Version Space Learning

Ronald J. Williams
CSG220, Spring 2007

Concept Learning

- Recall that a 2-class classification problem is also called a *concept learning* problem
- Given the mapping f : X → {+, -} to be learned, the set of instances $f^{-1}(+) \subseteq X$ is the corresponding *concept*
- A concept is just a subset of the instance space X, the set of all *positive* instances
- The complement of this set is the set of all *negative* instances
- Important but restrictive assumption: we assume there is no noise in the training examples

First key insight in version space method

- The subsets of any set form a lattice (i.e., partial order) based on the subset/superset relation
- Therefore the hypothesis space for a concept learning problem has a corresponding lattice structure
- For concepts
  - subset → specialization
  - superset → generalization
- If a positive instance is misclassified by a given hypothesis, the hypothesis is too specific
- If a negative instance is misclassified by a given hypothesis, the hypothesis is too general

Concept Description

- In what follows, we'll characterize concepts using *concept descriptions*
- A concept description is an expression in some logical language such that an instance is positive iff it satisfies the expression
- A concept description characterizes a concept (i.e., a subset of the instance space)

Illustrative Example

Suppose instances are vectors of the following attributes:

Attribute	Possible Values
Size	Large, Small
Shape	Triangle, Square, Circle

In this example there are only 6 possible instances

Use an n-tuple notation for each instance (n=2 here)

E.g. (Large, Triangle) is shorthand for
(Size=Large)^(Shape=Triangle)

- The number of possible concepts over this instance space = the number of subsets of a 6-element set = 2^6
- Each of these has many possible (logically equivalent) concept descriptions
- Here are some:
  - Size=Large
  - ~(Size=Small)
  - (Size=Small)^((Shape=Square)v(Shape=Triangle))
- The first 2 are logically equivalent and describe (are satisfied by) 3 of the 6 instances
- The last describes 2 instances

Restricting the hypothesis space

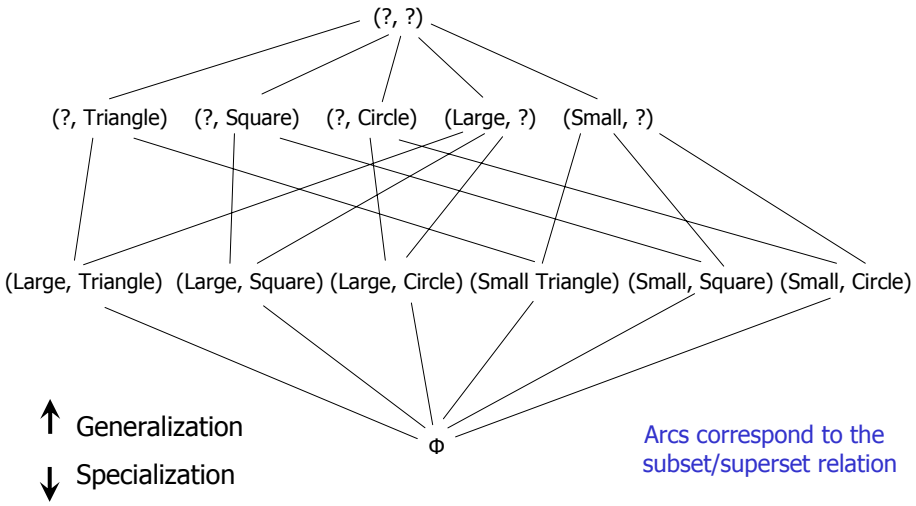
- Have lattice structure for the entire space of all possible concepts over this instance space (= the 64 possible subsets of this instance space)
- Let's assume that our hypothesis space is something smaller
- In particular, restrict attention to *pure conjunctive concepts*
- A pure conjunctive concept is one that has a pure conjunctive concept description
- A pure conjunctive concept description is one whose only logical operators are conjunctions (ANDs)
- No disjunctions (ORs) or negations (NOTs) are allowed

