Inductive Classification

Based on the ML lecture by Raymond J. Mooney
University of Texas at Austin

Classification (Categorization)

- Given:
  - A description of an instance, $x \in X$, where X is the ***instance language*** or ***instance space***.
  - A fixed set of categories: $C=\{c_1, c_2, \ldots c_n\}$
- Determine:
  - The category of x: $c(x) \in C$, where $c(x)$ is a categorization function whose domain is X and whose range is C.
  - If $c(x)$ is a binary function $C=\{0,1\}$ ({true,false}, {positive, negative}) then it is called a ***concept***.

Learning for Categorization

- A training example is an instance $x \in X$, paired with its correct category $c(x)$: $<x, c(x)>$ for an unknown categorization function, c.
- Given a set of training examples, D.
- Find a hypothesized categorization function, $h(x)$, such that:

$$\forall < x, c(x) > \in D : h(x) = c(x)$$

Consistency

Sample Category Learning Problem

- Instance language: <size, color, shape>
  - size ∈ {small, medium, large}
  - color ∈ {red, blue, green}
  - shape ∈ {square, circle, triangle}
- C = {positive, negative}
- D:

Example	Size	Color	Shape	Category
1	small	red	circle	positive
2	large	red	circle	positive
3	small	red	triangle	negative
4	large	blue	circle	negative

Hypothesis Selection

- Many hypotheses are usually consistent with the training data.
  - red & circle
  - (small & circle) or (large & red)
  - (small & red & circle) or (large & red & circle)

Generalization

- Hypotheses must generalize to correctly classify instances not in the training data.

- Simply memorizing training examples is a consistent hypothesis that does not generalize. But …

Hypothesis Space

- For learning concepts on instances described by n discrete-valued features, consider the space of conjunctive hypotheses represented by a vector of n constraints

  $<c_1, c_2, \ldots c_n>$ where each c_i is either:
  - ?, a wild card indicating no constraint on the ith feature
  - A specific value from the domain of the ith feature
  - Ø indicating no value is acceptable
- Other notations

  (Size = big) AND (Color = red)

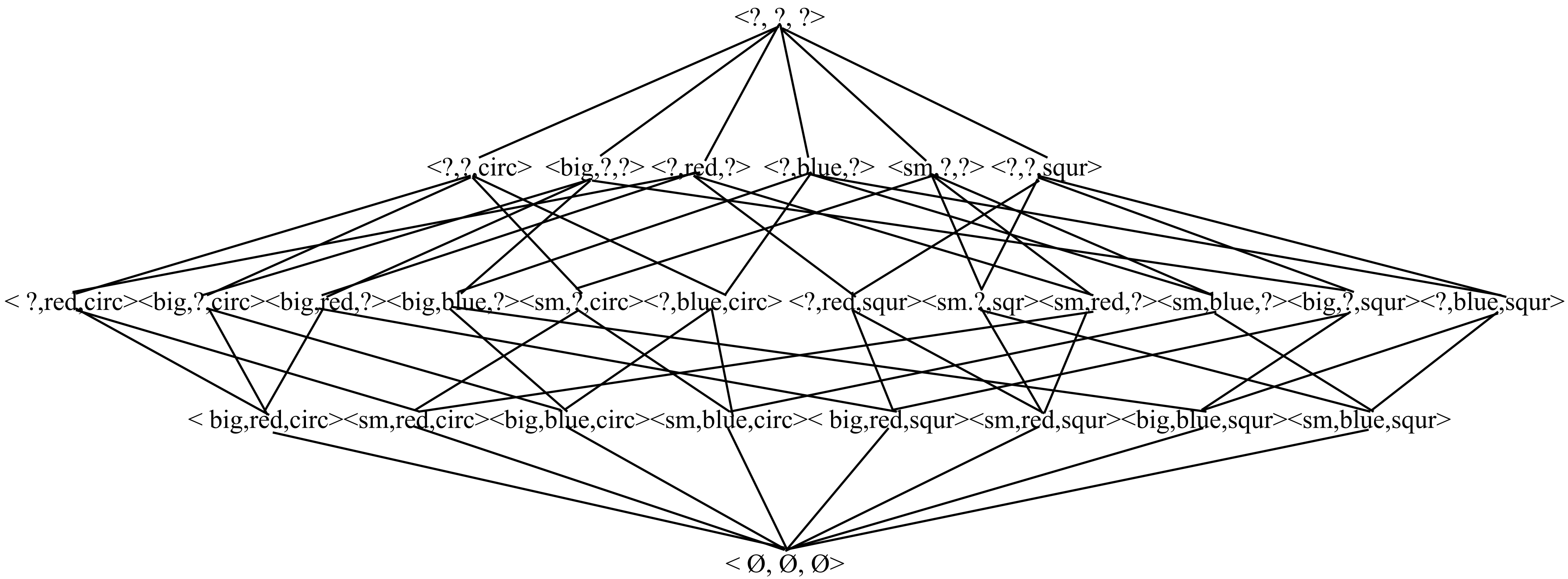
  size(Id, big), color(Id,red)

