boolean function from training examples
    - Many "hypothetical" boolean functions
      - Hypotheses; find h such that h = c.
  - Other more complex examples:
    - Non-boolean functions
- Generate hypotheses for concept from TE's

Representing Hypotheses

- Task of finding appropriate set of hypotheses for concept given training data
- Represent hypothesis as Conjunction of constraints of the following form:
  - Values possible in any hypothesis
    - Specific value : Water = *Warm*
    - Don't-care value: Water = ?
    - No value allowed : Water = ∅
      - i.e., no permissible value given values of other attributes
  - Use vector of such values as hypothesis:
    - ⟨ Sky AirTemp Humid Wind Water Forecast ⟩
      - Example: ⟨*Sunny* ? ? *Strong* ? *Same* ⟩
- Idea of *satisfaction of hypothesis* by some example
  - say "example satisfies hypothesis"
  - defined by a function *h(x):*

$$h(x) = 1 \text{ if } h \text{ is true on } x$$

$$= 0 \text{ otherwise}$$

- Want hypothesis that best fits examples:
  - Can reduce learning to search problem over space of hypotheses

Prototypical Concept Learning Task

TASK T: predicting when person will enjoy sport

- **Target function *c***: EnjoySport : $X \rightarrow \{0, 1\}$
- Cannot, in general, know Target function c
  - Adopt hypotheses H about c
- Form of hypotheses *H*:
  - Conjunctions of literals $\langle$?, Cold, High, ?, ?, ? $\rangle$

- **EXPERIENCE E**
  - **Instances *X***: possible days described by attributes *Sky*, *AirTemp*, *Humidity*, *Wind*, *Water*, *Forecast*
  - **Training examples *D***: Positive/negative examples of target function $\{\langle x_1, c(x_1)\rangle, \ldots \langle x_m, c(x_m)\rangle\}$
- **PERFORMANCE MEASURE P**: Hypotheses *h* in *H* such that $h(x) = c(x)$ for all x in D ()
  - There may exist several alternative hypotheses that fit examples

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

Approaches to learning algorithms

The choice of the hypothesis space reduces the number of hypotheses.

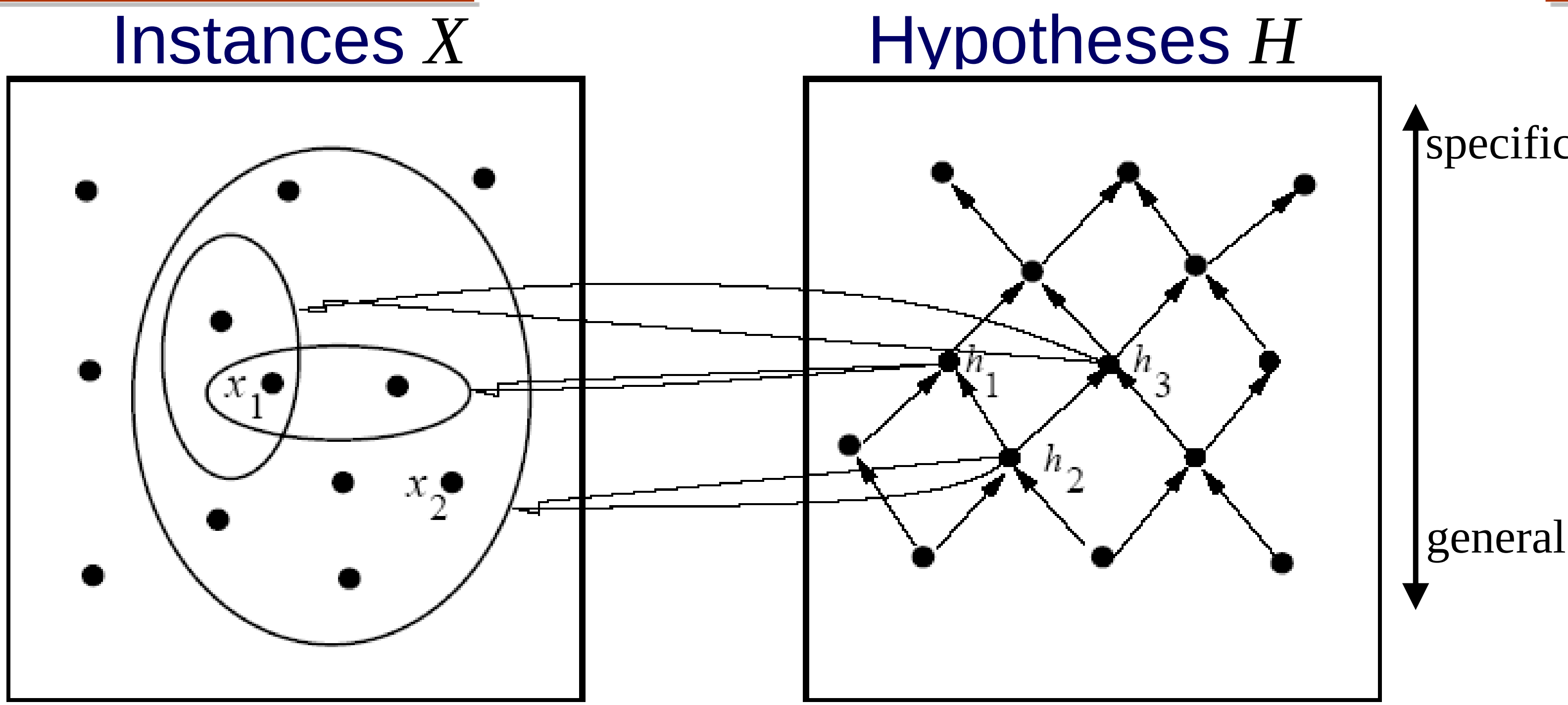
- Brute force search
  - Enumerate all possible hypotheses and evaluate
  - Highly inefficient even for small *EnjoySport* example
    - |X| = 3.2.2.2.2= 96 distinct *instances*
    - Large number of *syntactically distinct* hypotheses (0's, ?'s)
      - EnjoySport: |H| = 5.4.4.4.4.4=5120
      - Fewer when consider h's with 0's

