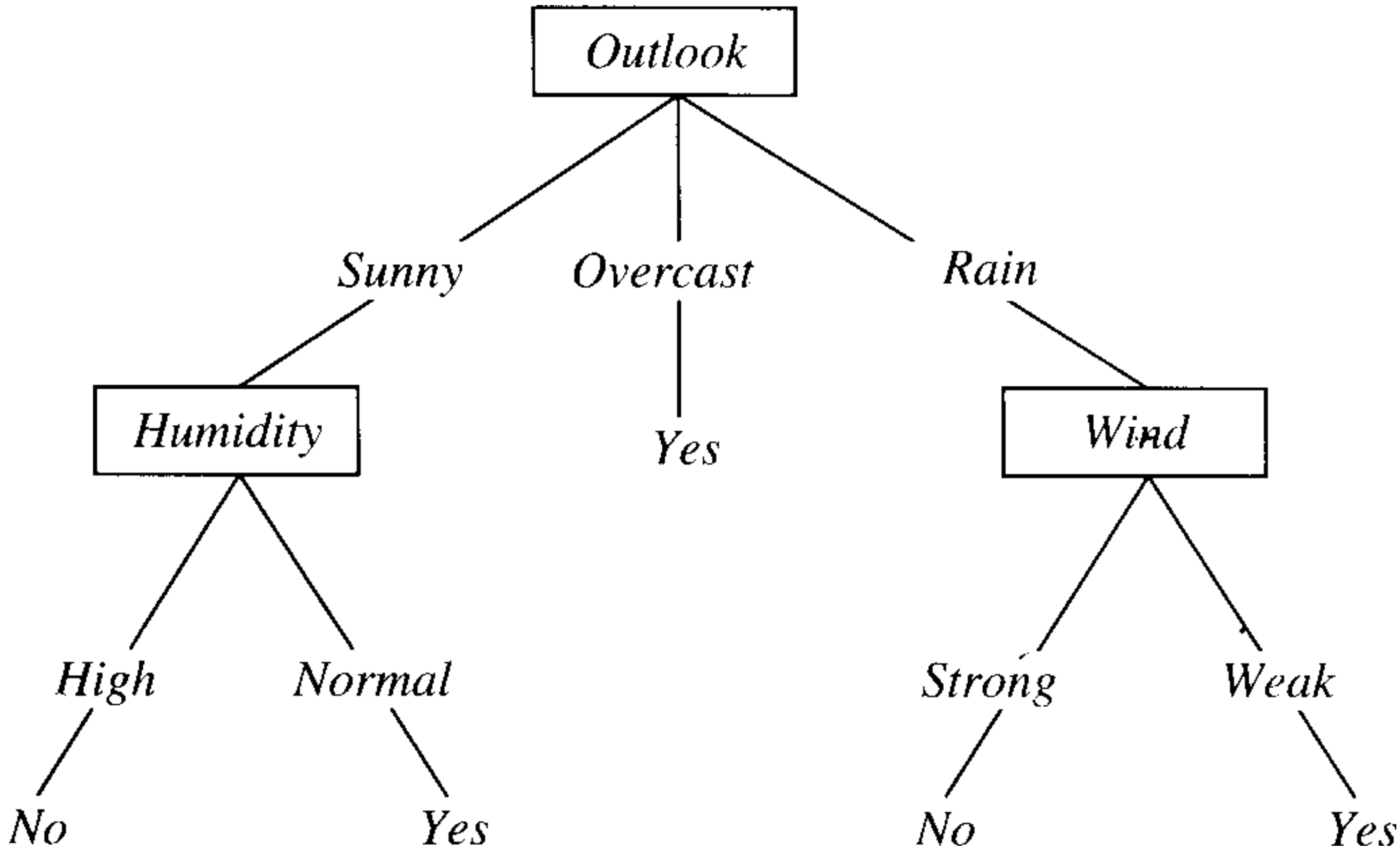
Decision Tree Learning

Patricia J Riddle
Computer Science 367

Decision Tree Learning

- Discrete valued target functions - Classification problems
- Represented as sets of if-then rules to improve human readability
- Used in many success stories
- Classify instances by sorting them down the tree
  – Each internal node is a test on some attribute
  – Each branch is one possible value for that test
  – Each leaf specifies classification value

Decision tree



Learned Rules

- Outlook=Sunny$\wedge$Humidity=High$\rightarrow$PlayTennis=No
- Outlook=Sunny$\wedge$Humidity=Normal$\rightarrow$PlayTennis=Yes
- Outlook=Overcast$\rightarrow$PlayTennis=Yes
- Outlook=Rain$\wedge$Wind=Strong$\rightarrow$PlayTennis=No
- Outlook=Rain$\wedge$Wind=Weak$\rightarrow$PlayTennis=Yes

When to use Decision Tree Learning

- Instances are represented by attribute value pairs (can be real valued).
- The target value has discrete output values (no need to be binary, some extensions even handle real valued targets).
- Disjunctive descriptions may be required
- The training data
  - may contain errors - errors in classification and errors in attribute values
  - may contain missing attribute values

Attribute Values

attributes	Outlook	Wind	Class
Instance 1	rainy	strong	red
Instance 2	sunny	normal	red
Instance 3	sunny	normal	green
Instance 4	cloudy	strong	red
Instance 5	rainy	normal	green

ID3 Algorithm

ID3($Examples$, $Target\_attribute$, $Attributes$)

Examples are the training examples. Target_attribute is the attribute whose value is to be predicted by the tree. Attributes is a list of other attributes that may be tested by the learned decision tree. Returns a decision tree that correctly classifies the given Examples.

- Create a *Root* node for the tree
- If all *Examples* are positive, Return the single-node tree *Root*, with label = +
- If all *Examples* are negative, Return the single-node tree *Root*, with label = −
- If *Attributes* is empty, Return the single-node tree *Root*, with label = most common value of *Target_attribute* in *Examples*
- Otherwise Begin
    - $A \leftarrow$ the attribute from *Attributes* that best* classifies *Examples*
    - The decision attribute for $Root \leftarrow A$
    - For each possible value, v_i, of A,
        - Add a new tree branch below *Root*, corresponding to the test $A = v_i$
        - Let $Examples_{v_i}$ be the subset of *Examples* that have value v_i for A
        - If $Examples_{v_i}$ is empty
            - Then below this new branch add a leaf node with label = most common value of *Target_attribute* in *Examples*
            - Else below this new branch add the subtree ID3($Examples_{v_i}$, $Target\_attribute$, $Attributes - \{A\}$))
- End
- Return *Root*

* The best attribute is the one with highest *information gain*, as defined in Equation (3.4).

