# Decision Tree Learning

Presented By--

Khushi Singh.
2003480100036
COE-CS3A

- Introduction
- Decision tree representation
- Learning algorithm
- Hypothesis space search
- Inductive bias
- Issues in decision tree learning

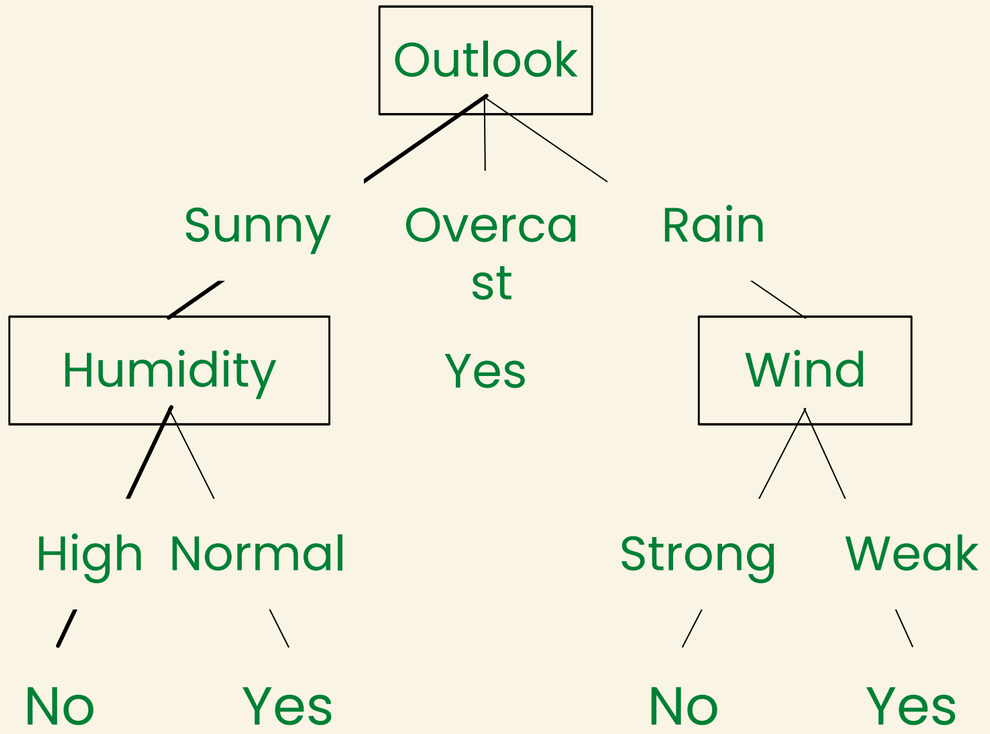
# Examples of Decision Tree

- Data set

| Day | Outlook | Temperatur | Humidity | Wind | Play Tennis |
|---|---|---|---|---|---|
| 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 |

# Examples of Decision Tree

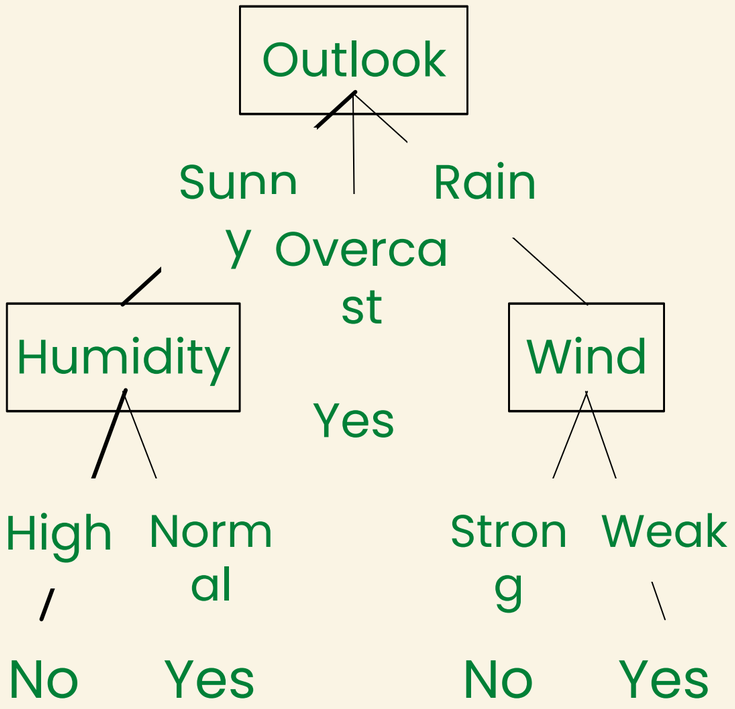
Play tennis?