  size(Id, X), color(Id,Y), shape(Id, Z) . . . In Prolog

- Sample conjunctive hypotheses are
  - <big, red, ?>
  - <?, ?, ?> (most general hypothesis)
  - < Ø, Ø, Ø> (most specific hypothesis)

Sample Hypothesis Space as A Generalization Lattice

Attributes (features): **Size**: {sm, big} **Color**: {red, blue} **Shape**: {circ, squr}



Inductive Learning Hypothesis

- Any function that is found to approximate the target concept well on a sufficiently good (large) set of training examples will also approximate the target function well on unobserved examples.
- Assumes that the training and test examples are drawn independently from the same underlying distribution.
- This is a fundamentally improvable hypothesis unless additional assumptions are made about the target concept and the notion of "approximating the target function well on unobserved examples" is defined appropriately (cf. computational learning theory).

Evaluation of Classification Learning

- Classification accuracy (% of instances classified correctly).
  - Measured on an independent test data.
- Training time (efficiency of training algorithm).
- Complexity of the hypotthesis that has been learned
- Testing time (efficiency of subsequent classification).

Category Learning as Search

- Category learning can be viewed as searching the hypothesis space for one (or more) hypotheses that are consistent with the training data.
- Consider an instance space consisting of n binary features which therefore has 2^n instances.
- For conjunctive hypotheses, there are 4 choices for each feature: Ø, T, F, ?, so there are 4^n syntactically distinct hypotheses.
- However, all hypotheses with 1 or more Øs are equivalent, so there are 3^n+1 semantically distinct hypotheses.

Category Learning as Search (cont.)

- The target binary categorization function in principle could be any of the possible 2^{2^n} functions on n input bits.
- Therefore, conjunctive hypotheses are a small subset of the space of possible functions, but both are intractably large.
- All reasonable hypothesis spaces are intractably large or even infinite.

How to learn?

- Learning in a limit
- Learning by enumeration
- Or … efficient learning?

Learning by Enumeration

- For any finite or countably infinite hypothesis space, one can simply enumerate and test hypotheses one at a time until a consistent one is found.

  For each *h* in *H* do:

  If *h* is consistent with the training data *D*,

  then terminate and return *h*.

- This algorithm is guaranteed to terminate with a consistent hypothesis if one exists; however, it is obviously computationally intractable for almost any practical problem.

Efficient Learning

- Is there a way to learn conjunctive concepts without enumerating them?
- How do human subjects learn conjunctive concepts?
- Is there a way to efficiently find an unconstrained boolean function consistent with a set of discrete-valued training instances?
- If so, is it a useful/practical algorithm?

Conjunctive Rule Learning

- Conjunctive descriptions are easily learned by finding all commonalities shared by all positive examples.

