CSCE 478/878 Lecture 3: Learning Decision Trees

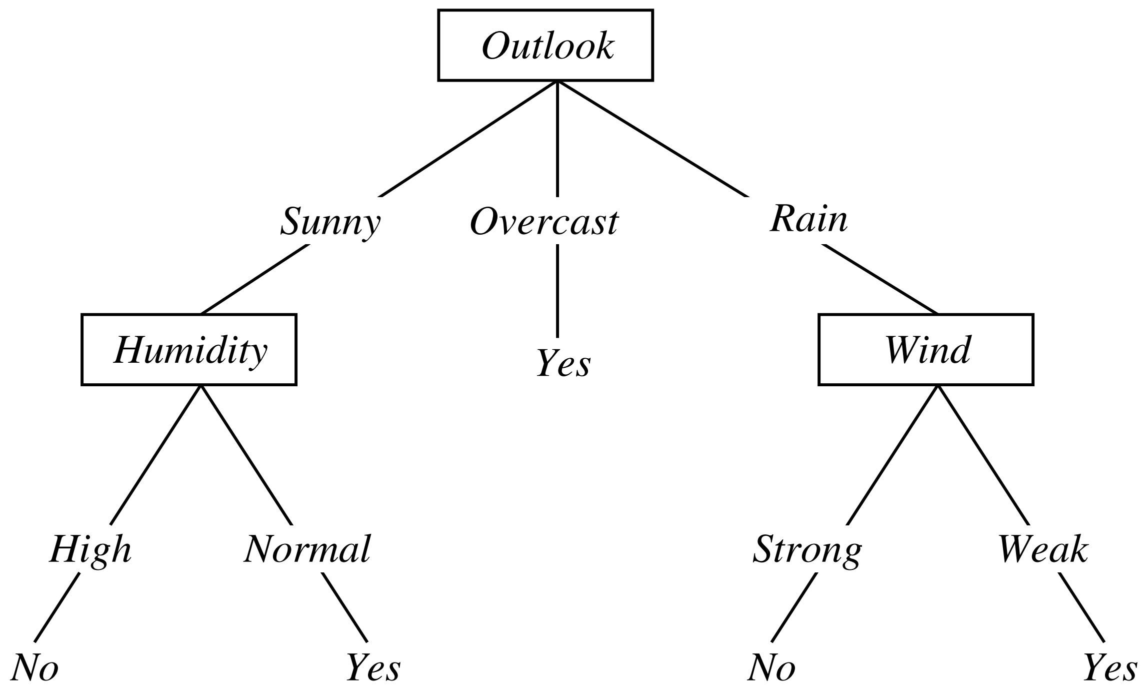
Stephen D. Scott
(Adapted from Tom Mitchell's slides)

September 10, 2008

Outline

- Decision tree representation
- ID3 learning algorithm
- Entropy, Information gain
- Overfitting and pruning
- Continuous, many-valued, unknown, and cost-associated attributes

Decision Tree for $PlayTennis$



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:

- $\wedge, \vee$, XOR
- $(A \wedge B) \vee (C \wedge \neg D \wedge E)$

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
- Human readability of result is important

Examples:

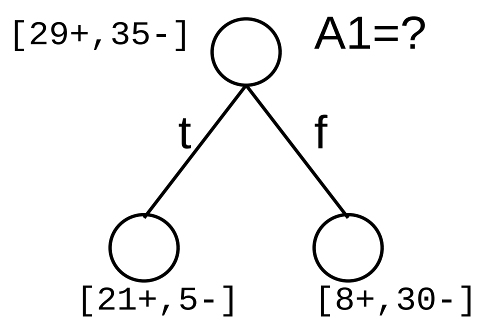
- Equipment or medical diagnosis
- Credit risk analysis
- Modeling calendar scheduling preferences

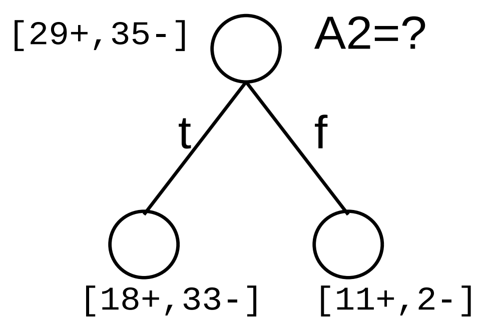
Top-Down Induction of Decision Trees
(ID3 Algorithm, Table 3.1)