What Attribute is the Best Classifier?

- Entropy (from information theory)
  - Measures the impurity of an arbitrary collection of examples
- $\text{Entropy}(S) \equiv -p_{\oplus}\log_2 p_{\oplus} - p_{\ominus}\log_2 p_{\ominus}$
  - for a boolean classification where $p_{\oplus}$ is the proportion of positive examples in S and $p_{\ominus}$ is the proportion of negative examples in S.
  - In all calculations involving entropy we define 0log0 to be 0

Entropy

- $Entropy(9+,5-)=-(9/14)\log_2(9/14)-(5/14)\log_2(5/14)=.94$
  - If all members of S are in the same class Entropy(S)=0
  - If there is an equal number of positive and negative instances in S then Entropy(S)=1

- Entropy specifies the minimum number of bits of information needed to encode the classification of an arbitrary member of S

Information Theory

- A fair coin has an entropy of one bit.
- However, if the coin is not fair, then the uncertainty is lower (if asked to bet on the next outcome, we would bet preferentially on the most frequent result), and thus the Shannon entropy is lower.
- A long string of repeating characters has an entropy rate of 0, since every character is predictable.

Information Theory History

- During World War II, Claude Shannon developed a model of the communication process using the earlier work of Nyquist and Hartley.
- Shannon thought that the fundamental problem of communication is that of reproducing at one point either exactly or approximately a message selected at another point.

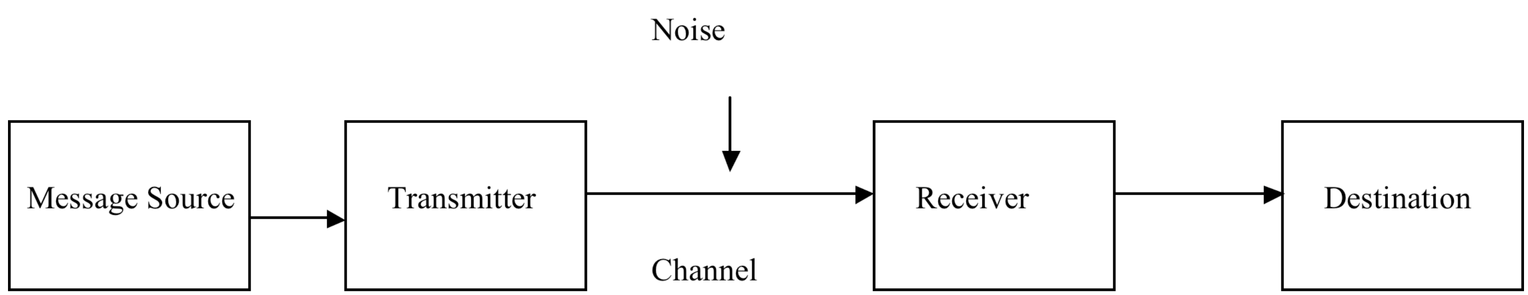
The Gist of Information Theory

- The significant aspect is that the actual message is one *selected from a set* of possible messages.
- Using this engineering perspective, the communication process may be understood as a *source* communicating to a *destination*.
- The source provides its message to a *transmitter* through a perfect connection.
- The transmitter communicates through a channel to the *receiver*, which receives the message and gives it in a lossless manner to the destination.

Importance of Noise

- One of the key additions that Shannon made to the earlier work of Nyquist and Hartley was the formal integration of noise into the communication model.
- Noise is introduced into the channel between the transmitter and the receiver and acts to changes messages so that what is received differs from what is transmitted.

Shannon's Channel Model



Coding

- Coding takes place at the transmitter not at the source of the message
- The coded form of the message is what leaves the transmitting process and moves to the receiving process.
- It is represented in some form that can be transmitted by the medium supporting the channel.
- Transmitting data inherently requires that a change of medium take place
- When a signal moves from one medium to another, it must be physically represented somewhat differently, making an encoder necessary.

How to Code

- Given a source producing symbols at a rate consistent with a set of probabilities governing their frequency of occurrence, Shannon asks ``how much information is `produced' by such a process, or better, at what rate information is produced?"

- For Shannon, the amount of self-information that is contained in or associated with a message being transmitted, when the probability of its transmission is *p*, is the logarithm of the inverse of the probability, or **I=log1/p**

- The choice of a logarithmic base corresponds to the choice of a unit for measuring information.

- If the base 2 is used the resulting units may be called binary digits, or more briefly *bits*, a word suggested by J. W. Tukey.

Deriving Entropy

- A device with two stable positions . . . can store one bit of information.
- *N* such devices can store *N* bits, since the total number of possible states is s^N and $\log_2 2^N = N$.
- The amount of information in the output of a process is proportional to the number of different values that the function might return.
- Given *n* different output values, the amount of information (I) may be computed as $I = \log_2 n$.

Restaurant Example

- Ordering food at a restaurant might be modeled as a channel based process.
- The thoughts concerning food preference might be seen as the source, the vocalized order comes from the transmitting mouth, the waiter's ear is the receiver, and the chef is the destination.
- For example, use of this model may suggest that noise effecting the channel might be examined.
- Using care in the choice of codes (names for food) might help decrease the error rate in recording customer orders.
- Also things ordered more often should have shorter names

English example

- People have a tendency to talk, and presumably think, at the basic level of categorization
  - to draw the boundary around "chairs", rather than around the more specific category "recliner", or the more general category "furniture".

- People are more likely to say "You can sit in that chair" than "You can sit in that recliner" or "You can sit in that furniture".

- And it is no coincidence that the word for "chair" contains fewer syllables than either "recliner" or "furniture”.

