# Section 18.3 Learning Decision Trees

CS4811 - Artificial Intelligence

Nilufer Onder
Department of Computer Science
Michigan Technological University

# Outline

Attribute-based representations

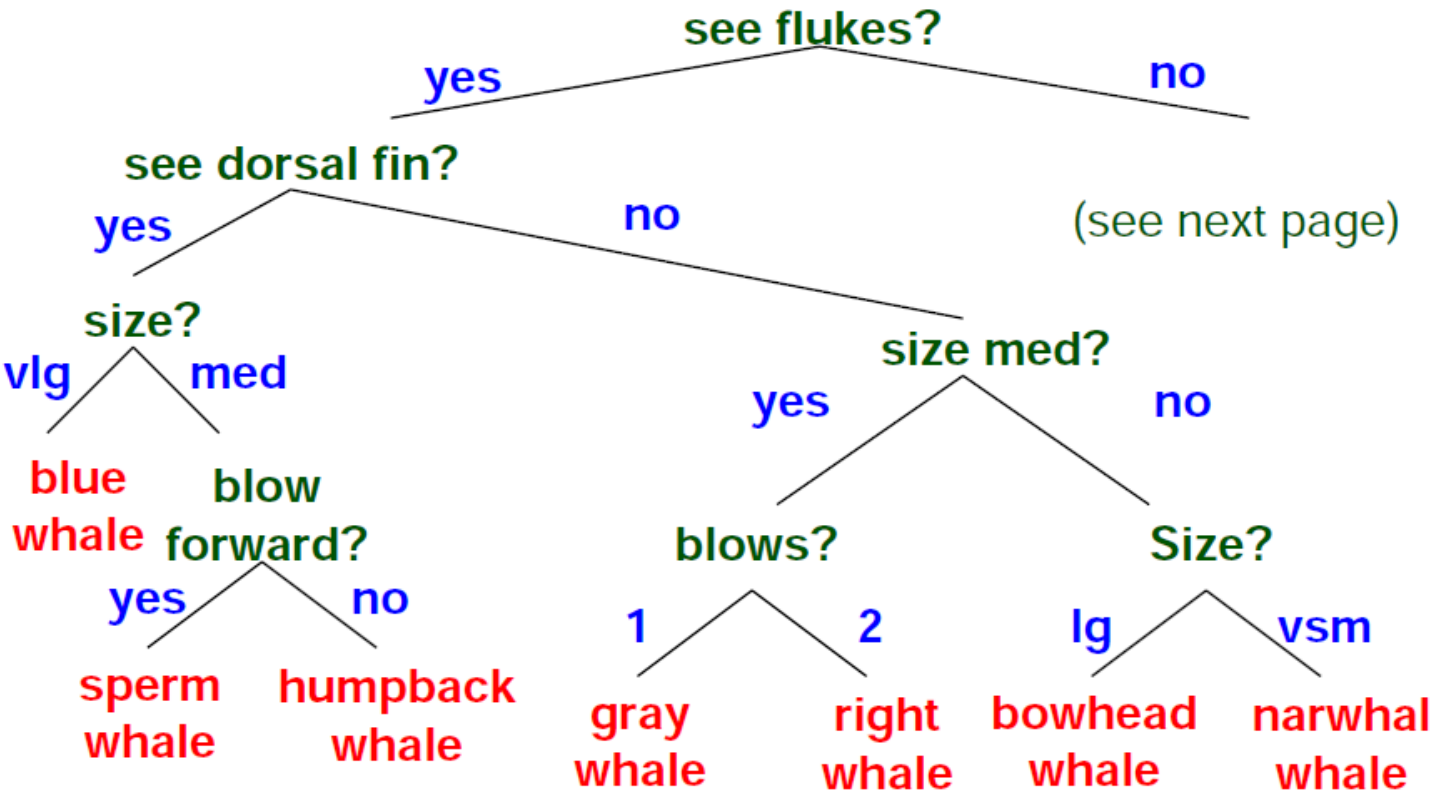
Decision tree learning as a search problem

