Introduction

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

Other definition . Inferring a boolean-valued function from training examples of its input and output.

Hypothesis

indicate by a "?' that any value is acceptable for this attribute,

specify a single required value (e.g., Warm) for the attribute, or

indicate by a "0" that no value is acceptable.

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

Example	*Sky*	*AirTemp*	*Humidity*	*Wind*	*Water*	*Forecast*	*EnjoySport*
1	Sunny	Warm	Normal	Strong	Warm	Same	Yes
2	Sunny	Warm	High	Strong	Warm	Same	Yes
3	Rainy	Cold	High	Strong	Warm	Change	No
4	Sunny	Warm	High	Strong	Cool	Change	Yes

Learning Task

The most general hypothesis-that every day is a positive example-is represented by (?, ?, ?, ?,?, ?)

and the most specific possible hypothesis-that no day is a positive example-is represented by (0,0,0,0,0,0)

To summarize, the **EnjoySport** concept learning task requires learning the set of days for which EnjoySport = yes,
describing this set by a conjunction of constraints over the instance attributes.

- **Given:**
  - Instances X: Possible days, each described by the attributes
    - *Sky* (with possible values *Sunny*, *Cloudy*, and *Rainy*),
    - *AirTemp* (with values *Warm* and *Cold*),
    - *Humidity* (with values *Normal* and *High*),
    - *Wind* (with values *Strong* and *Weak*),
    - *Water* (with values *Warm* and *Cool*), and
    - *Forecast* (with values *Same* and *Change*).
  - Hypotheses H: Each hypothesis is described by a conjunction of constraints on the attributes *Sky*, *AirTemp*, *Humidity*, *Wind*, *Water*, and *Forecast*. The constraints may be "?" (any value is acceptable), "Ø" (no value is acceptable), or a specific value.
  - Target concept c: $EnjoySport : X \rightarrow \{0, 1\}$
  - Training examples D: Positive and negative examples of the target function (see Table 2.1).
- **Determine:**
  - A hypothesis h in H such that $h(x) = c(x)$ for all x in X.

Inductive learning

When learning the target concept, the learner is presented a set of training examples, each consisting of an instance x from X, along with its target concept value c(x)

Instances for which c(x) = 1 are called positive examples, or members of the target concept. Instances for which c(x) = 0 are called negative examples, or nonmembers

