Introduction to Logic:
Inductive Classification

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

Sample Category Learning Problem

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

Sample Category Learning Problem

- we will work with a prolog notation, Horn clauses

- examples = ternary predicate category(Size, Color, Shape)
- positive:

  category(small, red, circle). category(large, red, circle).
- negative:

  category(small, red, triangle). category(large, blue, circle).

- hypothesis

  category(Size, Color, Shape) :- …, Var_i = value_i , …

Hypothesis Selection

- Many hypotheses are usually consistent with the training data.
  - red & circle

Hypothesis Selection

- Many hypotheses are usually consistent with the training data.
  - red & circle

    category(Size, Color, Shape) :- Color=red, Shape=circle

Hypothesis Selection

- Many hypotheses are usually consistent with the training data.
  - red & circle

    category(Size, Color, Shape) :- Color=red, Shape=circle
  - (small & circle) or (large & red)

Hypothesis Selection

- Many hypotheses are usually consistent with the training data.
  - red & circle

    category(Size, Color, Shape) :- Color=red, Shape=circle
  - (small & circle) or (large & red)

    category(Size, Color, Shape) :- Size=small, Shape=circle.

Hypothesis Selection

- Many hypotheses are usually consistent with the training data.
  - red & circle

    category(Size, Color, Shape) :- Color=red, Shape=circle
  - (small & circle) or (large & red)

    category(Size, Color, Shape) :- Size=small, Shape=circle.

    category(Size, Color, Shape) :- Size=large, Color=red.

Hypothesis Selection

- Many hypotheses are usually consistent with the training data.
  - red & circle

    category(Size, Color, Shape) :- Color=red, Shape=circle
  - (small & circle) or (large & red)

    category(Size, Color, Shape) :- Size=small, Shape=circle.

    category(Size, Color, Shape) :- Size=large, Color=red.
  - (small & red & circle) or (large & red & circle)

Bias

- any criteria other than consistency with the training data that is used to select a hypothesis.

  A hypothesis space that does not include all possible classification functions on the instance space incorporates a bias in the type of classifiers it can learn.

- search bias, termination condition
- language bias

  maximal length of clauses

  maximal number of clauses

  max. number of attribute repetition (e.g. for numeric attributes)

Inductive Bias

- Any means that a learning system uses to choose between two functions that are both consistent with the training data is called ***inductive bias***.
- Inductive bias can take two forms:
  - ***Language bias***: The language for representing concepts defines a hypothesis space that does not include all possible functions (e.g. conjunctive descriptions).
  - ***Search bias***: The language is expressive enough to represent all possible functions (e.g. disjunctive normal form) but the search algorithm embodies a preference for certain consistent functions over others (e.g. syntactic simplicity). Includes also ***termination condition.***

Examples of bias

- search bias

  search heuristics
- termination condition

  number of iterations

  consistency condition, or weaker: a number of negative examples covered
- language bias

  maximal length of clauses

  maximal number of clauses

  max. number of attribute repetition (e.g. for numeric attributes)

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 …
- *Occam's razor*:
  - Finding a *simple* hypothesis helps ensure generalization.

Hypothesis Space

- Restrict learned functions a priori to a given ***hypothesis space***, H, of functions $h(x)$ that can be considered as definitions of $c(x)$.
- 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:
  - X, a variable indicating no constraint on the ith feature
  - A specific value from the domain of the ith feature
  - Ø indicating no value is acceptable
- Sample conjunctive hypotheses are
  - <big, red, Z>
  - <X, Y, Z> (most general hypothesis)
  - < Ø, Ø, Ø> (most specific hypothesis)

Inductive Learning Hypothesis

- Any function that is found to approximate the target concept well on a sufficiently 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 unprovable 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).

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, X, 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.
- The target binary categorization function in principle could be any of the possible $2^{2\wedge n}$ 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.

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: category(Size,Color,Shape):-
Color=red, Shape=circle.

- 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

category(Size,Color,Shape):- Color=red, Shape=circle.

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:

category(Size,Color,Shape):- Size=small, Shape=circle.

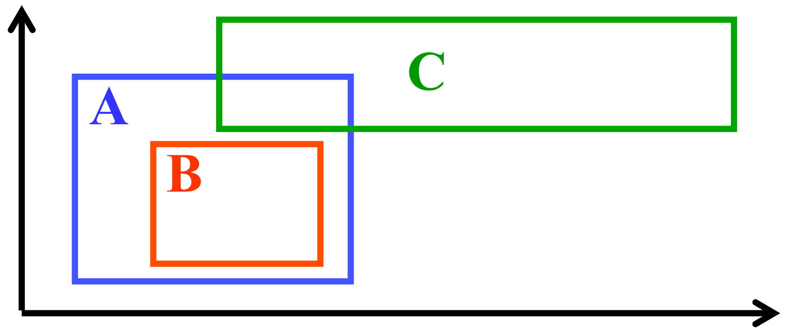
category(Size,Color,Shape):- Size=large, Color=red.

