# Decision Trees

Instructor: Alan Ritter

Many slides from Tom Mitchell, Pedro Domingos

# Supervised Learning: find $f$

- **Given:** Training set $\{(x_i, y_i) \mid i = 1 \ldots n\}$
- **Find:** A good approximation to $f : X \rightarrow Y$

## Examples: what are $X$ and $Y$ ?

- Spam Detection
  - Map email to {Spam,Ham}
- Digit recognition
  - Map pixels to {0,1,2,3,4,5,6,7,8,9}
- Stock Prediction
  - Map new, historic prices, etc. to $\Re$ (the real numbers)

# A Supervised Learning Problem

- Consider a simple, Boolean dataset:
  - $f : X \rightarrow Y$
  - $X = \{0,1\}^4$
  - $Y = \{0,1\}$
- Question 1: How should we pick the *hypothesis space*, the set of possible functions $f$?
- Question 2: How do we find the best $f$ in the hypothesis space?

Dataset:

| Example | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $y$ |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 | 1 |
| 4 | 1 | 0 | 0 | 1 | 1 |
| 5 | 0 | 1 | 1 | 0 | 0 |
| 6 | 1 | 1 | 0 | 0 | 0 |
| 7 | 0 | 1 | 0 | 1 | 0 |

# Most General Hypothesis Space

Consider all possible boolean functions over four input features!

- $2^{16}$ possible hypotheses
- $2^9$ are consistent with our dataset
- How do we choose the best one?

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $y$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | ? |
| 0 | 0 | 0 | 1 | ? |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 | ? |
| 1 | 0 | 0 | 0 | ? |
| 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | ? |
| 1 | 0 | 1 | 1 | ? |
| 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 1 | ? |
| 1 | 1 | 1 | 0 | ? |
| 1 | 1 | 1 | 1 | ? |

Dataset:

| Example | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $y$ |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 | 1 |
| 4 | 1 | 0 | 0 | 1 | 1 |
| 5 | 0 | 1 | 1 | 0 | 0 |
| 6 | 1 | 1 | 0 | 0 | 0 |
| 7 | 0 | 1 | 0 | 1 | 0 |

# A Restricted Hypothesis Space

Consider all conjunctive boolean functions.

- 16 possible hypotheses
- None are consistent with our dataset
- How do we choose the best one?

| Rule | Counterexample |
|---|---|
| $\Rightarrow y$ | 1 |
| $x_1 \Rightarrow y$ | 3 |
| $x_2 \Rightarrow y$ | 2 |
| $x_3 \Rightarrow y$ | 1 |
| $x_4 \Rightarrow y$ | 7 |
| $x_1 \wedge x_2 \Rightarrow y$ | 3 |
| $x_1 \wedge x_3 \Rightarrow y$ | 3 |
| $x_1 \wedge x_4 \Rightarrow y$ | 3 |
| $x_2 \wedge x_3 \Rightarrow y$ | 3 |
| $x_2 \wedge x_4 \Rightarrow y$ | 3 |
| $x_3 \wedge x_4 \Rightarrow y$ | 4 |
| $x_1 \wedge x_2 \wedge x_3 \Rightarrow y$ | 3 |
| $x_1 \wedge x_2 \wedge x_4 \Rightarrow y$ | 3 |
| $x_1 \wedge x_3 \wedge x_4 \Rightarrow y$ | 3 |
| $x_2 \wedge x_3 \wedge x_4 \Rightarrow y$ | 3 |
| $x_1 \wedge x_2 \wedge x_3 \wedge x_4 \Rightarrow y$ | 3 |

Dataset:

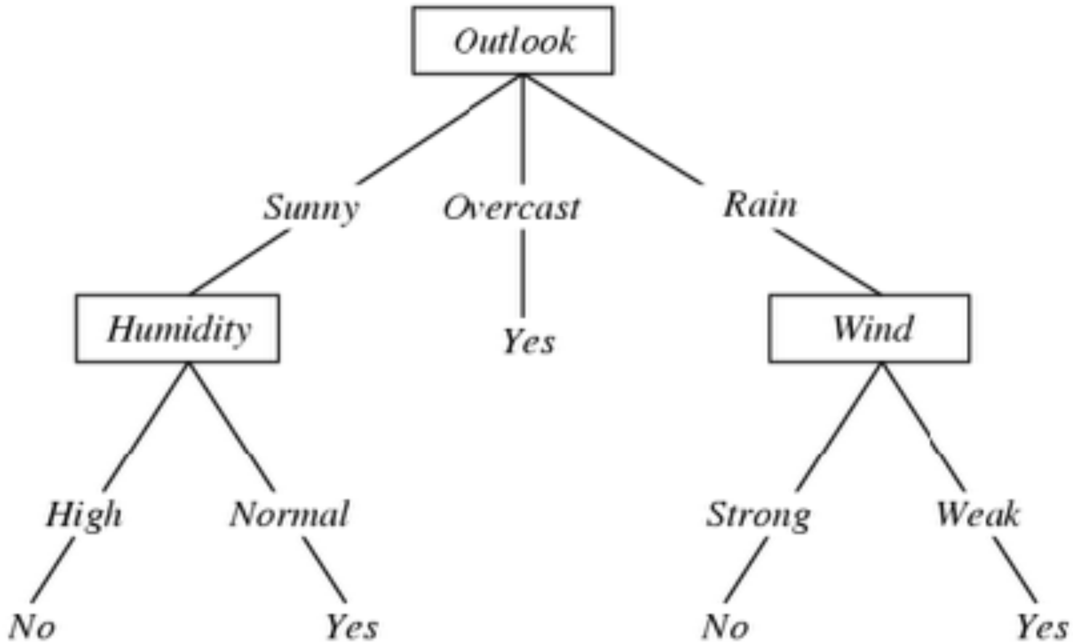
| Example | $x_1$ | $x_2$ | $x_3$ | $x_4$ | $y$ |
|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 | 0 |
| 3 | 0 | 0 | 1 | 1 | 1 |
| 4 | 1 | 0 | 0 | 1 | 1 |
| 5 | 0 | 1 | 1 | 0 | 0 |
| 6 | 1 | 1 | 0 | 0 | 0 |
| 7 | 0 | 1 | 0 | 1 | 0 |

# Simple Training Data Set

| Day | Outlook | Temperature | Humidity | Wind | PlayTennis? |
|---|---|---|---|---|---|
| D1 | Sunny | Hot | High | Weak | No |
| D2 | Sunny | Hot | High | Strong | No |
| D3 | Overcast | Hot | High | Weak | Yes |
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D6 | Rain | Cool | Normal | Strong | No |
| D7 | Overcast | Cool | Normal | Strong | Yes |
| D8 | Sunny | Mild | High | Weak | No |
| D9 | Sunny | Cool | Normal | Weak | Yes |
| D10 | Rain | Mild | Normal | Weak | Yes |
| D11 | Sunny | Mild | Normal | Strong | Yes |
| D12 | Overcast | Mild | High | Strong | Yes |
| D13 | Overcast | Hot | Normal | Weak | Yes |
| D14 | Rain | Mild | High | Strong | No |

# A Decision tree for

f: <Outlook, Temperature, Humidity, Wind> → PlayTennis?