Main loop:

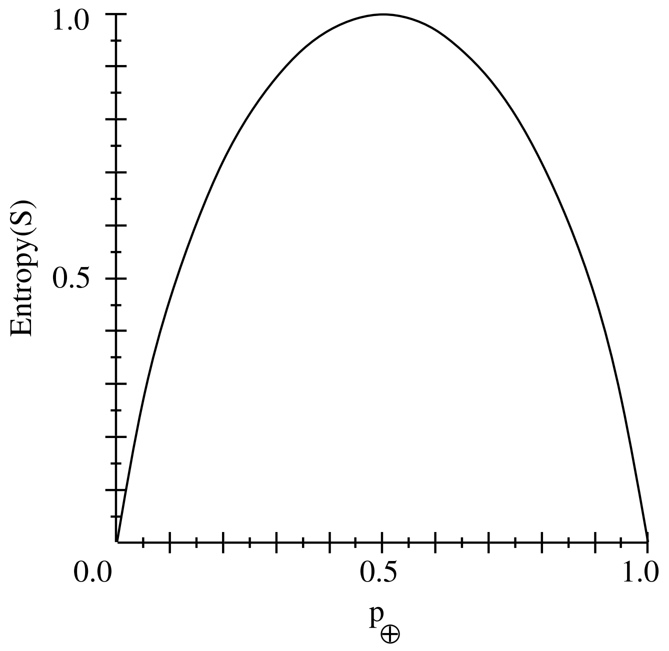
1. $A \leftarrow$ the "best" decision attribute for next $node$
2. Assign A as decision attribute for $node$
3. For each value of A, create new descendant of $node$
4. Sort (partition) training examples over children based on A's value
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

Which attribute is best?





Entropy



- S is a sample of training examples

- $p_\oplus$ is the proportion of positive examples in S

- $p_\ominus$ is the proportion of negative examples in S

- Entropy measures the impurity of S

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

Entropy
(cont'd)

- $Entropy(S)$ = 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:

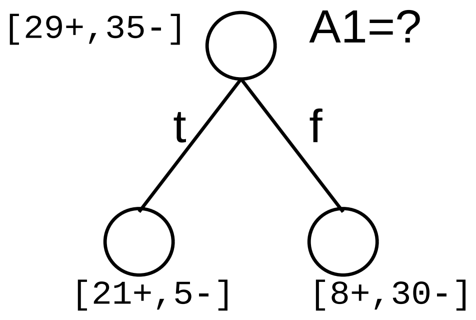
$$p_{\oplus}(-\log_2 p_{\oplus}) + p_{\ominus}(-\log_2 p_{\ominus})$$

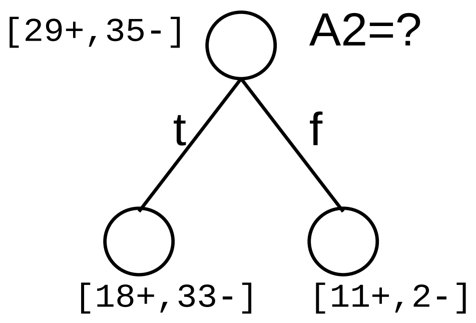
$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