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.
- 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
  - <X, red, Z> is more general than <X, red, circle>
  - Neither of <X, red, Z> and <X, Y, 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: $X \in \{sm, big\}$ Color: $Y \in \{red, blue\}$ Shape: $Z \in \{circ, squr\}$

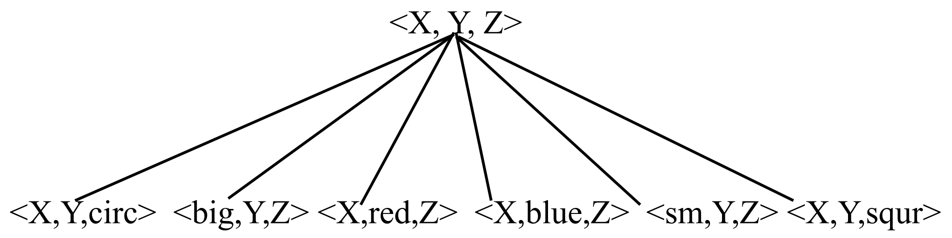
Sample Generalization Lattice

Size: X ∈ {sm, big} Color: Y ∈ {red, blue} Shape: Z ∈ {circ, squr}

<X, Y, Z>

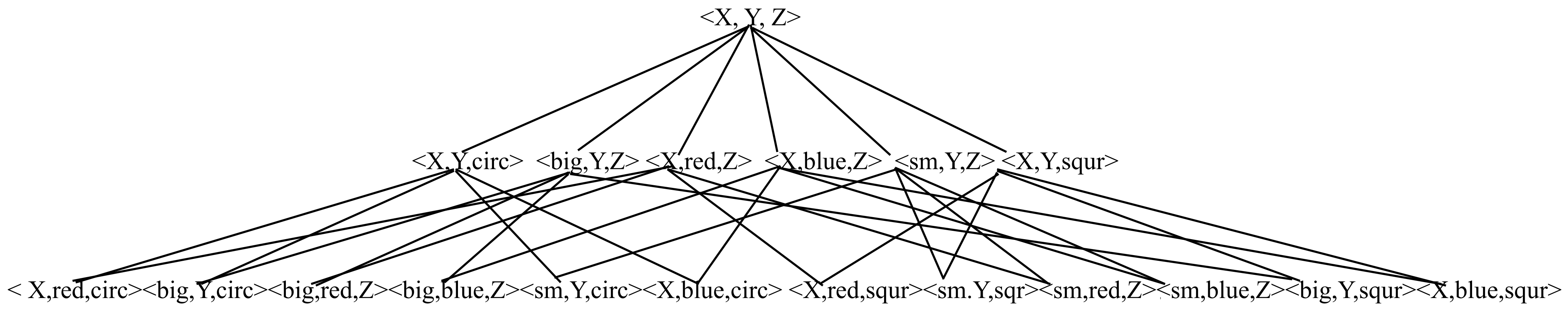
Sample Generalization Lattice

Size: $X \in$ {sm, big}   Color: $Y \in$ {red, blue}   Shape: $Z \in$ {circ, squr}



