Inductive Learning and Decision Trees

Doug Downey
EECS 349 Winter 2014

with slides from Pedro Domingos, Bryan Pardo

Outline

- Announcements
  - Homework #1 and #2 assigned
  - Have you completed them?
- Inductive learning
- Decision Trees

Outline

- Announcements
  - Homework #1 and #2 assigned
  - Have you completed them?
- **Inductive learning**
- Decision Trees

Instances

- E.g. Four Days, in terms of weather:

Sky	**Temp**	**Humid**	**Wind**	**Water**	**Forecast**
sunny	warm	normal	strong	warm	same
sunny	warm	high	strong	warm	same
rainy	cold	high	strong	warm	change
sunny	warm	high	strong	cool	change

Functions

- "Days on which my friend Aldo enjoys his favorite water sport"

INPUT

OUTPUT

Sky	**Temp**	**Humid**	**Wind**	**Water**	**Forecast**	**C(x)**
sunny	warm	normal	strong	warm	same	1
sunny	warm	high	strong	warm	same	1
rainy	cold	high	strong	warm	change	0
sunny	warm	high	strong	cool	change	1

Inductive Learning!

- **Predict** the output for a new instance

INPUT

OUTPUT

Sky	**Temp**	**Humid**	**Wind**	**Water**	**Forecast**	**C(x)**
sunny	warm	normal	strong	warm	same	1
sunny	warm	high	strong	warm	same	1
rainy	cold	high	strong	warm	change	0
sunny	warm	high	strong	cool	change	1
rainy	**warm**	**high**	**strong**	**cool**	**change**	**?**

General Inductive Learning Task

DEFINE:

- Set *X* of Instances (of *n*-tuples $\mathbf{x} = <x_1, ..., x_n>$)
  - E.g., days decribed by *attributes* (or *features*):
    Sky, Temp, Humidity, Wind, Water, Forecast
- Target function *y*, e.g.:
  - EnjoySport $X \rightarrow Y = \{0,1\}$
  - HoursOfSport $X \rightarrow Y = \{0, 1, 2, 3, 4\}$
  - InchesOfRain $X \rightarrow Y = [0, 10]$

GIVEN:

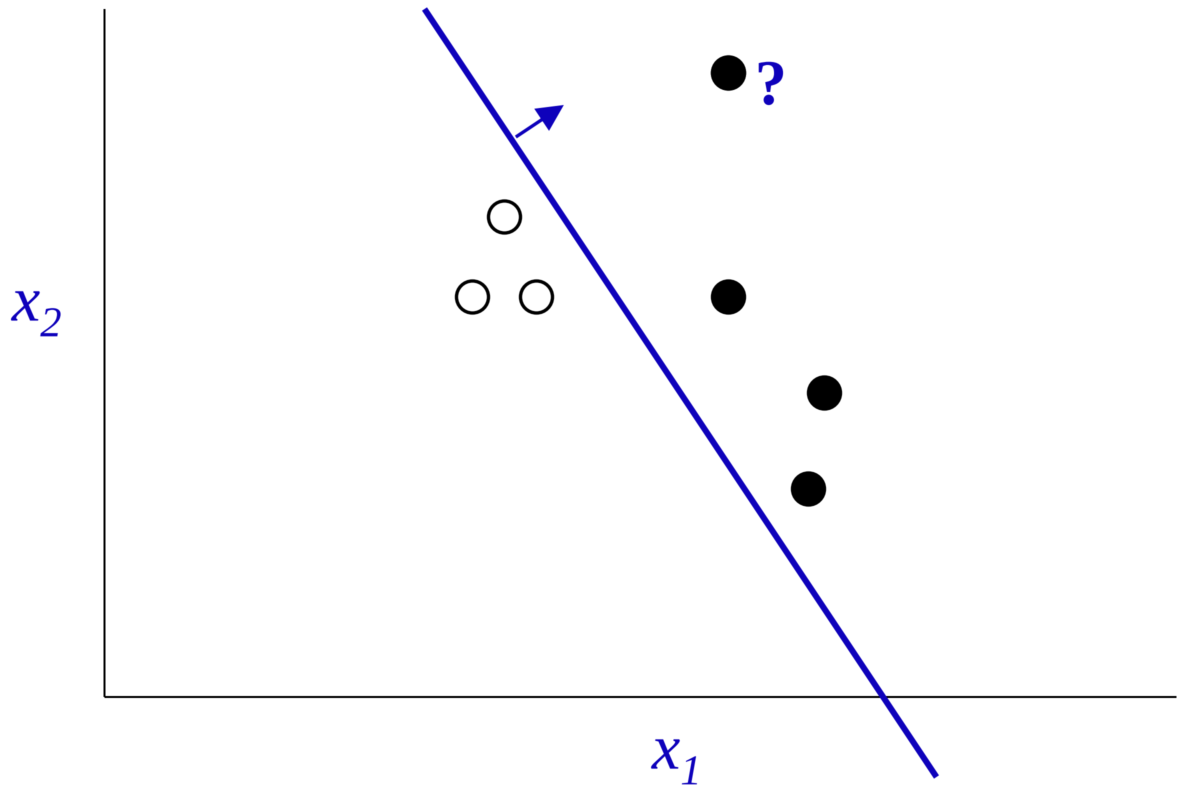
- *Training examples D*
  - examples of the target function: $<\mathbf{x}, y(\mathbf{x})>$

FIND:

- A hypothesis *h* such that $h(\mathbf{x})$ approximates $y(\mathbf{x})$.

Another example: continuous attributes

Learn function from $\mathbf{x} = (x_1, \ldots, x_d)$ to $y(\mathbf{x}) \in \{0, 1\}$
given labeled examples $(\mathbf{x}, y(\mathbf{x}))$



Hypothesis Spaces

- Hypothesis space *H* is a **subset** of all *y:* X $\rightarrow$ Y e.g.:
  - Linear separators
  - Conjunctions of constraints on attributes (humidity must be low, and outlook != rain)
  - Etc.
- In machine learning, we restrict ourselves to *H*
  - The *subset* thing turns out to be important

Examples

- Credit Risk Analysis
  - ***X*:** Properties of customer and proposed purchase
  - ***y*(*x*):** Approve (1) or Disapprove (0)
- Disease Diagnosis
  - ***X*:** Properties of patient (symptoms, lab tests)
  - ***y*(*x*):** Disease (if any)
- Face Recognition
  - ***X*:** Bitmap image
  - ***y*(*x*):**Name of person
- Automatic Steering
  - ***X*:** Bitmap picture of road surface in front of car
  - ***y*(*x*):** Degrees to turn the steering wheel

