Machine Learning

Concept Learning

Prof. Dr. Martin Riedmiller
AG Maschinelles Lernen und Natürlichsprachliche Systeme
Institut für Informatik
Technische Fakultät
Albert-Ludwigs-Universität Freiburg

Martin.Riedmiller@uos.de

Acknowledgment Slides were adapted from slides provided by
Tom Mitchell, Carnegie-Mellon-University
and Peter Geibel, University of Osnabrück

Overview of Today's Lecture: Concept Learning

read T. Mitchell, Machine Learning, chapter 2

- Learning from examples
- General-to-specific ordering over hypotheses
- Version spaces and candidate elimination algorithm
- Picking new examples
- The need for inductive bias

Note: simple approach assuming no noise, illustrates key concepts

Introduction

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

Training Examples for *EnjoySport*

- Examples: "Days at which my friend Aldo enjoys his favorite water sport"
- Result: classifier for days = description of Aldo's behavior

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

What is the general concept?

Representing Hypotheses

- Many possible representations
- in the following: h is conjunction of constraints on attributes
- Each constraint can be
  - a specfic value (e.g., $Water = Warm$)
  - don't care (e.g., "$Water =?$")
  - no value allowed (e.g.,"Water=$\emptyset$")
- For example,

Sky	AirTemp	Humid	Wind	Water	Forecst
$\langle Sunny$	$?$	$?$	$Strong$	$?$	$Same\rangle$

- We write $h(x) = 1$ for a day x, if x satisfies the description
- Note that much more expressive languages exists

Most General/Most Specific Hypothesis

- Most general hypothesis: $(?, ?, ?, ?, ?)$
- Most specific hypothesis: $(\emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$

Prototypical Concept Learning Task

- Given:
  - Instances X: Possible days, each described by the attributes

    Sky, AirTemp, Humidity, Wind, Water, Forecast

  - Target concept c: $EnjoySport : X \rightarrow \{0, 1\}$
  - Hypotheses H: Conjunctions of literals. E.g.

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

  - Training examples D: Positive and negative examples of the target function

$$\langle x_1, c(x_1) \rangle, \ldots \langle x_m, c(x_m) \rangle$$

- Determine: A hypothesis h in H with $h(x) = c(x)$ 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.

- I.e. the training set needs to 'represent' the whole domain (which may be infinite)
- Even if we have a 'good' training set, we can still construct bad hypotheses!

Concept Learning as Search

- The hypothesis representation language defines a potentially large space
- Learning can be viewed as a task of searching this space
- Assume, that Sky has three possible values, and each of the remaining attributes has 2 possible values
- $\rightarrow$ Instance space constains 96 distinct examples
- Hypothesis space contains 5120 syntactically different hypothesis
- What about the semantically different ones?
- Different learning algorithms search this space in different ways!

General-to-Specific Ordering of Hyothesis

- Many algorithms rely on ordering of hypothesis
- Consider

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

  and

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

General-to-Specific Ordering of Hyothesis

- Many algorithms rely on ordering of hypothesis
- Consider
$$h_1 = (Sunny, ?, ?, Strong, ?, ?)$$
and
$$h_2 = (Sunny, ?, ?, ?, ?, ?)$$
- h_2 is more general than h_1!
- How to formalize this?

