Ch3. Decision Tree Learning

S. Visa

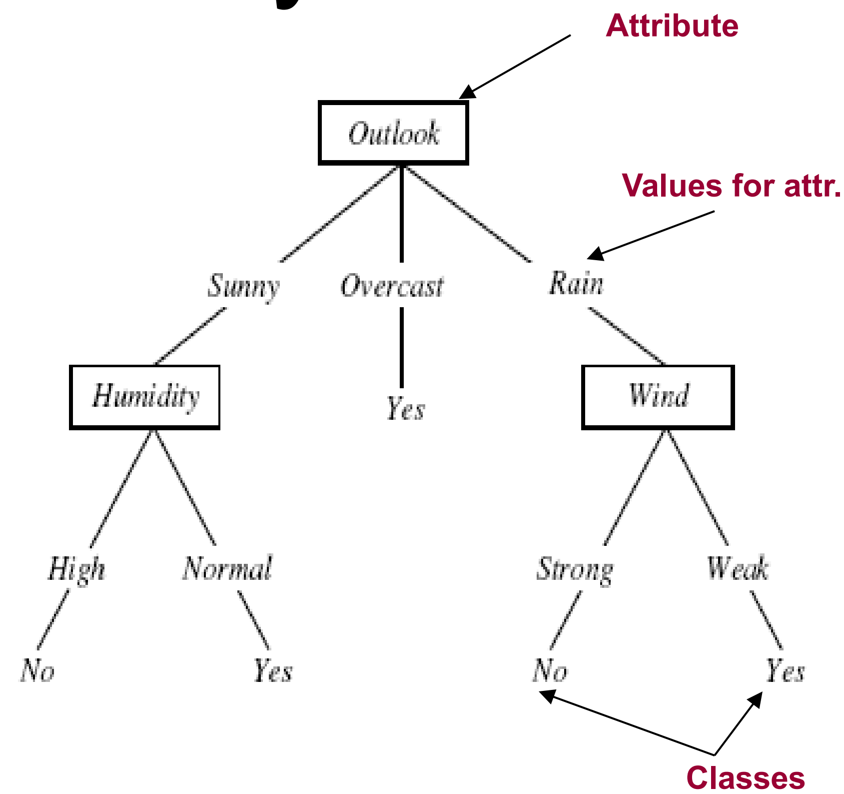
Training examples for PlayTennis

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

Decision tree for PlayTennis

- Is a representation for classification
- Each path = conjunction of attributes
- Tree = a disjunction of conjunctions

e.g.
(Outlook = Sunny and Humidity = Normal)
or
(Outlook = Overcast)
or
(Outlook = Rain and Wind = Weak)



Classify the following: (Outlook = Sunny, Temperature = Hot, Humidity = High, Wind = Strong)

Exercise

- Represent the following as decision trees
  - $\wedge, \vee, \mathrm{XOR}$
  - $(A \wedge B) \vee (C \wedge \neg D \wedge E)$

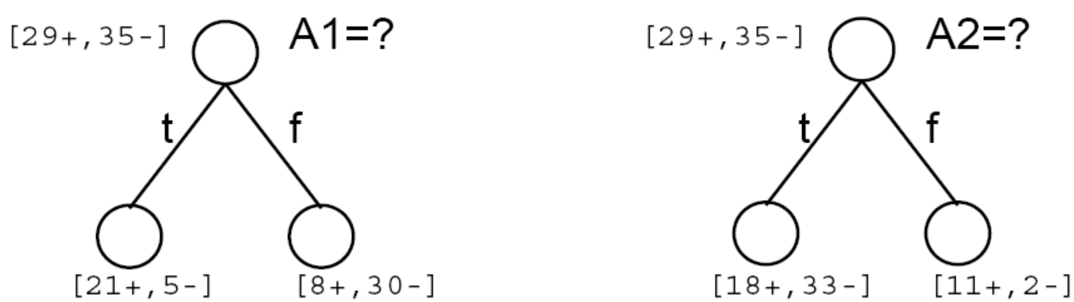
Top down induction of decision trees

- Greedy search through the space of possible 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?



ID3 and C4.5 algorithms

- Challenge in designing dec. trees →how to select current best splitting attribute?
- ID3 (Iterative Dichotomiser 3) - Quinlan ’86 → use information gain; algorithm at p56
- C4.5 – Quinlan ’93 → improvements: discrete and continuous attributes, missing attribute values, attributes with differing costs, pruning trees (replacing irrelevant branches with leaf nodes)
- C5 – advantages: faster, memory efficiency, smaller decision trees, ability to weight different attributes

ID3 algorithm (p56)

- The ID3 algorithm can be summarized as follows:

1. Take all unused attributes and count their entropy
2. Choose attribute for which entropy is minimum
3. Make node containing that attribute

