

# Decision Tree Learning

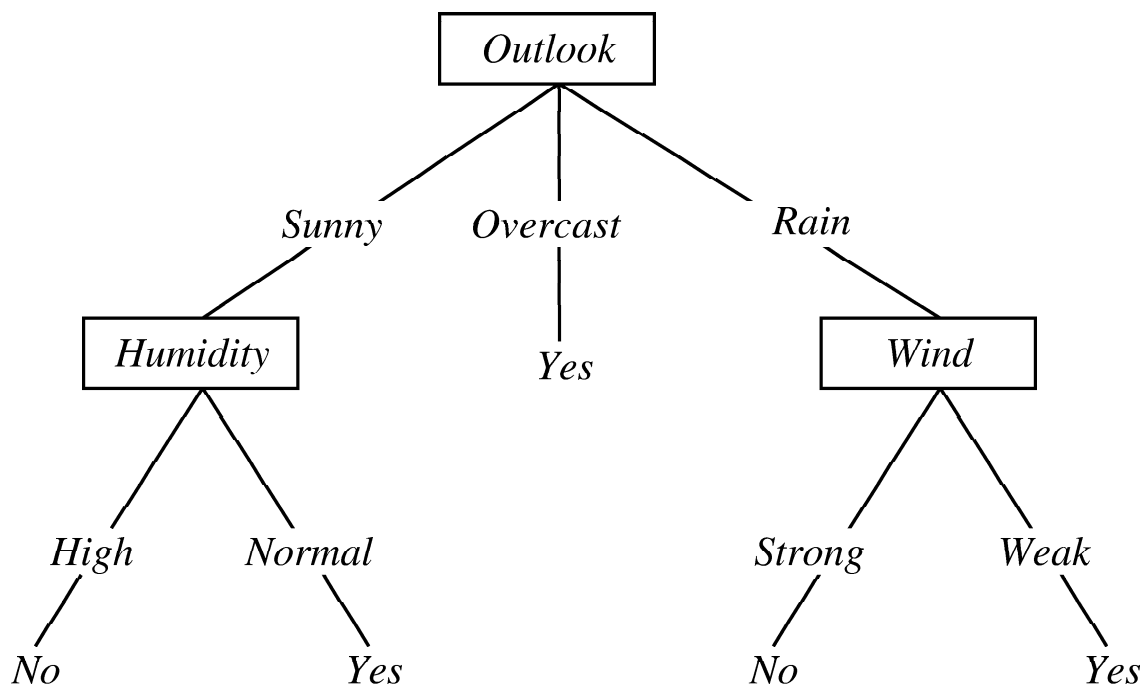
---

[read Chapter 3]  
[recommended exercises 3.1, 3.4]

- Decision tree representation
- ID3 learning algorithm
- Entropy, Information gain
- Overfitting

# Decision Tree for *PlayTennis*

---



# 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+ .05-
| | | | Birth_Weight >= 3349: [133+,36.4-] .78+ .22-
| | | 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-
```

# Decision Trees

---

Decision tree representation:

- Each internal node tests an attribute
- Each branch corresponds to attribute value
- Each leaf node assigns a classification

How would we represent:

- $\wedge, \vee, \text{XOR}$
- $(A \wedge B) \vee (C \wedge \neg D \wedge E)$
- $M$  of  $N$

# When to Consider Decision Trees

---

- Instances describable by attribute–value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data

Examples:

- Equipment or medical diagnosis
- Credit risk analysis
- Modeling calendar scheduling preferences

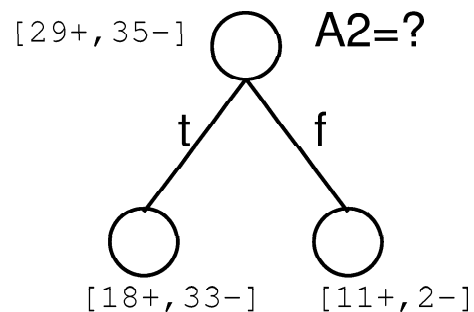
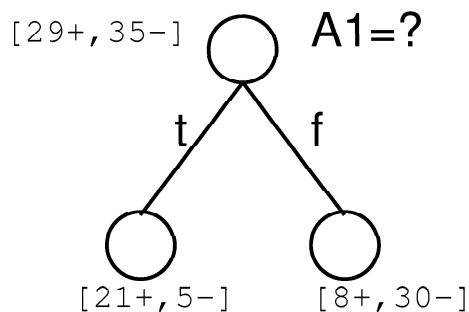
# Top-Down Induction of Decision Trees

---

Main loop:

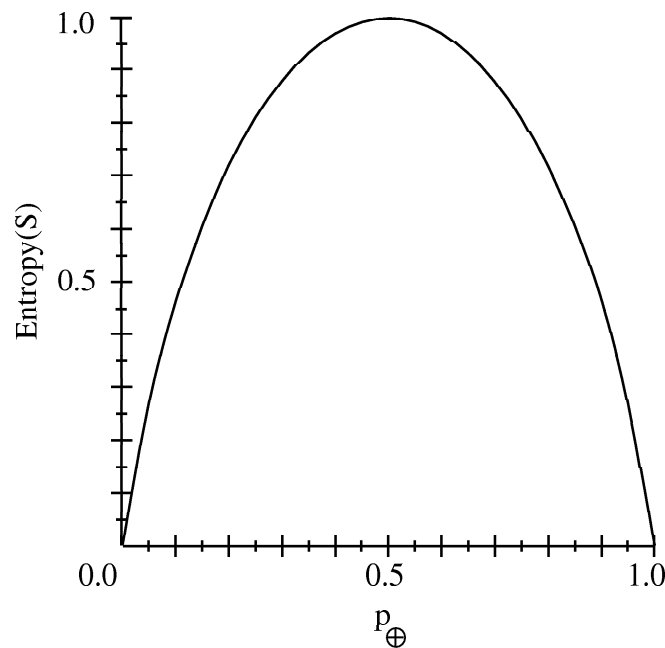
1.  $A \leftarrow$  the “best” decision attribute for next *node*
2. Assign  $A$  as decision attribute for *node*
3. For each value of  $A$ , create new descendant of *node*
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

Which attribute is best?



# Entropy

---



- $S$  is a sample of training examples
- $p_+$  is the proportion of positive examples in  $S$
- $p_-$  is the proportion of negative examples in  $S$
- Entropy measures the impurity of  $S$

$$Entropy(S) \equiv -p_+ \log_2 p_+ - p_- \log_2 p_-$$

# Entropy

---

$Entropy(S)$  = expected number of bits needed to encode class ( $\oplus$  or  $\ominus$ ) of randomly drawn member of  $S$  (under the optimal, shortest-length code)