Each internal node: test one discrete-valued attribute $X_i$

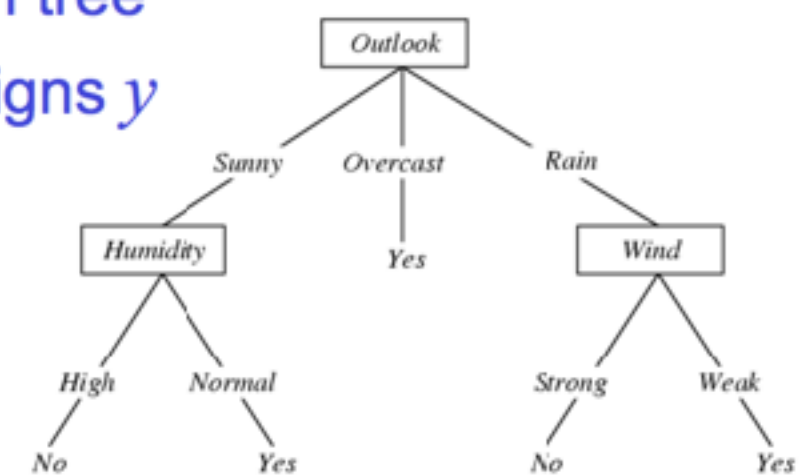
Each branch from a node: selects one value for $X_i$

Each leaf node: predict Y (or $P(Y|X \in \text{leaf})$)

# Decision Tree Learning

**Problem Setting**:

- Set of possible instances $X$
  - each instance $x$ in $X$ is a feature vector
  - e.g., *<Humidity=low, Wind=weak, Outlook=rain, Temp=hot>*
- Unknown target function $f : X \rightarrow Y$
  - $Y$=1 if we play tennis on this day, else 0
- Set of function hypotheses $H=\{ h \mid h : X \rightarrow Y \}$
  - each hypothesis $h$ is a decision tree
  - trees sorts $x$ to leaf, which assigns $y$



# Decision Tree Learning

**Problem Setting**:

- Set of possible instances $X$
  - each instance $x$ in $X$ is a feature vector $x = < x_1, x_2 \ldots x_n>$
- Unknown target function $f : X \rightarrow Y$
  - $Y$ is discrete-valued
- Set of function hypotheses $H=\{ h \mid h : X \rightarrow Y \}$
  - each hypothesis $h$ is a decision tree

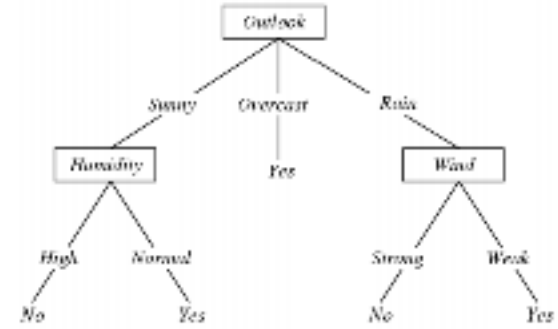
**Input**:

- Training examples $\{<x^{(i)},y^{(i)}>\}$ of unknown target function $f$

**Output**:

- Hypothesis $h \in H$ that best approximates target function $f$

# Decision Trees




Suppose $X = <X_1, \dots X_n>$
where $X_i$ are boolean-valued variables

How would you represent $Y = X_2 X_5$ ? $\quad Y = X_2 \vee X_5$

How would you represent $X_2 X_5 \vee X_3 X_4 (\neg X_1)$

# A Tree to Predict C-Section Risk

Learned from medical records of 1000 women

Negative examples are C-sections

```
[833+,167-] .83+ .17-
Fetal_Presentation = 1: [822+,116-] .88+ .12-
| Previous_Csection = 0: [767+,81-] .90+ .10-
| | Primiparous = 0: [399+,13-] .97+ .03-
| | Primiparous = 1: [368+,68-] .84+ .16-
| | | Fetal_Distress = 0: [334+,47-] .88+ .12-
| | | | Birth_Weight < 3349: [201+,10.6-] .95+ .
| | | | Birth_Weight >= 3349: [133+,36.4-] .78+
| | | Fetal_Distress = 1: [34+,21-] .62+ .38-
| Previous_Csection = 1: [55+,35-] .61+ .39-
Fetal_Presentation = 2: [3+,29-] .11+ .89-
Fetal_Presentation = 3: [8+,22-] .27+ .73-
```

