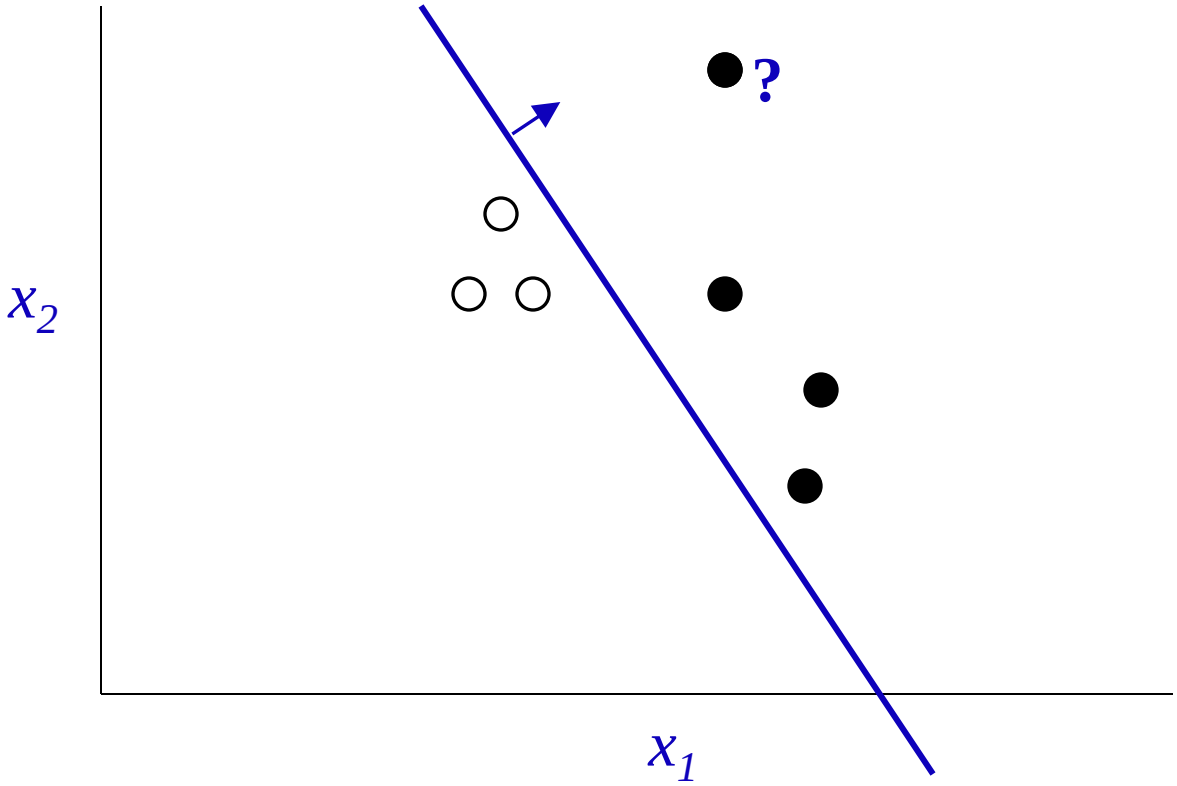
Decision Trees

Doug Downey
EECS 348 Spring 2010
several slides from Pedro Domingos

Recall: Toward “Modern” AI

- Classical AI Limitations:
  - Knowledge Acquisition Bottleneck, Brittleness
- “Modern” directions:
  - Situatedness, embodiment
  - **Learning from data (machine learning)**
  - Probability

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



General Machine Learning Task

DEFINE:

- Set *X* of Instances (of *n*-tuples **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 $X \rightarrow Y = \{0,1\}$ (our running example)
  - 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 $y(\mathbf{x})$.

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

Inductive Learning Hypothesis

- Any hypothesis found to approximate the target function well over the training examples will also approximate the target function well over unobserved examples.

Number of Instances, Concepts, Hypotheses

- **Sky**: Sunny, Cloudy, Rainy
- **AirTemp**: Warm, Cold
- **Humidity**: Normal, High
- **Wind**: Strong, Weak
- **Water**: Warm, Cold
- **Forecast**: Same, Change