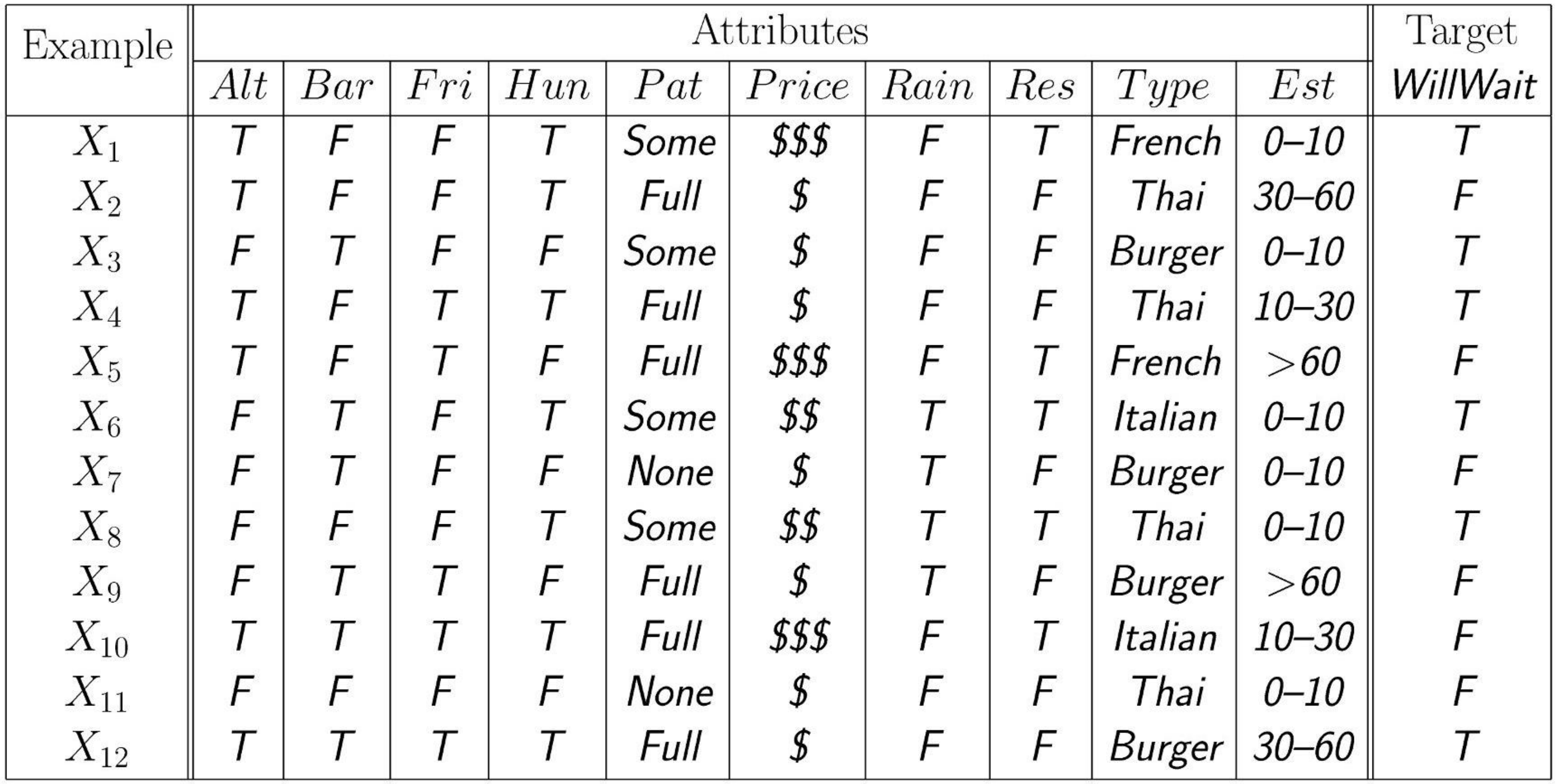
Appropriate applications

- Situations in which:
  - there is no human expert
  - Humans can perform the task but can’t describe how
  - The desired function changes frequently
  - Each user needs a customized *f*

Outline

- Announcements
  - Homework #1 and #2 assigned
- Inductive learning
- **Decision Trees**

Task: Will I wait for a table?