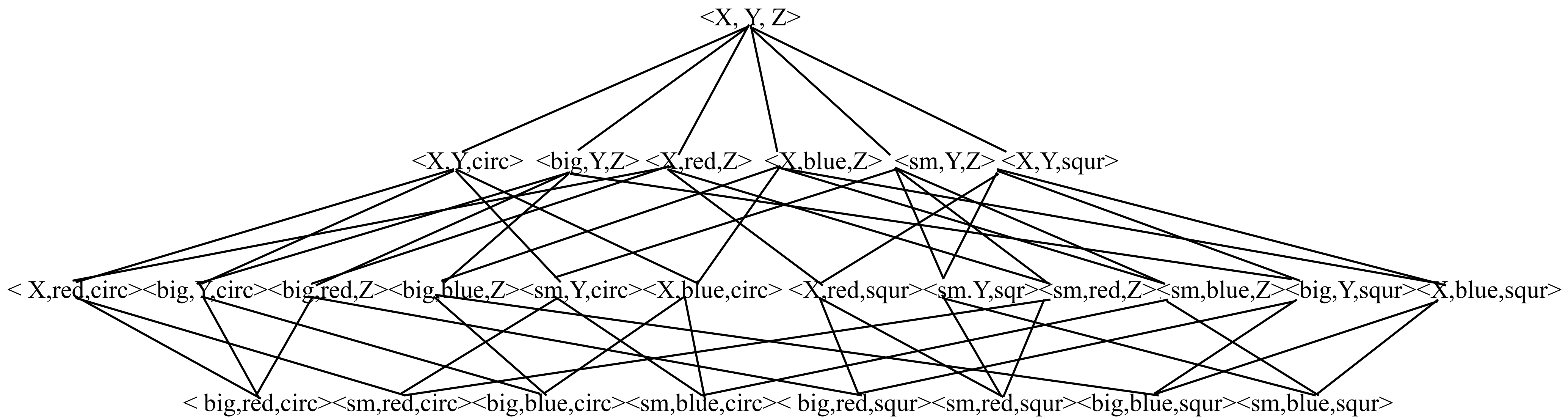
Sample Generalization Lattice

Size: X ∈ {sm, big}   Color: Y ∈ {red, blue}   Shape: Z ∈ {circ, squr}



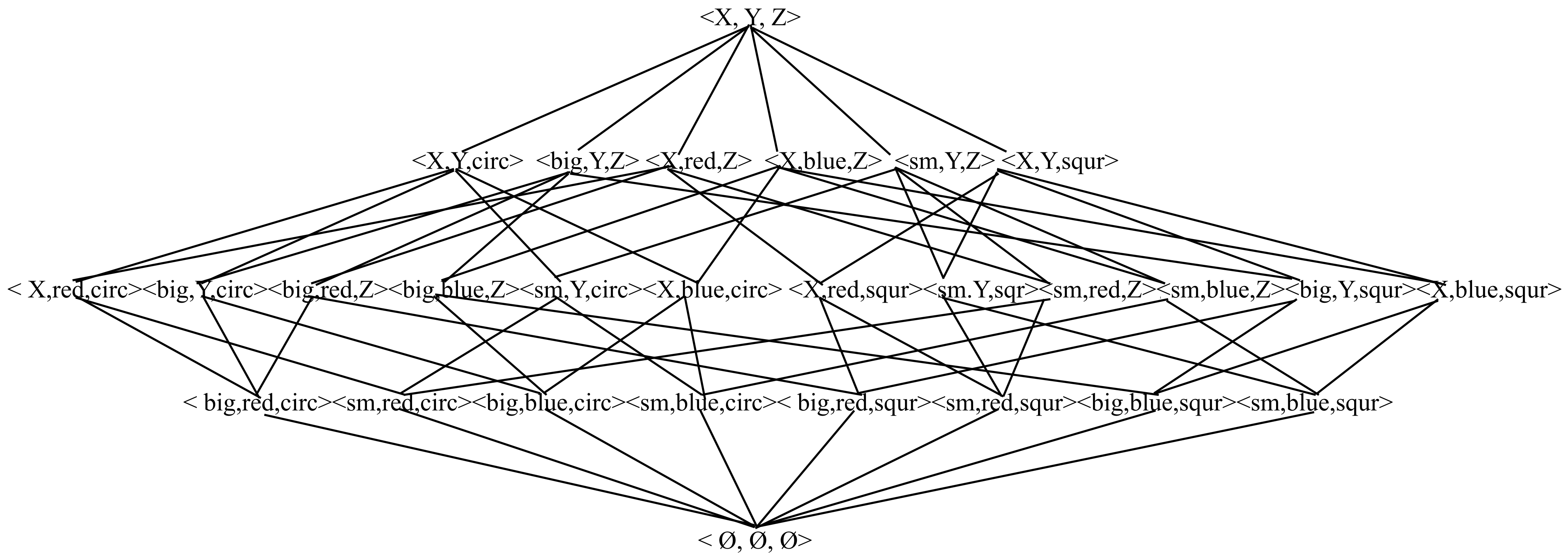
Sample Generalization Lattice

Size: X ∈ {sm, big}   Color: Y ∈ {red, blue}   Shape: Z ∈ {circ, squr}



Sample Generalization Lattice

Size: X ∈ {sm, big}   Color: Y ∈ {red, blue}   Shape: Z ∈ {circ, squr}



<X, Y, Z>
<X,Y,circ> <big,Y,Z> <X,red,Z> <X,blue,Z> <sm,Y,Z> <X,Y,squr>
< X,red,circ><big,Y,circ><big,red,Z><big,blue,Z><sm,Y,circ><X,blue,circ> <X,red,squr><sm.Y,sqr><sm,red,Z><sm,blue,Z><big,Y,squr><X,blue,squr>
< big,red,circ><sm,red,circ><big,blue,circ><sm,blue,circ>< big,red,squr><sm,red,squr><big,blue,squr><sm,blue,squr>
< Ø, Ø, Ø>