Why?

Information theory: optimal length code assigns  $-\log_2 p$  bits to message having probability  $p$ .

So, expected number of bits to encode  $\oplus$  or  $\ominus$  of random member of  $S$ :

$$p_{\oplus}(-\log_2 p_{\oplus}) + p_{\ominus}(-\log_2 p_{\ominus})$$

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

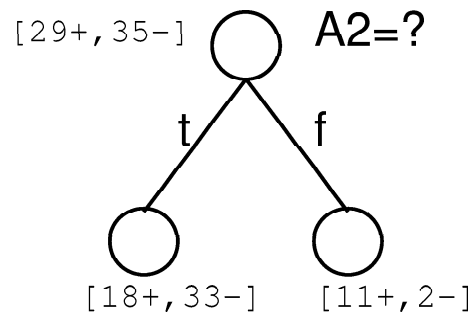
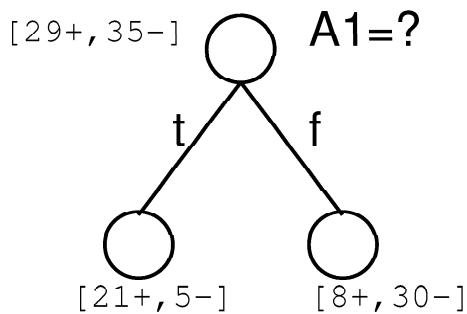


# Information Gain

---

$Gain(S, A) =$  expected reduction in entropy due to sorting on  $A$

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



# Training Examples

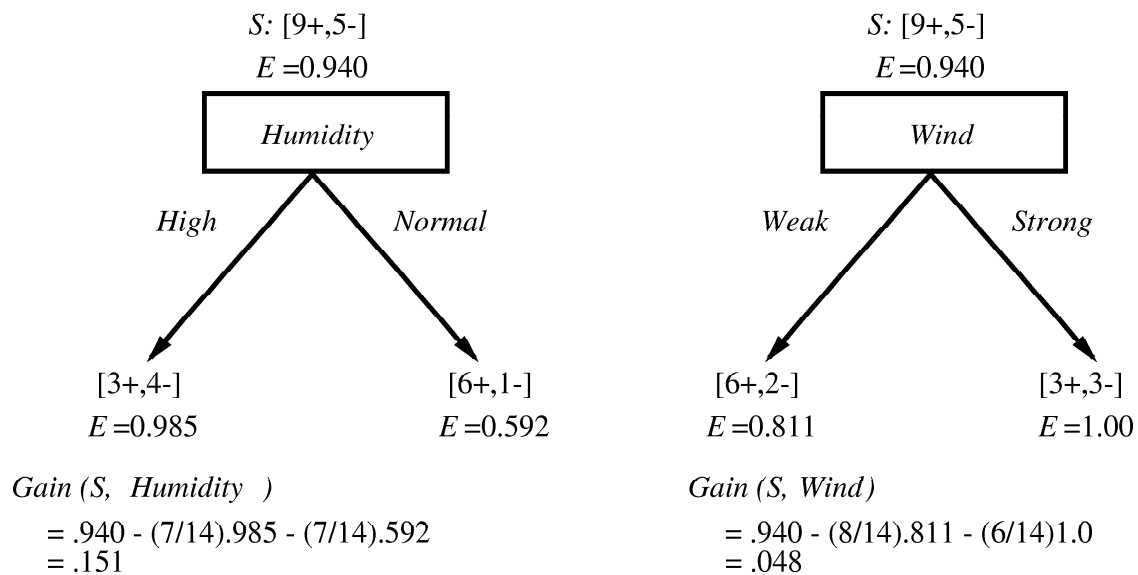
---

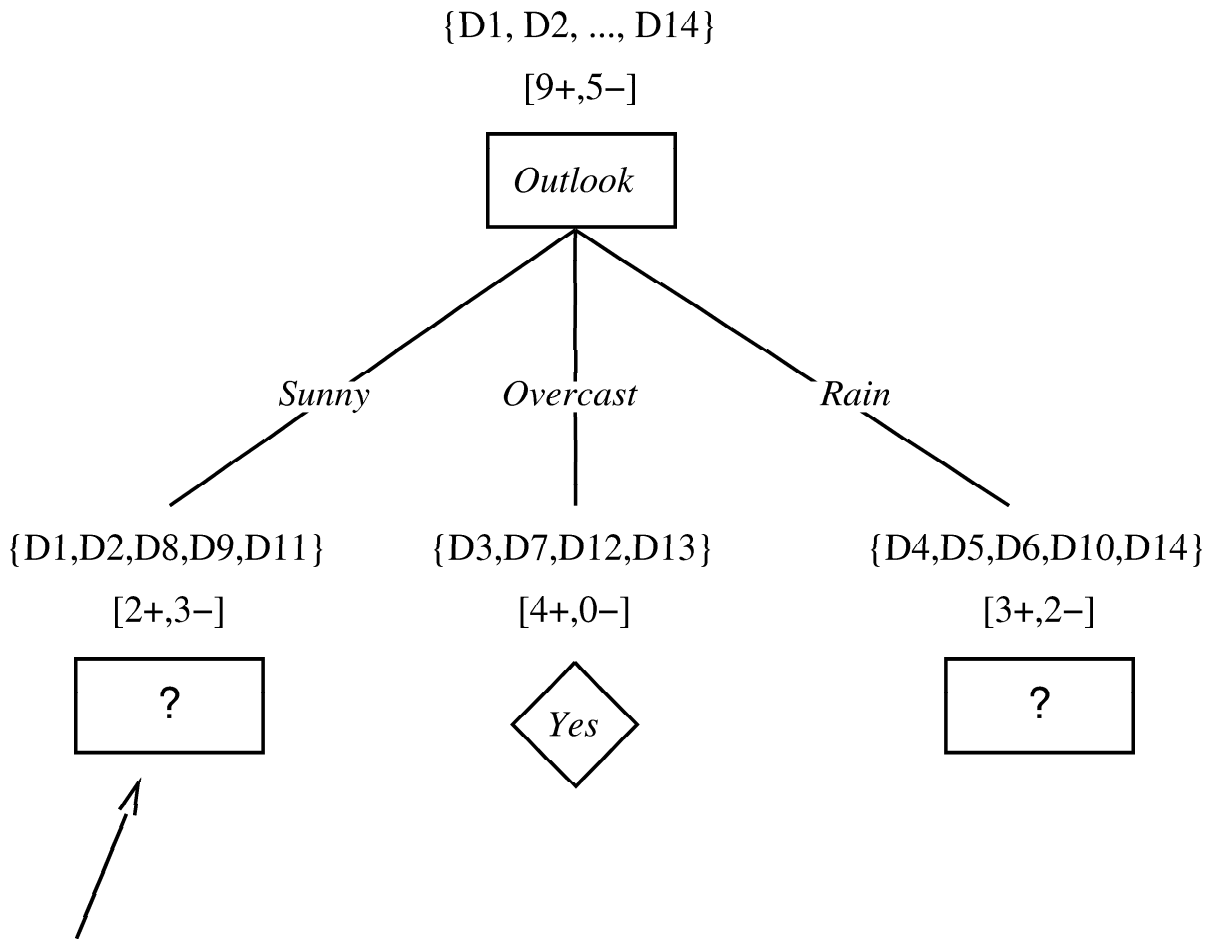
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	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

# Selecting the Next Attribute

---

Which attribute is the best classifier?