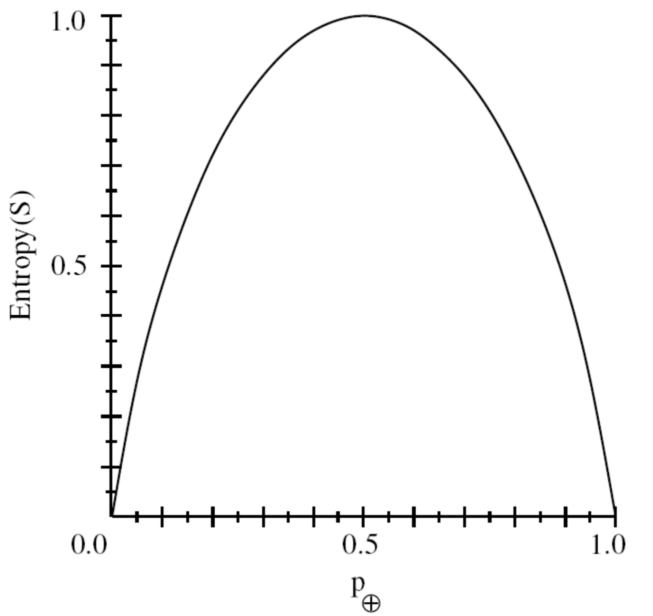
Entropy

- Def.1 Measures the impurity of S
  - e.g. most impure when 50 +ve and 50 –ve ➔ E(S) = 1
  - e.g. most pure/uniform when 0 +ve (or 0 –ve) ➔ E(S) = 0
- S – set of training ex.
- p_+ = proportion of pos. ex.
- p_- = proportion of pos. ex.

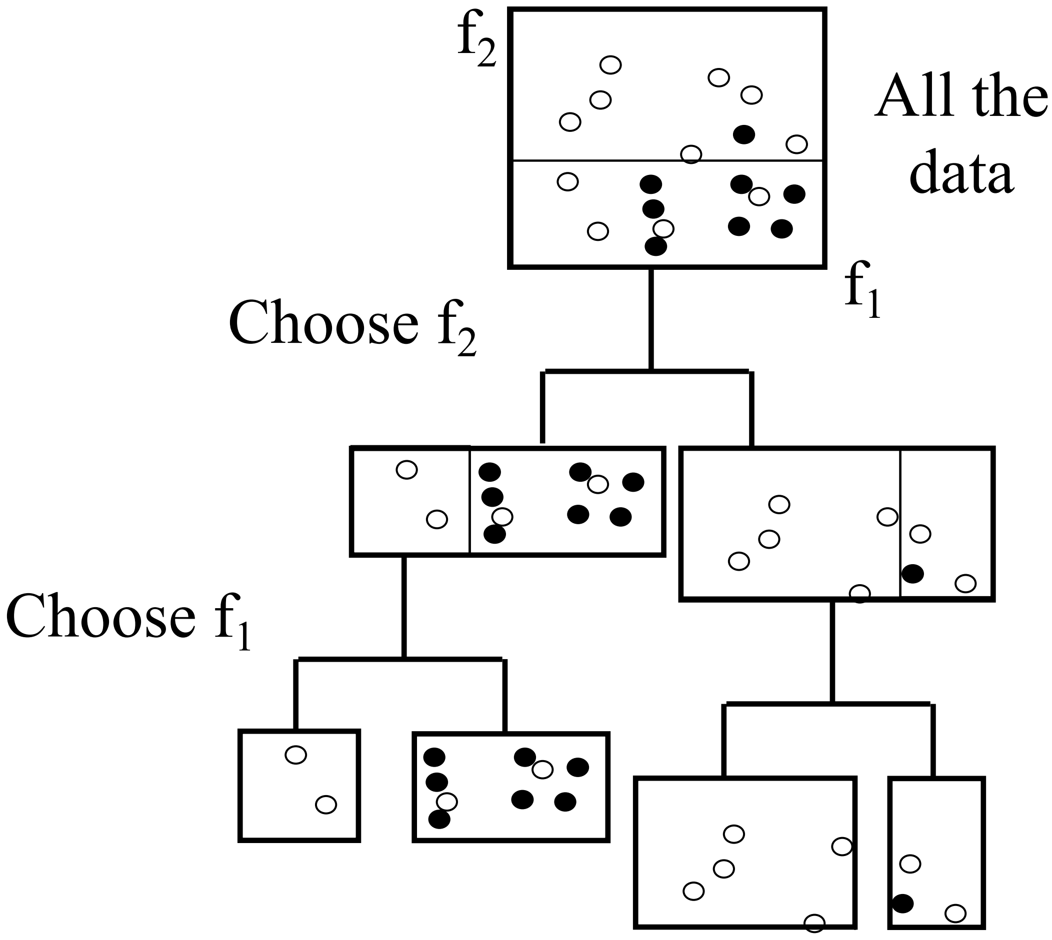
$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

1) Compute entropy for a set S of 9 +ve and 5 -ve ex.

2) Compute entropy for a set S of 8 +ve and 13 -ve ex.



Tree classifier – ex.