- Examples of pure conjunctive concept descriptions
  - Size=Large
  - (Size=Large)^(Shape=Circle)
- Note that ~(Size=Small) is not a pure conjunctive concept description, but since it's logically equivalent (in this instance space) to Size=Large, the underlying concept (i.e., the subset it characterizes) is a pure conjunctive concept
- Two special cases to note:
  - *true* is pure conjunctive because it can be thought of as a conjunction of no conjuncts – the corresponding concept (subset) is just the entire instance space
  - *false* is not considered pure conjunctive, but it's convenient to add it in – represents the empty set

Notation for pure conjunctive concept descriptions

- Use n-tuple notation similar to that used for instances
- But add wild card symbol ? to indicate a don't-care for the value of that attribute
- E.g.
  - (Large, ?) corresponds to Size=Large
  - (?, ?) corresponds to *true*
  - (Small, Circle) corresponds to (Size=Small)^(Shape=Circle)
- Since we're augmenting the pure conjunctive concept descriptions with *false*, use Φ to denote this additional concept

The entire lattice



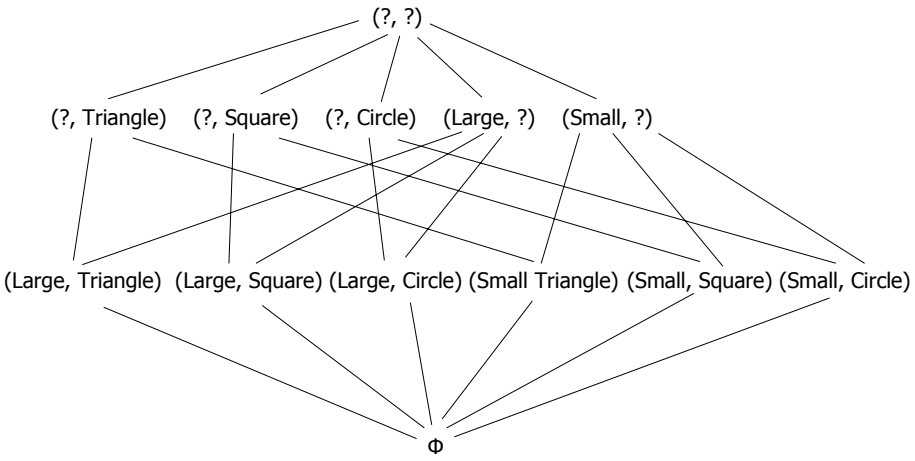
Version Space

- Given a set of training examples, any concept consistent with them must
  - include every positive instance
  - exclude every negative instance
- The set of concepts consistent with a set of training examples is called a *version space* (for that set of examples)
- Version space method involves identifying *all* concepts consistent with a set of training examples
- Can be implemented incrementally, one example at a time

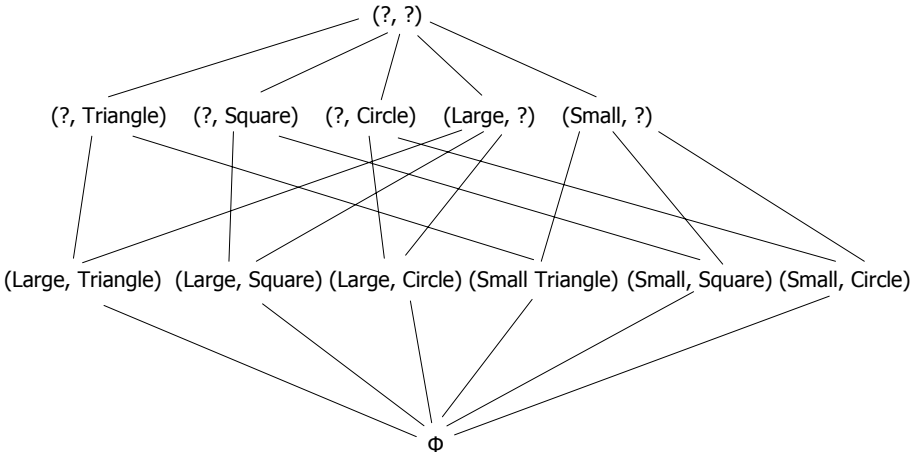
Version space method

- Start with a given set (lattice) of allowable concepts (⇔ selection bias)
- Process the training examples sequentially
- As each example is seen, the set of concepts consistent with all the training data so far is narrowed down
- After seeing enough examples, may converge to a unique concept and learning is complete
- Even if the process hasn't yet converged to a single concept, may still be able to classify some unseen examples reliably