distinct **instances** : $3*2*2*2*2*2 = 96$

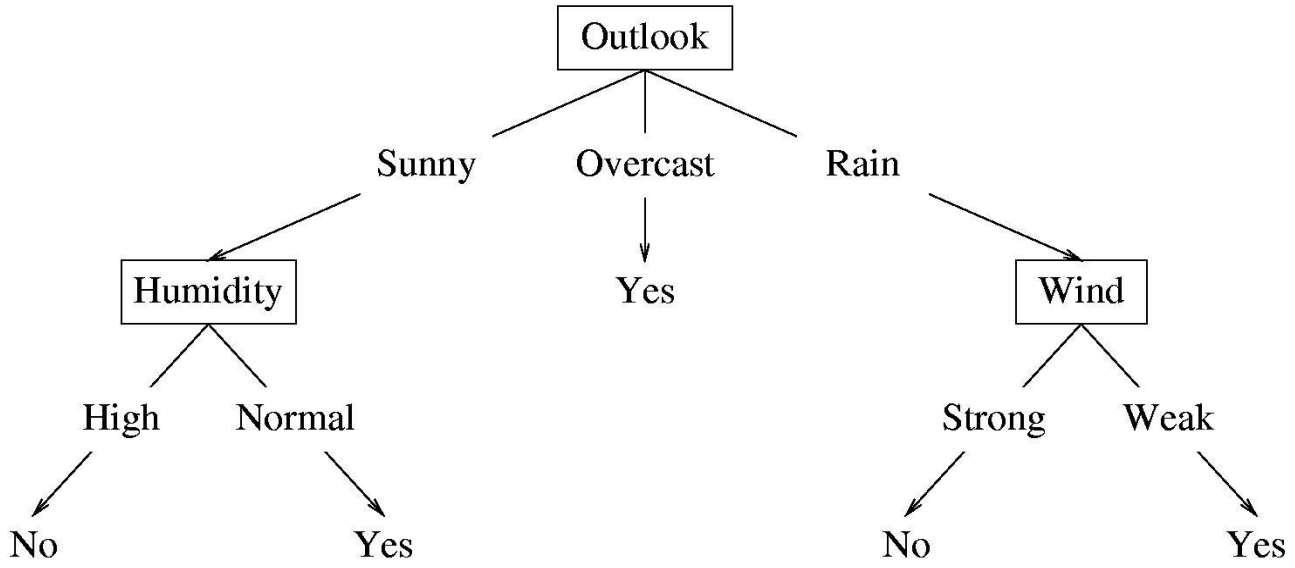
distinct **concepts**: 2^{96}

distinct conjunction-of-constraints **hypotheses**:

$$1 + 4*3*3*3*3*3 = 973$$

Decision Tree Hypothesis Space

- **Internal nodes** test the value of particular features x_j and branch according to the results of the test.
- **Leaf nodes** specify the class $h(\mathbf{x})$.



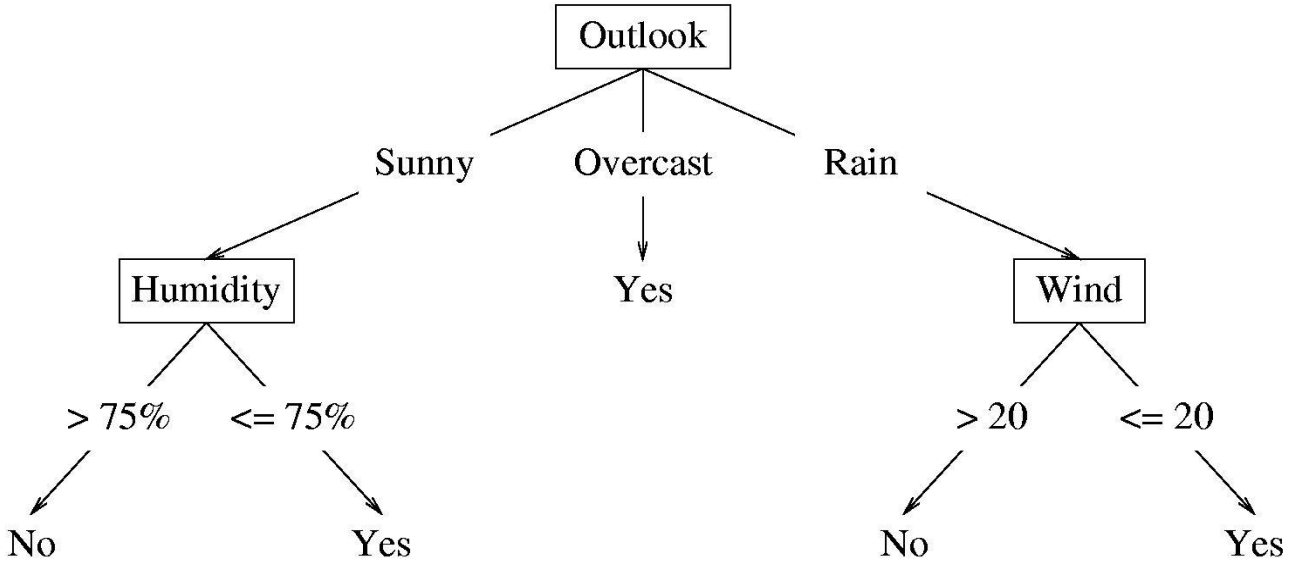
Suppose the features are **Outlook** (x_1), **Temperature** (x_2), **Humidity** (x_3), and **Wind** (x_4). Then the feature vector $\mathbf{x} = (Sunny, Hot, High, Strong)$ will be classified as **No**. The **Temperature** feature is irrelevant.

