2 Concept Learning

SS 19 – Multimedia II:

Machine Learning and Computer Vision

Multimedia Computing & Computer Vision, Universität Augsburg
Rainer.Lienhart@informatik.uni-augsburg.de
www.multimedia-computing.{de,org}

Outline

Concept learning = acquire the definition of a general category given a sample of positive and negative training examples of the category

- 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

What Do You Know?

- Induction ←→ Deduction: What is the difference?
  - Examples: *Law of Free Fall* in Newtonian Mechanics

    $d = \frac{1}{2} * g * t^2$

    $v = g * t$ $\qquad\qquad g \approx 9.81\ [m/s^2]$

    $a = g$

- Our Approach to Learning:
  - Search a very large hypotheses space to determine the hypothesis that best fits the observed data **and** any prior knowledge held by the learner

Training Examples for EnjoySport

Sky	Temp	Humid	Wind	Water	Forecast	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

Goal: What is the general concept?

Attribute:

Sunny, Rainy, Cloudy	Warm, Cold	Normal, High	Strong, Light	Warm, Cool	Same, Change

Prototypical Concept Learning Task

- Given:
  - Instances *X*: Possible days, each described by the attributes *Sky, AirTemp, Humidity, Wind, Water, Forecast*
    - Represented as 6-tuple such as $x_1 = \langle Sunny, Warm, Normal, Strong, Warm, Same \rangle$
  - Target function *c(x)*
  - Learned function *h(x)*: $EnjoySport: X \rightarrow \{0,1\}$
  - Hypotheses space *H*: what is our set of hypotheses $h \in H$?
  - Training examples *D*: Positive and negative examples of the target function

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

In our case: $\{\langle x_1, 1\rangle, \langle x_2, 1\rangle, \langle x_3, 0\rangle, \langle x_4, 1\rangle\}$

Representing Hypotheses

Many possible representations

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 allowed (e.g., "*Water* = Ø")

For example,

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

Prototypical Concept Learning Task

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

    Sky, AirTemp, Humidity, Wind, Water, Forecast
  - Target function *c(x)*
  - Learned function *h(x): $EnjoySport\colon X \rightarrow \{0,1\}$*
  - Hypotheses *H*: Conjunctions of literals.
    - E.g. `< ?, Cold, High, ?, ?, ? >`
  - Training examples *D*: Positive and negative examples of the target function

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

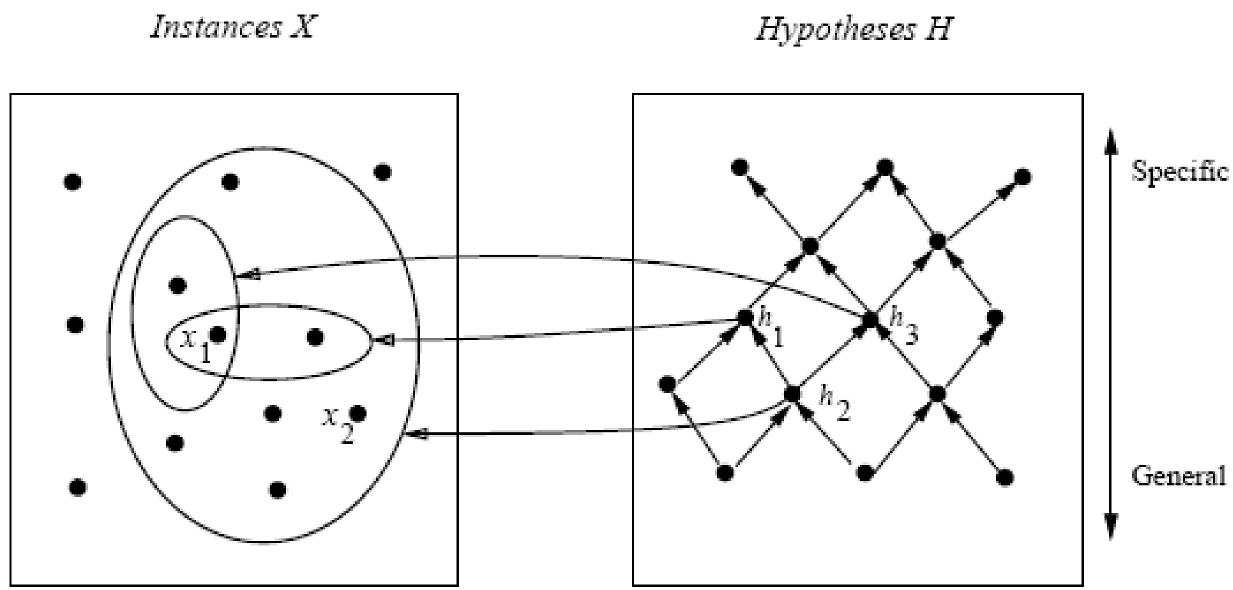
- **Determine**:

  A hypothesis *h* in *H* 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.

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, ?*>

96 distinct instances

5120 syntactically distinct hypotheses
973 semantically distinct hypotheses

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 in h is satisfied by x

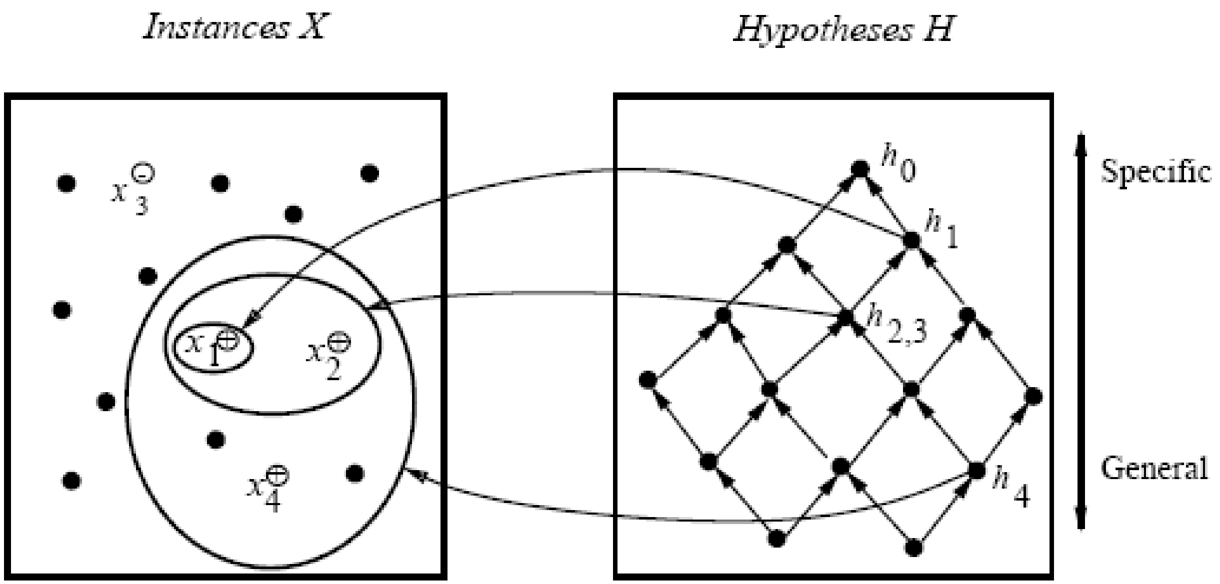
    Then do nothing

    Else

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

Hypothesis Space Search by Find-S

Implicit assumption: Target function c(x) is in H



$x_1 = \langle Sunny \quad Warm \quad Normal \quad Strong \quad Warm \quad Same \rangle, +$

$x_2 = \langle Sunny \quad Warm \quad High \quad Strong \quad Warm \quad Same \rangle, +$

$x_3 = \langle Rainy \quad Cold \quad High \quad Strong \quad Warm \quad Change \rangle, -$

$x_4 = \langle Sunny \quad Warm \quad High \quad Strong \quad Cool \quad Change \rangle, +$

$h_0 = \langle \varnothing \quad \varnothing \quad \varnothing \quad \varnothing \quad \varnothing \quad \varnothing \rangle$

$h_1 = \langle Sunny \quad Warm \quad Normal \quad Strong \quad Warm \quad Same \rangle$

$h_2 = \langle Sunny \quad Warm \quad ? \quad Strong \quad Warm \quad Same \rangle$

$h_3 = \langle Sunny \quad Warm \quad ? \quad Strong \quad Warm \quad Same \rangle$

$h_4 = \langle Sunny \quad Warm \quad ? \quad Strong \quad ? \quad ? \rangle$

Complaints about Find-S

- Can't tell whether it has learned the concept
- Can't tell when training data is inconsistent
- Picks a maximally specific *h* (why?)
- Depending on *H*, there might be several!
- Does not use "negative" training examples!