## Learning Algorithm for Decision Trees

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

$\textsc{GrowTree}(S)$
**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($x_j$, $\textsc{GrowTree}(S_0)$, $\textsc{GrowTree}(S_1)$)

# 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 $y \neq y_0$

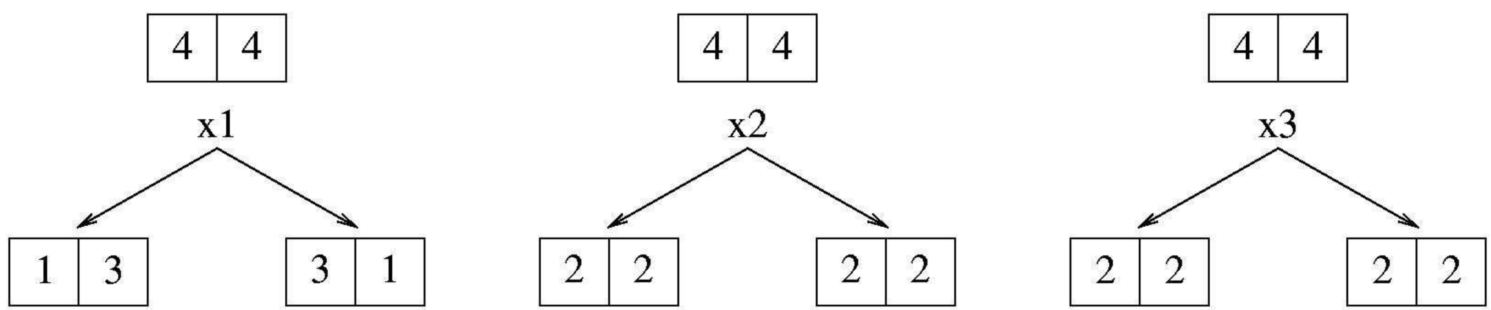
$J_1 =$ number of examples $\langle \mathbf{x}, y\rangle \in S_1$ with $y \neq y_1$

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

**return** $j$

# Choosing the Best Attribute—An Example

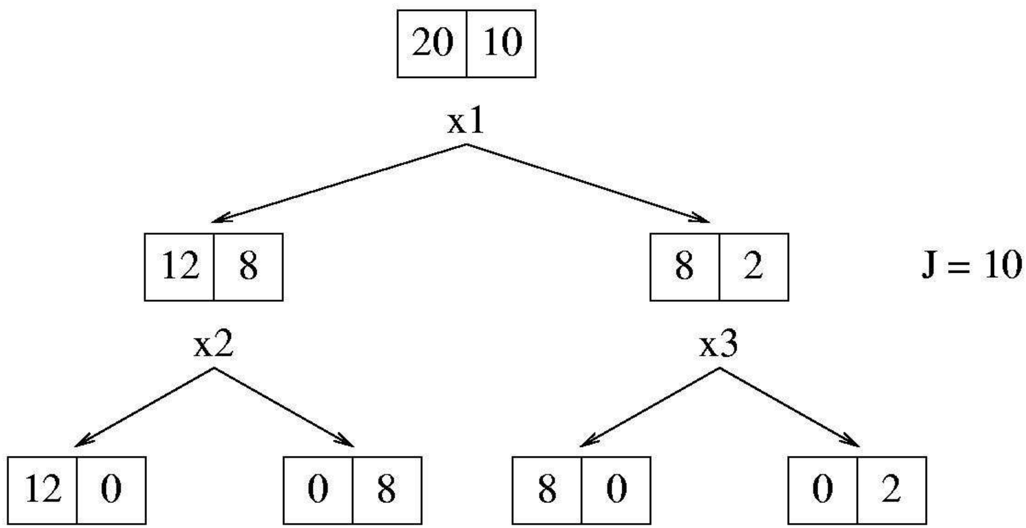
| $x_1$ | $x_2$ | $x_3$ | $y$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |




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 $V = v$.