- Decision tree learning
  - One of the most widely used and practical methods for inductive inference
  - Approximate discrete-valued target function
    - The learned function is represented by a decision tree
    - Decision tree can also be re-represented as sets of if-then rules to improve human readability
  - Robust to noisy data and capable of learning disjunctive expressions

- Classification of instances
  - Decision tree classify instances by sorting them down the tree from the root to some leaf node
- Node
  - Specifies test of some attribute
- Branch
  - Corresponds to one of the possible values for this attribute

# Decision Tree Representation (2/2)




- e.g.
  - Saturday morning
  - (Outlook=sunny, Temperature=Hot,
  - Humidity=high, Wind=Strong)
  - (Outlook=Sunny $\wedge$ Humidity=High) so NO
- Each path corresponds to a conjunction of attribute tests
- Decision trees represent a disjunction of conjunction of constraints on the attribute values of instances
  - (Outlook=Sunny $\wedge$Humidity=normal)
    $\vee$(Outlook=Overcast)
    $\vee$(Outlook=Rain $\wedge$Wind=Weak)

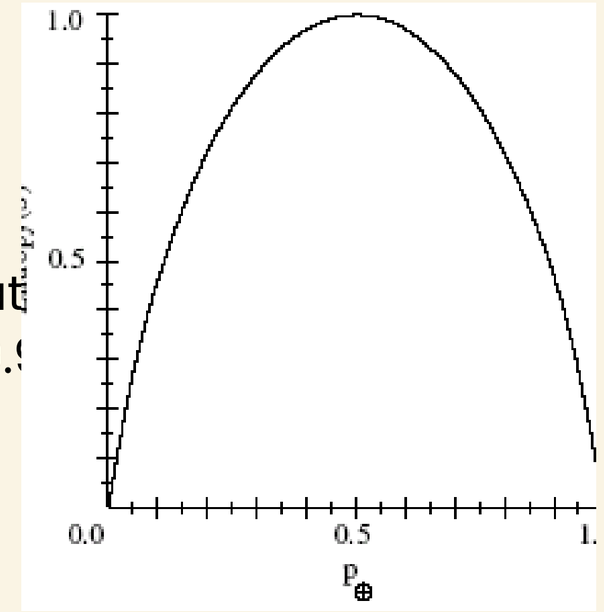
- Instances are represented by attribute-value pairs
- The target function has discrete output values
- Disjunctive descriptions may be required
- The training data may contain errors
  - Both errors in classification of the training examples and errors in the attribute values
- The training data may contain missing attribute values
- Suitable for classification

- Main question
  - Which attribute should be tested at the root of the (sub)tree?
    - Greedy search using some statistical measure
- Information gain
  - A quantitative measure of the worth of an attribute
  - How well a given attribute separates the training example according to their target classification
  - Information gain measures the expected reduction in entropy

- Entropy
  - characterizes the (im)purity of an arbitrary of examples
  - Entropy specifies the minimum
  - For example
    - The information required for classificat
    - =-(9/14)log$_2$(9/14)-(5/14)log$_2$(5/14)=0.

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

- $S$ 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 $S$



- According to information theory
  - Optimal length code assigns $-\log_2 p$ bits to message having probability p
- General form of entropy

c : Number of values.