General-to-Specific Ordering of Hyothesis

- Many algorithms rely on ordering of hypothesis
- Consider

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

  and

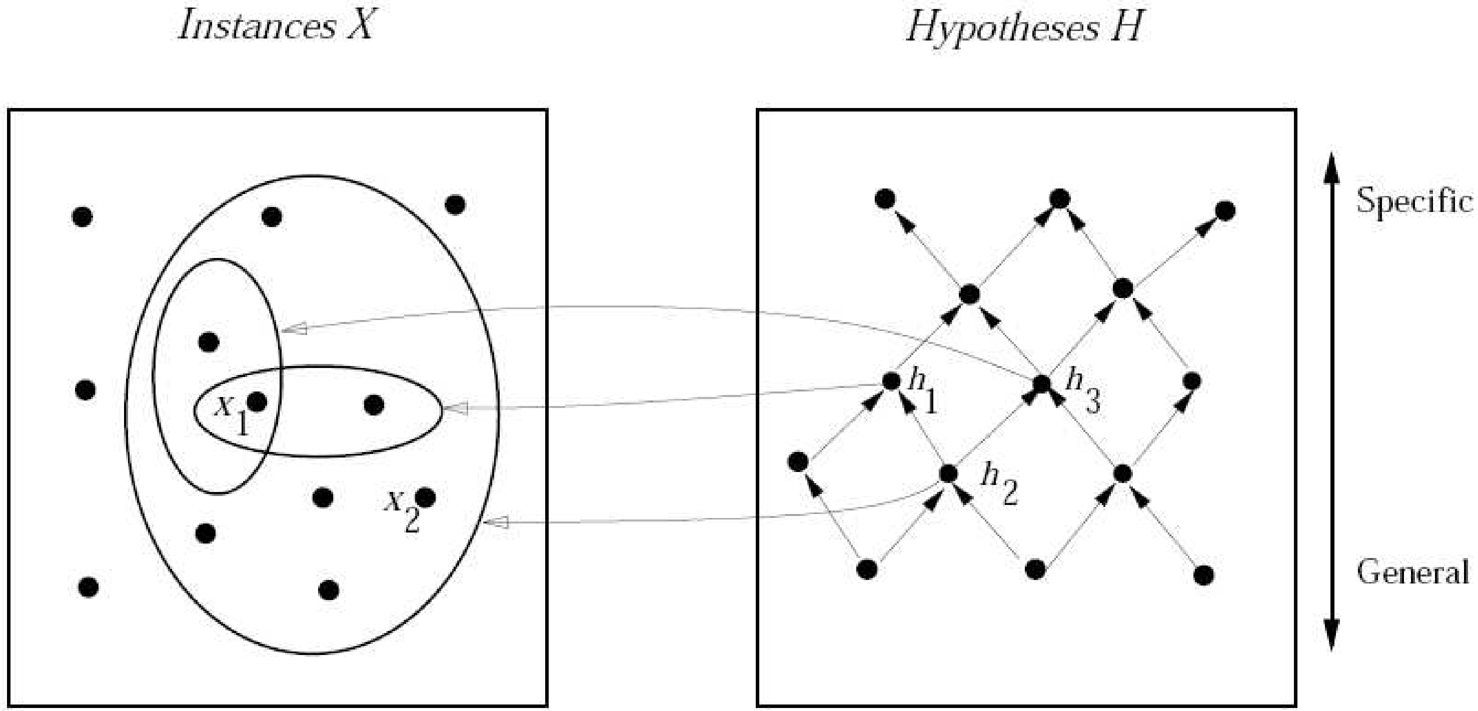
$$h_2 = (Sunny, ?, ?, ?, ?, ?)$$

- h_2 is more general than h_1!
- How to formalize this?

Definition h_2 is more general than h_1, if $h_1(x) = 1$ implies $h_2(x) = 1$. In symbols

$$h_2 \geq_g h_1$$

Instance, Hypotheses, and 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, ?>

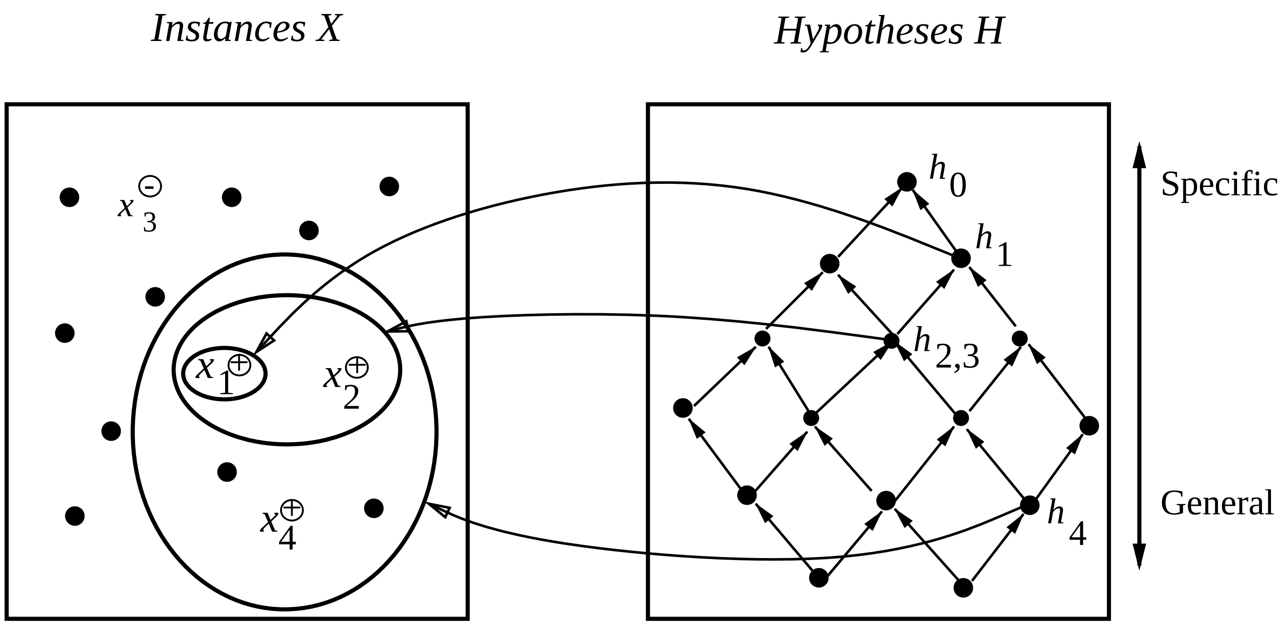
General-to-Specific Ordering of Hyothesis

- $\geq_g$ does not depend on the concept to be learned
- It defines a partial order over the set of hypotheses
- *strictly-more-general than*: $>_g$
- more-specific-than $\leq_g$
- Basis for the learning algorithms presented in the following!
- Find-S:
  - Start with most specific hypothesis $(\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)$
  - Generalize if positive example is not covered!

Find-S Algorithm

- Initialize h to the most specific hypothesis in H
- 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
- Output hypothesis h

Hypothesis Space Search by `Find-S`



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 Role of Negative Examples

- Basically, the negative examples are simply ignored!
- If we assume that the true target concept c is in H (and the training data contains no errors) then negative examples can be safely ignored.

The Role of Negative Examples

- Basically, the negative examples are simply ignored!
- If we assume that the true target concept c is in H (and the training data contains no errors) then negative examples can be safely ignored.