A greedy algorithm

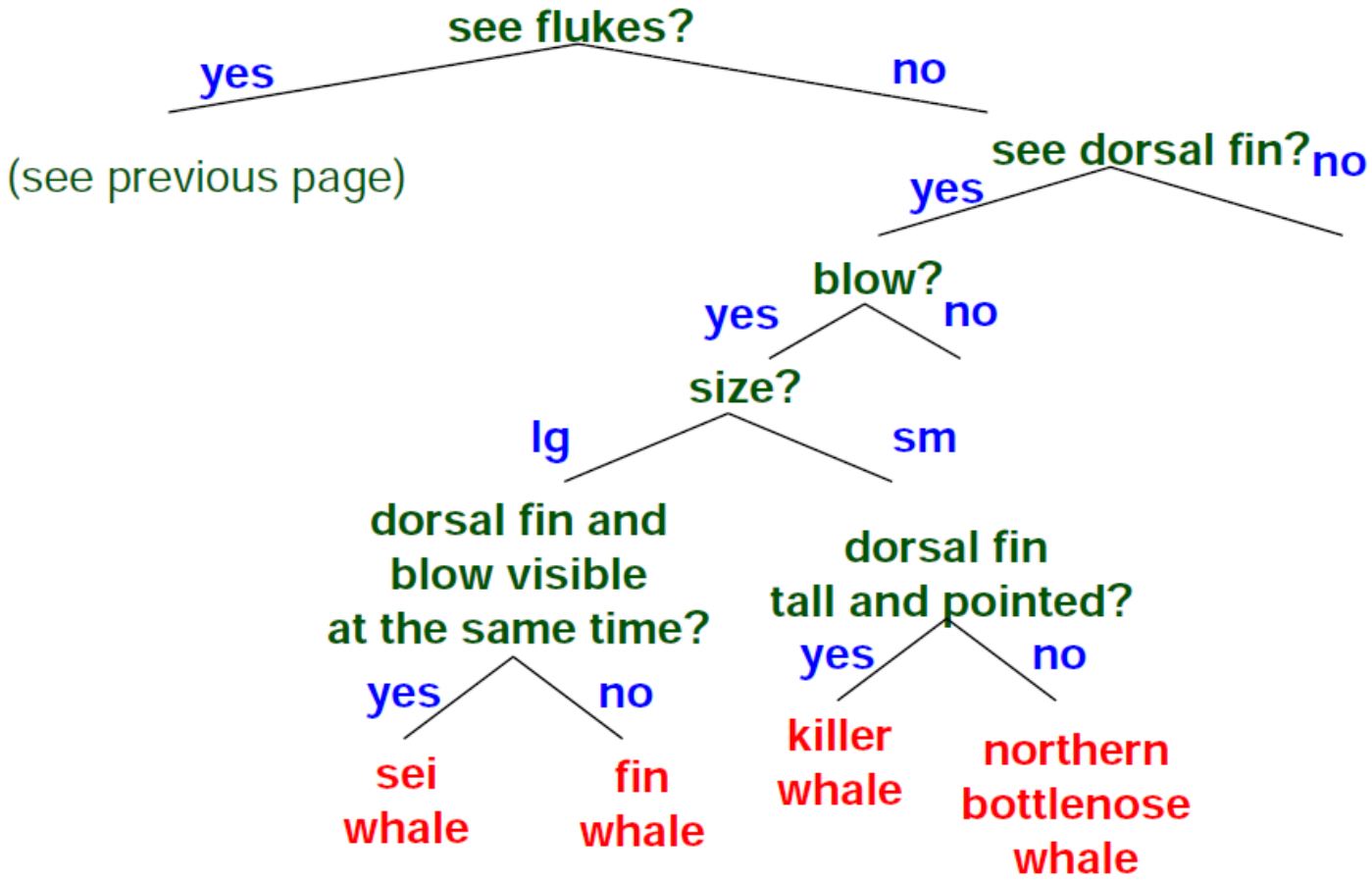
# Decision trees

- A *decision tree* allows a classification of an object by testing its values for certain properties.
- An example is the *20 questions game*.
  A player asks questions to an answerer and tries to guess the object that the answerer chose at the beginning of the game.
- The objective of *decision tree learning* is to learn a tree of questions which determines class membership at the leaf of each branch.
- Check out an online example at http://myacquire.com/aiinc/whalewatcher/

# Possible decision tree



# Possible decision tree (cont'd)



# What might the original data look like?

| Place | Time | Group | Fluke | Dorsal fin | Dorsal shape | Size | Blow | ... | Blow fwd | Type |
|---|---|---|---|---|---|---|---|---|---|---|
| Kaikora | 17:00 | Yes | Yes | Yes | small triang. | Very large | Yes | | No | Blue whale |
| Kaikora | 7:00 | No | Yes | Yes | small triang. | Very large | Yes | | No | Blue whale |
| Kaikora | 8:00 | Yes | Yes | Yes | small triang. | Very large | Yes | | No | Blue whale |
| Kaikora | 9:00 | Yes | Yes | Yes | squat triang. | Medium | Yes | | Yes | Sperm whale |
| Cape Cod | 18:00 | Yes | Yes | Yes | Irregu-lar | Medium | Yes | | No | Hump-back whale |
| Cape Cod | 20:00 | No | Yes | Yes | Irregu-lar | Medium | Yes | | No | Hump-back whale |
| Newb. Port | 18:00 | No | No | No | Curved | Large | Yes | | No | Fin whale |
| Cape Cod | 6:00 | Yes | Yes | No | None | Medium | Yes | | No | Right whale |
| ... | | | | | | | | | | |

# The search problem

This is an *attribute-based representation* where examples are described by *attribute values* (Boolean, discrete, continuous, etc.)

*Classification* of examples is positive (T) or negative (F).