Information Gain

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

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



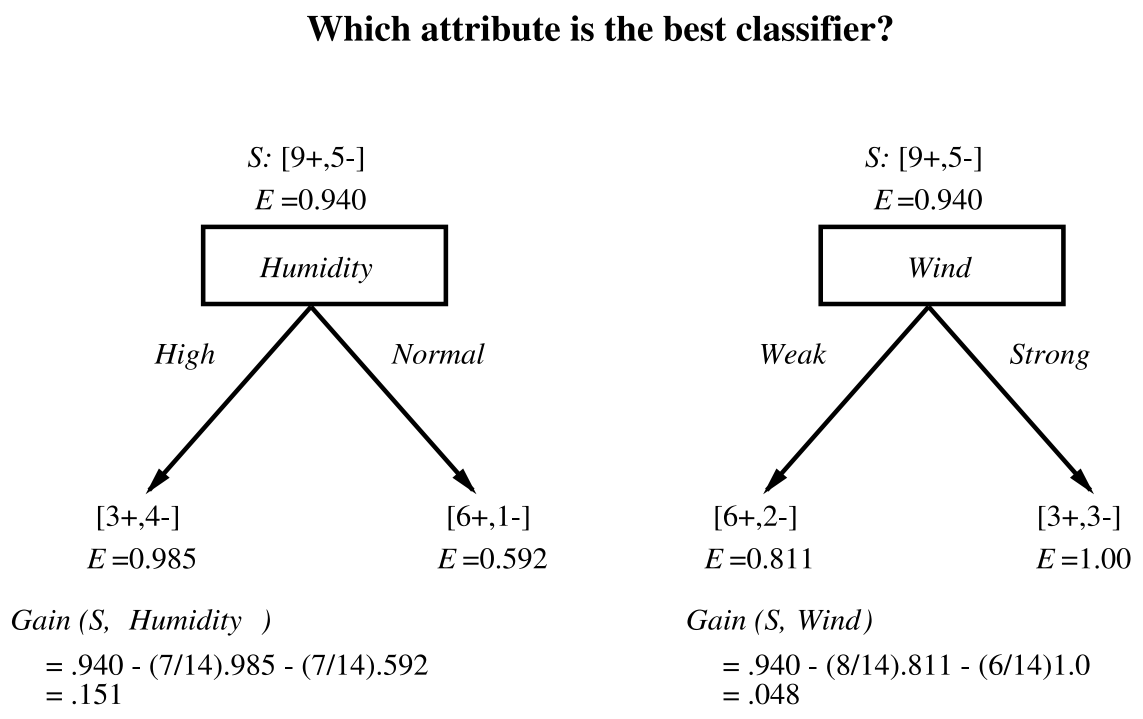


Training Examples

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

