Machine Learning

2 – Concept Learning

SS 2018

Gunther Heidemann

Overview

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

We use a simple approach assuming no noise to motivate the key concepts!

Idea

Fundamental problem:

How to learn general concepts from specific examples

(e.g., concept *car* from some specific cars)

A concept can be represented by a boolean function which assigns *true* to the appropriate entities, e.g.,

car(*MyPolo*) = *true*, *car*(*ThisChair*) = *false*

Introductory example

Task: Learn concept “days on which Aldo enjoys water sport” from samples

Training examples for *EnjoySport* :

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

[M]

- What is the general concept underlying the data ?
- How can a hypothesis for the concept be represented?

Representing hypotheses

Many possible representations!

Here, *h* is a conjunction of constraints on the attributes.

Each constraint can be

- A specific value, e.g., *Water = Warm.*
- Don't care, e.g., *Water = ?.*
- No value allowed, e.g., *Water = ∅.*

Example for a hypothesis:

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

Meaning: The hypothesis is that Aldo enjoys sports if the sky is sunny and the wind strong and the forecast the same. He does not care about air temperature, humidity and water.

Formal description of the learning task

Given:

- Instances X: Days, each described by the values of the attributes *Sky, AirTemp, Humidity, Wind, Water, Forecast*
- Target function c: *EnjoySport*: → {0, 1}.
- Training examples D: Positive and negative examples of the target function

$$D = \{<x_1, c(x_1)>, <x_2, c(x_2)>, \ldots, <x_m, c(x_m)>\}, \qquad x_i \in X.$$

- Hypotheses h: Conjunctions of literals, e.g.,

$$h = < ?, Cold, High, ?, ?, ? > .$$

Determine:

A hypothesis $h \in H$ such that

$$\forall x \in D: \quad h(x) = c(x).$$

Formal description of the learning task

Remarks:

- Concept learning is a search in the space of hypotheses for the hypothesis best fitting the examples.
- This space is defined by the hypothesis representation.
- Here: 3 values for *Sky*, 2 values for each of the other 5 attributes
- There are 3·2·2·2·2·2 = 96 distinct instances (distinct " day types")
- In addition to the attribute values, a hypothesis may contain the values ? and ∅, so there are 5·4·4·4·4·4 = 5120 syntactically correct and distinct hypotheses.
- But as a hypothesis containing at least one ∅ represents an empty set of instances, there are only 1+4·3·3·3·3·3 = 973 semantically distinct hypotheses.
- So the hypothesis representation yields a repertoire of 973 possible "explanations" for a set of sample data.

Inductive learning hypthesis

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.

Ordering of hypotheses

Idea how to search for hypotheses:

General to specific !

Exhaustive search in hypotheses space becomes possible without enumerating all hypotheses explicitly.

Example:

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

h_2 is a more general concept of *EnjoySport* than h_1.

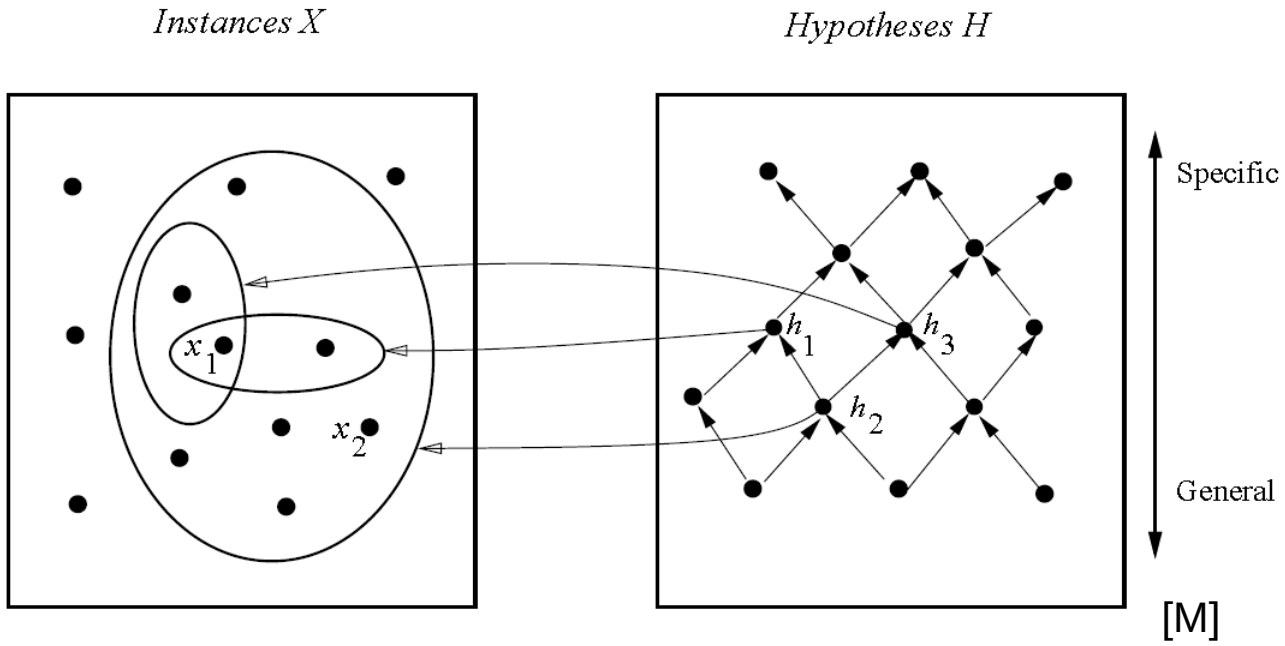
More precisely:

h_2 is more general than h_1 (or equally general), if and only if any instance that is classified positive by h_1 is also classified positive by h_2.

Instance, hypotheses, and more-general-than

Each hypothesis corresponds to the subset of instances that it classifies positive.

Arrows in *H* point from more to less general hypotheses.



[M]

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, ?>

Finding a maximally specific hypothesis

Find-S algorithm:

1. Initialize h to the most specific hypothesis in H.
2. For each positive training instance x do

    For each attribute constraint a_i in h do

        If (a_i is not satisfied by x) then

            Replace a_i in h by the next more general constraint that is satisfied by x.

        End if

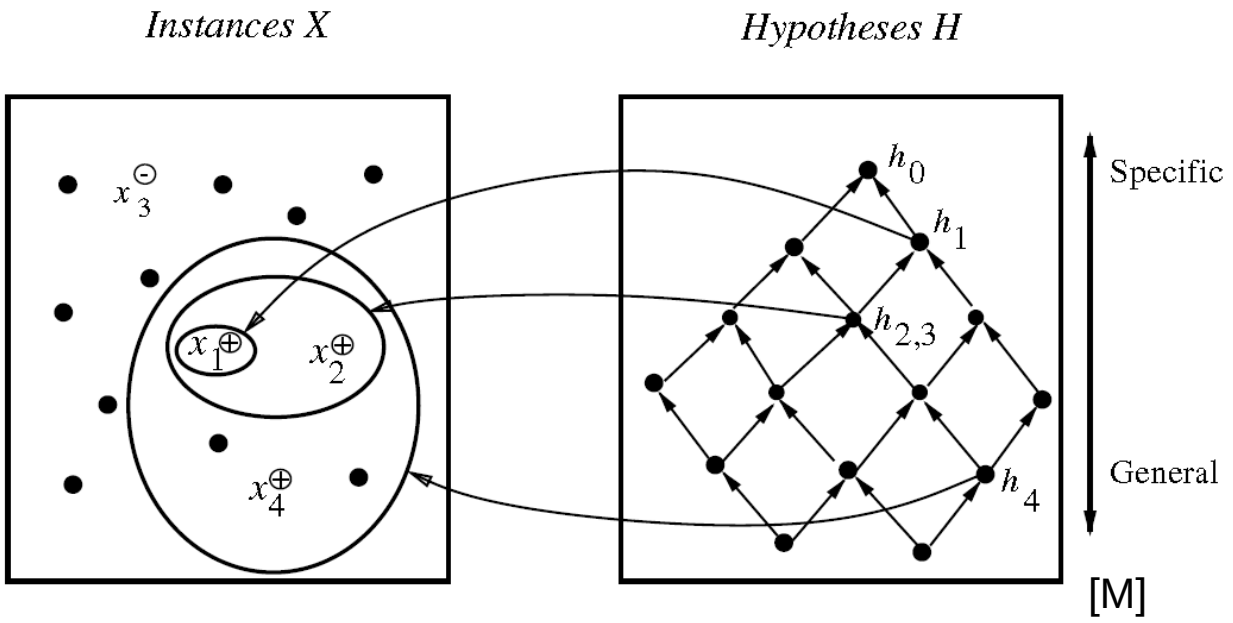
    End for

End for

3. Output h.

Hypotheses search by Find-S

Search starts with most specific hypothesis h_0 and increases generality as required by the examples.