Given a table of observable properties, search for a decision tree that

- correctly represents the data
  (for now, assume that the data is noise-free)
- is as small as possible

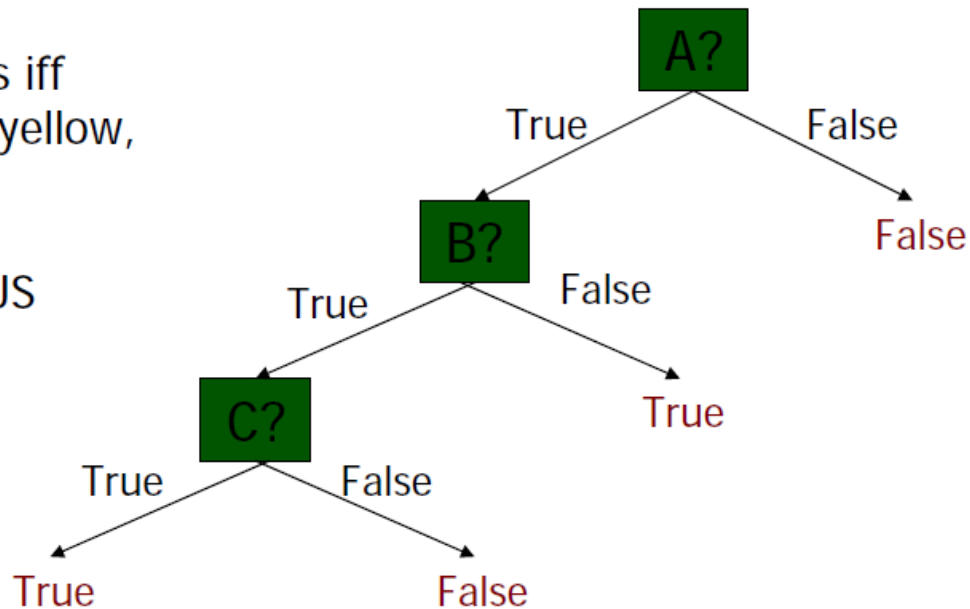
What does the search tree look like?

# Predicate as a decision tree

The predicate CONCEPT(x) $\Leftrightarrow$ A(x) $\wedge$ ($\neg$B(x) v C(x)) can be represented by the following decision tree:

Example:
A mushroom is poisonous iff
it is yellow and small, or yellow,
big and spotted

- x is a mushroom
- CONCEPT = POISONOUS
- A = YELLOW
- B = BIG
- C = SPOTTED
- D = FUNNEL-CAP
- E = BULKY

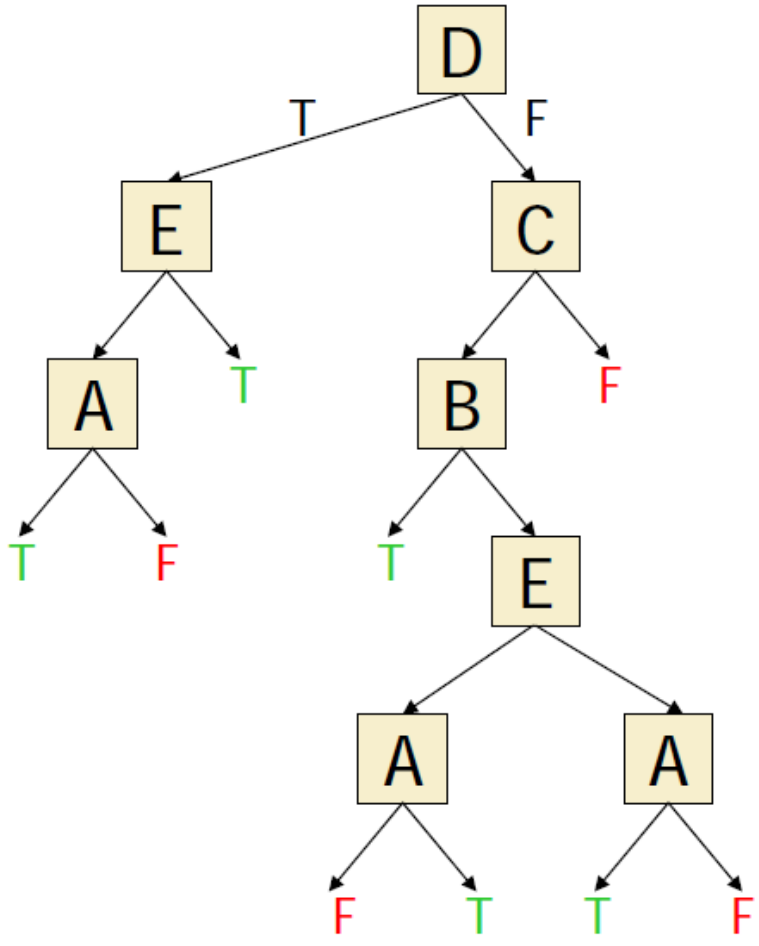


# The training set

| Ex. # | A | B | C | D | E | CONCEPT |
|---|---|---|---|---|---|---|
| 1 | False | False | True | False | True | False |
| 2 | False | True | False | False | False | False |
| 3 | False | True | True | True | True | False |
| 4 | False | False | True | False | False | False |
| 5 | False | False | False | True | True | False |
| 6 | True | False | True | False | False | True |
| 7 | True | False | False | True | False | True |
| 8 | True | False | True | False | True | True |
| 9 | True | True | True | False | True | True |
| 10 | True | True | True | True | True | True |
| 11 | True | True | False | False | False | False |
| 12 | True | True | False | False | True | False |
| 13 | True | False | True | True | True | True |