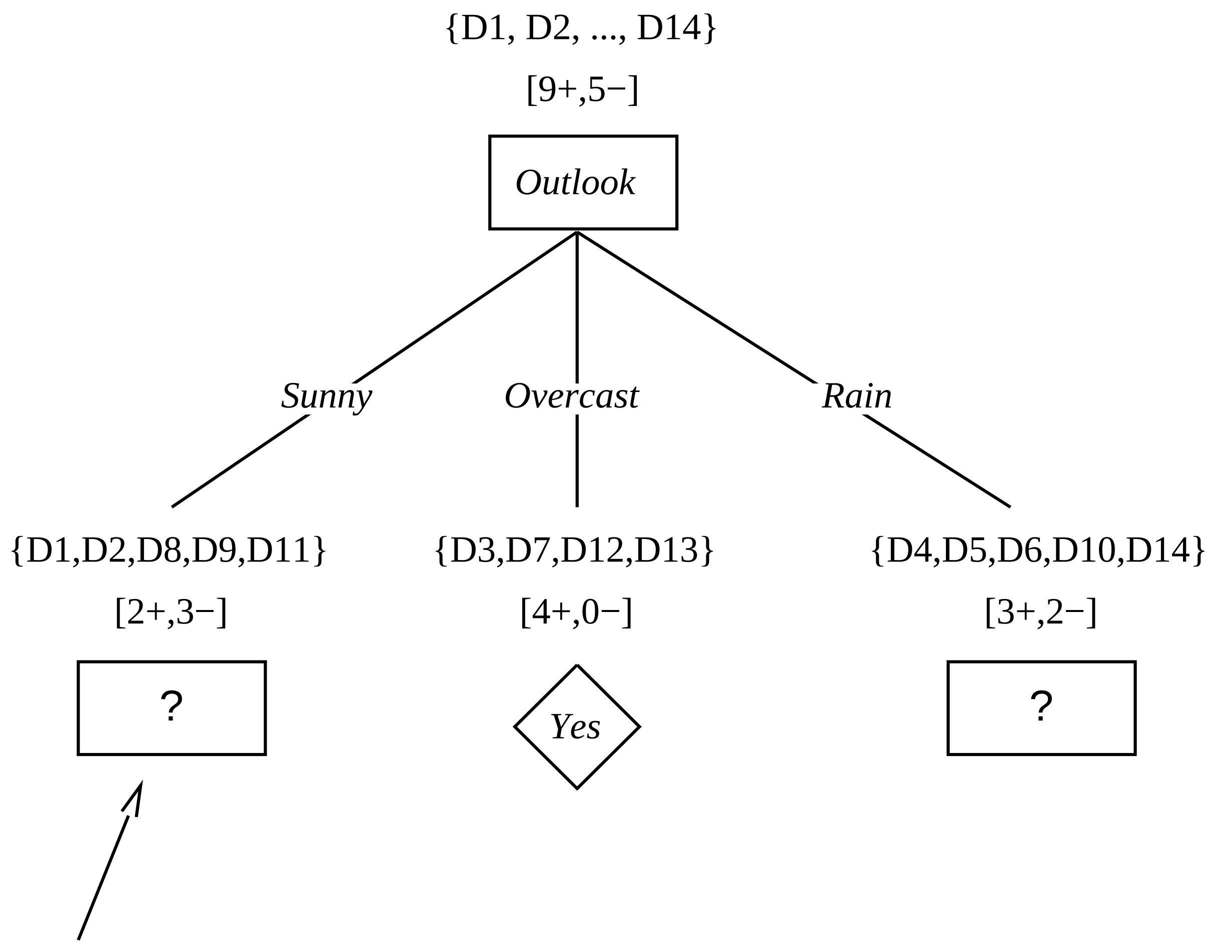
Selecting the First Attribute

- Comparing $Humidity$ to $Wind$:



- Other gain values: $Gain(S, Outlook) = 0.246$, $Gain(S, Temperature) = 0.029$

Selecting the Next Attribute



Which attribute should be tested here?