Pi : The proportion of S belonging to class i

# Learning Algorithm

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

- Values (A): the set of all possible values for attribute A
- Sv : the subset of S for which attribute A has value v

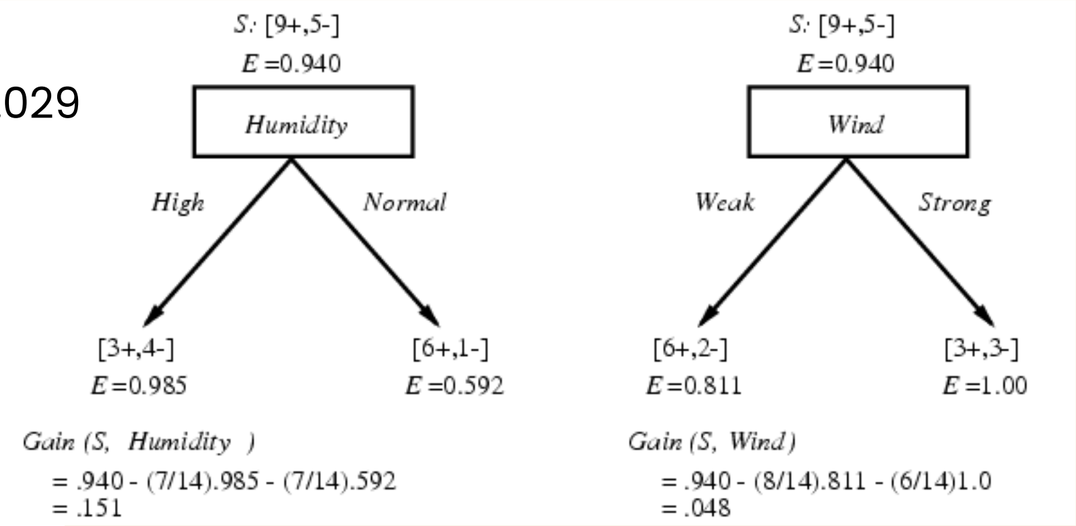
- Information gain and entropy
  - First term: the entropy of the original collection
  - Second term: the expected value of the entropy after S is partitioned using attribute A
- Gain (S ,A)
  - The expected reduction in entropy caused by knowing the value of attribute A
  - The information provided about the target function value, given the value of some other attribute A

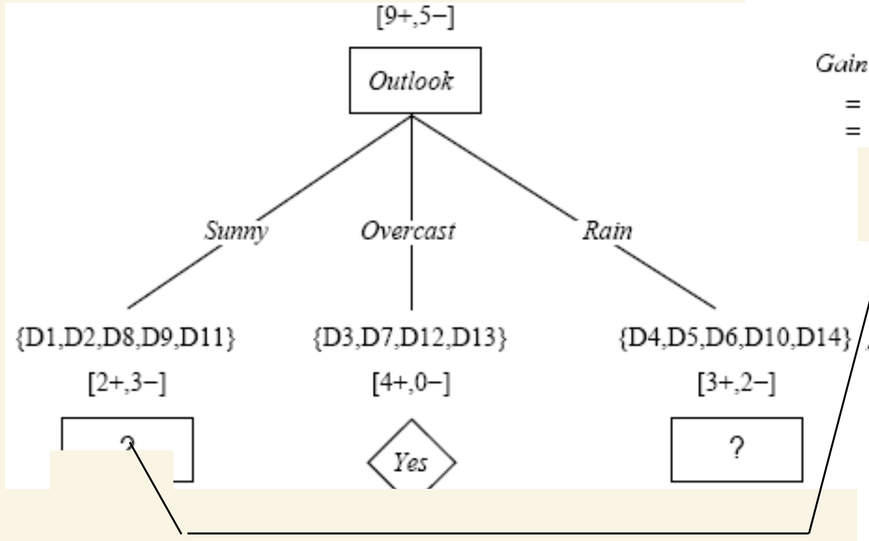
- ID3 (Examples, Target_attribute,Attributes)
  - Create a Root node for the tree
  - If all Examples are positive, Return the single node tree Root, with label= +
  - If all Examples are negative, Return the single node tree Root, with label= -
  - If Attributes is empty, Return the single-node tree Root, with label = most common value of Target_attribute in Examples
  - Otherwise Begin

- Otherwise Begin
  - A  the attribute from Attributes that best classifies Examples
  - The decision attribute for Root  A
  - For each  possible value, vi ,of A,
    - Add a new tree branch below Root, corresponding to the test A = vi
    - Let Examplesvi be the subset of Examples that have value vi for A
    - If Examplesvi  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(Examplesvi, Target_attribute, Attributes – {A})
- End
- Return Root