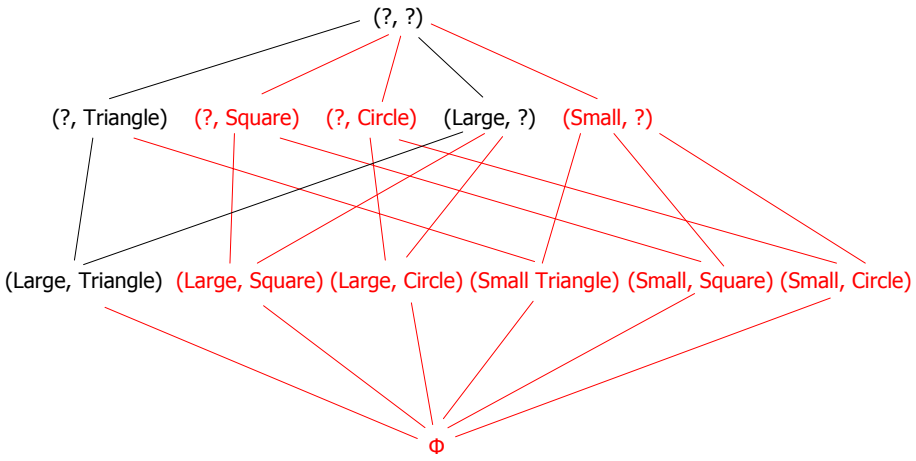
Example 1



1st training example: (Large, Triangle) → +

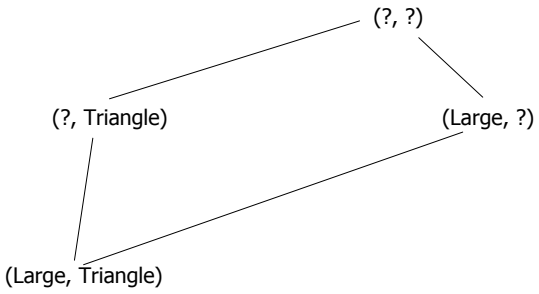


1st training example: (Large, Triangle) → +



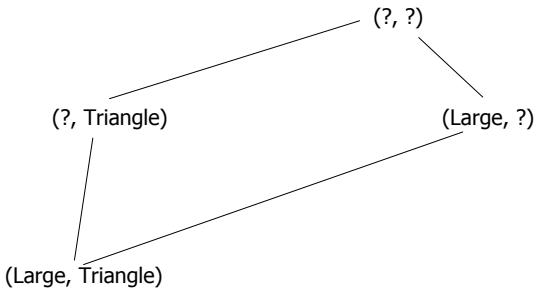
Remove all concepts that do not include (Large, Triangle)

I.e., remove all concept descriptions that (Large, Triangle) does not match

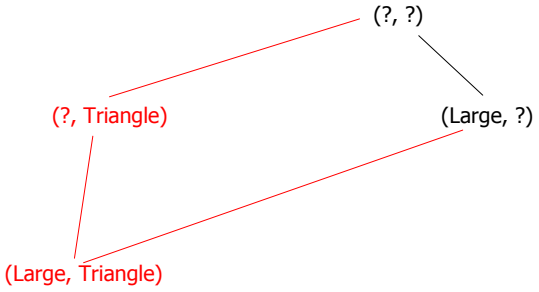


Updated version space after
(Large, Triangle) → +

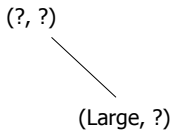
2nd training example: (Large, Circle) → +