Reason:
The current hypothesis h is the most specific hypothesis consistent with the observed positive examples.
c is in H and c is consistent with the positive examples, therefore $c \geq_g h$ (c is more general or equal to h)
c is the true target concept and therefore will never contain any negative example. Therefore h will not contain any negative example (by the definition of 'more general than')
Therefore, h will never need a revision due to a negative example

Thoughts about `Find-S`

- Assume a consistent and unknown h that has generated the training set
- $\rightarrow$ Algorithm can't tell whether it has learned the right concept because it picks one hypothesis out of many possible ones
- Can't tell when training data inconsistent because it ignores the negative examples: doesn't account for noise
- Picks a maximally specific h $\rightarrow$ is this reasonable?
- Depending on H, there might be several correct hypothesis!
- $\rightarrow$ Version spaces:
  - Characterize the set of all consistent hypotheses
  - ... without enumerating all of them

Version Spaces

Definition 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 $\langle x, c(x) \rangle$ in D.

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

Definition 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.

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

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 $VersionSpace$

Central idea: The Version Space can be represented by the most general and the most specific hypothesis.

Example Version Space

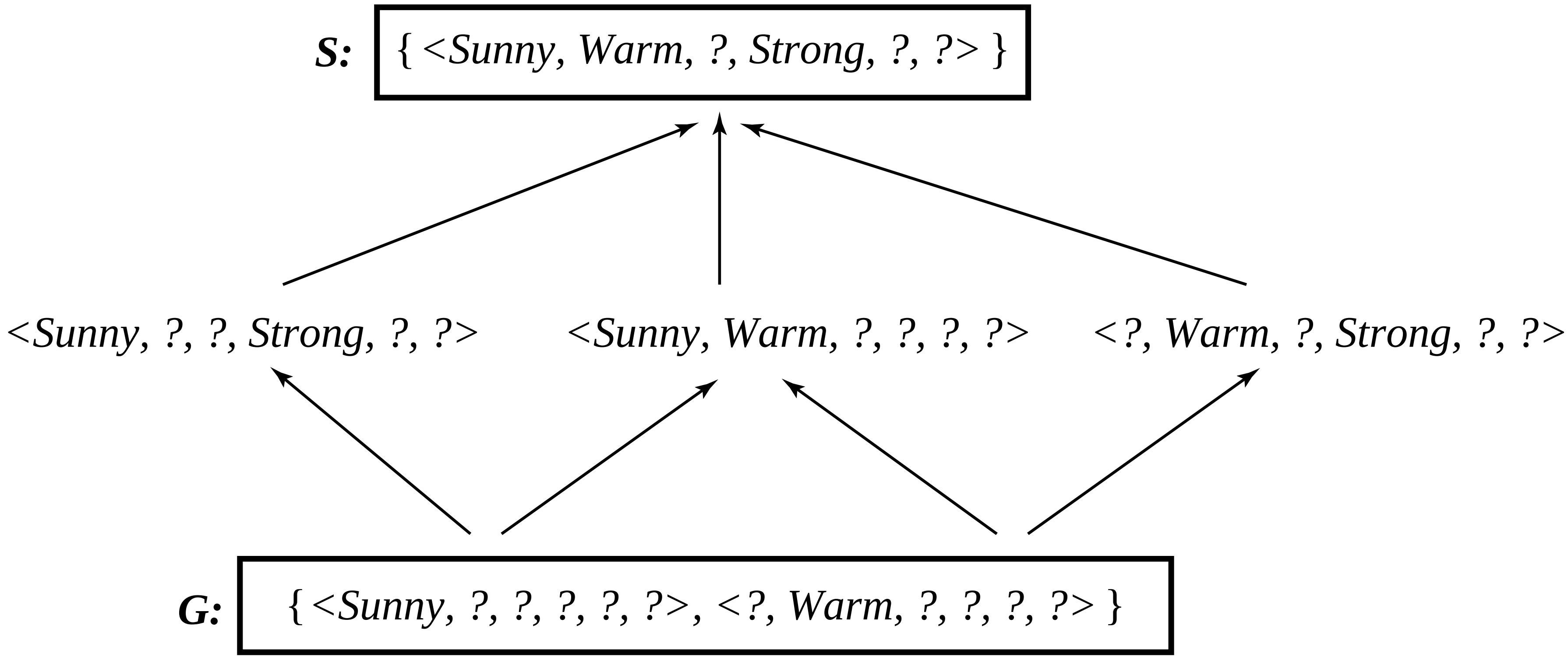
Sky	Temp	Humid	Wind	Water	Forecst	EnjoySpt
Sunny	Warm	Normal	Strong	Warm	Same	Yes
Sunny	Warm	High	Strong	Warm	Same	Yes
Rainy	Cold	High	Strong	Warm	Change	No
Sunny	Warm	High	Strong	Cool	Change	Yes

S: { *<Sunny, Warm, ?, Strong, ?, ?>* }

<Sunny, ?, ?, Strong, ?, ?> *<Sunny, Warm, ?, ?, ?, ?>* *<?, Warm, ?, Strong, ?, ?>*

G: { *<Sunny, ?, ?, ?, ?, ?>, <?, Warm, ?, ?, ?, ?>* }

Example Version Space



Representing Version Spaces

1. The General boundary, G, of version space $VS_{H,D}$ is the set of its maximally general members that are consistent with the given training set

2. The Specific boundary, S, of version space $VS_{H,D}$ is the set of its maximally specific members that are consistent with the given training set

3. Every member of the version space lies between these boundaries

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

where $x \geq y$ means x is more general or equal to y
proof: see Mitchell, Machine Learning, ch. 2

Candidate Elimination Algorithm – Pos. Examples

Input: training set
Output:

- $G =$ maximally general hypotheses in H
- $S =$ maximally specific hypotheses in H

Algorithm:
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
      (a) h is consistent with d, and
      (b) some member of G is more general than h
    * Remove from S any hypothesis that is more general than another hypothesis in S

Candidate Elimination Algorithm – Neg. Examples

- 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
      (a) h is consistent with d, and
      (b) some member of S is more specific than h
    * Remove from G any hypothesis that is less general than another hypothesis in G

Note that the algorithm contains operations such that computing 'minimal specialisations and generalisations' of given hypothesis or identifying nonminimal and nonmaximal hypothesis. The implementation will - of course - depend on the specific representation of hypothesis. The algorithm can be applied to any learning task and hypothesis space for which these operations are well defined.

