Section 19.1 Version Spaces

CS4811 - Artificial Intelligence

Nilufer Onder
Department of Computer Science
Michigan Technological University

Outline

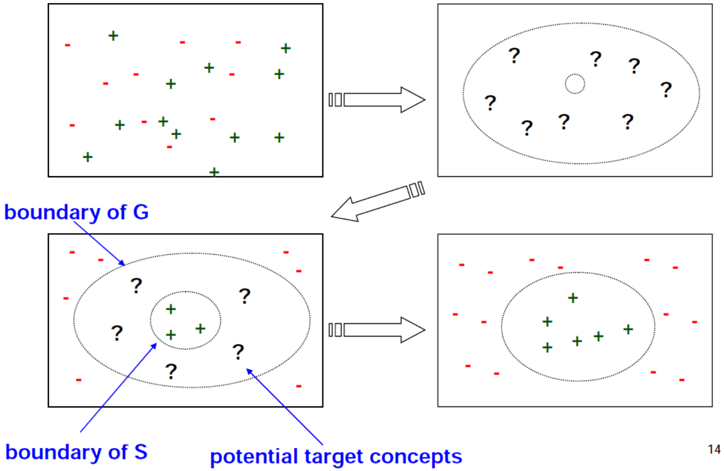
Example with playing cards

- Consider a deck of cards where a subset of these cards are "good cards." The *concept* we are trying to learn is the set of good cards.
- Someone shows cards one by one, and tells whether it is a good card or not.
- We maintain the description of the concept as *version space*. Everytime we see an example, we narrow down the version space to more accurately represent the concept.

The main components of the version space algorithm

- Initialize using two ends of the hypothesis space:
  the most general hypothesis and
  the most specific hypotheses
- When a positive example is seen, minimally generalize the most specific hypothesis.
- When a negative example is seen, minimally specialize the most general hypothesis.
- Stop when the most specific hypothesis and the most general hypothesis are the same.
  At this point, the algorithm has *converged*, and the target concept has been found.
- This is essentially a bidirectional search in the hypothesis space.

Progress of the version space algorithm



Simplified representation for the card problem

For simplicity, we represent a concept by rs, where r is the rank and s is the suit.

$r : a, n, f, 1, \ldots, 10, j, q, k$
$s : a, b, r, \clubsuit, \spadesuit, \diamondsuit, \heartsuit$

For example,
$n\spadesuit$ represents the cards that have a number rank, and spade suit.
aa represents all the cards: any rank, any suit.

Starting hypotheses in the card domain

- The most general hypothesis is:
  "Any card is a rewarded card."
  This will cover all the positive examples, but will not be able to eliminate any negative examples
- The most specific hypothesis possible is the list of rewarded cards
  "The rewarded cards are: $4\clubsuit, 7\clubsuit, 2\spadesuit$"
  This will correctly sort all the examples in the training set. However, it is overly specific, and will not be able to sort any new examples.

Extension of a hypothesis

The *extension* of an hypothesis h is
the set of objects that verifies h.

For instance,
the extension of $f\spadesuit$ is: $\{j\spadesuit, q\spadesuit, k\spadesuit\}$, and
the extension of aa is the set of all cards.

More general/specific relation

Let $h1$ and $h2$ be two hypotheses in H.

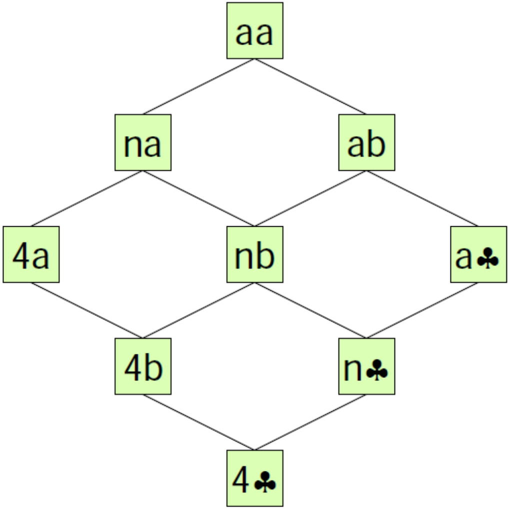
Hypothesis $h1$ is *more general* than $h2$ iff the extension of $h1$ is a proper superset of the extension of $h2$.

For instance,
aa is more general than $f\diamondsuit$,
$f\heartsuit$ is more general than $q\heartsuit$,
fr and nr are not comparable.

The inverse of the "more general" relation is the "*more specific*" relation.

The "more general" relation defines a partial ordering on the hypotheses in H.