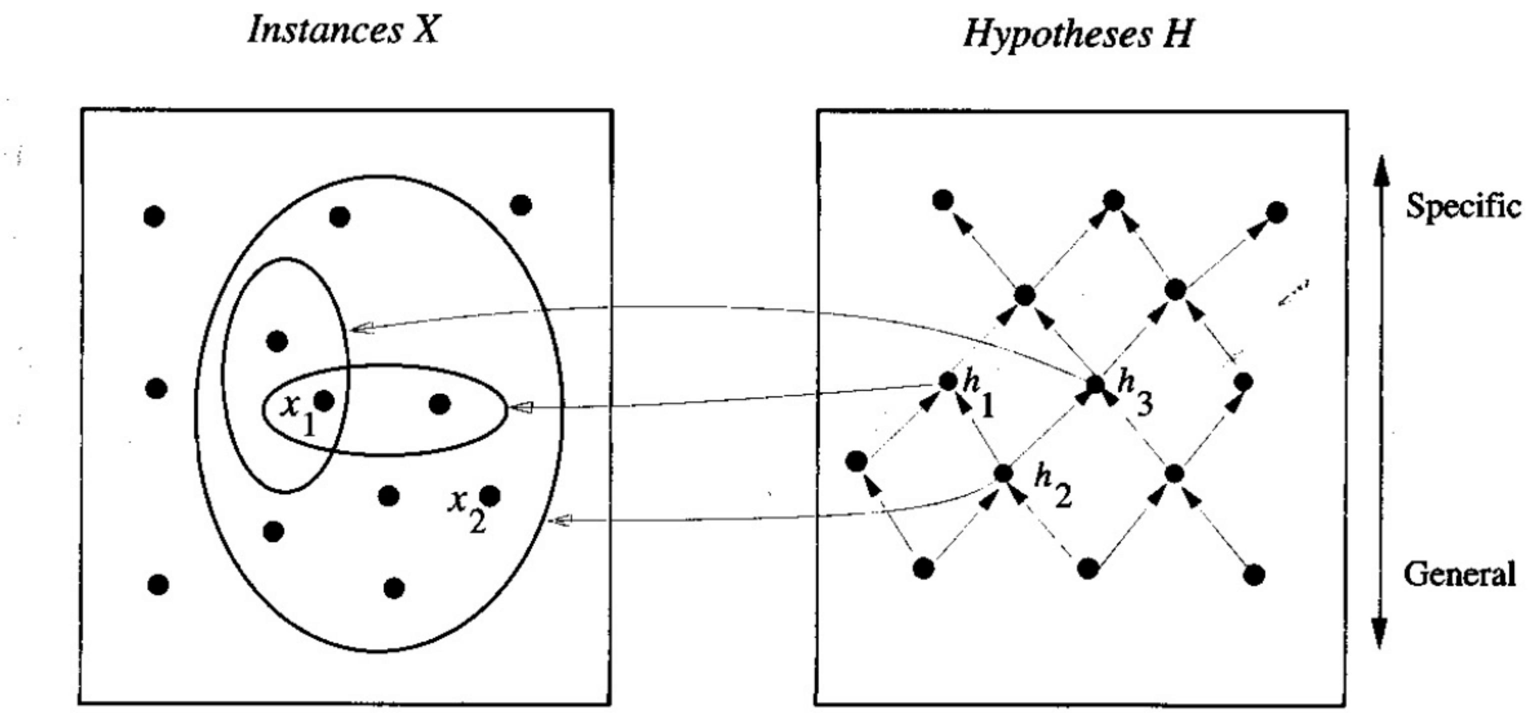
We use the symbol H to denote the set of all possible hypotheses that the learner may consider regarding the identity of the target concep

The goal of the learner is to find a hypothesis h such that
h(x) = c(x) for all x in X.

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.

Concept Learning as a search

Concept learning can be viewed as the task of searching through a large space of hypotheses implicitly defined by the hypothesis representation



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

Many algorithms for concept learning organize the search through the hypothesis space by relying on a very useful structure that exists for any concept Learning problem: a general-to-specific ordering of hypotheses.

Definition: Let h_j and h_k be boolean-valued functions defined over X. Then h_j is **more_general_than_or_equal_to** h_k (written $h_j \geq_g h_k$) if and only if

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

Given hypotheses hj and hk, hj is more-general-tan-or-equalto hk
if and only if any instance that satisfies hk also satisfies hi.

$$h_1 = \langle Sunny, ?, ?, Strong, ?, ? \rangle$$

$$h_2 = \langle Sunny, ?, ?, ?, ?, ? \rangle$$

Find-S

One way is to begin with the most specific possible hypothesis in H, then generalize this hypothesis each step, the hypothesis is generalized only as far as necesary to cover the new positive example