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

Minimal Specialization and Generalization

- Procedures generalize-to and specialize-against are specific to a hypothesis language and can be complex.
- For conjunctive feature vectors:
  - generalize-to: unique
  - specialize-against: not unique, can convert each VARIABLE to an alernative non-matching value for this feature.
    - Inputs:
      - *h* = <X, red, Z>
      - *i* = <small, red, triangle>
    - Outputs:
      - <big, red, Z>
      - <medium, red, Z>
      - <X, red, square>
      - <X, red, circle>

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*.

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

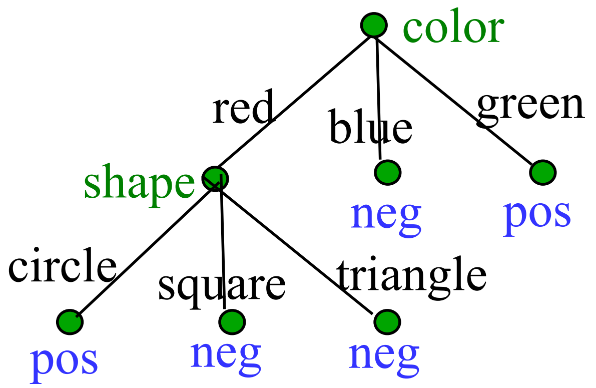
- Bacon (1561-1626), Newton (1643-1727) and the sound deductive derivation of knowledge from data.
- Hume (1711-1776) and the ***problem of induction***.
  - Inductive inferences can never be proven and are always subject to disconfirmation.
- Popper (1902-1994) and ***falsifiability***.
  - Inductive 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 that is clearly elegant and more explanatory must be available.
    - 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).

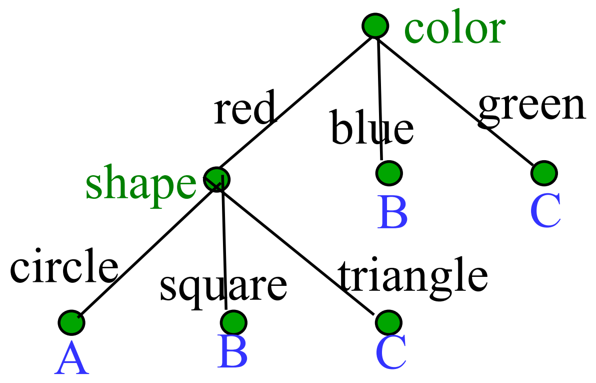
Ockham (Occam)’s Razor

- 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.

Decision Trees

- Tree-based classifiers for instances represented as feature-vectors. Nodes test features, there is one branch for each value of the feature, and leaves specify the category.





- Can represent arbitrary conjunction and disjunction. Can represent any classification function over discrete feature vectors.
- Can be rewritten as a set of rules, i.e. disjunctive normal form (DNF).
  - red $\wedge$ circle $\rightarrow$ pos
  - red $\wedge$ circle $\rightarrow$ A

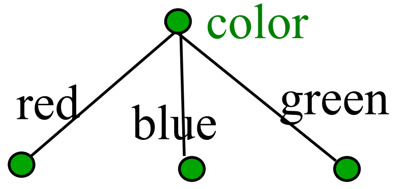
    blue $\rightarrow$ B; red $\wedge$ square $\rightarrow$ B

    green $\rightarrow$ C; red $\wedge$ triangle $\rightarrow$ C

Top-Down Decision Tree Induction

- Recursively build a tree top-down by divide and conquer.

