

Foundations of Artificial Intelligence

CS101
FALL 2007



Learning Theory

1

What is learning?

- “Learning denotes changes in a system that ... enable a system to do the same task more efficiently the next time.” –Herbert Simon
- “Learning is constructing or modifying representations of what is being experienced.” –Ryszard Michalski
- “Learning is making useful changes in our minds.” –Marvin Minsky

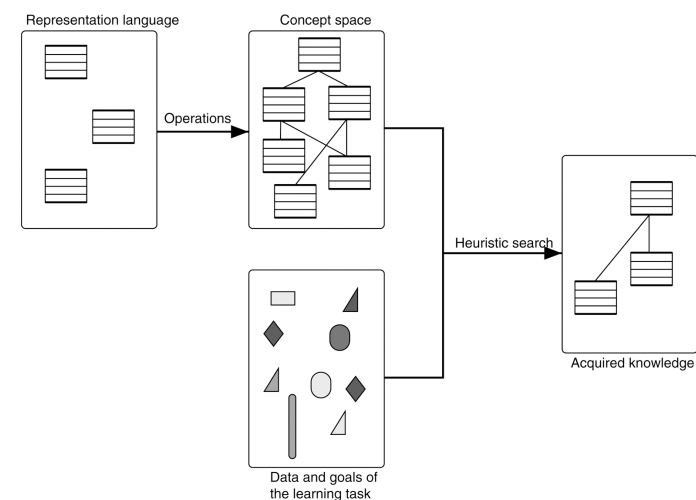
2

Why learn?

- Understand and improve efficiency of human learning
 - Use to improve methods for teaching and tutoring people (e.g., better computer-aided instruction)
- Discover new things or structure that were previously unknown to humans
 - Examples: data mining, scientific discovery
- Fill in skeletal or incomplete specifications about a domain
 - Large, complex AI systems cannot be completely derived by hand and require dynamic updating to incorporate new information.
 - Learning new characteristics expands the domain or expertise and lessens the “brittleness” of the system
- Build software agents that can adapt to their users or to other software agents

3

A general model of the learning process



4

Major paradigms of machine learning

- **Rote learning** – One-to-one mapping from inputs to stored representation. “Learning by memorization.” Association-based storage and retrieval.
- **Induction** – Use specific examples to reach general conclusions
- **Clustering** – Unsupervised identification of natural groups in data
- **Analogy** – Determine correspondence between two different representations
- **Discovery** – Unsupervised, specific goal not given
- **Genetic algorithms** – “Evolutionary” search techniques, based on an analogy to “survival of the fittest”
- **Reinforcement** – Feedback (positive or negative reward) given at the end of a sequence of steps

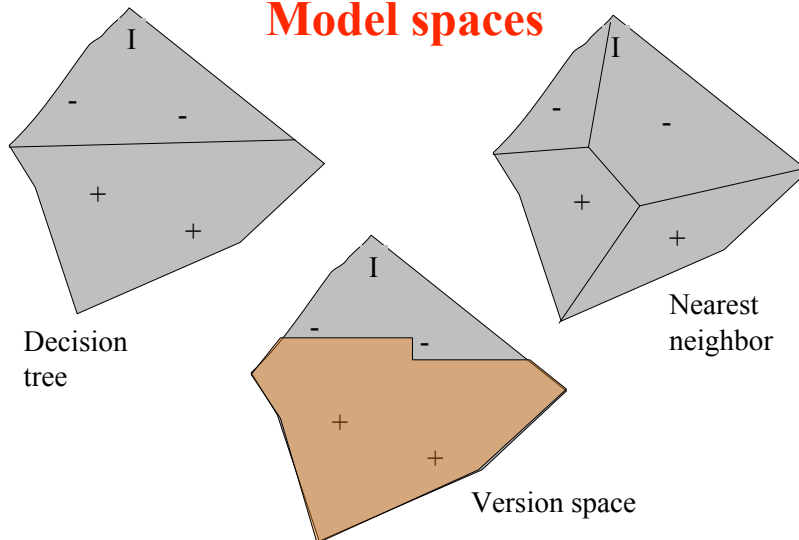
5

The inductive learning problem

- Extrapolate from a given set of examples to make accurate predictions about future examples
- Supervised versus unsupervised learning
 - Learn an unknown function $f(X) = Y$, where X is an input example and Y is the desired output.
 - **Supervised learning** implies we are given a **training set** of (X, Y) pairs by a “teacher”
 - **Unsupervised learning** means we are only given the X s and some (ultimate) feedback function on our performance.
- Concept learning or classification
 - Given a set of examples of some concept/class/category, determine if a given example is an instance of the concept or not
 - If it is an instance, we call it a positive example
 - If it is not, it is called a negative example
 - Or we can make a probabilistic prediction (e.g., using a Bayes net)

6

Model spaces



7

A learning game with playing cards

I would like to show what a *full house* is. I give you examples which are/are not full houses:

6♦ 6♠ 6♥ 9♣ 9♥	is a full house
6♦ 6♠ 6♥ 6♣ 9♥	is not a full house
3♣ 3♥ 3♣ 6♦ 6♠	is a full house
1♣ 1♥ 1♣ 6♦ 6♠	is a full house
Q♣ Q♥ Q♣ 6♦ 6♠	is a full house
1♦ 2♠ 3♥ 4♣ 5♥	is not a full house
1♦ 1♠ 3♥ 4♣ 5♥	is not a full house
1♦ 1♠ 1♥ 4♣ 5♥	is not a full house
1♦ 1♠ 1♥ 4♣ 4♥	is a full house

8

A learning game with playing cards

If you haven't guessed already, a *full house* is three of a kind and a pair of another kind.

6 ♦ 6 ♠ 6 ♥ 9 ♣ 9 ♥	is a full house
6 ♦ 6 ♠ 6 ♥ 6 ♣ 9 ♥	is not a full house
3 ♣ 3 ♥ 3 ♣ 6 ♦ 6 ♠	is a full house
1 ♣ 1 ♥ 1 ♣ 6 ♦ 6 ♠	is a full house
Q ♣ Q ♥ Q ♣ 6 ♦ 6 ♠	is a full house
1 ♦ 2 ♠ 3 ♥ 4 ♣ 5 ♥	is not a full house
1 ♦ 1 ♠ 3 ♥ 4 ♣ 5 ♥	is not a full house
1 ♦ 1 ♠ 1 ♥ 4 ♣ 5 ♥	is not a full house
1 ♦ 1 ♠ 1 ♥ 4 ♣ 4 ♥	is a full house

9

Supervised learning

This 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* of a target concept.

11

Intuitively,

I'm asking you to describe a *set*. This set is the *concept* I want you to *learn*.

This is called *inductive learning*, i.e., learning a generalization from a set of examples.

Concept learning is a typical inductive learning problem: given examples of some concept, such as "cat," "soy protein milkshake," or "good stock investment," we attempt to infer a definition that will allow the learner to correctly recognize future instances of that concept.

10

Why Supervised learning?

This definition might seem counter intuitive. If the teacher knows the concept, why doesn't s/he tell us directly and save us all the work?

12

Supervised learning – the 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.