2nd training example: (Large, Circle) → +

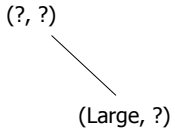


Remove all concept descriptions that (Large, Circle) does not match

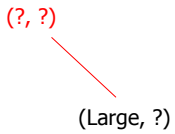


Updated version space after
(Large, Triangle) → +
(Large, Circle) → +

3rd training example: (Small, Circle) → -



3rd training example: (Small, Circle) → -

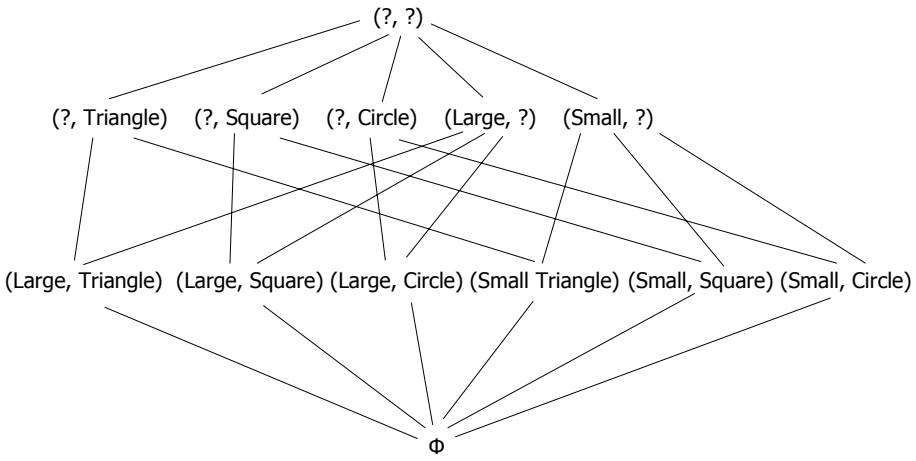


Remove all concept descriptions that (Small, Circle) does match

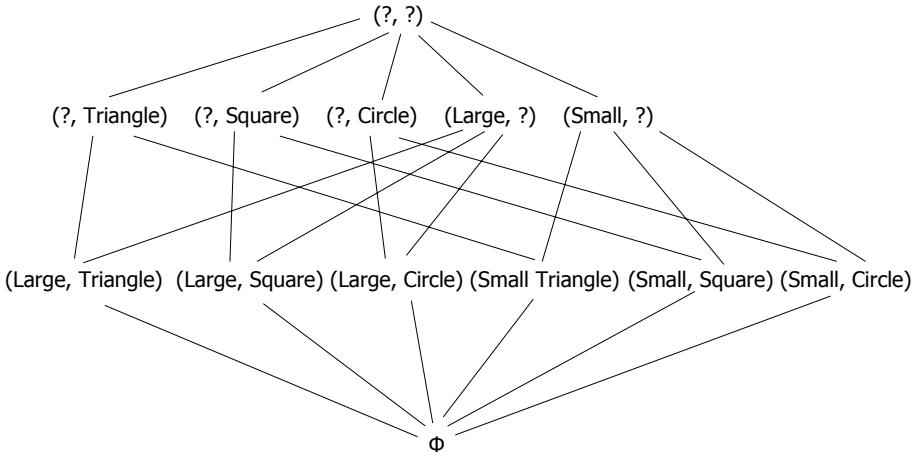
(Large, ?)

Updated version space after
(Large, Triangle) → +
(Large, Circle) → +
(Small, Circle) → -