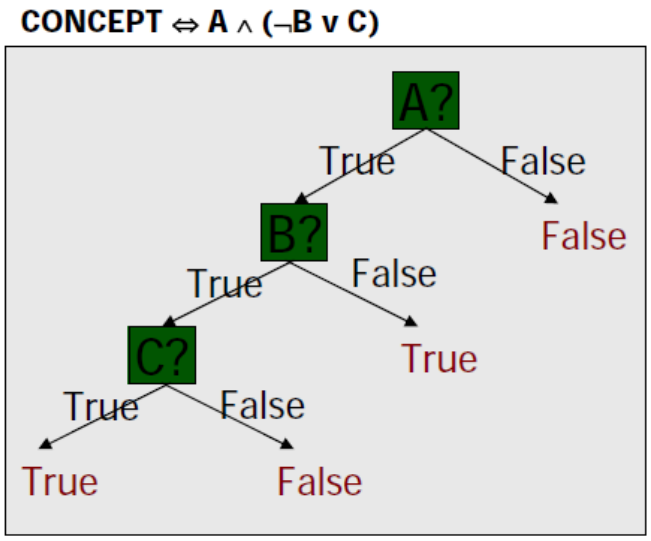
# Possible decision tree

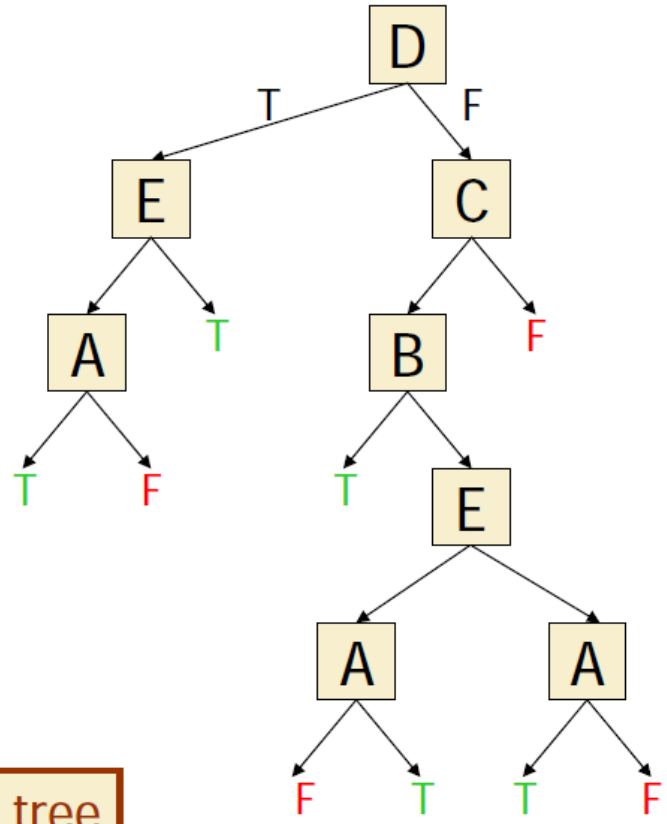
| Ex. # | A | B | C | D | E | CONCEPT |
|---|---|---|---|---|---|---|
| 1 | False | False | True | False | True | False |
| 2 | False | True | False | False | False | False |
| 3 | False | True | True | True | True | False |
| 4 | False | False | True | False | False | False |
| 5 | False | False | False | True | True | False |
| 6 | True | False | True | False | False | True |
| 7 | True | False | False | True | False | True |
| 8 | True | False | True | False | True | True |
| 9 | True | True | True | False | True | True |
| 10 | True | True | True | True | True | True |
| 11 | True | True | False | False | False | False |
| 12 | True | True | False | False | True | False |
| 13 | True | False | True | True | True | True |



# Smaller decision tree

CONCEPT ⇔
(D ∧ (¬E v A)) v
(C ∧ (B v ((E ∧ ¬A) v A)))




KIS bias → Build smallest decision tree

Computationally intractable problem→ greedy algorithm

# Building the decision tree - getting started (1)

The distribution of the training set is:

True: 6, 7, 8, 9, 10,13
False: 1, 2, 3, 4, 5, 11, 12

# Getting started (2)

The distribution of training set is:

True: 6, 7, 8, 9, 10,13
False: 1, 2, 3, 4, 5, 11, 12

Without testing any observable predicate, we could report that CONCEPT is False (majority rule) with an estimated probability of error P(E) = 6/13

# Getting started (3)

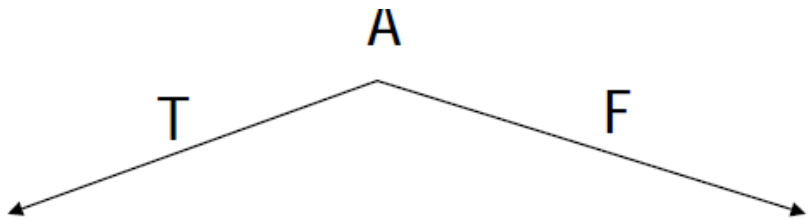
The distribution of training set is:

True: 6, 7, 8, 9, 10,13
False: 1, 2, 3, 4, 5, 11, 12

Without testing any observable predicate, we could report that CONCEPT is False (majority rule) with an estimated probability of error P(E) = 6/13

Assuming that we will only include one observable predicate in the decision tree, which predicate should we test to minimize the probability of error?