Consistency

- 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*.

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

- 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 all training examples in *D*.

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

List-Then-Eliminate Algorithm

1. VersionSpace ← 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*

Representing Version Spaces

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

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

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

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

- Every member of the version space lies between these boundaries

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

where $x \geq_g y$ means *x* is more general or equally general than *y.*

Candidate Elimination Algorithm

G ← maximally general hypotheses in *H*

S ← 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
      1. *h* is consistent with *d*, and
      2. 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

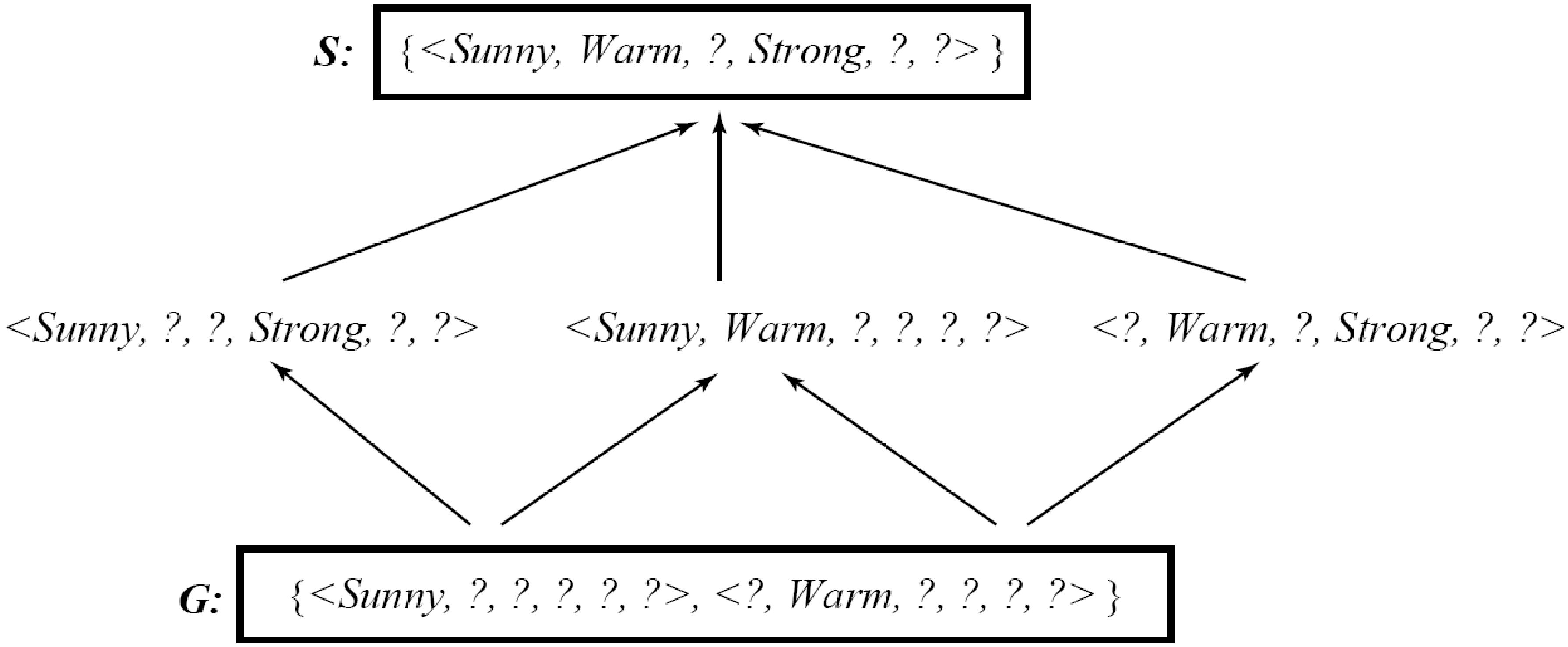
- If *d* is a negative example
  - Remove from *S* any hypothesis inconsistent with *d*
  - For each hypothesis *g* in *G* that is not consistent with *d*
    - Remove *g* from *G*
    - Add to *G* all minimal specializations *h* of *g* such that
      1. *h* is consistent with *d*, and
      2. some member of *S* is more specific than *h*
  - Remove from *G* any hypothesis that is less general than another hypothesis in *G*