Inductive Bias

- To learn, we **must** prefer some concepts to others
  - **Selection bias**
    - use a restricted hypothesis space (e.g., linear separators)
  - **Preference bias**
    - use the whole concept space, but state a preference over concepts (e.g., decision trees)

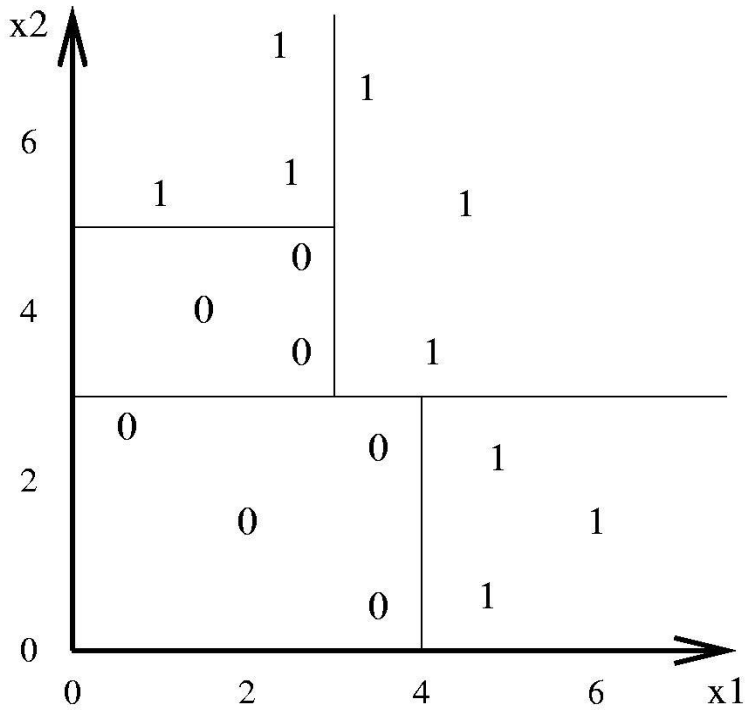
Decision Tree Hypothesis Space

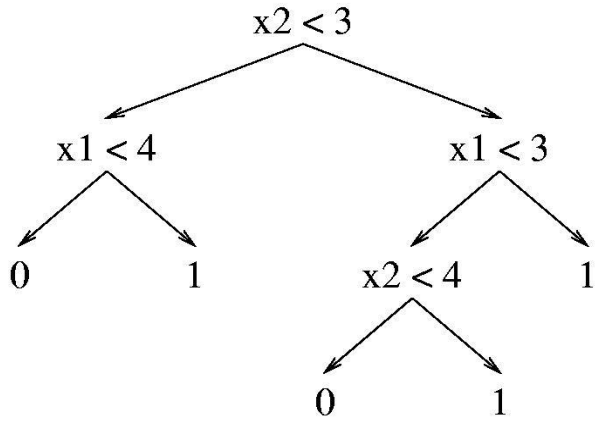
If the features are continuous, internal nodes may test the value of a feature against a threshold.



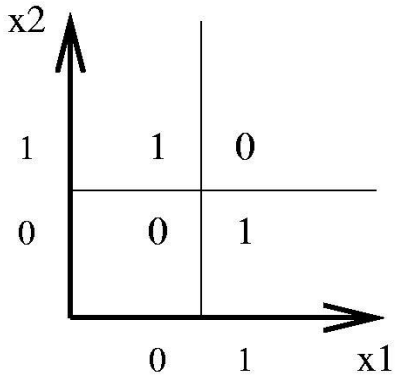
Decision Tree Decision Boundaries

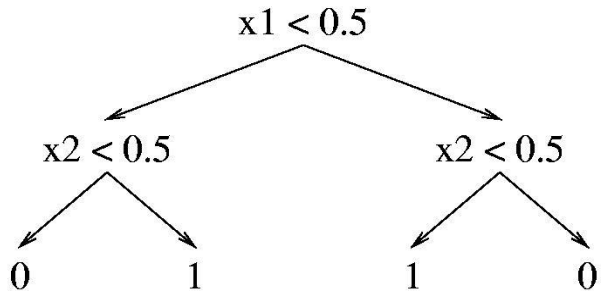
Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the K classes.





Decision Trees Can Represent Any Boolean Function





The tree will in the worst case require exponentially many nodes, however.

Learning Algorithm for Decision Trees

The same basic learning algorithm has been discovered by many people independently:

GrowTree(S)
if ($y = 0$ for all $\langle \mathbf{x}, y \rangle \in S$) **return** new leaf(0)
else if ($y = 1$ for all $\langle \mathbf{x}, y \rangle \in S$) **return** new leaf(1)
else
    choose best attribute x_j
    $S_0 =$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 0$;
    $S_1 =$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 1$;
    return new node(x_j, GrowTree(S_0), GrowTree(S_1))

Choosing the Best Attribute

One way to choose the best attribute is to perform a 1-step lookahead search and choose the attribute that gives the lowest error rate on the training data.

ChooseBestAttribute(S)

choose j to minimize J_j, computed as follows:

$S_0 =$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 0$;

$S_1 =$ all $\langle \mathbf{x}, y \rangle \in S$ with $x_j = 1$;

$y_0 =$ the most common value of y in S_0

$y_1 =$ the most common value of y in S_1

$J_0 =$ number of examples $\langle \mathbf{x}, y \rangle \in S_0$ with $y \neq y_0$

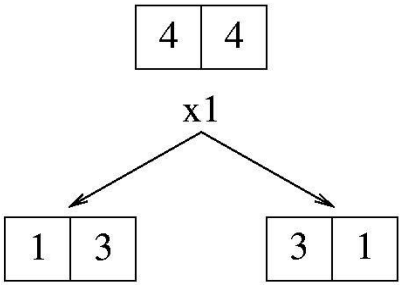
$J_1 =$ number of examples $\langle \mathbf{x}, y \rangle \in S_1$ with $y \neq y_1$