Example	Size	Color	Shape	Category
1	small	red	circle	positive
2	large	red	circle	positive
3	small	red	triangle	negative
4	large	blue	circle	negative

Learned rule: red & circle → positive

- Must check consistency with negative examples. If inconsistent, **no** conjunctive rule exists.

Limitations of Conjunctive Rules

- If a concept does not have a single set of necessary and sufficient conditions, conjunctive learning fails.

Example	Size	Color	Shape	Category
1	small	red	circle	positive
2	large	red	circle	positive
3	small	red	triangle	negative
4	large	blue	circle	negative
5	medium	red	circle	negative

Learned rule: red & circle → positive

Inconsistent with negative example #5!

Disjunctive Concepts

- Concept may be disjunctive.

Example	Size	Color	Shape	Category
1	small	red	circle	positive
2	large	red	circle	positive
3	small	red	triangle	negative
4	large	blue	circle	negative
5	medium	red	circle	negative

Learned rules: (small & circle → positive) OR
(large & red → positive)

Using the Generality Structure

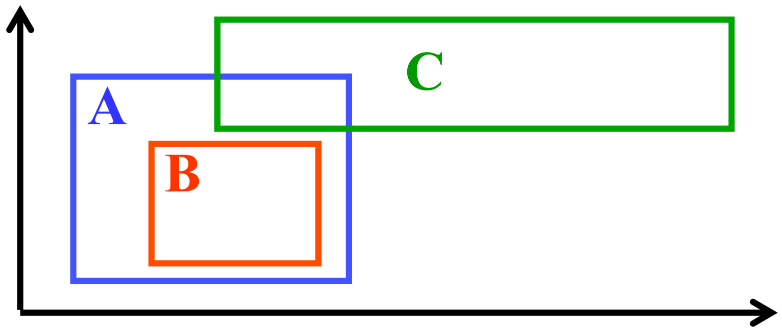
- By exploiting the structure imposed by the generality of hypotheses, an hypothesis space can be searched for consistent hypotheses without enumerating or explicitly exploring all hypotheses.

Using the Generality Structure

- An instance, $x \in X$, is said to ***satisfy*** an hypothesis, h, iff $h(x)=1$ (positive)
- Given two hypotheses h_1 and h_2, h_1 is ***more general than or equal to*** h_2 ($h_1 \geq h_2$) iff every instance that satisfies h_2 also satisfies h_1.
- Given two hypotheses h_1 and h_2, h_1 is **(*strictly*) *more general than*** h_2 ($h_1 > h_2$) iff $h_1 \geq h_2$ and it is not the case that $h_2 \geq h_1$.
- Generality defines a partial order on hypotheses.