[M]

$h_0 = <\ \varnothing,\ \varnothing,\ \varnothing,\ \varnothing,\ \varnothing,\ \varnothing\ >$

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

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

$x_3 = <Rainy,\ Warm,High,\ Strong,Cool,Change> -$ $h_3 = <Sunny,Warm,\ ?,\ Strong,Warm,Same >$

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

Complaints about Find-S

- Find-S learns nothing from negative examples.
- Can’t tell whether it has learned the concept.
- Can’t tell whether training data is inconsistent.
- Picks maximally specific *h*.
- Depending on *H*, there might be several solutions.

Version space

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 $(x,c(x)) \in D$.

$$Consistent(h, D) \Leftrightarrow \forall (x, c(x)) \in D: \; h(x) = c(x)$$

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.

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

List-Then-Eliminate algorithm

To obtain the version space, we start with all hypotheses. Then for each example the inconsistent hypotheses are eliminated. This is the List-Then-Eliminate algorithm:

1. $VS \leftarrow$ list containing all $h \in H$.
2. For each $(x,c(x)) \in D$ do
   For each $h \in VS$ do
   If $h(x) \neq c(x)$ then
   Remove h from VS
   End if
   End for
   End for
3. Output VS

List-then-eliminate algorithm

- List-Then-Eliminate can be applied if the hypothesis space *H* is finite.
- It computes the complete version space.
- Ideally, only one hypothesis remains, i.e., examples define the hypothesis exactly.
- If the version space is large, we need either more or more informative examples or other criteria to select a “good” hypothesis.
- If the version space is the empty set, there are inconsistent examples.
- List-Then-Eliminate requires enumerating all hypotheses.

➔ Impractical for real problems!

Example version space

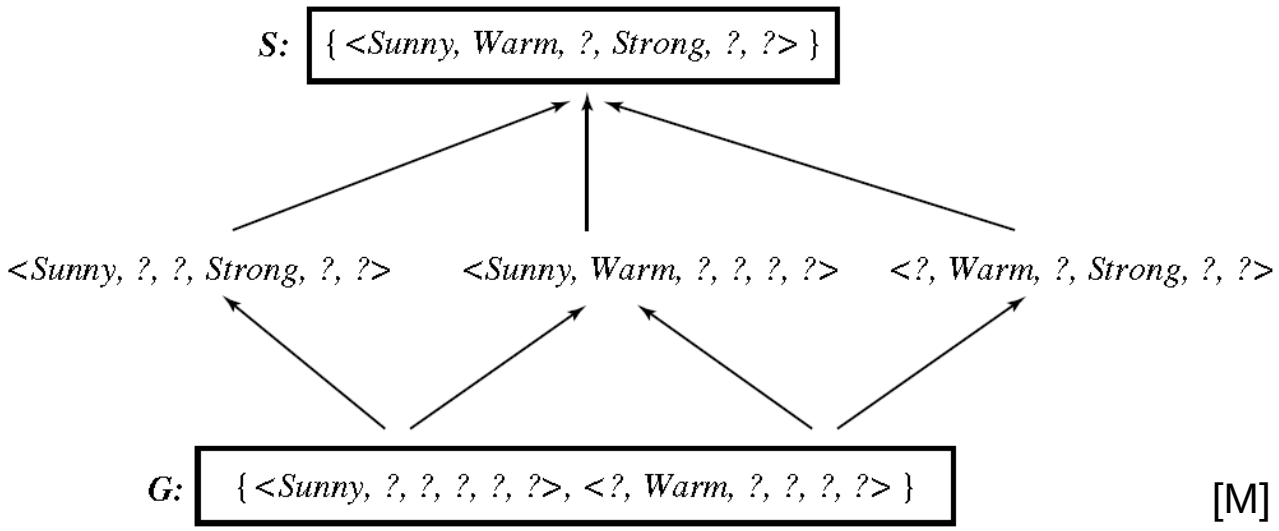
Examples and corresponding version space *VS*.

$|VS| = 6$.

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 point from general to less general hypotheses.

"Limits" of *VS* are the general boundary *G* and the specific boundary *S*.



[M]

Representing version spaces

To find an algorithm that computes the version space with reasonable effort, we first need to know its "boundaries":

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

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

Every member of the version space lies between (including) these boundaries:

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

where $x \geq y$ means "x is more general than y".

Representing version spaces

Idea how to compute the version space

- Like List-Then-Eliminate, start with complete *VS*, but do not enumerate all hypotheses *explicitly*.
- Instead, represent *VS* by its boundaries *G* and *S*. Start with the most general *G*

$$G_0 \leftarrow \{<?, ?, ?, ?, ?, ?>\}$$

and the most specific *S*

$$S_0 \leftarrow \{<\varnothing, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing>\}.$$

- G_0 and S_0 delimit the entire *VS*.
- For each example
  - specialize *G*,
  - generalize *S*.
- This is the Candidate-Elimination algorithm.

