CSCE 478/878 Lecture 2: Concept Learning and the General-to-Specific Ordering

Stephen D. Scott
(Adapted from Tom Mitchell's slides)

Outline

- Learning from examples
- General-to-specific ordering over hypotheses
- Version spaces and candidate elimination algorithm
- Picking new examples (making queries)
- The need for inductive bias
- Note: simple approach assuming no noise, illustrates key concepts

A Concept Learning Task: 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

Goal: Output a hypothesis to predict labels of future examples.

How to Represent the Hypothesis?

- Many possible representations
- Here, h will be conjunction of constraints on attributes
- Each constraint can be
  - a specfic value (e.g. $Water = Warm$)
  - don't care (i.e. "$Water = ?$")
  - no value allowed (i.e. "Water=∅")
- E.g.

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

(i.e. "If Sky == 'Sunny' and Wind == 'Strong' and Forecast == 'Same' then predict 'Yes' else predict 'No'.")

Prototypical Concept Learning Task

- **Given:**
  - Instance Space X, e.g. Possible days, each described by the attributes *Sky, AirTemp, Humidity, Wind, Water, Forecast* [all possible values listed in Table 2.2, p. 22]
  - Hypothesis Class H, e.g. conjunctions of literals, such as

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

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

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

    where $x_i \in X$ and $c : X \rightarrow \{0, 1\}$, e.g. $c = EnjoySport$
- **Determine:** A hypothesis $h \in H$ such that $h(x) = c(x)$ for all $x \in X$

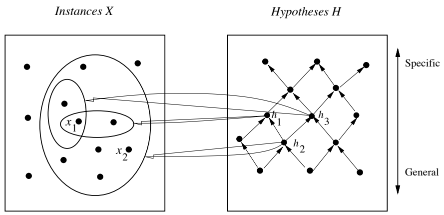
Prototypical Concept Learning Task (cont'd)

- Typically X is exponentially or infinitely large, so in general we can never be sure that $h(x) = c(x)$ for all $x \in X$ (can do this in special restricted, theoretical cases)
- Instead, settle for a good approximation, e.g. $h(x) = c(x)\ \forall x \in D$

The inductive learning hypothesis: Any hypothesis found to approximate the target function well over a sufficiently large set of training examples D will also approximate the target function well over other unobserved examples.

- Will study this more quantitatively later

The More-General-Than Relation



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, ?>

$$\boxed{h_j \geq_g h_k \text{ iff } (h_k(x) = 1) \Rightarrow (h_j(x) = 1)\ \forall x \in X}$$

$$h_2 \geq_g h_1, \ h_2 \geq_g h_3, \qquad h_1 \not\geq_g h_3, \ h_3 \not\geq_g h_1$$

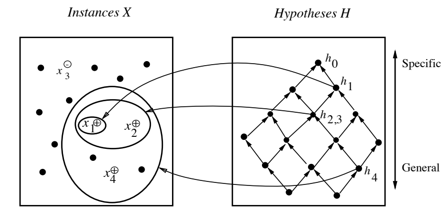
- So $\geq_g$ induces a partial order on hyps from H
- Can define $>_g$ similarly

Find-S Algorithm
(Find Maximally Specific Hypothesis)

1. Initialize h to $\langle \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset \rangle$, 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 replace a_i in h by the next more general constraint that is satisfied by x
3. Output hypothesis h

Why can we ignore negative examples?

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 ? ? >

Complaints about Find-S

- Assuming there exists some function in H consistent with D, Find-S will find one
- But Find-S cannot detect if there are other consistent hypotheses, or how many there are. In other words, if $c \in H$, has Find-S found it?
- Is a maximally specific hypothesis really the best one?
- Depending on H, there might be several maximally specific hyps, and Find-S doesn't backtrack
- Not robust against errors or noise, ignores negative examples
- Can address many of these concerns by tracking the entire set of consistent hyps.

Version Spaces

- 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

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

- 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 $VersionSpace$

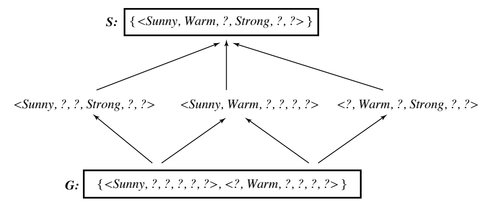
- Problem: Requires $\Omega\,(|H|)$ time to enumerate all hyps.

Representing Version Spaces

- 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 these boundaries

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

Example Version Space



Candidate Elimination Algorithm

$G \leftarrow$ set of maximally general hypotheses in H

$S \leftarrow$ set of maximally specific hypotheses in H

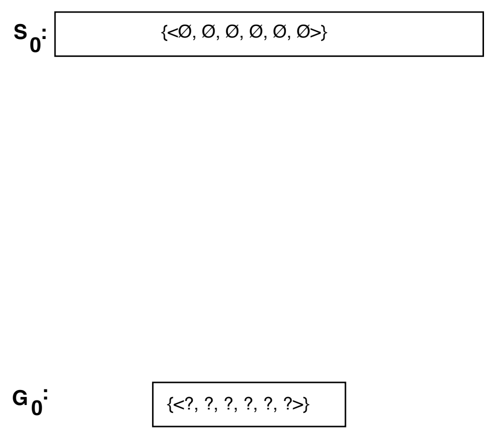
For each training example $d \in D$, do

- If d is a positive example
  - Remove from G any hyp. 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

