Decision Trees

Introduction

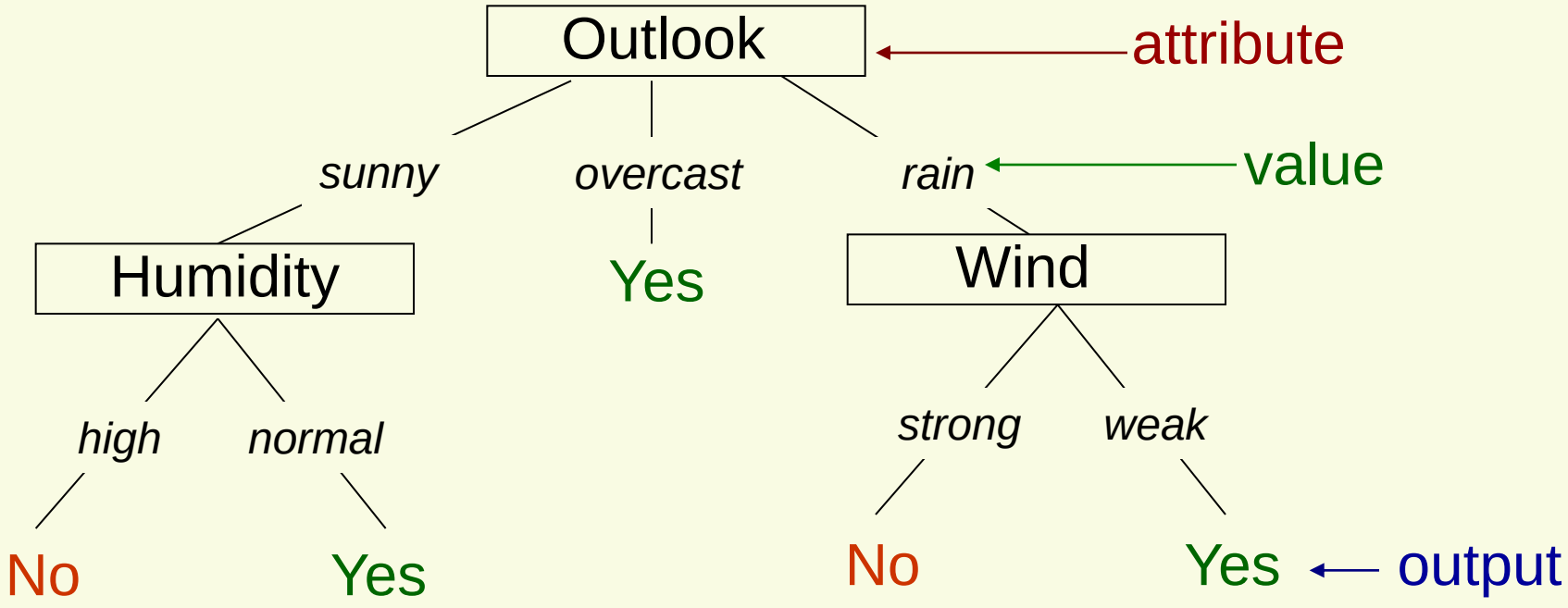
- Intuitive to classify a pattern through sequence of questions.
- Next question depends on the answer to the current question.
- Particularly useful for nonmetric data
- The answers could be yes/no, true/false,

  property $\in$ set_of_values.