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 a positive example then

        Remove from G any hypothesis inconsistent with d (*)

        For each hypothesis $s \in S$ do

            If s inconsistent with d then

                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.

            End if

        End for

Candidate-Elimination algorithm

Else (i.e., if *d* is a negative example)

Remove from *S* any hypothesis inconsistent with *d*

For each hypothesis $g \in G$ do

If *g* inconsistent with *d* then

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*.

End if

End for

End else

End for

Example trace: Specific boundary

$S_0 = \{< \varnothing, \varnothing, \varnothing, \varnothing, \varnothing, \varnothing >\}$

$x_1 = <Sunny,Warm,Normal,Strong,Warm,Same>$ + →

$S_1 = \{<Sunny,Warm,Normal,Strong,Warm,Same>\}$

$x_2 = <Sunny,Warm,High,Strong,Warm,Same>$ + →

$S_2 = \{<Sunny,Warm, ? ,Strong,Warm,Same>\}$

$x_3 = <Rainy,Cold,High,Strong,Warm,Change>$ – →

$S_3 = \{<Sunny,Warm, ? ,Strong,Warm,Same>\}$

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

$S_4 = \{<Sunny,Warm, ? ,Strong, ? , ? >\}$

Example trace: General boundary

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

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

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

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

G_2 = {< ?, ?, ?, ?, ?, ? >}

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

G_1 = {< ?, ?, ?, ?, ?, ? >}

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

G_0 = {< ?, ?, ?, ?, ?, ? >}

Question

Step from $G_2 = \{<?, ?, ?, ?, ?, ? >\}$

to $G_3 = \{<Sunny,?,?,?,?,?>, <?,Warm,?,?,?,?>, <?,?,?,?,?,Same>\}$

triggered by $x_3 = <Rainy,Cold,High,Strong,Warm,Change>$ – **:**

Note we still do not know if, e.g., $x_{42} = <Rainy,Cold,Normal,Strong,Change>$ **+**

So why did we not chose the more general set

$G'_3 = \{<?, ?, ?, ?, ?, ? >\}$ **** $\{<Rainy,Cold,High,Strong,Warm,Change>\}$ **?**

Answer:

Because G'_3 does not conform to the way of representing a hypothesis we have chosen (though it might be reasonable to include set operations).

By contrast, the correct G_3 enumerates three valid hypotheses.

x_{42} is not covered by G_3, but that's a problem of the hypothesis representation.

Comments on example

Step from $G_2 = \{<?, ?, ?, ?, ?, ? >\}$

to $G_3 = \{<Sunny,?,?,?,?,?>, <?,Warm,?,?,?,?>, <?,?,?,?,?,Same>\}$

triggered by $x_3 = <Rainy,Cold,High,Strong,Warm,Change> -$ **:**

Why is $<?,?,Normal,?,? ,?>$ not included in G_3 as another minimal specialization?

Because it would not be consistent with

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

Question:

How did the algorithm remember x_2 ?

Answer: By boundary S_2 :

$<?,?,Normal,?,? ,?>$ is ***not*** more general than

$S_2 = \{<Sunny,Warm,?,Strong,Warm,Same>\}$,

because, e.g., $<Sunny,Warm,High,Strong,Warm,Same>$ satisfies S_2 but does not satisfy $<?,?,Normal,?,? ,?>$.

Comments on example

Up to step 3:

- Positive examples make *S* less specific and do not affect *G*.
- Negative example makes *G* less general and does not affect *S*.

But step 4 makes *G more* general again:

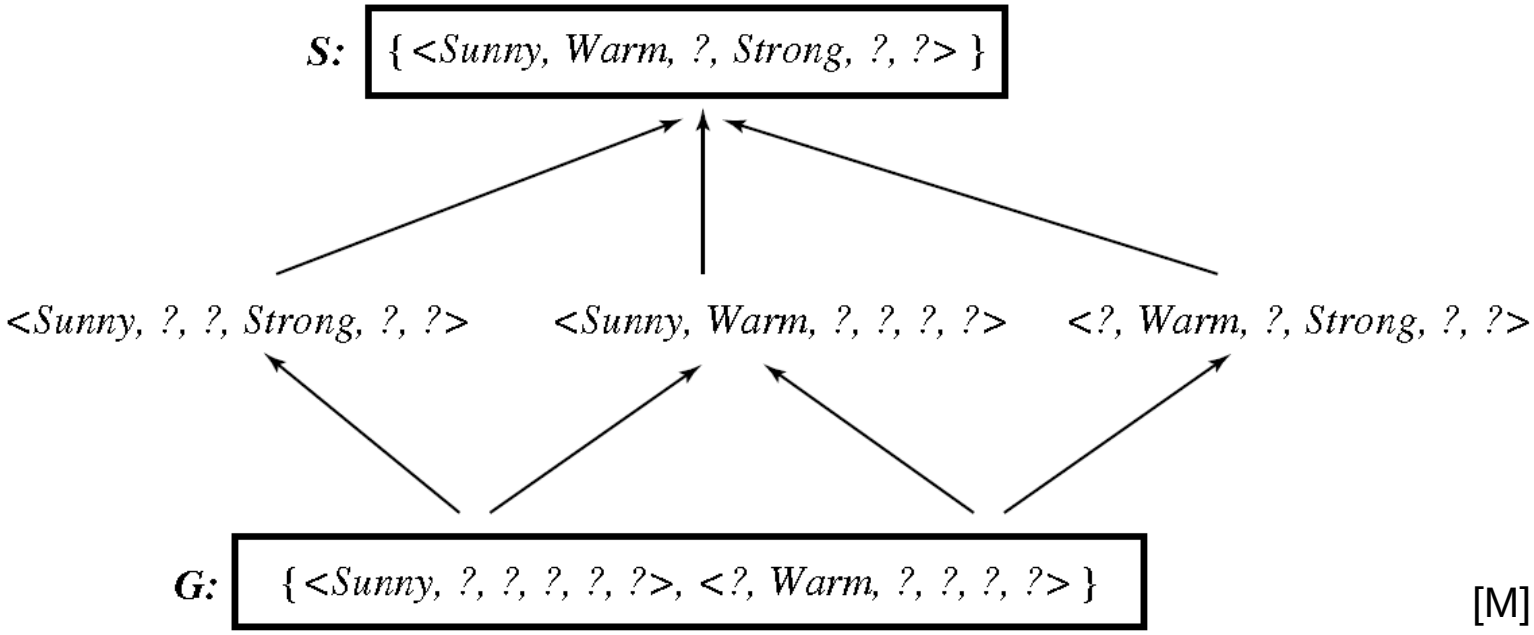
G_3 = { <*Sunny*, ?, ?, ?, ?, ?>, <?, *Warm*, ?, ?, ?, ?>, <?, ?, ?, ?, ?, *Same*> }

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

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

Comments on example

The resulting *VS*:



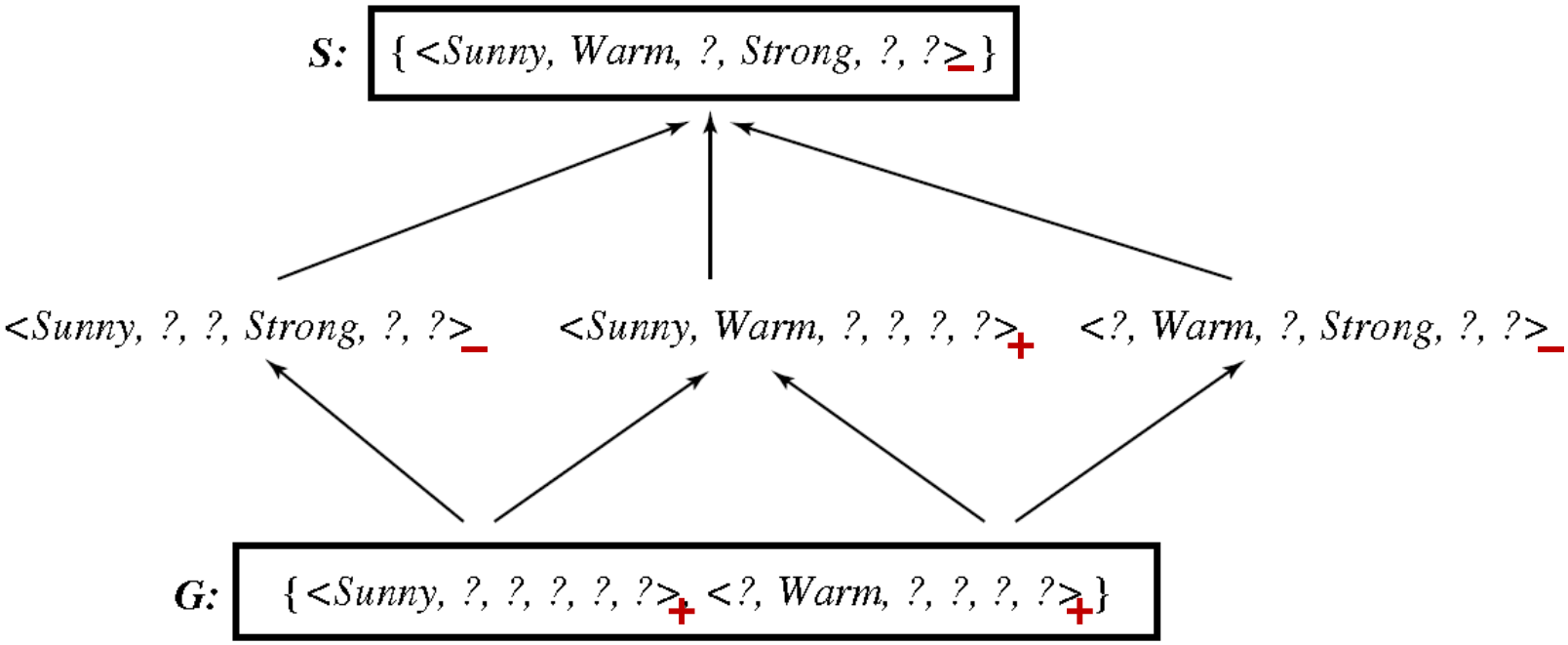
[M]