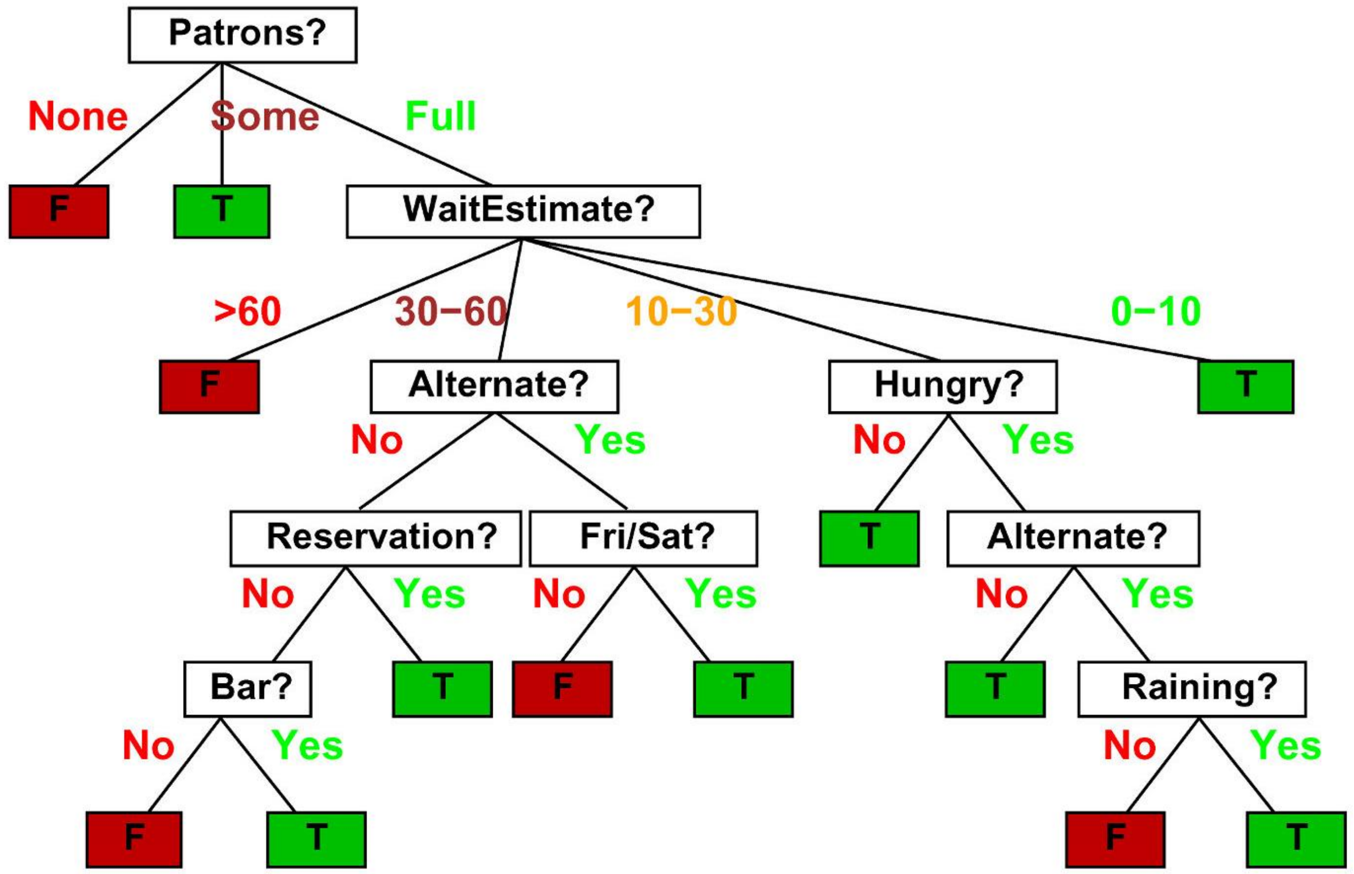


Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*T*	*F*	*F*	*T*	*Some*	*\$\$\$*	*F*	*T*	*French*	*0–10*	*T*
X_2	*T*	*F*	*F*	*T*	*Full*	*\$*	*F*	*F*	*Thai*	*30–60*	*F*
X_3	*F*	*T*	*F*	*F*	*Some*	*\$*	*F*	*F*	*Burger*	*0–10*	*T*
X_4	*T*	*F*	*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*

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

Decision Trees!

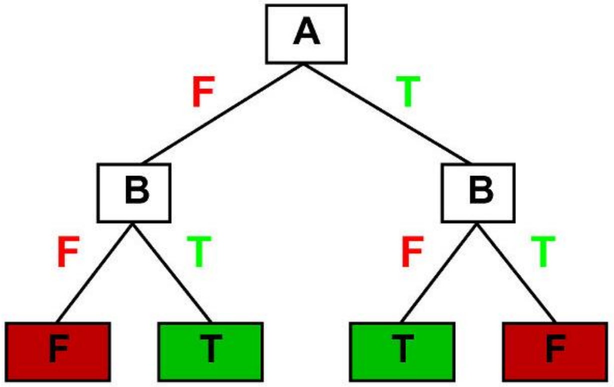
One possible representation for hypotheses
E.g., here is the "true" tree for deciding whether to wait:



Expressiveness of D-Trees

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

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

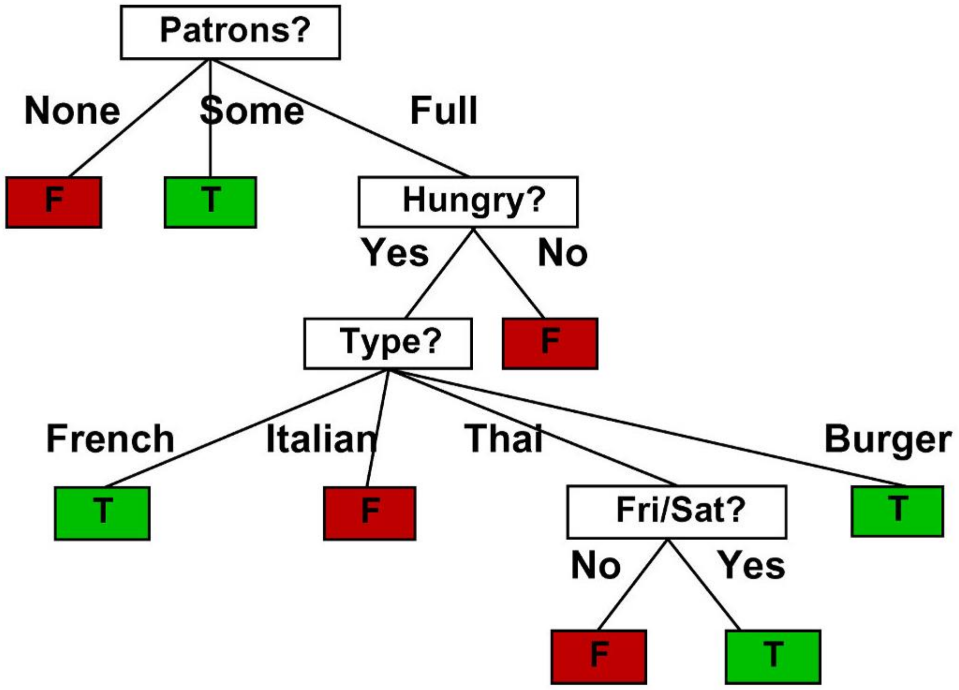


Trivially, there is a consistent decision tree for any training set w/ one path to leaf for each example (unless f nondeterministic in x) but it probably won't generalize to new examples

Prefer to find more **compact** decision trees

A learned decision tree

Decision tree learned from the 12 examples:



Substantially simpler than "true" tree—a more complex hypothesis isn't justified by small amount of data

Inductive Bias

- To learn, we **must** prefer some functions to others
  - **Selection bias**
    - use a **restricted** hypothesis space, e.g.:
      - linear separators
      - 2-level decision trees
  - **Preference bias**
    - use the whole concept space, but state a **preference** over concepts, e.g.:
      - *Lowest-degree* polynomial that separates the data
      - *shortest* decision tree that fits the data

Decision Tree Learning (ID3)