Decision Tree Representation

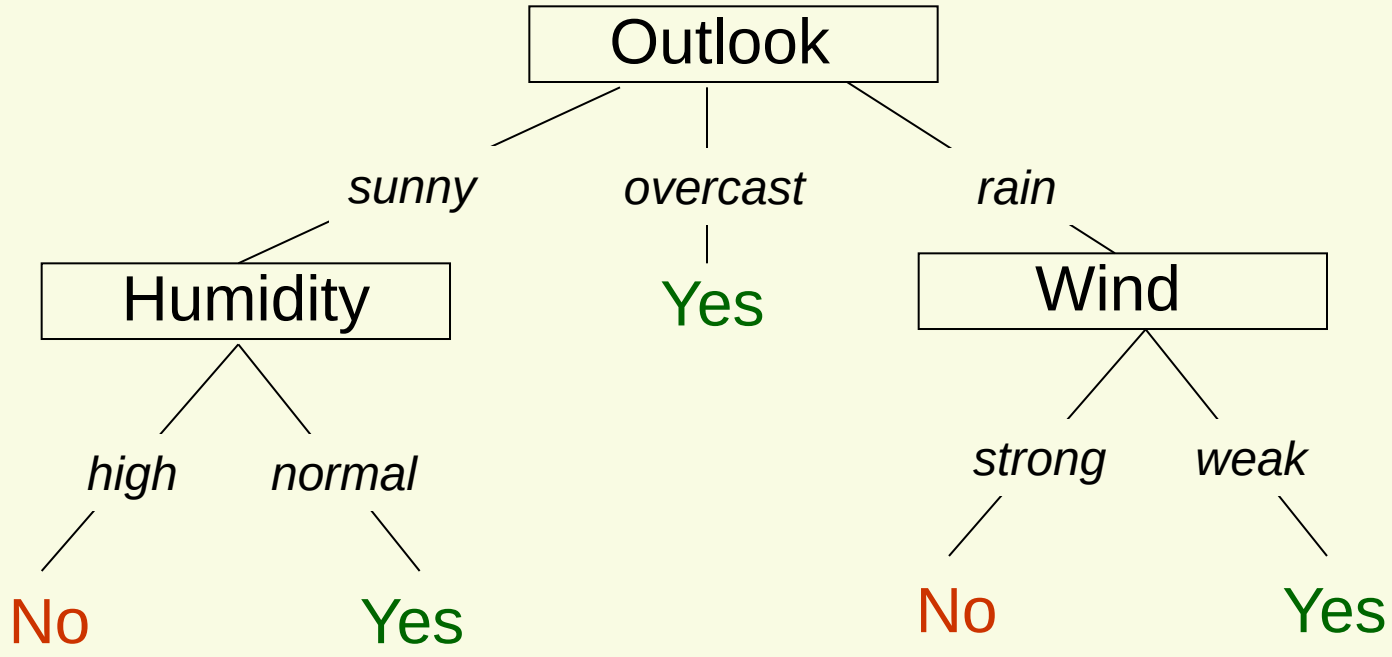
- Decision trees classify **instances** by sorting them down the tree from the **root node** to some **leaf node**, which provides the classification of the instance.
- Each node in the tree specifies a **test** of some **attribute** of the instance, and each **branch** descending from that node corresponds to one of the possible **values** for this attribute.

Decision Tree Representation



Each internal node: test one attribute Xi
Each branch from a node: selects one value for Xi
Each leaf node: predicts Y

Decision Tree Representation: Example



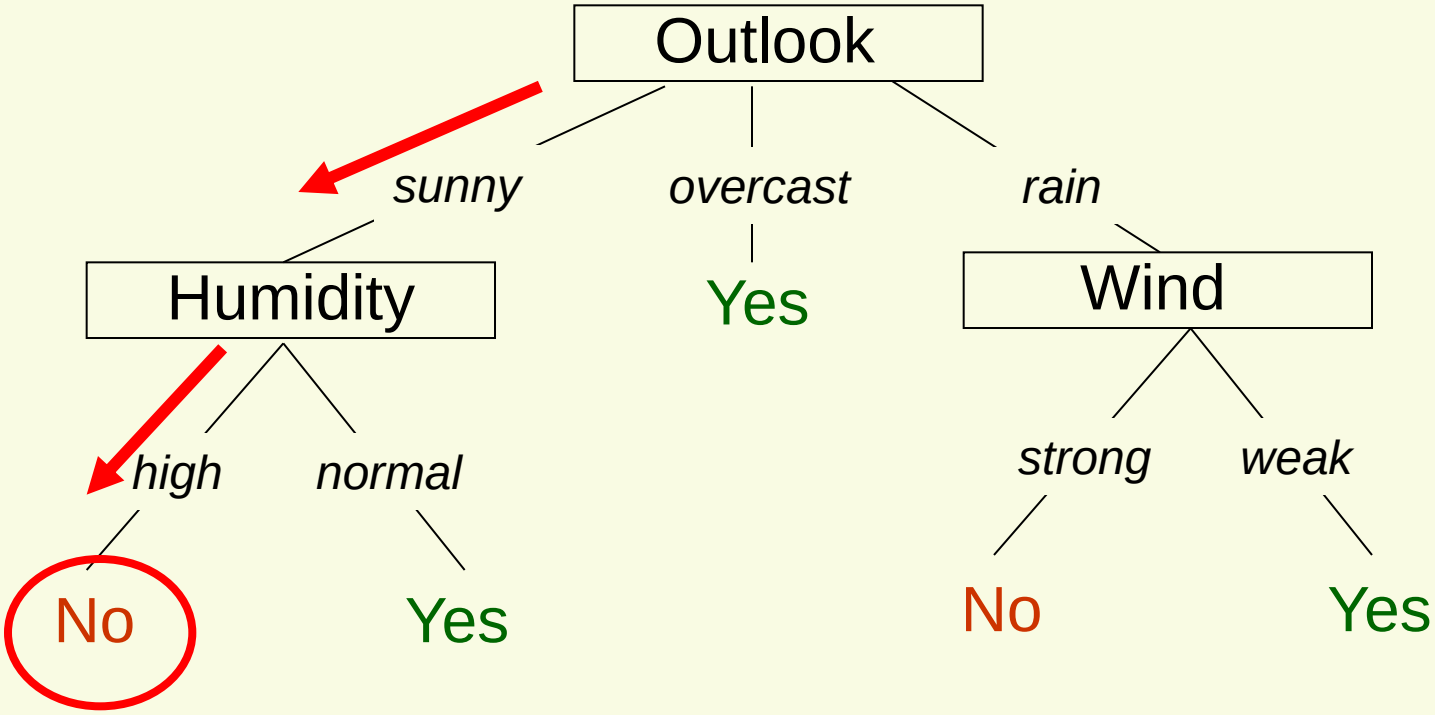
Instance:

(Outlook=Sunny, Temp=Hot, Humidity=High, Wind=Strong)

Classification :

Play tennis: Yes/No

Decision Tree Representation: Example



Instance:

(Outlook=Sunny, Temp=Hot, Humidity=High, Wind=Strong)

Building a Decision Tree

I. First test all attributes and select the one that would function as the “best” root;

II. Main Loop:

1. A← the “best” decision attribute for the 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 node
5. Continue this process until the training examples are perfectly classified

Greedy search for an acceptable decision tree

Attribute selection

Which attribute should be taken next, A1or A2? Which test?

- We seek an attribute that makes the data reaching the immediate descendent nodes as ***pure*** as possible.
- More convenient to define *impurity* of a node.
- Let $i(N)$ define the impurity of a node *N:* in all classes we want $i(N)$ =0 if all the patterns that reach the node belong to the same category, $i(N)$ to be large if all the categories are equally presented.

Entropy Impurity

The most popular measure is *entropy impurity*

$$i(N) = -\sum_{i=1}^{n} P(w_j)\log_2 P(w_j)$$

Fraction of patterns at node A that are in category w_j

Comes from

Entropy $H(X)$ of a random variable X

$$H(X) = -\sum_{i=1}^{n} P(X=i)\log_2 P(X=i)$$

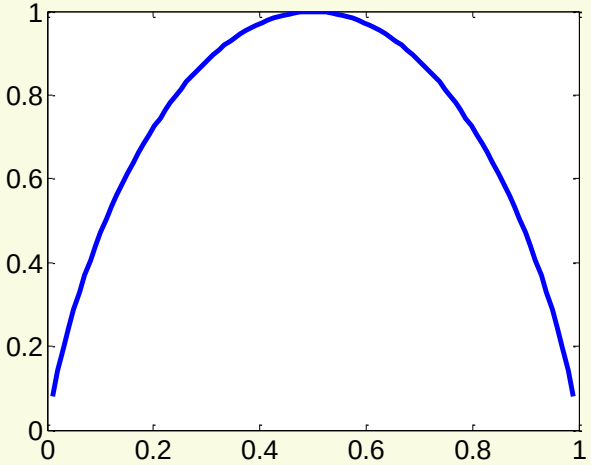
Example of two classes

$$H(X) = -P(w_+)\log P(w_+) - P(w_-)\log P(w_-)$$

$$P(w_+) = 0, P(w_-) = 1 \rightarrow H(X) = 0$$

$$P(w_+) = 0.5, P(w_-) = 0.5 \rightarrow H(X) = 1$$

Plot of H for $P_+ = 1 - P_-$



Information Gain

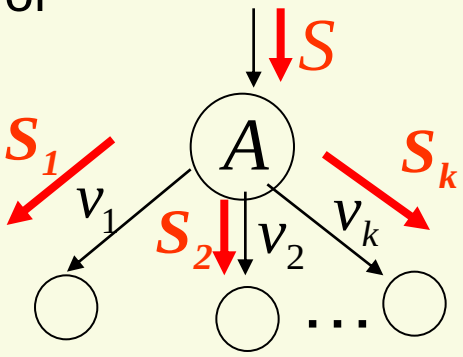
Choose the attribute that most reduces the impurity of the node