# An Illustrative Example (2/2)

- Selecting root
  - The information gain values for all four attributes
    - Gain(S, Outlook)= 0.246  selected as root attribute
    - Gain(S, Humidity)= 0.151
    - Gain(S, Wind)= 0.048
    - Gain(S, Temperature)= 0.029
- Adding a subtree




*Which attribute should be tested here?*

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

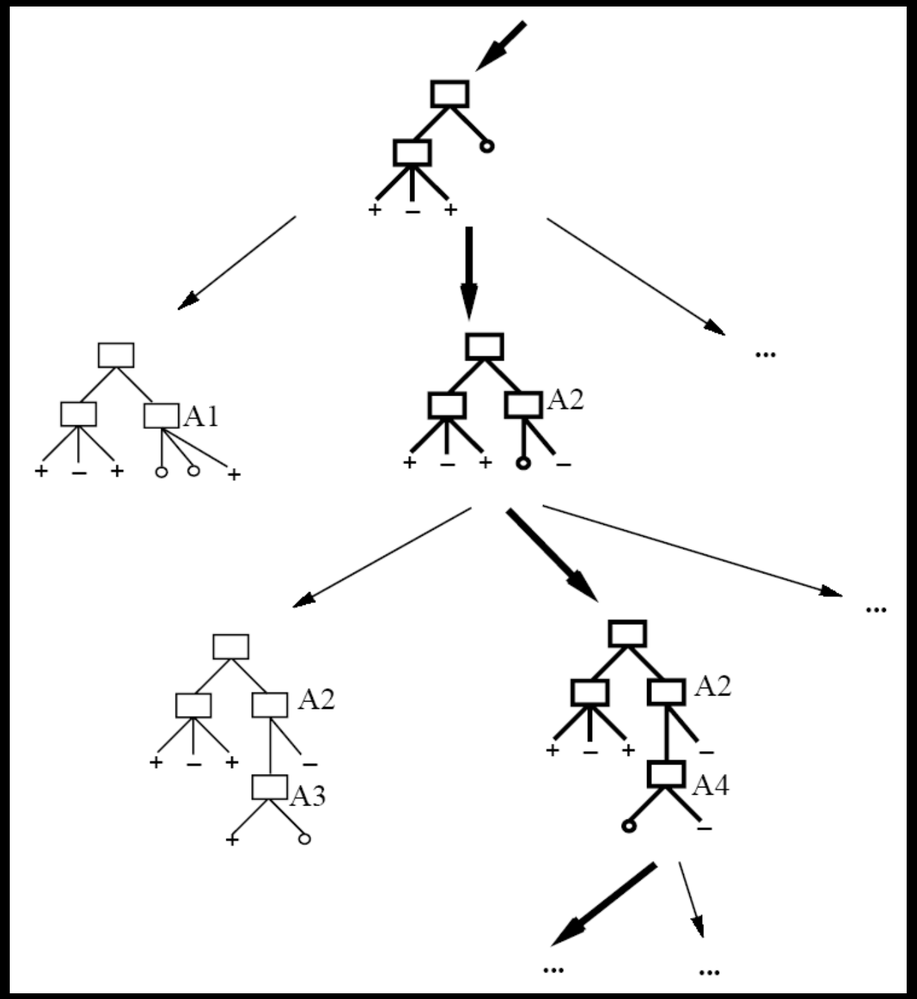
*Gain* ($S_{sunny}$ , *Humidity*) = .970 − (3/5) 0.0 − (2/5) 0.0 = .970

*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 (1/2)

- Hypothesis space
  - The set of possible decision trees
  - Simple to complex, hill-climbing search



- Capability
  - Hypothesis space of all decision trees is a complete space of finite discrete-valued functions
  - ID3 maintains only a single current hypothesis
    - Can not determine  how many alternative decision trees are consistent with the available training data
    - Can not pose new instance queries that optimally resolve among competing hypothesis
  - No backtracking in its search
    - Converging to local minima
  - ID3 uses all training example at each step to make statistically based decisions regarding how to refine its current hypothesis
    - Advantage: The resulting search is much less sensitive to errors in individual training examples

- Approximate inductive bias of ID3
  - Shorter trees are preferred over larger tress
  - BFS-ID3
- A closer approximation to the inductive bias of ID3
  - Shorter trees are preferred over longer trees.
  - Trees that place high information gain attributes close to the root are preferred over those that do not.