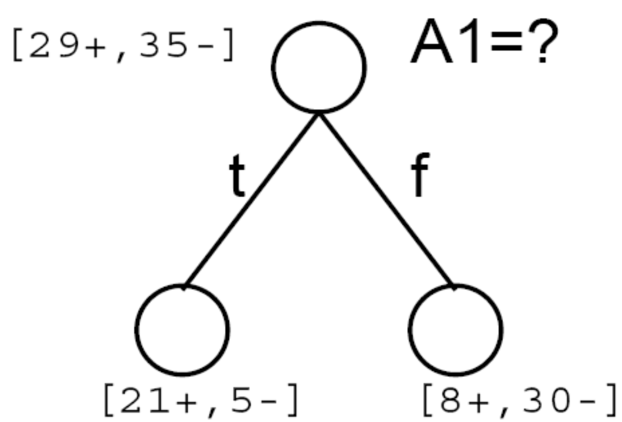


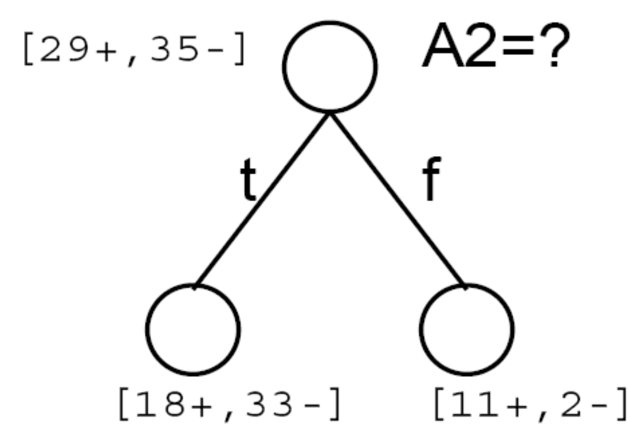
At each step, choose the feature that "reduces entropy" most. Work towards "node purity".

Information gain

- Gain(S,A) = expected reduction in entropy after S is partitioned using attr. A

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





Example1 – information gain

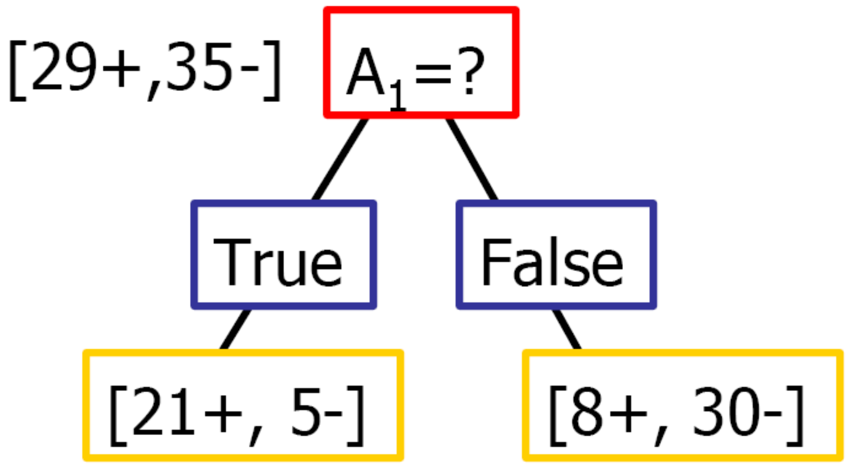
- $G(S, A1) = ?$
- $G(S, A2) = ?$

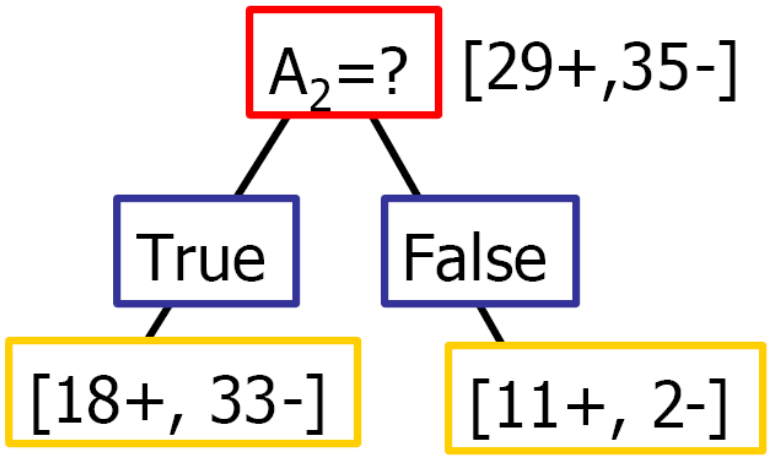
- $E(S) = ?$

- For $G(S, A1)$
  - $E(S_t) = ?$
  - $E(S_f) = ?$
- For $G(S, A2)$
  - $E(S_t) = ?$
  - $E(S_f) = ?$