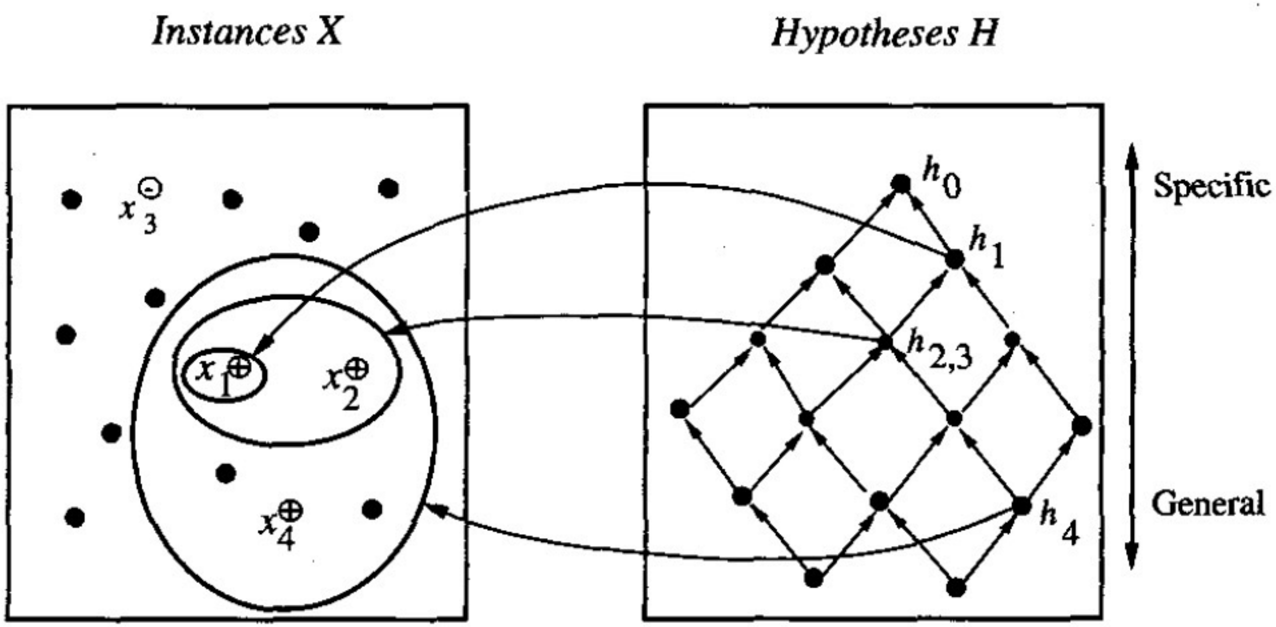
1. Initialize h to the most specific hypothesis in H
2. For each positive training instance x
   - For each attribute constraint a_i in h
     If the constraint a_i is satisfied by x
     Then do nothing
     Else replace a_i in h by the next more general constraint that is satisfied by x
3. Output hypothesis h

$$h \leftarrow \langle Sunny, Warm, Normal, Strong, Warm, Same \rangle$$

$$h \leftarrow \langle Sunny, Warm, ?, Strong, Warm, Same \rangle$$

$$h \leftarrow \langle Sunny, Warm, ?, Strong, ?, ? \rangle$$

Find-S



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

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_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 SPACES AND THE CANDIDATE-ELIMINATION ALGORITHM

TARGET: output a description of the set of all hypotheses consistent with the trainIng example

Definition: A hypothesis h is **consistent** with a set of training examples D if and only if $h(x) = c(x)$ for each 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**, denoted $VS_{H,D}$, with respect to hypothesis space H and training examples D, is the subset of hypotheses from H consistent with the training examples in D.

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

LIST-THEN-ELIMINATATE ALGORITHM

The LIST-THEN-ELIMINAalTgoEr ithm first initializes the version space to contain all hypotheses in H, then eliminates any hypothesis found inconsistent with any training example.