13

Supervised concept learning

- Given a training set of positive and negative examples of a concept
- Construct a description that will accurately classify whether future examples are positive or negative
- That is, learn some good estimate of function f given a training set $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$ where each y_i is either + (positive) or - (negative), or a probability distribution over +/-

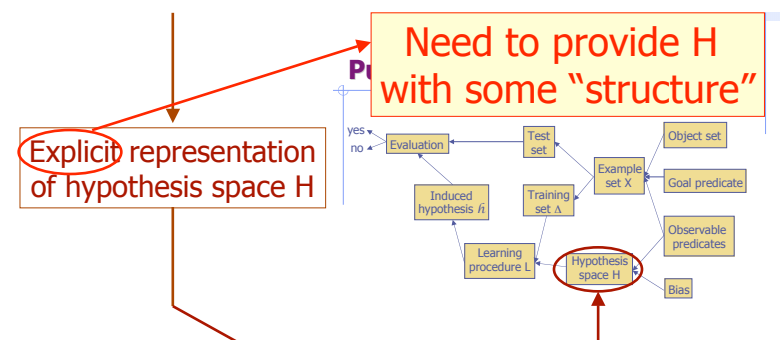
14

Inductive learning as search

- Instance space I defines the language for the training and test instances
 - Typically, but not always, each instance $i \in I$ is a feature vector
 - Features are also sometimes called attributes or variables
 - $I: V_1 \times V_2 \times \dots \times V_k, i = (v_1, v_2, \dots, v_k)$
- Class variable C gives an instance's class (to be predicted)
- Model space M defines the possible classifiers
 - $M: I \rightarrow C, M = \{m_1, \dots, m_n\}$ (possibly infinite)
 - Model space is sometimes, but not always, defined in terms of the same features as the instance space
- Training data can be used to direct the search for a good (consistent, complete, simple) hypothesis in the model space

Predicate-Learning Methods

- Decision tree
- Version space



15

16

Learning a predicate

Set E of objects (e.g., cards, drinking cups, writing instruments)

Goal predicate CONCEPT (X), where X is an object in E, that takes the value True or False (e.g., REWARD, MUG, PENCIL, BALL)

Observable predicates A(X), B(X), ... (e.g., NUM, RED, HAS-HANDLE, HAS-ERASER)

Training set: values of CONCEPT for some combinations of values of the observable predicates

Find a representation of CONCEPT of the form
 $\text{CONCEPT}(X) \Leftrightarrow A(X) \wedge (B(X) \vee C(X))$

17

How can we do this?

Go with the most general hypothesis possible:
 "any card is a rewarded card"

This will cover all the positive examples, but will not be able to eliminate any negative examples.

Go with the most specific hypothesis possible:
 "the rewarded cards are 4 ♣, 7 ♣, 2 ♠"

This will correctly sort all the examples in the training set, but it is overly specific, will not be able to sort any new examples.

But the above two are good starting points.

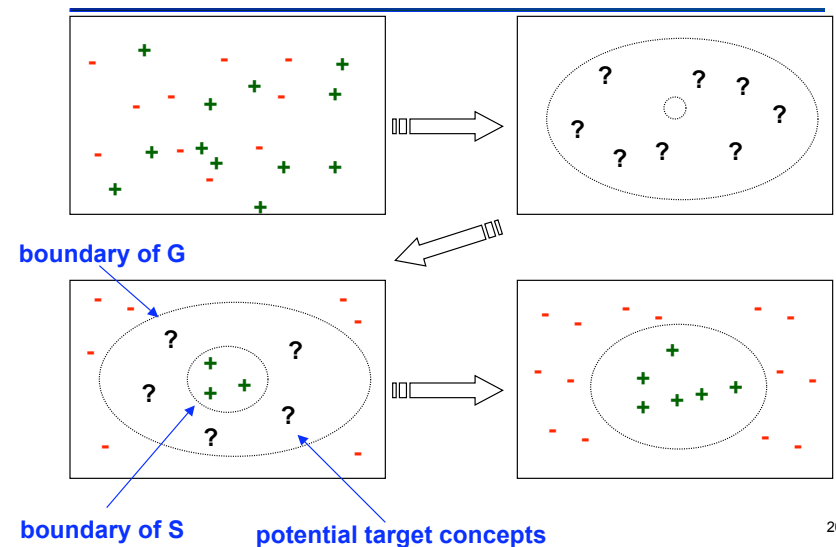
18

Version Spaces

- The "version space" is the set of all hypotheses that are consistent with the training instances processed so far.
- An algorithm:
 - $V := H$;; the version space V is ALL hypotheses H
 - For each example e:
 - Eliminate any member of V that disagrees with e
 - If V is empty, FAIL
 - Return V as the set of consistent hypotheses

19

Pictorially



20

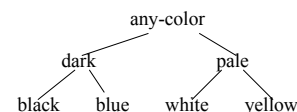
Version Spaces: The Problem

- PROBLEM: V is huge!!
- Suppose you have N attributes, each with k possible values
- Suppose you allow a hypothesis to be any disjunction of instances
- There are k^N possible instances $\rightarrow |H| = 2^{k^N}$
- If $N=5$ and $k=2$, $|H| = 2^{32}$!!

21

Version Spaces: The Tricks

- First Trick: Don't allow arbitrary disjunctions
 - Organize the feature values into a hierarchy of allowed disjunctions, e.g.



- Now there are only 7 “abstract values” instead of 16 disjunctive combinations (e.g., “black or white” isn't allowed)
- Second Trick: Define a partial ordering on H (“general to specific”) and only keep track of the upper bound and lower bound of the version space
- RESULT: An incremental, efficient algorithm!

22

Rewarded Card Example

- $(r=1) \vee \dots \vee (r=10) \vee (r=J) \vee (r=Q) \vee (r=K) \Leftrightarrow \text{ANY-RANK}(r)$
 $(r=1) \vee \dots \vee (r=10) \Leftrightarrow \text{NUM}(r)$
 $(r=J) \vee (r=Q) \vee (r=K) \Leftrightarrow \text{FACE}(r)$
 $(s=\spadesuit) \vee (s=\clubsuit) \vee (s=\diamond) \vee (s=\heartsuit) \Leftrightarrow \text{ANY-SUIT}(s)$
 $(s=\spadesuit) \vee (s=\clubsuit) \Leftrightarrow \text{BLACK}(s)$
 $(s=\diamond) \vee (s=\heartsuit) \Leftrightarrow \text{RED}(s)$

A hypothesis is any sentence of the form:

$$R(r) \wedge S(s) \Leftrightarrow \text{IN-CLASS}([r,s])$$

where:

- $R(r)$ is ANY-RANK(r), NUM(r), FACE(r), or ($r=j$)
- $S(s)$ is ANY-SUIT(s), BLACK(s), RED(s), or ($s=k$)

23

Simplified Representation

For simplicity, we represent a concept by rs , with:

- $r \in \{a, n, f, 1, \dots, 10, j, q, k\}$
- $s \in \{a, b, r, \clubsuit, \spadesuit, \diamond, \heartsuit\}$

For example:

- $n\spadesuit$ represents:
 $\text{NUM}(r) \wedge (s=\spadesuit) \Leftrightarrow \text{IN-CLASS}([r,s])$
- aa represents:
 $\text{ANY-RANK}(r) \wedge \text{ANY-SUIT}(s) \Leftrightarrow \text{IN-CLASS}([r,s])$