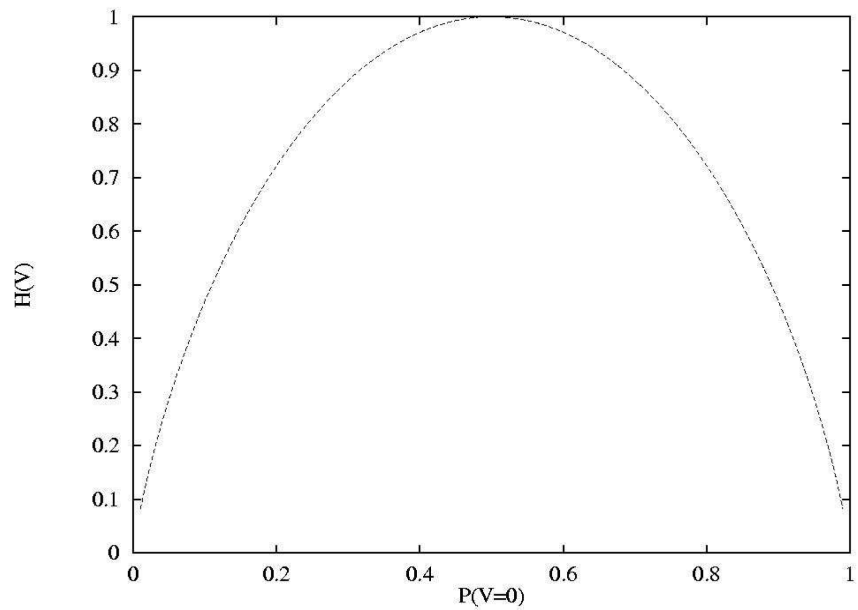
Fractional bits only make sense if they are part of a longer message (e.g., describe a whole sequence of coin tosses).

# 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).



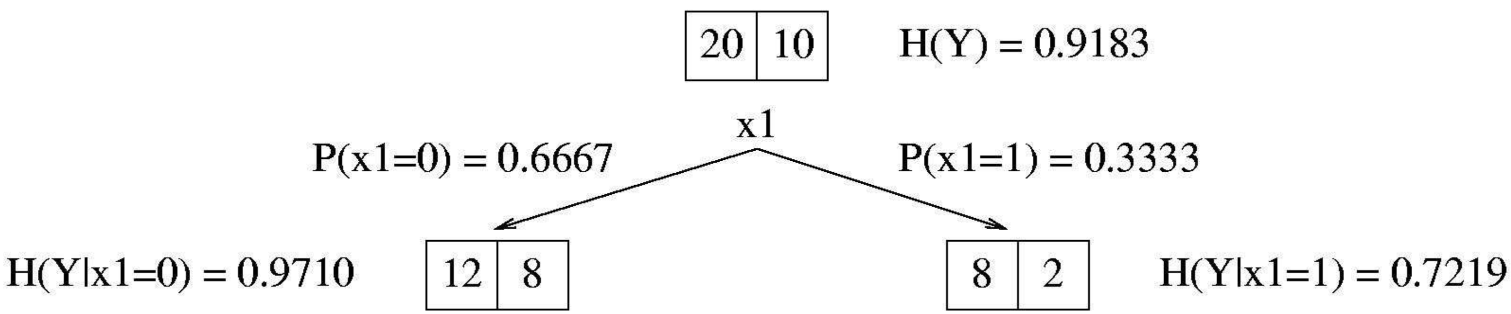

Entropy can be viewed as a measure of uncertainty.