$J_j = J_0 + J_1$ (total errors if we split on this feature)

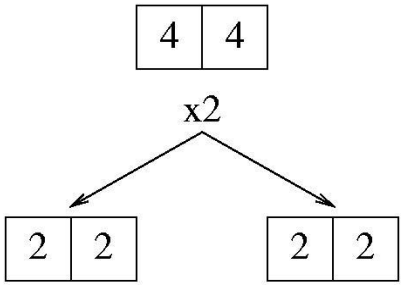
return j

Choosing the Best Attribute—An Example

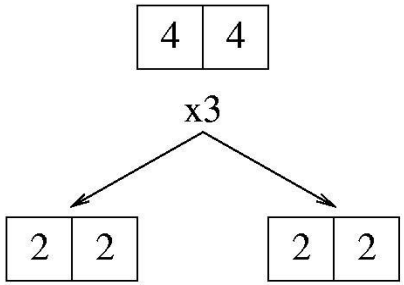
x_1	x_2	x_3	y
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



J=2



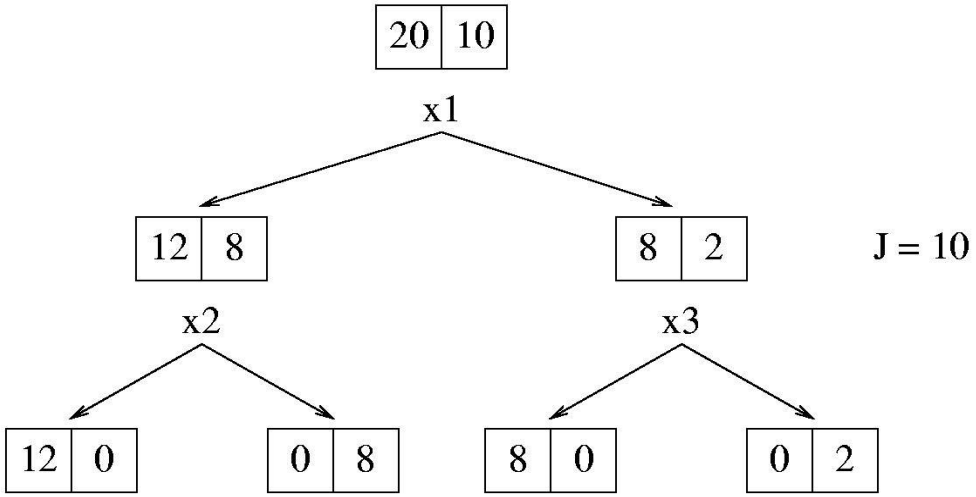
J=4



J=4

Choosing the Best Attribute (3)

Unfortunately, this measure does not always work well, because it does not detect cases where we are making "progress" toward a good tree.



A Better Heuristic From Information Theory

Let V be a random variable with the following probability distribution:

$P(V = 0)$	$P(V = 1)$
0.2	0.8

The *surprise*, $S(V = v)$ of each value of V is defined to be

$$S(V = v) = -\lg P(V = v).$$

An event with probability 1 gives us zero surprise.

An event with probability 0 gives us infinite surprise!

It turns out that the surprise is equal to the number of bits of information that need to be transmitted to a recipient who knows the probabilities of the results.

This is also called the *description length* of $V = v$.

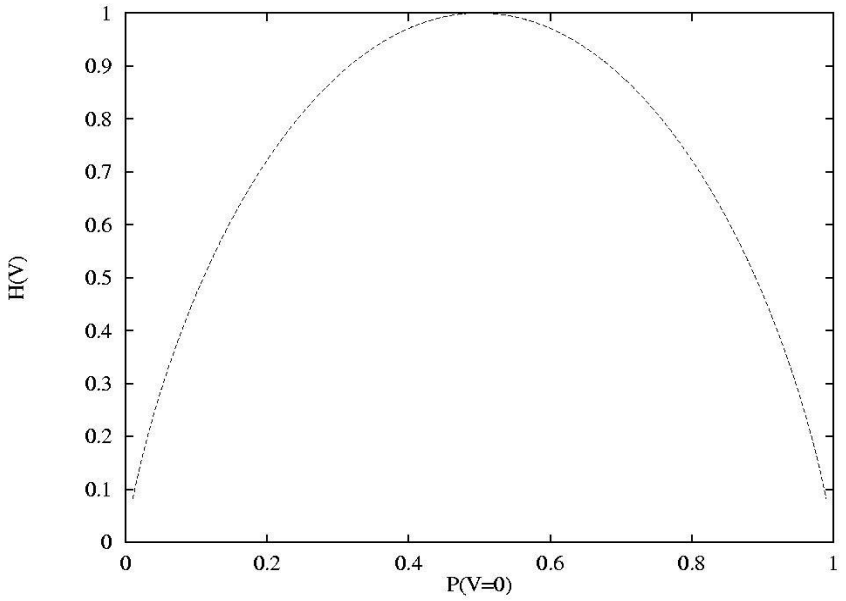
Fractional bits only make sense if they are part of a longer message (e.g., describe a whole sequence of coin tosses).