A subset of the partial order for cards

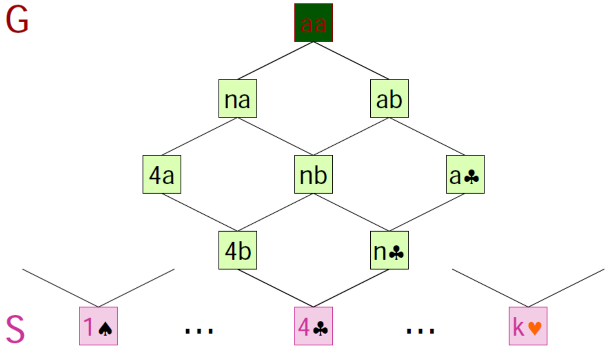


G-Boundary and S-Boundary

Let V be a version space.

- A hypothesis in V is *most general* iff no hypothesis in V is more general.
- *G-boundary* G of V: Set of most general hypotheses in V.
- A hypothesis in V is *most specific* iff no hypothesis in V is more general.
- *S-boundary* S of V: Set of most specific hypotheses in V.

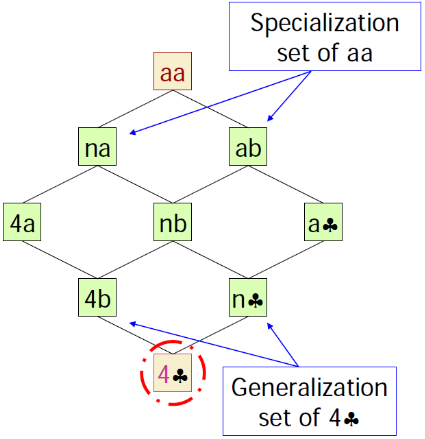
Example: The starting hypothesis space



4♣ is a positive example

We replace every hypothesis in S whose extension does not contain 4♣ by its generalization set

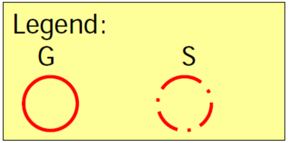
The generalization set of a hypothesis h is the set of the hypotheses that are immediately more general than h

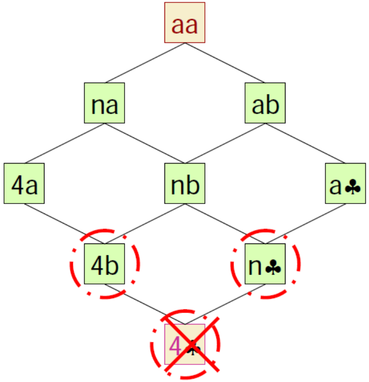


7♣ is the next positive example

Minimally generalize the most specific hypothesis set

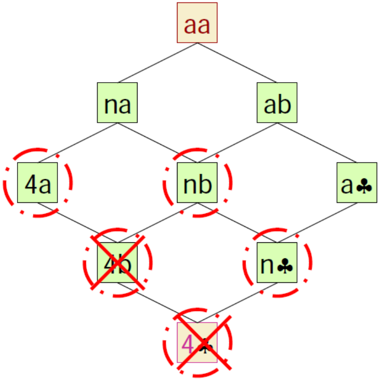
We replace every hypothesis in S whose extension does not contain 7♣ by its generalization set





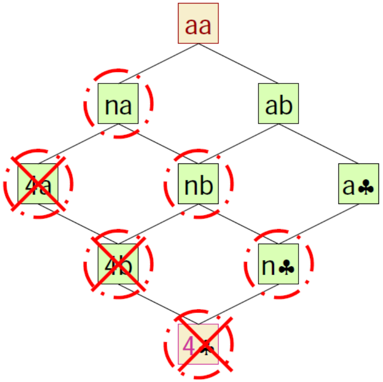
7♣ is the next positive example (cont'd)

Minimally generalize the most specific hypothesis set



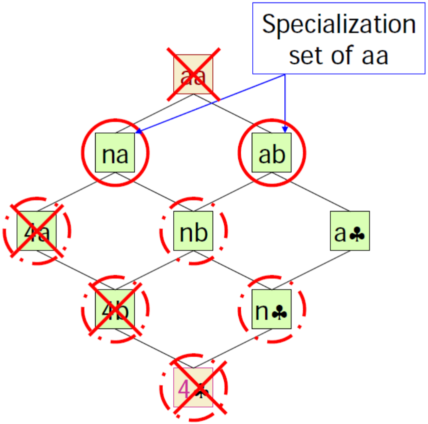
7♣ is the next positive example (cont'd)