24

Rewarded card example

Training set:

$\text{REWARD}([4,C]) \wedge \text{REWARD}([7,C]) \wedge$
 $\text{REWARD}([2,S]) \wedge \neg \text{REWARD}([5,H]) \wedge$
 $\neg \text{REWARD}([J,S])$

Card	In the target set?
4 ♣	yes
7 ♣	yes
2 ♠	yes
5 ♥	no
J ♠	no

Possible *inductive hypothesis*, h :

$h = (\text{NUM}(r) \wedge \text{BLACK}(s) \leftrightarrow \text{REWARD}([r,s])$

25

Extension of a Hypothesis

The **extension** of a hypothesis h is the set of objects that satisfies h

Examples:

- The extension of $f♠$ is: $\{j♠, q♠, k♠\}$
- The extension of aa is the set of all cards

26

More General/Specific Relation

- Let h_1 and h_2 be two hypotheses in H
- h_1 is **more general** than h_2 iff the extension of h_1 is a proper superset of the extension of h_2

Examples:

- aa is more general than $f♦$
- $f♥$ is more general than $q♥$
- fr and nr are not comparable

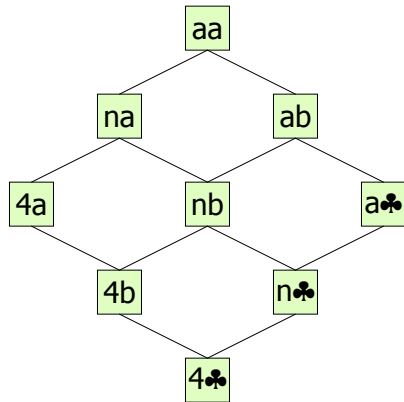
27

More General/Specific Relation

- Let h_1 and h_2 be two hypotheses in H
- h_1 is **more general** than h_2 iff the extension of h_1 is a proper superset of the extension of h_2
- 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