Note the resulting *VS* is independent of the order of presenting the examples.

Comments on version space

- Candidate-Elimination narrows the version space gradually.
- Size of the remaining version space reflects amount of missing information.
- Version space can be consulted to select the most informative examples by
  - selecting from a given set, or
  - active acquisition (make an experiment, ask a teacher).
- *S* and *G* converge to the correct hypothesis (if it exists).

Active example selection

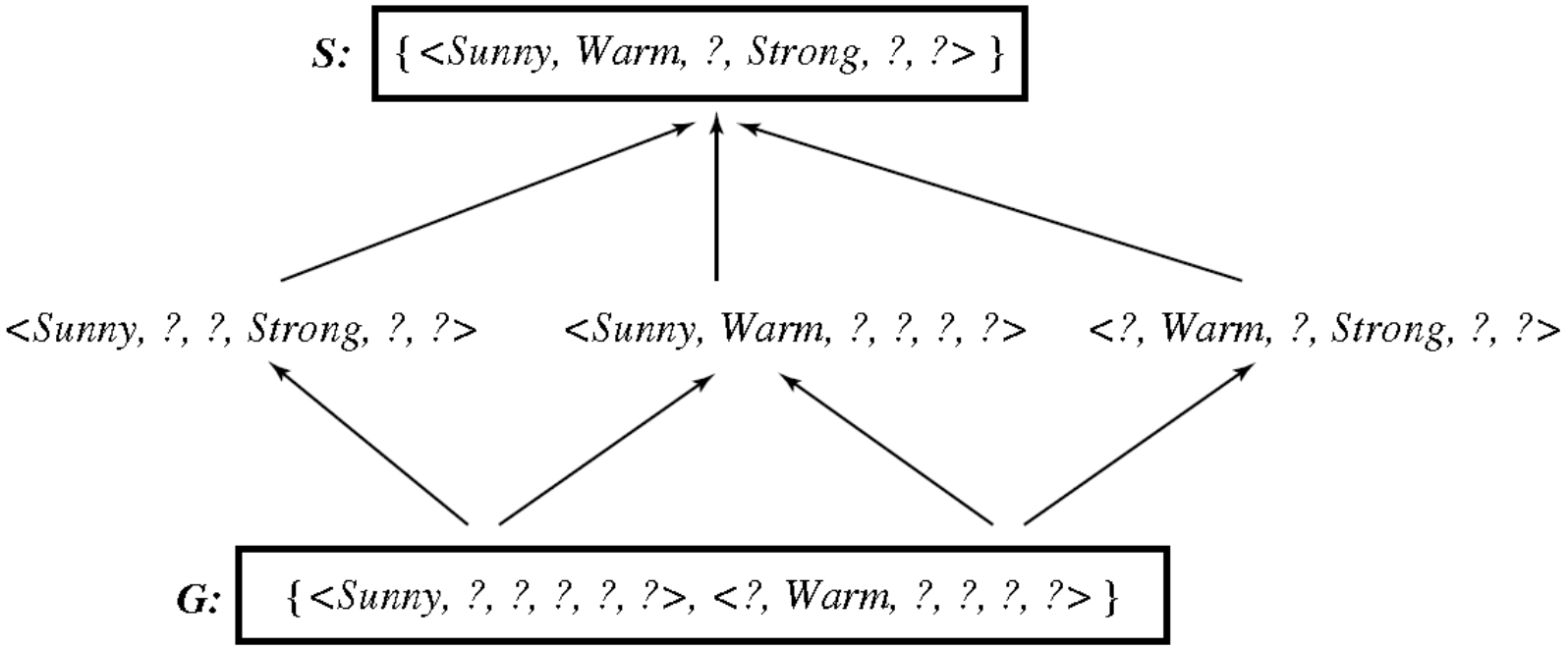


[M]

Which example would be most informative?