Example1 – information gain

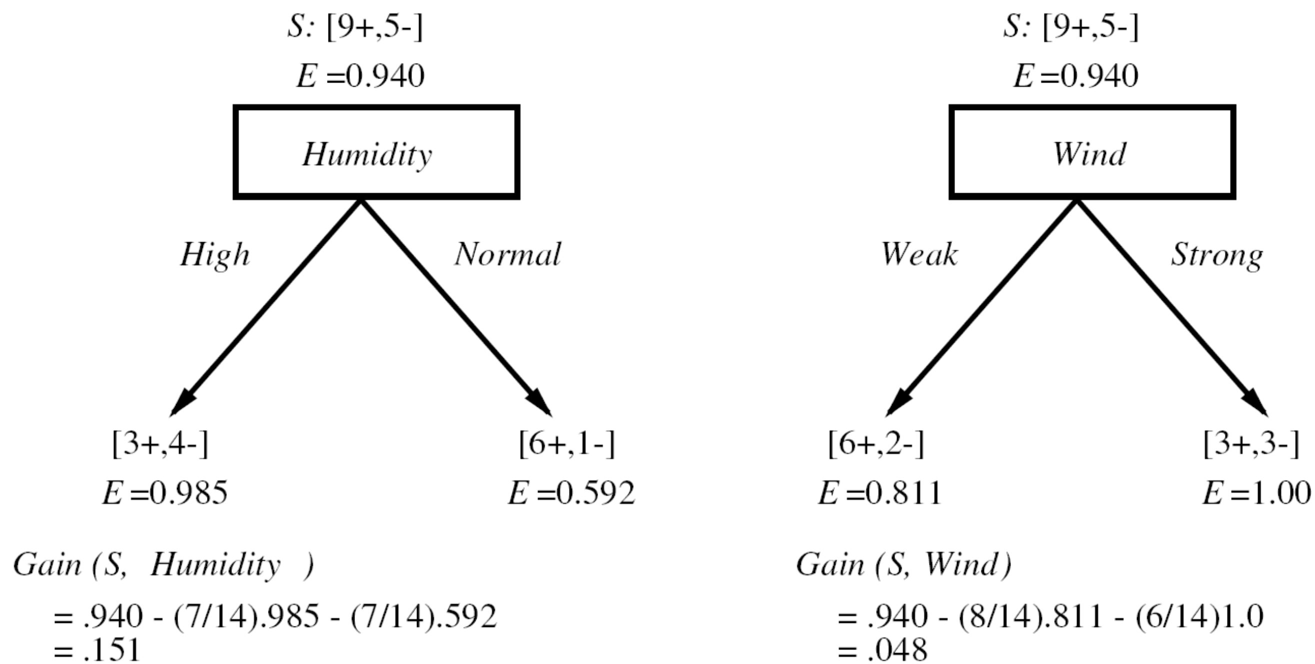
- $E(S) = E([29+,35-]) = -29/64 \log_2 29/64 - 35/64 \log_2 35/64 = 0.99$
- For G(S, A1)
  - $E(S_t) = E([21+,5-]) = 0.71$
  - $E(S_f) = E([8+,30-]) = 0.74$
  - **$G(S, A1) = E(S)-26/64\ E([21+,5-])-38/64\ E([8+,30-]) = 0.27$**
- For G(S, A2)
  - $E(S_t) = E([18+,33-]) = 0.94$
  - $E(S_f) = E([11+,2-]) = 0.62$
  - **$G(S, A2) = E(S)-51/64\ E([18+,33-])-13/64\ E([11+,2-]) = 0.12$**





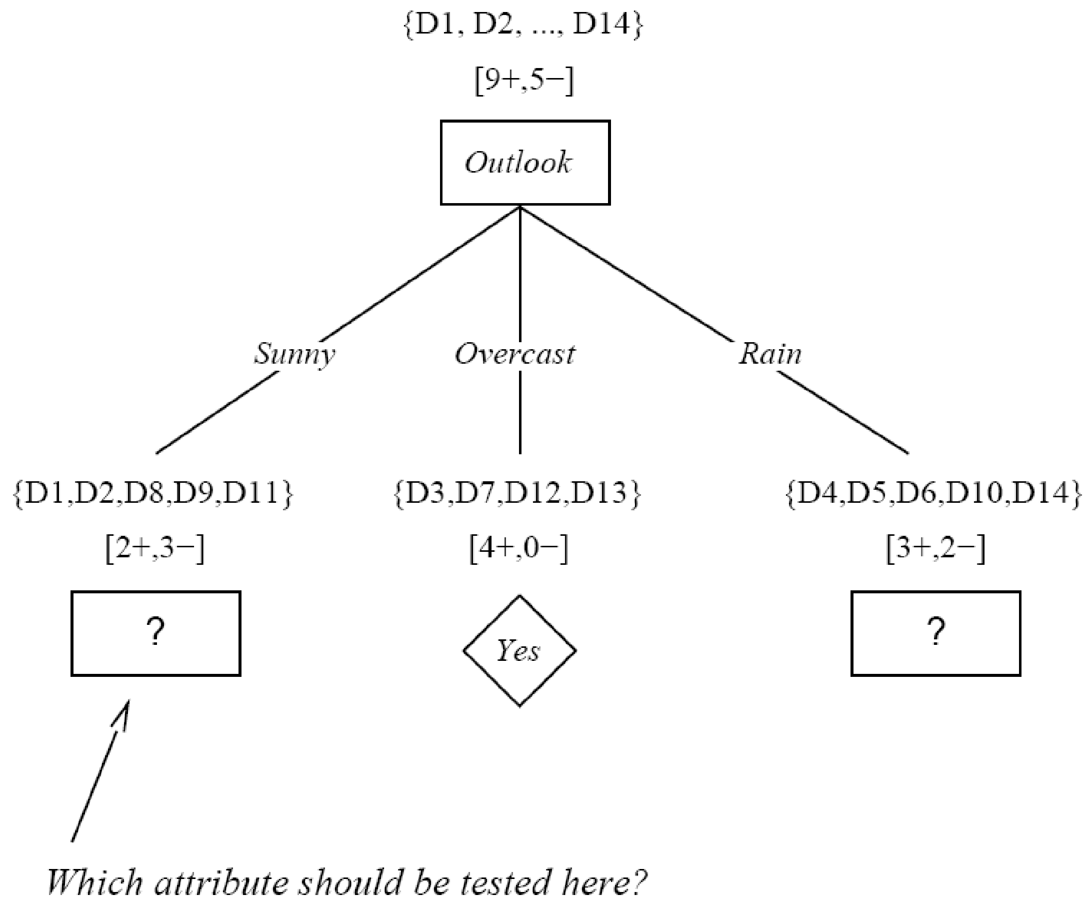
Example 2 - selecting the next attribute