Examples of Generality

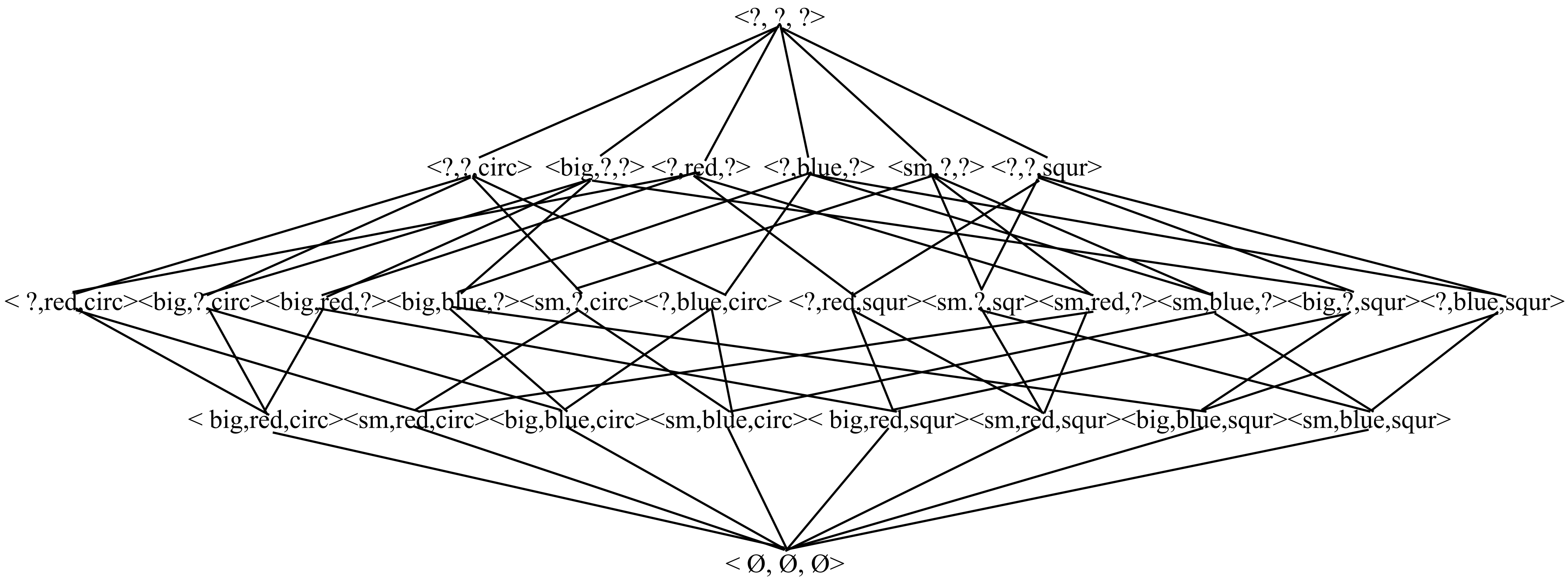
- Conjunctive feature vectors
  - <?, red, ?> is more general than <?, red, circle>
  - Neither of <?, red, ?> and <?, ?, circle> is more general than the other.
- Axis-parallel rectangles in 2-d space



  - A is more general than B
  - Neither of A and C are more general than the other.

Sample Generalization Lattice

Size: {sm, big}   Color: {red, blue}   Shape: {circ, squr}



Number of hypotheses = $3^3 + 1 = 28$

Version Space

- Given an hypothesis space, *H*, and training data, *D*, the ***version space*** is the complete subset of *H* that is consistent with *D*.
- The version space can be naively generated for any finite *H* by enumerating all hypotheses and eliminating the inconsistent ones.
- Can one compute the version space more efficiently than using enumeration?

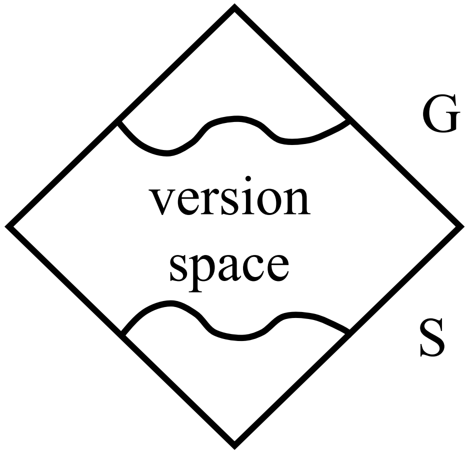
Version Space with S and G

- The version space can be represented more compactly by maintaining two boundary sets of hypotheses, *S*, the set of most specific consistent hypotheses, and *G*, the set of most general consistent hypotheses:

$$S = \{s \in H \mid Consistent(s, D) \wedge \neg\exists s' \in H[s > s' \wedge Consistent(s', D)]\}$$