Aim: find a small tree consistent with the training examples

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

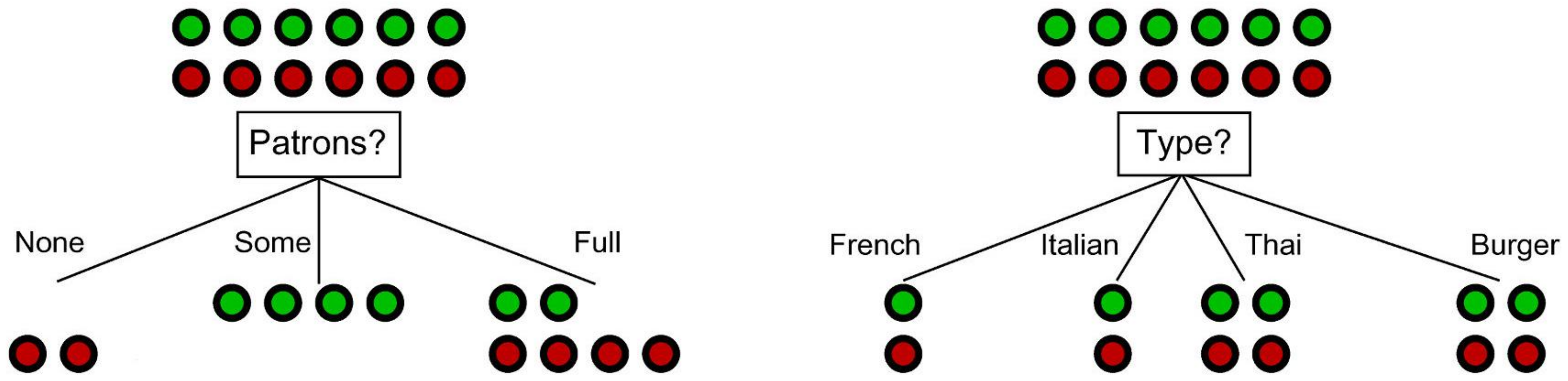
```
function DTL(examples, attributes, default) returns a decision tree
    if examples is empty then return default
    else if all examples have the same classification then return the classification
    else if attributes is empty then return MODE(examples)
    else
        best ← CHOOSE-ATTRIBUTE(attributes, examples)
        tree ← a new decision tree with root test best
        for each value $v_i$ of best do
            $examples_i$ ← {elements of examples with best = $v_i$}
            subtree ← DTL($examples_i$, attributes − best, MODE(examples))
            add a branch to tree with label $v_i$ and subtree subtree
        return tree
```

Recap

- Inductive learning
  - Goal: generate a **hypothesis** – a function from **instances** described by **attributes** to an output – using **training examples**.
  - Requires **inductive bias**
    - a restricted **hypothesis space**, or preferences over hypotheses.
- Decision Trees
  - Simple representation of hypotheses, recursive learning algorithm
  - Prefer smaller trees!

Choosing an attribute

Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"



Patrons? is a better choice—gives **information** about the classification

Information

Information answers questions

The more clueless I am about the answer initially, the more information is contained in the answer

Scale: 1 bit = answer to Boolean question with prior $\langle 0.5, 0.5\rangle$

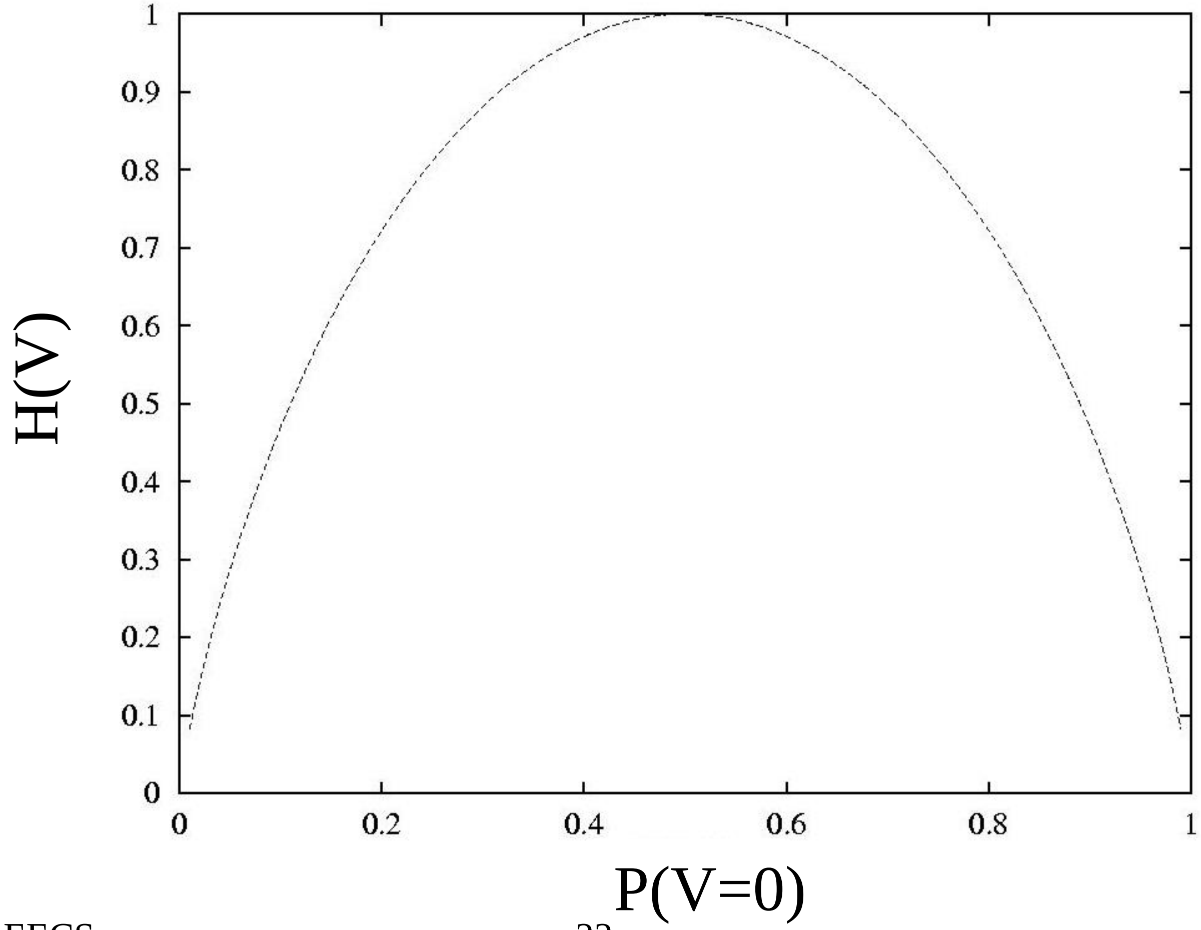
Information in an answer when prior is $\langle P_1, \ldots, P_n\rangle$ is

$$H(\langle P_1, \ldots, P_n\rangle) = \Sigma_{i=1}^{n} - P_i \log_2 P_i$$

(also called entropy of the prior)

Entropy

The entropy H(V) of a Boolean random variable V as the probability of V = 0 varies from 0 to 1



Using Information

Suppose we have p positive and n negative examples at the root

$\Rightarrow$ $H(\langle p/(p+n), n/(p+n)\rangle)$ bits needed to classify a new example

E.g., for 12 restaurant examples, $p=n=6$ so we need 1 bit

An attribute splits the examples E into subsets E_i, each of which (we hope) needs less information to complete the classification

Let E_i have p_i positive and n_i negative examples

$\Rightarrow$ $H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i)\rangle)$ bits needed to classify a new example

$\Rightarrow$ **expected** number of bits per example over all branches is

$$\Sigma_i \ \frac{p_i+n_i}{p+n} H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i)\rangle)$$