In Summary

- Basic-level categories, in general, tend to have short names; and nouns with short names tend to refer to basic-level categories.
- Not a perfect rule, of course, but a definite tendency.
- Frequent use goes along with short words; short words go along with frequent use.
- Or as Douglas Hofstadter put it, there's a reason why the English language uses "the" to mean "the" and "antidisestablishmentarianism" to mean "antidisestablishmentarianism" instead of antidisestablishmentarianism other way around.

What does this have to do with ML

- Machine Learning is the same as compression
- Now you just have to transmit the tree and the mistakes or errors
- A lot of compression algorithms are machine learning algorithms and vice versa
- Information Theory is the basis of them both

General Entropy Formula

- Generally, $$Entropy(S) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i$$
  - For example if there are 4 classes and the set is split evenly, 2 bits will be needed to encode the classification of an arbitrary member of S.
  - If it is split less evenly an average message length of less then 2 can be used.

Entropy Function

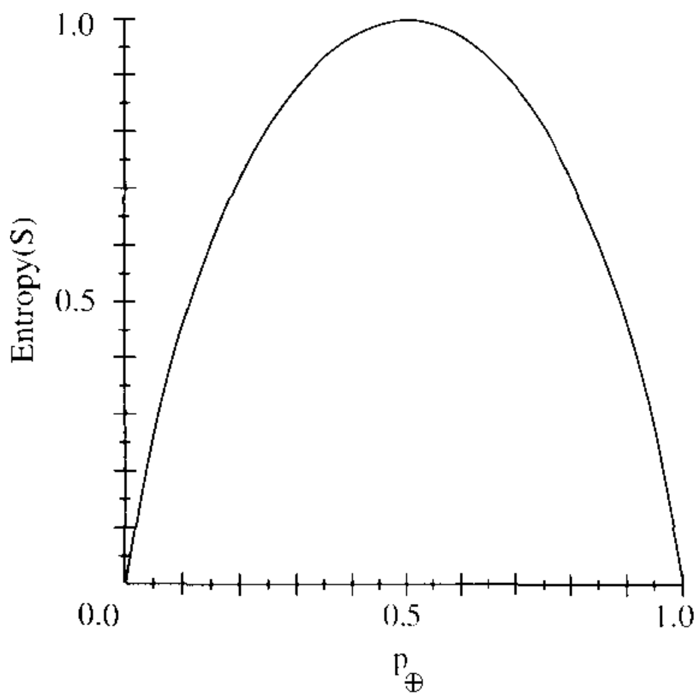


FIGURE 3.2
The entropy function relative to a boolean [illegible] as the proportion, $p_{\oplus}$, of positive [illegible] between 0 and 1.

Information Gain

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

– Where Values(A) is the set of possible values for the attribute A and S_v is the subset of S for which attribute A has value v.

- Information Gain is the expected reduction in entropy caused by knowing the value of attribute A.

Information Gain Intuition

- Information Gain is the information provided about the target function value, given the value of some other attribute A.

- The value of Gain(S,A) is the number of bits saved when encoding the target value of an arbitrary member S, by knowing the value of A.

Information Gain Example

- Of our 14 examples suppose 6 positive and 2 negative have Wind=Weak.
- Values(Wind)=Weak,Strong

$S=[9+,5-]$

$S_{weak} \leftarrow [6+,2-]$

$S_{strong} \leftarrow [3+,3-]$

Information Gain Example II

$$Gain(S,Wind) = Entropy(S) - \sum_{v \in \{weak,strong\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

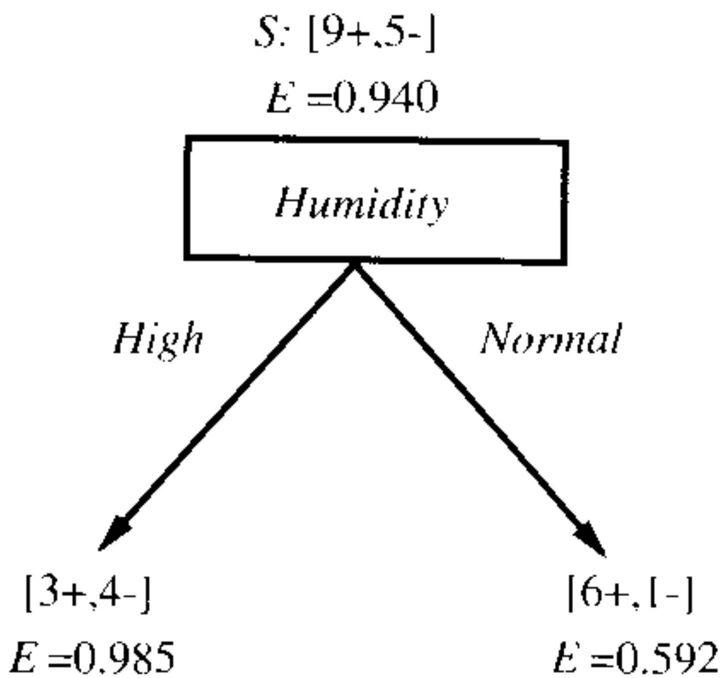
The information gain by sorting the 14 examples by Wind is:

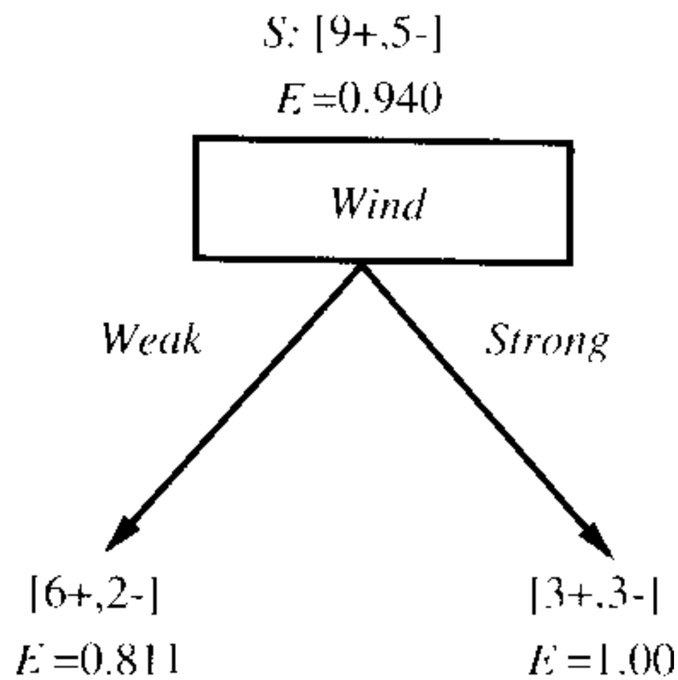
$$Entropy(S) - (8/14)Entropy(S_{Weak}) - (6/14)Entropy(S_{Strong})$$