*Which attribute should be tested here?*

$$S_{sunny} = \{D1,D2,D8,D9,D11\}$$

$$Gain(S_{sunny}, Humidity) = .970 - (3/5) 0.0 - (2/5) 0.0 = .970$$

$$Gain(S_{sunny}, Temperature) = .970 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = .570$$

$$Gain(S_{sunny}, Wind) = .970 - (2/5) 1.0 - (3/5) .918 = .019$$



# Hypothesis Space Search by ID3

---

- Hypothesis space is complete!
  - Target function surely in there...
- Outputs a single hypothesis (which one?)
  - Can't play 20 questions...
- No back tracking
  - Local minima...
- Statically-based search choices
  - Robust to noisy data...
- Inductive bias: approx “prefer shortest tree”

# Inductive Bias in ID3

---

Note  $H$  is the power set of instances  $X$

→ Unbiased?

Not really...

- Preference for short trees, and for those with high information gain attributes near the root
- Bias is a *preference* for some hypotheses, rather than a *restriction* of hypothesis space  $H$
- Occam's razor: prefer the shortest hypothesis that fits the data

# Occam's Razor

---

Why prefer short hypotheses?

Argument in favor:

- Fewer short hyps. than long hyps.

→ a short hyp that fits data unlikely to be coincidence

→ a long hyp that fits data might be coincidence

Argument opposed:

- There are many ways to define small sets of hyps
- e.g., all trees with a prime number of nodes that use attributes beginning with "Z"
- What's so special about small sets based on *size* of hypothesis??



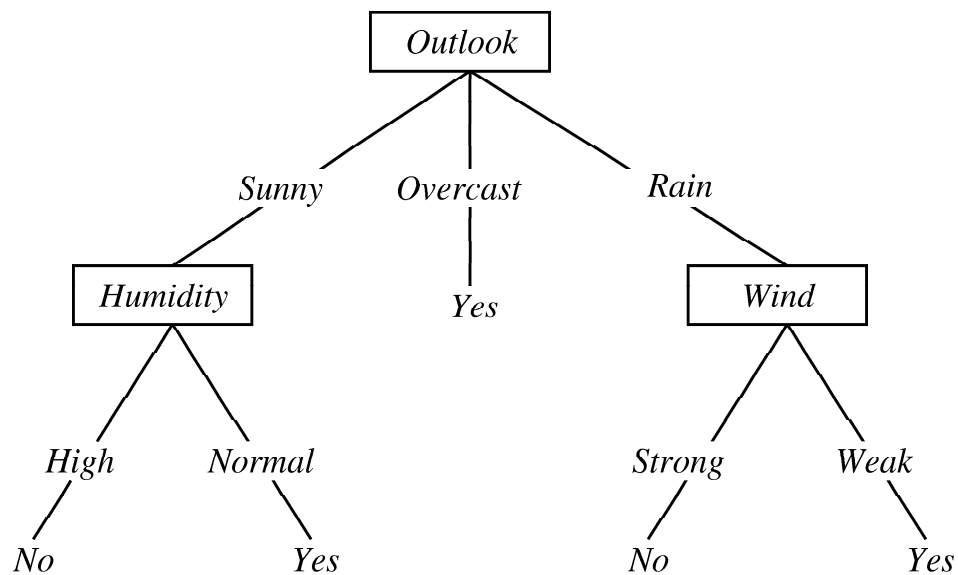
# Overfitting in Decision Trees

---

Consider adding noisy training example #15:

*Sunny, Hot, Normal, Strong, PlayTennis = No*

What effect on earlier 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

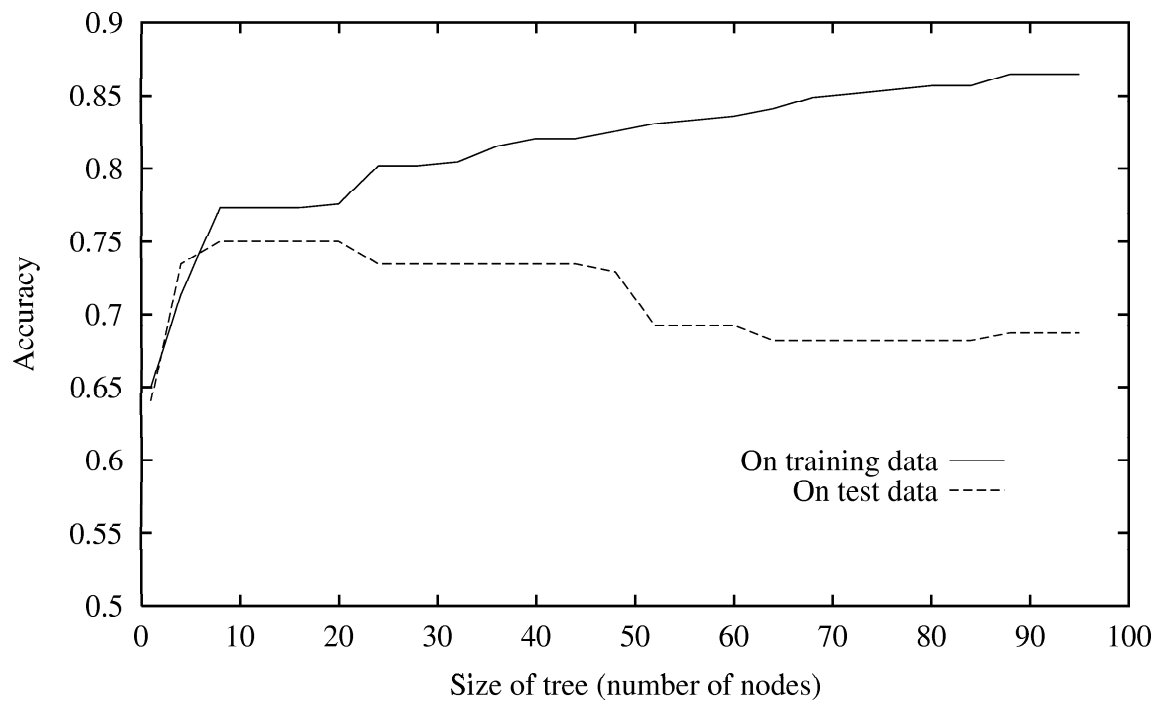
$$error_{train}(h) < error_{train}(h')$$

and

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

# Overfitting 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
- MDL: minimize  
 $size(tree) + size(misclassifications(tree))$

# Reduced-Error Pruning

---

Split data into *training* and *validation* set

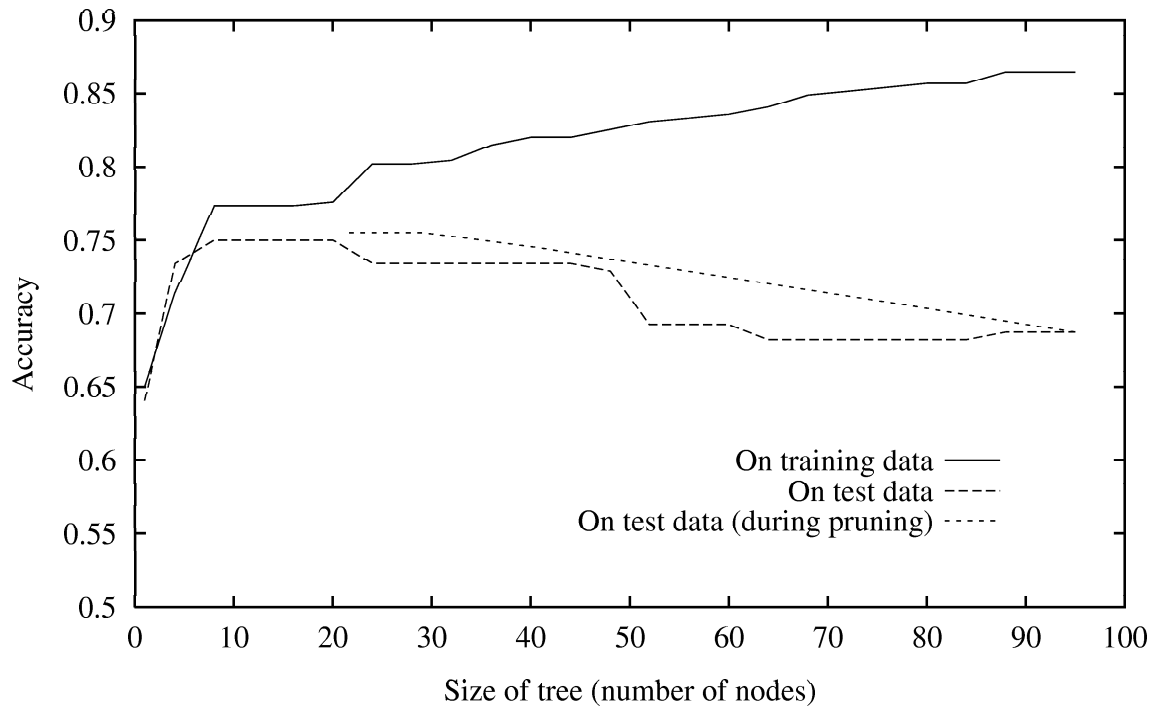
Do until further pruning is harmful:

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

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

# Effect of Reduced-Error Pruning

---