For $Patrons?$, this is 0.459 bits, for $Type$ this is (still) 1 bit

$\Rightarrow$ choose the attribute that minimizes the remaining information needed

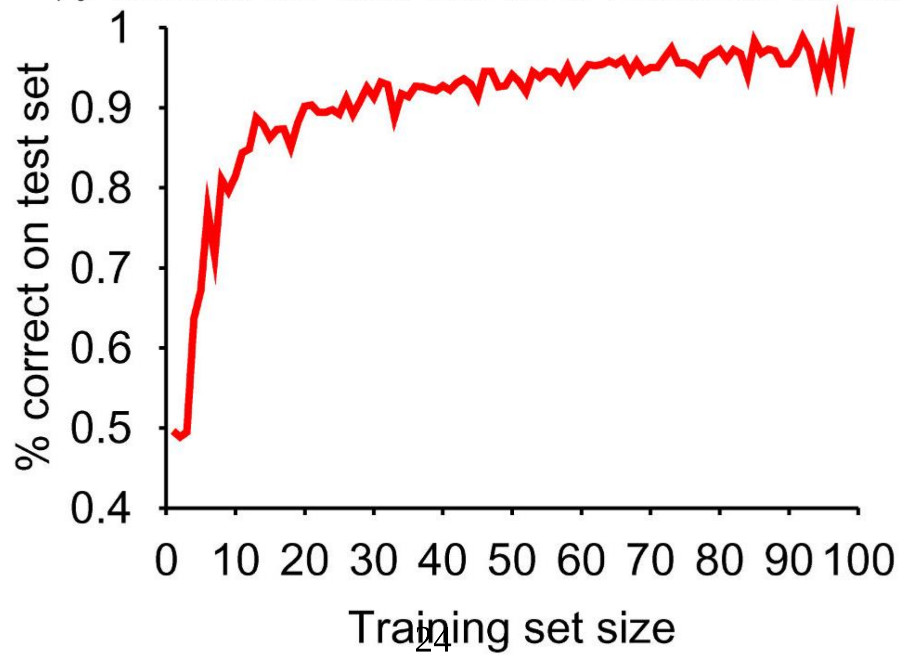
Measuring Performance

How do we know that $h \approx f$? (Hume's **Problem of Induction**)

1) Use theorems of computational/statistical learning theory

2) Try h on a new test set of examples
   (use **same distribution over example space** as training set)

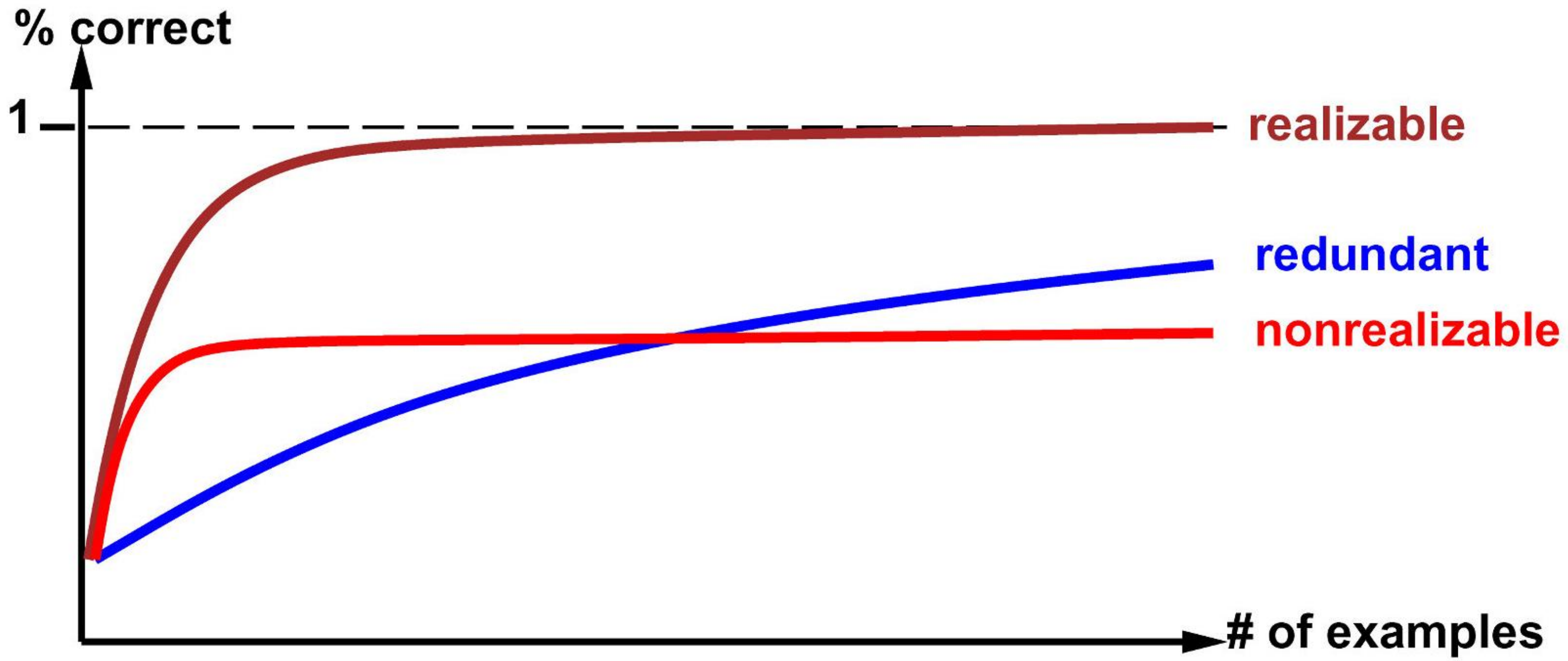
Learning curve $=$ % correct on test set as a function of training set size



What the learning curve tells us

Learning curve depends on

– realizable (can express target function) vs. non-realizable
  non-realizability can be due to missing attributes
  or restricted hypothesis class (e.g., thresholded linear function)
– redundant expressiveness (e.g., loads of irrelevant attributes)