Minimally generalize the most specific hypothesis set



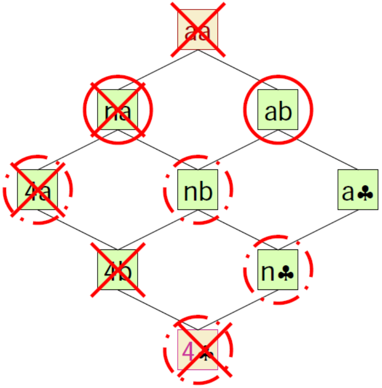
5♡ is a negative example

Minimally specialize the most general hypothesis set

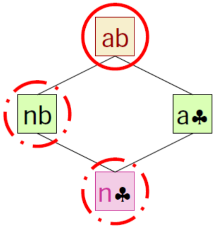


5♡ is a negative example (cont'd)

Minimally specialize the most general hypothesis set



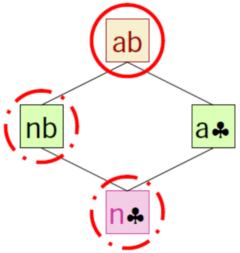
After 3 examples – 2 positive (4♣, 7♣), 1 negative (5♡)



G and S, and all hypotheses in between form the version space.

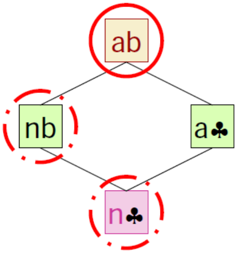
- If a hypothesis between G and S disagrees with an example x, then a hypothesis G or S would also disagree with x, hence would have to be removed.
- If there were a hypothesis not in this set which agreed with all examples, then it would have to be either no more specific than any member of G but then it would be in G or no more general than some member of S but then it would be in S.

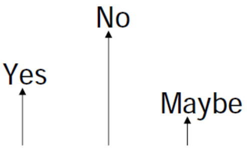
At this stage



Do 8♣, 6♦, j♠ satisfy CONCEPT?

At this stage

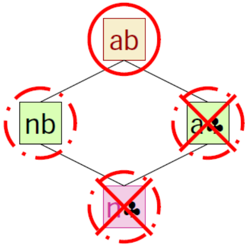




Do 8♣, 6♦, j♠ satisfy CONCEPT?

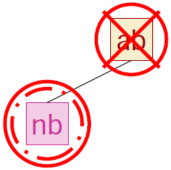
2♠ is the next positive example

Minimally generalize the most specific hypothesis set



$j\spadesuit$ is the next negative example

Minimally specialize the most general hypothesis set



The result

+ 4♣ 7♣ 2♠

– 5♥ j♠

nb

$$(\mathrm{NUM}(r) \wedge \mathrm{BLACK}(s)) \Leftrightarrow \mathrm{REWARD}([r,s])$$

The version space algorithm

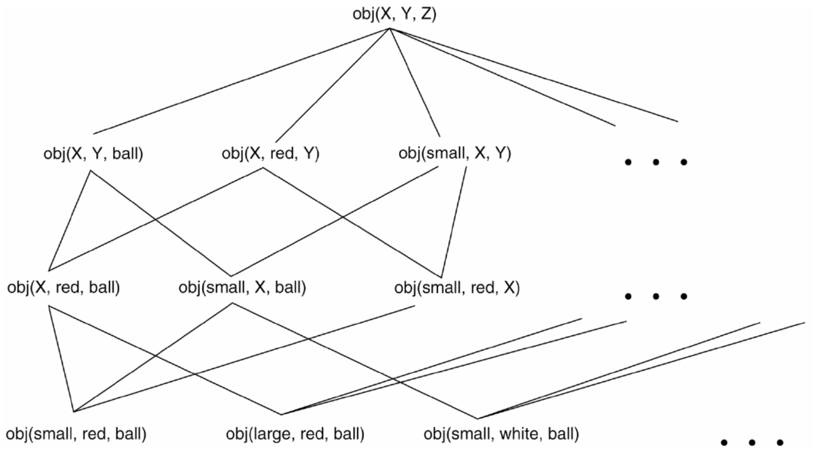
```
function VERSION-SPACE-LEARNING (examples)
returns a version space
  $V \leftarrow$ the set of all hypotheses
  for each example $e$ in examples do
    if $V$ is not empty then
      $V \leftarrow$ VERSION-SPACE-UPDATE($V$,$e$)
  return $V$

function VERSION-SPACE-UPDATE ($V$, $e$)
returns an updated version space
  $V \leftarrow \{\ h \in V\text{: } h$ is consistent with $e\ \}$
```

Another example

- Objects defines by their attributes:
  `object (size, color, shape)`
  - `sizes = {large, small}`
  - `colors = {red, white, blue}`
  - `shapes = {sphere, brick, cube}`
- If the target concept is a “red ball,” then size should not matter, color should be red, and shape should be sphere.
- If the target concept is “ball,” then size or color should not matter, shape should be sphere.