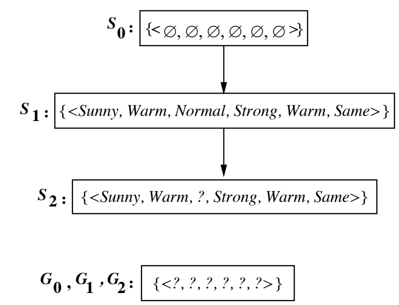
Candidate Elimination Algorithm
(cont'd)

- If d is a negative example
  - Remove from S any hyp. 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



Example Trace
(cont'd)

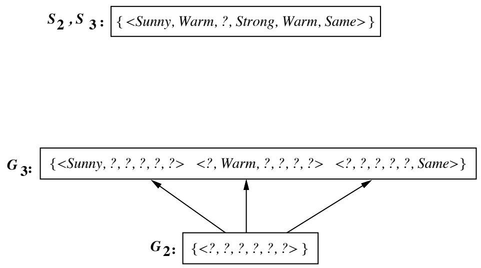


Training examples:

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

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

Example Trace
(cont'd)

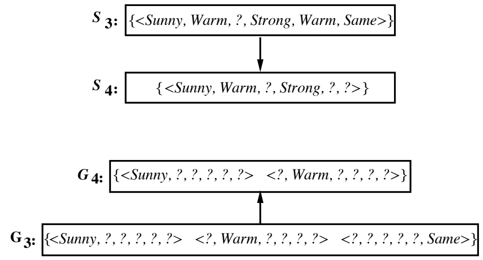


Training Example:

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

Why is $|G_3|$ only 3?
E.g. why $\langle ?, ?, Normal, ?, ?, ? \rangle \notin G_3$

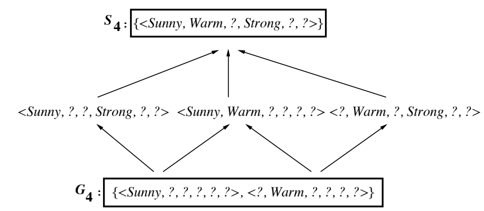
Example Trace
(cont'd)



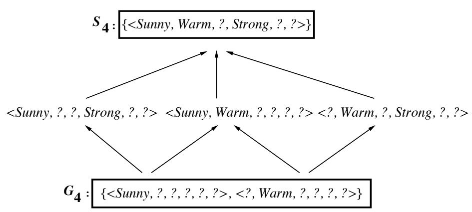
Training Example:

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

Final Version Space



Aside: Asking Queries

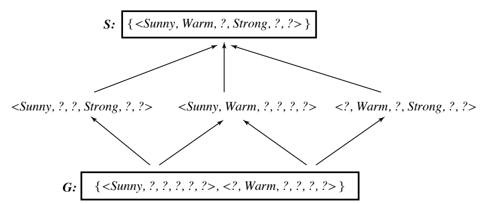


- What if the learner can ask queries, i.e. present an example and have a teacher (oracle) give classification? [Like running experiments]
- Why is

  $$\langle Sunny, Warm, Normal, Light, Warm, Same \rangle$$

  a good query to make?
- In general, what is a good strategy?

Generalizing Beyond Training Data



$$\langle Sunny\ Warm\ Normal\ Strong\ Cool\ Change \rangle \quad (1)$$

[Unanimous "yes" over version space]

$$\langle Rainy\ Cool\ Normal\ Light\ Warm\ Same \rangle \quad (2)$$

[Unanimous "no" over version space]

$$\langle Sunny\ Warm\ Normal\ Light\ Warm\ Same \rangle \quad (3)$$

[1/2 no, 1/2 yes]

Why believe we can accurately classify (1) and (2)?

Why not (3)?

An UNBiased Learner

- What if we assumed nothing about the structure of c?
- Then learning becomes rote memorization, e.g. if c is any boolean function over 3 variables with $D = \{\langle (000), + \rangle, \langle (110), + \rangle, \langle (010), - \rangle, \langle (101), - \rangle\}$, then version space is defined by $S = \{(000) \vee (110)\}$ and $G = \{\neg((101) \vee (010))\}$
- Originally $VS = 2^X =$ power set of X; now it is the set of truth tables satisfying the following:

000	$+$	010	$-$	100		110	$+$
001		011		101	$-$	111	

- Since there are 4 holes, $|VS| = 2^4 = 16 =$ number of ways to fill holes, and for any yet unclassified example x, exactly half of hyps in VS classify x as $+$ and half as $-$
- Thus, cannot generalize without bias!

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 classification assigned to instance x_i by L after training on data D_c

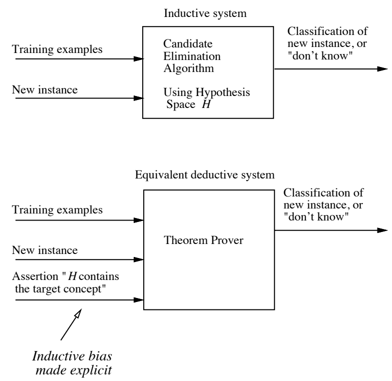
Definition:

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 $y \vdash z$ means y logically entails z

Inductive Systems and Equivalent Deductive Systems



Three Learners with Different Biases

1. *Rote learner:* Store examples, Classify x iff it matches previously observed example
   Bias:
2. *Version space candidate elimination algorithm*
   Bias:
3. *Find-S*
   Bias:

Generally, stronger bias $\Rightarrow$ ability to generalize on more examples from X, but correctness of learner depends on correctness of bias!