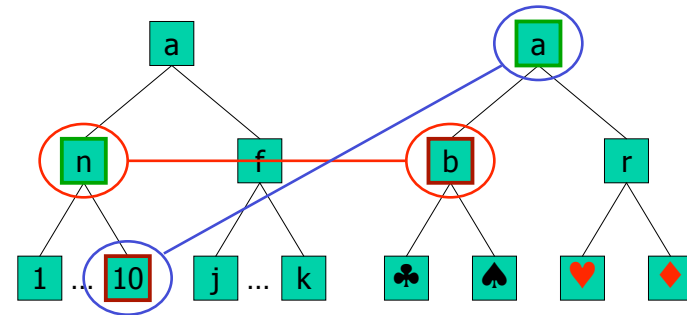
28

Example: Subset of Partial Order



29

Construction of Ordering Relation



30

G-Boundary / S-Boundary of V

- 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

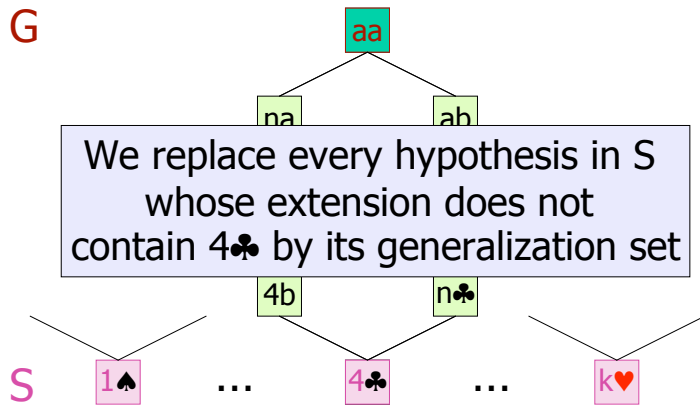
31

G-Boundary / S-Boundary of V

- 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

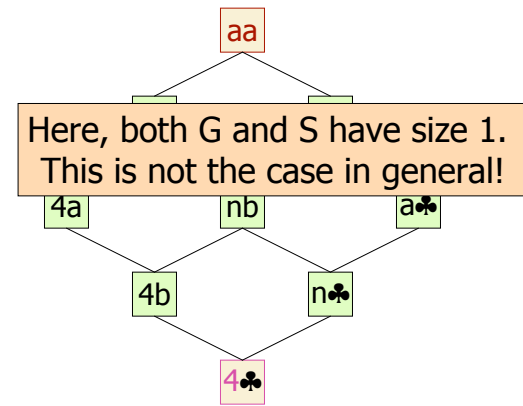
32

Example: G-/S-Boundaries of V



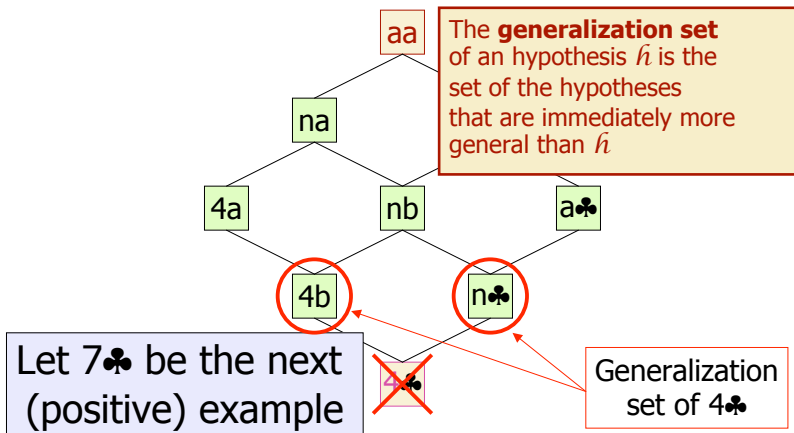
33

Example: G-/S-Boundaries of V



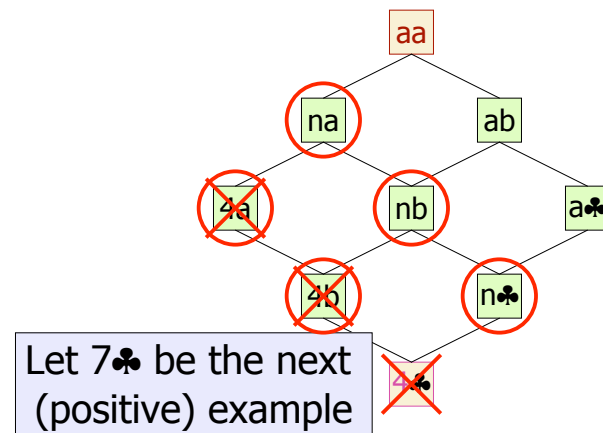
34

Example: G-/S-Boundaries of V



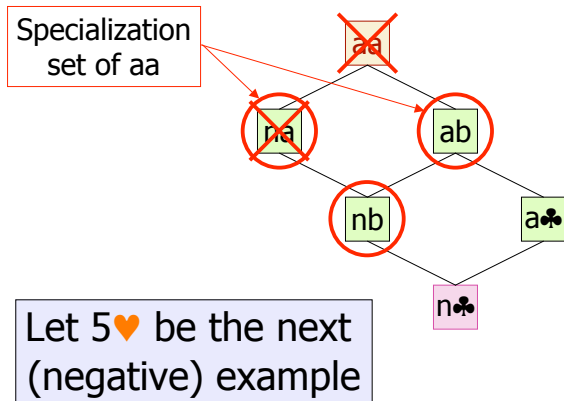
35

Example: G-/S-Boundaries of V



36

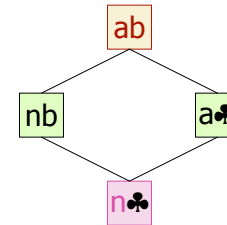
Example: G-/S-Boundaries of V



37

Example: G-/S-Boundaries of V

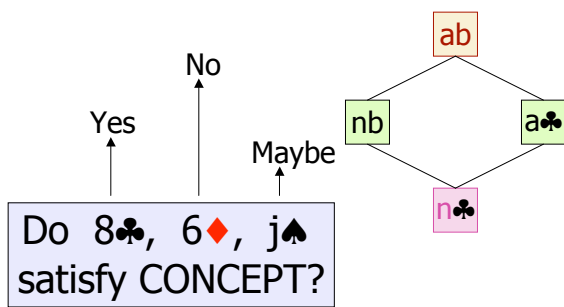
G and S, and all hypotheses in between form exactly the version space



38

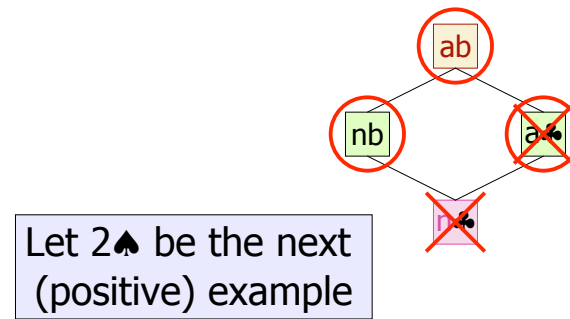
Example: G-/S-Boundaries of V

At this stage ...



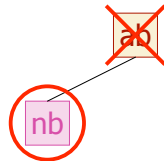
39

Example: G-/S-Boundaries of V



40

Example: G-/S-Boundaries of V



Let j^{\spadesuit} be the next
(negative) example

41

Example: G-/S-Boundaries of V

+ $4^{\clubsuit} 7^{\clubsuit} 2^{\spadesuit}$
- $5^{\heartsuit} j^{\spadesuit}$

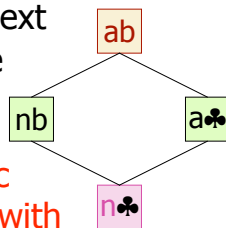


$\text{NUM}(r) \wedge \text{BLACK}(s) \Leftrightarrow \text{IN-CLASS}([r,s])$

42

Example: G-/S-Boundaries of V

Let us return to the
version space ...
... and let 8^{\clubsuit} be the next
(negative) example

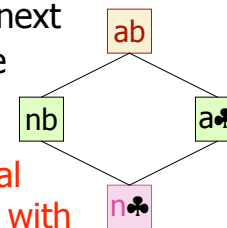


The only most specific
hypothesis disagrees with
this example, so no
hypothesis in H agrees with
all examples

43

Example: G-/S-Boundaries of V

Let us return to the
version space ...
... and let j^{\heartsuit} be the next
(positive) example



The only most general
hypothesis disagrees with
this example, so no
hypothesis in H agrees with
all examples

44

Version Space Update

1. $x \leftarrow$ new example
2. If x is positive then
 $(G,S) \leftarrow \text{POSITIVE-UPDATE}(G,S,x)$
3. Else
 $(G,S) \leftarrow \text{NEGATIVE-UPDATE}(G,S,x)$
4. If G or S is empty then return failure

45

POSITIVE-UPDATE(G,S,x)

1. Eliminate all hypotheses in G that do not agree with x

46

POSITIVE-UPDATE(G,S,x)

1. Eliminate all hypotheses in G that do not agree with x
2. Minimally generalize all hypotheses in S until they are consistent with x

Using the generalization sets of the hypotheses

47

POSITIVE-UPDATE(G,S,x)

1. Eliminate all hypotheses in G that do not agree with x
2. Minimally generalize all hypotheses in S until they are consistent with x
3. Remove from S every hypothesis that is neither more specific than nor equal to a hypothesis in G

This step was not needed in the card example

48

POSITIVE-UPDATE(G,S,x)

1. Eliminate all hypotheses in G that do not agree with x
2. Minimally generalize all hypotheses in S until they are consistent with x
3. Remove from S every hypothesis that is neither more specific than nor equal to a hypothesis in G
4. Remove from S every hypothesis that is more general than another hypothesis in S
5. Return (G,S)

49

NEGATIVE-UPDATE(G,S,x)

1. Eliminate all hypotheses in S that do not agree with x
2. Minimally specialize all hypotheses in G until they are consistent with (exclude) x
3. Remove from G every hypothesis that is neither more general than nor equal to a hypothesis in S
4. Remove from G every hypothesis that is more specific than another hypothesis in G
5. Return (G,S)

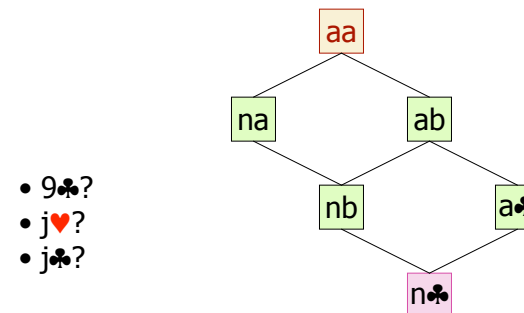
50

Example-Selection Strategy

- Suppose that at each step the learning procedure has the possibility to select the object (card) of the next example
- Let it pick the object such that, whether the example is positive or not, it will eliminate one-half of the remaining hypotheses
- Then a single hypothesis will be isolated in $O(\log |H|)$ steps

51

Example



52

Example-Selection Strategy

- Suppose that at each step the learning procedure has the possibility to select the object (card) of the next example
- Let it pick the object such that, whether the example is positive or not, it will eliminate one-half of the remaining hypotheses
- Then a single hypothesis will be isolated in $O(\log |H|)$ steps
- But picking the object that eliminates half the version space may be expensive

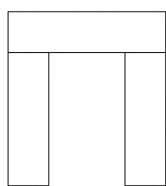
53

Noise

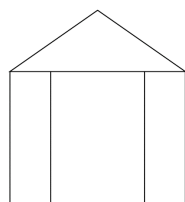
- If some examples are **misclassified**, the version space may collapse
- **Possible solution:**
Maintain several G- and S-boundaries, e.g., consistent with all examples, all examples but one, etc...