=0.940-(8/14)0.811-(6/14)1.00

=0.048

Example Continued





Gain (S, Humidity)

= .940 - (7/14).985 - (7/14).592
= .151

Gain (S, Wind)

= .940 - (8/14).811 - (6/14)1.0
= .048

$$Entropy(S) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i$$

$$Gain(S, Wind) = Entropy(S) - \sum_{v \in \{weak, strong\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

In more Detail - Humidity

- S: [9+,5-]
- E=-(9/14)$\log_2$(9/14)-(5/14)$\log_2$(5/14) = 0.940

- S[3+,4-]
- E=-(3/7)$\log_2$(3/7)-(4/7)$\log_2$(4/7) = 0.985

- S[6+,1-]
- E=-(6/7)$\log_2$(6/7)-(1/7)$\log_2$(1/7) = 0.592

- GR=0.940-(7/14) x 0.985 - (7/14) x 0.592 = .151

$$Entropy(S) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i$$

$$Gain(S, Wind) = Entropy(S) - \sum_{v \in \{weak, strong\}} \frac{|S_v|}{|S|} Entropy(S_v)$$

One More Time - Wind

- S: [9+,5-]
- E=-(9/14)$\log_2$(9/14)-(5/14)$\log_2$(5/14) = 0.940

- S[6+,2-]
- E=-(6/8)$\log_2$(6/8)-(2/8)$\log_2$(2/8) = 0.811

- S[3+,3-]
- E=-(3/6)$\log_2$(3/6)-(3/6)$\log_2$(3/6) = 1.00

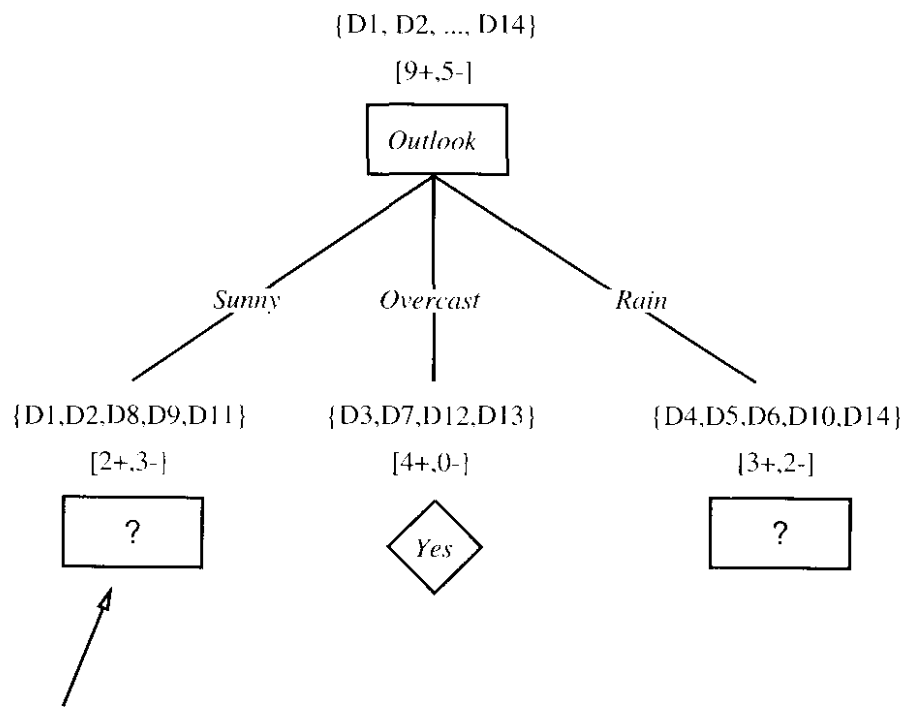
- GR=0.940-(8/14) x 0.811 - (6/14) x 1.00 = .048

Decision Tree Example

- ID3 uses Information Gain to select the best attribute at each step in growing the tree.

- Gain(S,Outlook)=0.246
- Gain(S,Humidity)=0.151
- Gain(S,Wind)=0.048
- Gain(S,Temperature)=0.029

Partially Grown Tree



Which attribute should be tested here?