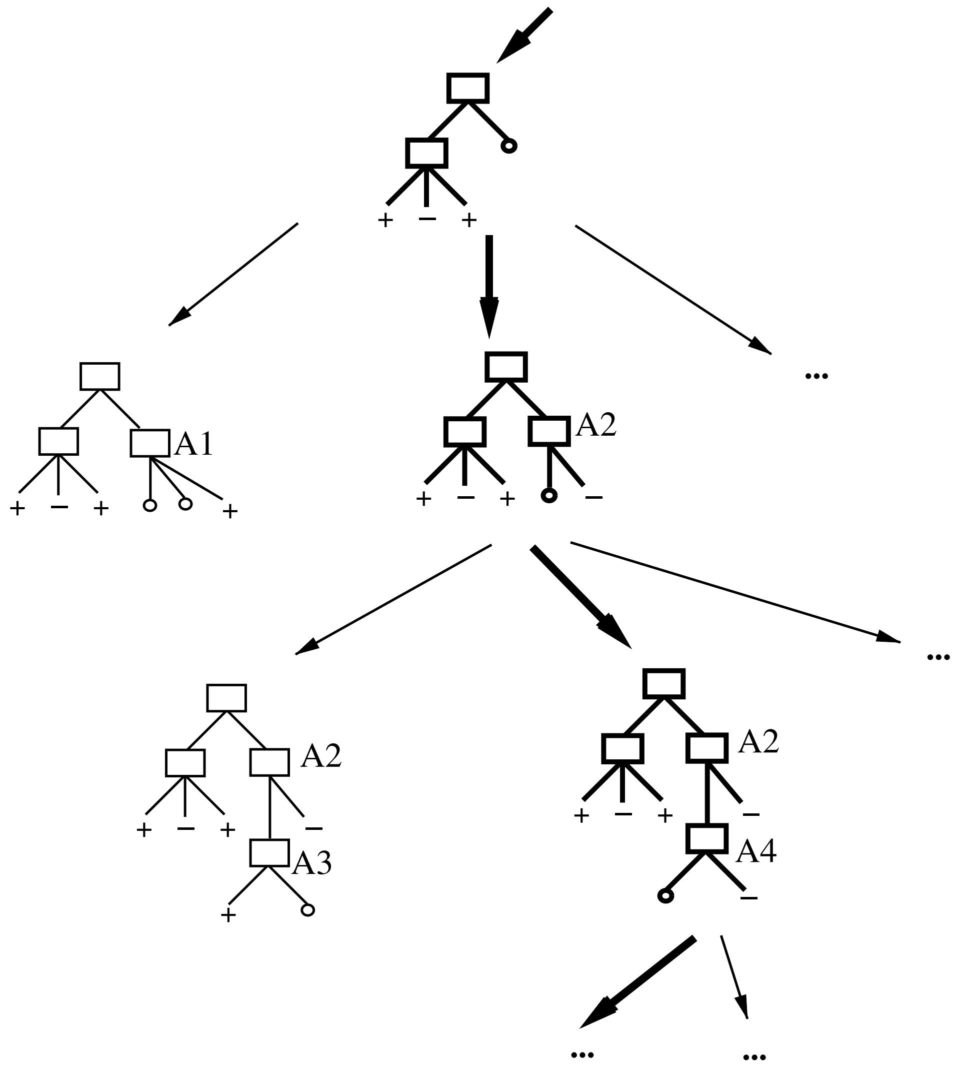
S_{sunny} = {D1,D2,D8,D9,D11}

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

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

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

Hypothesis Space Search by ID3



Hypothesis Space Search by ID3
(cont'd)

- Hypothesis space is complete!
  - Target function surely in there...
- Maintains a single hypothesis versus a representation of the version space
  - Can't use queries in this algorithm to reduce the VS
- No back tracking, pure hill climbing (maximizing inf. gain)
  - Problems with local optima
- Statisically-based search choices
  - Robust to noisy data (can terminate before perfectly fitting training data)
- Inductive bias $\approx$ "prefer shortest tree"

Inductive Bias in ID3

- Note H is the power set of instances X

$\Rightarrow$ 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 (like with candidate elim.)
    - Checkers player of Chapter 1 had both
  - 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:

- 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??

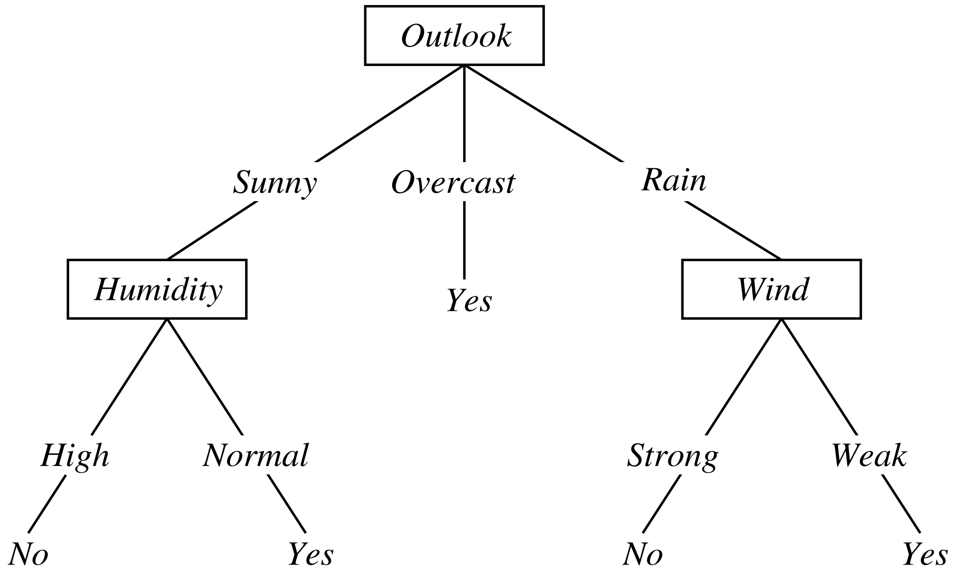
Occam's razor reappears in MDL (Chapt. 6) and in learning theory (not discussed)

Overfitting in Decision Trees

- Consider adding noisy training example #15:

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

- What effect on earlier tree?