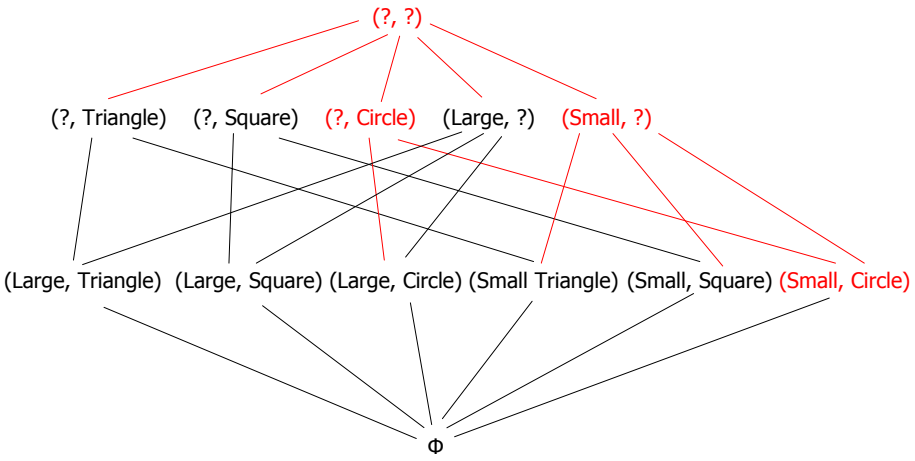
Example 2



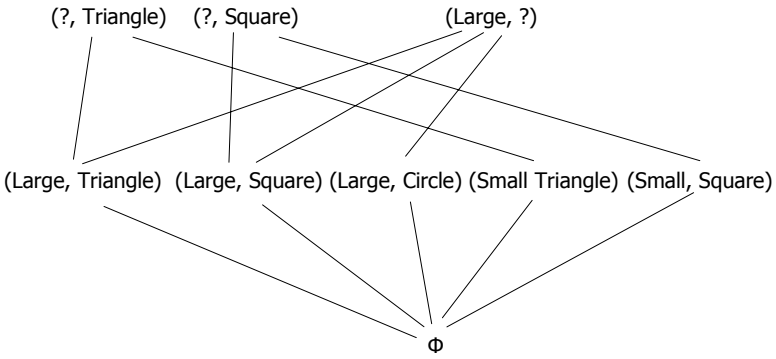
1st training example: (Small, Circle) → -



1st training example: (Small, Circle) → -



Remove all concept descriptions that (Small, Circle) matches



Updated version space

Classifying unseen instances

- Process need not have converged
- If an instance satisfies *all* concept descriptions in the version space, it must be a positive instance
- If an instance satisfies *no* concept description in the version space, it must be a negative instance
- Otherwise, it could be either positive or negative

Second key insight in version space method

- Not necessary to explicitly represent the entire version space
- Let G denote the set of all maximally general concepts in the version space (= maximal elements of the lattice)
- Let S denote the set of all maximally specific concepts in the version space (= minimal elements of the lattice)
- Only represent S and G explicitly

Working with S and G only

- To classify an unseen instance
  - If it matches all concept descriptions in S, it must be a positive instance
  - If it matches no concept descriptions in G, it must be a negative instance
  - Otherwise, it could be either positive or negative
- Interesting facts about pure conjunctive concepts (augmented by Φ) after any number of training examples
  - S always contains exactly one concept
  - G may contain several concepts

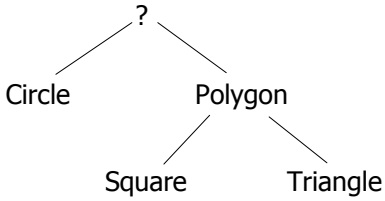
Working with S and G only (cont.)

- Each positive training instance leaves G unchanged, but will require "raising" the S boundary if it does not match all concept descriptions in S
  - positive instances → generalization
- Each negative training instance leaves S unchanged, but will require "lowering" the G boundary if it matches any concept description in G
  - negative instances → specialization
- Actual details of how S and G are updated depend on the hypothesis space used
  - For pure conjunctive hypotheses, updating S is straightforward – always involves replacing a value by ?
  - Updating G for pure conjunctive case only slightly more involved – consult the book

I don't expect you to know this

Extensions

- There could be a pre-specified hierarchy among the values of an attribute
- For example, might want to consider polygons as a natural generalization of squares and triangles, but not circles, like this:



- Could still use pure conjunctive concept descriptions (now allowing Polygon as another possible value for Shape), and such a structure on these values would lead to a finer-grained generalization/specialization lattice for the version space

Extensions (cont.)

- And of course the algorithm is by no means limited to pure conjunctive concepts
- For example, a slightly more expressive language might allow a disjunction of no more than k pure conjunctive expressions for some fixed k
- If arbitrary disjunctions and/or negations are allowed as well, then the true concept could be *any* subset
- In this case
  - exactly one maximally specific concept description, consisting of the disjunction of all the positive training instances
  - exactly one maximally general concept description, consisting of the negation of the disjunction of all the negative training instances
  - maximally general concept treats all unseen instances as positive
  - maximally specific concept treats all unseen instances as negative
- Exactly as hard as trying to learn an arbitrary function from instances to {+, -}

Remarks

- This approach probably of more conceptual than practical interest
- Illustrates how using restrictive hypothesis spaces can lead to very fast learning
- Illustrates one approach to representing all possible hypotheses consistent with training data and how that might be useful
- Shows how hypothesis spaces may have structure that can be exploited