Example Trace

S_0: {<∅, ∅, ∅, ∅, ∅, ∅>}

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

Example Version Space



Remarks

VS learned by *Candidate-Elimination* algorithm will converge towards hypothesis that correctly describes the target concept, provided

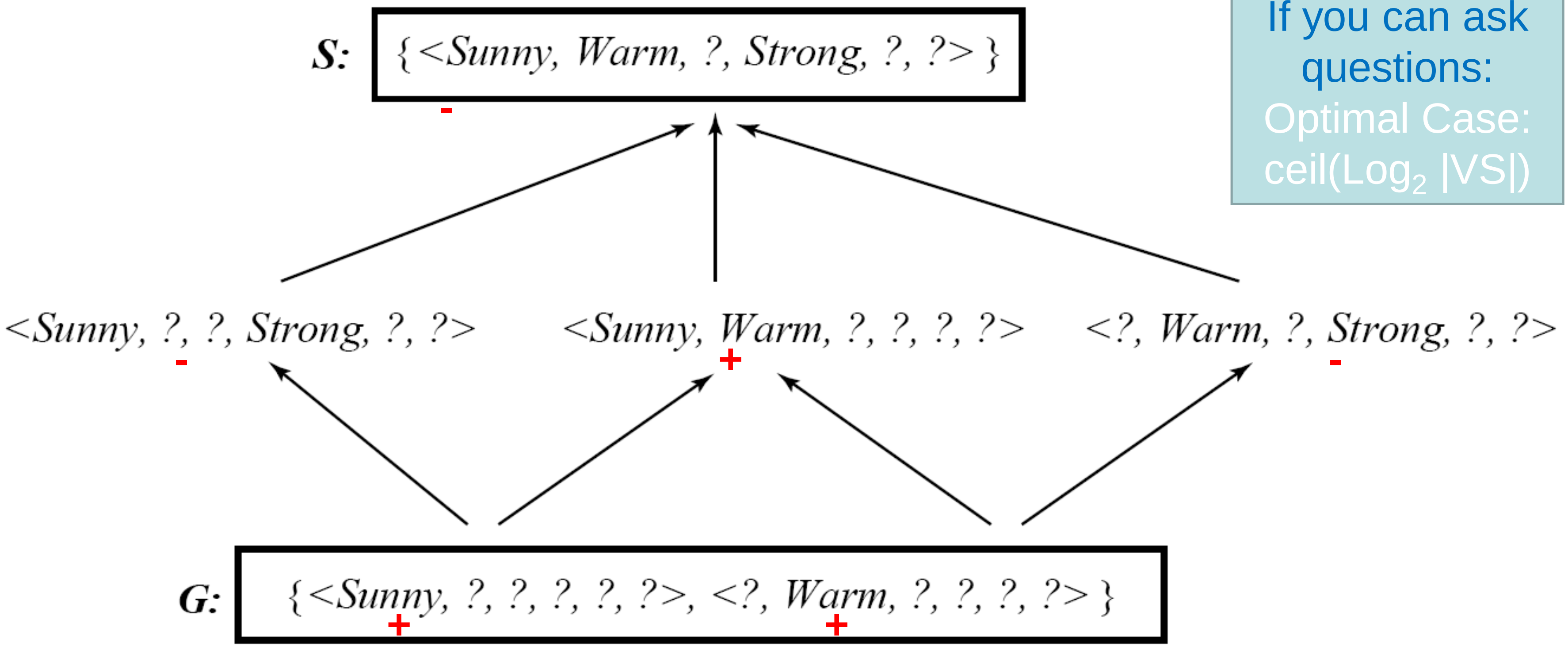
1. No errors in training examples
2. Target concept c is in H

The target concept is exactly learned when S and G boundary sets converge to a single & identical hypothesis.