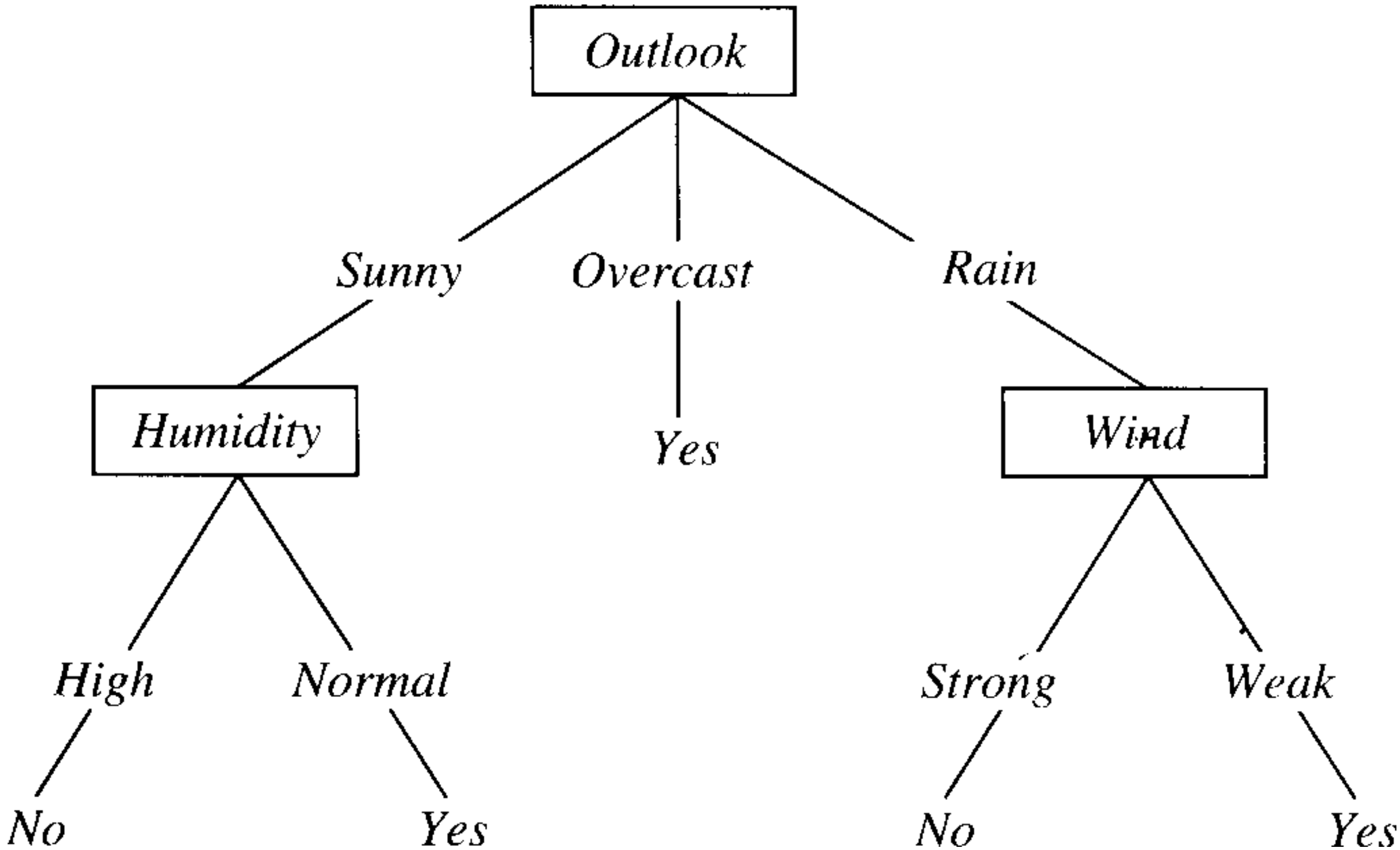
S_{sunny} = {D1,D2,D8,D9,D11}

$Gain(S_{sunny}, Humidity)$ = .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

Final Tree


Outlook
Sunny
Overcast
Rain
Humidity
Yes
Wind
High
Normal
Strong
Weak
No
Yes
No
Yes

Searching in Decision Trees

- ID3 can be seen as searching the space of possible decision trees:
  - Simple to complex hill-climbing search
  - Complete hypothesis space of finite discrete-valued functions
  - ID3 maintains only a single current hypothesis
  - Greedy Search (no backtracking)

Searching II

- Can’t tell how many alternative decision trees are consistent with the available training data

- Can’t pose queries for new instances that optimally resolve the competing hypothesis

- Pure ID3 performs no backtracking - can converge to local optimum - greedy search

- ID3 not incremental - less sensitive to errors in individual training instances - easily extended to handle noisy data

ID3 Hypothesis Space

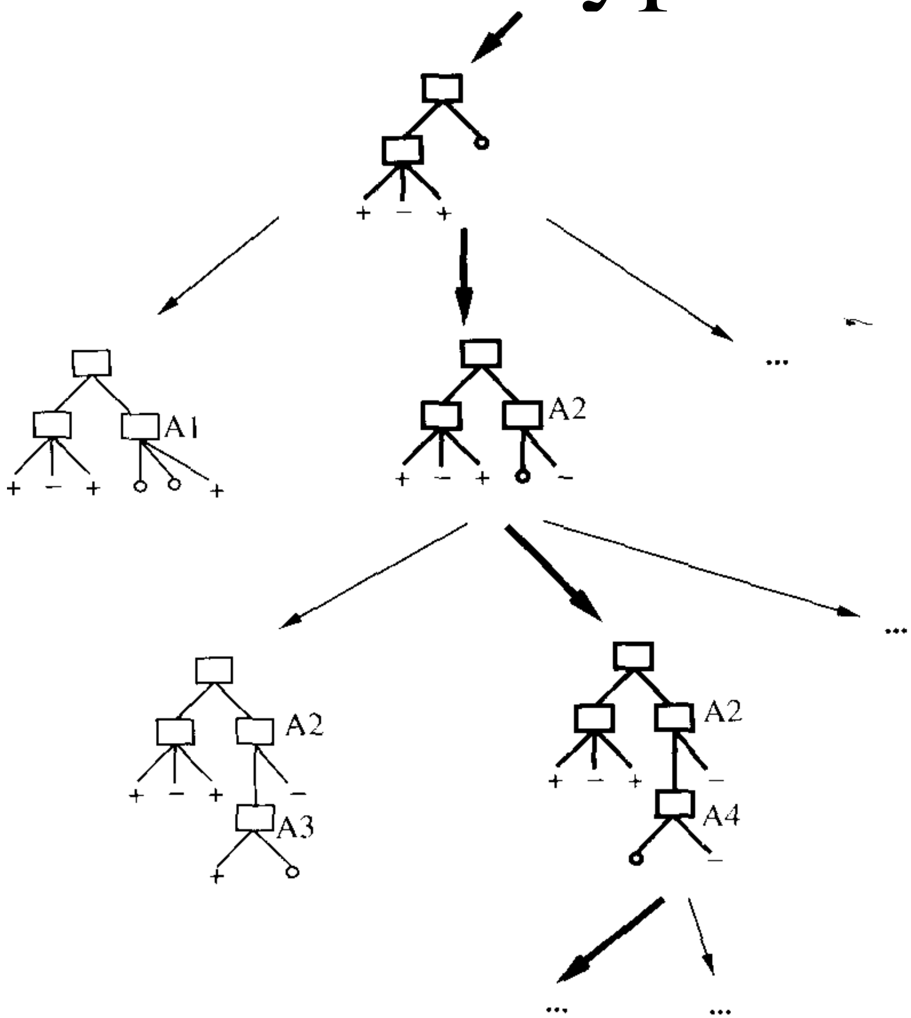


FIGURE 3.5
Hypothesis space search by ID3. ID3 searches through the space of possible decision trees from simplest to increasingly complex, guided by the information gain heuristic.