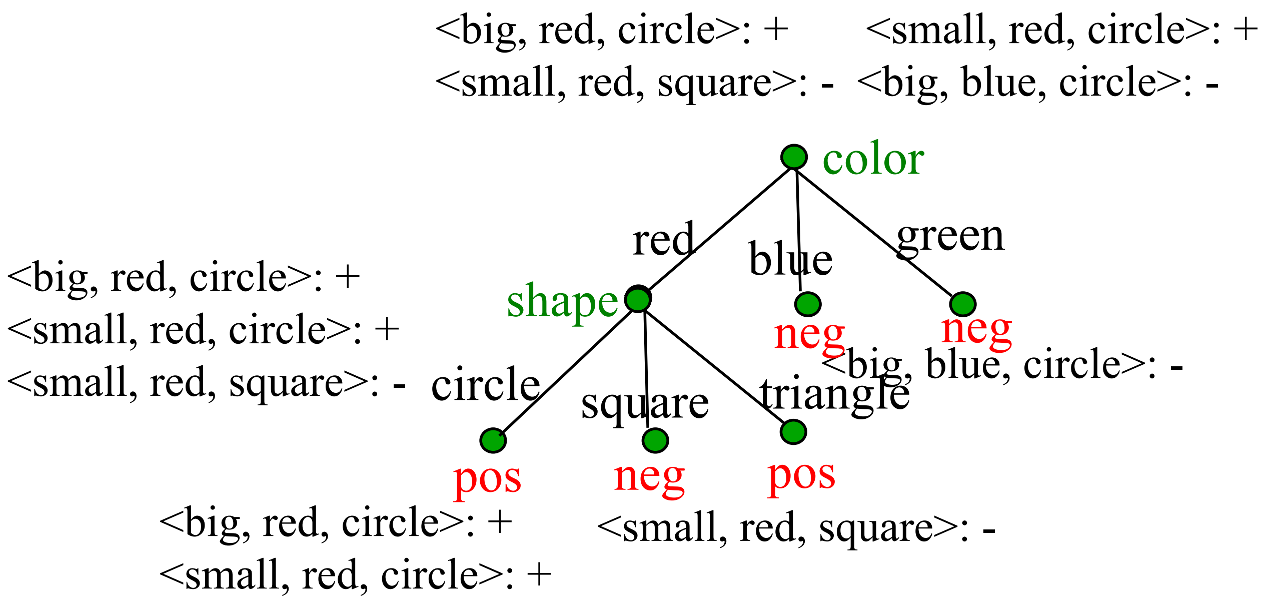
<big, red, circle>: + <small, red, circle>: +
<small, red, square>: - <big, blue, circle>: -



<big, red, circle>: +
<small, red, circle>: +
<small, red, square>: -

Top-Down Decision Tree Induction

- Recursively build a tree top-down by divide and conquer.



Decision Tree Induction Pseudocode

```
DTree($examples$, $features$) returns a tree
 If all $examples$ are in one category, return a leaf node with that category label.
 Else if the set of $features$ is empty, return a leaf node with the category label that
        is the most common in examples.
 Else pick a feature $F$ and create a node $R$ for it
     For each possible value $v_i$ of $F$:
         Let $examples_i$ be the subset of examples that have value $v_i$ for $F$
         Add an out-going edge $E$ to node $R$ labeled with the value $v_{i.}$
          If $examples_i$ is empty
              then attach a leaf node to edge $E$ labeled with the category that
                    is the most common in $examples$.
              else call DTree($examples_i$ , $features - \{F\}$) and attach the resulting
                   tree as the subtree under edge $E$.
     Return the subtree rooted at $R$.
```

Picking a Good Split Feature

- Goal is to have the resulting tree be as small as possible, per Occam's razor.
- Finding a minimal decision tree (nodes, leaves, or depth) is an NP-hard optimization problem.
- Top-down divide-and-conquer method does a greedy search for a simple tree but does not guarantee to find the smallest.
  - General lesson in ML: "Greed is good."
- Want to pick a feature that creates subsets of examples that are relatively "pure" in a single class so they are "closer" to being leaf nodes.
- There are a variety of heuristics for picking a good test, a popular one is based on information gain that originated with the ID3 system of Quinlan (1979).

Entropy

- Entropy (disorder, impurity) of a set of examples, S, relative to a binary classification is:

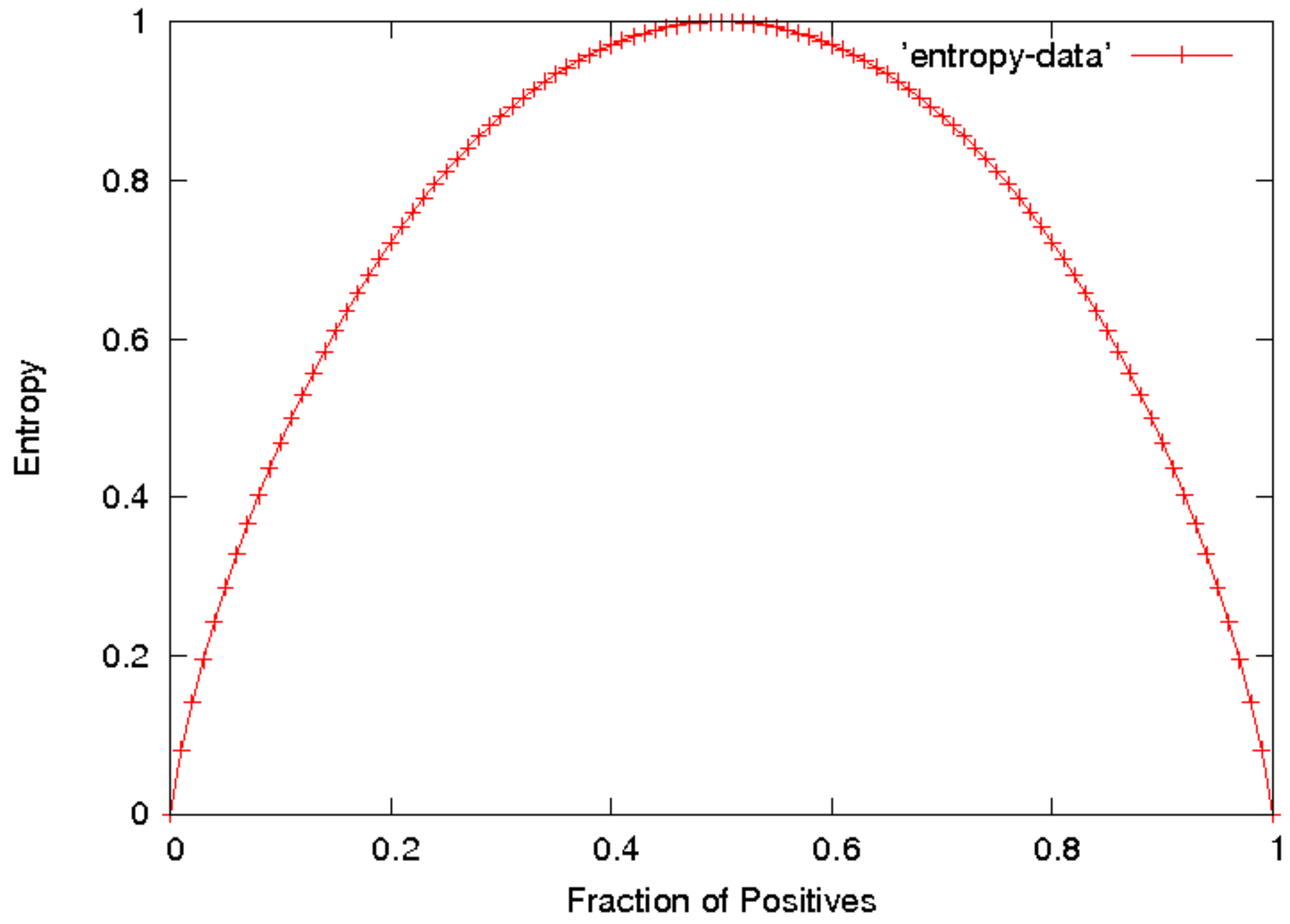
$$Entropy(S) = -p_1 \log_2(p_1) - p_0 \log_2(p_0)$$

where p_1 is the fraction of positive examples in S and p_0 is the fraction of negatives.

- If all examples are in one category, entropy is zero (we define $0 \cdot \log(0)=0$)
- If examples are equally mixed ($p_1=p_0=0.5$), entropy is a maximum of 1.
- Entropy can be viewed as the number of bits required on average to encode the class of an example in *S* where data compression (e.g. Huffman coding) is used to give shorter codes to more likely cases.
- For multi-class problems with c categories, entropy generalizes to:

$$Entropy(S) = \sum_{i=1}^{c} -p_i \log_2(p_i)$$

Entropy Plot for Binary Classification



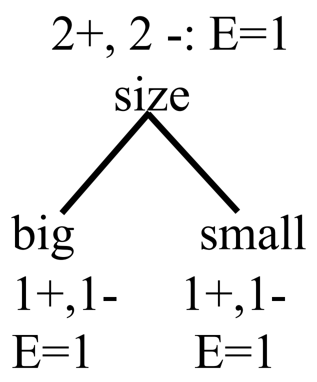
Information Gain

- The information gain of a feature *F* is the expected reduction in entropy resulting from splitting on this feature.