- Which attribute is the best classifier for S = 9+ and 5-?



- ➔select attr. Humidity (gives grater info. gain)

Example 3 – selecting the next attribute



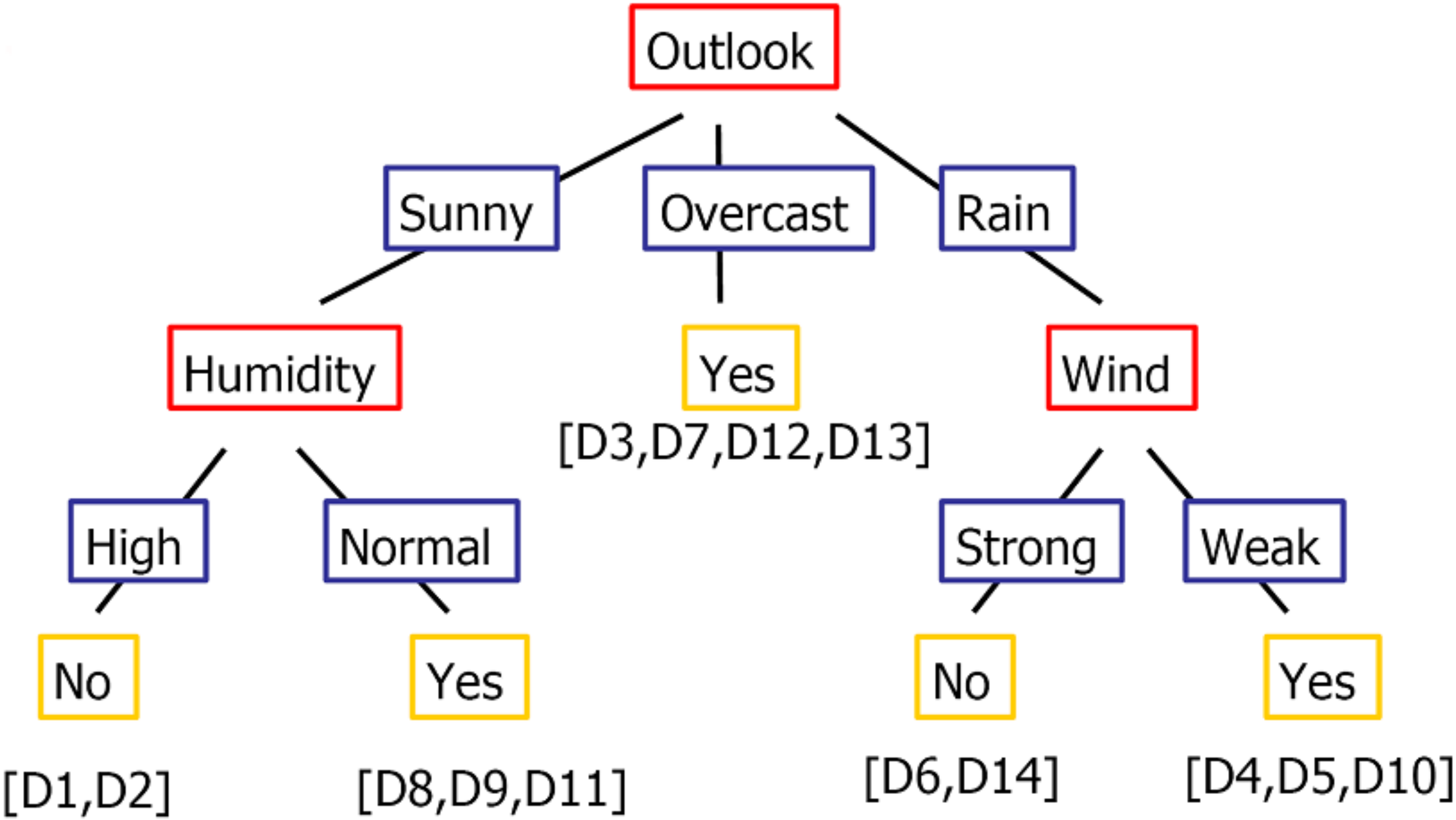
S_{sunny} = {D1,D2,D8,D9,D11}

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*) $= .970 - (2/5)\ 1.0 - (3/5)\ .918 = .019$