Inductive Bias in Decision Tree Learning

- Much harder to define because of heuristic search
  - Shorter trees are preferred over long ones.
  - Trees that place high information gain attributes close to the root are preferred over those that do not.

Restriction Biases and Preference Biases

- ID3 ***incompletely searches a complete hypothesis space*** from simple to complex hypothesis. Its bias is solely a consequence of the ordering of hypothesis searched. Its hypothesis space introduces no additional bias - ***preference or search bias***.

- Candidate-Elimination ***completely searches an incomplete hypothesis space.*** Its bias is solely a consequence of the expressive power of its hypothesis representation. Its search strategy introduces no additional bias - ***restriction or language bias.***

What is the Best Bias?

- A preference bias is more desirable
- First learner
  - restriction bias (linear function),
  - preference bias (LMS algorithm for parameter tuning)

Occam's razor

- Prefer the simplest hypothesis that fits the data.
- Why?

- Fewer short hypothesis then long ones - it is less likely that one will find a short hypothesis that coincidently fits the training data
- This is really rubbish!!!!

Occam’s razor is Cut

- Prefer decision trees containing exactly 17 leaf nodes with 11 nonleaf nodes, that use the decision attribute A1 at the root and test attributes A2 through A11, in numerical order.
- There are relatively few such trees and we might argue (by the same reasoning above) that our a priori chance of finding one consistent with an arbitrary set of data is therefore small.

- Another problem - based on internal learner’s representation

Different Representations

	outlook	humidity	wind	temp
	rainy	high	normal	low

	Outlook & humidity	Wind & temp
	Rainy-high	Normal-low

Overfitting Definition

- Given a hypothesis space H, a hypothesis $h \in H$ is said to overfit the training data if there exists some alternative hypothesis $h' \in H$, such that h has a smaller error than h´ over the training examples, but h´ has a smaller error than h over the entire distribution of instances.

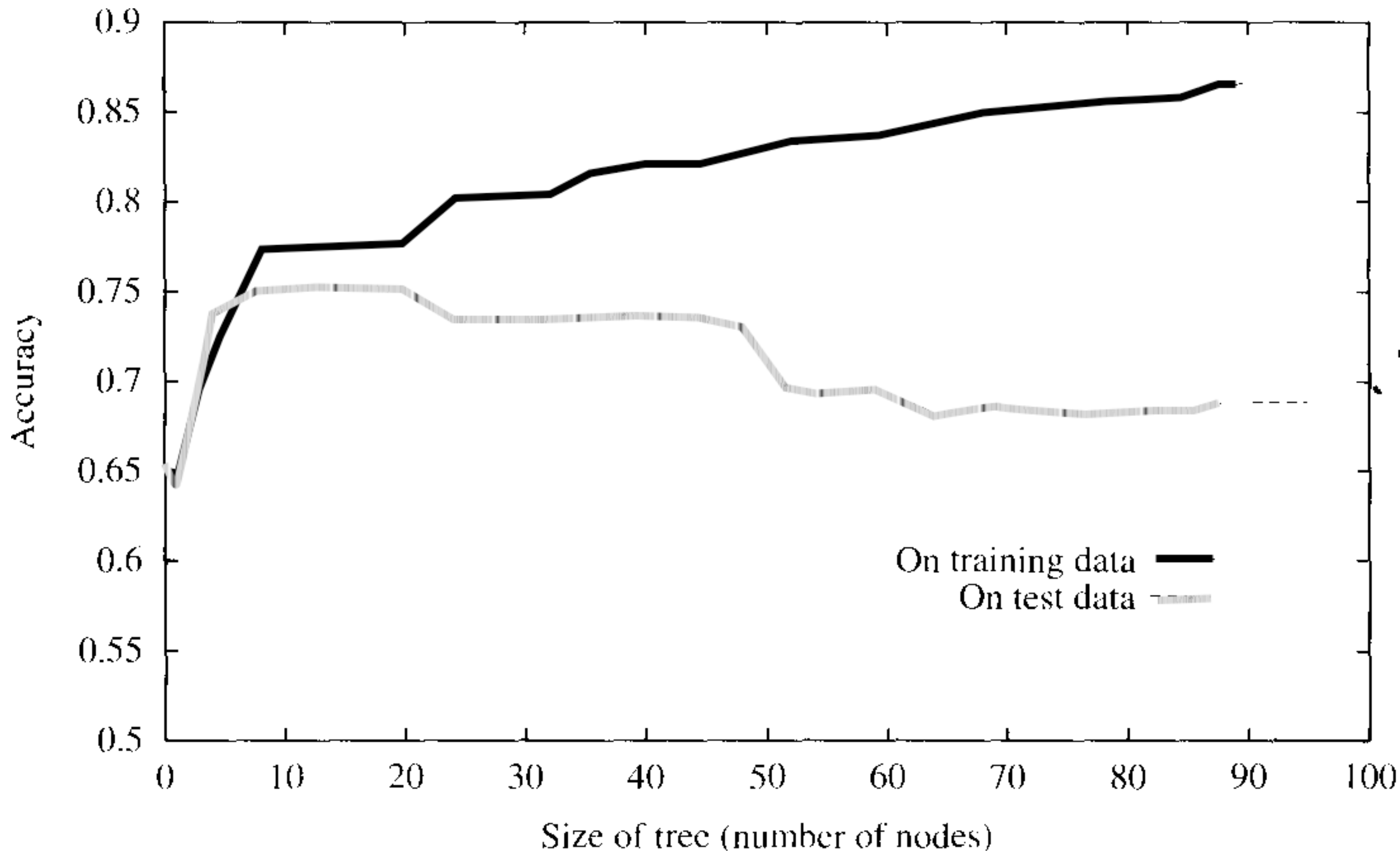
- Pretty useless definition - not causal

What Increases Overfitting

- Noise (errors) in the data,

- Number of training instances too small

Overfitting in Decision Trees



Approaches to Overfitting

- Stop growing tree earlier
- Post-prune the tree
- Separate set of examples -
  - training and validation set approach - even if the training set is mislead by random errors the validation set is unlikely to exhibit the same random fluctuations - 2/3 training, 1/3 validation
- Statistical test