- ID3
  - Searches a complete hypothesis space incompletely
  - Inductive bias is solely a consequence of the ordering of hypotheses by its search strategy

- Candidate-Elimination
  - Searches an incomplete hypothesis space completely
  - Inductive bias is solely a consequence of the expressive power of its hypothesis representation

- Preference Bias
  - ID3
  - Preference for certain hypotheses over others
  - Work within a complete hypothesis space

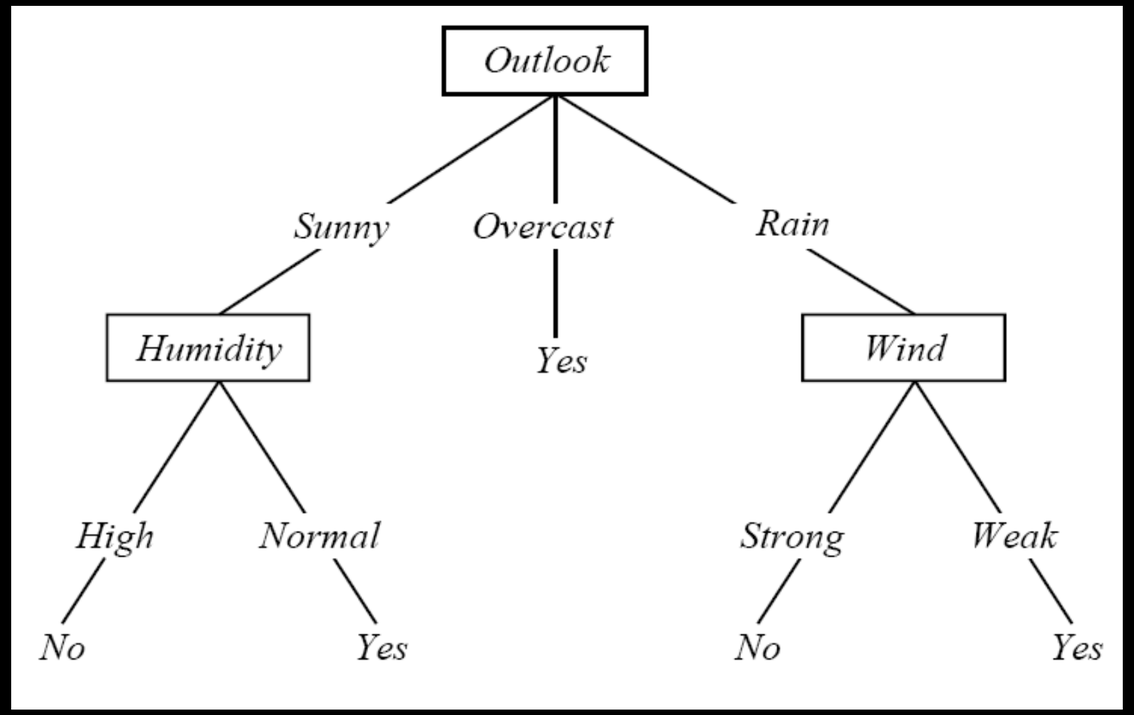
- Restriction Bias
  - Candidate-Elimination
  - Categorical restriction on the set of hypotheses considered
  - Possibility of excluding the unknown target function

- Prefer the simplest hypothesis that fits the data
- Argument in favor
  - Fewer short hypotheses than long hypotheses
- Argument opposed
  - There are many ways to define small sets of hypotheses
  - The size of a hypothesis is determined by the particular representation used internally by the learner

- Determine how deeply to grow the decision tree
- Handling continuous attributes
- Choosing an appropriate attribute selection measure
- Handling training data with missing attribute values
- Handling attributes with differing costs
- Improving computational efficiency

# Overfitting in Decision Trees

- Consider adding noisy training example
  - <Sunny, Hot, Normal, Strong, PlayTennis = No>
  - What effect on earlier tree?