# How to compute the probability of error (1)



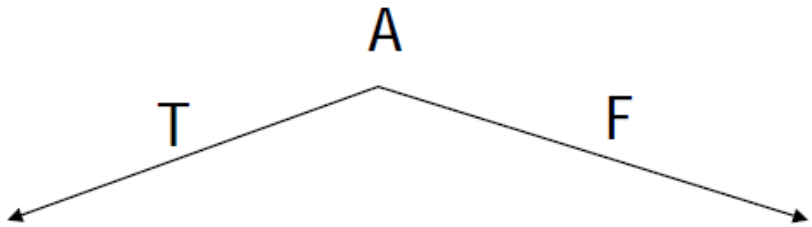

| | | |
|---|---|---|
| True: | 6, 7, 8, 9, 10, 13 | |
| False: | 11, 12 | 1, 2, 3, 4, 5 |

If we test only A, we will report that CONCEPT is True if A is True (majority rule) and False otherwise.

The estimated probability of error is:
$Pr(E) = (8/13)x(2/8) + (5/13)x(0/5) = 2/13$

8/13 is the probability of getting True for A, and
2/8 is the probability that the report was incorrect
(we are always reporting True for the concept).

# How to compute the probability of error (2)



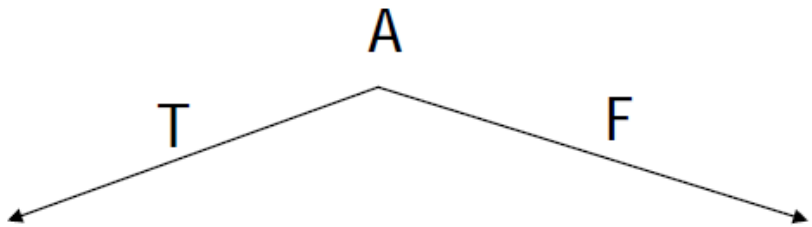

| | | |
|---|---|---|
| True: | 6, 7, 8, 9, 10, 13 | |
| False: | 11, 12 | 1, 2, 3, 4, 5 |

If we test only A, we will report that CONCEPT is True if A is True (majority rule) and False otherwise.

The estimated probability of error is:
$Pr(E) = (8/13)x(2/8) + (5/13)x(0/5) = 2/13$

5/13 is the probability of getting False for A, and
0 is the probability that the report was incorrect
(we are always reporting False for the concept).

# Assume it's A



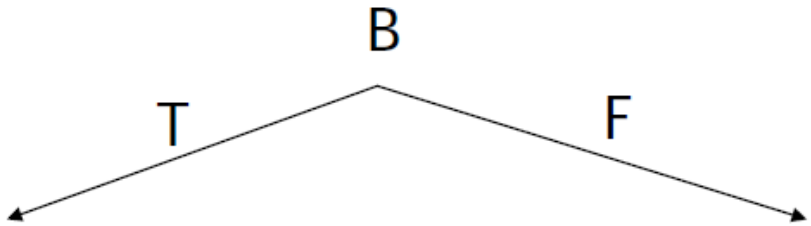

True: 6, 7, 8, 9, 10, 13

False: 11, 12 1, 2, 3, 4, 5

If we test only A, we will report that CONCEPT is True if A is True (majority rule) and False otherwise

The estimated probability of error is:
Pr(E) = (8/13)x(2/8) + (5/8)x0 = 2/13

# Assume it's B

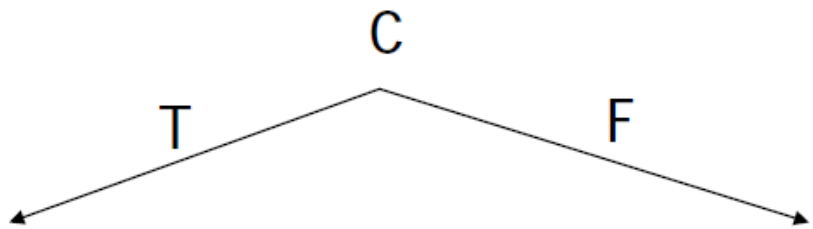



| | | |
|---|---|---|
| True: | 9, 10 | 6, 7, 8, 13 |
| False: | 2, 3, 11, 12 | 1, 4, 5 |

If we test only B, we will report that CONCEPT is False if B is True and True otherwise

The estimated probability of error is:
$Pr(E) = (6/13)x(2/6) + (7/13)x(3/7) = 5/13$

# Assume it's C

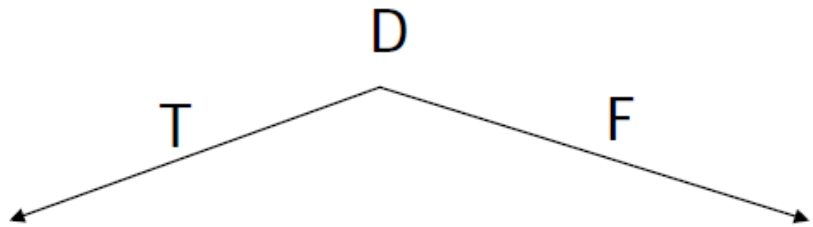



| | | |
|---|---|---|
| True: | 6, 8, 9, 10, 13 | 7 |
| False: | 1, 3, 4 | 1, 5, 11, 12 |