Example Trace

S_0: {<Ø, Ø, Ø, Ø, Ø, Ø>}

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

Example Trace

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

↓

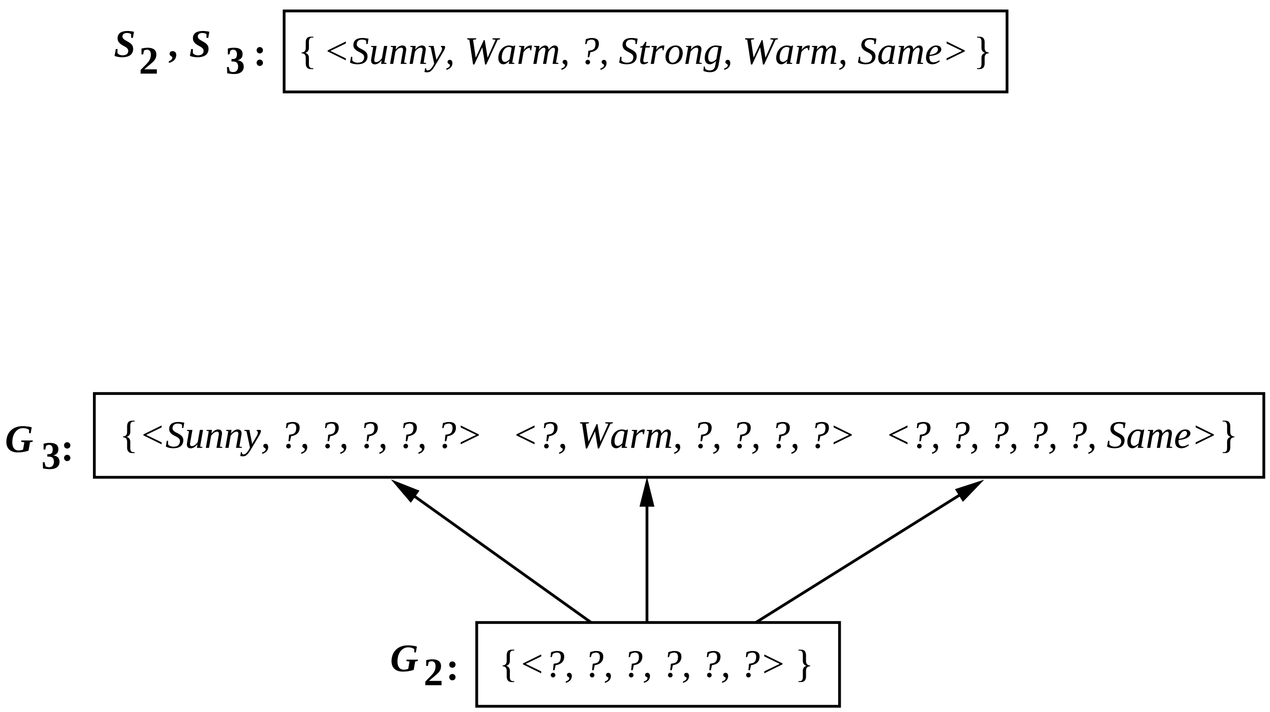
S_2: {<Sunny, Warm, ?, Strong, Warm, Same>}

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

Example Trace



Training Example:

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

Example Trace

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

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

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

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

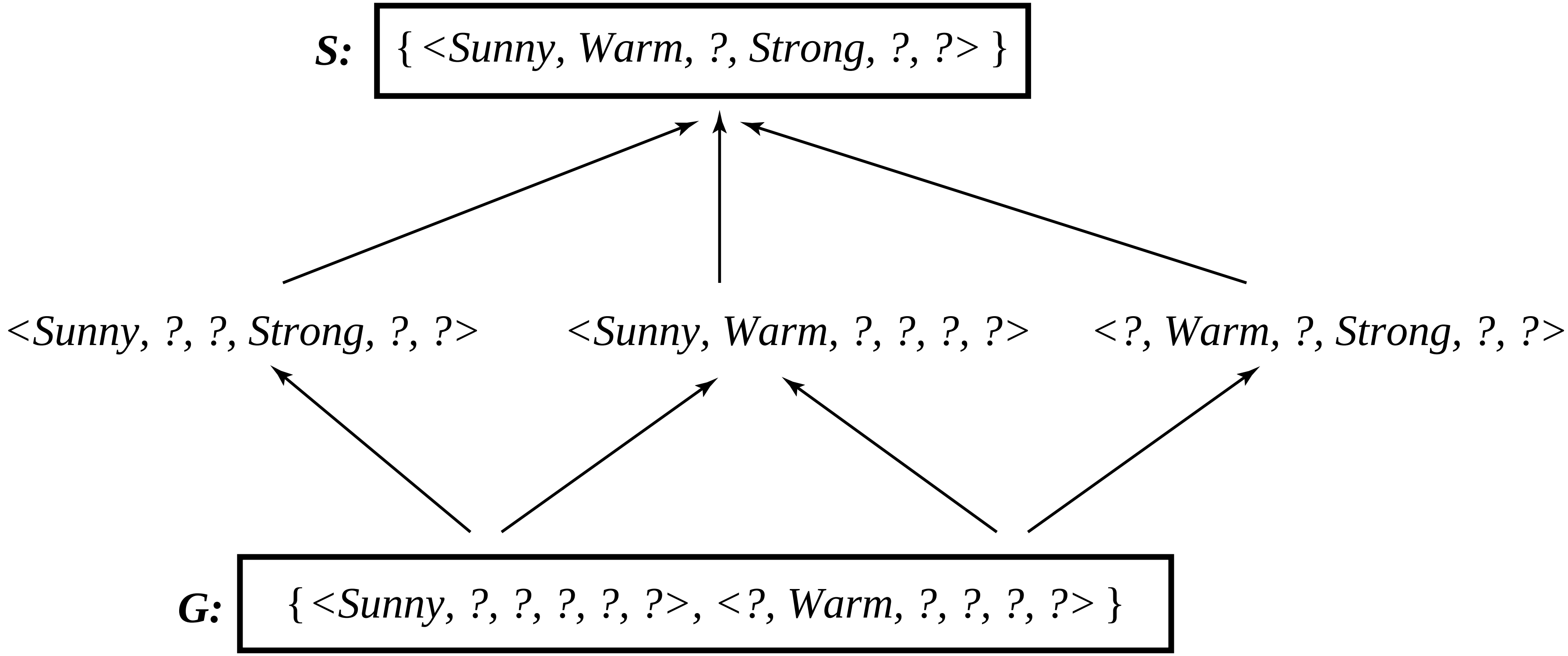
Training Example:

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

Properties of the two Sets

- S can be seen as the summary of the positive examples
- Any hypothesis more general than S covers all positive examples
- More specific hypothesis fail to cover at least one pos. ex.
- G can be seen as the summary of the negative examples
- Any hypothesis more specific than G covers no previous negative example
- More general hypothesis cover at least one negative example

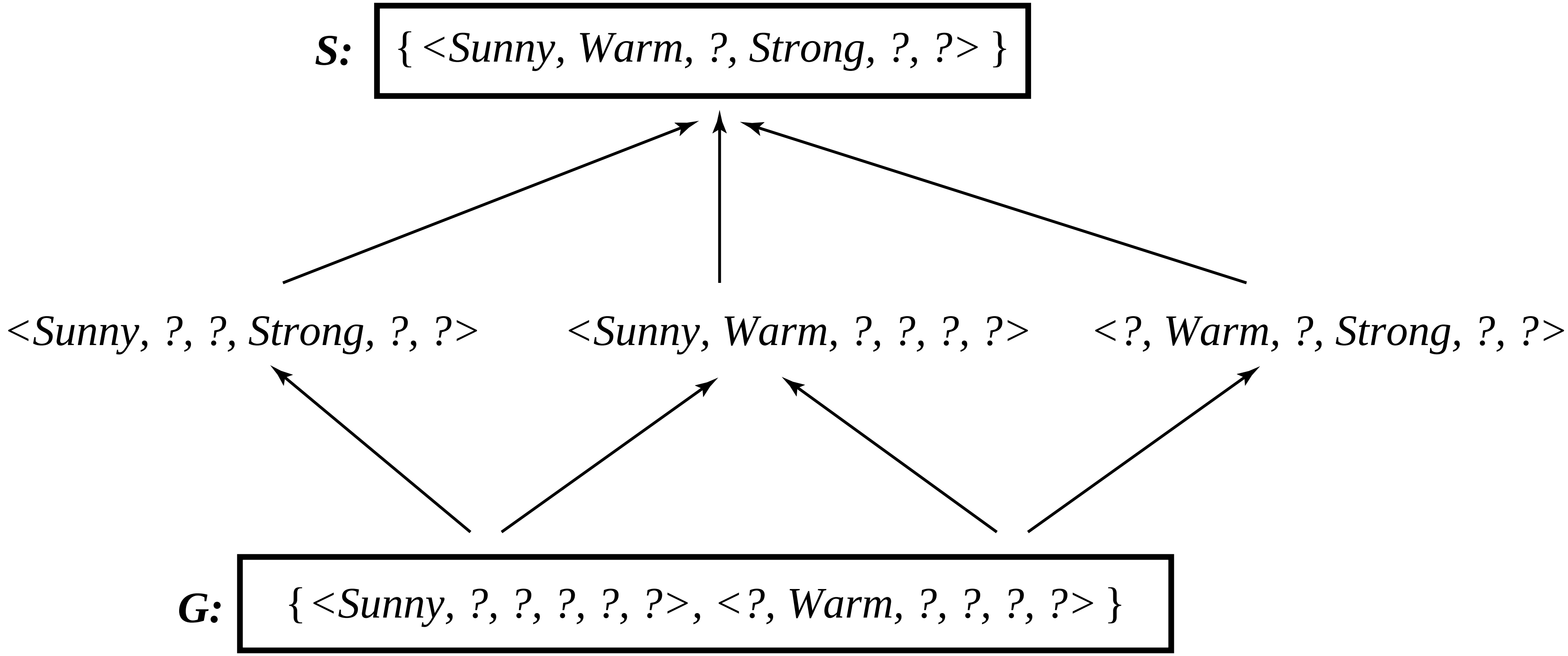
Resulting Version Space