# 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:

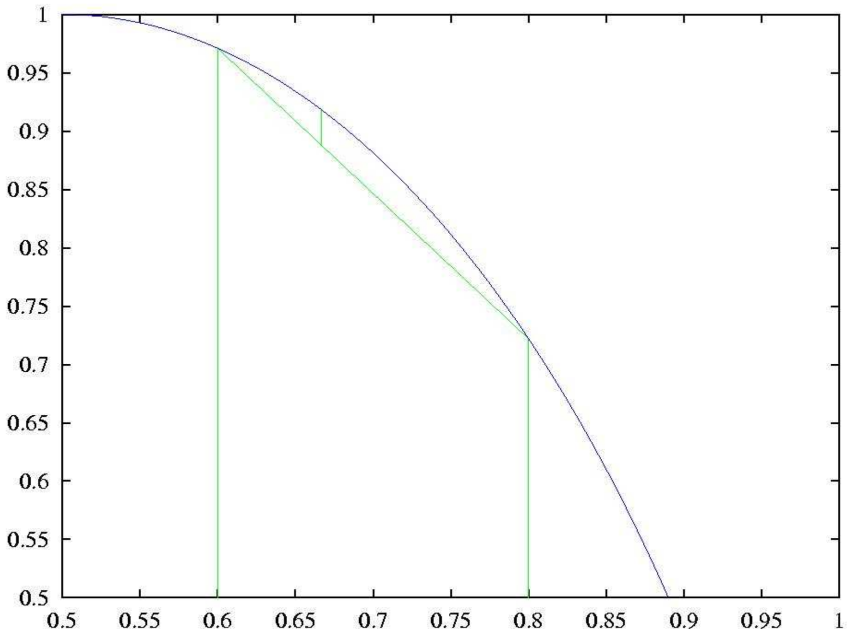
$$I(A;B) = H(B) - \sum_b P(B=b) \cdot H(A|B=b)$$

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 $Y$.

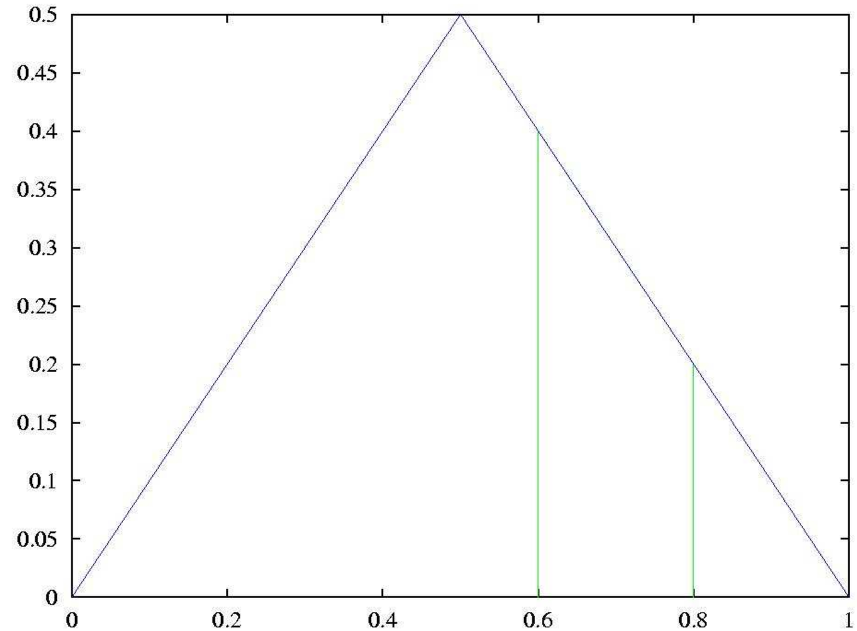



I(Y;x1) = 0.0304

# Visualizing Heuristics




Entropy




Absolute Error

Mutual information works because it is a convex measure.