- Expect old tree to generalize better since new one fits noisy example

Overfitting

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

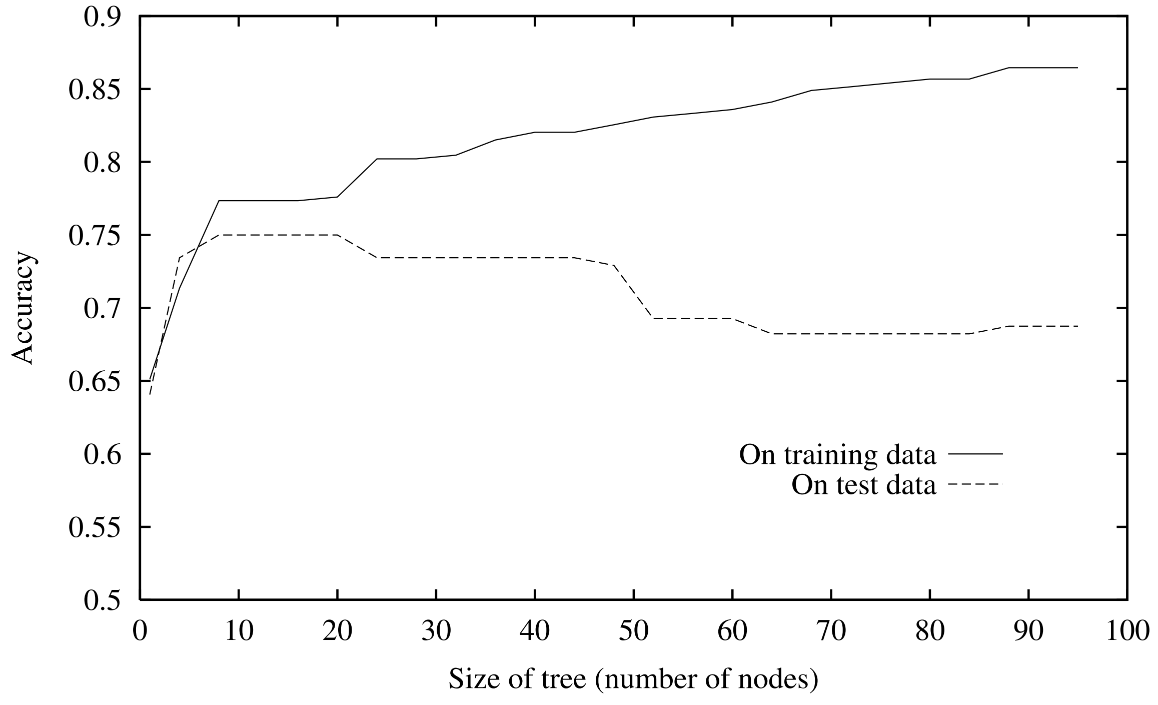
- Hypothesis $h \in H$ overfits training data if there is an alternative hypothesis $h' \in H$ such that

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

and

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

Overfitting in Decision Tree Learning



Avoiding Overfitting

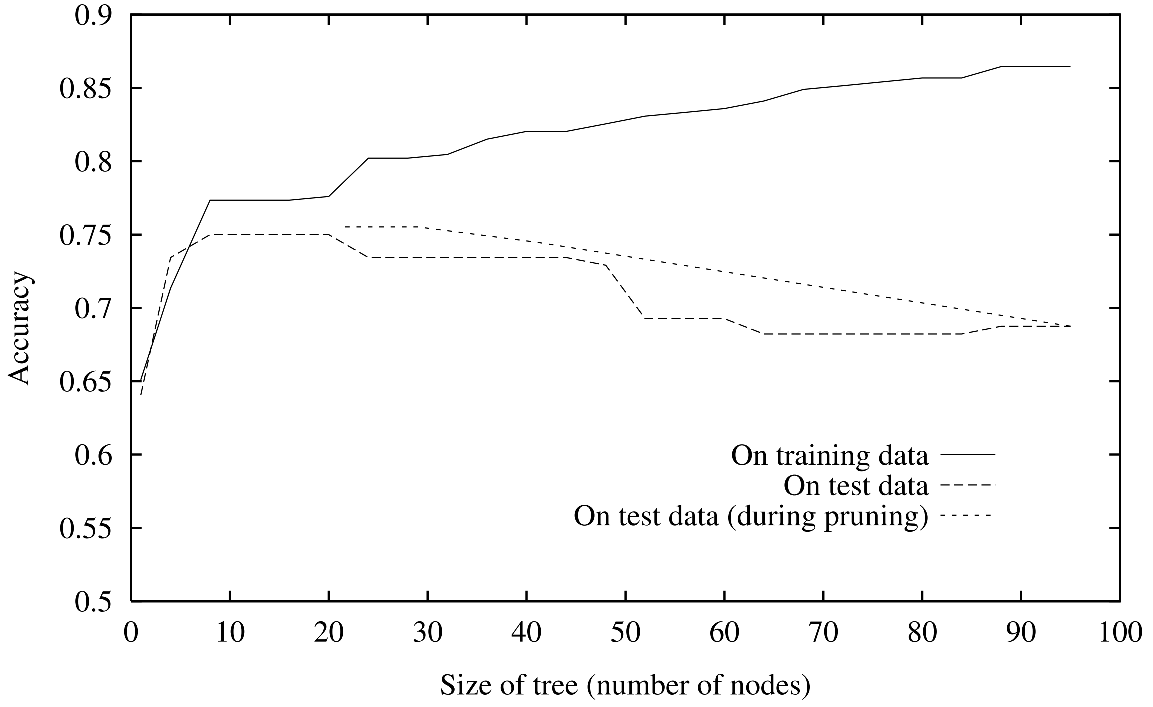
- How can we avoid overfitting?
  - Stop growing when data split doesn't help
  - Grow full tree, then post-prune