Information Gain measured the expected reduction of entropy due to sorting on attribute A

$$Gain(A) = i(N) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} i(N_v)$$

Subset of S for which $A=v$

Patterns that reach the node



Example

Training set:

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

Example

- Let E([X+,Y-]) represent that there are X positive training elements and Y negative elements.
- Therefore the Entropy for the training data, E(S), can be represented as E([9+,5-]) because of the 14 training examples 9 of them are **yes** and 5 of them are **no**.

- Let's start off by calculating the Entropy of the Training Set.

$$E(S) = E([9+,5-]) = (-9/14 \log_2 9/14) + (-5/14 \log_2 5/14) = 0.94$$

Example (cont)

- Next we will need to calculate the information gain G(S,A) for each attribute A where A is taken from the set {Outlook, Temperature, Humidity, Wind}.
- The information gain for Outlook is:
  - G(S,Outlook) = E(S) – [5/14 * E(Outlook=sunny) + 4/14 * E(Outlook = overcast) + 5/14 * E(Outlook=rain)]

    = E([9+,5-]) – [5/14*E(2+,3-) + 4/14*E([4+,0-]) + 5/14*E([3+,2-])]

    = 0.94 – [5/14*0.971 + 4/14*0.0 + 5/14*0.971]

    = 0.246

Example (cont)

- The information gain for Temperature is:

- G(S,Temperature) = 0.94 – [4/14*E(Temperature=hot) + 6/14*E(Temperature=mild) + 4/14*E(Temperature=cool)]
- G(S,Temperature) = 0.94 – [4/14*E([2+,2-]) + 6/14*E([4+,2-]) + 4/14*E([3+,1-])]

  = 0.94 – [4/14 + 6/14*0.918 + 4/14*0.811]

  = 0.029

Example (cont)

- The information gain for Humidity is:
- G(S,Humidity) = 0.94 – [7/14*E(Humidity=high) + 7/14*E(Humidity=normal)]

  = 0.94 – [7/14*E([3+,4-]) + 7/14*E([6+,1-])]

  = 0.94 – [7/14*0.985 + 7/14*0.592]