Entropy

The *entropy* of V, denoted $H(V)$ is defined as follows:

$$H(V) = \sum_{v=0}^{1} -P(H = v) \lg P(H = v).$$

This is the average surprise of describing the result of one "trial" of V (one coin toss).



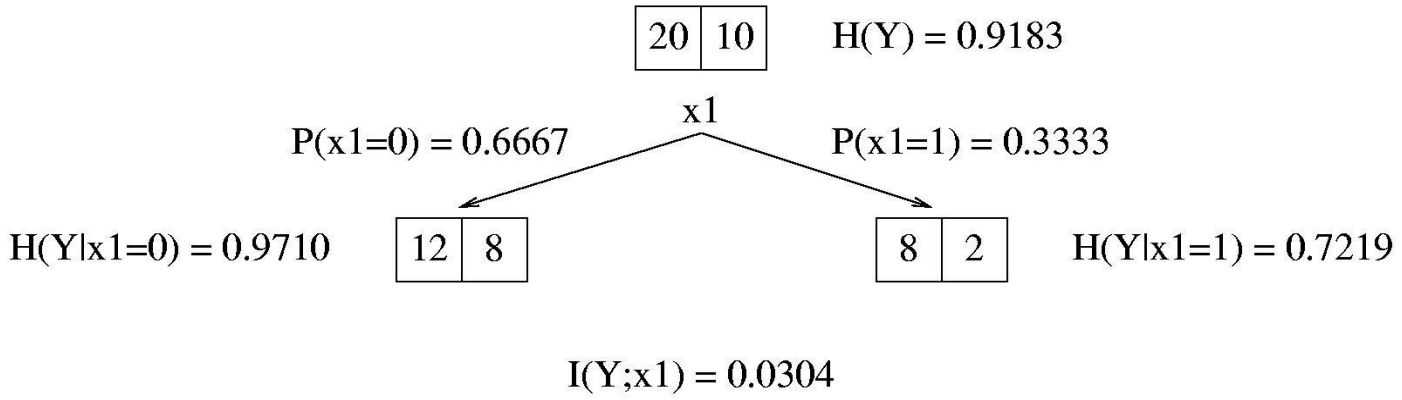
Entropy can be viewed as a measure of uncertainty.

Mutual Information

Now consider two random variables A and B that are not necessarily independent. The *mutual information* between A and B is the amount of information we learn about B by knowning the value of A (and vice versa—it is symmetric). It is computed as follows:

$$I(A;B) = H(B) - \sum_b P(B=b) \cdot H(A|B=b)$$

In particular, consider the class Y of each training example and the value of feature x_1 to be random variables. Then the mutual information quantifies how much x_1 tells us about the value of the class Y.



Non-Boolean Features

- **Features with multiple discrete values**

  Construct a multiway split?

  Test for one value versus all of the others?

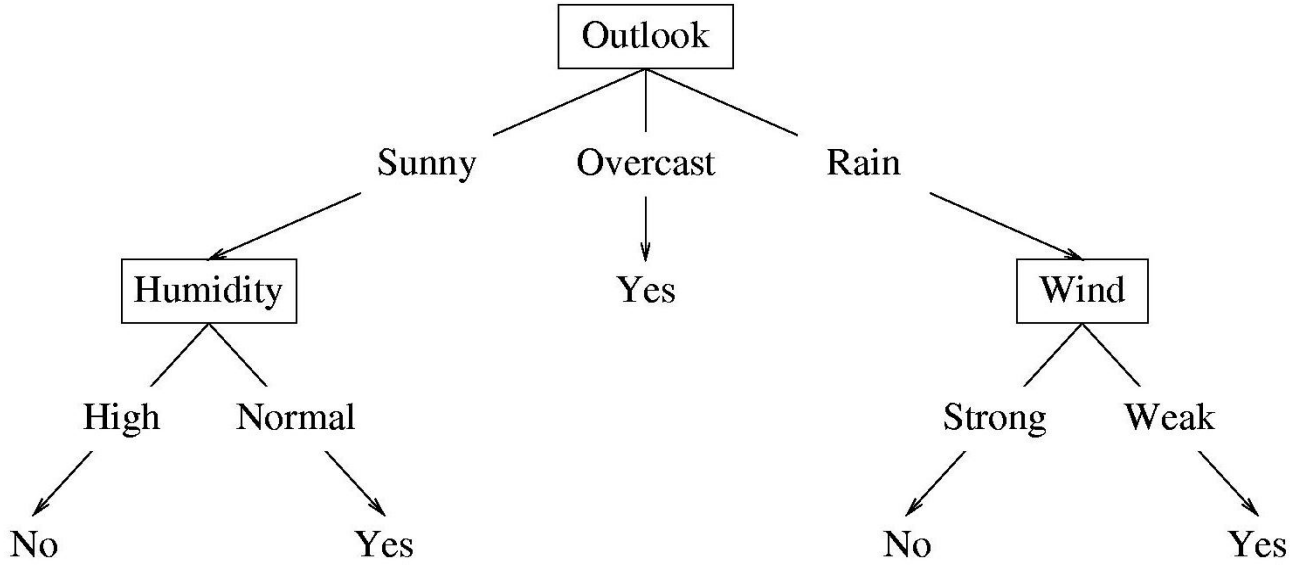
  Group the values into two disjoint subsets?

