INTRODUCTION TO DATA SCIENCE

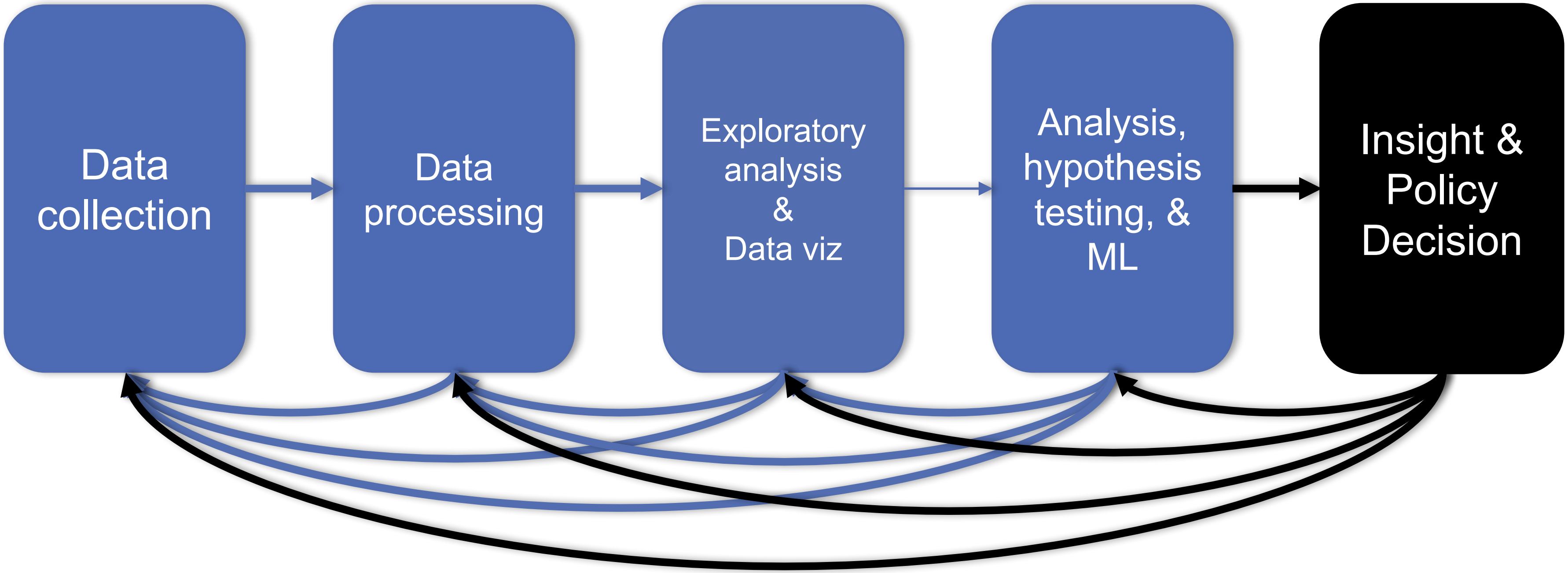
JOHN P DICKERSON

Lecture #20 – 10/31/2019
Lecture #21 – 11/5/2019

CMSC320
Tuesdays & Thursdays
5:00pm – 6:15pm

COMPUTER SCIENCE
UNIVERSITY OF MARYLAND

TODAY'S LECTURE


Data collection
Data processing
Exploratory analysis & Data viz
Analysis, hypothesis testing, & ML
Insight & Policy Decision

TODAY'S LECTURE

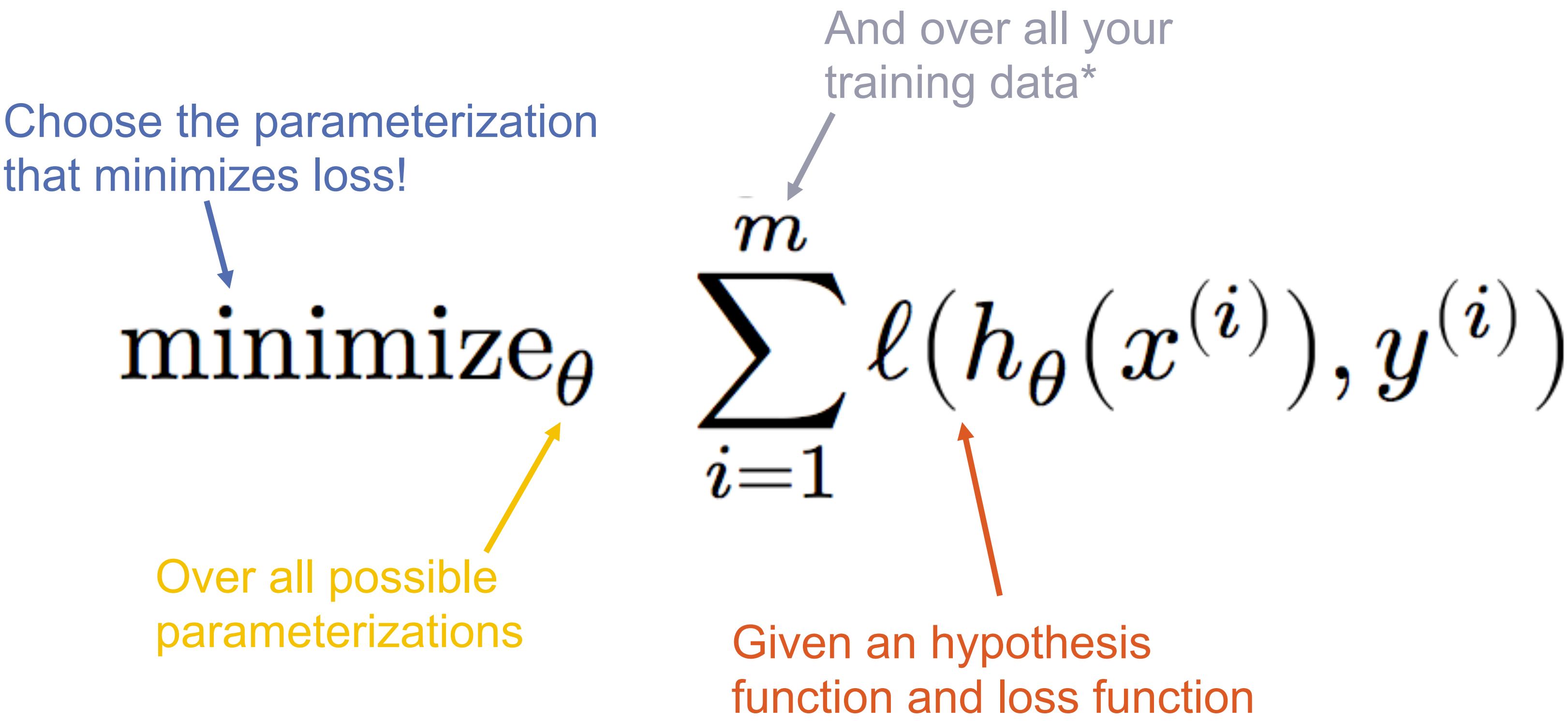
Decision Trees & Basic Model Evaluation

- What is a decision tree, and what can it represent?
- How do we learn tree structures?
- How do we tell if it's any good in practice?

BIG THANKS to Bart Selman's CS4700 lecture.

THE CANONICAL MACHINE LEARNING PROBLEM

At the end of the day, we want to learn a hypothesis function that predicts the actual outputs well.



And over all your training data*

Choose the parameterization that minimizes loss!

$$\text{minimize}_\theta \quad \sum_{i=1}^{m} \ell(h_\theta(x^{(i)}), y^{(i)})$$

Over all possible parameterizations

Given an hypothesis function and loss function

*Not actually what we want – want it over the world of inputs – will discuss later …

Big Picture of Learning

Learning can be seen as fitting a function to the data. We can consider different target functions and therefore different hypothesis spaces.

Examples:

Propositional if-then rules

Decision Trees

First-order if-then rules

First-order logic theory

Linear functions

Polynomials of degree at most k

Neural networks

Java programs

Turing machine

Etc

A learning problem is realizable if its hypothesis space contains the true function.

Tradeoff between expressiveness of a hypothesis space and the complexity of finding simple, consistent hypotheses within the space.

Decision Tree Learning

Task:

– Given: collection of examples (x, f(x))

– Return: a function *h* (*hypothesis*) that approximates *f*

– *h is a decision tree*

Input: an object or situation described by a set of attributes (or features)

Output: a "decision" – the predicts output value for the input.

The input attributes and the outputs can be discrete or continuous.

We will focus on decision trees for Boolean classification:

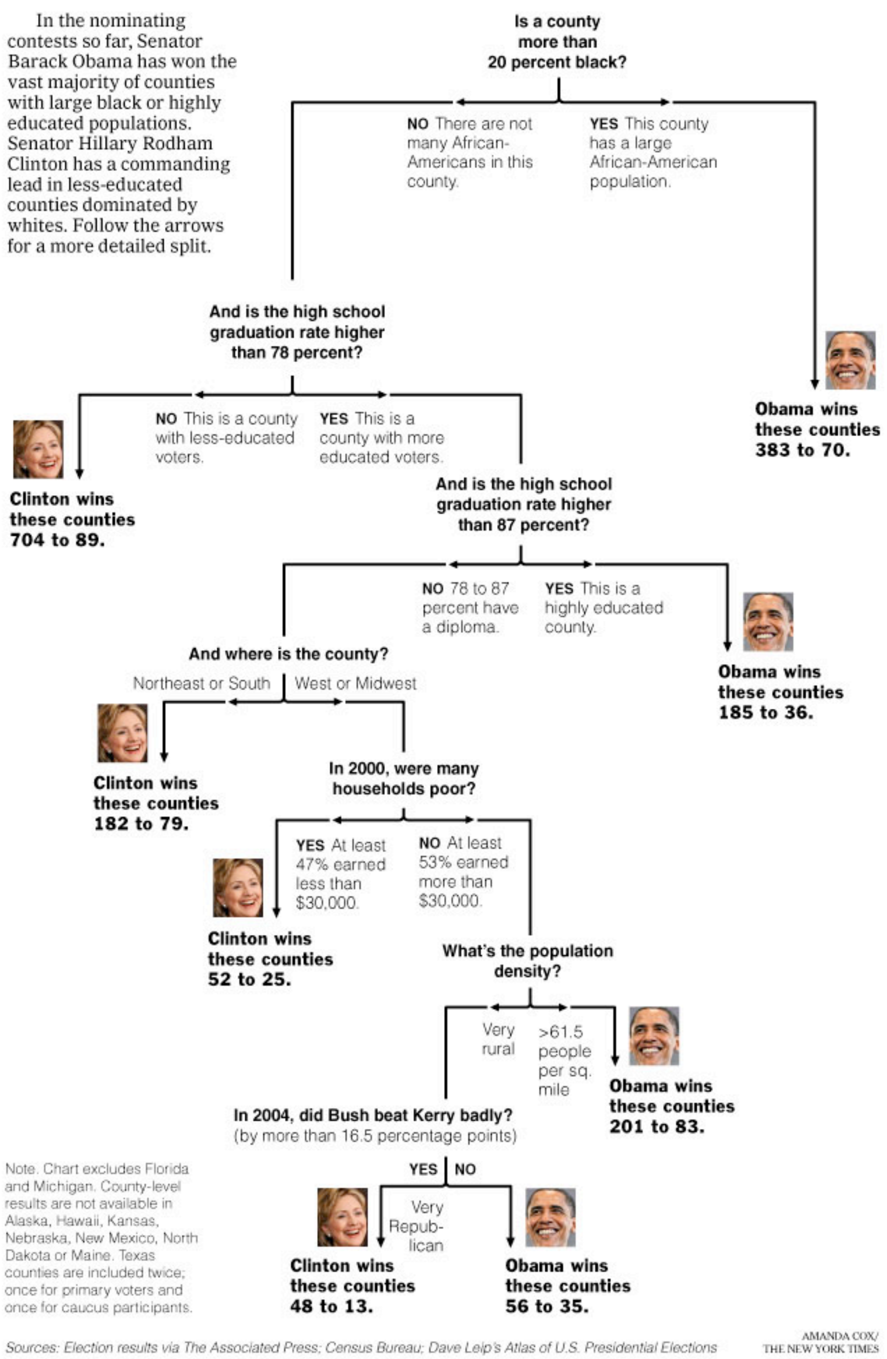
each example is classified as positive or negative.

Can we learn how counties vote?

New York Times
April 16, 2008

Decision Trees:
a sequence of tests.
Representation very natural for humans.
Style of many “How to” manuals and trouble-shooting procedures.

Decision Tree: The Obama-Clinton Divide



In the nominating contests so far, Senator Barack Obama has won the vast majority of counties with large black or highly educated populations. Senator Hillary Rodham Clinton has a commanding lead in less-educated counties dominated by whites. Follow the arrows for a more detailed split.

Is a county more than 20 percent black?

NO There are not many African-Americans in this county.

YES This county has a large African-American population.

Obama wins these counties 383 to 70.

And is the high school graduation rate higher than 78 percent?

NO This is a county with less-educated voters.

YES This is a county with more educated voters.

Clinton wins these counties 704 to 89.

And is the high school graduation rate higher than 87 percent?

NO 78 to 87 percent have a diploma.

YES This is a highly educated county.

Obama wins these counties 185 to 36.

And where is the county?

Northeast or South | West or Midwest

Clinton wins these counties 182 to 79.

In 2000, were many households poor?

YES At least 47% earned less than $30,000.

NO At least 53% earned more than $30,000.

Clinton wins these counties 52 to 25.

What's the population density?

Very rural

>61.5 people per sq. mile

Obama wins these counties 201 to 83.

In 2004, did Bush beat Kerry badly? (by more than 16.5 percentage points)

YES | **NO**

Very Repub-lican

Clinton wins these counties 48 to 13.

Obama wins these counties 56 to 35.

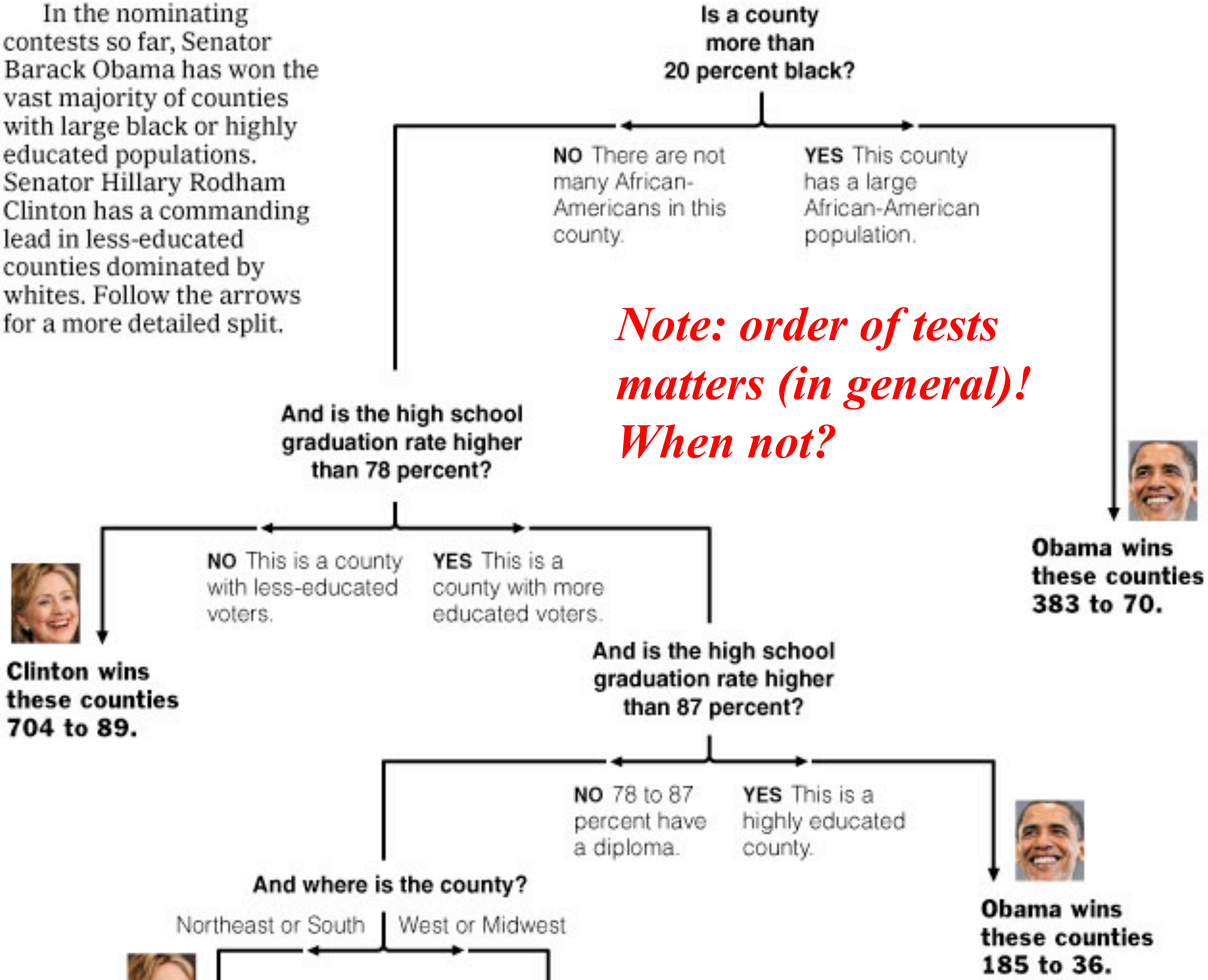
Note. Chart excludes Florida and Michigan. County-level results are not available in Alaska, Hawaii, Kansas, Nebraska, New Mexico, North Dakota or Maine. Texas counties are included twice; once for primary voters and once for caucus participants.

Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections

AMANDA COX/
THE NEW YORK TIMES

Decision Tree: The Obama-Clinton Divide

In the nominating contests so far, Senator Barack Obama has won the vast majority of counties with large black or highly educated populations. Senator Hillary Rodham Clinton has a commanding lead in less-educated counties dominated by whites. Follow the arrows for a more detailed split.



Is a county more than 20 percent black?

NO There are not many African-Americans in this county.

YES This county has a large African-American population.

Obama wins these counties 383 to 70.

And is the high school graduation rate higher than 78 percent?

NO This is a county with less-educated voters.

YES This is a county with more educated voters.

Clinton wins these counties 704 to 89.

And is the high school graduation rate higher than 87 percent?

NO 78 to 87 percent have a diploma.

YES This is a highly educated county.

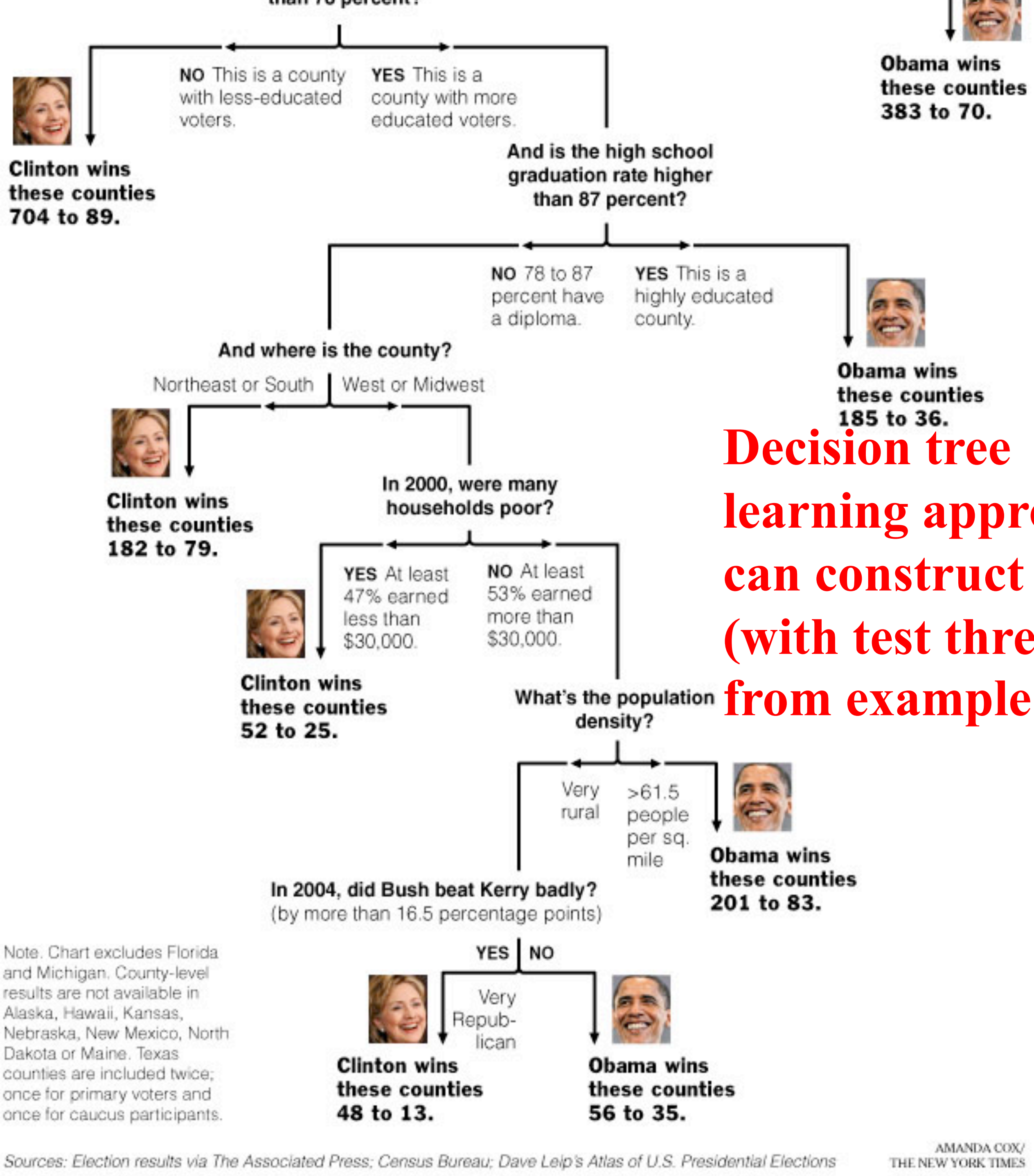
Obama wins these counties 185 to 36.

And where is the county?

Northeast or South

West or Midwest

Note: order of tests matters (in general)! When not?


than 78 percent?
NO This is a county with less-educated voters.
YES This is a county with more educated voters.
Obama wins these counties 383 to 70.
Clinton wins these counties 704 to 89.
And is the high school graduation rate higher than 87 percent?
NO 78 to 87 percent have a diploma.
YES This is a highly educated county.
Obama wins these counties 185 to 36.
And where is the county?
Northeast or South
West or Midwest
Clinton wins these counties 182 to 79.
In 2000, were many households poor?
YES At least 47% earned less than $30,000.
NO At least 53% earned more than $30,000.
Clinton wins these counties 52 to 25.
What's the population density?
Very rural
>61.5 people per sq. mile
Obama wins these counties 201 to 83.
In 2004, did Bush beat Kerry badly? (by more than 16.5 percentage points)
YES
NO
Very Repub-lican
Clinton wins these counties 48 to 13.
Obama wins these counties 56 to 35.
Note. Chart excludes Florida and Michigan. County-level results are not available in Alaska, Hawaii, Kansas, Nebraska, New Mexico, North Dakota or Maine. Texas counties are included twice; once for primary voters and once for caucus participants.
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections
AMANDA COX/ THE NEW YORK TIMES

Decision tree learning approach can construct tree (with test thresholds) from example counties.

Decision Tree

What is a decision tree?

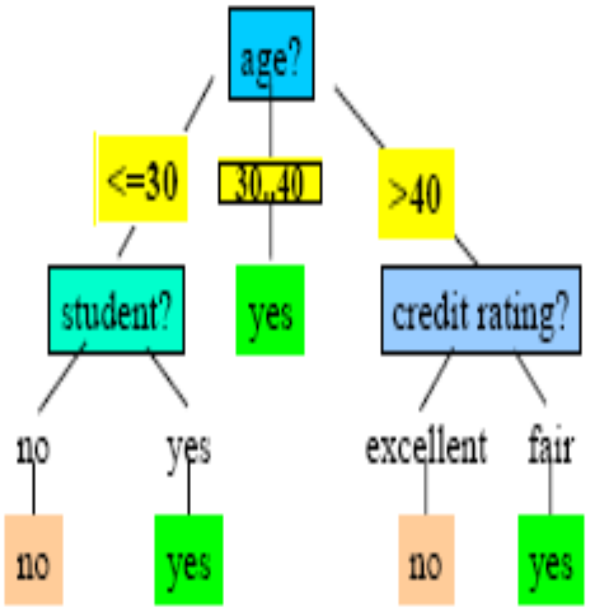
A tree with two types of nodes:

Decision nodes
Leaf nodes

Decision node: Specifies a choice or test of some attribute with 2 or more alternatives;
→ every decision node is part of a path to a leaf node

Leaf node: Indicates classification of an example

- Decision Tree example (is a customer going to buy a computer or not):



Inductive Learning Example

Food (3)	Chat (2)	Fast (2)	Price (3)	Bar (2)	BigTip
great	yes	yes	normal	no	yes
great	no	yes	normal	no	yes
mediocre	yes	no	high	no	no
great	yes	yes	normal	yes	yes

Etc.

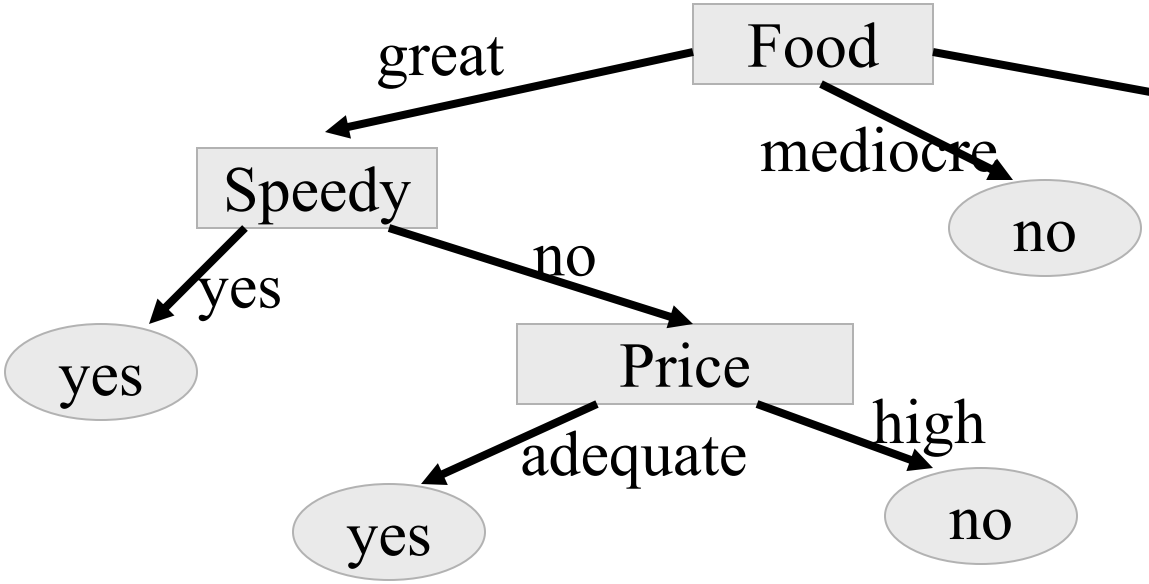
Instance Space X: Set of all possible objects described by attributes (often called features).

Target Function f: Mapping from Attributes to Target Feature (often called label) (f is unknown)

Hypothesis Space H: Set of all classification rules h_i we allow.

Training Data D: Set of instances labeled with Target Feature

Decision Tree Example: “BigTip”



Our data

F S P	BigTip
$\vec{x}_1 = ($ g, y, a $)$	$f(\vec{x}_1) = 1$
$\vec{x}_2 = ($ g, n, h $)$	$f(\vec{x}_2) = 0$
$\vec{x}_3 = ($ g, y, h $)$	$f(\vec{x}_3) = 1$
$\vec{x}_4 = ($ g, n, a $)$	$f(\vec{x}_4) = 1$
$\vec{x}_5 = ($ m, y, a $)$	$f(\vec{x}_5) = 0$
$\vec{x}_6 = ($ y, y, a $)$	$f(\vec{x}_6) = 0$
$\vec{x}_7 = ($ g, y, a $)$	$f(\vec{x}_7) = 1$
$\vec{x}_8 = ($ g, y, h $)$	$f(\vec{x}_8) = 1$
$\vec{x}_9 = ($ m, y, a $)$	$f(\vec{x}_9) = 0$
$\vec{x}_{10} = ($ g, y, a $)$	$f(\vec{x}_{10}) = 1$

Is the decision tree we learned consistent?

Yes, it agrees with all the examples!

Data: Not all 2x2x3 = 12 tuples
Also, some repeats! These are literally “observations.”

Learning decision trees: An example

Problem: decide whether to wait for a table at a restaurant. What attributes would you use?

Attributes used by R&N

1. Alternate: is there an alternative restaurant nearby?
2. Bar: is there a comfortable bar area to wait in?
3. Fri/Sat: is today Friday or Saturday?
4. Hungry: are we hungry?
5. Patrons: number of people in the restaurant (None, Some, Full)
6. Price: price range ($, $$, $$$)
7. Raining: is it raining outside?
8. Reservation: have we made a reservation?
9. Type: kind of restaurant (French, Italian, Thai, Burger)
10. WaitEstimate: estimated waiting time (0-10, 10-30, 30-60, >60)

Goal predicate: WillWait?

What about restaurant name?

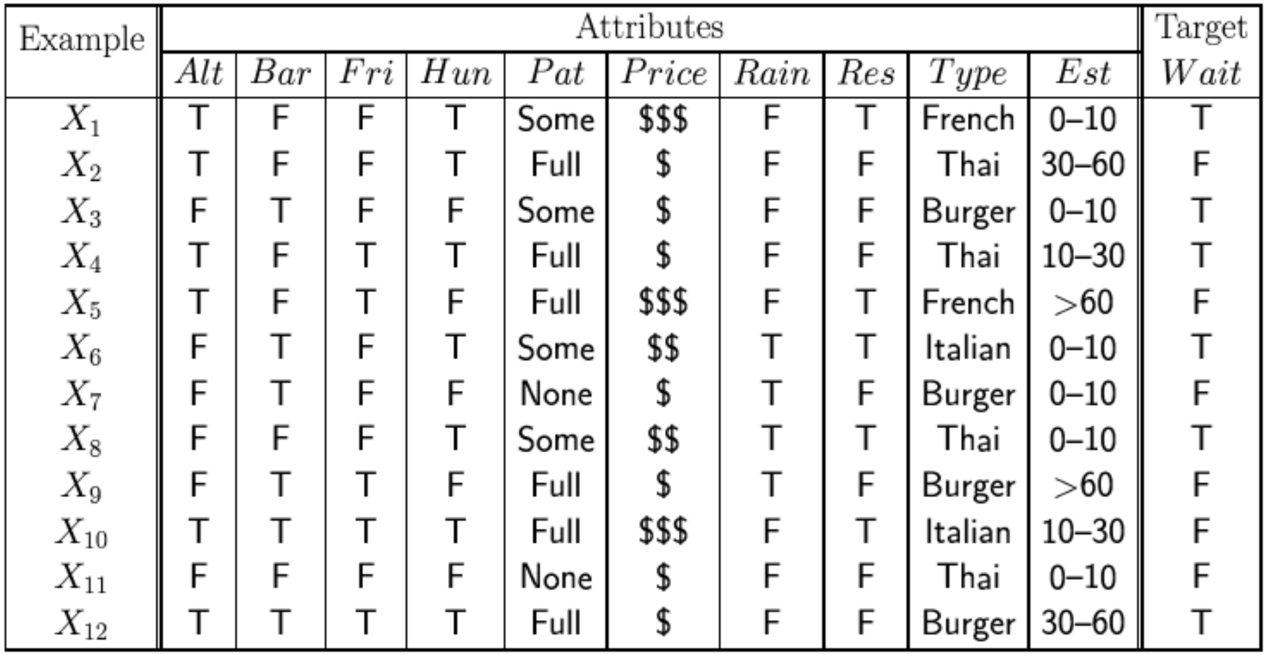
It could be great for generating a small tree but …

It doesn’t generalize!

Attribute-based representations

Examples described by attribute values (Boolean, discrete, continuous)

E.g.



Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	\$\$\$	F	T	French	0–10	T
X_2	T	F	F	T	Full	\$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	\$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	\$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	\$\$\$	F	T	French	>60	F
X_6	F	T	F	T	Some	\$\$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	\$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	\$\$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	\$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	\$\$\$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	\$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	\$	F	F	Burger	30–60	T

12 examples

6 +

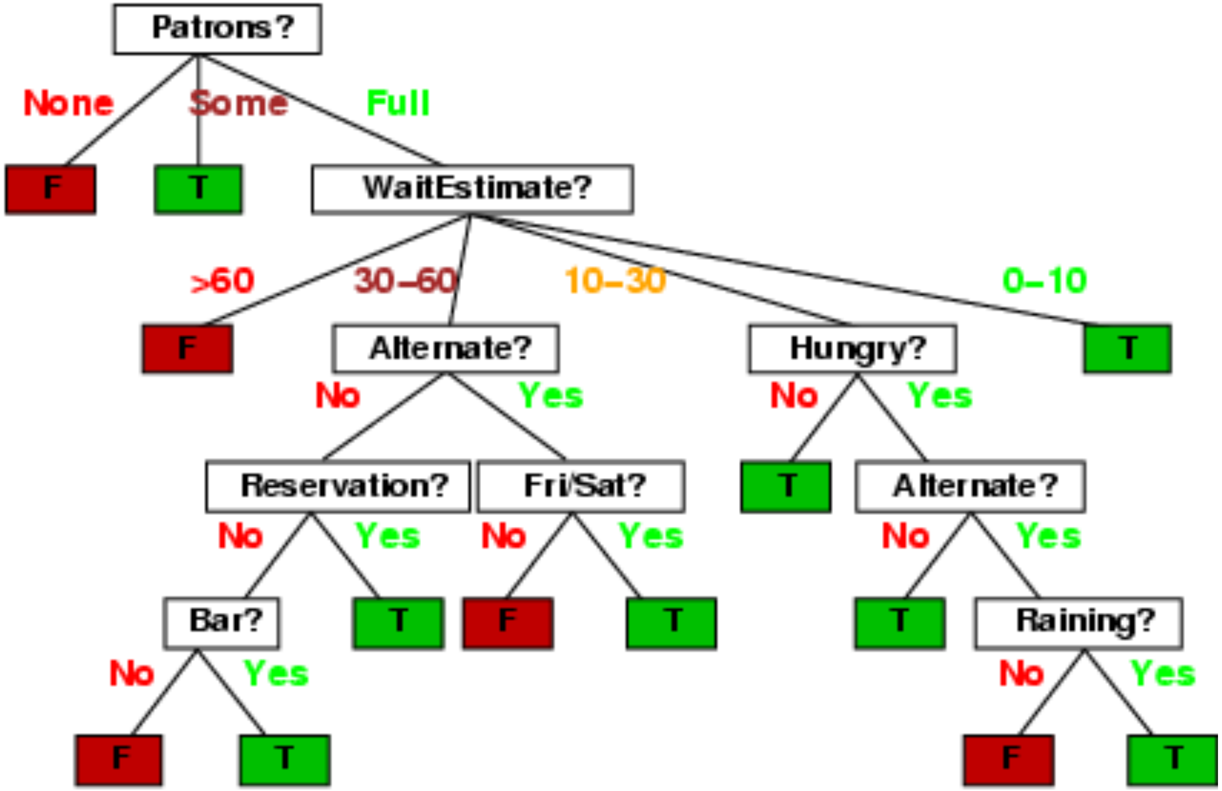
6 -

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

Decision trees

One possible representation for hypotheses

E.g., here is a tree for deciding whether to wait:



Expressiveness of Decision Trees

Any particular decision tree hypothesis for WillWait goal predicate can be seen as a disjunction of a conjunction of tests, i.e., an assertion of the form:

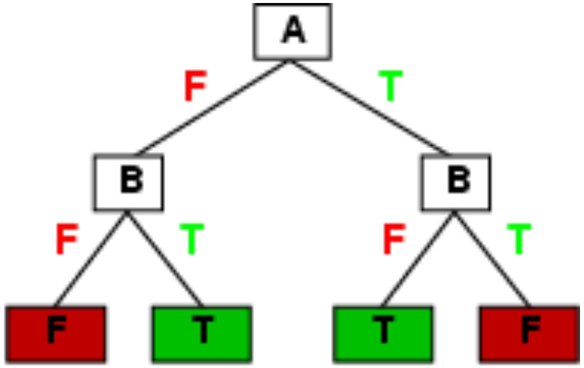
$$\forall s \;\; WillWait(s) \leftrightarrow (P1(s) \vee P2(s) \vee \ldots \vee Pn(s))$$

Where each condition Pi(s) is a conjunction of tests corresponding to the path from the root of the tree to a leaf with a positive outcome.

Expressiveness

Decision trees can express any Boolean function of the input attributes.
E.g., for Boolean functions, truth table row → path to leaf:

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



Number of Distinct Decision Trees

How many distinct decision trees with *10* Boolean attributes?

= number of Boolean functions with 10 propositional symbols

Input features	Output
0 0 0 0 0 0 0 0 0 0	0/1
0 0 0 0 0 0 0 0 0 1	0/1
0 0 0 0 0 0 0 0 1 0	0/1
0 0 0 0 0 0 0 1 0 0	0/1
…	
1 1 1 1 1 1 1 1 1 1	0/1

How many entries does this table have?

2^{10}

So how many Boolean functions with 10 Boolean attributes are there, given that each entry can be 0/1?

$$= 2^{2^{10}}$$

Hypothesis spaces

How many distinct decision trees with *n* Boolean attributes?

= number of Boolean functions

= number of distinct truth tables with 2^n rows = 2^{2^n}

E.g. how many Boolean functions on 6 attributes? A lot…

With 6 Boolean attributes, there are 18,446,744,073,709,551,616 possible trees!

Many calculators can't handle 10 attributes ☺!

There are even more decision trees! (see later)

Decision tree learning Algorithm

Decision trees can express any Boolean function.

Goal: Finding a decision tree that agrees with training set.

We could construct a decision tree that has one path to a leaf for each example, where the path tests sets each attribute value to the value of the example.

What is the problem with this from a learning point of view?

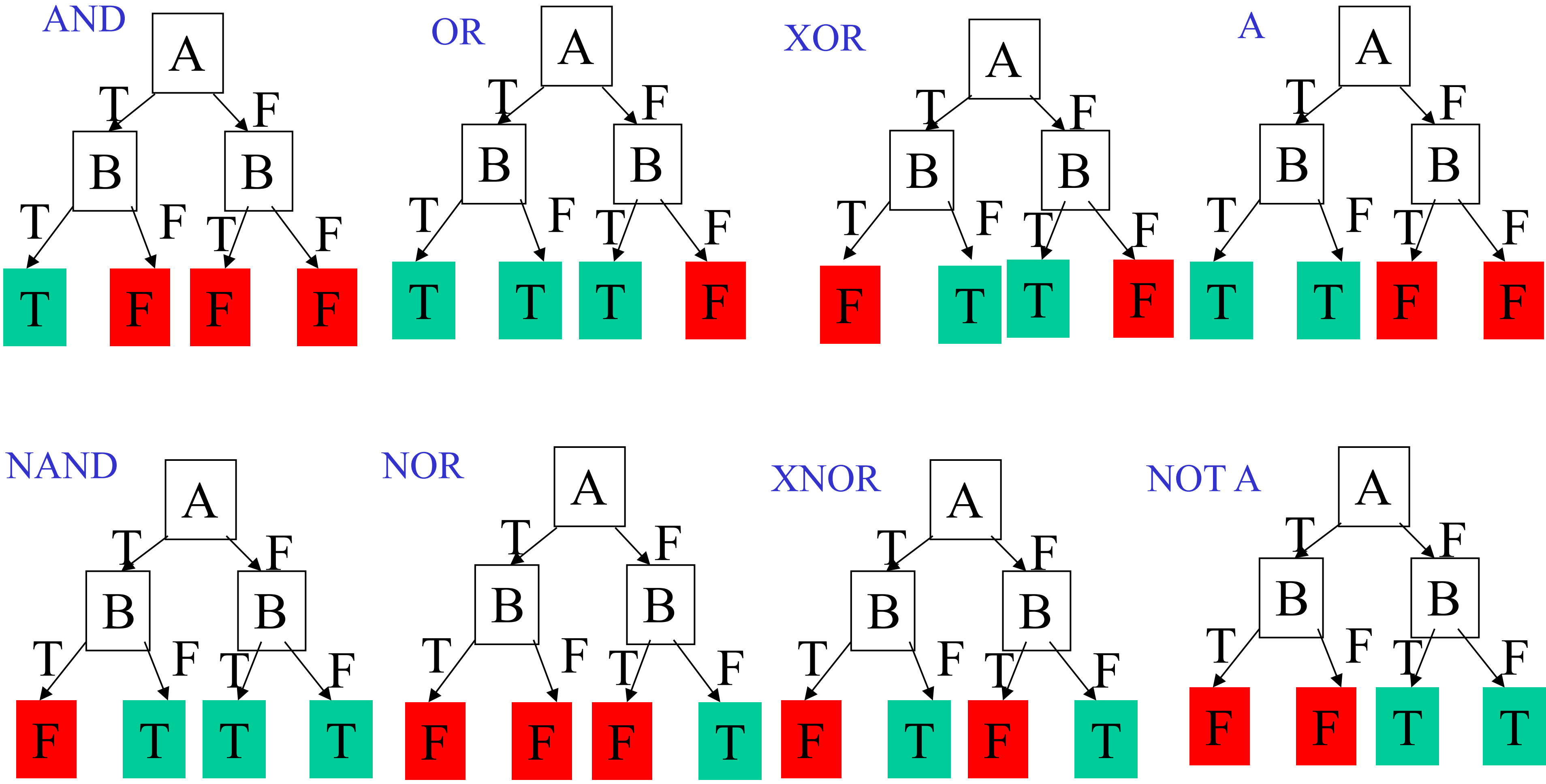
Problem: This approach would just memorize example.
How to deal with new examples? It doesn't generalize!

(But sometimes hard to avoid --- e.g. parity function, 1, if an even number of inputs, or majority function, 1, if more than half of the inputs are 1).

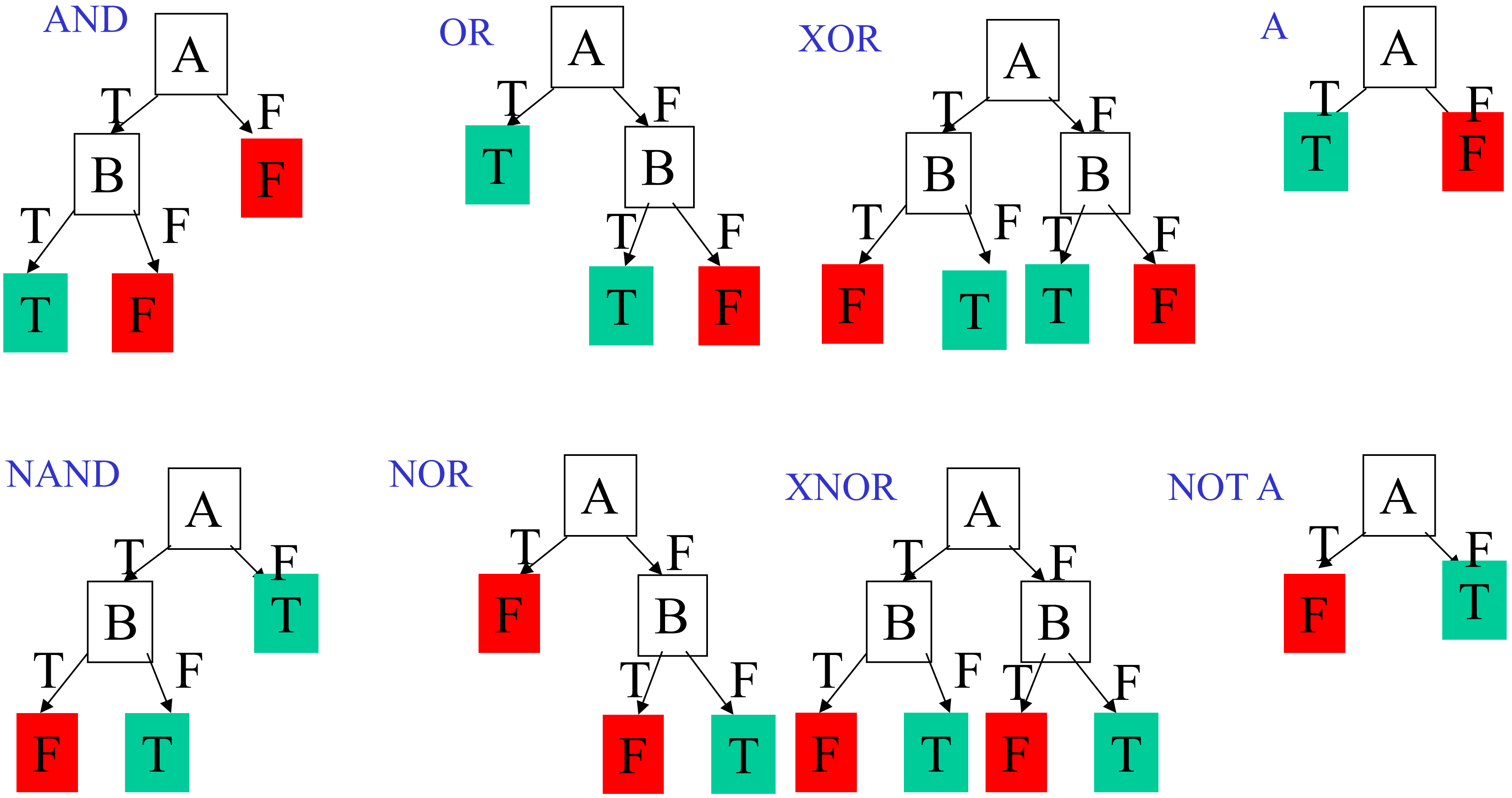
We want a compact/smallest tree.
But finding the smallest tree consistent with the examples is NP-hard!

Overall Goal: get a good classification with a small number of tests.