54

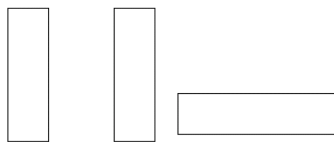
Examples and near misses for "arch" concept



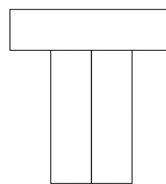
Arch



Arch



Near miss



Near miss

55

More on generalization operators

Replacing constants with variables. For example,

`color (ball,red)`
generalizes to
`color (X,red)`

Dropping conditions from a conjunctive expression. For example,

`shape (X, round) \wedge size (X, small) \wedge color (X, red)`
generalizes to
`shape (X, round) \wedge color (X, red)`

56

More on generalization operators (cont'd)

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)

57

Another example

sizes = {large, small}

colors = {red, white, blue}

shapes = {sphere, brick, cube}

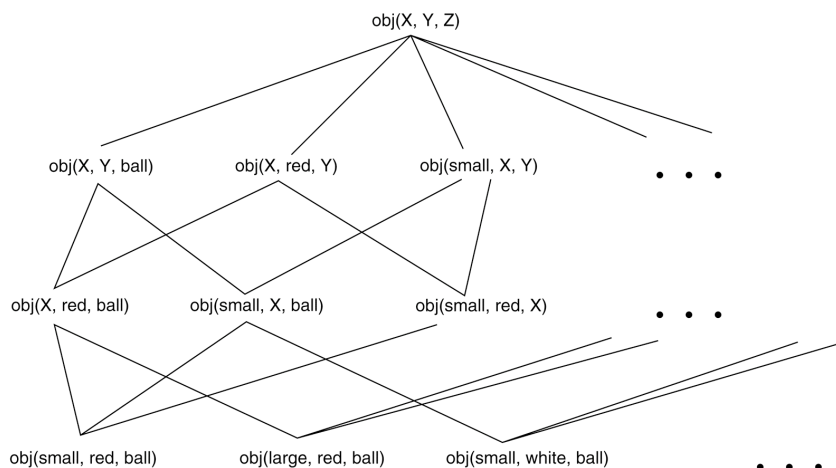
object (size, color, shape)

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.

58

A portion of the concept space



59

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) }

60

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

Decision Tree Learning

61

62

Learning decision trees

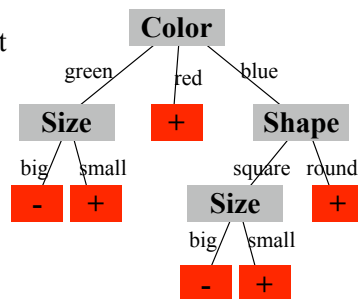
- Goal: Build a **decision tree** to classify examples as positive or negative instances of a concept using supervised learning from a training set

- A **decision tree** is a tree where

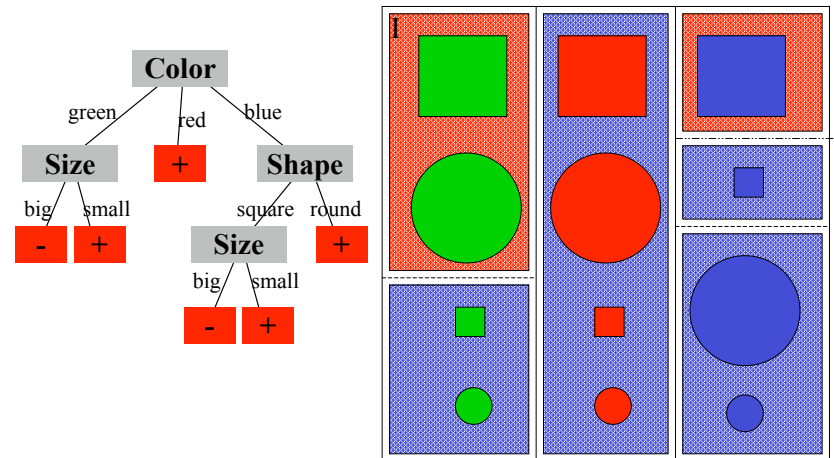
- each non-leaf node has associated with it an attribute (feature)
- each leaf node has associated with it a classification (+ or -)
- each arc has associated with it one of the possible values of the attribute at the node from which the arc is directed

- Generalization: allow for >2 classes

- e.g., {sell, hold, buy}



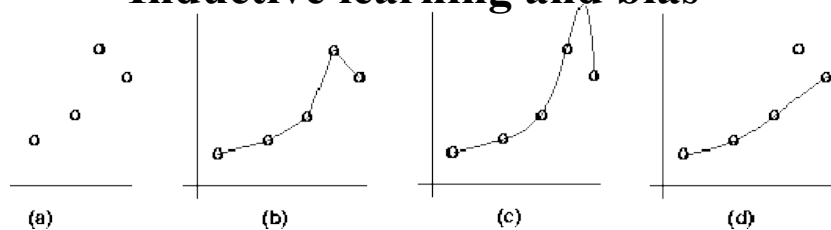
Decision tree-induced partition – example



63

64

Inductive learning and bias



- Suppose that we want to learn a function $f(x) = y$ and we are given some sample (x,y) pairs, as in figure (a)
- There are several hypotheses we could make about this function, e.g.: (b), (c) and (d)
- A preference for one over the others reveals the **bias** of our learning technique, e.g.:
 - prefer piece-wise functions
 - prefer a smooth function
 - prefer a simple function and treat outliers as noise

65

Preference bias: Ockham's Razor

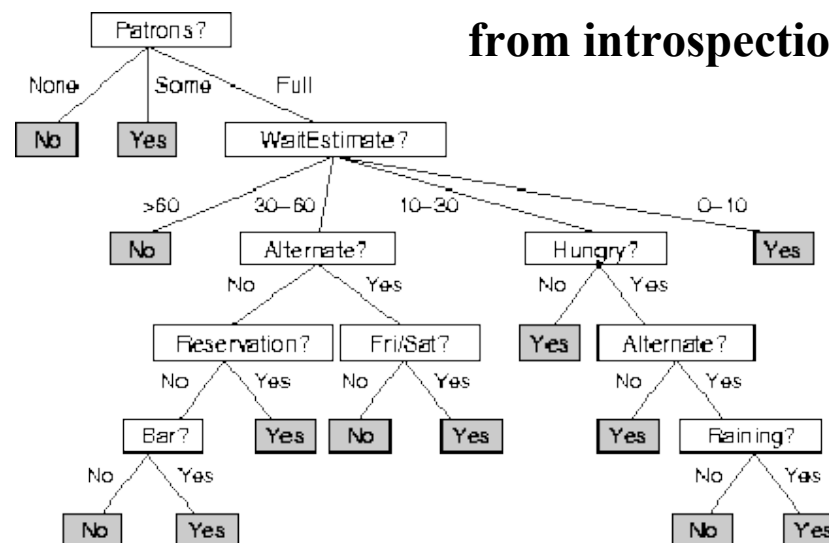
- A.k.a. Occam's Razor, Law of Economy, or Law of Parsimony
- Principle stated by William of Ockham (1285-1347/49), a scholastic, that
 - “*non sunt multiplicanda entia praeter necessitatem*”
 - or, entities are not to be multiplied beyond necessity
- The simplest consistent explanation is the best
- Therefore, the smallest decision tree that correctly classifies all of the training examples is best.
- Finding the provably smallest decision tree is NP-hard, so instead of constructing the absolute smallest tree consistent with the training examples, construct one that is pretty small