- **Real-valued features**

  Consider a threshold split using each observed value of the feature.

Whichever method is used, the mutual information can be computed to choose the best split.

Decision Trees represent *disjunctions* of *conjunctions*

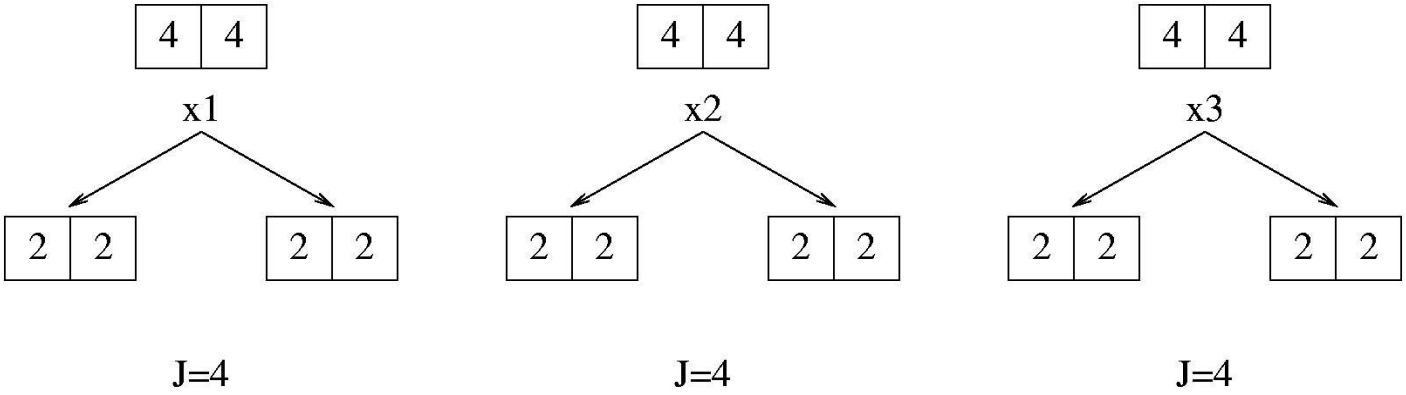


=(Sunny ^ Normal) v Overcast v (Rain ^ Weak)

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



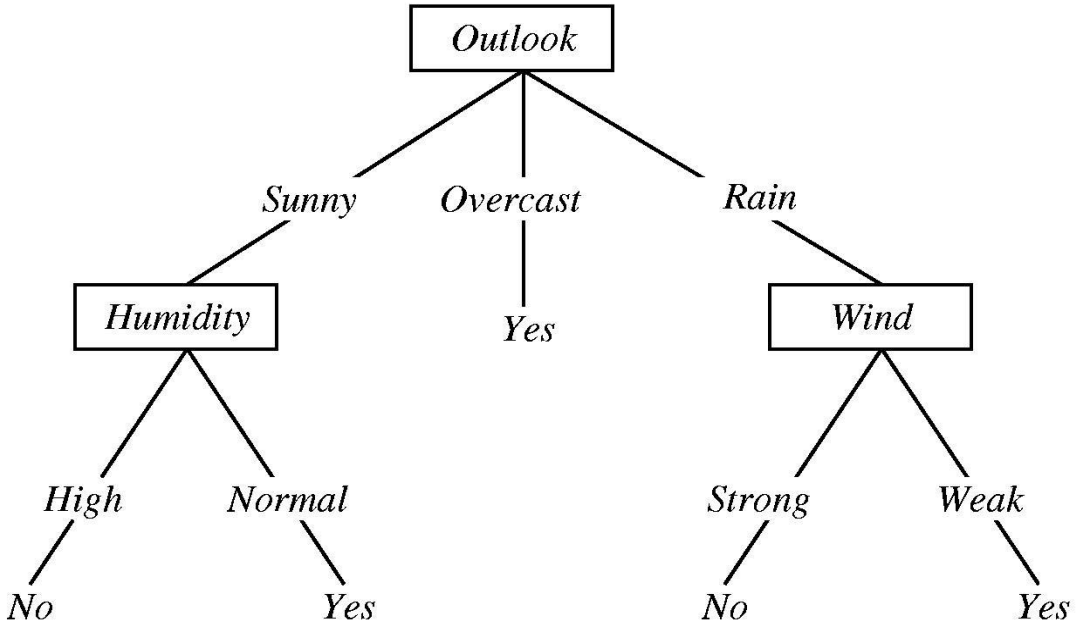
Decision Trees Inductive Bias

- 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?