Properties

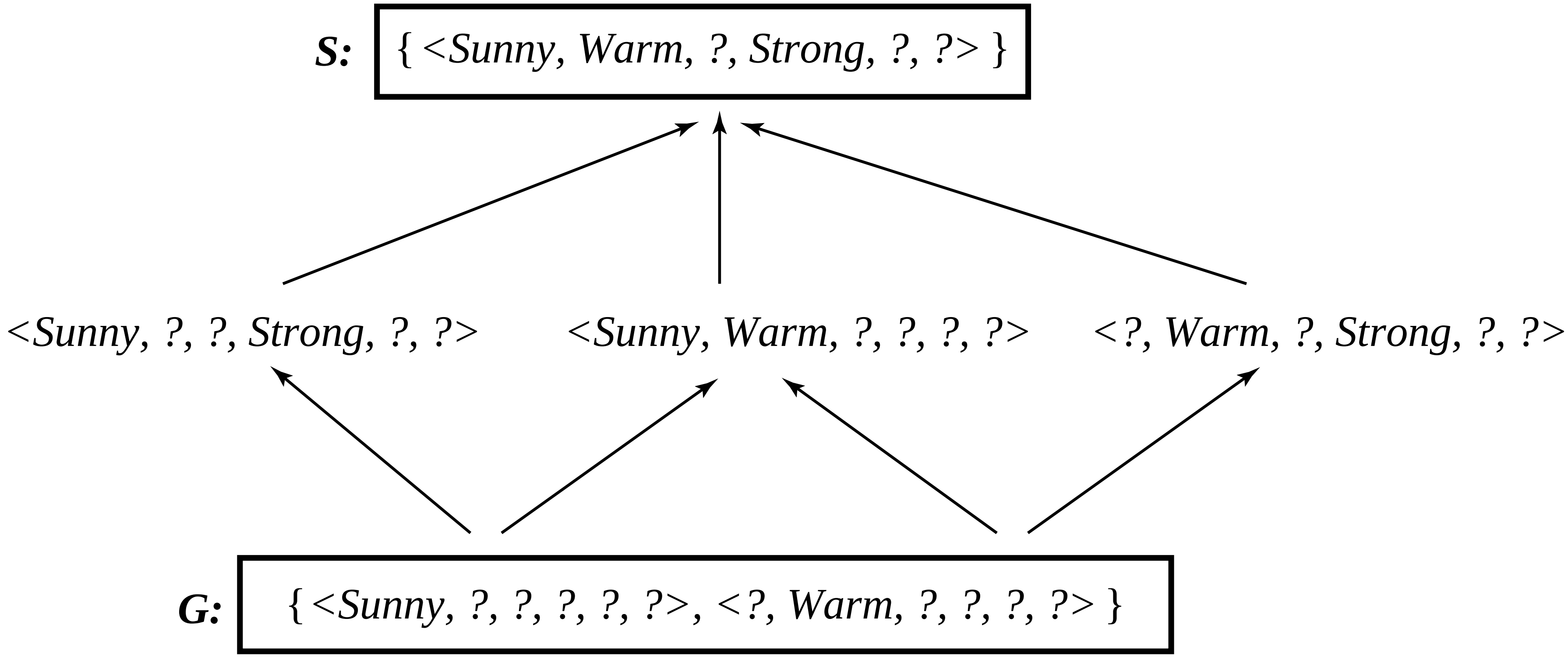
- If there is a consistent hypothesis then the algorithm will converge to $S = G = \{h\}$ when enough examples are provided
- False examples may cause the removal of the correct h
- If the examples are inconsistent, S and G become empty
- This can also happen, when the concept to be learned is not in H

What Next Training Example?



- If the algorithm is allowed to select the next example, which is best?

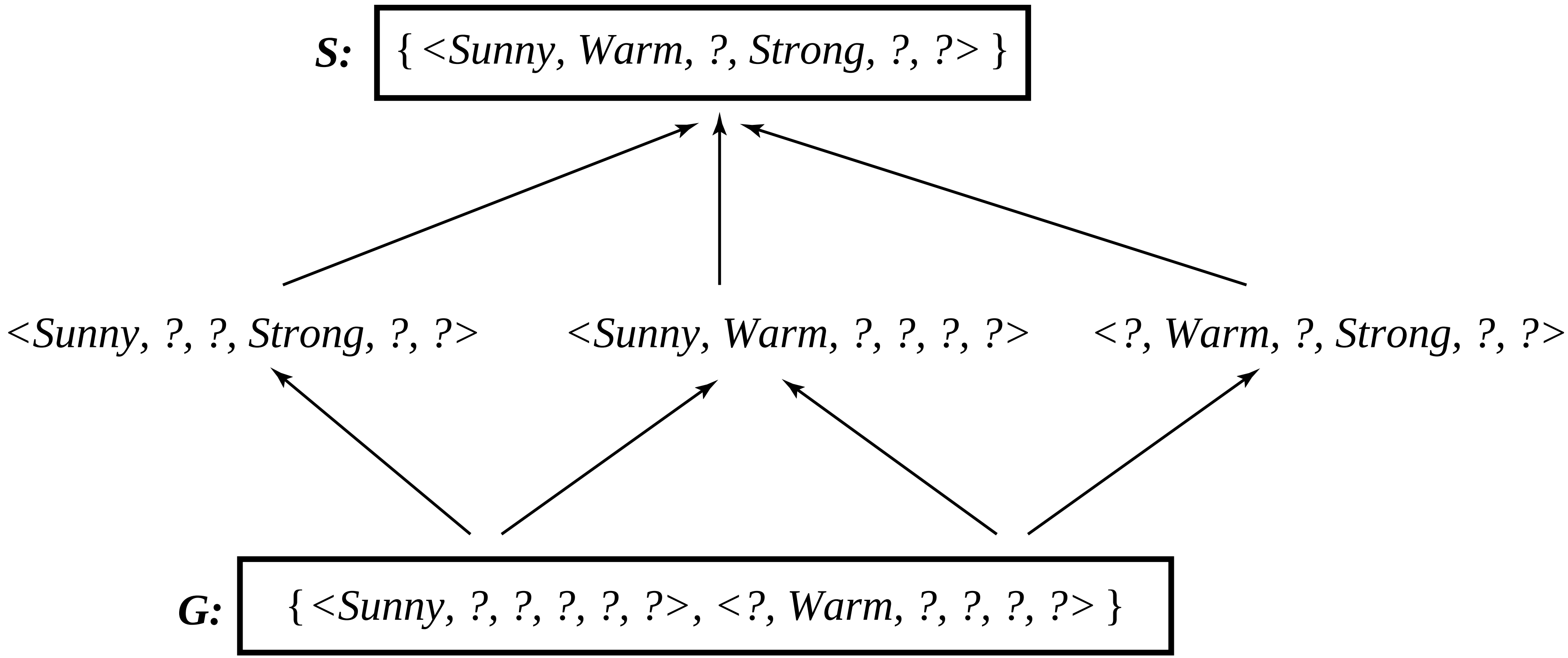
What Next Training Example?