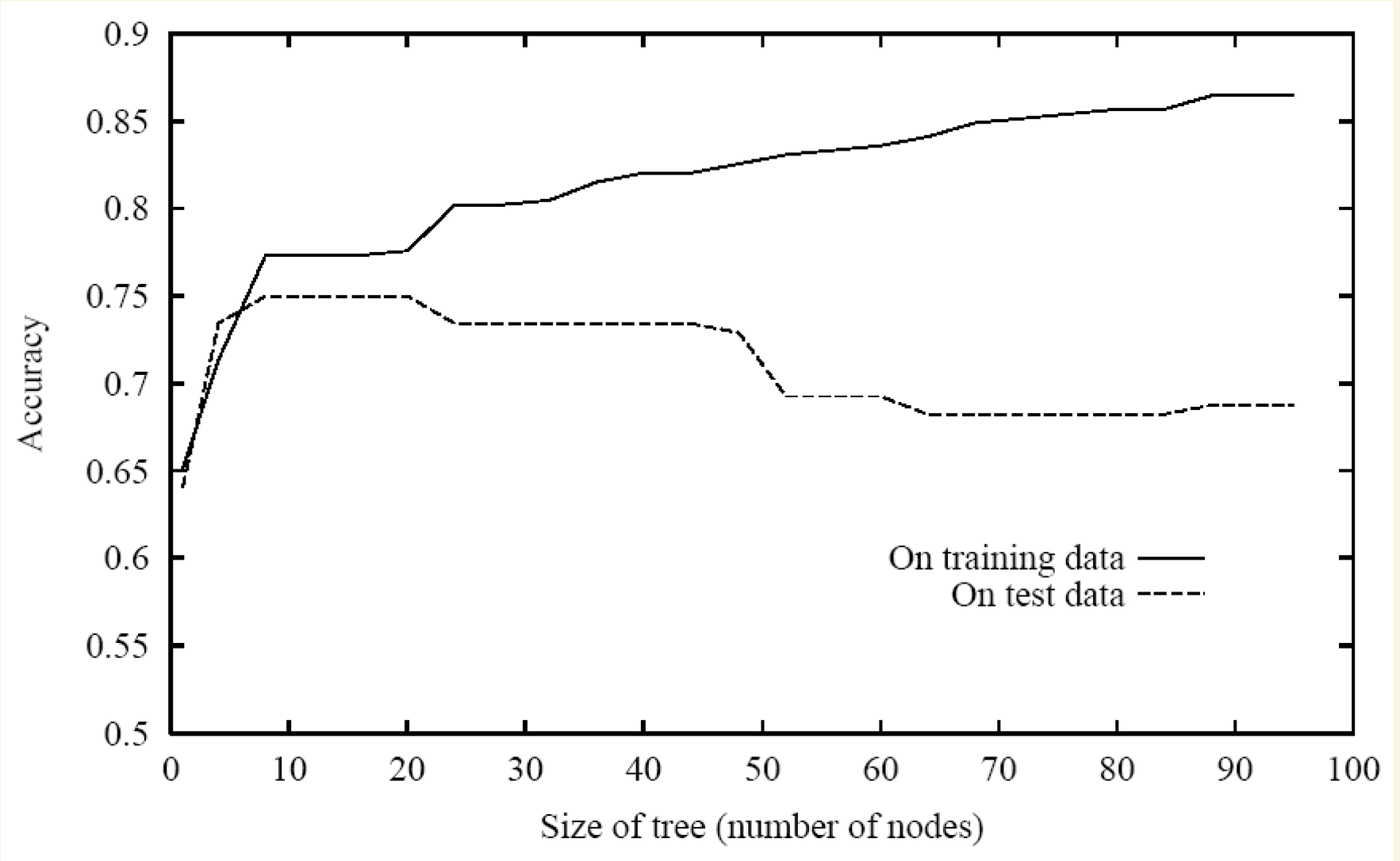


- Consider error of hypothesis h over
  - Training data: $error_{train}(h)$
  - Entire distribution D of data: $error_D(h)$
- Hypothesis $h \in H$ overfits training data if there is an alternative hypothesis $h' \in H$ such that

