Version space learning

Language for examples/hypotheses

$L = \{[A, B], A \in T_1, B \in T_2\}$.
T_1 – taxomomy of colors
T_2 – taxomomy of planar geometric shapes.

```
                         any_color
                    --------|--------
                   |                 |
                 mono               poly
             ------|--------        --|--
            |   |     |     |      |     |
           red blue white black  orange pink
```

```
                           any_shape
               ---------------|---------------
              |                               |
           polygon                           oval
     ---------|-------                        |
    |                 |                       |
 3-sided           4-sided                    |
    |          -------|--------            ---|----
    |         |       |        |          |        |
triangle  rectangle square trapezoid    circle  ellipse
```

Ordering relations

$[A_1, B_1] \geq [A_2, B_2]$, if A_2 is a successor of A_1 in T_1, and B_2 is a successor of B_1 in T_2.

$[red, polygon] \geq [red, triangle]$
$[any\_color, any\_shape]$ covers all possible examples in L (how many?)

Induction task

Given:
$E_1^+ = [red, square]$
$E_2^+ = [blue, rectangle]$
$E_3^- = [orange, triangle]$

Find H, such that:
$H \geq E_1^+$, $H \geq E_2^+$
$H \not\geq E^-$

Hypothesis space

- Generate all generalizations of E_1^+, E_2^+. $S_H = \{[mono, 4 - sided]$, $[mono, polygon]$, ..., $[any\_color, any\_shape]\}$.
- Remove from S_H all hypotheses that cover E_3^-. *Version space*, $VS = \{[mono, 4{-}sided], [mono, polygon], [mono, any\_shape], [any\_color, 4{-}sided]\}$.

Specific to general search (Find-S)

Maximally specific generalizations S**:** $H \in S$, if $H \in VS$ and for any $H' \in VS$, $H' \geq H$.

Begin Find-S

Initialize S to the first positive example

Initialize N to all negative examples seen so far

For each positive example E^+ do begin

Replace every $H \in S$, such that $H \ngeq E^+$, with all its generalizations that cover E^+

Delete from S all hypotheses that cover other hypotheses in S

Delete from S all hypotheses that cover any element from N

End

For every negative example E^- do begin

Delete all members of S that cover E^-

Add E^- to N

End

End

End Find-S

General to specific search (Find-G)

Maximally general hypotheses G: $H \in G$ if it covers none of the negative examples, and for any other hypothesis H' that covers no negative examples, $H \geq H'$.

Begin Find-G

Initialize G to the most general concept in the version space

Initialize P to all positive examples seen so far

For each negative example E^- do begin

Replace every $H \in G$, such that $H \geq E^-$, with all its specializations that do not cover E^-

Delete from G all hypotheses more specific (covered by) other hypotheses in G

Delete from G all hypotheses that fail to cover some example from P

End

For every positive example E^+ do begin

Delete all members of G that fail to cover E^+

Add E^+ to P

End

End

End Find-G

Combining Find-S and Find-G

Boundary set theorem (Genesereth and Nilsson, 1987): For every $H \in VS$, there exist $H' \in S$ and $H'' \in G$, such that $H \geq H'$ and $H'' \geq H$.

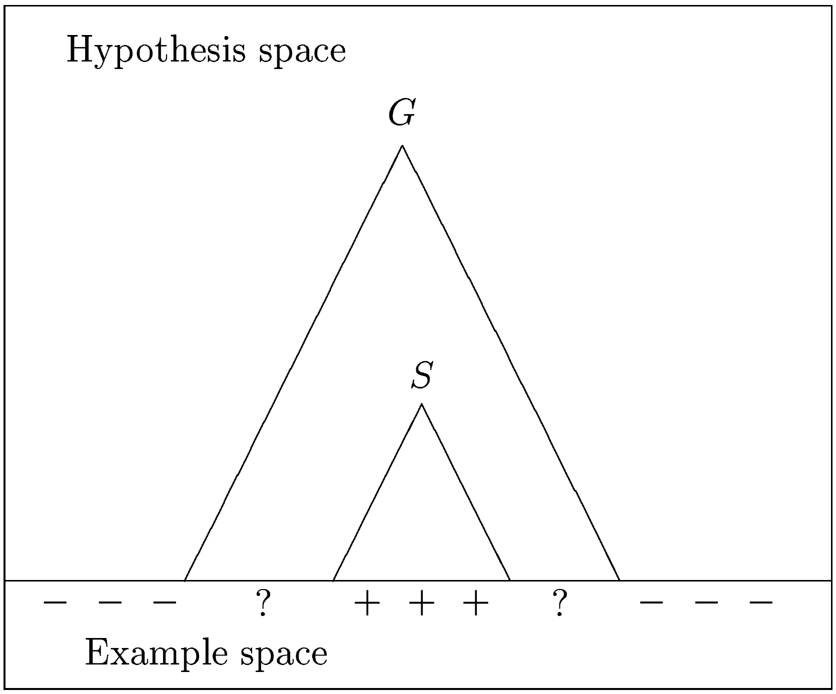
Candidate elimination algorithm (Mitchel, 82)

Put together Find-S and Find-G and:

- Replace "Delete from S all hypotheses that cover any element from N" with "Delete from S any hypothesis not more specific than some hypothesis in G"
- Replace "Delete from G all hypotheses that fail to cover some example from P" with "Delete from G any hypothesis more specific than some hypothesis in S"
- No need of maintaining P and N.

Stopping conditions

- If $G = S$ and both are singletons, then stop. The algorithm has found a single hypothesis consistent with the examples.
- If G or S becomes empty then stop. Indicate that there is no hypothesis that covers all positive and none of the negative examples.



Experiment Generation, Interactive Learning

1. Ask for the first positive example
2. Calculate S and G using the candidate elimination algorithm
3. Find E, such that $G \geq E, \forall s \in S, E \not\geq s$ (E is not in the version space).
4. Ask about the classification of E
5. Go to 2