What Next Training Example?

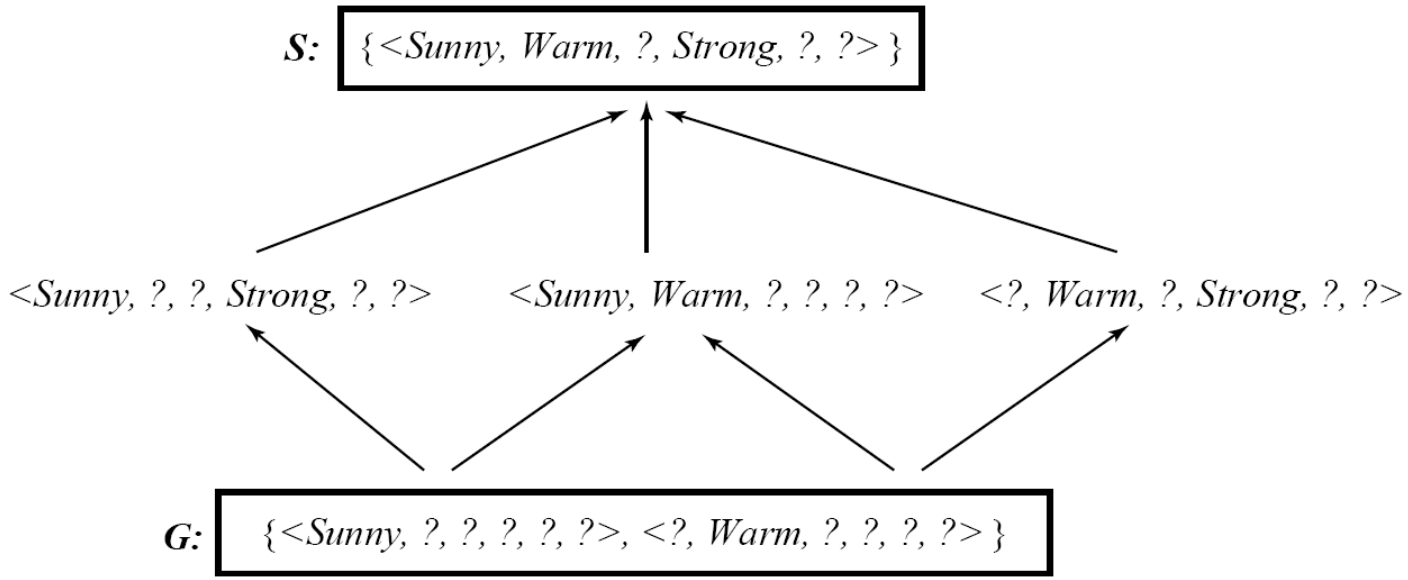
If you can ask questions:

Optimal Case: ceil(Log$_2$ |VS|)



<Sunny Warm Normal Light Warm Same>

How should these be classified?



<Sunny Warm Normal Strong Cool Change>

<Rainy Cool Normal Light Warm Same>

<Sunny Warm Normal Light Warm Same>

What justifies this?

- Inductive leap:

$$
\begin{array}{ll}
+ & \langle Sunny\ Warm\ Normal\ Strong\ Cool\ Change \rangle \\
+ & \langle Sunny\ Warm\ Normal\ Light\ Warm\ Same \rangle \\
\hline
S: & \langle Sunny\ Warm\ Normal\ ?\ ?\ ? \rangle
\end{array}
$$

- Why believe we can classify the unseen ?

  `<Sunny Warm Normal Strong Warm Same>`

An Unbiased Learner

- Idea: Choose *H* that expresses every teachable concept (i.e., *H* is the power set of *X*)
- Let *H'* be the set of disjunctions, conjunctions, negations over previous *H*. E.g.,

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

- What are *S*, *G* in the candidate-elimination algorithm in this case?
  - S = disjunction (OR) of positive examples
  - G = conjunction (AND) of negations (NOT) of negative examples

Universität Augsburg University

Fundamental property of inductive inference:

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

Inductive Bias

Consider

- 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.