        Every h with a 0 is empty set of instances (classifies instance as neg)

        Hence # semantically distinct h's is:

        1+ (4.3.3.3.3.3) = 973
    - EnjoySport is VERY small problem compared to many
- Hence use other search procedures.
  - Approach 1: Search based on ordering of hypotheses
  - Approach 2: Search based on finding all possible hypotheses using a good representation of hypothesis space
    - All hypotheses that fit data

Ordering on Hypotheses



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 ?*⟩

- h is more general than h'($h \geq_g h'$) if for each instance x,

$$h'(x) = 1 \rightarrow h(x) = 1$$

- Which is the most general/most specific hypothesis?

Find-S Algorithm

Assumes

There is hypothesis h in H describing target function c

There are no errors in the TEs

Procedure

1. Initialize *h* to the most specific hypothesis in *H (what is this?)*
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*

   do nothing

   Else

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

Note

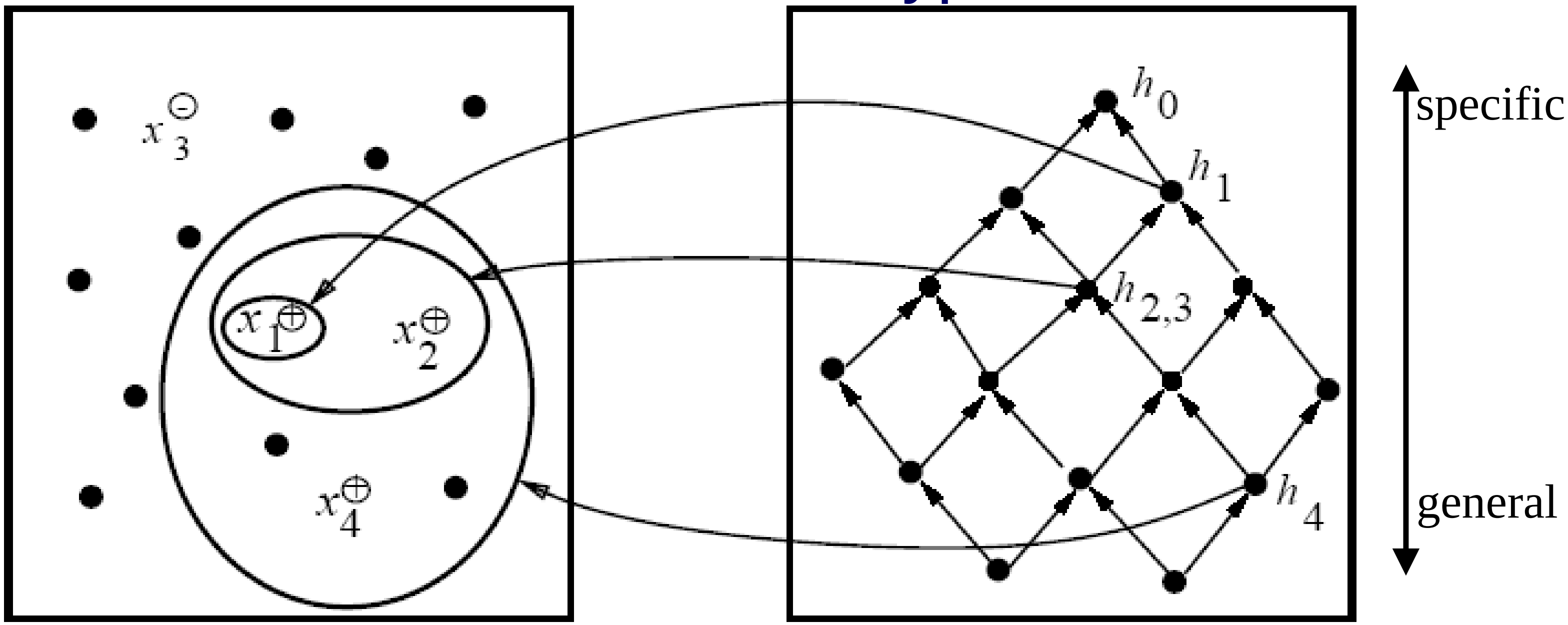
There is no change for a negative example, so they are ignored.