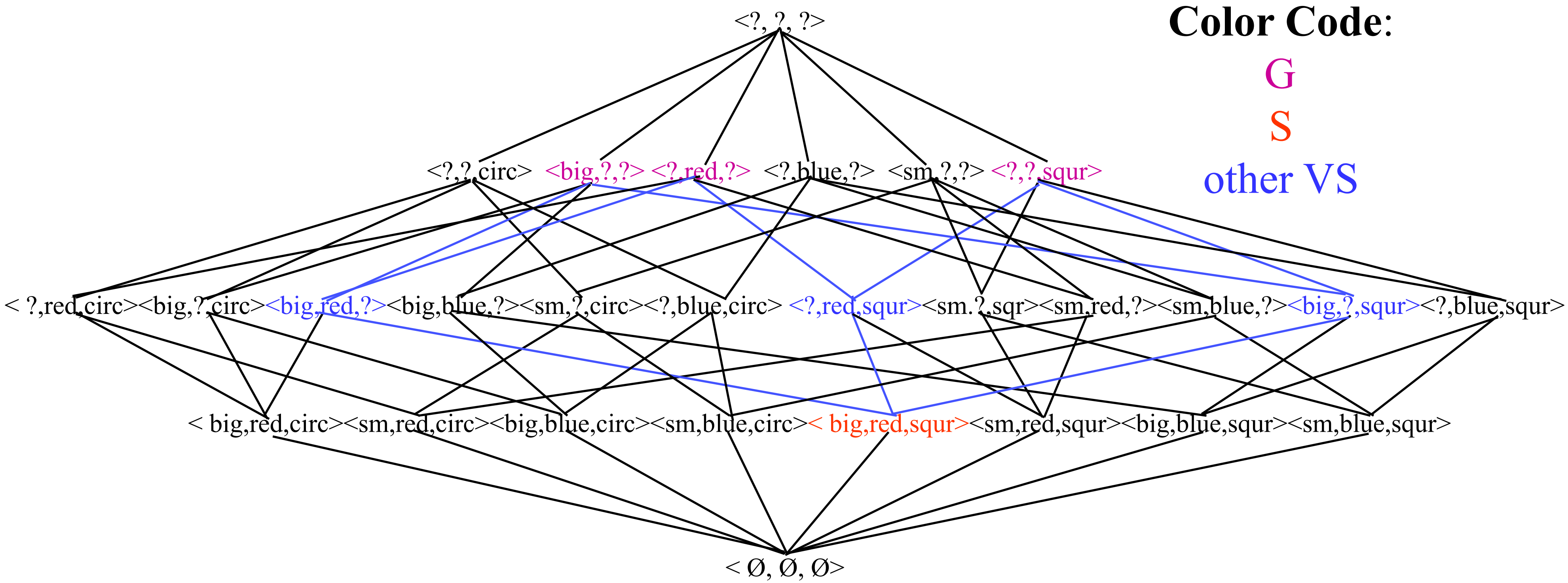
$$G = \{g \in H \mid Consistent(g, D) \wedge \neg\exists g' \in H[g' > g \wedge Consistent(s', D)]\}$$

- S and G represent the entire version space via its boundaries in the generalization lattice:



Version Space Lattice

Size: {sm, big}   Color: {red, blue}   Shape: {circ, squr}



<<big, red, squr> positive>
<<sm, blue, circ> negative>

Candidate Elimination (Version Space) Algorithm

Initialize *G* to the set of most-general hypotheses in *H*
Initialize *S* to the set of most-specific hypotheses in *H*
For each training example, *d*, do:

- If *d* is a positive example then:
  - Remove from *G* any hypotheses that do not match *d*
  - For each hypothesis *s* in *S* that does not match *d*
    - Remove *s* from *S*
    - Add to *S* all minimal generalizations, *h*, of *s* such that:
      1) *h* matches *d*
      2) some member of *G* is more general than *h*
  - Remove from *S* any *h* that is more general than another hypothesis in *S*
- If *d* is a negative example then:
  - Remove from *S* any hypotheses that match *d*
  - For each hypothesis *g* in *G* that matches *d*
    - Remove *g* from *G*
    - Add to *G* all minimal specializations, *h*, of *g* such that:
      1) *h* does not match *d*
      2) some member of *S* is more specific than *h*
  - Remove from *G* any *h* that is more specific than another hypothesis in *G*