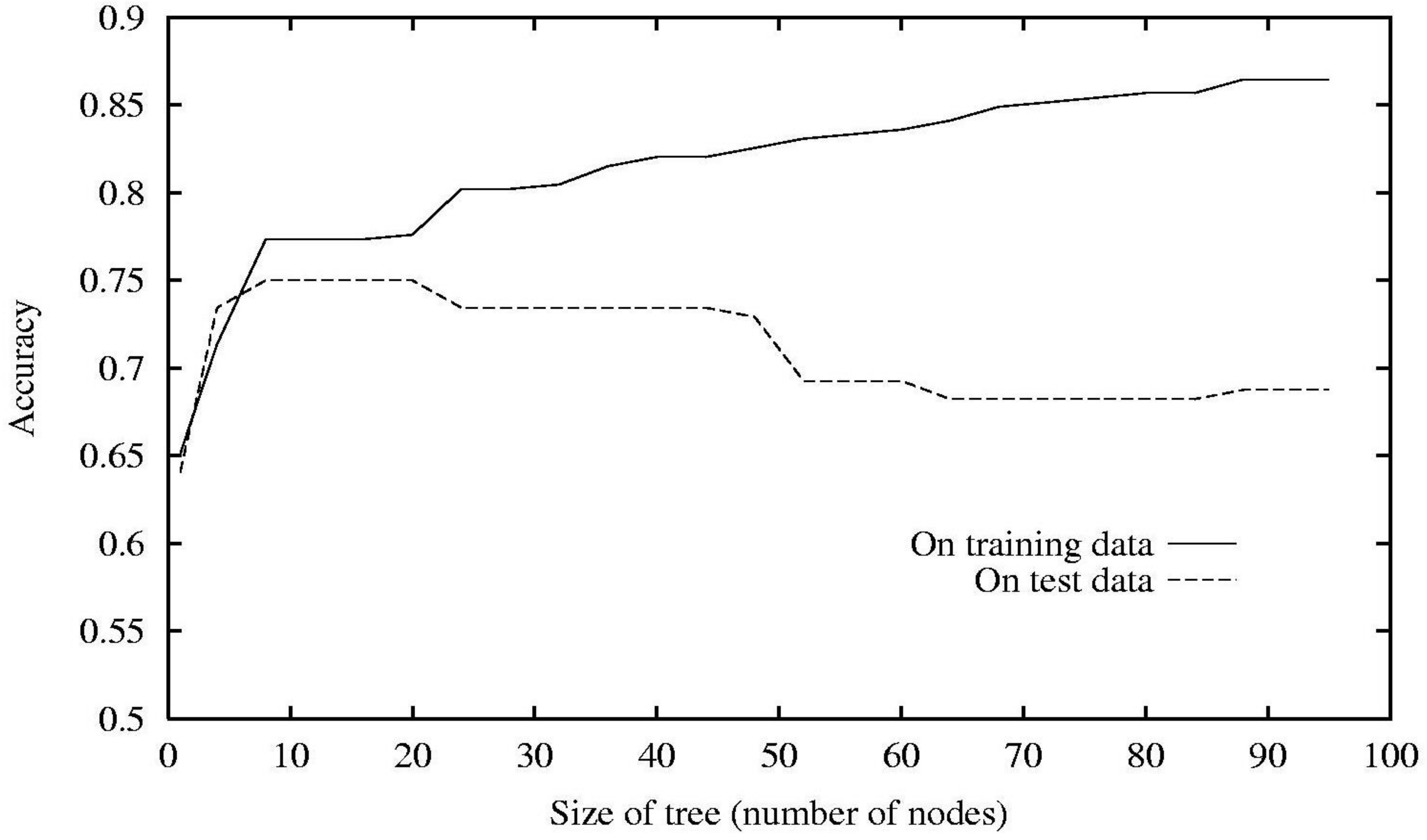


Rule #2 of Machine Learning

The *best* hypothesis almost never achieves 100% accuracy on the training data.

(Rule #1 was: you can’t learn anything without inductive bias)

Overfitting



Overfitting is due to “noise”

- Sources of noise:
  - Erroneous training data
    - concept variable incorrect (annotator error)
    - Attributes mis-measured
  - Much more significant:
    - Irrelevant attributes
    - Target function not deterministic in attributes

Irrelevant attributes

- If many attributes are noisy, information gains can be spurious, e.g.:
    - 20 noisy attributes
    - 10 training examples
    - Expected # of different depth-3 trees that split the training data perfectly using *only* noisy attributes: **13.4**

Non-determinism

- In general:
  - We can’t measure all the variables we need to do perfect prediction.
  - => Target function is not uniquely determined by attribute values

Non-determinism: Example

Humidity	EnjoySport
0.90	0
0.87	1
0.80	0
0.75	0
0.70	1
0.69	1
0.65	1
0.63	1

Decent hypothesis:
Humidity > 0.70 $\rightarrow$ No
Otherwise $\rightarrow$ Yes

Overfit hypothesis:
Humidity > 0.89 $\rightarrow$ No
Humidity > 0.80
^ Humidity <= 0.89 $\rightarrow$ Yes
Humidity > 0.70
^ Humidity <= 0.80 $\rightarrow$ No
Humidity <= 0.70 $\rightarrow$ Yes

Avoiding Overfitting

- Approaches
  - Stop splitting when information gain is low or when split is not statistically significant.
  - Grow full tree and then **prune** it when done

- How to pick the "best" tree?
  - Performance on training data?
  - Performance on validation data?
  - Complexity penalty?