This follows from assumptions that there is h in H describing target function c *(ie.,for this h, h=c)* *and* that there are no errors in data. In particular, it follows that the hypothesis at any stage cannot be changed by neg example.

Assumption: Everything except the positive examples is negative

Example of Find-S

Instances X | Hypotheses H



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 ? ?*⟩

Problems with Find-S

- Problems:
  - Throws away information!
    - Negative examples
  - Can’t tell whether it has learned the concept
    - Depending on *H*, there might be several h’s that fit TEs!
    - Picks a maximally specific *h* (why?)
  - Can’t tell when training data is inconsistent
    - Since ignores negative TEs
- But
  - It is simple
  - Outcome is independent of order of examples
    - Why?
- What alternative overcomes these problems?
  - Keep *all* consistent hypotheses!
    - Candidate elimination algorithm

Consistent Hypotheses and Version Space

- A hypothesis h is consistent with a set of training examples D of target concept c if $h(x) = c(x)$ for each training example $\langle x, c(x)\rangle$ in D
  - Note that consistency is with respect to specific D.
- Notation:

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

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

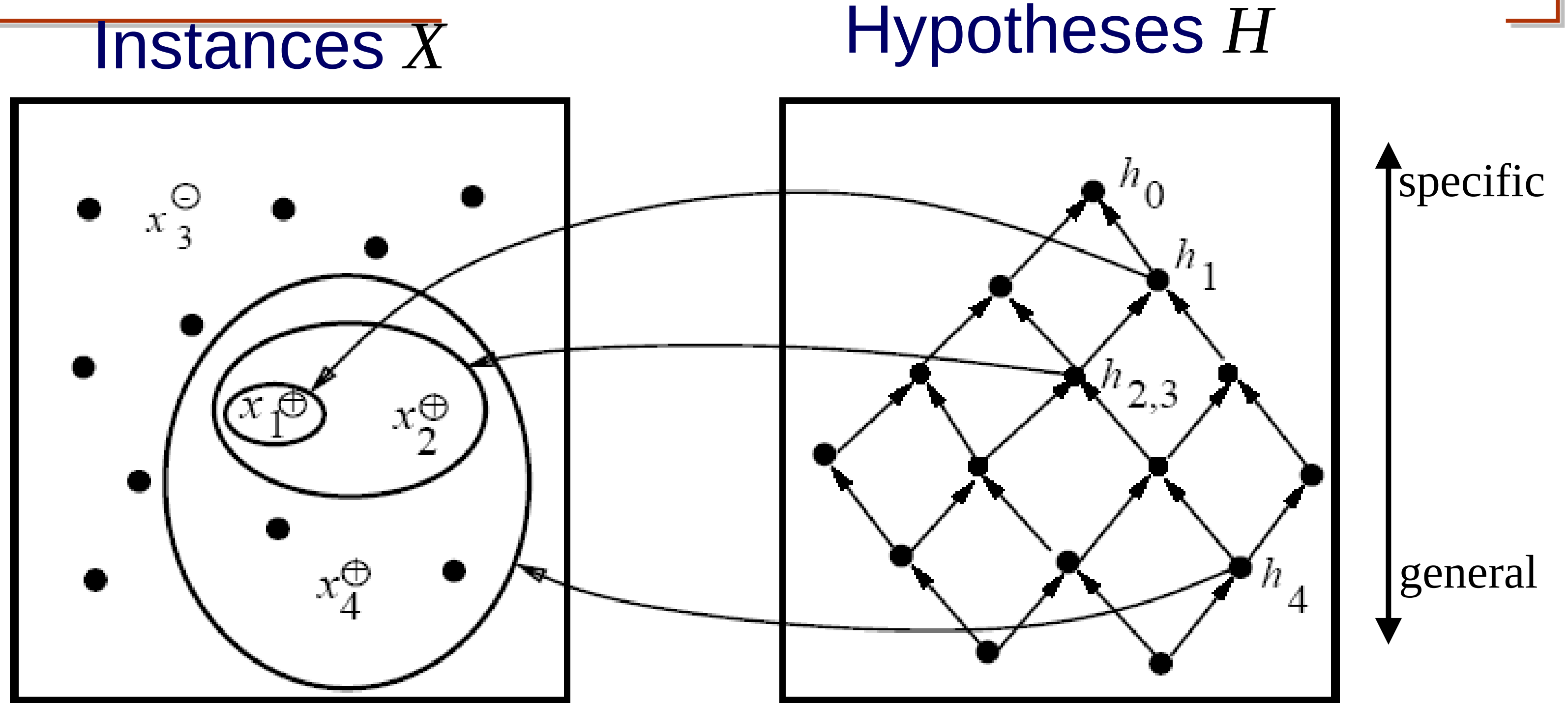
$$VS_{H,D} = \{h \mid h \in H \wedge Consistent(h, D)\}$$

List-Then-Eliminate Algorithm

1. *VersionSpace* $\leftarrow$ list of all hypotheses 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*
4. This is essentially a brute force procedure