A portion of the concept space



More methods for generalization

- Replacing constants with variables. For example, `color(ball,red)` generalizes to `color(X,red)`.
- Dropping conditions from a conjunctive expression. E.g., `shape(X, round)` $\wedge$ `size(X, small)` $\wedge$ `color(X, red)` generalizes to `shape(X, round)` $\wedge$ `color(X, red)`.
- Adding a disjunct to an expression. For example, `shape(X, round)` $\wedge$ `size(X, small)` $\wedge$ `color (X, red)` generalizes to
  `shape(X, round)` $\wedge$ `size(X, small)` $\wedge$
  `(color(X, red)` $\vee$ `(color(X, blue) )`.
- Replacing a property with its parent in a class hierarchy. If we know that primary-color is a superclass of red, then
  `color(X, red)` generalizes to
  `color(X, primary-color)`.

Learning the concept of a "red ball"

```
G: { obj (X, Y, Z)}
S: {}
```

positive: `obj (small, red, sphere)`

```
G: { obj (X, Y, Z )}
S: { obj (small, red, sphere) }
```

negative: `obj (small, blue, sphere)`

```
G: { obj (large, Y, Z), obj (X, red, Z),
obj (X, white, Z) obj (X,Y, brick), obj (X, Y, cube)}

S: { obj (small, red, sphere) }
```

delete from G every hypothesis that is neither more general than nor equal to a hypothesis in S.

```
G: { obj (X, red, Z) }
S: { obj (small, red, sphere) }
```

Learning the concept of a “red ball” (cont’d)

`G: { obj (X, red, Z) }`
`S: { obj (small, red, sphere) }`

positive: `obj (large, red, sphere)`
`G: { obj (X, red, Z) }`
`S: { obj (X, red, sphere) }`

negative: `obj (large, red, cube)`
`G: { obj (small, red, Z), obj (X, red, sphere), obj (X, red, brick) }`
`S: { obj (X, red, sphere) }`

delete from G every hypothesis that is neither more general than nor equal to a hypothesis in S.
`G: { obj (X, red, sphere) }`
`S: { obj (X, red, sphere) }`
Converged to a single concept.

Comments on version space learning

- It is a bi-directional search. One direction is specific to general and is driven by positive instances. The other direction is general to specific and is driven by negative instances.
- It is an *incremental learning algorithm*. The examples do not have to be given all at once (as opposed to learning decision trees.) The version space is meaningful even before it converges.
- The order of examples matters for the speed of convergence.
- As is, it cannot tolerate noise (misclassified examples), the version space might collapse.
  Can address by maintaining several G and S sets,

Inductive learning

- *Inductive learning* is the process of learning a generalization from a set of examples (*training set*).
- *Concept learning* is a typical inductive learning problem: given examples of some concept, such as cat, soybean disease, or good stock investment, we attempt to infer a definition that will allow the learner to correctly recognize future instances of that *concept*.
- The *concept* is a description of a set where everything inside the set is a *positive examples*, and everything outside the set is a *negative example*.

Supervised learning

- Inductive concept learning is called *supervised learning* because we assume that there is a “teacher” who classified the training data: the learner is told whether an instance is a positive or negative example.
- This definition might seem counter intuitive. If the teacher knows the concept, why doesnt s/he tell us directly and save us all the work?
- Answer: The teacher only knows the classification, the learner has to find out what the classification is.
- Imagine an online store: there is a lot of data concerning whether a customer returns to the store. The information is there in terms of attributes and whether they come back or not. However, it is up to the learning system to characterize the concept, e.g.,
  If a customer bought more than 4 books, s/he will return.
  If a customer spent more than $50, s/he will return.

Summary

- Neural networks, decision trees, and version spaces are examples of *supervised learning*.
- The *hypothesis space* defines what will be learned.

Sources for the slides

- AIMA textbook (3rd edition)
- AIMA slides:
  http://aima.cs.berkeley.edu/
- Luger's AI book (5th edition)
- Jean-Claude Latombe's CS121 slides
  http://robotics.stanford.edu/ latombe/cs121
  (Accessed prior to 2009)