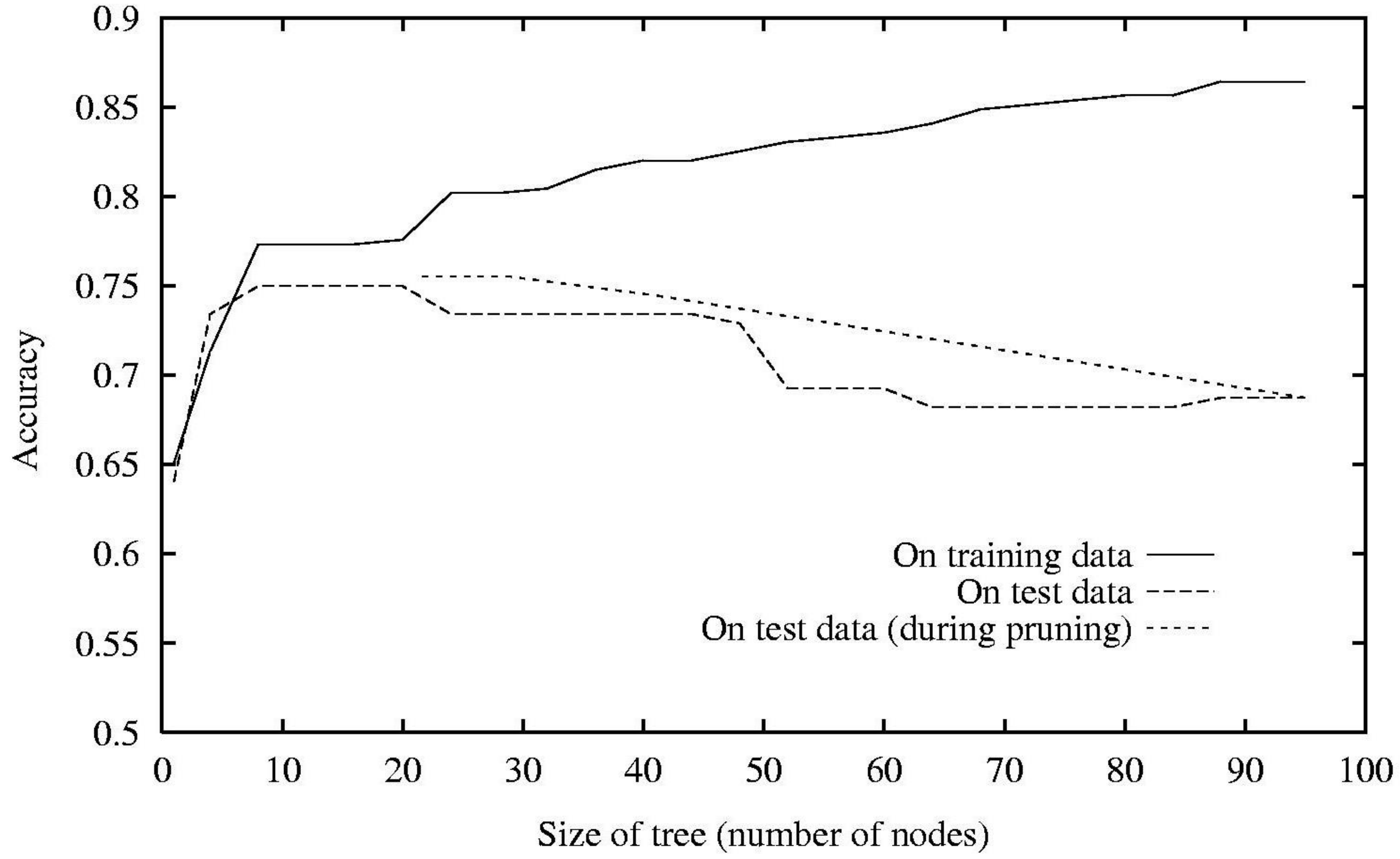
Reduced-Error Pruning

Split data into *training* and *validation* set

Do until further pruning is harmful:

1. Evaluate impact on *validation* set of pruning each possible node (plus those below it)
2. Greedily remove the one that most improves *validation* set accuracy

Effect of Reduced Error Pruning



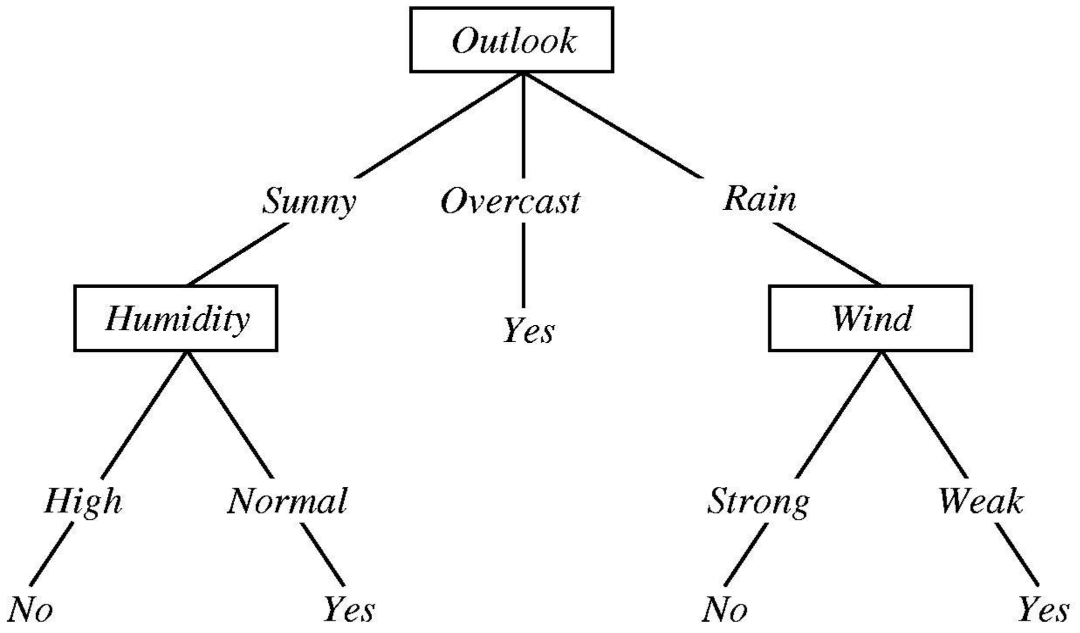
C4.5 Algorithm

- Builds a decision tree from labeled training data
- Also by Ross Quinlan
- Generalizes ID3 by
  - Allowing continuous value attributes
  - Allows missing attributes in examples
  - Prunes tree after building to improve generality

Rule post pruning

- Used in C4.5
- Steps
    1. Build the decision tree
    2. Convert it to a set of logical rules
    3. Prune each rule independently
    4. Sort rules into desired sequence for use

Converting A Tree to Rules



IF	$(Outlook = Sunny)\ AND\ (Humidity = High)$
THEN	$PlayTennis = No$
IF	$(Outlook = Sunny)\ AND\ (Humidity = Normal)$
THEN	$PlayTennis = Yes$

$\ldots$

Scaling Up

- ID3, C4.5, etc. assume data fits in main memory (OK for up to hundreds of thousands of examples)
- SPRINT, SLIQ: multiple sequential scans of data (OK for up to millions of examples)
- VFDT: at most one sequential scan (OK for up to billions of examples)

Unknown Attribute Values

What if some examples are missing values of A?

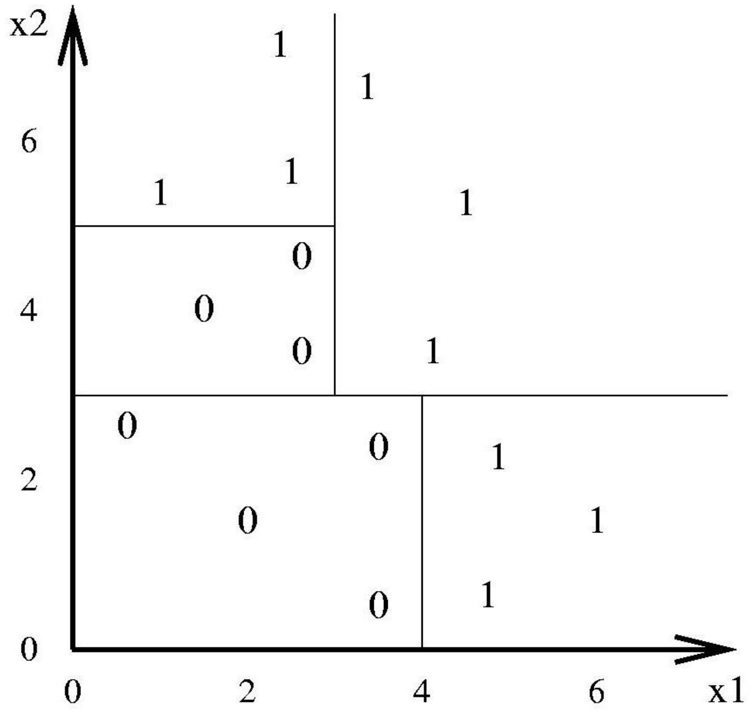
Use training example anyway, sort through tree