Example of Find-S, Revisited



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_3=⟨*Sunny Warm High Strong Cool Change*⟩ +

h_0=⟨∅ ∅ ∅ ∅ ∅ ∅⟩

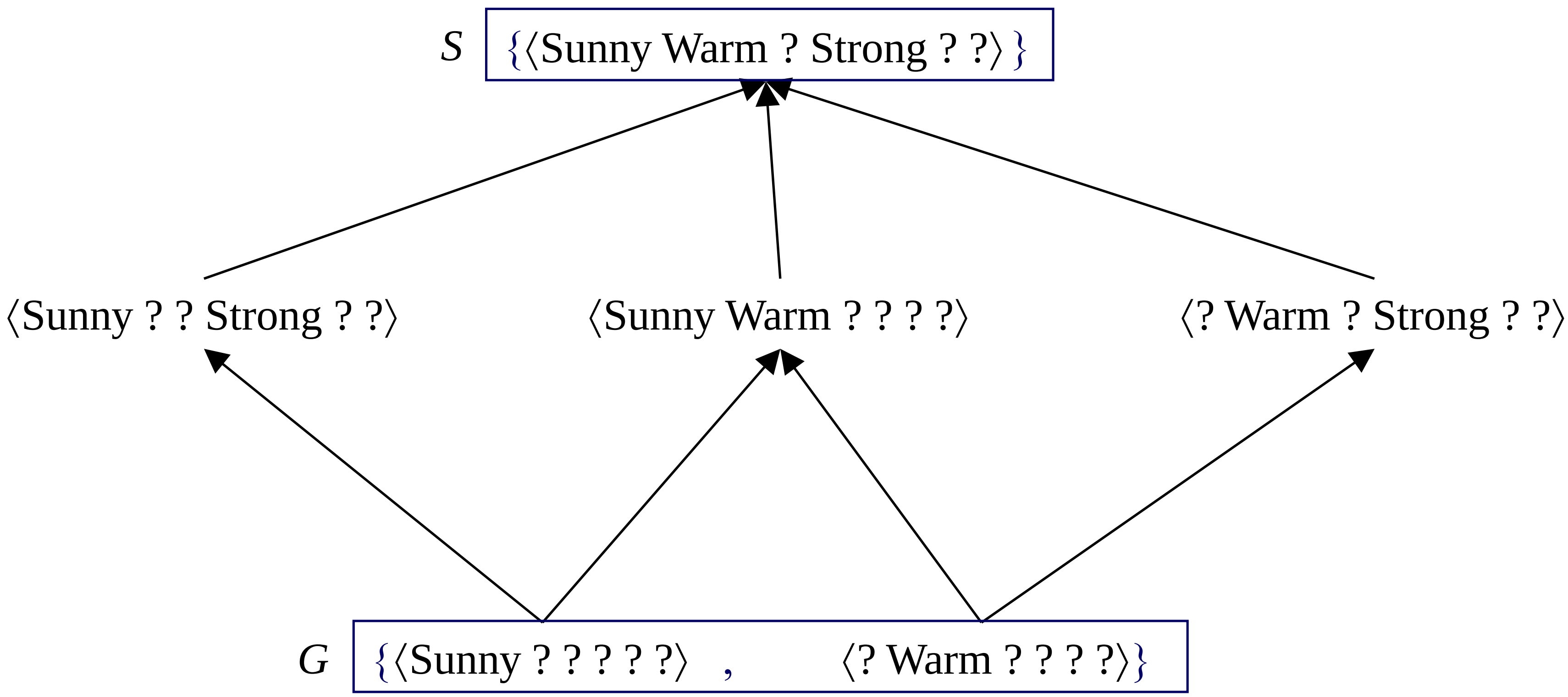
h_1=⟨*Sunny Warm Normal Strong Warm Same*⟩

h_2=⟨*Sunny Warm ? Strong Warm Same*⟩

h_3=⟨*Sunny Warm ? Strong Warm Same*⟩

h_4=⟨*Sunny Warm ? Strong ? ?*⟩

Version Space for this Example



Representing Version Spaces

- Want more compact representation of VS
  - Store most/least general boundaries of space
  - Generate all intermediate h's in VS
  - Idea that any h in VS must be consistent with all TE's
    - Generalize from most specific boundaries
    - Specialize from most general boundaries
- The general boundary, *G*, of version space $VS_{H,D}$ is the set of its maximally general members consistent with D
  - Summarizes the negative examples; anything more general will cover a negative TE
- The specific boundary, *S*, of version space $VS_{H,D}$ is the set of its maximally specific members consistent with D
  - Summarizes the positive examples; anything more specific will fail to cover a positive TE

Theorem

Every member of the version space lies between the S,G boundary

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

- Must prove:
  - 1) every h satisfying RHS is in $VS_{H,D;}$
  - 2) every member of $VS_{H,D}$ satisfies RHS.
- For 1), let g, h, s be arbitrary members of G, H, S respectively with g>h>s
  - s must be satisfied by all + TEs and so must h because it is more general;
  - g cannot be satisfied by any – TEs, and so nor can h
  - h is in $VS_{H,D}$ since satisfied by all + TEs and no – TEs
- For 2),
  - Since h satisfies all + TEs and no – TEs, h ≥ s, and $g \geq h$.