Overfitting

Consider error of hypothesis h over

- training data: $error_{train}(h)$
- entire distribution $\mathcal{D}$ of data: $error_{\mathcal{D}}(h)$

Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

$$error_{train}(h) < error_{train}(h')$$

and

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

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 depth-3 trees that split the training data perfectly using *only* noisy attributes: 13.4
- Potential solution: statistical significance tests (e.g., chi-square)

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 → No

Otherwise → Yes

Overfit hypothesis:

Humidity > 0.89 → No

Humidity > 0.80

^ Humidity <= 0.89 → Yes

Humidity > 0.70

^ Humidity <= 0.80 → No

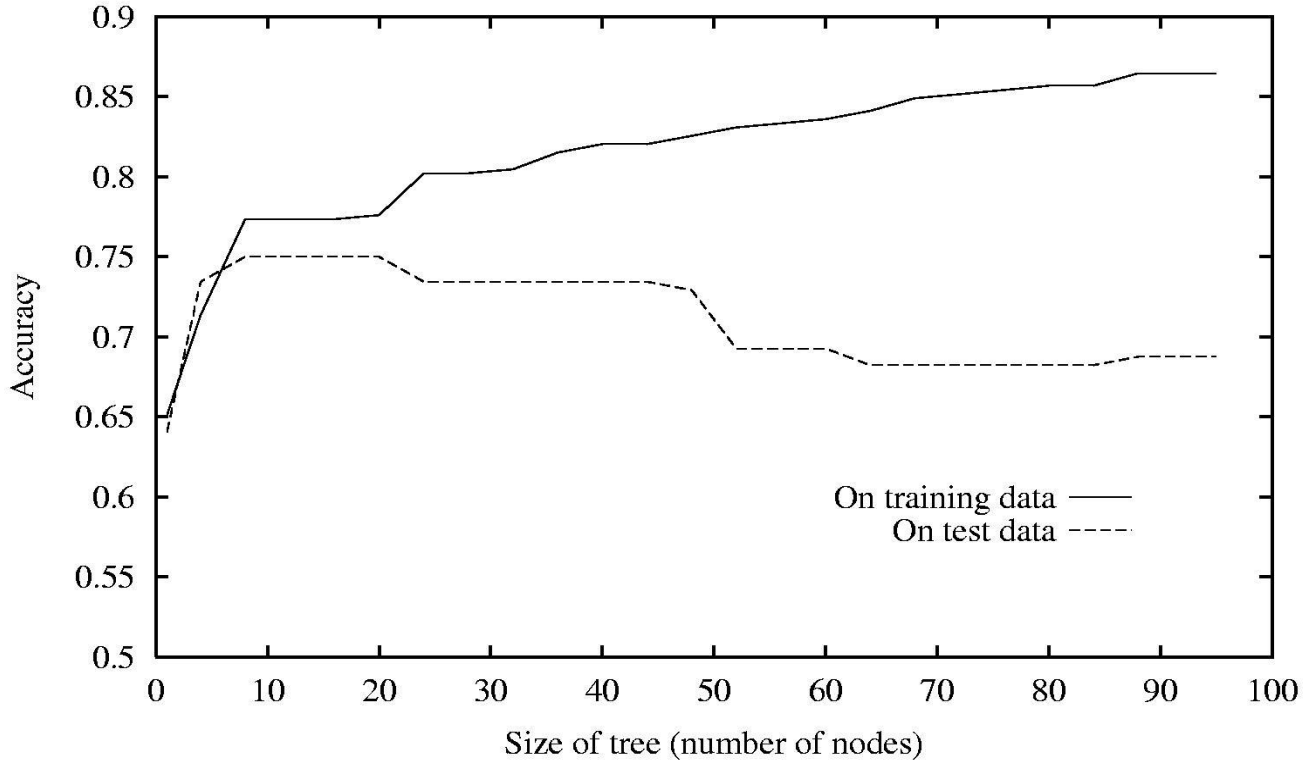
Humidity <= 0.70 → Yes

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 in Decision Tree Learning



Avoiding Overfitting

How can we avoid overfitting?

- Stop growing when data split not statistically significant
- Grow full tree, then post-prune

How to select "best" tree:

- Measure performance over training data
- Measure performance over separate validation data set
- Add complexity penalty to performance measure