If we test only C, we will report that CONCEPT is True if C is True and False otherwise

The estimated probability of error is:
$Pr(E) = (8/13)x(3/8) + (5/13)x(1/5) = 4/13$

# Assume it's D

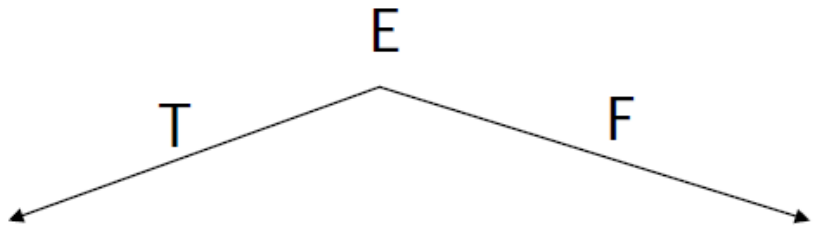



| | | |
|---|---|---|
| True: | 7, 10, 13 | 6, 8, 9 |
| False: | 3, 5 | 1, 2, 4, 11, 12 |

If we test only D, we will report that CONCEPT is True if D is True and False otherwise

The estimated probability of error is:
Pr(E) = (5/13)x(2/5) + (8/13)x(3/8) = 5/13

# Assume it's E




| | | |
|---|---|---|
| True: | 8, 9, 10, 13 | 6, 7 |
| False: | 1, 3, 5, 12 | 2, 4, 11 |

If we test only E we will report that CONCEPT is False, independent of the outcome

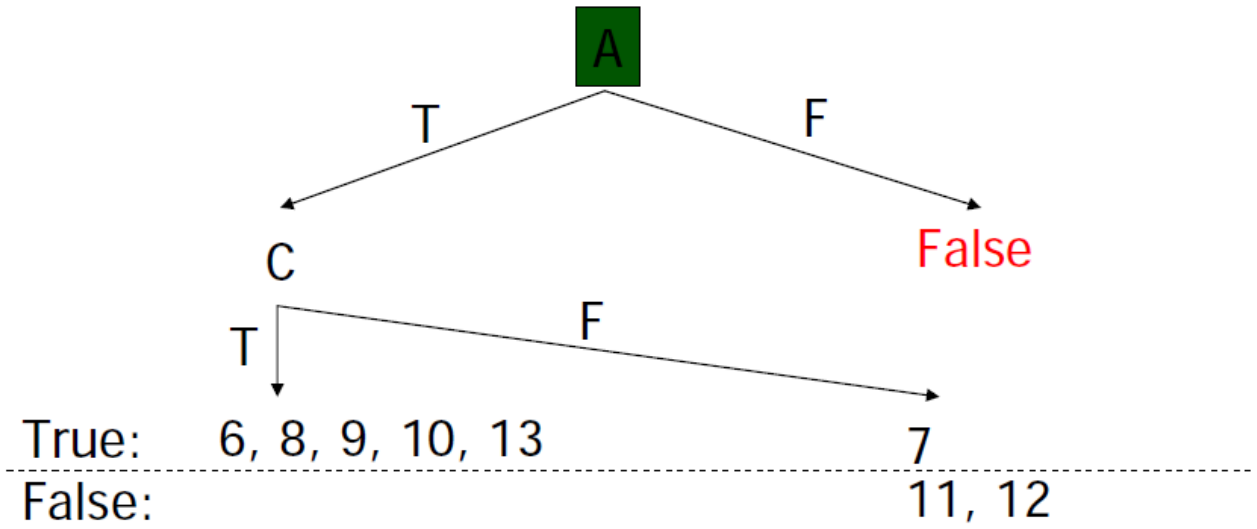
The estimated probability of error is:
Pr(E) = (8/13)x(4/8) + (5/13)x(2/5) = 6/13