ID3 algorithm



D8 also comes here

Strengths (s) and weaknesses (w) of ID3

- Can learn any concept (s) and (w) → overtraining if no inductive bias (e.g. limit no. of nodes in tree)
- Develops only a single h (w)
- No backtracking → may converge to local minima (w)
- Robust to noise (err in tr. data) – trains on statistical properties of entire tr. set rather than learning in response to individual ex. (s)

Inductive bias in ID3

- Search of h in the space of all possible trees
- Prefer shorter trees to longer trees
- Prefer trees with high info gain nodes close to root
- ID3 vs. Candidate elimination
  – ID3 - searches a complete H incompletely
  – Candidate elimination – completely searches an incomplete H

Occam's Razor

- ID3 biased towards short trees ➔ KISS: Keep it simple, stupid!
- Many people feel that there is some natural law (philosophy) stating that simple sol. are better than complicated ones
- William of Occam (14th century) while shaving:

  Given the choice of 2 ways to solve a problem, select the simpler of two
- Mitchell: **prefer the simplest h that fits the data**
- Arguments in favor of short h
  - Fewer short h than long ones
  - A short h that fits data is unlikely to be a coincidence
- Occam's R. relates well with the pb. of overfitting: simple hypothesis generalize well

Overfitting(p67)

- Def. h overfits the tr. data if exists h’ s.t. h has smaller err than h’ on training, but h’ has smaller err than h over all data

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

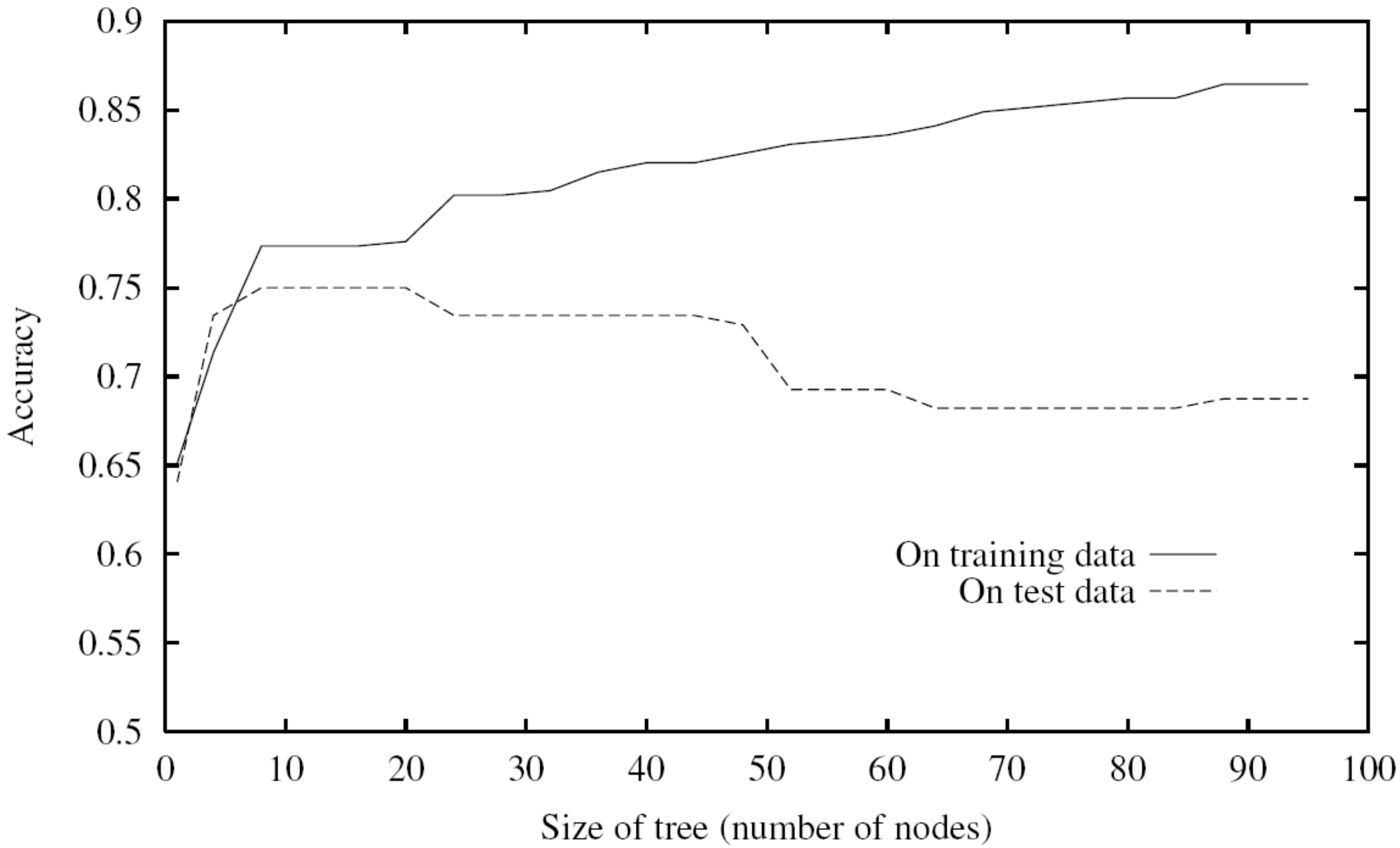
$$error_{train}(h) < error_{train}(h')$$

and

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

Overfitting – ex.