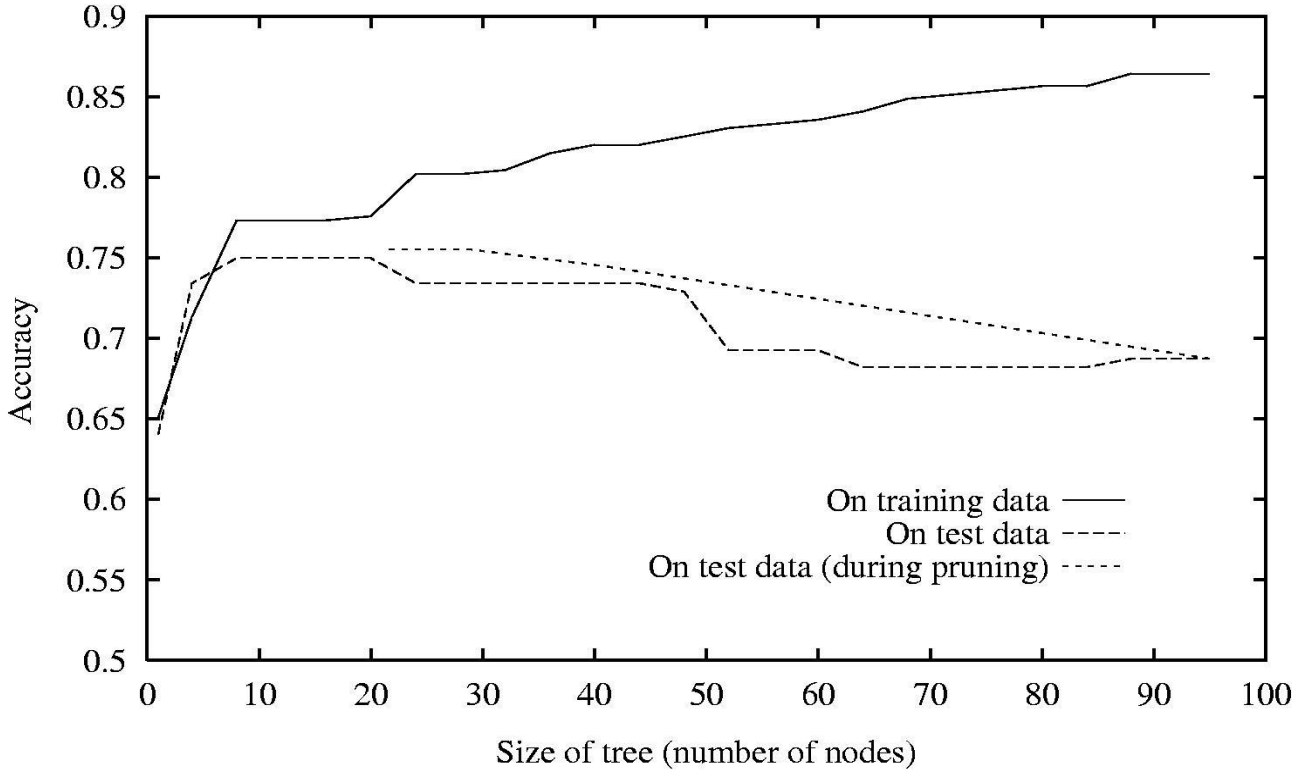
Reduced-Error Pruning

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

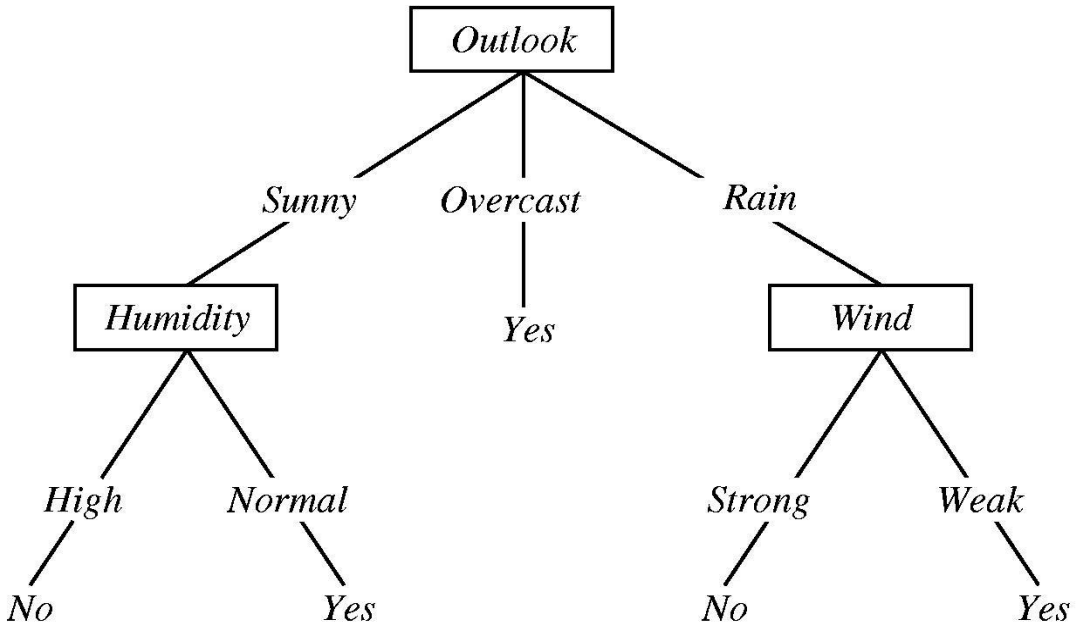


Rule Post-Pruning

1. Convert tree to equivalent set of rules
2. Prune each rule independently of others
3. Sort final rules into desired sequence for use

Perhaps most frequently used method (e.g., C4.5)

Converting A Tree to Rules



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

IF $(Outlook = Sunny)\ AND\ (Humidity = Normal)$
THEN $PlayTennis = Yes$

...

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)

Hypothesis Space comparisons

Task: concept learning with *k* binary attributes

Hypothesis Space H	# of Semantically distinct *h*
Rote Learning	2^{2^k}
MC2	$1 + 3^k$
1-level decision tree	k
n-level decision tree	$\min\left(\prod_{i=0}^{n-1}(k-i)^{2(i+1)}, 2^{2^k}\right)$

Decision Trees – Strengths

- Very Popular Technique
- Fast
- Useful when
  - Instances are attribute-value pairs
  - Target Function is discrete
  - Concepts are likely to be disjunctions
  - Attributes may be noisy

Attributes with Many Values

Problem:

- If attribute has many values, $Gain$ will select it
- Imagine using $Date = Jun\_3\_1996$ as attribute

One approach: use $GainRatio$ instead

$$GainRatio(S, A) \equiv \frac{Gain(S, A)}{SplitInformation(S, A)}$$

$$SplitInformation(S, A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

where S_i is subset of S for which A has value v_i

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