# Probability of error for each

- If A: 2/13
- If B: 5/13
- If C: 4/13
- If D: 5/13
- If E: 6/13

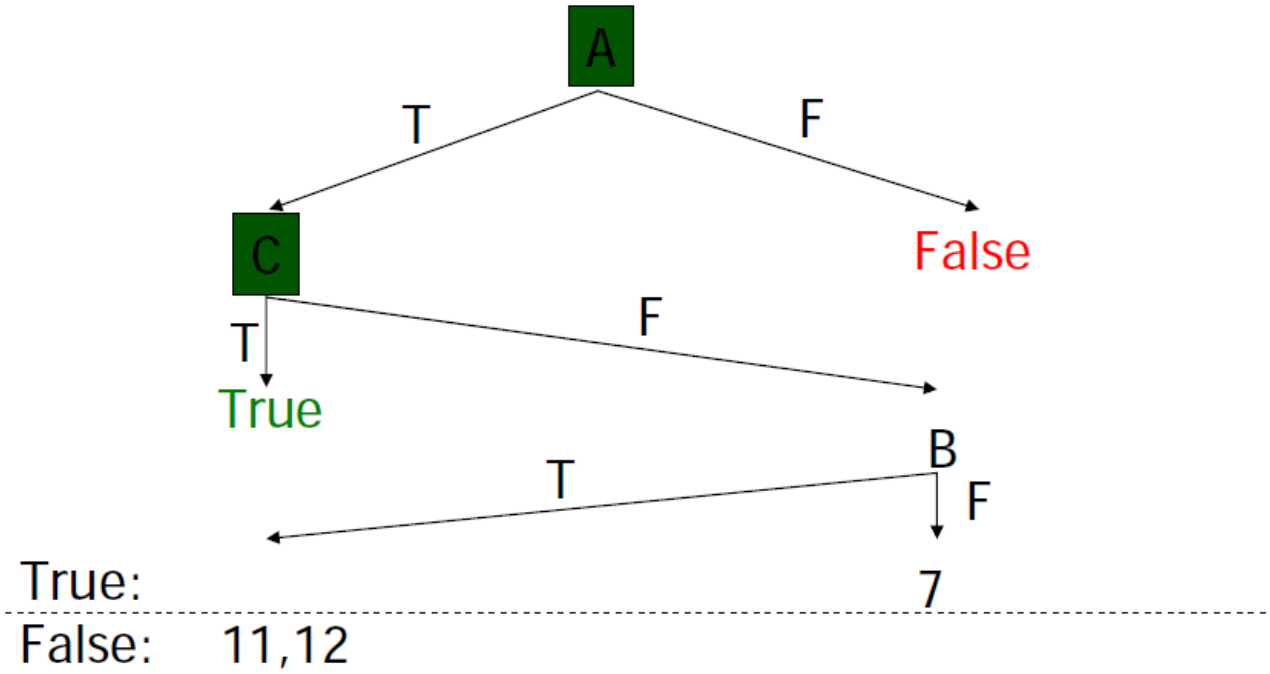
So, the best predicate to test is A

# Choice of second predicate

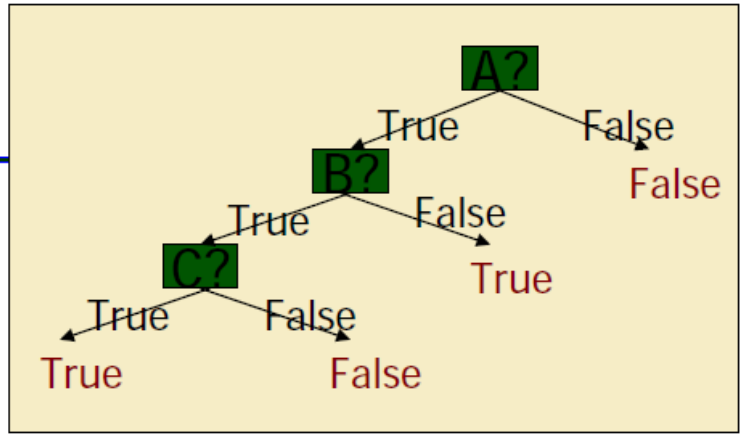


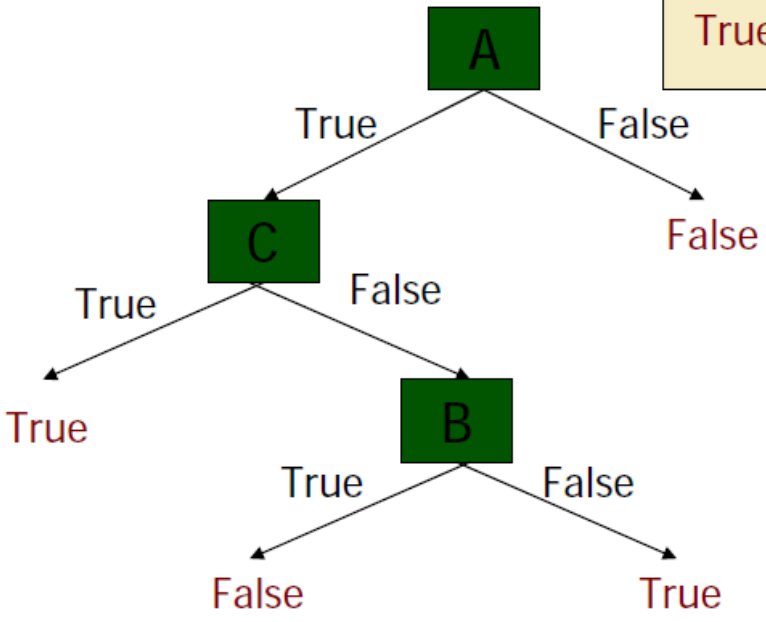

The majority rule gives the probability of error $\Pr(E|A) = 1/8$ and $\Pr(E) = 1/13$

# Choice of third predicate



# Final Tree




$$L \equiv \text{CONCEPT} \Leftrightarrow A \wedge (C \vee \neg B)$$

# The decision tree learning algorithm

**function** DECISION-TREE-LEARNING (*examples*, *attributes*, *parent-examples* )
**returns** a tree
  **if** *examples* is empty **then**
    **return** PLURALITY-VALUE(*parent-examples*)
  **else if** all *examples* have the same classification **then**
    **return** the classification
  **else if** *attributes* is empty **then**
    **return** PLURALITY-VALUE(*examples*)
  **else**
    $A \leftarrow \text{argmax}_{a \in attributes}$ IMPORTANCE($a$, *examples*)
    *tree* $\leftarrow$ a new decision tree with root test $A$
    **for each** value $v_k$ of $A$ do
      *exs* $\leftarrow \{ e : e \in examples$ **and** $e.A = v_k \}$
      *subtree* $\leftarrow$ DECISION-TREE-LEARNING (*exs*, *attributes-A*, *examples*)
      add a branch to *tree* with label $(A = v_k)$ and subtree *subtree*
  **return** *tree*