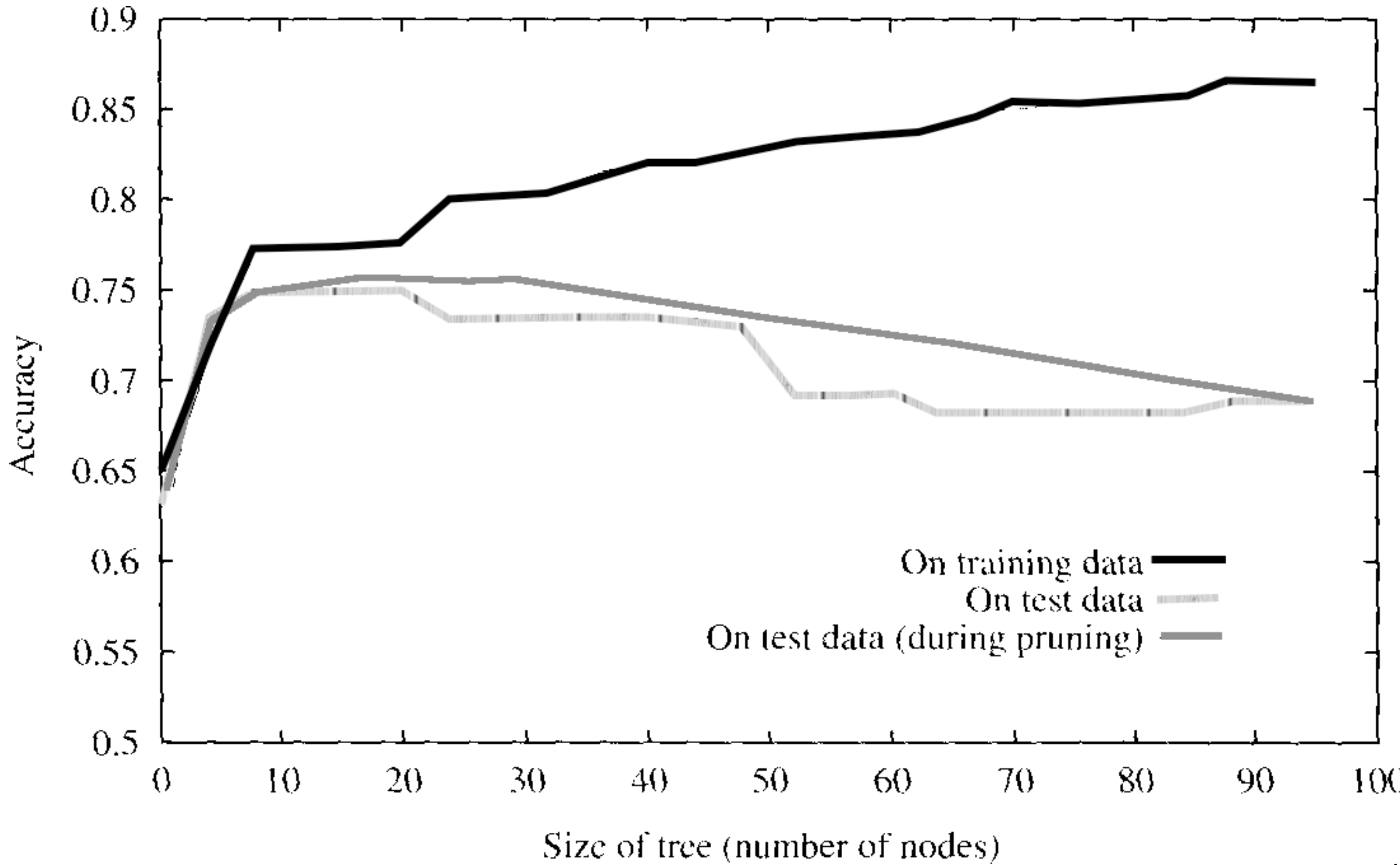
Reduced Error Pruning

- Consider each node for pruning
- Pruning = removing the subtree at that node, make it a leaf and assign the most common class at that node
- A node is removed if the resulting tree performs no worse then the original on the validation set - removes coincidences and errors

Reduced Error Pruning II

- Nodes are removed iteratively choosing the node whose removal most increases the decision tree accuracy on the graph.
- Pruning continues until further pruning is harmful.
- Uses training, validation & test sets
  - effective approach if a large amount of data is available

Impact of Reduced Error Pruning



Rule Post Pruning

1. Infer decision tree from training set

2. Convert tree to rules - one rule per branch

3. Prune each rule by removing preconditions that result in improved estimated accuracy

4. Sort the pruned rules by their estimated accuracy and consider them in this sequence when classifying unseen instances