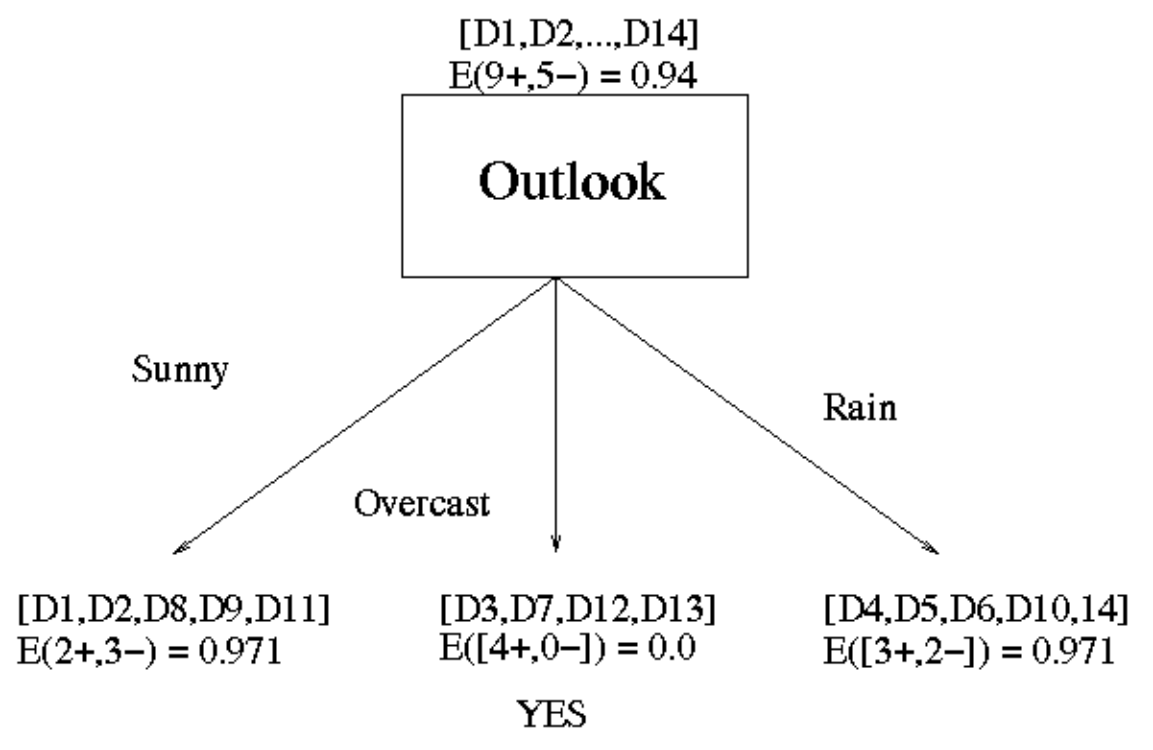
  = 0.1515

- The information gain for Wind is:
- G(S,Wind) = 0.94 – [8/14*0.811 + 6/14*1.00]

  = 0.048

Example (cont)

- Outlook is our winner!



Example (cont)

- Now that we have discovered the root of our decision tree we must now recursively find the nodes that should go below Sunny, Overcast, and Rain.

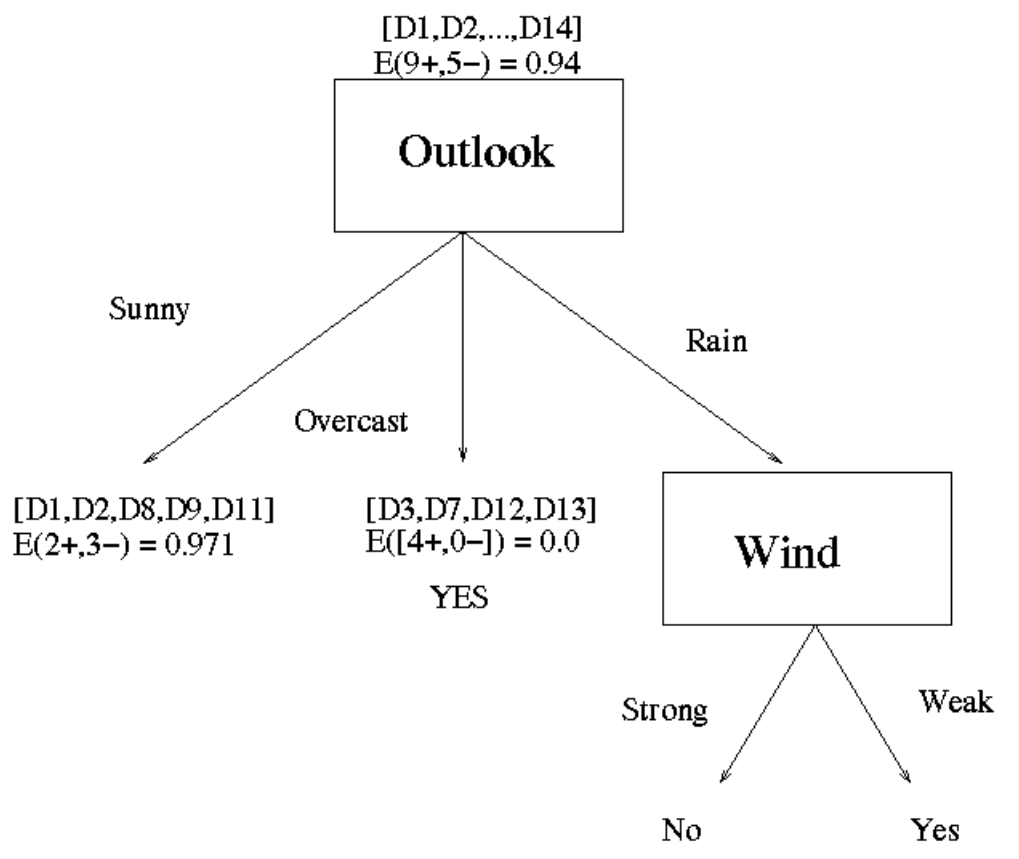
- G(Outlook=Rain, Humidity) = 0.971 – [2/5*E(Outlook=Rain ^ Humidity=high) + 3/5*E(Outlook=Rain ^Humidity=normal] **= 0.02**