- → Example should discriminate among alternatives.
- → Choose example that is classified positive by some hypotheses, negative by others (optimally half – half).
- → x_5 = *<Sunny,Warm,Normal,Light,Warm,Same>*

Active example selection



[M]

Bad choices:

x_5 = <Sunny,Warm,Normal,Strong,Cool,Change> (all positive)

x_5 = <Rainy, Cool, Normal, Light, Warm,Same> (all negative)

Inductive bias

- Hypothesis space limits what any learning algorithm can find.
- Candidate-Elimination may end up with an empty VS though in principle a solution was possible (with another hypothesis space).
- In our example,

  x_1 = <*Sunny,Warm,Normal,Strong,Cool,Change*> +

  x_2 = <*Cloudy,Warm,Normal,Strong,Cool,Change*> +

  x_3 = <*Rainy,Warm,Normal,Strong,Cool,Change*> –

  has an empty version space (because x_1 and x_2 leave

  S_2 = <*?,Warm,Normal,Strong,Cool,Change*>)
- By choosing conjunctive hypotheses we have **biased** the learner.
- Should we use a hypothesis space that includes *every possible* hypothesis?

Representing all concepts by H'

- Set of instances X contains $|X| = 96$ "day types". How many concepts (= binary partitionings) can be defined over X ?
- # possible concepts = size of power set (set of all subsets) of X.
- Size of the power set is $2^{|X|} = 2^{96} \approx 10^{28}$.
- We define H' that can represent all subsets of X like H but allow also disjunctions, e.g.,

  $h = \langle Sunny,?,?,?,?,Change\rangle \vee \langle Rainy,Warm,?,?,?,?\rangle$

- What will Candidate-Elimination do on H' ?
  - Can learn every imaginable target concept.
  - S is always disjunction of positive, G the negated disjunction of negative examples.
  - **No generalization** beyond observed examples !
  - Converges to single hypothesis only when all instances of X have been presented.

Generalization in *H*

How was generalization achieved on *H* ?

From

x_1 = <*Sunny,Warm,Normal,Strong,Cool,Change*> +

x_2 = <*Sunny,Warm,Normal, Light, Warm, Same*> +

we inferred

S = <*Sunny,Warm,Normal, ?, ?, ?* >

without knowing the examples

x_3 = <*Sunny,Warm,Normal,Strong,Cool, Same*> + etc.

This generalization or “inductive leap” came about by the independence of the attribute underlying the construction of *H*.

Inductive bias

- Bias-free learning system makes no a priori assumptions.
- Thus it can merely collect the examples without generalization.
- Inductive learning makes sense only with prior assumptions (bias)!
- There is no more to learning than
  - collecting examples,
  - “interpolation” of some kind among the examples according to the inductive bias,
  - actively acquiring examples according to the version space.
- For every learning systems the employed inductive bias should be clear.
- In the following we will define the inductive bias more precisely.

Definition of the inductive bias

Given:

- Training set $D_c = \{<x, c(x)>\}$ for target concept c.
- Learning system L. After training on D_c, L yields the classification $L(x_i, D_c)$, for an unknown instance x_i. Notation:

$$D_c \wedge x_i \quad \mathsf{f} \quad L(x_i, D_c),$$

where $a \ \mathsf{f} \ b$ denotes b is *inductively inferred* from a.