66

R&N's restaurant domain

- Develop a decision tree to model the decision a patron makes when deciding whether or not to wait for a table at a restaurant
- Two classes: wait, leave
- Ten attributes: Alternative available? Bar in restaurant? Is it Friday? Are we hungry? How full is the restaurant? How expensive? Is it raining? Do we have a reservation? What type of restaurant is it? What's the purported waiting time?
- Training set of 12 examples
- ~ 7000 possible cases

A decision tree from introspection



67

68

A training set

Example	Attributes										Goal
	Aff	Bar	Fri	Fun	Hot	Price	Rain	Res	Type	Est	WSU/Wait
X ₁	Yes	No	No	Yes	Some	\$\$\$	No	Yes	French	0-10	Yes
X ₂	Yes	No	No	Yes	Full	\$	No	No	Thai	30-60	No
X ₃	No	Yes	No	No	Some	\$	No	No	Burger	0-10	Yes
X ₄	Yes	No	Yes	Yes	Full	\$	No	No	Thai	10-30	Yes
X ₅	Yes	No	Yes	No	Full	\$\$\$	No	Yes	French	>60	No
X ₆	No	Yes	No	Yes	Some	\$\$	Yes	Yes	Italian	0-10	Yes
X ₇	No	Yes	No	No	None	\$	Yes	No	Burger	0-10	No
X ₈	No	No	No	Yes	Some	\$\$	Yes	Yes	Thai	0-10	Yes
X ₉	No	Yes	Yes	No	Full	\$	Yes	No	Burger	>60	No
X ₁₀	Yes	Yes	Yes	Yes	Full	\$\$\$	No	Yes	Italian	10-30	No
X ₁₁	No	No	No	No	None	\$	No	No	Thai	0-10	No
X ₁₂	Yes	Yes	Yes	Yes	Full	\$	No	No	Burger	30-60	Yes

69

ID3

- A greedy algorithm for decision tree construction developed by Ross Quinlan, 1987
- Top-down construction of the decision tree by recursively selecting the “best attribute” to use at the current node in the tree
 - Once the attribute is selected for the current node, generate children nodes, one for each possible value of the selected attribute
 - Partition the examples using the possible values of this attribute, and assign these subsets of the examples to the appropriate child node
 - Repeat for each child node until all examples associated with a node are either all positive or all negative

70

Choosing the best attribute

- The key problem is choosing which attribute to split a given set of examples
- Some possibilities are:
 - **Random:** Select any attribute at random
 - **Least-Values:** Choose the attribute with the smallest number of possible values
 - **Most-Values:** Choose the attribute with the largest number of possible values
 - **Max-Gain:** Choose the attribute that has the largest expected information gain—i.e., the attribute that will result in the smallest expected size of the subtrees rooted at its children
- The ID3 algorithm uses the Max-Gain method of selecting the best attribute

71

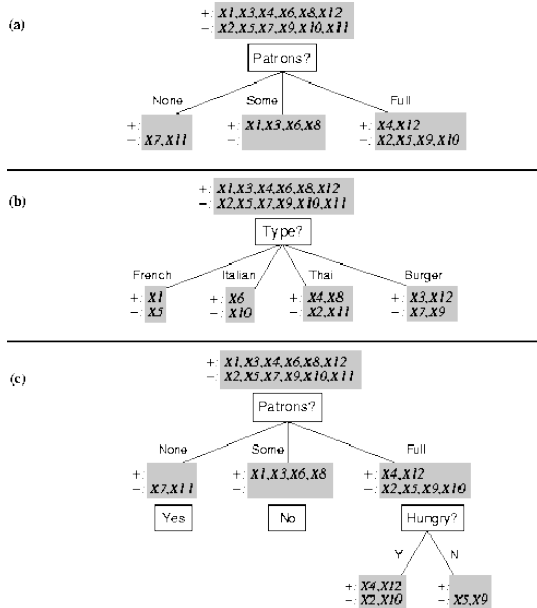
Restaurant example

Random: Patrons or Wait-time; **Least-values:** Patrons; **Most-values:** Type; **Max-gain:** ???

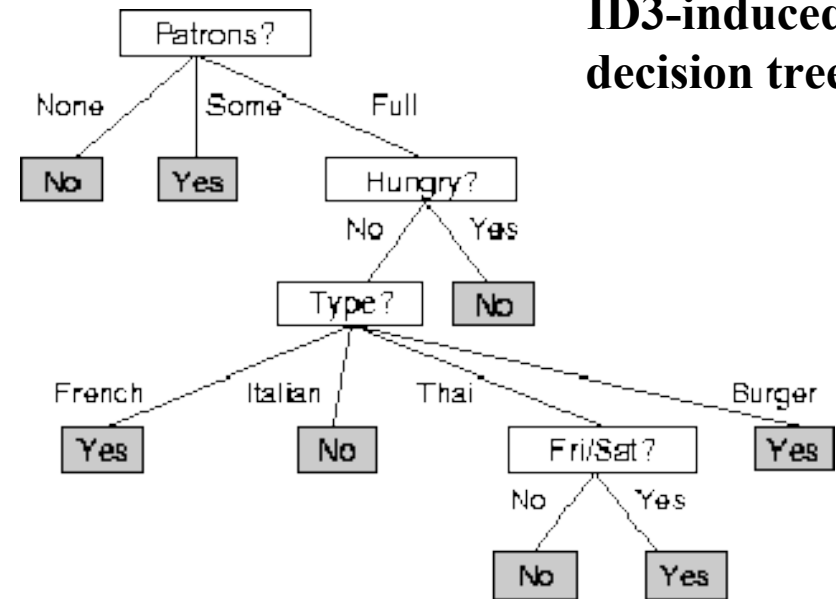
French		Y	N
Italian		Y	N
Thai	N	Y	N Y
Burger	N	Y	N Y
	Empty	Some	Full

72

Splitting examples by testing attributes



ID3-induced decision tree



Information theory

- If there are n equally probable possible messages, then the probability p of each is $1/n$
- Information conveyed by a message is $-\log(p) = \log(n)$
- E.g., if there are 16 messages, then $\log(16) = 4$ and we need 4 bits to identify/send each message
- In general, if we are given a probability distribution $P = (p_1, p_2, \dots, p_n)$
- Then the information conveyed by the distribution (aka *entropy* of P) is:

$$I(P) = -(p_1 * \log(p_1) + p_2 * \log(p_2) + \dots + p_n * \log(p_n))$$

Information theory II

- Information conveyed by distribution (a.k.a. *entropy* of P):

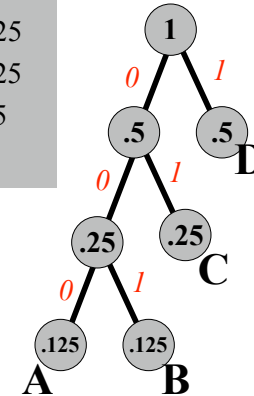
$$I(P) = -(p_1 * \log(p_1) + p_2 * \log(p_2) + \dots + p_n * \log(p_n))$$
- Examples:
 - If P is (0.5, 0.5) then $I(P)$ is 1
 - If P is (0.67, 0.33) then $I(P)$ is 0.92
 - If P is (1, 0) then $I(P)$ is 0
- The more uniform the probability distribution, the greater its information: More information is conveyed by a message telling you which event actually occurred
- Entropy is the average number of bits/message needed to represent a stream of messages