Expressiveness: Boolean Function with 2 attributes → 2^{2^2} DTs

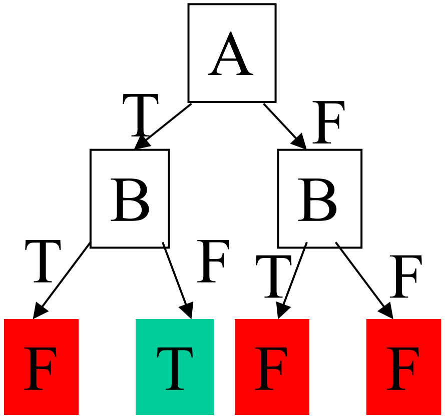


Expressiveness: 2 attribute → 2^{2^2} DTs

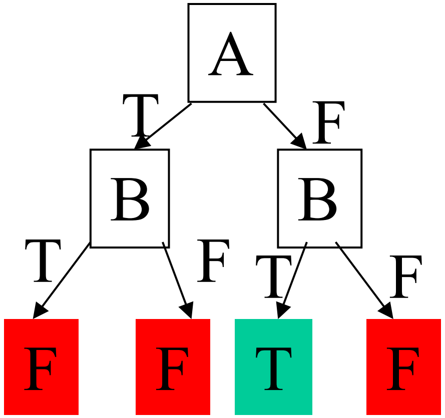


Expressiveness:
2 attribute → 2^{2^2} DTs

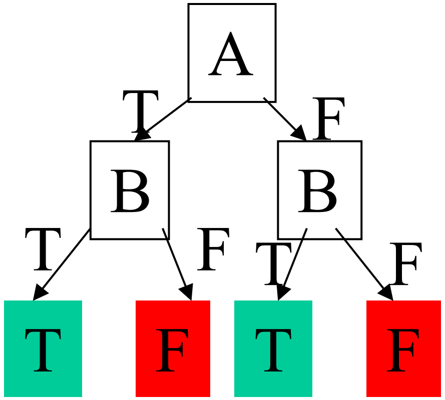
A AND-NOT B



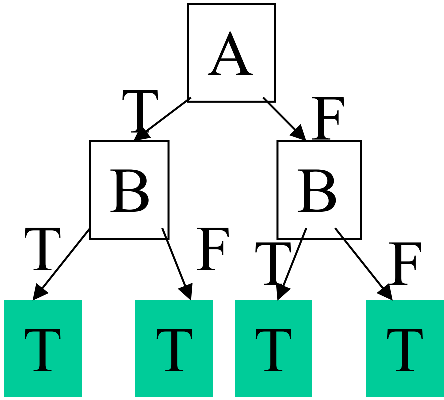
NOT A AND B



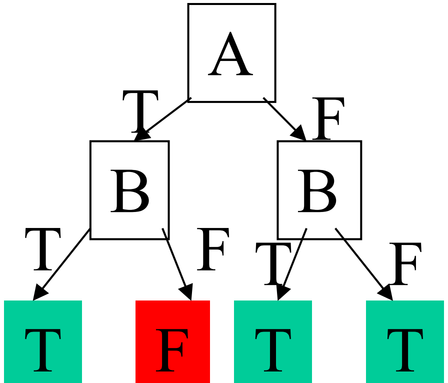
B



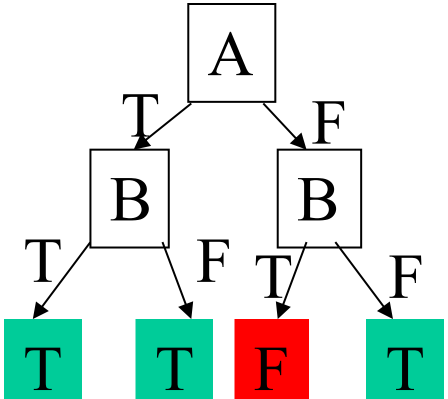
TRUE



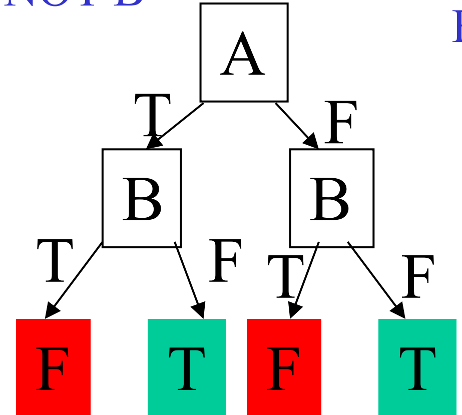
A OR NOT B



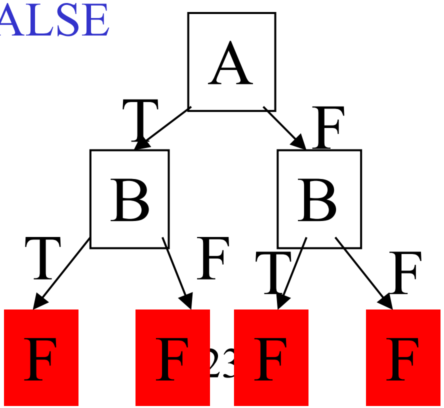
NOR A OR B



NOT B

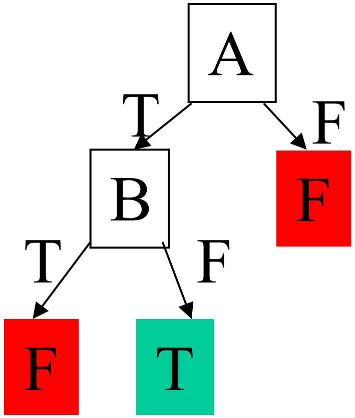


FALSE

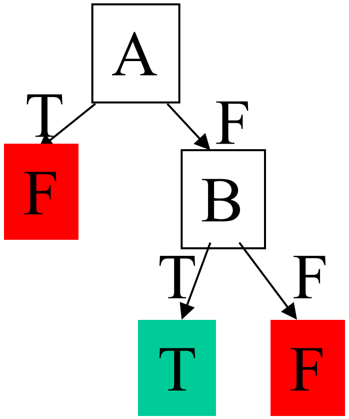


Expressiveness:
2 attribute → 2^{2^2} DTs

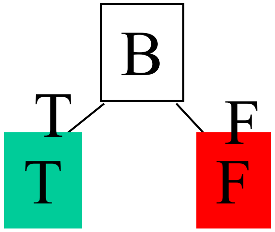
A AND-NOT B



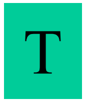
NOT A AND B



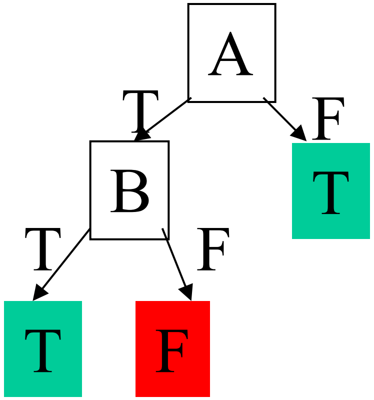
B



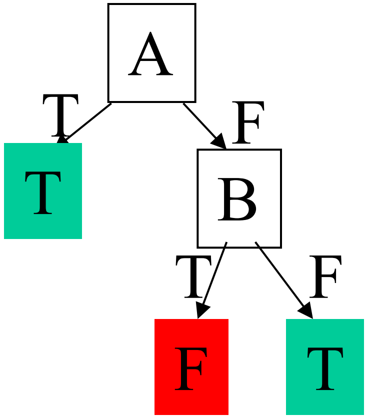
TRUE



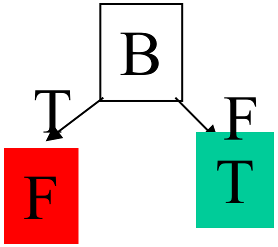
A OR NOT B



NOR A OR B



NOT B



FALSE

“most significant”
In what sense?

Basic DT Learning Algorithm

Goal: find a *small* tree consistent with the training examples

Idea: (recursively) choose "most significant" attribute as root of (sub)tree;

Use a top-down greedy search through the space of possible decision trees.

Greedy because there is no backtracking. It picks highest values first.

Variations of known algorithms ID3, C4.5 (Quinlan -86, -93)

Top-down greedy construction

– Which attribute should be tested?
    - Heuristics and Statistical testing with current data
– Repeat for descendants

(ID3 Iterative Dichotomiser 3)

Big Tip Example

10 examples:

6+ 1 3 4 7 8 10

4- 2 5 6 9

Attributes:

•Food with values g,m,y

•Speedy? with values y,n

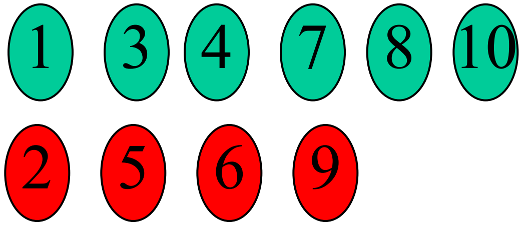
•Price, with values a, h

F S P	BigTip
$\vec{x}_1 = ($ g, y, a $)$	$f(\vec{x}_1) = 1$
$\vec{x}_2 = ($ g, n, h $)$	$f(\vec{x}_2) = 0$
$\vec{x}_3 = ($ g, y, h $)$	$f(\vec{x}_3) = 1$
$\vec{x}_4 = ($ g, n, a $)$	$f(\vec{x}_4) = 1$
$\vec{x}_5 = ($ m, y, a $)$	$f(\vec{x}_5) = 0$
$\vec{x}_6 = ($ y, y, a $)$	$f(\vec{x}_6) = 0$
$\vec{x}_7 = ($ g, y, a $)$	$f(\vec{x}_7) = 1$
$\vec{x}_8 = ($ g, y, h $)$	$f(\vec{x}_8) = 1$
$\vec{x}_9 = ($ m, y, a $)$	$f(\vec{x}_9) = 0$
$\vec{x}_{10} = ($ g, y, a $)$	$f(\vec{x}_{10}) = 1$

Let’ s build our decision tree starting with the attribute Food, (3 possible values: g, m, y).

Top-Down Induction of Decision Tree: Big Tip Example

Node “done” when uniform label or “no further uncertainty.”

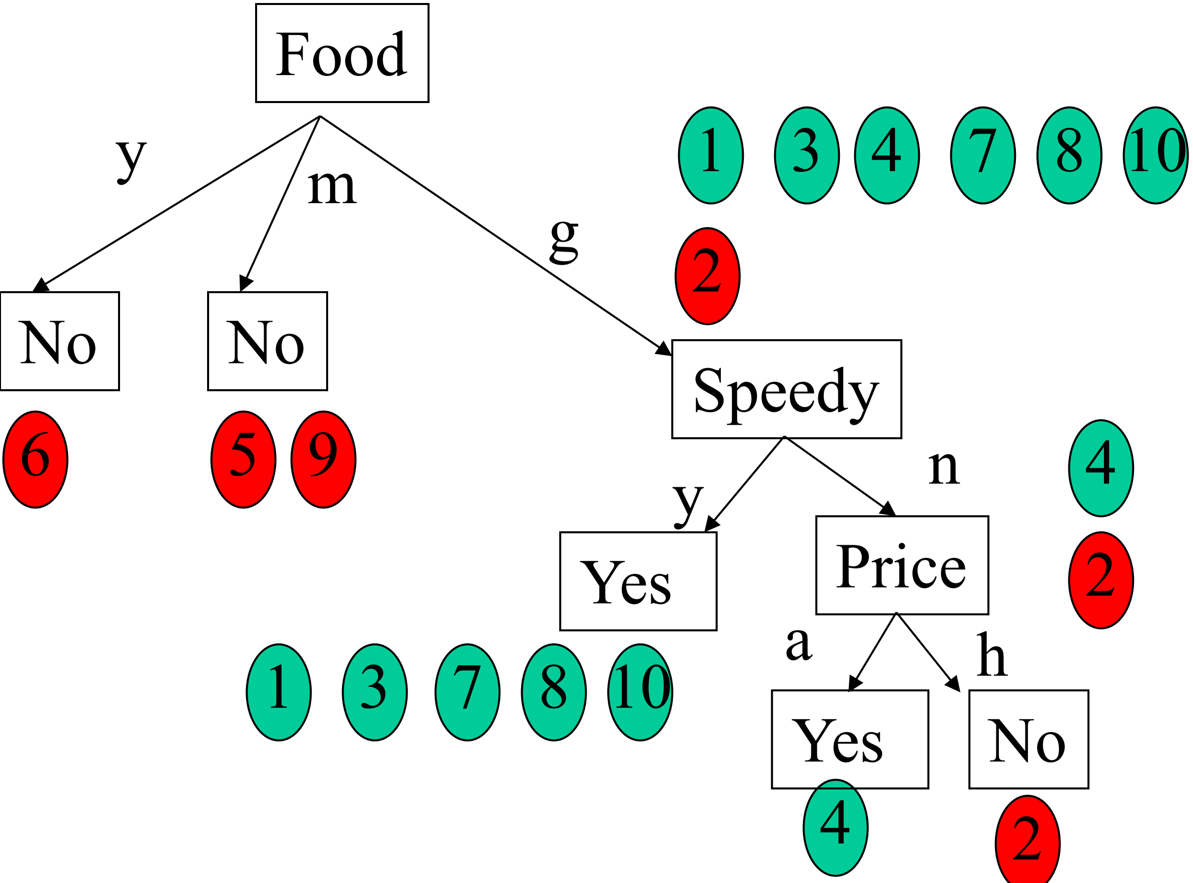


10 examples:

🟢 🟢 🟢 🟢 🟢 🟢 6+

🔴 🔴 🔴 🔴 4-

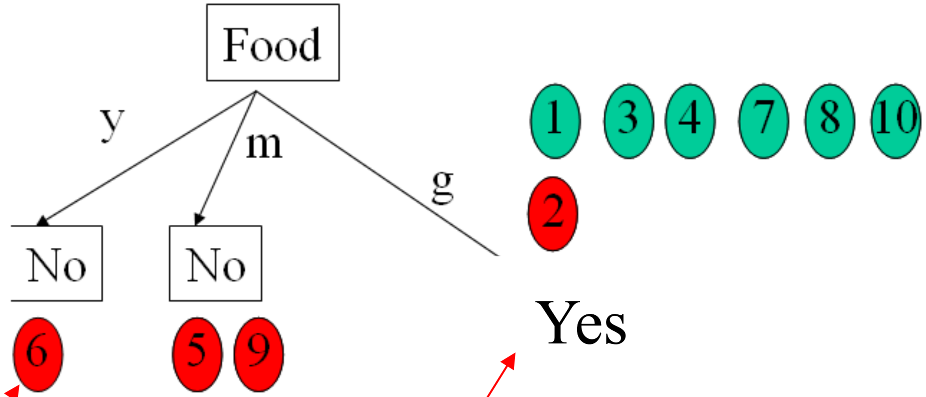
F S P	BigTip
$\vec{x}_1 = ($ g, y, a $)$	$f(\vec{x}_1) = 1$
$\vec{x}_2 = ($ g, n, h $)$	$f(\vec{x}_2) = 0$
$\vec{x}_3 = ($ g, y, h $)$	$f(\vec{x}_3) = 1$
$\vec{x}_4 = ($ g, n, a $)$	$f(\vec{x}_4) = 1$
$\vec{x}_5 = ($ m, y, a $)$	$f(\vec{x}_5) = 0$
$\vec{x}_6 = ($ y, y, a $)$	$f(\vec{x}_6) = 0$
$\vec{x}_7 = ($ g, y, a $)$	$f(\vec{x}_7) = 1$
$\vec{x}_8 = ($ g, y, h $)$	$f(\vec{x}_8) = 1$
$\vec{x}_9 = ($ m, y, a $)$	$f(\vec{x}_9) = 0$
$\vec{x}_{10} = ($ g, y, a $)$	$f(\vec{x}_{10}) = 1$



How many + and - examples per subclass, starting with y?

Let’ s consider next the attribute Speedy

Top-Down Induction of DT (simplified)



```
TDIDF(D,$c_{def}$)

IF(all examples in D have same class c)
    – Return leaf with class c (or class $c_{def}$, if D is empty)
ELSE IF(no attributes left to test)
    – Return leaf with class c of majority in D
ELSE
    – Pick A as the "best" decision attribute for next node
    – FOR each value $v_i$ of A create a new descendent of node
        • $D_i = \{(\vec{x}, y) \in D : \text{attribute A of } \vec{x} \text{ has value } v_i\}$
        • Subtree $t_i$ for $v_i$ is TDIDT($D_i$,$c_{def}$)

    – RETURN tree with A as root and $t_i$ as subtrees
```

Training Data: $D = \{(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)\}$

Picking the Best Attribute to Split

Ockham's Razor:

- All other things being equal, choose the simplest explanation

Decision Tree Induction:

- Find the smallest tree that classifies the training data correctly

Problem

- Finding the smallest tree is computationally hard ☹!

Approach

- Use heuristic search (greedy search)

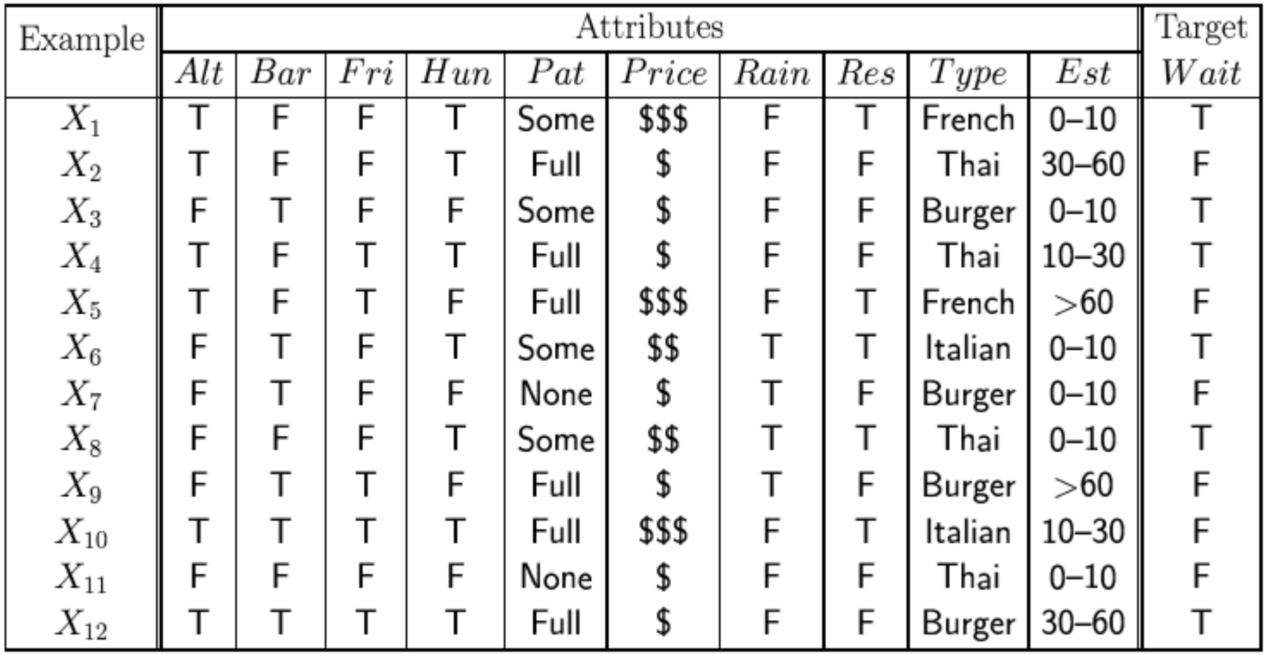
Key Heuristics:

- Pick attribute that *maximizes information (Information Gain)* ←
  i.e. "most informative"
- Other statistical tests

Attribute-based representations

Examples described by attribute values (Boolean, discrete, continuous)

E.g.



Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

12 examples

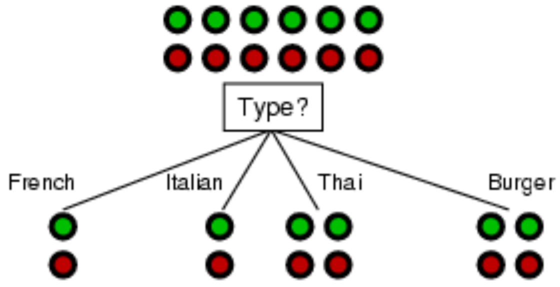
6 +

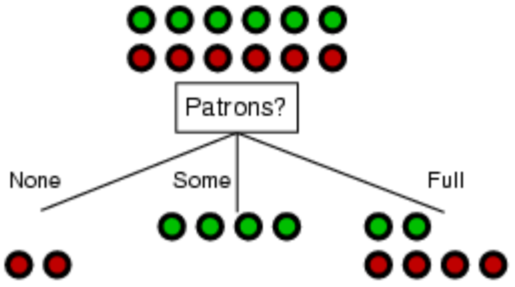
6 -

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

Choosing an attribute: Information Gain

Goal: trees with short paths to leaf nodes





Is this a good attribute to split on?

Which one should we pick?

A perfect attribute would ideally divide the examples into sub-sets that are all positive or all negative… i.e. maximum information gain.

Information Gain

Most useful in classification

- how to measure the ‘worth’ of an attribute ***information gain***
- how well attribute separates examples according to their classification

Next

- precise definition for gain

→ measure from Information Theory

Shannon and Weaver 49

One of the most successful and impactful mathematical theories known.

Information

"Information" answers questions.

The more clueless I am about a question, the more information the **answer** to the question contains.

Example – fair coin → prior <0.5,0.5>

By definition Information of the prior (or entropy of the prior):

$$I(P1,P2) = - P1 \log_2(P1) – P2 \log_2(P2) =$$
$$I(0.5,0.5) = -0.5 \log_2(0.5) – 0.5 \log_2(0.5) = 1$$

We need 1 bit to convey the outcome of the flip of a fair coin.

Scale: 1 bit = answer to Boolean question with prior <0.5, 0.5>

Why does a biased coin have less information?
(How can we code the outcome of a biased coin sequence?)

Information (or Entropy)

Information in an answer given possible answers $v_1, v_2, \ldots v_n$:

$$I(P(v_1), \ldots, P(v_n)) = \sum_{i=1}^{n} -P(v_i)\ \ log_2(P(v_i))$$

— $v_1, \ldots, v_n$ possible answers

— $P(v_i)$ probabibilty of answer v_i

(Also called entropy of the prior.)

Example – biased coin → prior <1/100,99/100>

$I(1/100,99/100) = -1/100 \log_2(1/100) – 99/100 \log_2(99/100)$
= 0.08 bits (so not much information gained from "answer.")

Example – fully biased coin → prior <1,0>

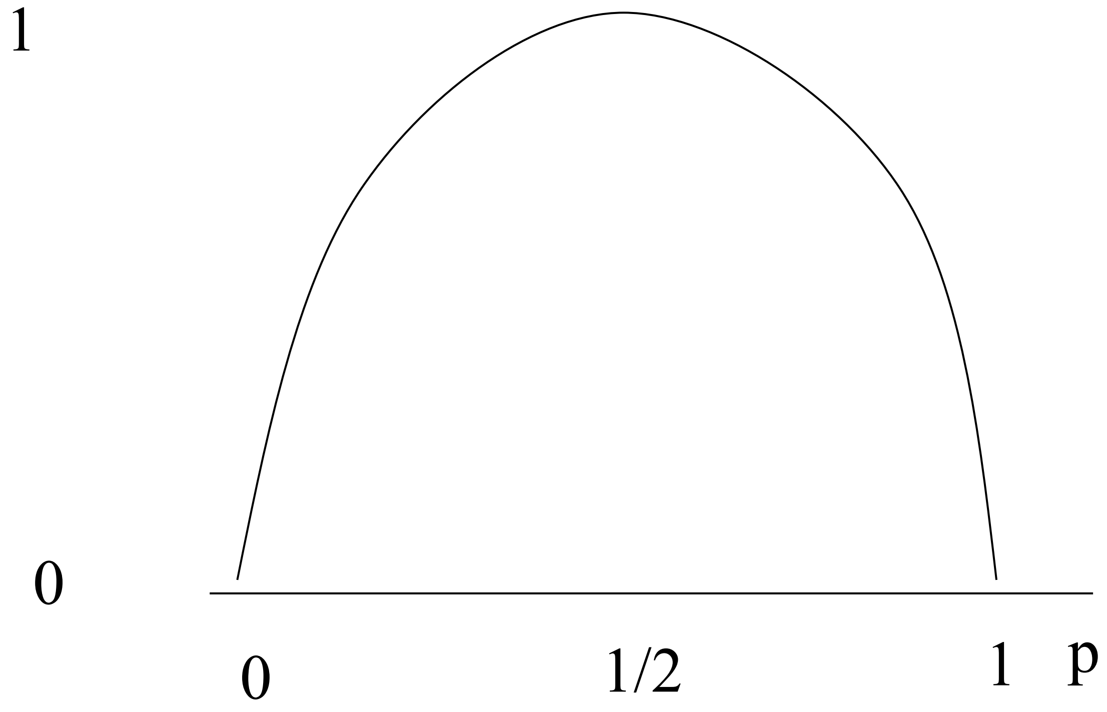
$I(1,0) = -1 \log_2(1) – 0 \log_2(0) = 0$ bits

$0 \log_2(0) \ = 0$

i.e., no uncertainty left in source!

Shape of Entropy Function

Roll of an unbiased die



The more uniform the probability distribution, the greater is its entropy.

Information or Entropy

Information or Entropy measures the "randomness" of an arbitrary collection of examples.

We don't have exact probabilities but our training data provides an estimate of the probabilities of positive vs. negative examples given a set of values for the attributes.

For a collection S, entropy is given as:

$$I(\frac{p}{p+n}, \frac{n}{p+n}) = -\frac{p}{p+n} \ log_2(\frac{p}{p+n}) - \frac{n}{p+n} \ log_2(\frac{n}{p+n})$$

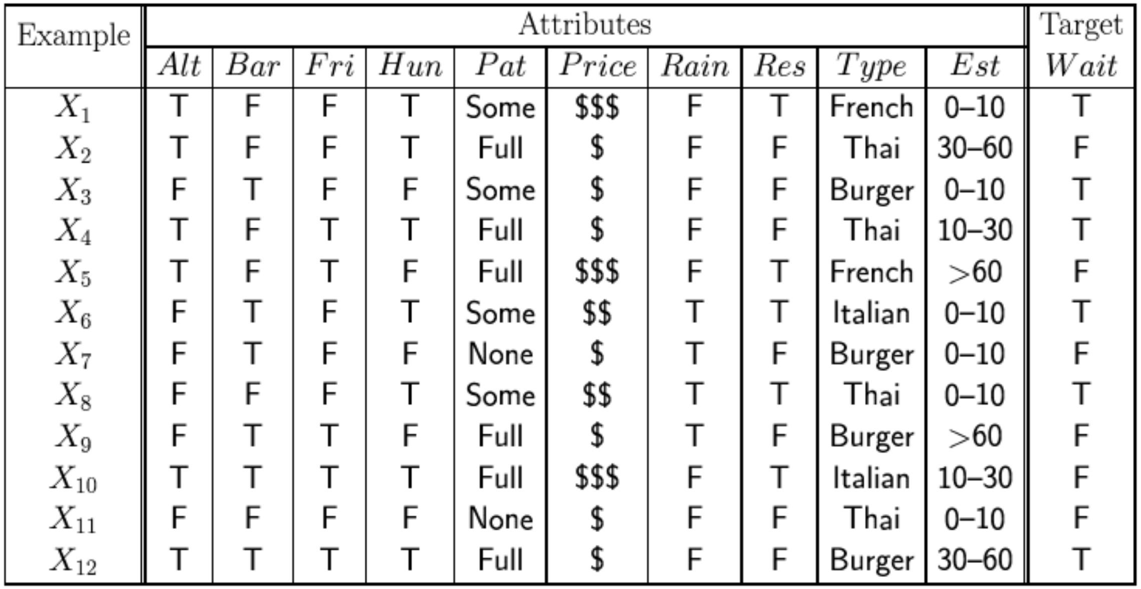
For a collection S having positive and negative examples

p - # positive examples;

n - # negative examples

Attribute-based representations

Examples described by attribute values (Boolean, discrete, continuous)
E.g., situations where I will/won't wait for a table:



Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*Wait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

12 examples
6 +
6 -

What' s the entropy of this collection of examples?

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

p = n = 6; I(0.5,0.5) = -0.5 log2(0.5) –0.5 log2(0.5) = 1

So, we need 1 bit of info to classify a randomly picked example, assuming no other information is given about the example.

Choosing an attribute: Information Gain

Intuition: Pick the attribute that reduces the entropy (the uncertainty) the most.

So we measure the information gain after testing a given attribute A:

$$Gain(A) = I\left(\frac{p}{p+n}, \frac{n}{p+n}\right) - \boxed{Remainder(A)}$$

Remainder(A) → gives us the remaining uncertainty after getting info on attribute A.

Choosing an attribute: Information Gain

Remainder(A)

→ gives us the amount information we still need after testing on A.

Assume A divides the training set E into E_1, E_2, … E_v, corresponding to the different v distinct values of A.

Each subset E_i has p_i positive examples and n_i negative examples.

So for total information content, we need to weigh the contributions of the different subclasses induced by A

Weight (relative size) of each subclass

$$Remainder(A) = \sum_{i=1}^{v} \frac{p_i+n_i}{p+n} \; I\left(\frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i}\right)$$

Choosing an attribute: Information Gain

Measures the expected reduction in entropy. The higher the Information Gain (IG), or just Gain, with respect to an attribute A , the more is the expected reduction in entropy.

Weight of each subclass

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

where Values(A) is the set of all possible values for attribute A,

S_v is the subset of S for which attribute A has value v.

Interpretations of gain

Gain(S,A)

- expected reduction in entropy caused by knowing A
- information provided about the target function value given the value of A
- number of bits saved in the coding a member of S knowing the value of A

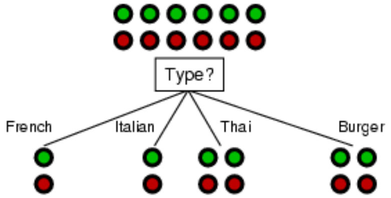
Used in ID3 (Iterative Dichotomiser 3) Ross Quinlan

What if we used attribute “example label” uniquely specifying the answer? Info gain? Issue?
High branching: can correct with “info gain ratio”

Information gain

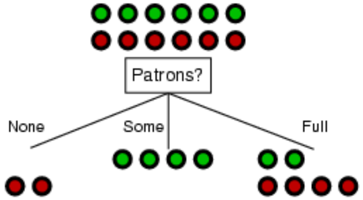
For the training set, $p = n = 6$, $I(6/12, 6/12) = 1$ bit

Consider the attributes *Type* and *Patrons*:



Info gain?

$$IG(Type) = 1 - [\frac{2}{12} I(\frac{1}{2},\frac{1}{2}) + \frac{2}{12} I(\frac{1}{2},\frac{1}{2}) + \frac{4}{12} I(\frac{2}{4},\frac{2}{4}) + \frac{4}{12} I(\frac{2}{4},\frac{2}{4})] = 0 \text{ bits}$$



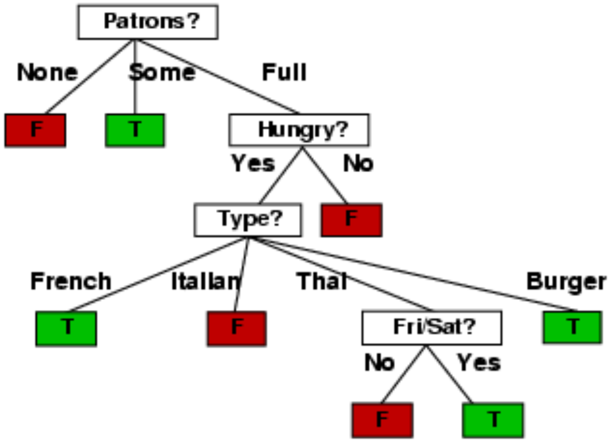
$$IG(Patrons) = 1 - [\frac{2}{12} I(0,1) + \frac{4}{12} I(1,0) + \frac{6}{12} I(\frac{2}{6},\frac{4}{6})] = .0541 \text{ bits}$$

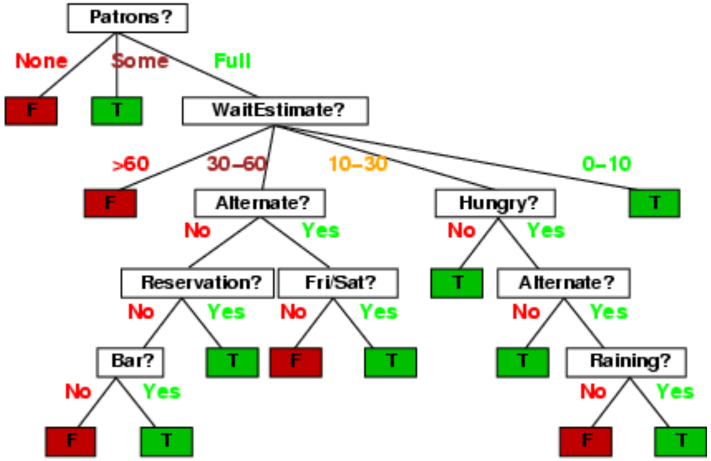
Patrons **has the highest IG of all attributes and so is chosen by the DTL algorithm as the root.**

Example contd.

Decision tree learned from the 12 examples:

“personal R&N Tree”





Substantially simpler than “true” tree --- but a more complex hypothesis isn’t justified from just the data.

Inductive Bias

Roughly: prefer

- shorter trees over deeper/more complex ones
  - E.g., Occam's Razor
- ones with high gain attributes near root

Difficult to characterize precisely

- attribute selection heuristics
- interacts closely with given data

Evaluation Methodology
General for Machine Learning

Evaluation Methodology

How to evaluate the quality of a learning algorithm, i.e.,:

How good are the hypotheses produced by the learning algorithm?
How good are they at classifying unseen examples?

Standard methodology **("Holdout Cross-Validation")**:

1. Collect a large set of examples.
2. Randomly divide collection into two disjoint sets: **training set** and **test set**.
3. Apply learning algorithm to training set generating hypothesis *h*
4. Measure performance of *h* w.r.t. test set (a form of cross-validation)

→ measures generalization to unseen data

Important: keep the training and test sets disjoint! "No peeking"!

Note: The first two questions about any learning result: Can you describe your training and your test set? What's your error on the test set?

Peeking

Example of peeking:

We generate four different hypotheses – for example by using different criteria to pick the next attribute to branch on.