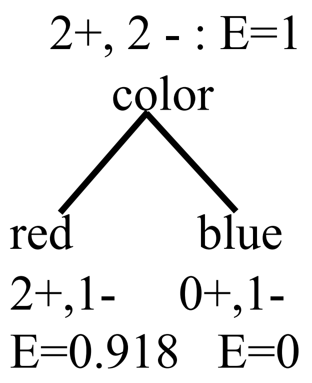
$$Gain(S,F) = Entropy(S) - \sum_{v \in Values(F)} \frac{|S_v|}{|S|} Entropy(S_v)$$

where S_v is the subset of *S* having value *v* for feature *F*.

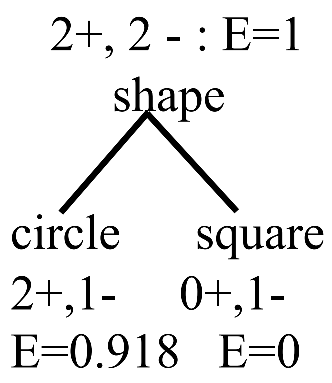
- Entropy of each resulting subset weighted by its relative size.
- Example:
  - <big, red, circle>: +   <small, red, circle>: +
  - <small, red, square>: -   <big, blue, circle>: -



Gain=1-(0.5·1 + 0.5·1) = 0



Gain=1-(0.75·0.918 + 0.25·0) = 0.311



Gain=1-(0.75·0.918 + 0.25·0) = 0.311

Bayesian Categorization

- Determine category of x_k by determining for each y_i

$$P(Y = y_i \mid X = x_k) = \frac{P(Y = y_i)P(X = x_k \mid Y = y_i)}{P(X = x_k)}$$

- P(X=x_k) can be determined since categories are complete and disjoint.

$$\sum_{i=1}^{m} P(Y = y_i \mid X = x_k) = \sum_{i=1}^{m} \frac{P(Y = y_i)P(X = x_k \mid Y = y_i)}{P(X = x_k)} = 1$$

$$P(X = x_k) = \sum_{i=1}^{m} P(Y = y_i)P(X = x_k \mid Y = y_i)$$

Bayesian Categorization (cont.)

- Need to know:
  - Priors: $P(Y=y_i)$
  - Conditionals: $P(X=x_k \mid Y=y_i)$
- $P(Y=y_i)$ are easily estimated from data.
  - If n_i of the examples in D are in y_i then $P(Y=y_i) = n_i / |D|$
- Too many possible instances (e.g. 2^n for binary features) to estimate all $P(X=x_k \mid Y=y_i)$.
- Still need to make some sort of independence assumptions about the features to make learning tractable.

Naïve Bayesian Categorization

- If we assume features of an instance are independent **given the category** (***conditionally independent***).

$$P(X \mid Y) = P(X_1, X_2, \ldots X_n \mid Y) = \prod_{i=1}^{n} P(X_i \mid Y)$$

- Therefore, we then only need to know P(X_i | Y) for each possible pair of a feature-value and a category.
- If Y and all X_i and binary, this requires specifying only 2n parameters:
  - P(X_i=true | Y=true) and P(X_i=true | Y=false) for each X_i
  - P(X_i=false | Y) = 1 – P(X_i=true | Y)

- Compared to specifying 2^n parameters without any independence assumptions.

Naïve Bayes Example

Probability	positive	negative
P(Y)	0.5	0.5
P(small | Y)	0.4	0.4
P(medium | Y)	0.1	0.2
P(large | Y)	0.5	0.4
P(red | Y)	0.9	0.3
P(blue | Y)	0.05	0.3
P(green | Y)	0.05	0.4
P(square | Y)	0.05	0.4
P(triangle | Y)	0.05	0.3
P(circle | Y)	0.9	0.3

Test Instance:
<medium ,red, circle>

Naïve Bayes Example

Probability	positive	negative
$P(Y)$	0.5	0.5
$P(\text{medium} \mid Y)$	0.1	0.2
$P(\text{red} \mid Y)$	0.9	0.3
$P(\text{circle} \mid Y)$	0.9	0.3