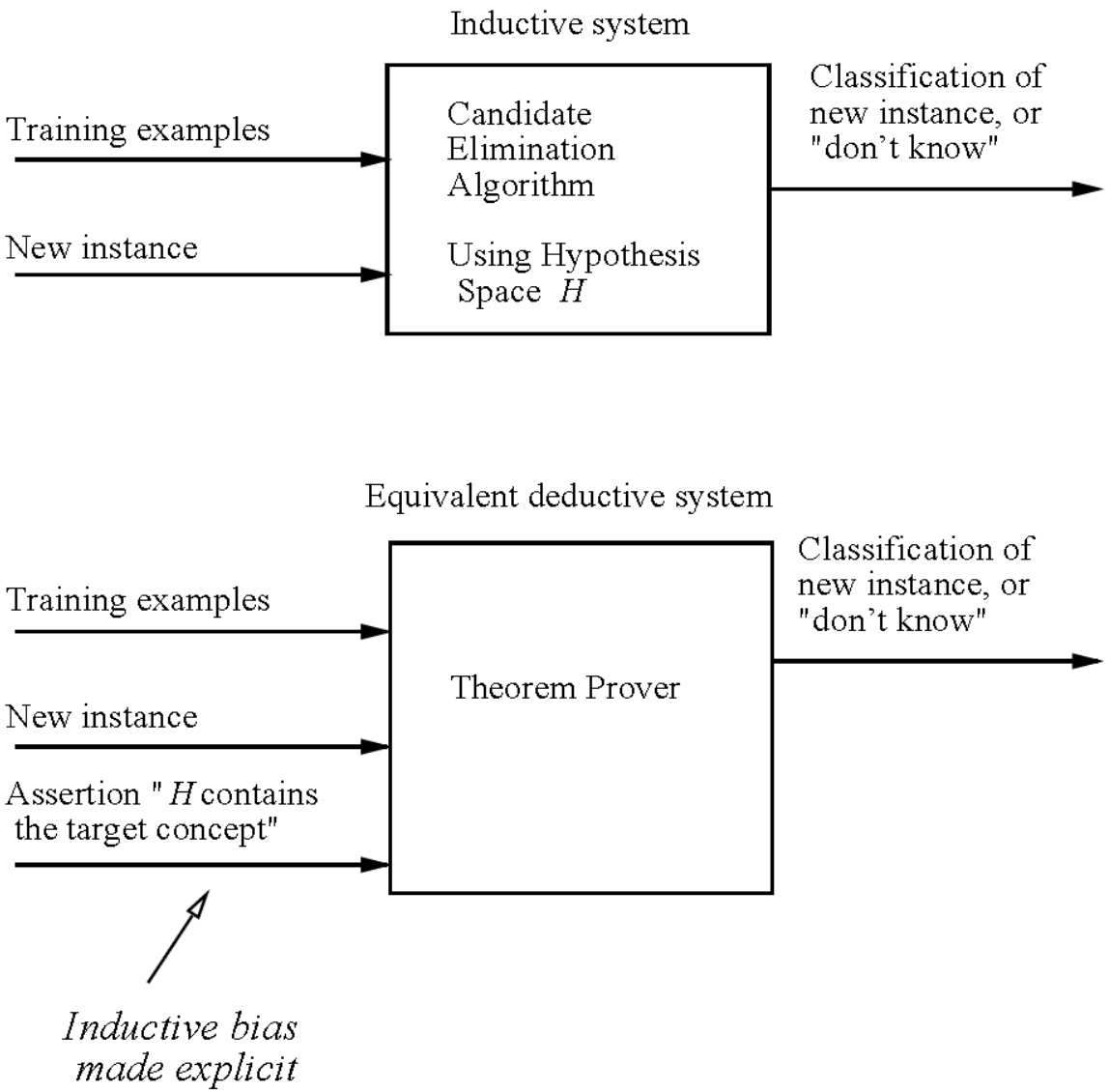
The **inductive bias** of L is any *minimal set of assertions* B such that

$$\forall \ x_i \in X : \quad B \wedge D_c \wedge x_i \quad \models \quad L(x_i, D_c)$$

where $a \models b$ denotes b is *deductively inferred* from (or logically entailed by) a.

So B is the knowledge necessary for the deductive inference that L has not gained from the examples but from prior assumptions.

Inductive systems and equivalent deductive systems



[M]

Summary

- Concept learning is a search in hypotheses space *H*
- General-to-specific ordering of hypotheses is useful
- Candidate-Elimination searches *H* by narrowing the version space using a specific boundary *S* and a general boundary *G*.
- Examples can be actively queried by analyzing the version space.
- Generalizations ("inductive leaps") are possible only if the learning system is biased by a priori assumptions.
- Inductive learning systems can be modeled by equivalent deductive systems to make the bias explicit.
- We will see that for many systems the bias
  - is distributed over many modules of the system
  - can not be made explicit because it results from the processing strategy, not a limitation on hypothesis representation.

Image sources

[M] Online material available at www.cs.cmu.edu/~tom/mlbook.html
for the textbook: Tom M. Mitchell: *Machine Learning*, McGraw-Hill