We test the performance of the four different hypothesis on the test set and we select the best hypothesis.

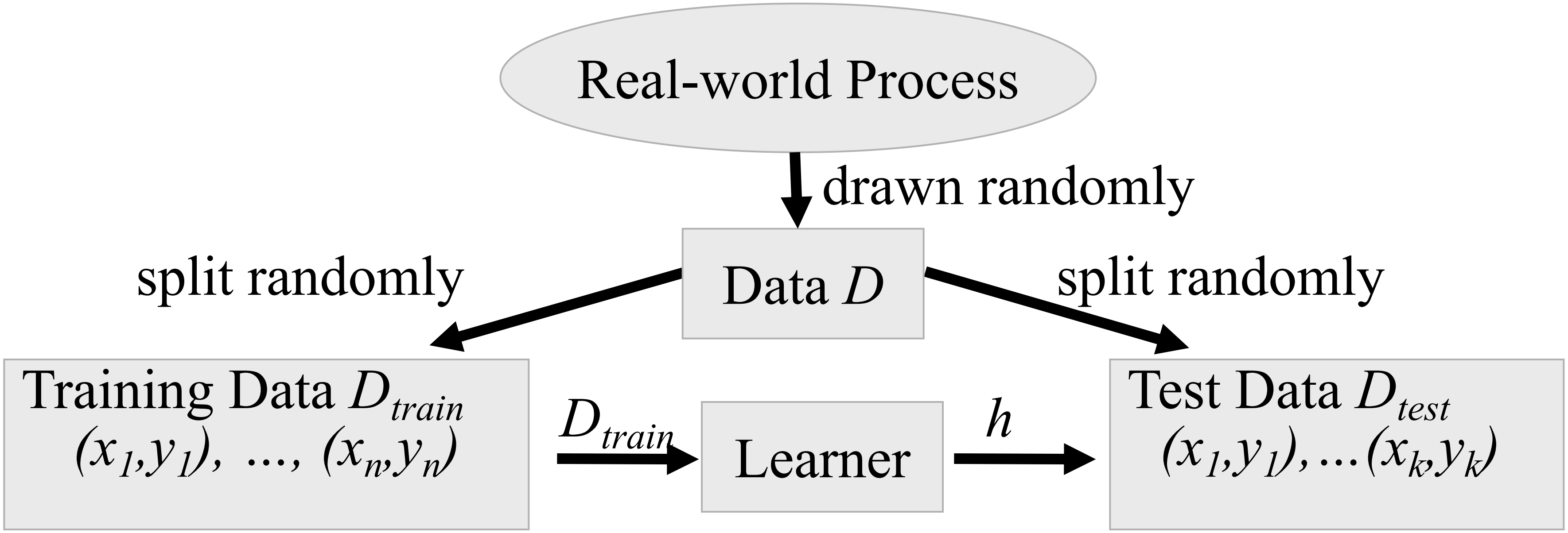
Voila: Peeking occurred! Why?

The hypothesis was selected on the basis of its performance on the **test set**, so information about the test set has leaked into the learning algorithm.

So a new (separate!) test set would be required!

Note: In competitions, such as the "Netflix $1M challenge," test set is not revealed to the competitors. (Data is held back.)

Test/Training Split



Measuring Prediction Performance

Definition: *The* **training error** $Err_{D_{train}}(h)$ *on training data* $D_{train} = ((\vec{x}_1, y_1), ..., (\vec{x}_n, y_n))$ *of a hypothesis* h *is* $Err_{D_{train}}(h) = \frac{1}{n}\sum_{i=1}^{n} \Delta(h(\vec{x}_i), y_i)$.

Definition: *The* **test error** $Err_{D_{test}}(h)$ *on test data* $D_{test} = ((\vec{x}_1, y_1), ..., (\vec{x}_k, y_k))$ *of a hypothesis* h *is* $Err_{D_{test}}(h) = \frac{1}{k}\sum_{i=1}^{k} \Delta(h(\vec{x}_i), y_i)$.

Definition: *The* **prediction/generalization/true error** $Err_P(h)$ *of a hypothesis* h *for a learning task* $P(X, Y)$ *is*

$$Err_P(h) = \sum_{\vec{x} \in X, y \in Y} \Delta(h(\vec{x}), y) P(X = \vec{x}, Y = y).$$

Definition: *The* zero/one-loss function $\Delta(a, b)$ *returns 1 if* $a \neq b$ *and 0 otherwise.*

Performance Measures

Error Rate

- Fraction (or percentage) of false predictions

Accuracy

- Fraction (or percentage) of correct predictions

Precision/Recall

Example: binary classification problems (classes pos/neg)

- Precision: Fraction (or percentage) of correct predictions among all examples predicted to be positive
- Recall: Fraction (or percentage) of correct predictions among all real positive examples

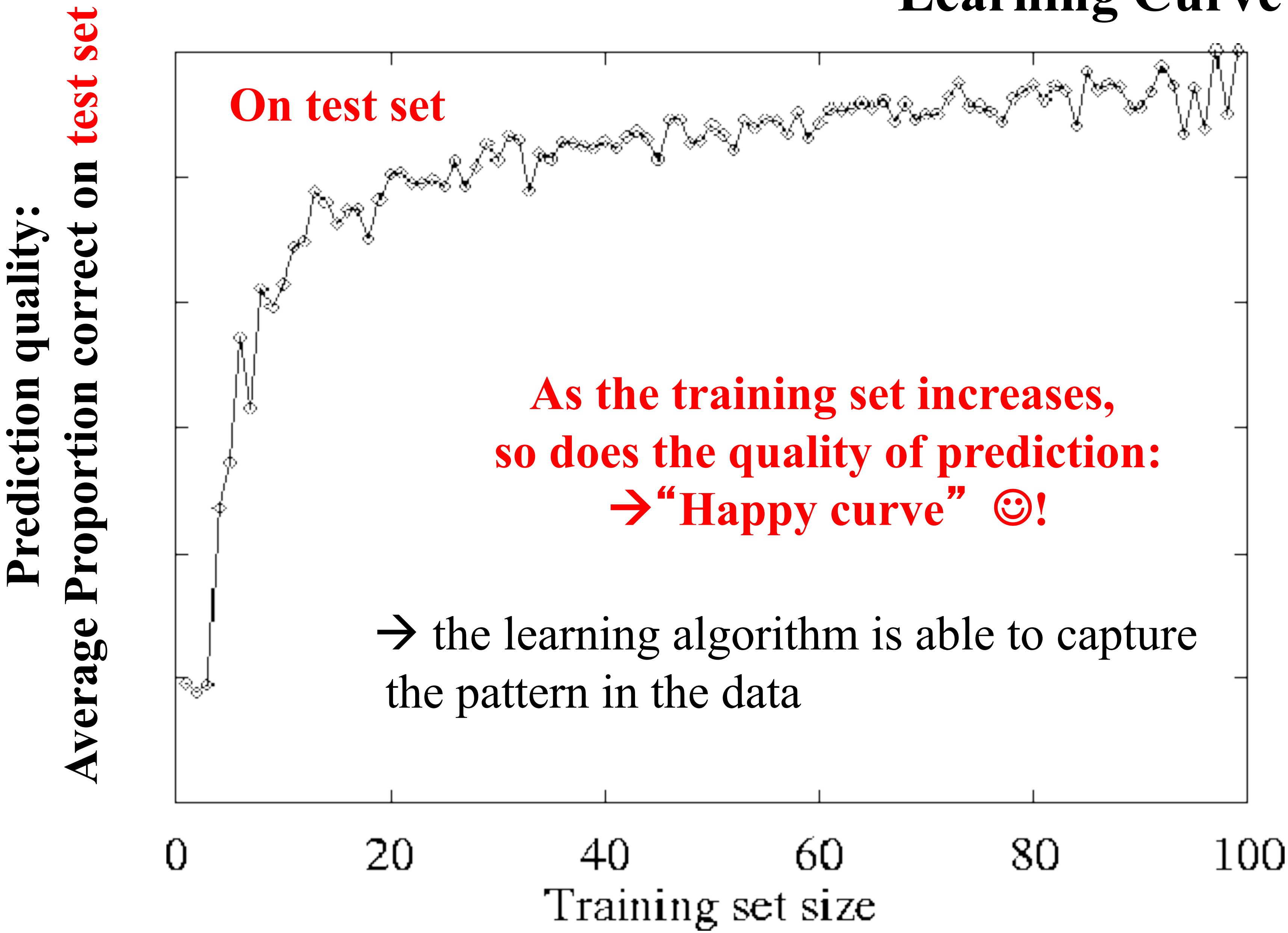
(Can be generalized to multi-class case.)

Learning Curve Graph

Learning curve graph

average prediction quality – proportion correct   on test set – as a function of the size of the training set..

Restaurant Example: Learning Curve



On test set

As the training set increases, so does the quality of prediction: → "Happy curve" ☺!

→ the learning algorithm is able to capture the pattern in the data

Precision vs. Recall

Precision

- # of true positives / (# true positives + # false positives)

Recall

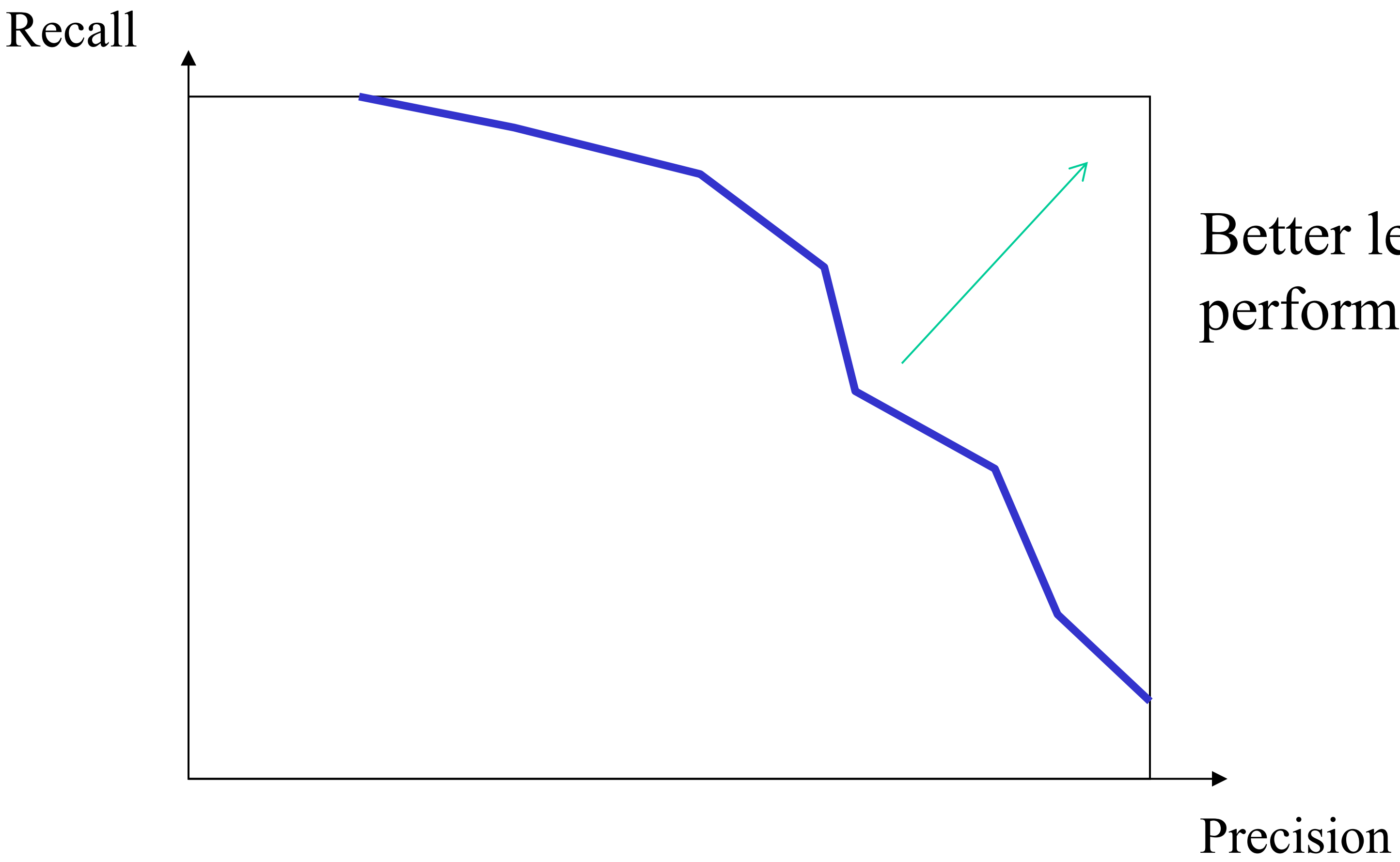
- # of true positives / (# true positives + # false negatives)

A precise classifier is selective

A classifier with high recall is inclusive

Precision-Recall curves

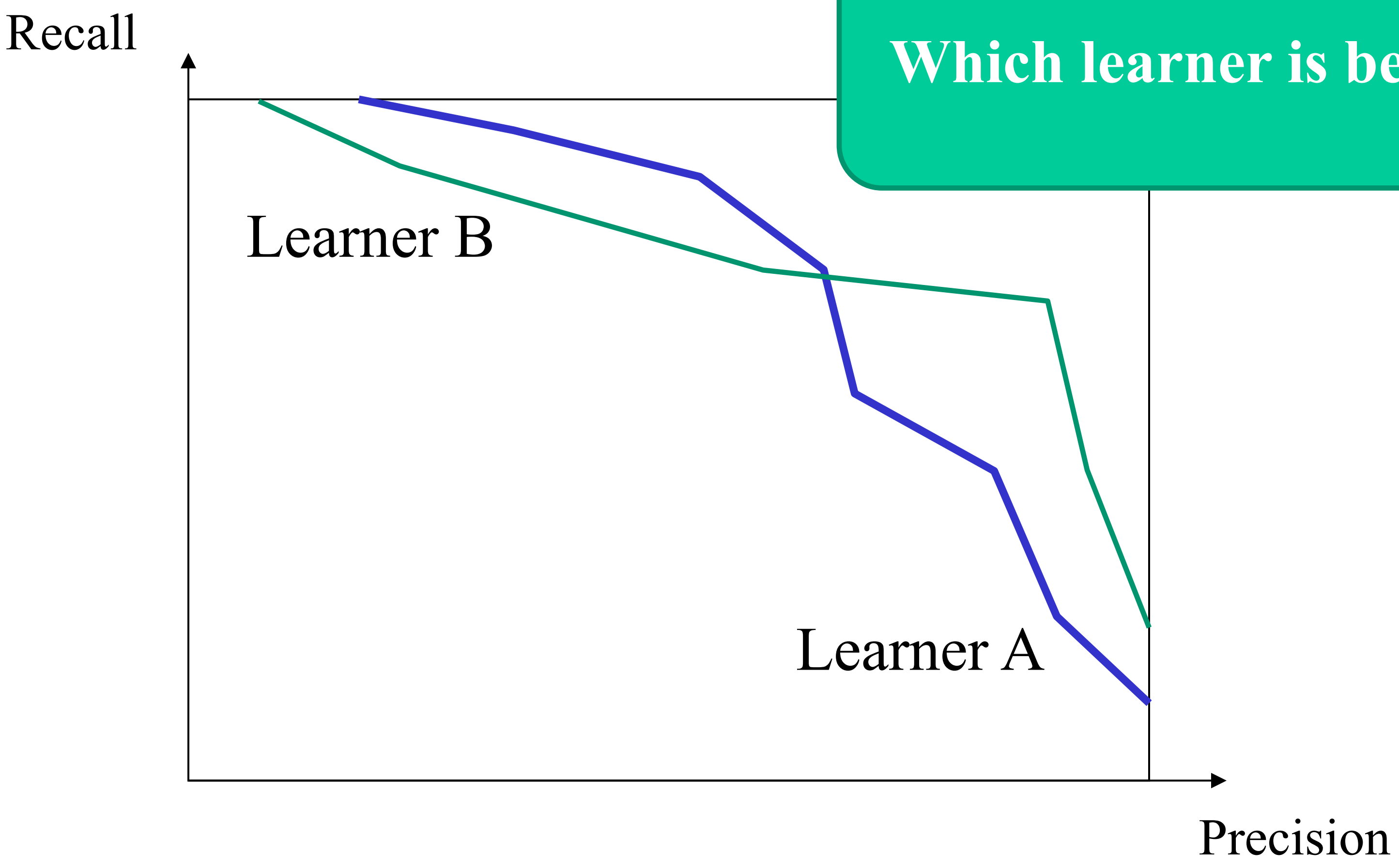
Measure Precision vs Recall as the classification boundary is tuned