Candidate Elimination Algorithm

$G \leftarrow$ maximally general hypotheses in H
$S \leftarrow$ maximally specific hypotheses in H

For each training example d, do

- If d is positive
  - Remove from G every hypothesis inconsistent with d
  - For each hypothesis s in S that is inconsistent with d
    - Remove s from S
    - Add to S all minimal generalizations h of s such that

      1. h is consistent with d, and

      2. some member of G is more general than h
  - Remove from S every hypothesis that is more general than another hypothesis in S

Candidate Elimination Algorithm (cont)

- If *d* is a negative example
  - Remove from *S* every hypothesis inconsistent with *d*
  - For each hypothesis *g* in *G* that is inconsistent with *d*
    - Remove *g* from *G*
    - Add to *G* all minimal specializations *h* of *g* such that

      1. *h* is consistent with *d*, and
      2. some member of *S* is more specific than *h*
  - Remove from *G* every hypothesis that is less general than another hypothesis in *G*
- Essentially use
  - Pos TEs to generalize S
  - Neg TEs to specialize G
- Independent of order of TEs
- Convergence guaranteed if:
  - ***no errors***
  - ***there is h in H describing c*.**

Example

Recall : If d is positive

Remove from *G* every hypothesis inconsistent with *d*

For each hypothesis s in S that is inconsistent with d

- Remove s from S
- Add to S all minimal generalizations h of s that are specializations of a hypothesis in G
- Remove from S every hypothesis that is more general than another hypothesis in S

S_0 $\boxed{\{\langle \varnothing\ \varnothing\ \varnothing\ \varnothing\ \varnothing\ \varnothing \rangle\}}$

G_0 $\boxed{\{\langle ?\ ?\ ?\ ?\ ?\ ? \rangle\}}$

$\langle \textit{Sunny Warm Normal Strong Warm Same} \rangle$ +

S_1 $\boxed{\{\langle \text{Sunny Warm Normal Strong Warm Same} \rangle\}}$

G_1 $\boxed{\{\langle ?\ ?\ ?\ ?\ ?\ ? \rangle\}}$

Example (contd)

S_1 $\boxed{\{\langle \text{Sunny Warm Normal Strong Warm Same} \rangle\}}$

G_1 $\boxed{\{\langle ?\ ?\ ?\ ?\ ?\ ? \rangle\}}$

$\langle \textit{Sunny Warm High Strong Warm Same} \rangle$ **+**

S_2 $\boxed{\{\langle \text{Sunny Warm ? Strong Warm Same} \rangle\}}$

G_2 $\boxed{\{\langle ?\ ?\ ?\ ?\ ?\ ? \rangle\}}$

Example (contd)

S_2 $\boxed{\{\langle \text{Sunny Warm ? Strong Warm Same} \rangle\}}$

G_2 $\boxed{\{\langle \text{? ? ? ? ? ?} \rangle\}}$

Recall: If d is a negative example

- Remove from S every hypothesis inconsistent with d
- For each hypothesis g in G that is inconsistent with d
  - ❖ Remove g from G
  - ❖ Add to G all minimal specializations h of g that generalize some hypothesis in S
  - ❖ Remove from G every hypothesis that is less general than another hypothesis in G

$\langle$*Rainy Cold High Strong Warm Change*$\rangle$ –

Current G boundary is incorrect
So, need to make it more specific.

S_3 $\boxed{\{\langle \text{Sunny Warm ? Strong Warm Same} \rangle\}}$

G_3 $\boxed{\{\langle \text{Sunny ? ? ? ? ?} \rangle, \langle \text{? Warm ? ? ? ?} \rangle, \langle \text{? ? ? ? ? Same} \rangle\}}$

Example (contd)

- Why are there no hypotheses left relating to:
  - ⟨ *Cloudy ? ? ? ? ?*⟩
- The following specialization using the third value

⟨*? ? Normal ? ? ?*⟩,

is not more general than the specific boundary

{⟨Sunny Warm ? Strong Warm Same⟩}

- The specializations ⟨*? ? ? Weak ? ?*⟩,

⟨*? ? ? ? Cool ?*⟩ are also inconsistent with S

Example (contd)

S_3 $\boxed{\{\langle \text{Sunny Warm ? Strong Warm Same} \rangle\}}$

G_3 $\boxed{\{\langle \text{Sunny ? ? ? ? ?} \rangle, \langle \text{? Warm ? ? ? ?} \rangle, \langle \text{? ? ? ? ?Same} \rangle\}}$

$\langle \textit{Sunny Warm High Strong Cool Change} \rangle$ +

S_4 $\boxed{\{\langle \text{Sunny Warm ? Strong ? ?} \rangle\}}$

G_4 $\boxed{\{\langle \text{Sunny ? ? ? ? ?} \rangle, \langle \text{? Warm ? ? ? ?} \rangle\}}$