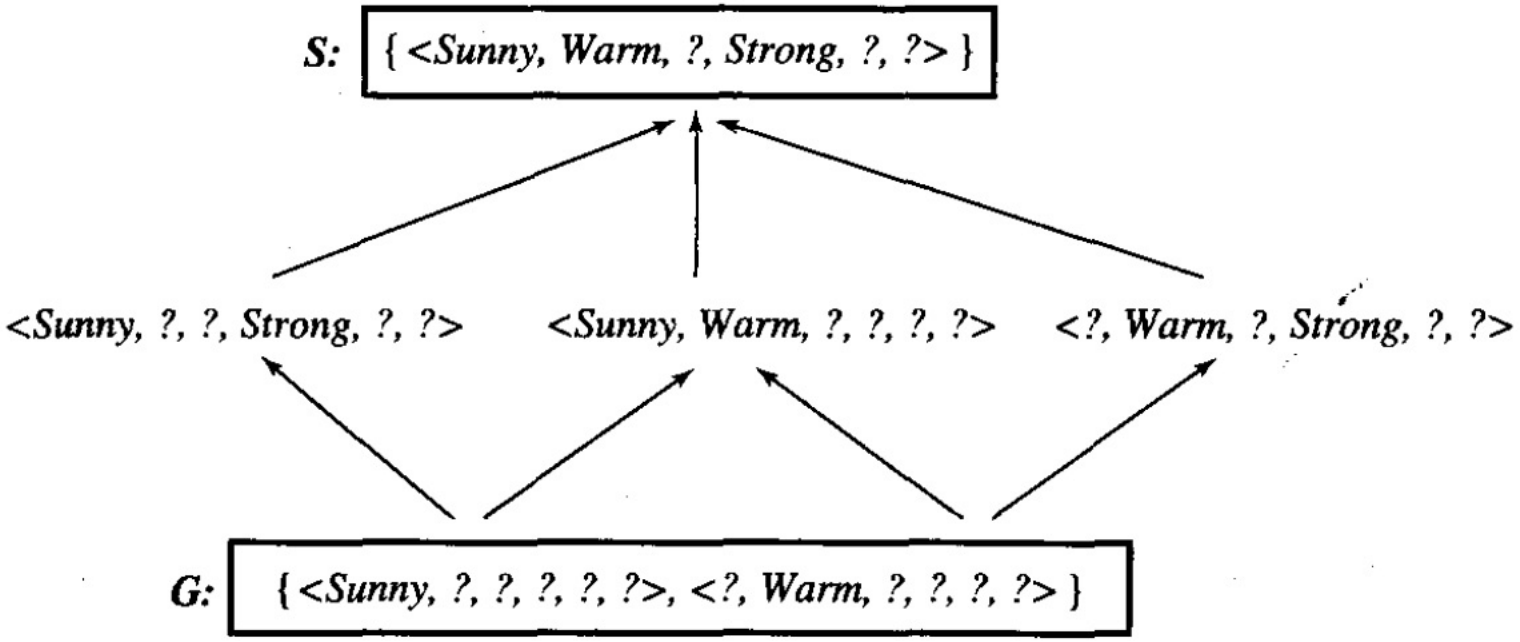
The LIST-THEN-ELIMINATE Algorithm

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 $VersionSpace$

Unfortunately,it requires exhaustively enumerating all hypotheses in H-an Unrealistic requirement for all but the most trivial hypothesis spaces.

A more compact representation for version spaces



The version space includes all six hypotheses shown here, but can be represented more simply by S and G. Arrows indicate instances of the relation *more-general-than*

A more compact representation

Output for Find_S

$$h = \langle Sunny, Warm, ?, Strong, ?, ? \rangle$$

this is just one of six different hypotheses from H that are consistent with these training examples

Definition: The **general boundary** G, with respect to hypothesis space H and training data D, is the set of maximally general members of H consistent with D.

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

Definition: The **specific boundary** S, with respect to hypothesis space H and training data D, is the set of minimally general (i.e., maximally specific) members of H consistent with D.

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

A more compact representation

Output for Find_S

$$h = \langle Sunny, Warm, ?, Strong, ?, ? \rangle$$

this is just one of six different hypotheses from H that are consistent with these training examples

Definition: The **general boundary** G, with respect to hypothesis space H and training data D, is the set of maximally general members of H consistent with D.

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

Definition: The **specific boundary** S, with respect to hypothesis space H and training data D, is the set of minimally general (i.e., maximally specific) members of H consistent with D.

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

the version space is precisely the set of hypotheses contained in G, plus those contained in S, plus those that lie between G and S in the partially ordered hypothesis space

CANDIDATE-ELIMINATION LEARNING ALGORITHM

Computes the version space containing all hypotheses from H that are consistent with an observed sequence of training examples.