Better learning performance

Precision-Recall curves

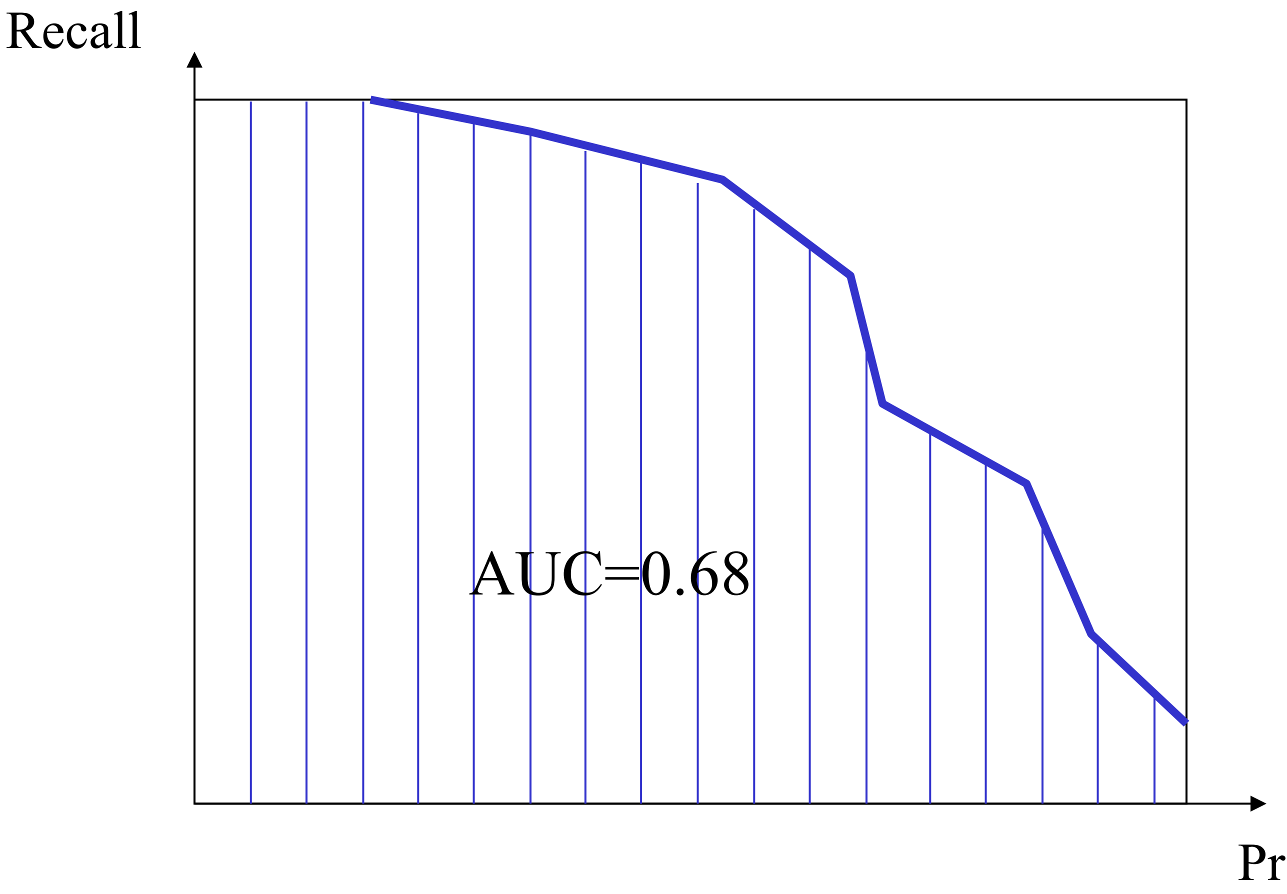
Measure Precision vs Recall as the classification boundary is tuned



Which learner is better?

Area Under Curve

AUC-PR: measure the area under the precision-recall curve



AUC metrics

A single number that measures “overall” performance across multiple thresholds

- Useful for comparing many learners
- “Smears out” PR curve

Note training / testing set dependence

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, and 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.

Summary

Decision tree learning is a particular case of supervised learning,

For supervised learning, the aim is to find a simple hypothesis approximately consistent with training examples

Decision tree learning using information gain

Learning performance = prediction accuracy measured on test set

Extensions of the Decision Tree Learning Algorithm (Briefly)

Noisy data

Overfitting and Model Selection

Cross Validation

Missing Data (R&N, Section 18.3.6)

Using gain ratios (R&N, Section 18.3.6)

Real-valued data (R&N, Section 18.3.6)

Generation of rules and pruning

Noisy data

Many kinds of "noise" that could occur in the examples:

- Two examples have same attribute/value pairs, but different classifications
  - →report majority classification for the examples corresponding to the node deterministic hypothesis.
  - →report estimated probabilities of each classification using the relative frequency (if considering stochastic hypotheses)
- 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

One important reason why you don't want to "overfit" your learned model.

Overfitting

Ex.: Problem of trying to predict the roll of a die. The experiment data include:

Day of the week; (2) Month of the week; (3) Color of the die;

….

DTL may find an hypothesis that fits the data but with irrelevant attributes.

Some attributes are irrelevant to the decision-making process, e.g., color of a die is irrelevant to its outcome but they are used to differentiate examples → Overfitting.

Overfitting means fitting the **training set** "too well" → **performance on the test set degrades.**

Example overfitting risk: Using restaurant name.

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 to lower # nodes in the decision tree or put a limit on number of nodes created.**

- **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 no**des.

Overfitting is a key problem in learning. There are formal results on the number of examples needed to properly train an hypothesis of a certain complexity ("number of parameters" or # nodes in DT). The more params, the more data is needed. We'll see some of this in our discussion of PAC learning.

Overfitting

Let's consider D, the entire distribution of data, and T, the training set.

Hypothesis h $\in$ H overfits D if

$\exists$ h' $\neq$ h $\in$ H such that

$\text{error}_T(h) < \text{error}_T(h')$ but

$\text{error}_D(h) > \text{error}_D(h')$

Note: estimate error on full distribution by using test data set.

Data overfitting is the arguably the most common pitfall in machine learning.

Why?

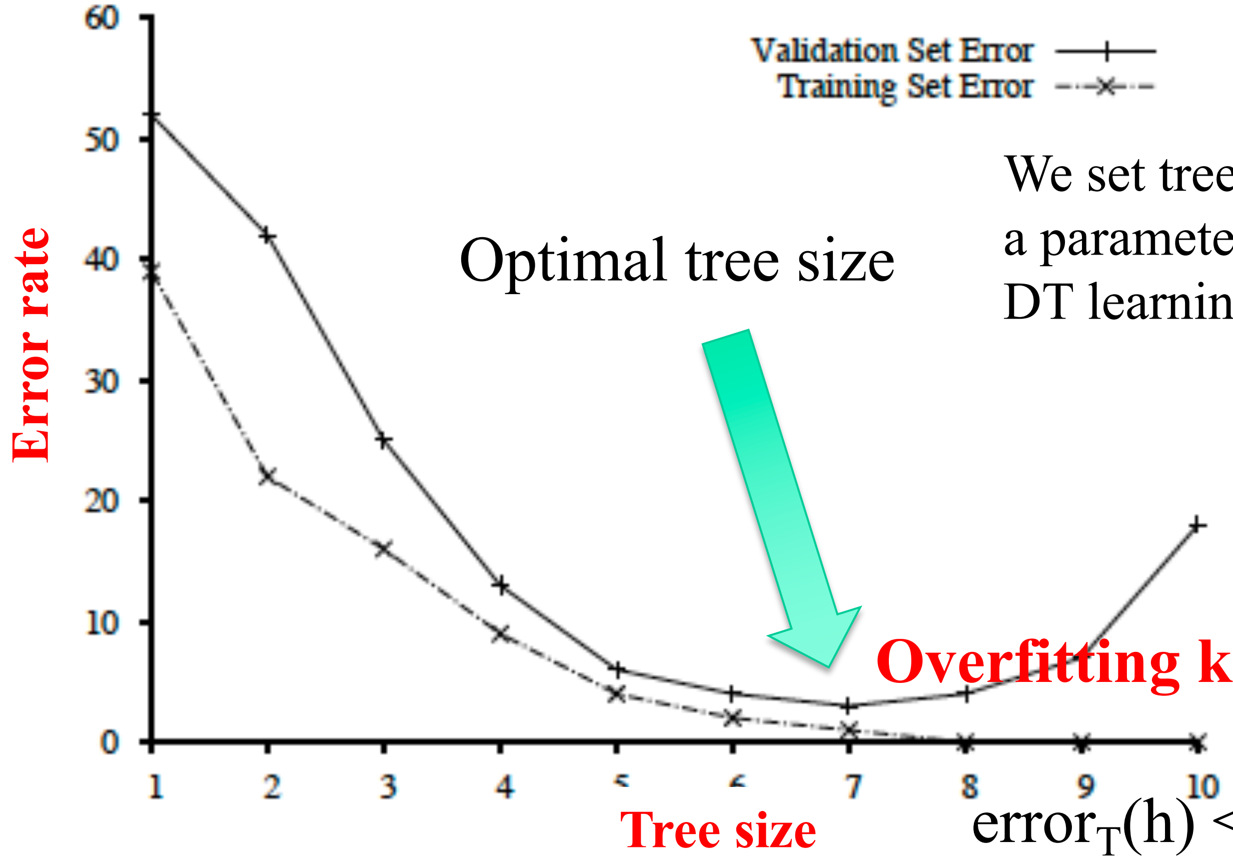
1) **Temptation to use as much data as possible to train on. ("Ignore test till end." Test set too small.) Data "peeking" not noticed.**

1) **Temptation to fit very complex hypothesis (e.g. large decision tree). In general, the larger the tree, the better the fit to the training data.**

It's hard to think of a better fit to the training data as a "worse" result. Often difficult to fit training data well, so it seems that "a good fit to the training data means a good result."

Note: Modern "savior:" Massive amounts of data to train on! Somewhat characteristic of ML AI community vs. traditional statistics community. Anecdote: Netflix competition.

Key figure in machine learning



Optimal tree size

We set tree size as
a parameter in our
DT learning alg.

Overfitting kicks in…

$\text{error}_T(h) < \text{error}_T(h')$ but
$\text{error}_D(h) > \text{error}_D(h')$

Note: with larger and larger trees,
we just do better and better on the training set!

But note the performance on the validation set…

Procedure for finding the optimal tree size is called "model selection." See section 18.4.1 R&N and Fig. 18.8.

To determine validation error for each tree size, use k-fold cross-validation. (Uses the data better than "holdout cross-validation.") Uses "all data - test set" --- k times splits that set into a training set and a validation set.

After right decision tree size is found from the error rate curve on validation data, train on all training data to get final decision tree (of the right size).

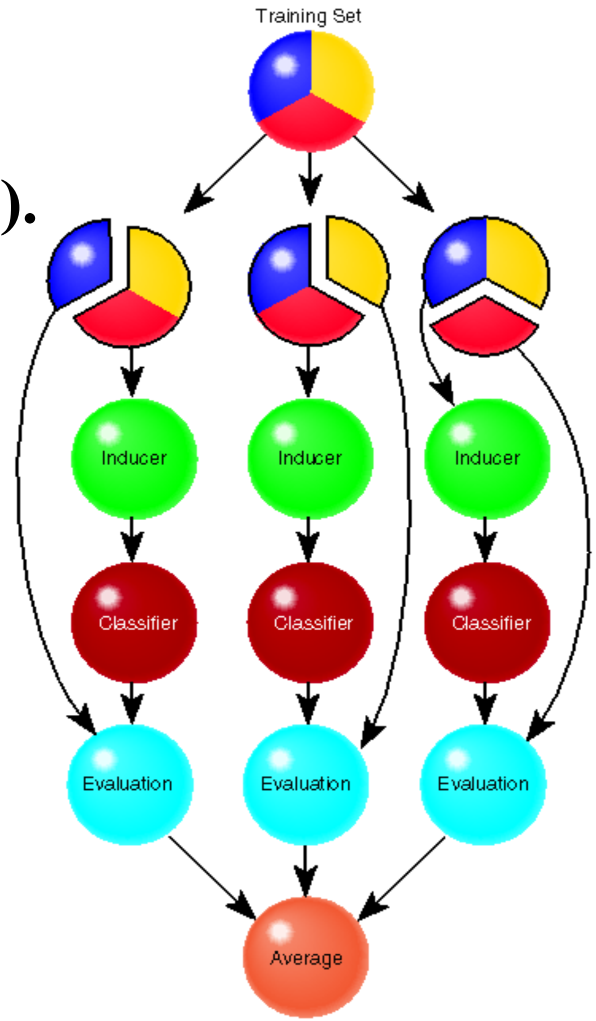
Finally, evaluate tree on the test data (not used before) to get true generalization error (to unseen examples).

Cross Validation

Learner L is e.g. DT learner for "tree with 7 nodes" max.

A method for estimating the accuracy (or error) of a learner (using validation set).

```
CV( data S, alg L, int k )
    Divide S into k disjoint sets { $S_1, S_2, \ldots, S_k$ }
    For i = 1..k do
        Run L on $S_{-i} = S - S_i$
        obtain $L(S_{-i}) = h_i$
        Evaluate $h_i$ on $S_i$
            $err_{S_i}(h_i) = 1/|S_i| \sum_{\langle x,y \rangle \in S_i} I(h_i(x) \neq y)$
    Return Average $1/k \sum_i err_{S_i}(h_i)$
```



Specific techniques for dealing with overfitting
(Model selection provides general framework)

1) **Decision tree pruning or grow only up to certain size.**
Prevent splitting on features that are not clearly relevant.

Testing of relevance of features --- “does split provide new information”:
statistical tests ---> Section 18.3.5 R&N χ^2 test.