- How to select "best" tree:
  - Measure performance over training data
  - Measure performance over separate validation data set
  - MDL (minimum description length principle): minimize $size(tree) + size(misclassifications(tree))$ based on some size measure of trees and examples (Chapt. 6)

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 (with respect to validation set)
- What if data is limited?

Effect of Reduced-Error Pruning

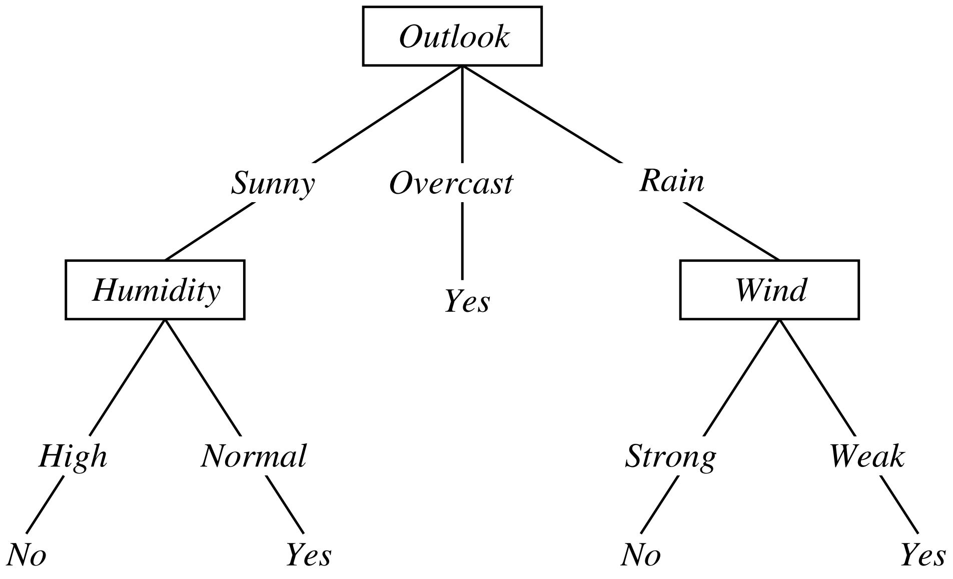


Rule Post-Pruning

1. Convert tree to equivalent set of rules
2. Prune each rule independently of others by removing selected preconditions (the ones that improve accuracy the most)
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

Use threshold to map continuous to boolean, e.g.
$(Temperature > 72.3) \in \{t, f\}$

Temperature:	40	48	60	72	80	90
PlayTennis:	No	No	Yes	Yes	Yes	No

- Can show that threshold maximizing inf. gain must lie between two adjacent attribute values in training set such that label changed, so try all such values, e.g. $(48 + 60)/2 = 54$ and $(80 + 90)/2 = 85$

- Now (dynamically) replace continuous attribute with boolean attributes $Temperature_{>54}$ and $Temperature_{>85}$ and run algorithm normally

- Other options: Split into multiple intervals rather than two; use thresholded linear combinations of continuous attributes

Attributes with Many Values

Problem:

- If attribute has many values, $Gain$ will select it
- E.g. if $Date$ is attribute, inf. gain will be high because several very small subsets will be created

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 (measures how broadly and uniformly A splits data)

Attributes with Costs

- 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 are missing values of A?

Use them anyway (sift each through tree)

- If node n tests A, assign most common value of A among other training examples sifted to node n
- Assign most common value of A among other examples with same target value (either overall or at node n)
- 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

Topic summary due in 1 week!