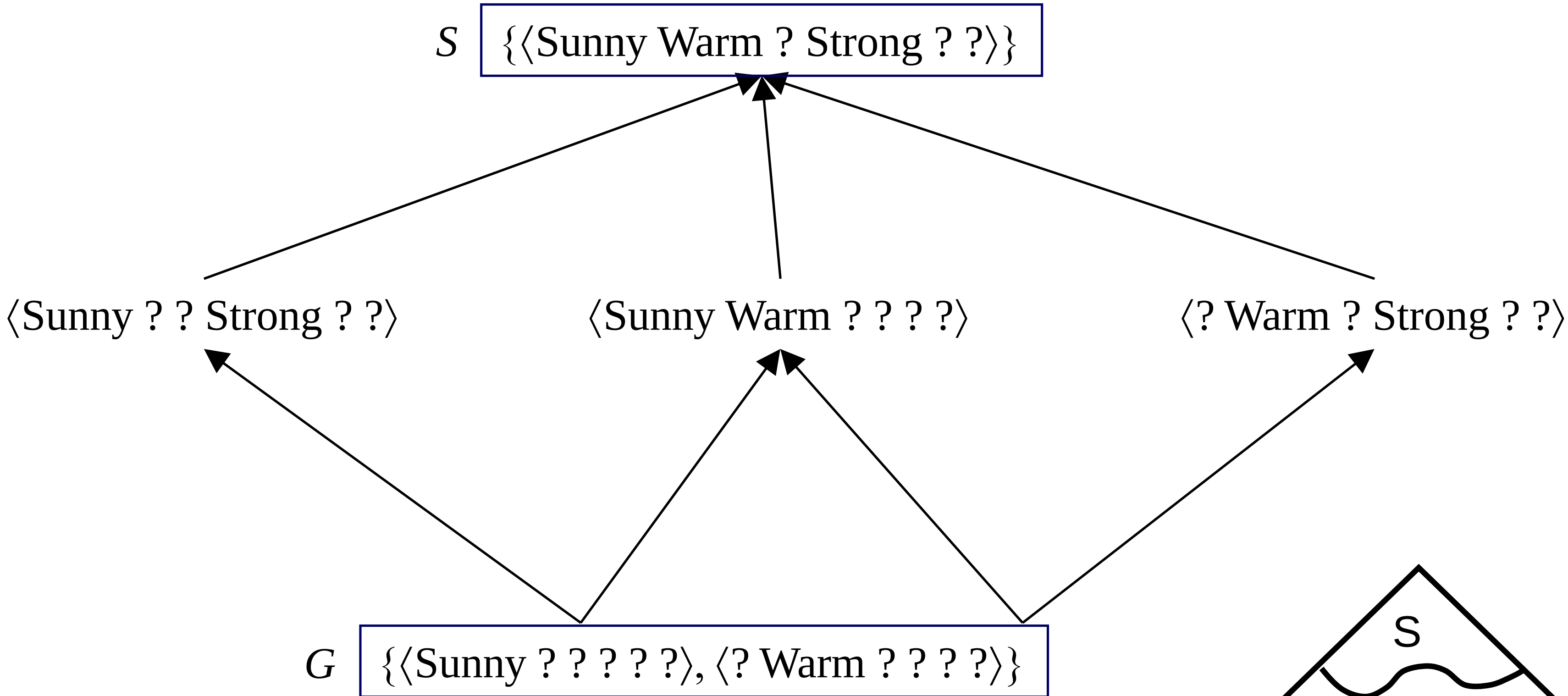
Example (contd)

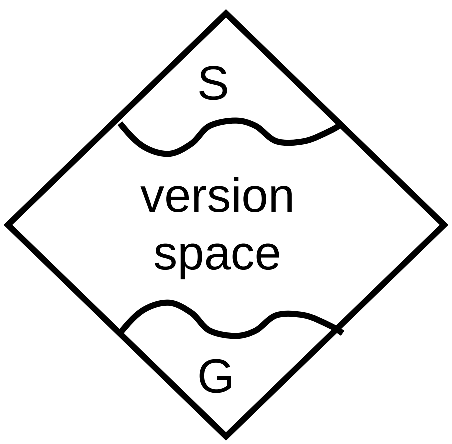
$\langle$*Sunny Warm High Strong Cool Change*$\rangle$ +

- Why does this example remove a hypothesis from G?:
  - $\langle$*? ? ? ? ? Same*$\rangle$

- This hypothesis
  - Cannot be specialized, since would not cover new TE
  - Cannot be generalized, because more general would cover negative TE.
  - Hence must drop hypothesis.