Test Instance:
<medium ,red, circle>

$$P(\text{positive} \mid X) = \underset{0.5}{P(\text{positive})} * \underset{0.1}{P(\text{medium} \mid \text{positive})} * \underset{0.9}{P(\text{red} \mid \text{positive})} * \underset{0.9}{P(\text{circle} \mid \text{positive})} / P(X)$$

$$= 0.0405 / P(X) = 0.0405 / 0.0495 = 0.8181$$

$$P(\text{negative} \mid X) = \underset{0.5}{P(\text{negative})} * \underset{0.2}{P(\text{medium} \mid \text{negative})} * \underset{0.3}{P(\text{red} \mid \text{negative})} * \underset{0.3}{P(\text{circle} \mid \text{negative})} / P(X)$$

$$= 0.009 / P(X) = 0.009 / 0.0495 = 0.1818$$

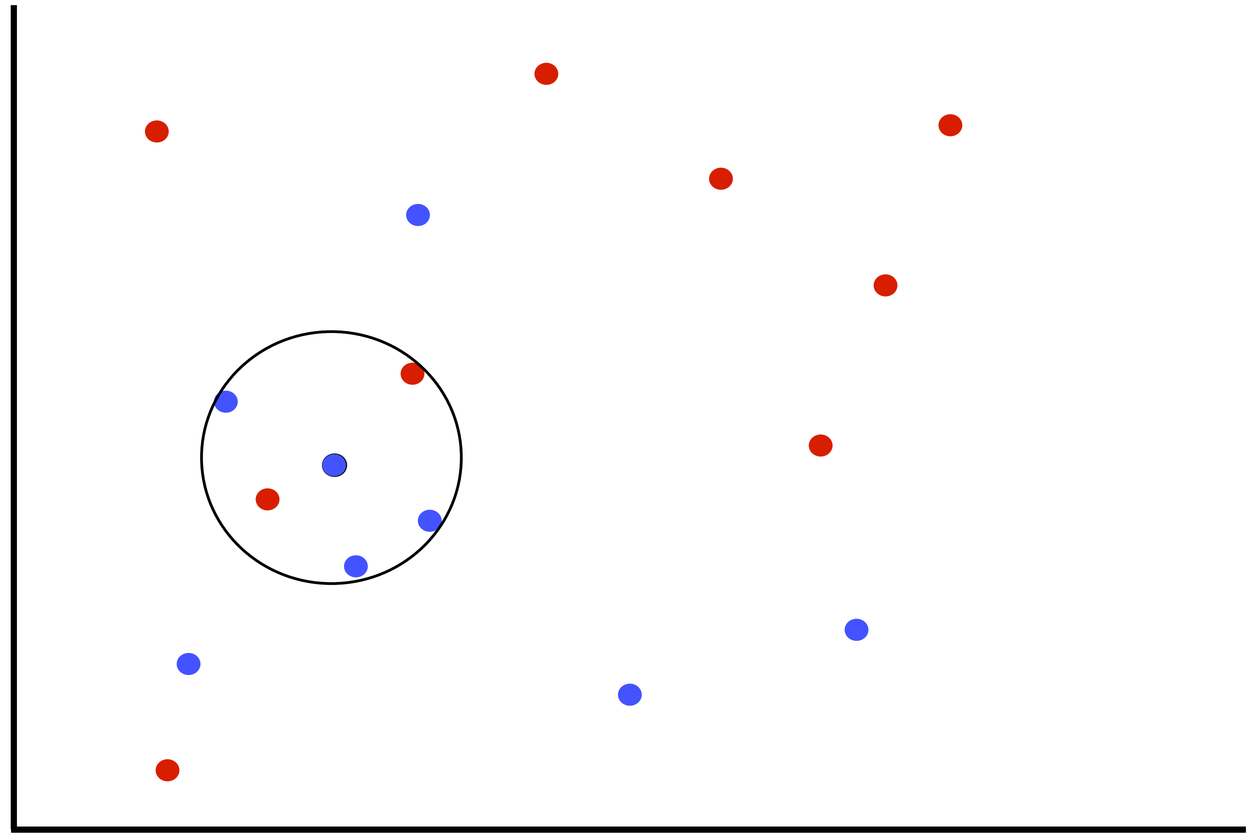
$$P(\text{positive} \mid X) + P(\text{negative} \mid X) = 0.0405 / P(X) + 0.009 / P(X) = 1$$

$$P(X) = (0.0405 + 0.009) = 0.0495$$

Instance-based Learning. K-Nearest Neighbor

- Calculate the distance between a test point and every training instance.
- Pick the *k* closest training examples and assign the test instance to the most common category amongst these nearest neighbors.
- Voting multiple neighbors helps decrease susceptibility to noise.
- Usually use odd value for *k* to avoid ties.

5-Nearest Neighbor Example



Applications

- Data mining:

  mining in IS MU - e-learning tests; ICT competencies

- Text mining: text categorization, part-of-speech (morphological) tagging, information extraction

  Spam filtering, Czech newspaper analysis, reports on flood, firemen data vs. web

- Web mining: web usage analysis, web content mining

  e-commerce, stubs in Wikipedia, web pages of SME