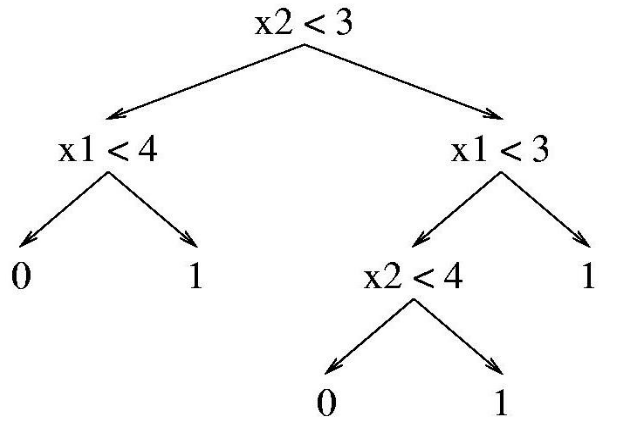
- If node n tests A, assign most common value of A among other examples sorted to node n
- Assign most common value of A among other examples with same target value
- Assign probability p_i to each possible value v_i of A
  Assign fraction p_i of example to each descendant in tree

Classify new examples in same fashion

Decision Tree Boundaries

Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.

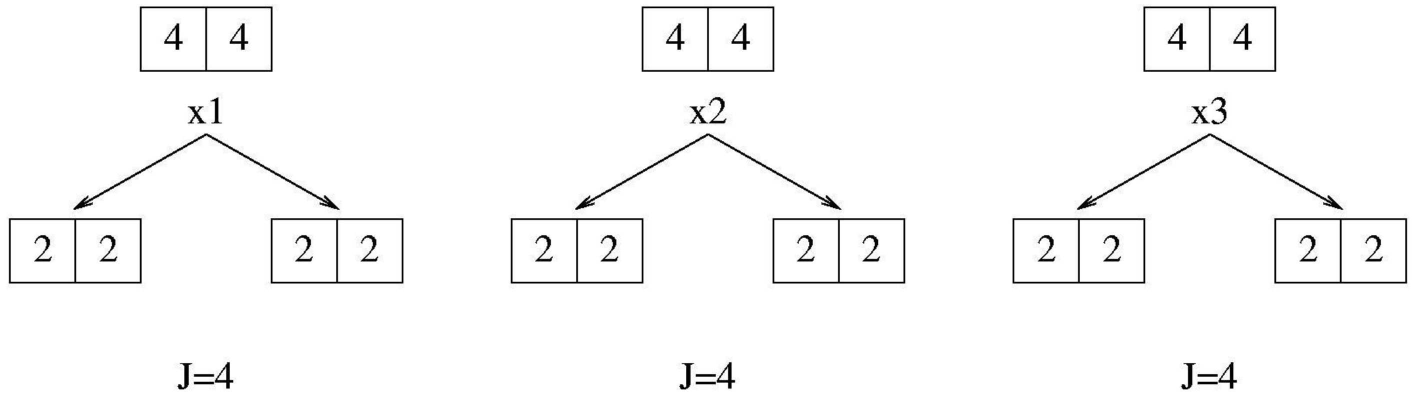




Learning Parity with Noise

When learning exclusive-or (2-bit parity), all splits look equally good. If extra random boolean features are included, they also look equally good. Hence, decision tree algorithms cannot distinguish random noisy features from parity features.

x_1	x_2	x_3	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0



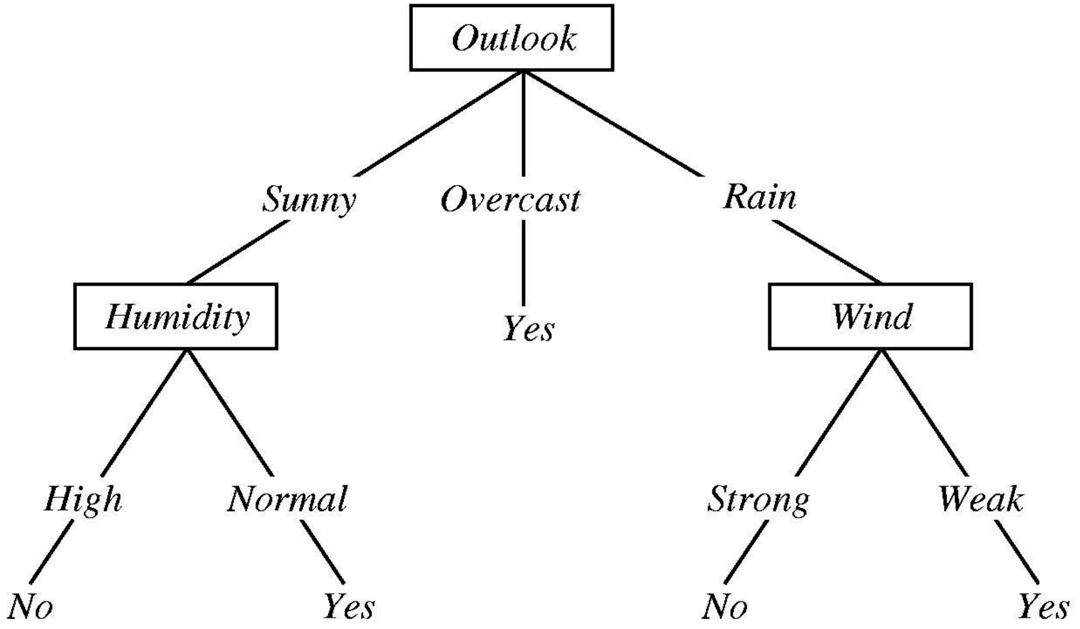
Decision Trees Inductive Bias

- How to solve 2-bit parity:
  - Two step look-ahead, or
  - Split on *pairs* of attributes at once

- For *k*-bit parity, why not just do *k*-step look ahead? Or split on *k* attribute values?

=>Parity functions are the "victims" of the decision tree's inductive bias.

Overfitting in Decision Trees



Consider adding a noisy training example:

Sunny, Hot, Normal, Strong, PlayTennis=No

What effect on tree?

Take away about decision trees

- Used as classifiers
- Supervised learning algorithms (ID3, C4.5)
- (mostly) Batch processing
- Good for situations where
  - The classification categories are finite
  - The data can be represented as vectors of attributes