- If the algorithm is allowed to select the next example, which is best?

  ideally, choose an instance that is classified positive by half and negative by the other half of the hypothesis in VS. In either case (positive or negative example), this will eliminate half of the hypothesis. E.g:
  $\langle Sunny\ Warm\ Normal\ Light\ Warm\ Same \rangle$

How Should These Be Classified?



- $\langle Sunny\ Warm\ Normal\ Strong\ Cool\ Change \rangle$
- $\langle Rainy\ Cool\ Normal\ Light\ Warm\ Same \rangle$
- $\langle Sunny\ Warm\ Normal\ Light\ Warm\ Same \rangle$

Classification

- Classify a new example as positive or negative, if all hypotheses in the version space agree in their classification
- Otherwise:
  - Rejection or
  - Majority vote

Inductive Bias

- What if target concept not contained in hypothesis space?
- Should we include every possible hypothesis?
- How does this influence the generalisation ability?

Inductive Leap

- Induction vs. deduction (=theorem proving)
- Induction provides us with new knowledge!
- What Justifies this "Inductive Leap?"

$$+ \ \langle Sunny\ Warm\ Normal\ Strong\ Cool\ Change \rangle$$
$$+ \ \langle Sunny\ Warm\ Normal\ Light\ Warm\ Same \rangle$$

$$S: \ \langle Sunny\ Warm\ Normal\ ?\ ?\ ? \rangle$$

Question: Why believe we can classify the unseen

$$\langle Sunny\ Warm\ Normal\ Strong\ Warm\ Same \rangle ?$$

An UNBiased Learner

- Idea: Choose H that expresses every teachable concept
- I.e., H corresponds to the power set of $X \rightarrow |H| = 2^{|X|}$
- $\rightarrow$ much bigger than before, where $|H| = 937$
- Consider $H' =$ disjunctions, conjunctions, negations over previous H. E.g.,

$$\langle Sunny\ Warm\ Normal\ ?\ ?\ ?\rangle \ \vee\ \neg\langle ?\ ?\ ?\ ?\ ?\ Change\rangle$$

- It holds $h(x) = 1$ if x satisfies the logical expression.
- What are S, G in this case? (next slide)

The Futility of Bias-Free Learning

Example: x_1, x_2, x_3 positive, x_4, x_5 negative.
Then: $G = \{\neg(x_4 \vee x_5)\}, S = \{(x_1 \vee x_2 \vee x_3)\}$

- S $= \{$s$\}$, with $s =$ disjunction of positive examples
- G $= \{$g$\}$, with $g =$ Negated disjunction of negative examples
- $\rightarrow$ Only training examples will be unambiguously classified
- Is majority vote a solution? No:
  Unknown pattern will be classified positive by exactly half of the hypothesis and negative by the other half.
  Reason: If H is the power set of X, and x is some unobserved instance, then for any h in the version space that covers x there is another hypothesis h' in the power set that is identical to h except for the classification of x. If h is in the version space, then h' will be as well, because it agrees with h on all observed training examples.

A learner that makes no a priori assumptions regarding the identity of the target concept has no rational basis for classifying any unseen instances.

- Inductive bias = underyling assumptions
- These assumption explain the result of learning
- The inductive bias explains the inductive leap!

Inductive Bias

- Concept learning algorithm L
- Instances X, target concept c
- Training examples $D_c = \{\langle x, c(x) \rangle\}$
- Let $L(x_i, D_c)$ denote the classification assigned to the instance x_i by L after training on data D_c, e.g. $EnjoySport = yes$

Definition 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

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

where $A \vdash B$ means A logically entails B.