Improved Estimated Accuracy

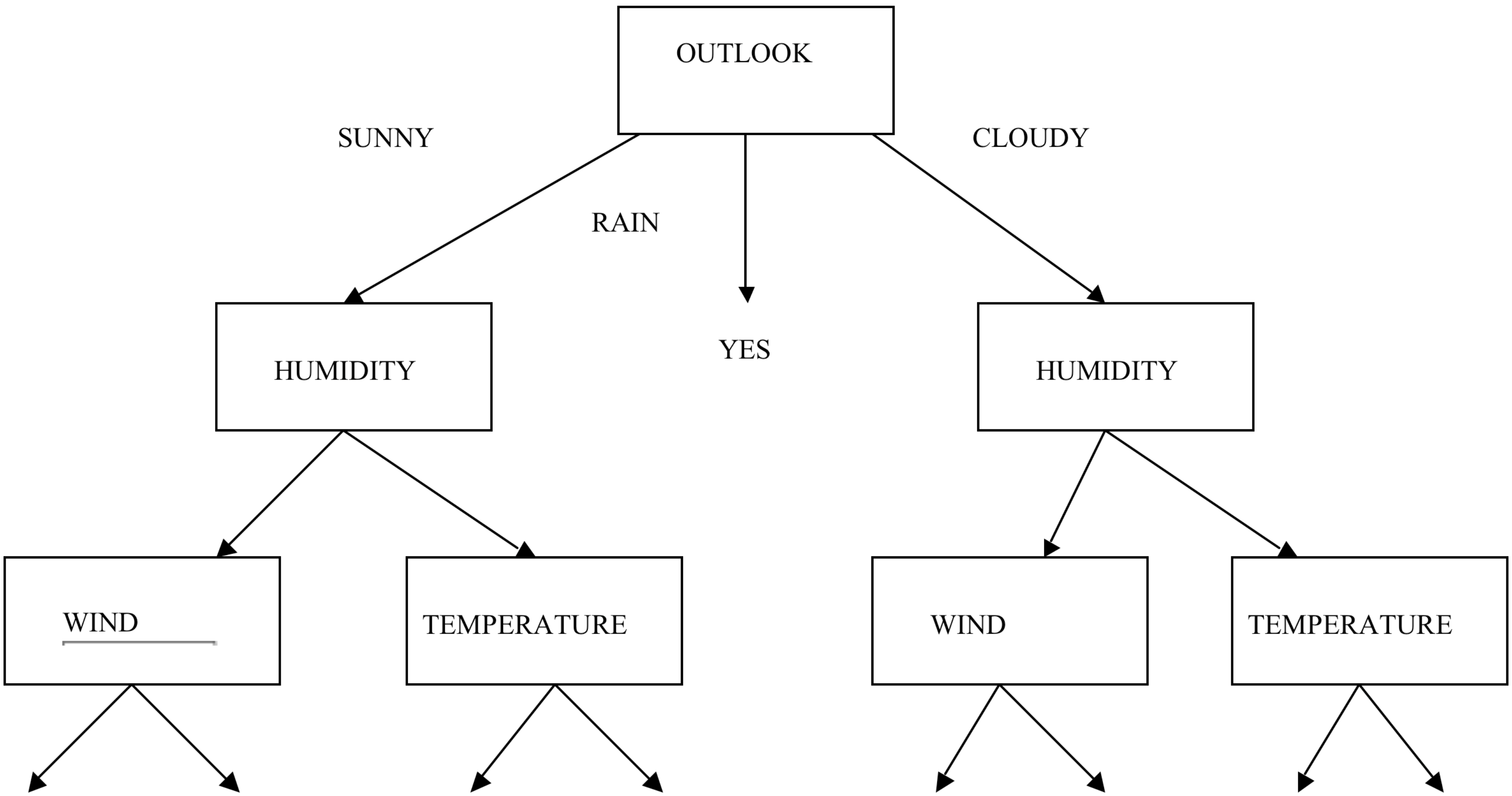
1. Calculate the rule accuracy over training data
2. Calculate the standard deviation assuming a binomial distribution
3. For a given confidence interval, lower bound estimate is taken as measure of rule performance

- For large data sets the estimated accuracy is very close to the observed whereas it grows further away as the data set size decreases
- Not statistically valid, but found useful in practice

Why Convert to Rules?

- Allows distinguishing among different contexts in which a node might be used
- Removes distinction between attribute tests near the root versus leafs
  - no messy bookkeeping
- Easier for people to understand

Tree With Redundancies



Continuous Valued Attributes?

- Dynamically creating new discrete valued attributes A_c that is true if $A < c$
  1. Sort examples according to the continuous attribute value
  2. Identify adjacent examples that differ in their target classifications
  3. Generate candidate threshold midway between these points
  4. Calculate the information gain of each candidate and pick best
  5. Dynamically created boolean attributes to compete with others to appear in tree

- The value of c that maximizes information gain must be one of these points

Example

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

- $(48+60)/2 = 54$
- $(80+90)/2 = 85$
- $\text{Temperature}_{>54}$, $\text{Temperature}_{>85}$

Other Measures for Picking Attributes

- Information Gain has natural bias towards attributes with many values over ones with few
  - For instance Date attribute has highest information gain

- Use Gain Ratio

Gain Ratio

- Entropy of S with respect to the values of A

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

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

Gain Ratio Intuition

- If attribute A splits the examples each into separate unique values (Date), SplitInfo = $\log_2 n$

- If attribute B splits the examples in half, SpiltInfo=1

- Then if attributes A and B have the same Gain then B will clearly score higher

Problems with Gain Ratio

- If $|S_i| \approx |S|$, then GainRatio is undefined or very large

- To avoid selecting attributes on this basis
  1. Calculate Gain of each attribute
  2. Calculate GainRatio only on attributes with above average Gain
  3. Choose best GainRatio

Other Evaluation Functions

- Many other evaluation functions
- Distance metric Lopez de Mantaras, 1991
  - Distance between our partition and the perfect partition
  - Not biased by number of values for an attribute

Missing Attribute Values in Training Examples

- Blood-Test_Result
  1. Standard methodology from Statistics is to throw away data
  2. Assign missing value to the most common value at node n
  3. Alternatively, assign missing value to the most common value at node n for examples with the same target value
  4. Assign probability to each possible value, estimated by frequencies at node n

Missing Attribute II

- Latter tack, can be subdivided again later in the tree
- Same approach can be used to classify examples

Attributes with Differing Costs

- Temperature, BiopsyResult, Pulse, BloodTestResults

- Prefer decision trees that use low-cost attributes where possible
  - Divide Gain by the cost of the attribute
  - Do not guarantee optimal cost-senstive decision tree, but bias the search in favor of low cost attributes

Differing Costs II

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

- Medical Domain $$\frac{2^{Gain(S,A)}-1}{(Cost(A)+1)^w}$$
  - Where w∈{0,1} is a constant that determines the relative important of cost versus information gain

Summary

- Decision Trees are practical for discrete-valued functions, grows tree from root down, selecting next best attribute at each new node added to tree.
- ID3 searches complete hypothesis space. It can represent any discrete-valued function defined over discrete values instances, therefore it avoids the problem of the target function not being in the hypothesis space.
- Inductive Bias implicit in ID3 is ***preference*** for smaller trees, only grows as large as needed to classify training examples.

Summary continued

- Overfitting data is an important issue.
- Very large variety of extensions: post-pruning, handling real-valued attributes, accommodating missing attribute values, incrementally refining decision trees, other attribute selection measures, considering costs associated with instance attributes (or target values).