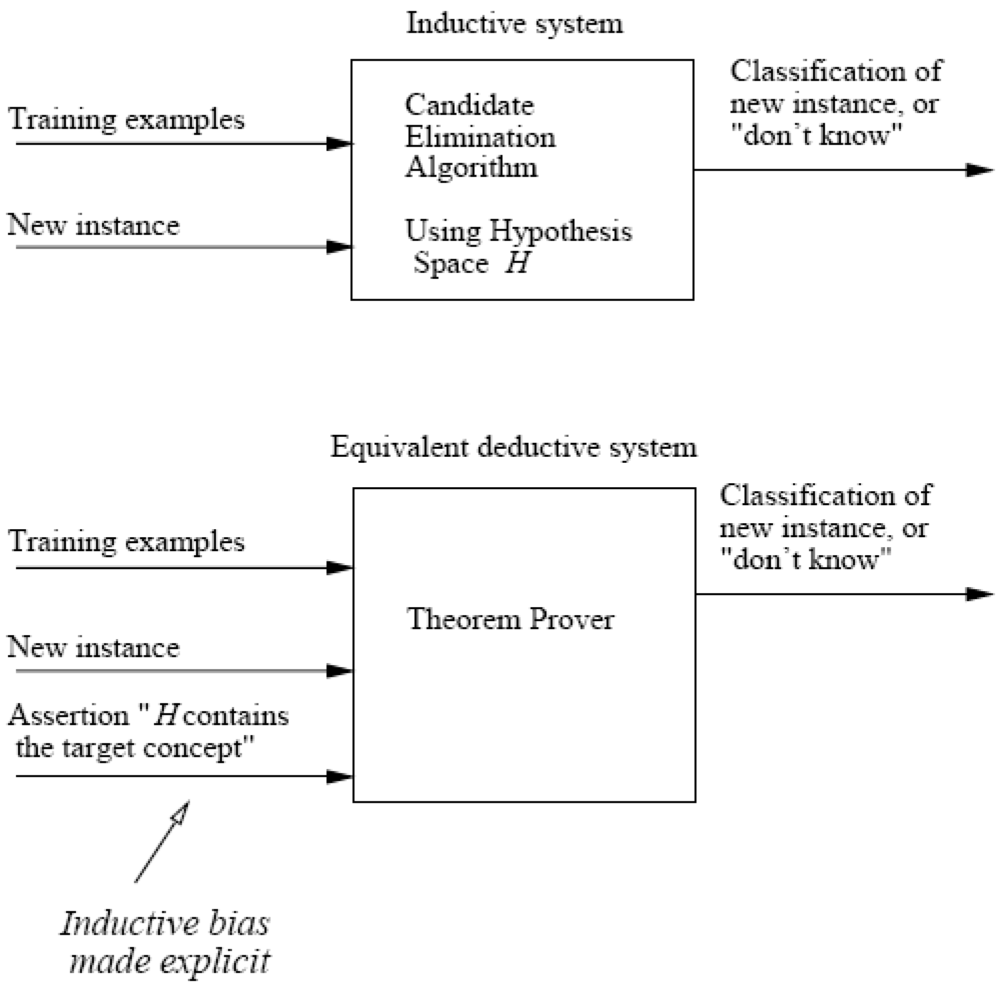
Definition:

- 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 and Equivalent Deductive Systems



lecture slides for textbook Machine Learning, T. Mitchell, McGraw Hill, 1997

Three Learners with Different Biases

1. *Rote learner*: Store examples, Classify *x* iff it matches previously observed example.
   ➔ no inductive bias

2. Version space *Candidate-Elimination* algorithm
   ➔ target concept can be represented in its hypothesis space

3. Find-S
   ➔ target concept can be represented it its hypothesis space + all instances are negative instances unless the opposite is entailed by the training data

Summary Points

1. Concept learning as search through *H*
2. General-to-specific ordering over *H*
3. Version space *Candidate-Elimination* algorithm
4. *S* and *G* boundaries characterize learner's uncertainty
5. Learner can generate useful queries
6. Inductive leaps possible only if learner is biased
7. Inductive learners can be modeled by equivalent deductive systems