# Rule Post-Pruning

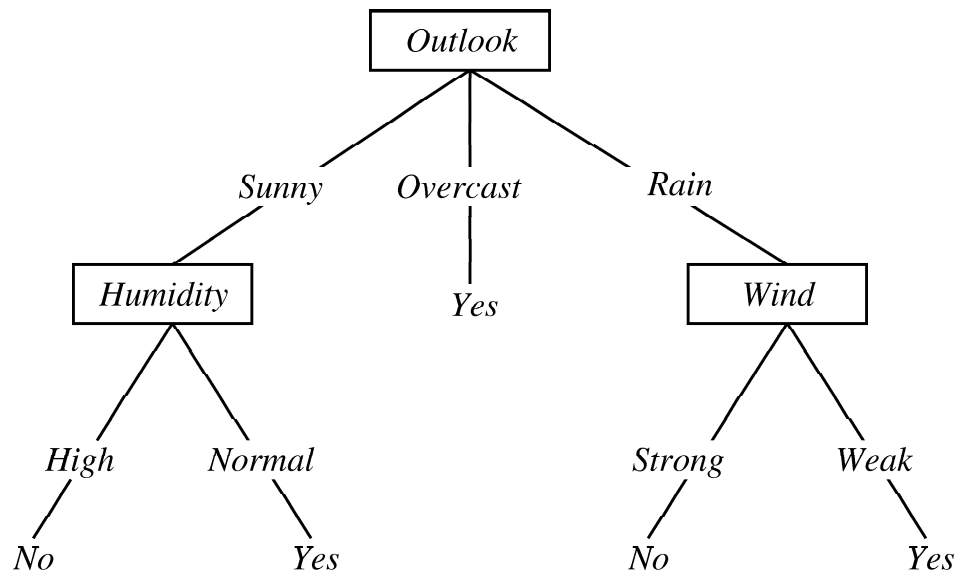
---

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

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

...

# Continuous Valued Attributes

---

Create a discrete attribute to test continuous

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

<i>Temperature:</i>	40	48	60	72	80	90
<i>PlayTennis:</i>	No	No	Yes	Yes	Yes	No

# 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

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

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

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

# Attributes with Costs

---

Consider

- medical diagnosis, *BloodTest* has cost \$150
- robotics, *Width\_from\_1ft* has cost 23 sec.

How to learn a consistent tree with low expected cost?

One approach: replace gain by

- Tan and Schlimmer (1990)

$$\frac{Gain^2(S, A)}{Cost(A)}$$

- Nunez (1988)

$$\frac{2^{Gain(S,A)} - 1}{(Cost(A) + 1)^w}$$

where  $w \in [0, 1]$  determines importance of cost

# Unknown Attribute Values

---

What if some examples 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