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

and

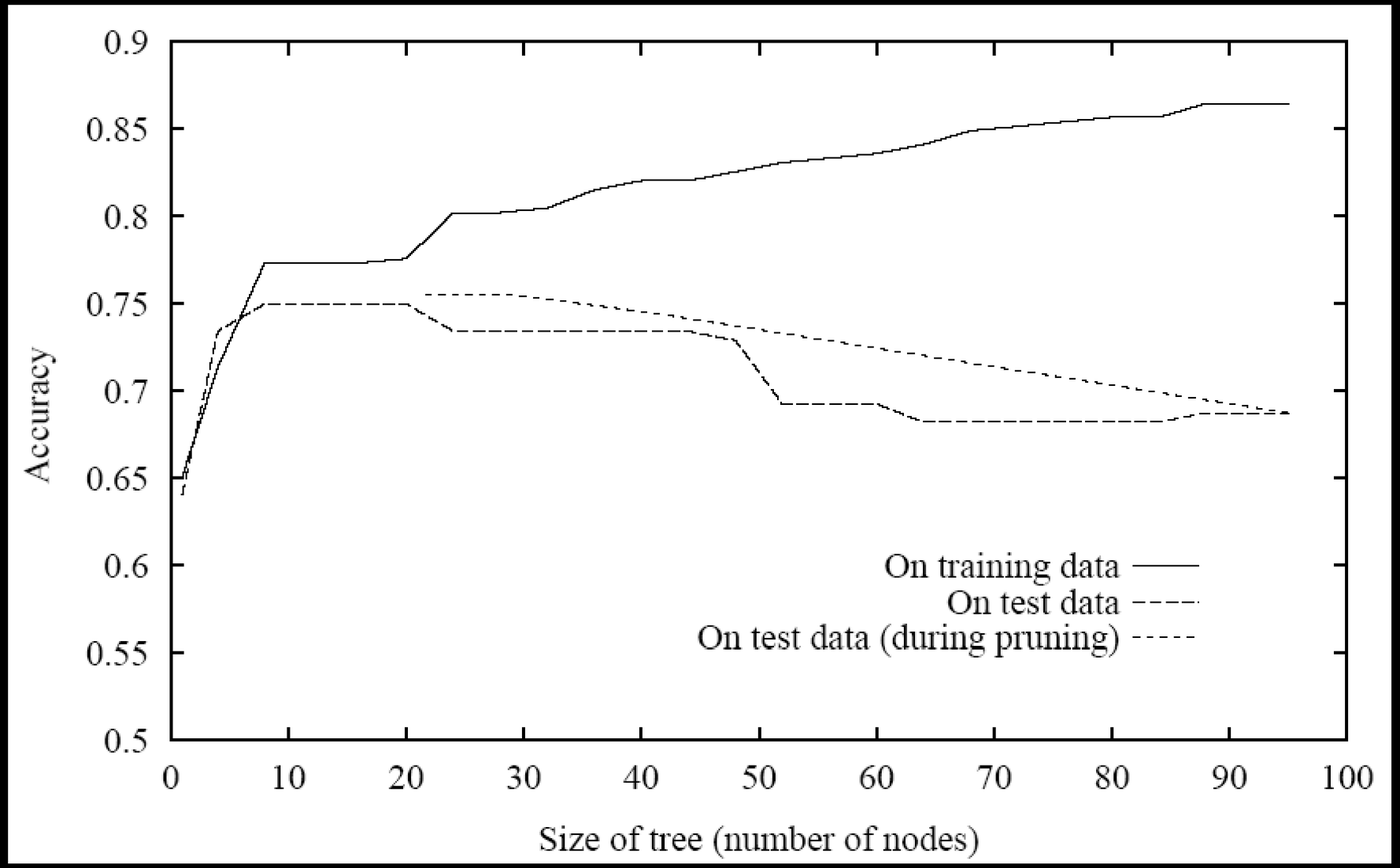
$$error_D(h) > error_D(h')$$


0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
Accuracy
0
10
20
30
40
50
60
70
80
90
100
Size of tree (number of nodes)
On training data
On test data

- How can we avoid overfitting?
  - Stop growing before it reaches the point where it perfectly classifies the training data
  - Grow full tree, then post-prune
    - Reduced-error pruning
    - Rule post-pruning
- How to select best tree?
  - Measure performance statistically over training data
  - Measure performance over separate validation data set
  - MDL(Minimum Description Length): minimize the complexity for encoding the training examples and the decision tress