# 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**

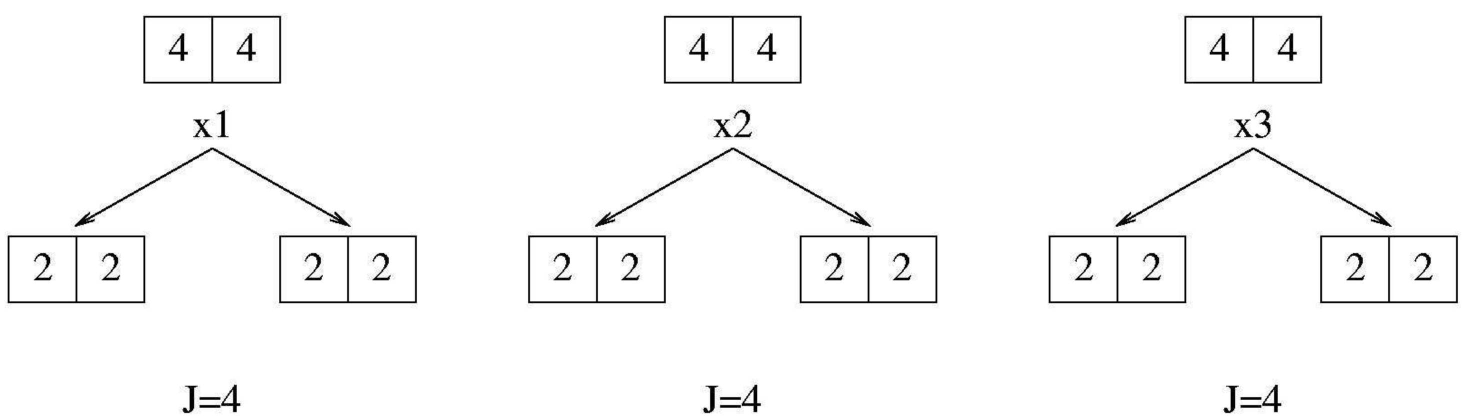
  Consider a threshold split using each observed value of the feature.

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

# 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 |



# 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

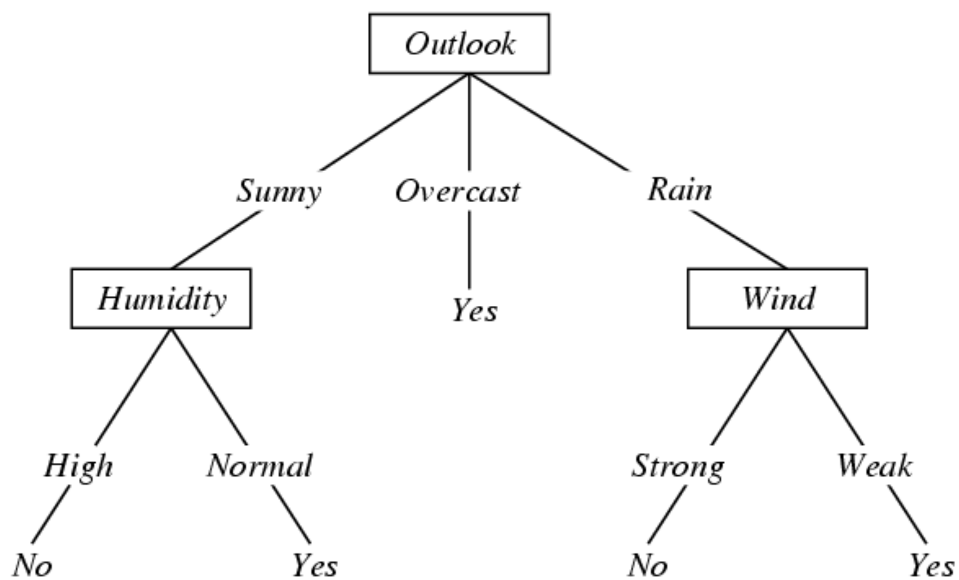
Classify new examples in same fashion

# Overfitting in Decision Trees

Consider adding noisy training example #15:

$Sunny,\ Hot,\ Normal,\ Strong,\ PlayTennis = No$

What effect on earlier tree?