# Notes on the algorithm

- Notice that the "probability of error" calculations boil down to summing up the "minority numbers" and dividing by the total number of examples in that category. This is due to fraction cancellations. Probability of error is:

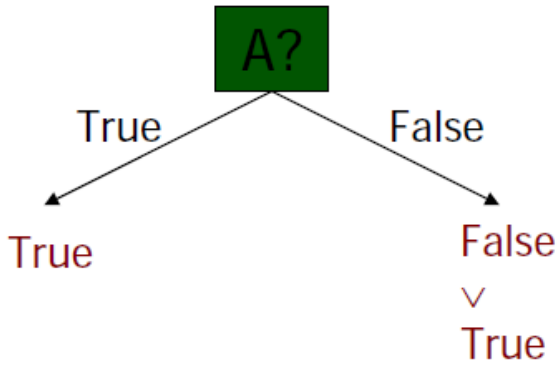
$$\frac{\text{minority } 1 + \text{minority } 2 + \ldots}{\text{total number of examples in this category}}$$

- After an attribute is selected take only the examples that have the attribute as labelled on the branch.

# What happens if there is noise in the training set?

Consider a very small but inconsistent data set:

| **A** | **classification** |
|---|---|
| T | T |
| F | F |
| F | T |



# Issues in learning decision trees

- If data for some attribute is missing and is hard to obtain, it might be possible to *extrapolate* or use unknown.
- If some attributes have continuous values, *groupings* might be used.
- If the data set is too large, one might use *bagging* to select a sample from the training set. Or, one can use *boosting* to assign a weight showing importance to each instance. Or, one can *divide the sample set into subsets* and train on one, and test on others.

# How large is the hypothesis space?

How many decision trees with $n$ Boolean attributes?

$=$ number of Boolean functions

$=$ number of distinct truth tables with $2^n$ rows.

$= 2^{2^n}$

# Using "probability of error"

- The "probability of error" is based on a measure of the *quantity of information* that is contained in the truth value of an observable attribute.
- It shows how predictable the classification is after getting information about an attribute.
- The lower the probability of error, the higher the predictability.
- The attribute with the minimal probability of error yields the maximum predictability. That is what we chose A at the root of the decision tree.

# Using information theory

- *Entropy* gives information about unpredictability.
- The scale is to use *1 bit* to answer a Boolean question with prior $< 0.5, 0.5 >$. This is least predictability (highest unpredicatability).
- Information answers questions: the more clueless we are about the answer initially, the more information is contained in the answer. i.e., we have a *gain* after getting an answer about attribute A.
- We select the attribute with the highest gain.
- Let $p$ be the number of positive examples, and $n$ the number of negative examples. Entropy$(p, n)$ is defined as

$$-p\log_2 p - n\log_2 n$$

# Information gain

- *Gain(A)* is the expected reduction on entropy after getting an answer on attribute A.
- Let $p_i$ be the number of positive examples when the answer to A is $i$, and $n_i$ be the number of negative examples when the answer to A is $i$.
- Assuming two possible answers, Gain(A) is defined as

$$\text{entropy}(p, n) - \frac{p_1 + n_1}{p + n}\text{entropy}(p_1, n_1) - \frac{p_2 + n_2}{p + n}\text{entropy}(p_2, n_2)$$

# Example

- Assuming two possible answers, Gain(A) is defined as

$$\text{entropy}(p,n) - \frac{p_1+n_1}{p+n}\text{entropy}(p_1,n_1) - \frac{p_2+n_2}{p+n}\text{entropy}(p_2,n_2)$$

- Initially there are 6 positive and 7 negative examples. Entropy(6,7) = 0.9957
- There are 6 positive and 2 negative examples for A being true and 0 positive and 5 negative example for A being false. Therefore the gain is

$$0.9957 - \frac{8}{13} \times \text{entropy}(6,2) - \frac{5}{13} \times \text{entropy}(5,0) =$$

$$0.9957 - \frac{8}{13} \times 0.8113 - \frac{5}{13} \times 0 = 0.4965$$

# Example(cont'd)

The gain values are:
A: 0.4992
B: 0.0414
C: 0.1307
D: 0.0349
E: 0.0069

# Summary

- Decision tree learning is a *supervised learning* paradigm.
- The *hypothesis* is a decision tree.
- The greedy algorithm uses *information gain* to decide which attribute should be placed at each node of the tree.
- Due to the greedy approach, the decision tree might not be optimal but the algorithm is fast.
- If the data set is complete and not noisy, then the learned decision tree will be accurate.

# Sources for the slides

- AIMA textbook ($3^{rd}$ edition)
- AIMA slides:
  http://aima.cs.berkeley.edu/
- Jean-Claude Latombe's CS121 slides
  http://robotics.stanford.edu/ latombe/cs121
  (Accessed prior to 2009)
- Wikipedia article for Twenty Questions
  http://en.wikipedia.org/wiki/Twenty_Questions
  (Accessed in March 2012)