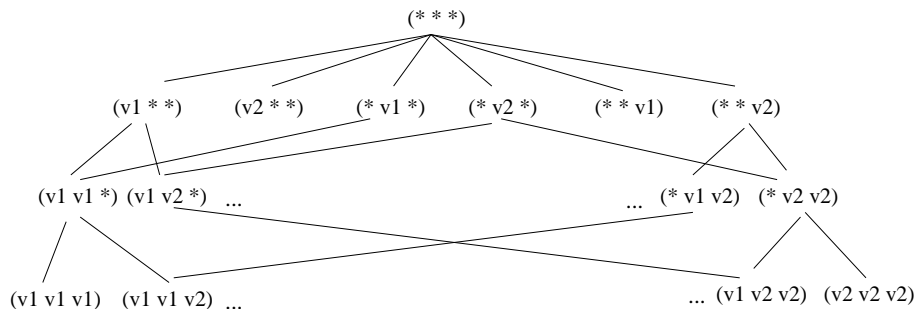


The Version Space

- General supervised inductive learning algorithm
- Examples are represented as value–attribute pairs
- It can learn concepts described by logic formulas
- The concepts that can be learned are organized in a hierarchy (general/specific)
- A partial order is defined among concepts (lattice)
- (We are only going to use pure conjunctive formulas)

Hierarchy of concepts



Principles

- Learning is obtained by searching for the concept in the hierarchy that best fits the examples
- There are two classes of examples, the positives (examples of the concept to learn) and negative (counterexamples of the concept)
- We define as **version space** the set of all hypothesis consistent with the examples that have been presented so far
- The goal is to reduce the hypothesis set to a single concept

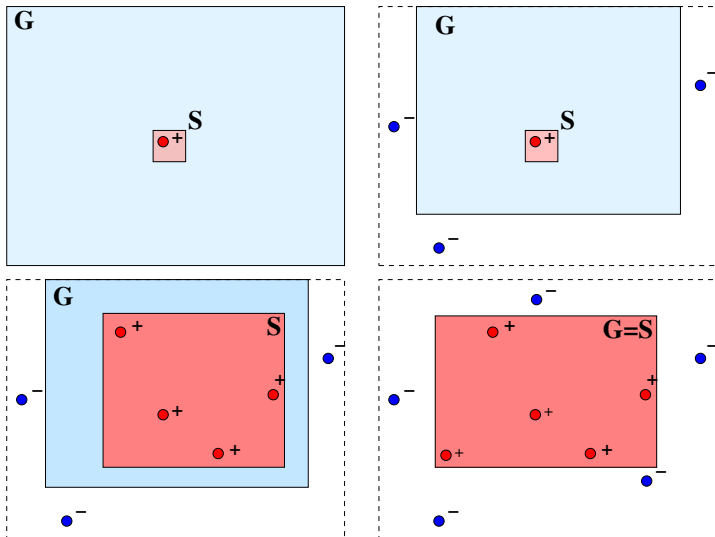
Algorithm

- A breadth first bidirectional search is performed
- The more general concepts consistent with the examples are stored in (**G**) and the more specific ones in (**S**)
- Positive examples are used to prune the more specific hypothesis
- Negative examples will be use to prune the more general hypothesis
- If the set of learning examples is correct the search converge

Algorithm

- Set G = Most general hypothesis consistent with the examples
- Set S = Most specific hypothesis consistent with the examples
- Adequate generalization and specialization operators for the concept representation language must be chosen
- Positive examples allow to generalize the most specific hypothesis (for instance, deleting conditions)
- Negative examples allow to specialize the most general hypothesis
- Also must hold that $S \subset G$

Algorithm



Candidate elimination algorithm (I)

Initialize \mathbf{G} to the most general concept

Initialize \mathbf{S} to the first positive example

while there are examples

if it is a positive example (\mathbf{p})

* **Delete** from \mathbf{G} any hypothesis inconsistent with \mathbf{p}

(Concepts from \mathbf{G} that **do not include** \mathbf{p})

* **for each** concept from \mathbf{S} inconsistent with \mathbf{p} (\mathbf{s})

- **Delete** \mathbf{s}

- **Add** to \mathbf{S} all **minimal generalizations** of \mathbf{s} that are consistent with \mathbf{p} and an element from \mathbf{G} is more general than them

* **Delete** from \mathbf{S} all concepts more general than any from \mathbf{S}

Candidate elimination algorithm (II)

if it is a negative example (\mathbf{n})

- * **Delete** from \mathbf{S} any hypothesis inconsistent with \mathbf{n}
(Concepts from \mathbf{S} that **include** \mathbf{n})

- * **For each** concept from \mathbf{G} inconsistent with \mathbf{n} (\mathbf{g})

- **Delete** \mathbf{g}

- **Add** to \mathbf{G} all **minimal specializations** of \mathbf{g} that are consistent with \mathbf{n} and an element from \mathbf{S} is more specific than them

- * **Delete** from \mathbf{G} all concepts less general than any from \mathbf{G}

end while

if $\mathbf{G}=\mathbf{S}$ and both have only one element this is the goal concept

Otherwise the set of examples is inconsistent

Shortcomings of the algorithm

- The exhaustive search is very costly
- Improvements:
 - To use simpler description languages (some concepts can not be learned)
 - To use heuristics to prune concepts from G and S (give a preference criteria over the hypothesis space, a bias)
- It is not tolerant to misclassified examples (noise)

LEX: an application to symbolic integration

- LEX is a symbolic integrator that learns from experience
- The hypothesis space of LEX is all the algebraic expressions
- Concepts: What integration operators are more adequate for different kinds of indefinite integrals

$$OP1 : \int rf(x)dx \rightarrow r \int f(x)dx$$

$$OP2 : \int udv \rightarrow uv - \int vdu$$

$$OP3 : \int f_1(x) + f_2(x)dx \rightarrow \int f_1(x) + \int f_2(x)$$

LEX: an application to symbolic integration

- The system is able to generate problems and label each operator depending on its success in solving a specific kind of integral as positive or negative example of application
- Each operator appears has one or more version spaces (disjunction)
- The version spaces are modified with the new positive or negative examples of application of operators
- If an expression is inside a version space of an operator this means that could be applicable to solve the integration of the expression

LEX: Example

EV OP2

$$\int f_1(x) f_2(x) dx$$

$$\int \text{pol}(x) f_2(x) dx \qquad \int f_1(x) \text{trig}(x) dx$$

$$\int \text{pol}(x) \text{sen}(x) dx \qquad \int 3x \text{trig}(x) dx$$

$$\int 3x \text{sen}(x) dx$$