Huffman code

- In 1952 MIT student David Huffman devised, in the course of doing a homework assignment, an elegant coding scheme which is optimal in the case where all symbols' probabilities are integral powers of 1/2.
- A Huffman code can be built in the following manner:
 - Rank all symbols in order of probability of occurrence
 - Successively combine the two symbols of the lowest probability to form a new composite symbol; eventually we will build a binary tree where each node is the probability of all nodes beneath it
 - Trace a path to each leaf, noticing the direction at each node

Huffman code example

Msg.	Prob.
A	.125
B	.125
C	.25
D	.5



M	code	length	prob	
A	000	3	0.125	0.375
B	001	3	0.125	0.375
C	01	2	0.250	0.500
D	1	1	0.500	0.500
average message length				1.750

If we use this code to many messages (A,B,C or D) with this probability distribution, then, over time, the average bits/message should approach 1.75

77

78

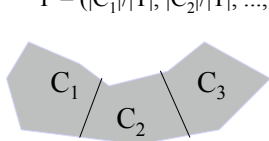
Information for classification

- If a set T of records is partitioned into disjoint exhaustive classes (C_1, C_2, \dots, C_k) on the basis of the value of the class attribute, then the information needed to identify the class of an element of T is

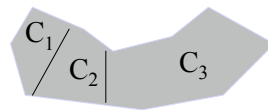
$$\text{Info}(T) = I(P)$$

where P is the probability distribution of partition (C_1, C_2, \dots, C_k):

$$P = (|C_1|/|T|, |C_2|/|T|, \dots, |C_k|/|T|)$$



High information



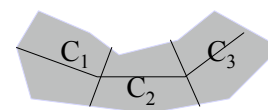
Low information

79

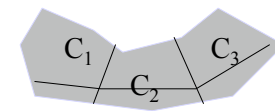
Information for classification II

- If we partition T w.r.t attribute X into sets $\{T_1, T_2, \dots, T_n\}$ then the information needed to identify the class of an element of T becomes the weighted average of the information needed to identify the class of an element of T_i , i.e. the weighted average of $\text{Info}(T_i)$:

$$\text{Info}(X, T) = \sum |T_i|/|T| * \text{Info}(T_i)$$



High information



Low information

80

Information gain

- Consider the quantity Gain(X,T) defined as
 $\text{Gain}(X,T) = \text{Info}(T) - \text{Info}(X,T)$
- This represents the difference between
 - information needed to identify an element of T and
 - information needed to identify an element of T after the value of attribute X has been obtained

That is, this is the **gain in information due to attribute X**

- We can use this to rank attributes and to build decision trees where at each node is located the attribute with greatest gain among the attributes not yet considered in the path from the root
- The intent of this ordering is:
 - To create small decision trees so that records can be identified after only a few questions
 - To match a hoped-for minimality of the process represented by the records being considered (Occam's Razor)

Computing information gain

$$\begin{aligned} \bullet I(T) &= \\ &= (.5 \log .5 + .5 \log .5) \\ &= .5 + .5 = 1 \end{aligned}$$

$$\begin{aligned} \bullet I(\text{Pat}, T) &= \\ &= \frac{1}{6}(0) + \frac{1}{3}(0) + \\ &\quad \frac{1}{2}(-(\frac{2}{3} \log \frac{2}{3} + \\ &\quad \frac{1}{3} \log \frac{1}{3})) \\ &= \frac{1}{2}(2/3 * .6 + \\ &\quad 1/3 * 1.6) \\ &= .47 \end{aligned}$$

$$\begin{aligned} \bullet I(\text{Type}, T) &= \\ &= \frac{1}{6}(1) + \frac{1}{6}(1) + \\ &= \frac{1}{3}(1) + \frac{1}{3}(1) = 1 \end{aligned}$$

French	Y	N
Italian	Y	N
Thai	Y	N Y
Burger	N	N Y
Empty	Y	N Y
	Some	Full

$$\begin{aligned} \text{Gain}(\text{Pat}, T) &= 1 - .47 = .53 \\ \text{Gain}(\text{Type}, T) &= 1 - 1 = 0 \end{aligned}$$

81

82

The ID3 algorithm is used to build a decision tree, given a set of non-categorical attributes C1, C2, ..., Cn, the class attribute C, and a training set T of records.

```
function ID3 (R: a set of input attributes,
             C: the class attribute,
             S: a training set) returns a decision tree;
begin
  If S is empty, return a single node with value Failure;
  If every example in S has the same value for C, return
  single node with that value;
  If R is empty, then return a single node with most
  frequent of the values of C found in examples S;
  [note: there will be errors, i.e., improperly classified
  records];
  Let D be attribute with largest Gain(D,S) among attributes in R;
  Let {dj| j=1,2, .., m} be the values of attribute D;
  Let {Sj| j=1,2, .., m} be the subsets of S consisting
  respectively of records with value dj for attribute D;
  Return a tree with root labeled D and arcs labeled
  d1, d2, .., dm going respectively to the trees
  ID3 (R-{D}, C, S1), ID3 (R-{D}, C, S2) ,... , ID3 (R-{D}, C, Sm);
end ID3;
```

83

How well does it work?

Many case studies have shown that decision trees are at least as accurate as human experts.

- A study for diagnosing breast cancer had humans correctly classifying the examples 65% of the time; the decision tree classified 72% correct
- British Petroleum designed a decision tree for gas-oil separation for offshore oil platforms that replaced an earlier rule-based expert system
- Cessna designed an airplane flight controller using 90,000 examples and 20 attributes per example

84

Extensions of the decision tree learning algorithm

- Using gain ratios
- Real-valued data
- Noisy data and overfitting
- Generation of rules
- Setting parameters
- Cross-validation for experimental validation of performance
- C4.5 is an extension of ID3 that accounts for unavailable values, continuous attribute value ranges, pruning of decision trees, rule derivation, and so on

Using gain ratios

- The information gain criterion favors attributes that have a large number of values
 - If we have an attribute D that has a distinct value for each record, then $\text{Info}(D,T)$ is 0, thus $\text{Gain}(D,T)$ is maximal
- To compensate for this Quinlan suggests using the following ratio instead of Gain:

$$\text{GainRatio}(D,T) = \text{Gain}(D,T) / \text{SplitInfo}(D,T)$$
- $\text{SplitInfo}(D,T)$ is the information due to the split of T on the basis of value of categorical attribute D

$$\text{SplitInfo}(D,T) = I(|T1|/|T|, |T2|/|T|, \dots, |Tm|/|T|)$$
 where $\{T1, T2, \dots, Tm\}$ is the partition of T induced by value of D