Version Space of the Example





Convergence of algorithm

- Convergence guaranteed if:
  - ***no errors***
  - ***there is h in H describing c***.
- Ambiguity removed from VS when S = G
  - Containing single h
  - When have seen enough TEs
- If have false negative TE, algorithm will remove every h consistent with TE, and hence will remove correct target concept from VS
  - If observe enough TEs will find that S, G boundaries converge to empty VS

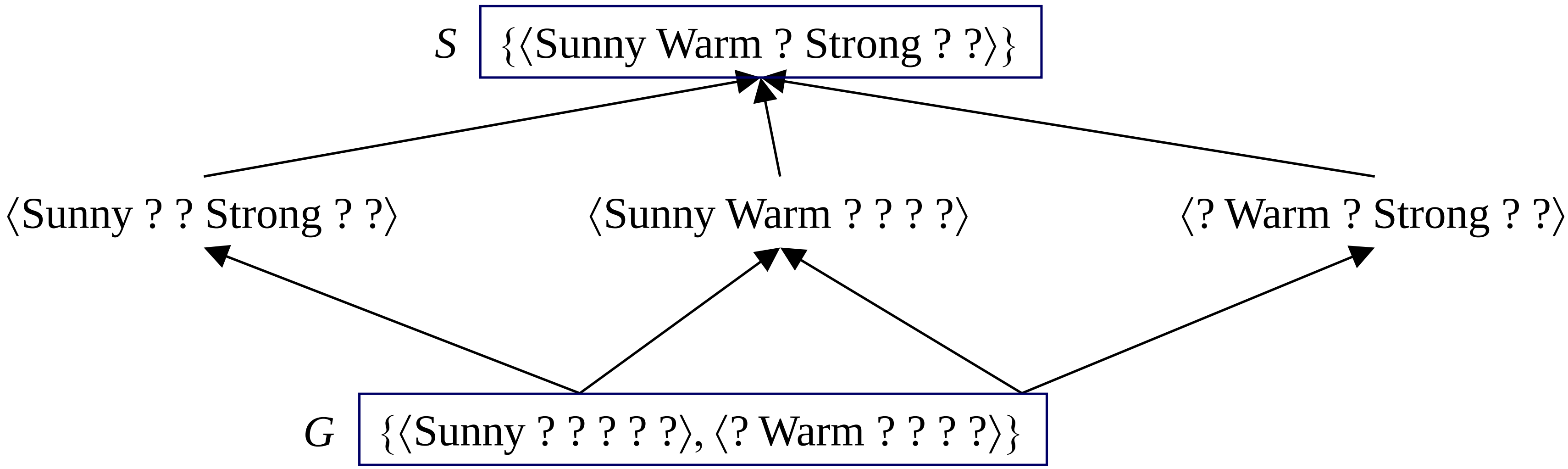
Let us try this

Origin	Manufacturer	Color	Decade	Type	
Japan	Honda	Blue	1980	Economy	+
Japan	Toyota	Green	1970	Sports	-
Japan	Toyota	Blue	1990	Economy	+
USA	Chrysler	Red	1980	Economy	-
Japan	Honda	White	1980	Economy	+

And this

Origin	Manufacturer	Color	Decade	Type	
Japan	Honda	Blue	1980	Economy	+
Japan	Toyota	Green	1970	Sports	-
Japan	Toyota	Blue	1990	Economy	+
USA	Chrysler	Red	1980	Economy	-
Japan	Honda	White	1980	Economy	+
Japan	Toyota	Green	1980	Economy	+
Japan	Honda	Red	1990	Economy	-

Which Next Training Example?

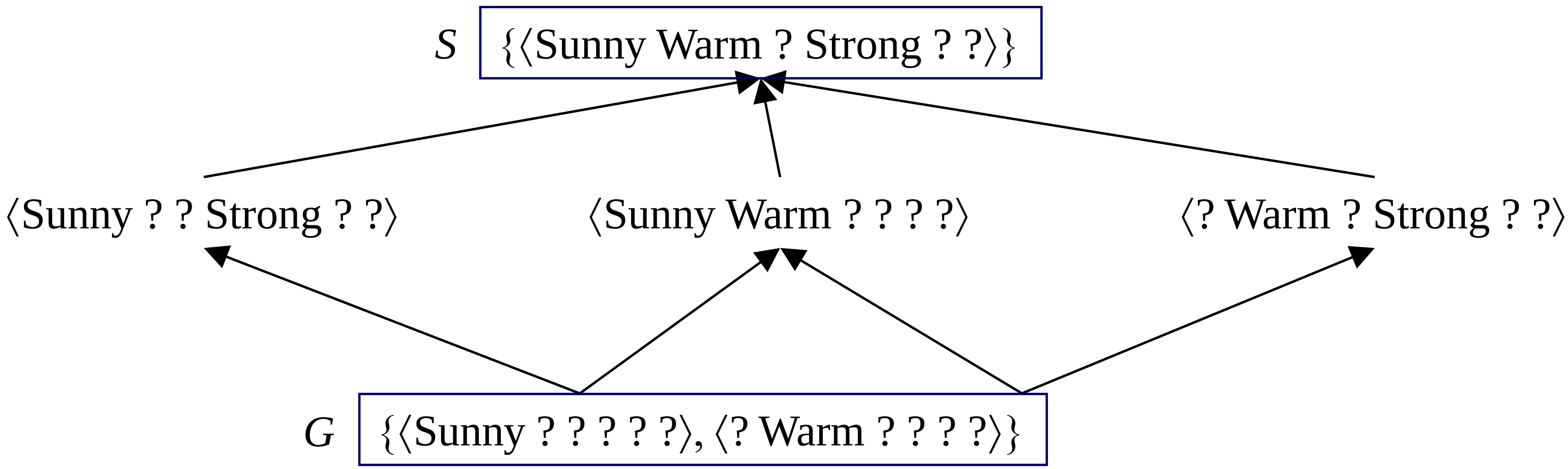


Assume learner can choose the next TE

- Should choose d such that
  - Reduces maximally the number of hypotheses in VS
  - Best TE: satisfies precisely 50% hypotheses;
    - Can’t always be done
  - Example:
    - ⟨Sunny Warm Normal Weak Warm Same⟩ ?
    - If pos, generalizes S
    - If neg, specializes G