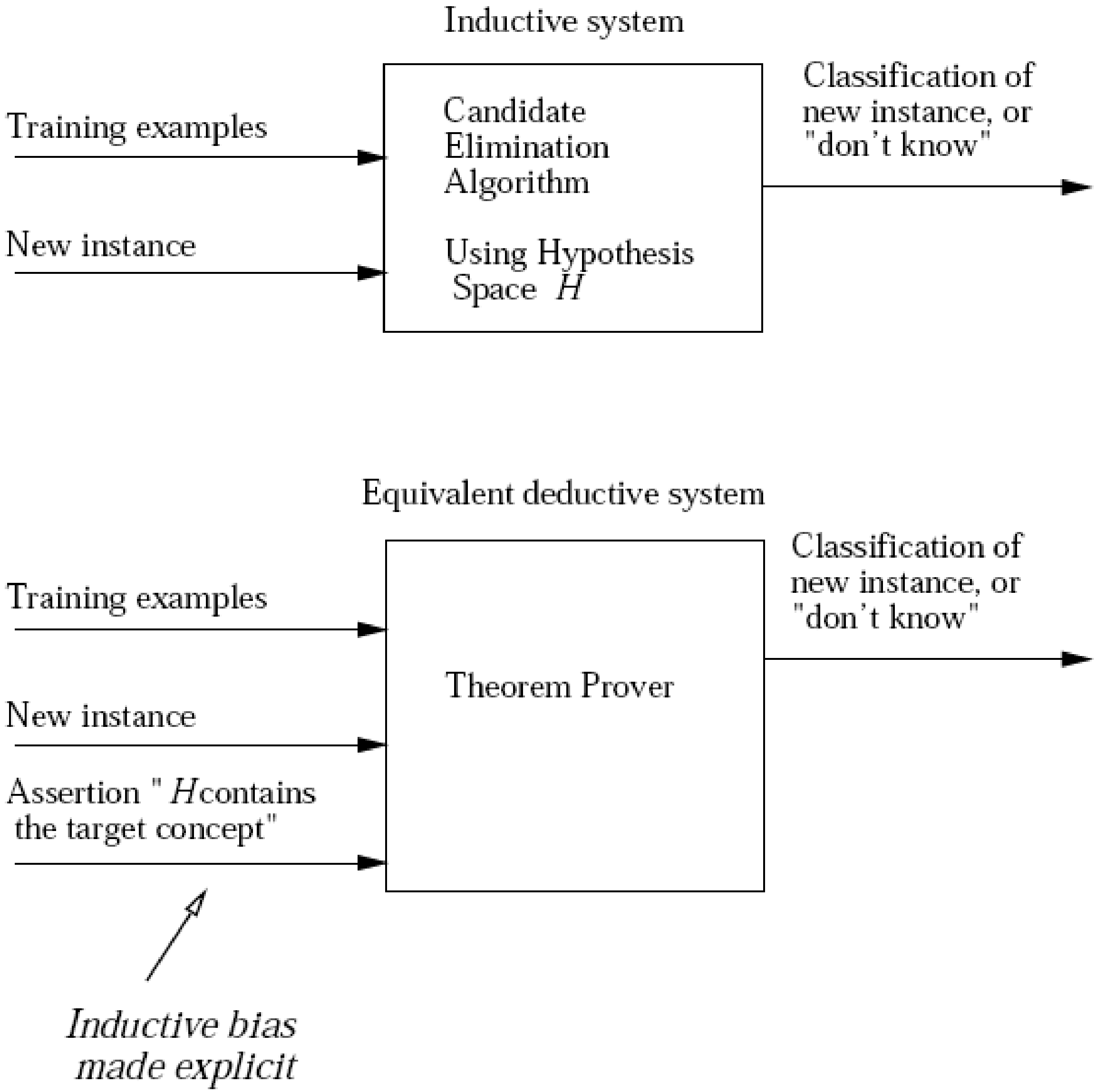
Inductive Bias for Candidate Elimination

- Assume instance x_i and training set D_c
- The algorithm computes the version space
- x_i is classified by unanimous voting (using the instances in the version space); otherwise systems answers 'don't know'
- $\rightarrow$ this way $L(x_i, D_c)$ is computed

Inductive Bias for Candidate Elimination

- Assume instance x_i and training set D_c
- The algorithm computes the version space
- x_i is classified by unanimous voting (using the instances in the version space); otherwise systems answers 'don't know'
- $\rightarrow$ this way $L(x_i, D_c)$ is computed
- Now assume that the underlying concept c is in H
- This means that c is a member of its version space
- $EnjoySport = k$ implies that all members of VS, including c vote for class k
- Because unanimous voting is required, $k = c(x_i)$
- This is also the output of the algorithm $L(x_i, D_c)$
- $\rightarrow$ The inductive bias of the Candidate Elimination Algorithm is: c is in H

Inductive Systems and Equivalent Deductive Systems



Three Learners with Different Biases

- Note that the inductive bias is often only implicitly encoded in the learning algorithm
- In the general case, it's much more difficult to determine the inductive bias
- Often properties of the learning algorithm have to be included, e.g. it's search strategy
- What is inductive bias of
  - Rote learner: Store examples, Classify x iff it matches previously observed example.

Three Learners with Different Biases

- Note that the inductive bias is often only implicitly encoded in the learning algorithm
- In the general case, it's much more difficult to determine the inductive bias
- Often properties of the learning algorithm have to be included, e.g. it's search strategy
- What is inductive bias of
  - Rote learner: Store examples, Classify x iff it matches previously observed example.
    No inductive bias ($\rightarrow$ no generalisation!)
  - Candidate elimination algorithm

Three Learners with Different Biases

- Note that the inductive bias is often only implicitly encoded in the learning algorithm
- In the general case, it's much more difficult to determine the inductive bias
- Often properties of the learning algorithm have to be included, e.g. it's search strategy
- What is inductive bias of
  - Rote learner: Store examples, Classify x iff it matches previously observed example.
    No inductive bias ($\rightarrow$ no generalisation!)
  - Candidate elimination algorithm
    c is in H (see above)
  - Find-S c is in H and that all instances are negative examples unless the opposite is entailed by its training data

A good generalisation capability of course depends on the appropriate choice of the inductive bias!

Summary Points

- 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 modelled by equivalent deductive systems