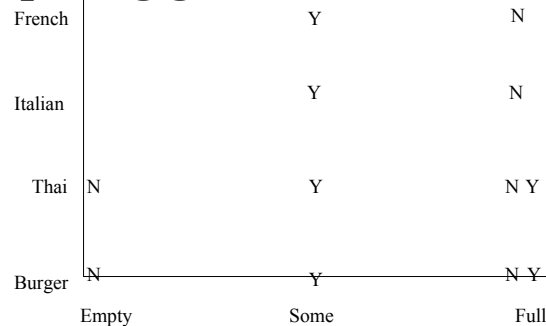
85

86

Computing gain ratio

- $I(T) = 1$
- $I(\text{Pat}, T) = .47$
- $I(\text{Type}, T) = 1$

Gain (Pat, T) = .53
Gain (Type, T) = 0



$$\text{SplitInfo}(\text{Pat}, T) = -(1/6 \log 1/6 + 1/3 \log 1/3 + 1/2 \log 1/2) = 1/6 * 2.6 + 1/3 * 1.6 + 1/2 * 1 = 1.47$$

$$\text{SplitInfo}(\text{Type}, T) = 1/6 \log 1/6 + 1/6 \log 1/6 + 1/3 \log 1/3 + 1/3 \log 1/3 = 1/6 * 2.6 + 1/6 * 2.6 + 1/3 * 1.6 + 1/3 * 1.6 = 1.93$$

$$\text{GainRatio}(\text{Pat}, T) = \text{Gain}(\text{Pat}, T) / \text{SplitInfo}(\text{Pat}, T) = .53 / 1.47 = .36$$

$$\text{GainRatio}(\text{Type}, T) = \text{Gain}(\text{Type}, T) / \text{SplitInfo}(\text{Type}, T) = 0 / 1.93 = 0$$

87

Real-valued data

- Select a set of thresholds defining intervals
- Each interval becomes a discrete value of the attribute
- Use some simple heuristics...
 - always divide into quartiles
- Use domain knowledge...
 - divide age into infant (0-2), toddler (3 - 5), school-aged (5-8)
- Or treat this as another learning problem
 - Try a range of ways to discretize the continuous variable and see which yield “better results” w.r.t. some metric
 - E.g., try midpoint between every pair of values

88

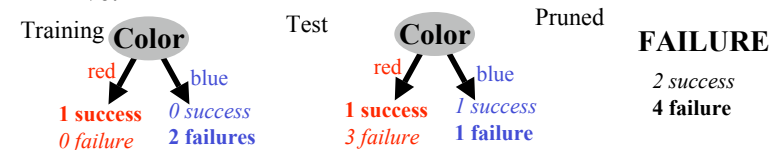
Noisy data and overfitting

- Many kinds of “noise” can occur in the examples:
 - Two examples have same attribute/value pairs, but different classifications
 - Some values of attributes are incorrect because of errors in the data acquisition process or the preprocessing phase
 - The classification is wrong (e.g., + instead of -) because of some error
 - Some attributes are irrelevant to the decision-making process, e.g., color of a die is irrelevant to its outcome
- The last problem, irrelevant attributes, can result in overfitting the training example data.
 - If the hypothesis space has many dimensions because of a large number of attributes, we may find **meaningless regularity** in the data that is irrelevant to the true, important, distinguishing features
 - Fix by pruning lower nodes in the decision tree
 - For example, if Gain of the best attribute at a node is below a threshold, stop and make this node a leaf rather than generating children nodes

89

Pruning decision trees

- Pruning of the decision tree is done by replacing a whole subtree by a leaf node
- The replacement takes place if a decision rule establishes that the expected error rate in the subtree is greater than in the single leaf. E.g.,
 - Training: one training red success and two training blue failures
 - Test: three red failures and one blue success
 - Consider replacing this subtree by a single Failure node.
- After replacement we will have only two errors instead of five:



90

Converting decision trees to rules

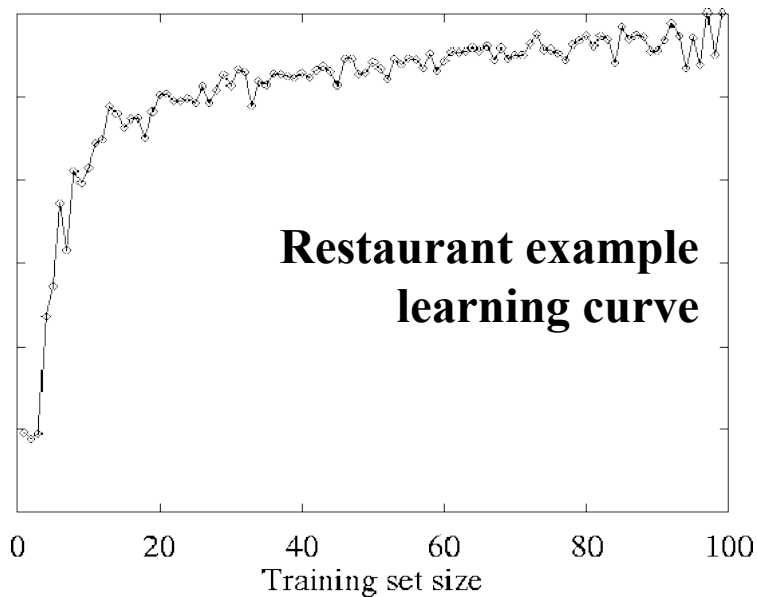
- It is easy to derive a rule set from a decision tree: write a rule for each path in the decision tree from the root to a leaf
- In that rule the left-hand side is easily built from the label of the nodes and the labels of the arcs
- The resulting rules set can be simplified:
 - Let LHS be the left hand side of a rule
 - Let LHS' be obtained from LHS by eliminating some conditions
 - We can certainly replace LHS by LHS' in this rule if the subsets of the training set that satisfy respectively LHS and LHS' are equal
 - A rule may be eliminated by using metaconditions such as “if no other rule applies”

91

Evaluation methodology

- Standard methodology:
 1. Collect a large set of examples (all with correct classifications)
 2. Randomly divide collection into two disjoint sets: training and test
 3. Apply learning algorithm to training set giving hypothesis H
 4. Measure performance of H w.r.t. test set
- Important: keep the training and test sets disjoint!
- To study the efficiency and robustness of an algorithm, repeat steps 2-4 for different training sets and sizes of training sets
- If you improve your algorithm, start again with step 1 to avoid evolving the algorithm to work well on just this collection

92



93

Summary: Decision tree learning

- Inducing decision trees is one of the most widely used learning methods in practice
- Can out-perform human experts in many problems
- Strengths include
 - Fast
 - Simple to implement
 - Can convert result to a set of easily interpretable rules
 - Empirically valid in many commercial products
 - Handles noisy data
- Weaknesses include:
 - Univariate splits/partitioning using only one attribute at a time so limits types of possible trees
 - Large decision trees may be hard to understand
 - Requires fixed-length feature vectors
 - Non-incremental (i.e., batch method)

94

VSL vs DTL

- Decision tree learning (DTL) is more efficient if all examples are given in advance; else, it may produce successive hypotheses, each poorly related to the previous one
- Version space learning (VSL) is incremental
- DTL can produce simplified hypotheses that do not agree with all examples
- DTL has been more widely used in practice

95

Computational learning theory

- Probably approximately correct (PAC) learning:
 - Sample complexity (# of examples to “guarantee” correctness) grows with the size of the model space
- Stationarity assumption: Training set and test sets are drawn from the same distribution
 - Lots of recent work on what to do if this assumption is violated, but you know something about the relationship between the two distributions
- Theoretical results apply to fairly simple learning models (e.g., decision list learning)

96