1- Initializing the G boundary set to contain the most general hypothesis in H

2- initializing the S boundary set to contain the most specific (least general) hypothesis

$$G_0 \leftarrow \{\langle ?, ?, ?, ?, ?, ? \rangle\} \qquad S_0 \leftarrow \{\langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle\}$$

Initialize G to the set of maximally general hypotheses in H

Initialize S to the set of 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
      - 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
- 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

CANDIDATE-ELIMINATION LEARNING ALGORITHM

Step 1: Training examples 1 and 2 force the boundary to become S more general, as in the FIND-S algorithm. They have no effect on the boundary G

S_0: {<∅, ∅, ∅, ∅, ∅, ∅>}

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

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

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

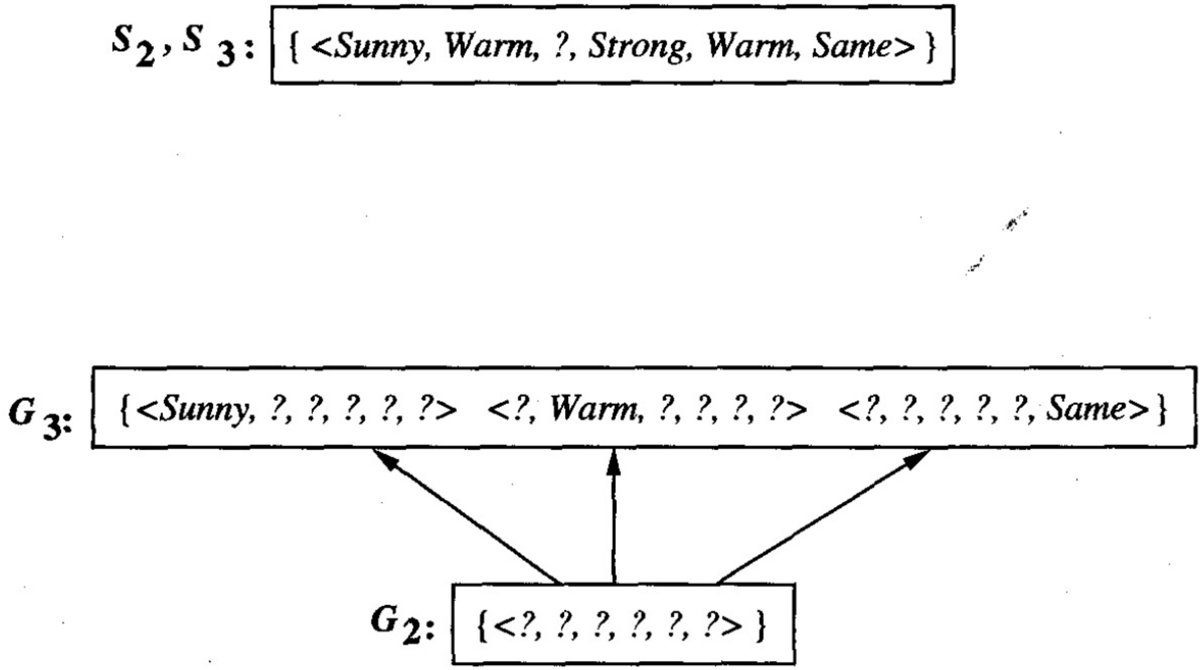
Training examples:

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

positive training examples may forcé the S boundary of the version space to become increasingly general.
Negative training examples play the complimentary role of forcing the G boundary to become increasingly specific

CANDIDATE-ELIMINATION LEARNING ALGORITHM

Step 2: Training example 3 is a negative example that forces the G2 boundary to be specialized to G3



Training Example:

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

positive training examples may forcé the S boundary of the version space to become increasingly general.
Negative training examples play the complimentary role of forcing the G boundary to become increasingly specific