Properties of VS Algorithm

- *S* summarizes the relevant information in the positive examples (relative to *H*) so that positive examples do not need to be retained.
- *G* summarizes the relevant information in the negative examples, so that negative examples do not need to be retained.
- Result is not affected by the order in which examples are processes but computational efficiency may.
- Positive examples move the *S* boundary up; Negative examples move the *G* boundary down.
- If *S* and *G* converge to the same hypothesis, then it is the only one in *H* that is consistent with the data.
- If *S* and *G* become empty (if one does the other must also) then there is no hypothesis in *H* consistent with the data.

Correctness of Learning

- Since the entire version space is maintained, given a continuous stream of noise-free training examples, the VS algorithm will eventually converge to the correct target concept if it is in the hypothesis space, *H*, or eventually correctly determine that it is not in *H*.
- Convergence is correctly indicated when S=G.

Computational Complexity of VS

- Computing the *S* set for conjunctive feature vectors is linear in the number of features and the number of training examples.
- Computing the *G* set for conjunctive feature vectors is exponential in the number of training examples in the worst case.
- In more expressive languages, both *S* and *G* can grow exponentially.
- The order in which examples are processed can significantly affect computational complexity.

No Panacea

- No Free Lunch (NFL) Theorem (Wolpert, 1995)
  Law of Conservation of Generalization Performance (Schaffer, 1994)
  - One can prove that improving generalization performance on unseen data for some tasks will always decrease performance on other tasks (which require different labels on the unseen instances).
  - Averaged across all possible target functions, no learner generalizes to unseen data any better than any other learner.
- There does not exist a learning method that is uniformly better than another for all problems.
- Given any two learning methods A and B and a training set, D, there always exists a target function for which A generalizes better (or at least as well) as B.
  - Train both methods on D to produce hypotheses h_A and h_B.
  - Construct a target function that labels all unseen instances according to the predictions of h_A.
  - Test h_A and h_B on any unseen test data for this target function and conclude that h_A is better.

Logical View of Induction

- Deduction is inferring sound specific conclusions from general rules (axioms) and specific facts.
- Induction is inferring general rules and theories from specific empirical data.
- Induction can be viewed as inverse deduction.
  - Find a hypothesis h from data D such that
    - $h \cup B \vdash D$
      where B is optional background knowledge
- ***Abduction*** is similar to induction, except it involves finding a specific hypothesis, h, that best *explains* a set of evidence, D, or inferring cause from effect. Typically, in this case B is quite large compared to induction and h is smaller and more specific to a particular event.

Induction and the Philosophy of Science

- Bacon (1561-1626), Newton (1643-1727) and the sound deductive derivation of knowledge from data.
- Hume (1711-1776) and the ***problem of induction***.
  - Inductive inferences can never be proven and are always subject to disconfirmation.
- Popper (1902-1994) and ***falsifiability***.
  - Inductive hypotheses can only be falsified not proven, so pick hypotheses that are most subject to being falsified.
- Kuhn (1922-1996) and ***paradigm shifts***.
  - Falsification is insufficient, an alternative paradigm must be available that is clearly elegant and more explanatory.
    - Ptolmaic epicycles and the Copernican revolution
    - Orbit of Mercury and general relativity
    - Solar neutrino problem and neutrinos with mass
- Postmodernism: Objective truth does not exist; relativism; science is a social system of beliefs that is no more valid than others (e.g. religion).

Ockham (Occam)'s Razor

- William of Ockham (1295-1349) was a Franciscan friar who applied the criteria to theology:
  - "Entities should not be multiplied beyond necessity" (Classical version but not an actual quote)
  - "The supreme goal of all theory is to make the irreducible basic elements as simple and as few as possible without having to surrender the adequate representation of a single datum of experience." (Einstein)
- Requires a precise definition of simplicity.
- Acts as a bias which assumes that nature itself is simple.
- Role of Occam's razor in machine learning remains controversial.