# Overfitting

Consider a hypothesis $h$ and its

- Error rate over training data: $error_{train}(h)$
- True error rate over all data: $error_{true}(h)$

# Overfitting

Consider a hypothesis $h$ and its

- Error rate over training data: $error_{train}(h)$
- True error rate over all data: $error_{true}(h)$

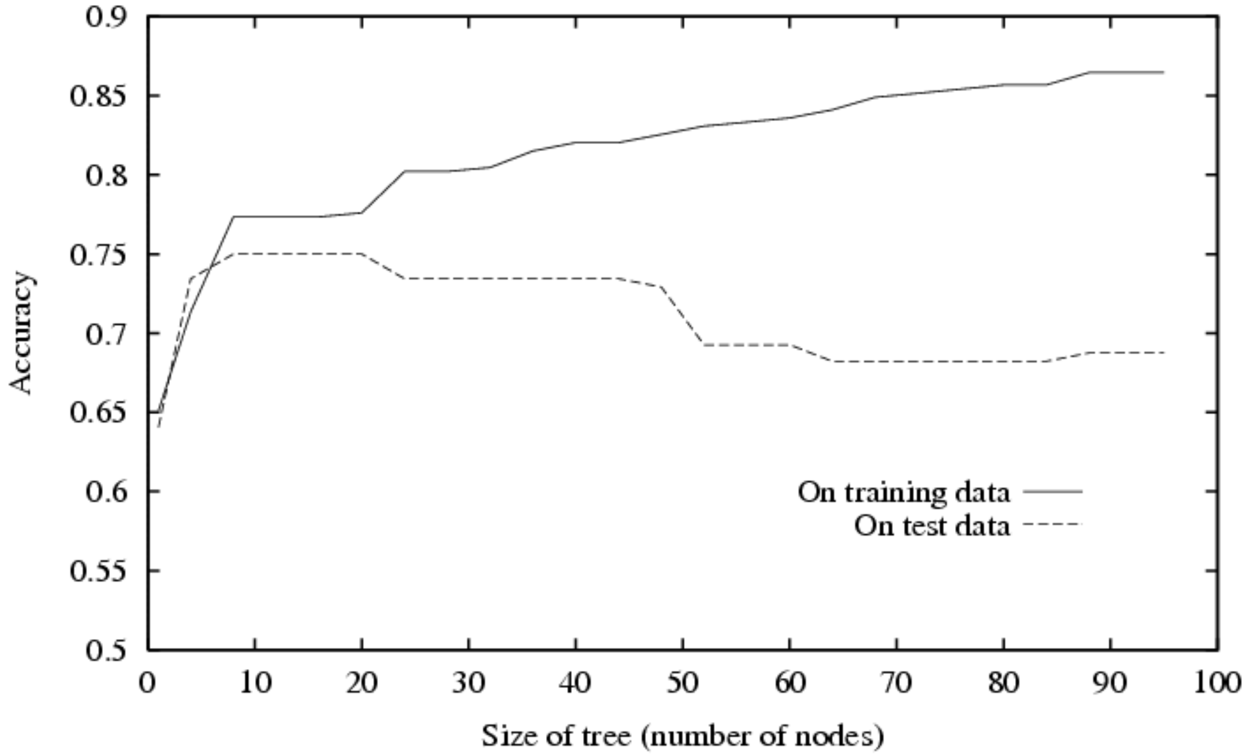
We say $h$ <u>overfits</u> the training data if

$$error_{true}(h) > error_{train}(h)$$

Amount of overfitting =

$$error_{true}(h) - error_{train}(h)$$

# 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

# Reduced-Error Pruning

Split data into *training* and *validation* set

Create tree that classifies *training* set correctly

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

- produces smallest version of most accurate subtree
- What if data is limited?

# Continuous Valued Attributes

Create a discrete attribute to test continuous

- $Temperature = 82.5$
- $(Temperature > 72.3) = t, f$

| | | | | | | |
|---|---|---|---|---|---|---|
| *Temperature*: | 40 | 48 | 60 | 72 | 80 | 90 |
| *PlayTennis*: | No | No | Yes | Yes | Yes | No |