- What is the significance of the intersection point at x = 6?



Avoiding overfitting

- How to avoid overfitting?
  - Stop growing when data split not statistically different
  - Grow full tree and then prune it

- How to select “best” tree?
  - Measure performance over tr. data
  - Measure performance over validation data (test)
    - 65% - training
    - 10% - validation (for pruning)
    - 25% - testing

Pruning

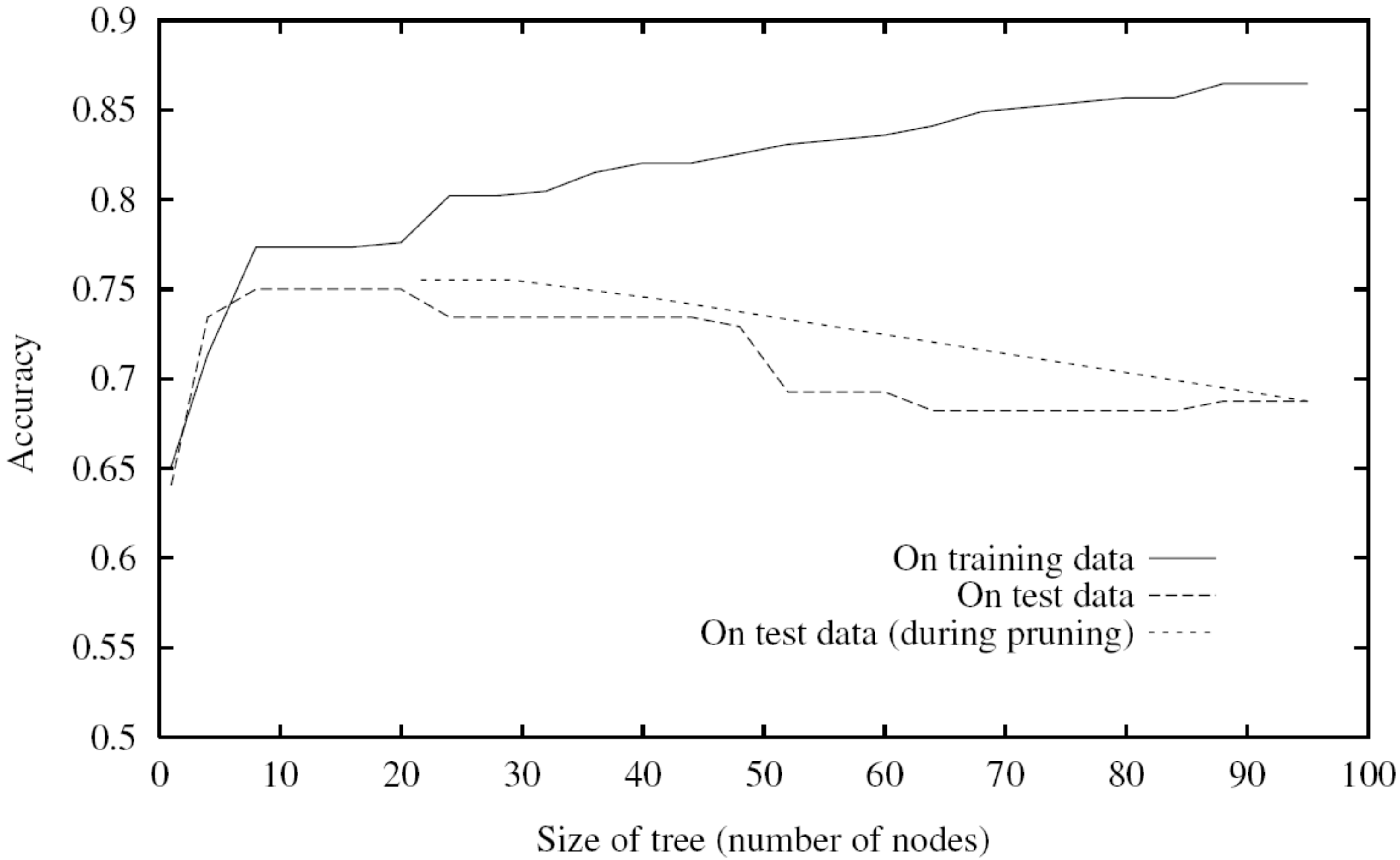
- Def. Removing all descendant nodes of a node and replace the node by a leaf that classifies all of its ex. to have same class as the majority of its ex.

- Prune a node if the tree performs **equally well** or **better** on the validation set

- The tree is pruned bottom-up
  - For each node, keep subtree or change to leaf
  - Choose by comparing estimated err

Effect of pruning

- Trace evolution of pruning starting at right with completely trained tree and moving back to left until additional pruning no longer improves performance on validation set



Rules post-pruning

- Abandon the tree structure and deal only with the rules resulted from the tree

$$\begin{array}{ll} \text{IF} & (Outlook = Sunny) \wedge (Humidity = High) \\ \text{THEN} & PlayTennis = No \end{array}$$

$$\begin{array}{ll} \text{IF} & (Outlook = Sunny) \wedge (Humidity = Normal) \\ \text{THEN} & PlayTennis = Yes \end{array}$$