Order of examples matters for intermediate sizes of S,G; not for the final S, G

Classifying new cases using VS



- Use *voting procedure* on following examples:
  - ⟨Sunny Warm Normal Strong Cool Change⟩
  - ⟨Rainy Cool Normal Weak Warm Same⟩
  - ⟨Sunny Warm Normal Weak Warm Same⟩
  - ⟨Sunny Cold Normal Strong Warm Same⟩

Effect of incomplete hypothesis space

- Preceding algorithms work if target function is in H
  - Will generally not work if target function *not* in H
- Consider following examples which represent target function

  "sky = sunny or sky = cloudy":
  - ⟨Sunny Warm Normal Strong Cool Change⟩ Y
  - ⟨Cloudy Warm Normal Strong Cool Change⟩ Y
  - ⟨⟨Rainy Warm Normal Strong Cool Change⟩ N
- If apply CE algorithm as before, end up with empty VS
  - After first two TEs, S= ⟨? Warm Normal Strong Cool Change⟩
  - New hypothesis is overly general
    - it covers the third negative TE!
- Our H does not include the appropriate c

Need more expressive hypotheses

Incomplete hypothesis space

- If c not in H, then consider generalizing representation of H to contain c
  - For example, add disjunctions or negations to representation of hypotheses in H
- One way to avoid problem is to allow **all** possible representations of h's
  - Equivalent to allowing all possible subsets of instances as defining the concept of EnjoySport
    - Recall that there are 96 instances in EnjoySport; hence there are 2^{96} possible hypotheses in full space H
    - Can do this by using full propositional calculus with AND, OR, NOT
    - Hence H defined only by conjunctions of attributes is biased (containing only 973 h's)

Unbiased Learners and Inductive Bias

- BUT if have no limits on representation of hypotheses

(i.e., full logical representation: *and*, *or*, *not*), can only learn examples…no generalization possible!

  - Say have 5 TEs {x1, x2, x3, x4, x5}, with x4, x5 negative TEs
- Apply CE algorithm
  - S will be disjunction of positive examples (S={x1 OR x2 OR x3})
  - G will be negation of disjunction of negative examples (G={*not (*x4 or x5)})
  - Need to use all instances to learn the concept!
- Cannot predict usefully:
  - TEs have unanimous vote
  - other h's have 50/50 vote!
    - For every h in H that predicts +, there is another that predicts -

Unbiased Learners and Inductive Bias

- Approach:
  - Place constraints on representation of hypotheses
    - Example of limiting connectives to conjunctions
    - Allows learning of generalized hypotheses
    - Introduces bias that depends on hypothesis representation
- Need formal definition of inductive bias of learning algorithm

Inductive Syst and Equiv Deductive Syst

- Inductive bias made explicit in *equivalent deductive system*
  - *Logically represented system that produces same outputs (classification) from inputs (TEs, instance x, bias B) as CE procedure*
- Inductive bias (IB) of learning algorithm L is any minimal set of assertions B such that for any target concept c and training examples D, we can logically infer value c(x) of any instance x from B, D, and x
  - E.g., for rote learner, B = {}, and there is no IB
- Difficult to apply in many cases, but a useful guide

Inductive Bias and specific learning algs

- Rote learners:

  no IB

- Version space candidate elimination algorithm:

  c can be represented in H

- Find-S: c can be represented in H;

  all instances that are not positive are negative

Computational Complexity of VS

- The *S* set for conjunctive feature vectors and tree-structured attributes is linear in the number of features and the number of training examples.
- The *G* set for conjunctive feature vectors and tree-structured attributes can be exponential in the number of training examples.
- In more expressive languages, both *S* and *G* can grow exponentially.
- The order in which examples are processed can significantly affect computational complexity.

Exponential size of G

- n Boolean attributes
- 1 positive example: (T, T, .., T)
- n/2 negative examples:
  - (F,F,T,..T)
  - (T,T,F,F,T..T)
  - (T,T,T,T,F,F,T..T)
  - ..
  - (T,..T,F,F)
- Every hypothesis in G needs to choose from n/2 2-element sets.
  - Number of hypotheses = $2^{n/2}$

Summary

- Concept learning as search through *H*
- General-to-specific ordering over *H*
- Version space candidate elimination algorithm
- *S* and *G* boundaries characterize learner's uncertainty
- Learner can generate useful queries
- Inductive leaps possible only if learner is biased!
- Inductive learners can be modeled as equiv deductive systems
- Biggest problem is inability to handle data with errors
  - Overcome with procedures for learning decision trees