2) **Grow full tree, then post-prune rule post-pruning**

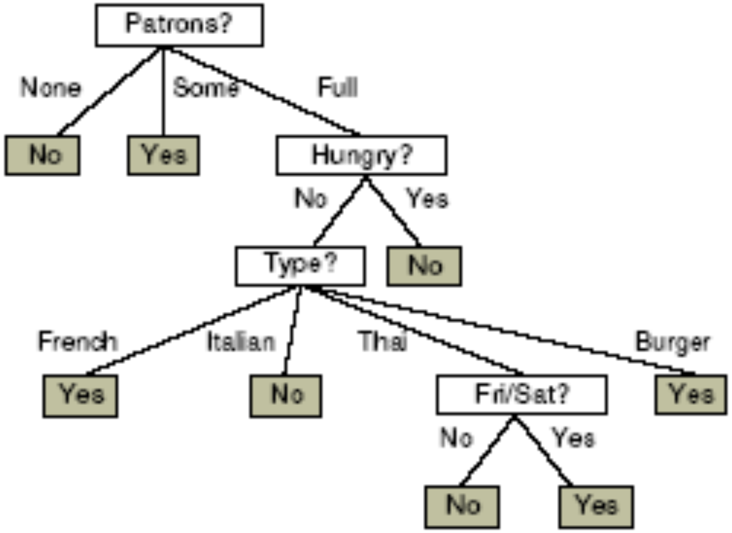
3) **MDL (minimal description length):**

minimize
size(tree) + size(misclassifications(tree))

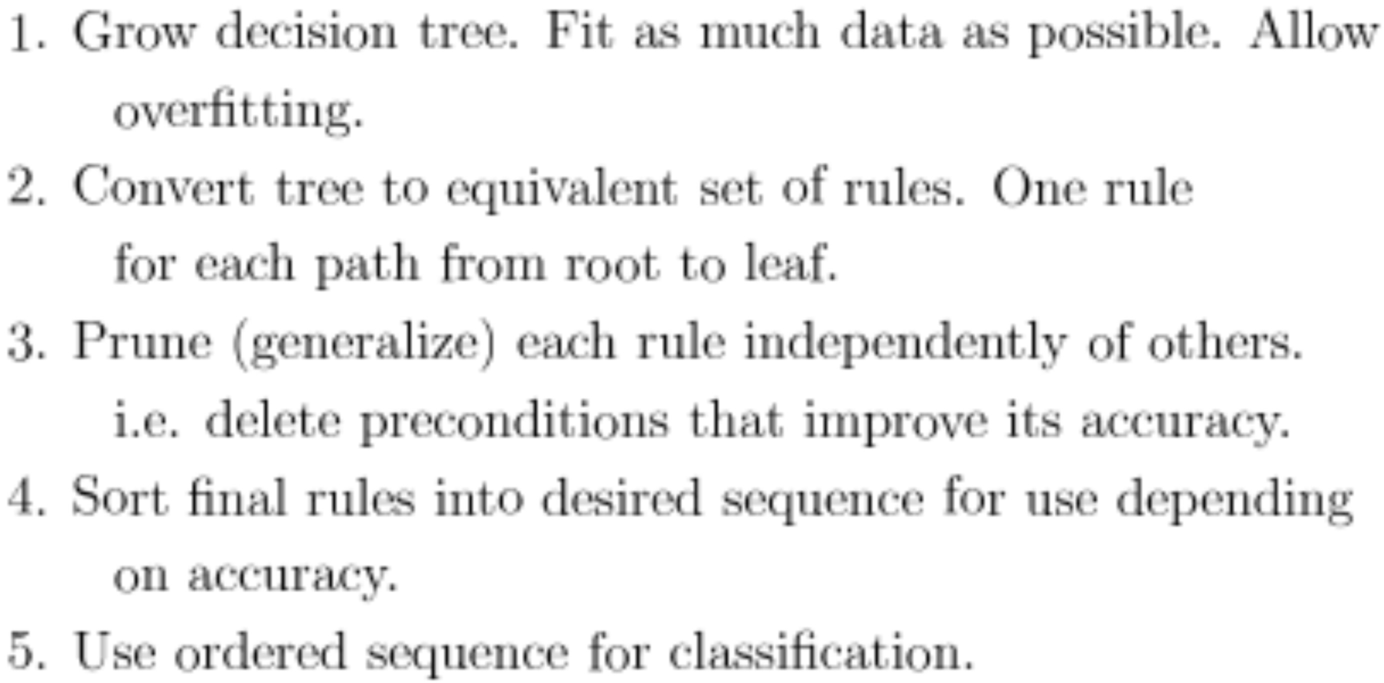
Converting Trees to Rules

Every decision tree corresponds to set of rules:

- IF (Patrons = None)
  THEN WillWait = No
- IF (Patrons = Full)
  & (Hungry = No)
  &(Type = French)
  THEN WillWait = Yes
- ...



Fighting Overfitting: Using Rule Post-Pruning

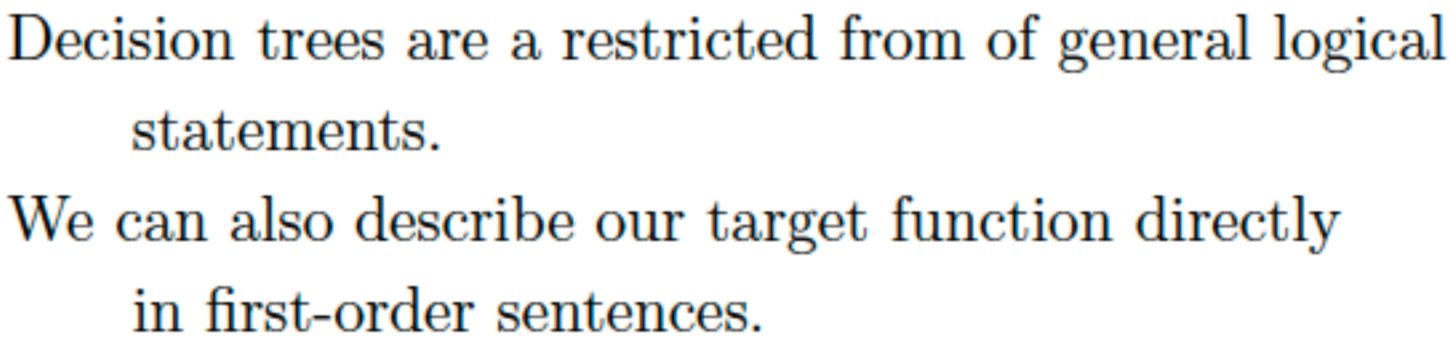


1. Grow decision tree. Fit as much data as possible. Allow overfitting.
2. Convert tree to equivalent set of rules. One rule for each path from root to leaf.
3. Prune (generalize) each rule independently of others. i.e. delete preconditions that improve its accuracy.
4. Sort final rules into desired sequence for use depending on accuracy.
5. Use ordered sequence for classification.

This is the strategy of the most successful commercial decision tree learning method (C4.5 — Quinlan 1993). Widely used in data mining.

What is advantage of rule representation over the decision tree?

Logical aside



Decision trees are a restricted from of general logical statements.

We can also describe our target function directly in first-order sentences.

Example:

$$\forall WillWait(r) \Leftrightarrow Patrons(r, Some)$$

$$\vee(Patrons(r, Full) \wedge \neg Hungry(r) \wedge Type(r, French))$$

$$\vee(Patrons(r, Full) \wedge \neg Hungry(r) \wedge Type(r, Thai) \wedge Fri/Sat$$

$$\vee(Patrons(r, Full) \wedge \neg Hungry(r) \wedge Type(r, Burger))$$

This is our **hypothesis** H_r. In general, we search from among a space of hypotheses:

$$H_1 \vee H_2 \vee H_3 \vee \ldots \vee H_n.$$

End logical aside

Here's an **example** in logical form:

$$Alternate(X1) \land \neg Bar(X_1) \land \neg Fri/Sat(X_1) \land$$
$$Hungry(X_1) \land \ldots \land WillWait(X_1)$$

We can test if this example is **consistent** with our hyposthesis. If not, we may have to **generalize** or **specialize** our hypothesis:

Current-best-hypothesis search.

Summary: When to use Decision Trees

Instances presented as attribute-value pairs

Method of approximating discrete-valued functions

Target function has discrete values: classification problems

Robust to noisy data:

Training data may contain

- errors
- missing attribute values

Typical bias: prefer smaller trees (Ockham's razor)

Widely used, practical and easy to interpret results

Inducing decision trees is one of the most widely used learning methods in practice

Can outperform human experts in many problems

Strengths include

- Fast
- simple to implement
- human readable

Can be a legal requirement! Why?

- 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)

DECISION TREES IN SCIKIT

```
from sklearn.datasets import load_iris
from sklearn import tree

# Load a common dataset, fit a decision tree to it
iris = load_iris()
clf = tree.DecisionTreeClassifier()
clf = clf.fit(iris.data, iris.target)
```

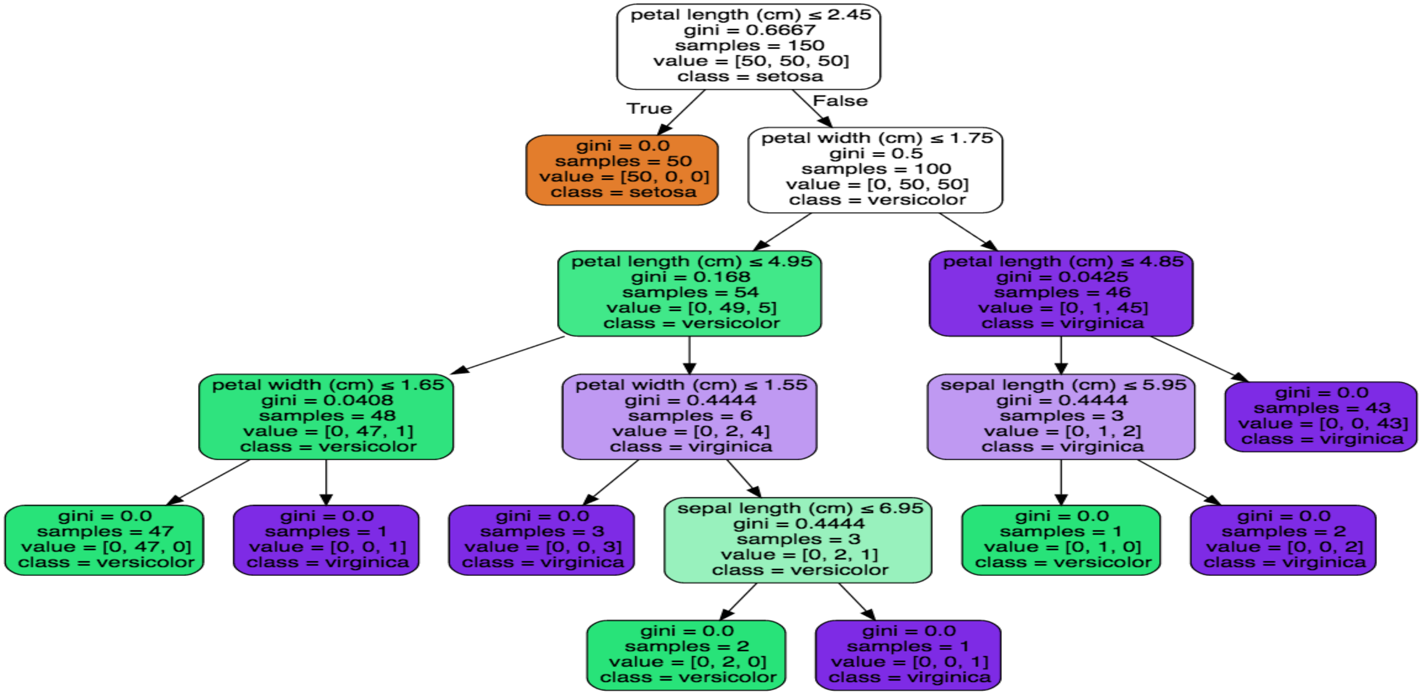
Trains a decision tree using default parameters (attribute chosen to split on either Gini or entropy, no max depth, etc)

```
# Predict most likely class
clf.predict([[2., 2.]])
```

```
# Predict PDF over classes (%training samples in leaf)
clf.predict_proba([[2., 2.]])
```

VISUALIZING A DECISION TREE

```
from IPython.display import Image
dot_data = tree.export_graphviz(clf,
              out_file=None,
              feature_names=iris.feature_names,
              class_names=iris.target_names,
              filled=True, rounded=True)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```



RANDOM FORESTS

Decision trees are very interpretable, but may be brittle to changes in the training data, as well as noise

Random forests are an ensemble method that:

- Resamples the training data;
- Builds many decision trees; and
- Averages predictions of trees to classify.

This is done through bagging and random feature selection

BAGGING

Bagging: Bootstrap aggregation

Resampling a training set of size n via the bootstrap:

- Sample with replacement n elements

General scheme for random forests:

1. Create B bootstrap samples, $\{Z_1, Z_2, \ldots, Z_B\}$
2. Build B decision trees, $\{T_1, T_2, \ldots, T_B\}$, from $\{Z_1, Z_2, \ldots, Z_B\}$

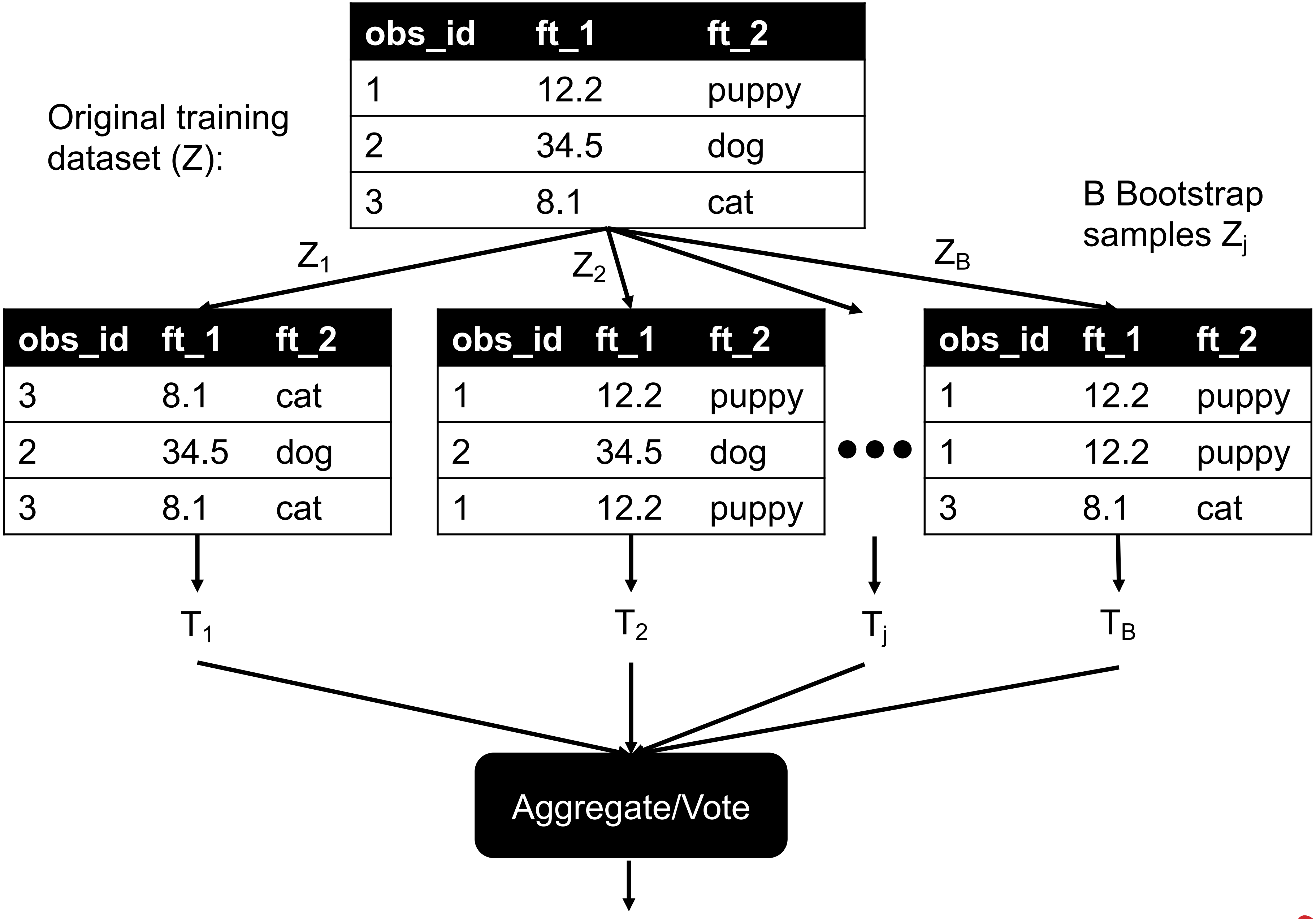
Classification/Regression:

1. Each tree T_j predicts class/value y_j
2. Return average $1/B\ \Sigma_{j=\{1,\ldots,B\}}\ y_j$ for regression,
   or majority vote for classification

Original training dataset (Z):

obs_id	ft_1	ft_2
1	12.2	puppy
2	34.5	dog
3	8.1	cat

B Bootstrap samples Z_j



obs_id	ft_1	ft_2
3	8.1	cat
2	34.5	dog
3	8.1	cat

obs_id	ft_1	ft_2
1	12.2	puppy
2	34.5	dog
1	12.2	puppy

obs_id	ft_1	ft_2
1	12.2	puppy
1	12.2	puppy
3	8.1	cat

Class estimate or predicted value

RANDOM ATTRIBUTE SELECTION

We get some randomness via bootstrapping

- We like this! Randomness increases the bias of the forest slightly at a huge decrease in variance (due to averaging)

We can further reduce correlation between trees by:

1. **For each tree, at every split point …**
2. **… choose a random subset of attributes …**
3. **… then split on the "best" (entropy, Gini) within only that subset**

RANDOM FORESTS IN SCIKIT-LEARN

```
from sklearn.ensemble import RandomForestClassifier

# Train a random forest of 10 default decision trees
X = [[0, 0], [1, 1]]
Y = [0, 1]
clf = RandomForestClassifier(n_estimators=10)
clf = clf.fit(X, Y)
```

Can we get even more random?!

Extremely randomized trees (`ExtraTreesClassifier`) **do bagging, random attribute selection, but also:**

1. **At each split point, choose random splits**
2. **Pick the best of those random splits**

Similar bias/variance performance to RFs, but can be faster computationally