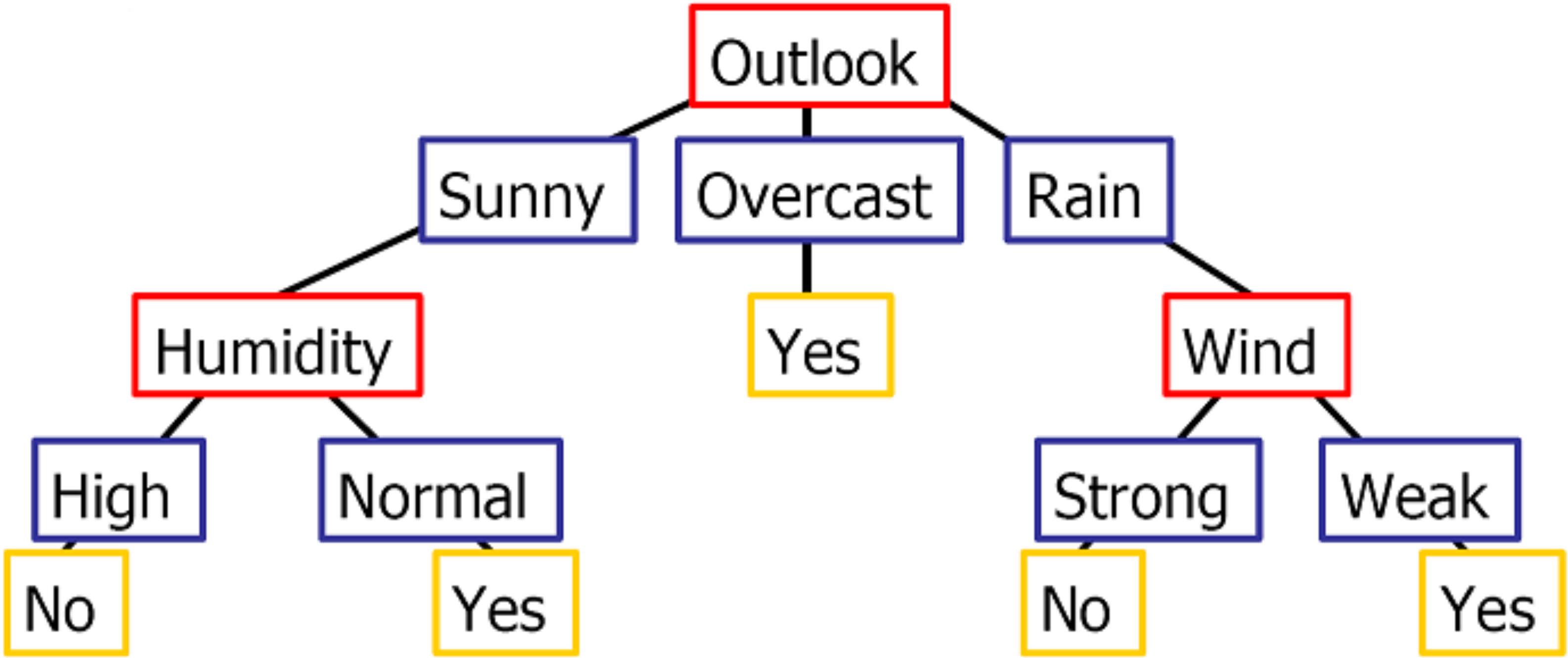
preconditions

postcondition

Rule post-pruning – alg.

- Used more when tr. data is limited
- Used in C4.5:
  1. Obtain tree from training data
  2. Convert tree to if-then rules
  3. Prune each rule **independently** (= remove preconditions if the resulted rule performs better on the validation set)
  4. Sort final rules by their estimated accuracy for use

Converting a tree to rules



R_1: If (Outlook=Sunny) $\wedge$ (Humidity=High) Then PlayTennis=No
R_2: If (Outlook=Sunny) $\wedge$ (Humidity=Normal) Then PlayTennis=Yes
R_3: If (Outlook=Overcast) Then PlayTennis=Yes
R_4: If (Outlook=Rain) $\wedge$ (Wind=Strong) Then PlayTennis=No
R_5: If (Outlook=Rain) $\wedge$ (Wind=Weak) Then PlayTennis=Yes

Continuous valued attributes

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

- Create discrete attribute to test continuous

e.g. (Temperature > 54) = yes

Attributes with many values

- Problem – if attribute has many values, Gain will select it
- E.g. imagine using Date (e.g. Jun_3_1996) as attribute
- ➔ better use GainRatio rather then Gain alone

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

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

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

Unknown attribute values

- Assign most common value of attribute A among the other examples

- Assign most common value of attribute A among the other examples with same label

- Assign probability (used by C4.5)

e.g. if node n contains 6 ex. with A=1 and 4 with A=0 ➔ P(A(x)=1) = 0.6 and P(A(x)=0) = 0.4

Conclusions - decision trees

- Easy to interpret → easy to generate if-then rules
- Robust to noise
- Learn disjunctive hypothesis
- Learn discrete value target function
- When to consider decision trees
  - Target function is discrete value
  - Missing attribute values
  - Possibly noisy data
  - Disjunctive hypothesis may be required
  - Learn discrete value target function