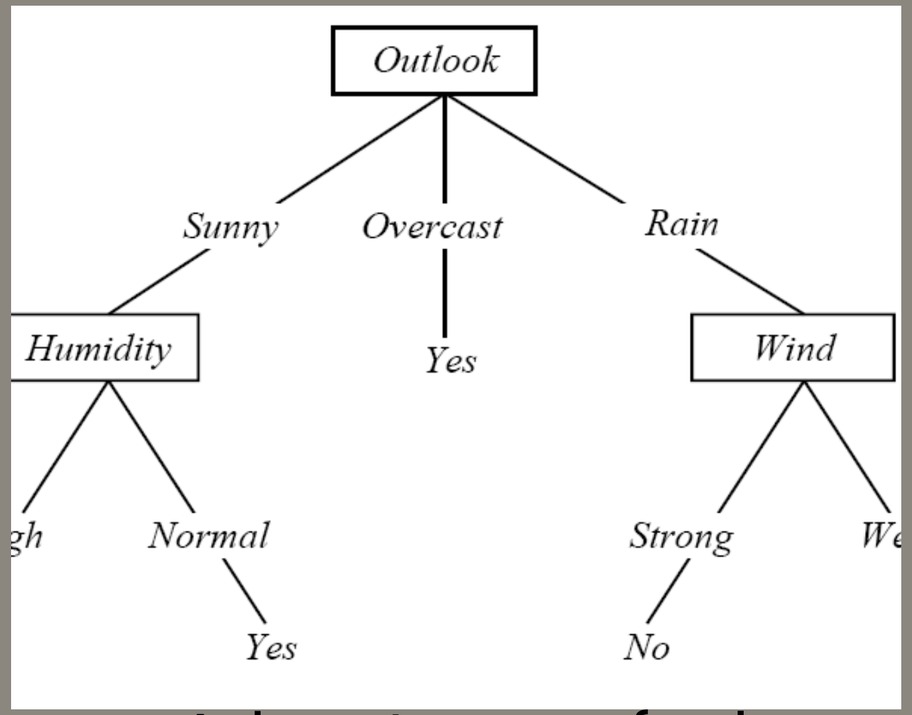
- Split data into training set, validation set used for pruning, and test set for measuring accuracy over future unseen examples.
- Do until further pruning is harmful:
    1. Evaluate impact on validation set of pruning each possible node (plus those below it), starting at its maximum size and lowest accuracy over test set.
    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



- Most frequently used method (e.g., C4.5)
  1. Infer the decision tree from the training set, growing the tree until the training data fit as well as possible and allowing overfitting to occur
  2. Convert the tree into an equivalent set of rules
  3. Prune (generalize) each rule by removing any preconditions that result in improving its estimated accuracy
  4. Sort the pruned rules by their estimated accuracy

# Rule Post-Pruning




- Converting a tree to rules
  IF (Outlook = Sunny) ∧ (Humidity = High)
  THEN PlayTennis = No
  IF (Outlook = Sunny) ∧ (Humidity = Normal)
  THEN PlayTennis = Yes
  …

- Advantages of rule post-pruning over reduced-error pruning
  - Allows distinguishing among the different contexts in which a decision node is used.
  - Removes the distinction between attributes near the root and those near the leaves.
  - Improves readability.

- Define new discrete valued attributes that partition the continuous attribute value into a discrete set of intervals
- Fir n different ta ure>54 and Temperature>85
- Pick a threshold, c, that produces the greatest information gain : temperature>54

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

- Problem
  - If attribute has many values, Gain will select it
  - Imagine using Date = Oct_13_2004 as attribute
- One approach: use GainRatio instead

where Si is subset of S for which A has value vi

- 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 pi to each possible value vi of A
    - Assign fraction pi of example too each descendant in tree
- Classify new examples in same fashion

- Use low-cost attributes where possible, relying on high-cost attributes only when needed to produce reliable classficiations
- Tan and Schlimmer (1990)
- Nunez (1988)

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

- Allows for attributes that have a whole range of discrete or continuous values
- Post-pruning after induction of trees, e.g. based on test sets, in order to increase accuracy
- Uses gain ratio as the information gain measure to replace the old biased method
- Handles training data with missing attribute values by replacing them with the most common or the most probable value

- Practical method using greedy search for concept learning and for learning other discrete-valued functions
- ID3 searches a complete hypothesis space
- Preference for smaller trees
- Overfitting the training data
- Large variety of extensions to the basic ID3