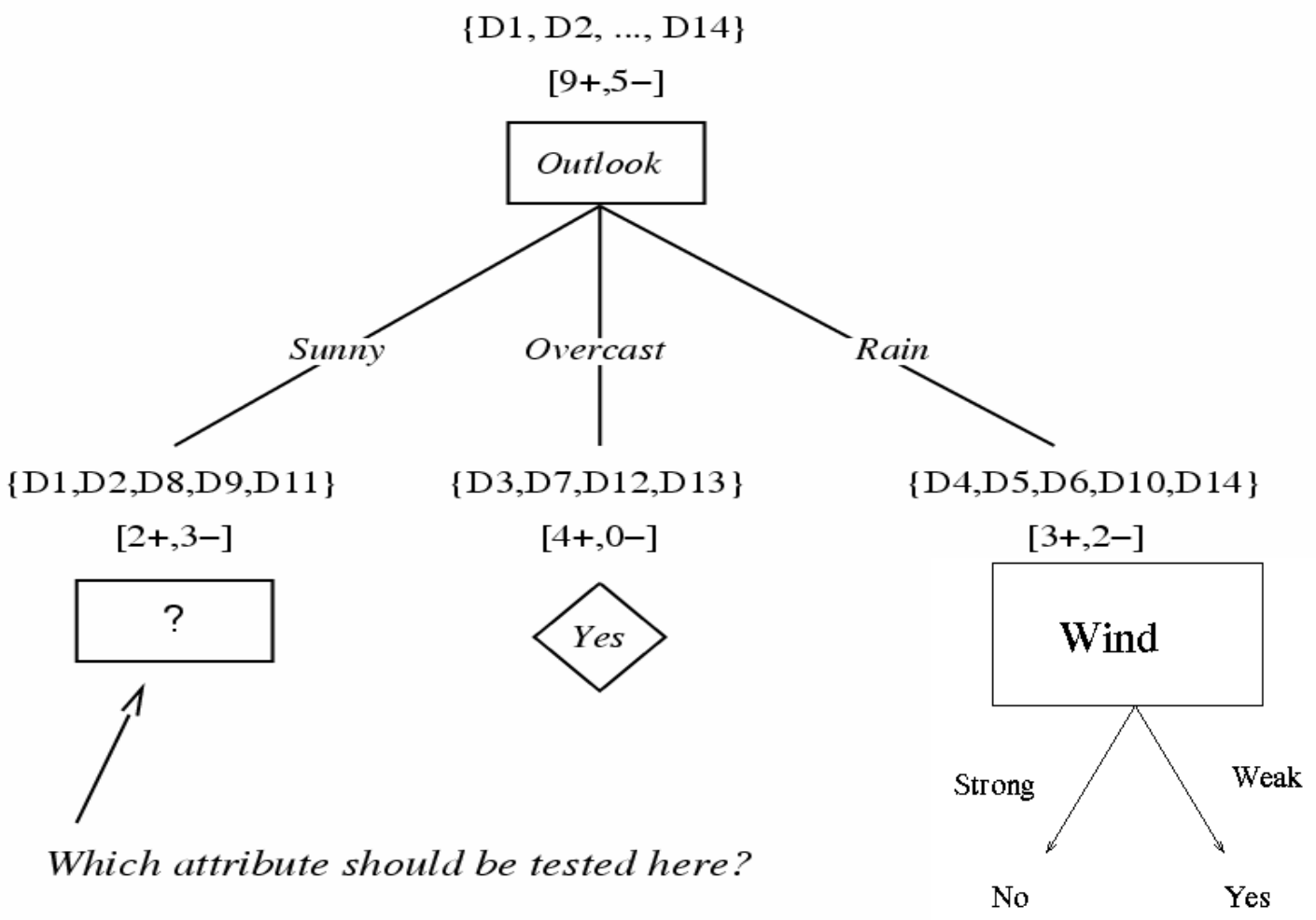
- G(Outlook=Rain,Wind) = 0.971- [3/5*0 + 2/5*0]

  = 0.971

Example (cont)

- Now our decision tree looks like:



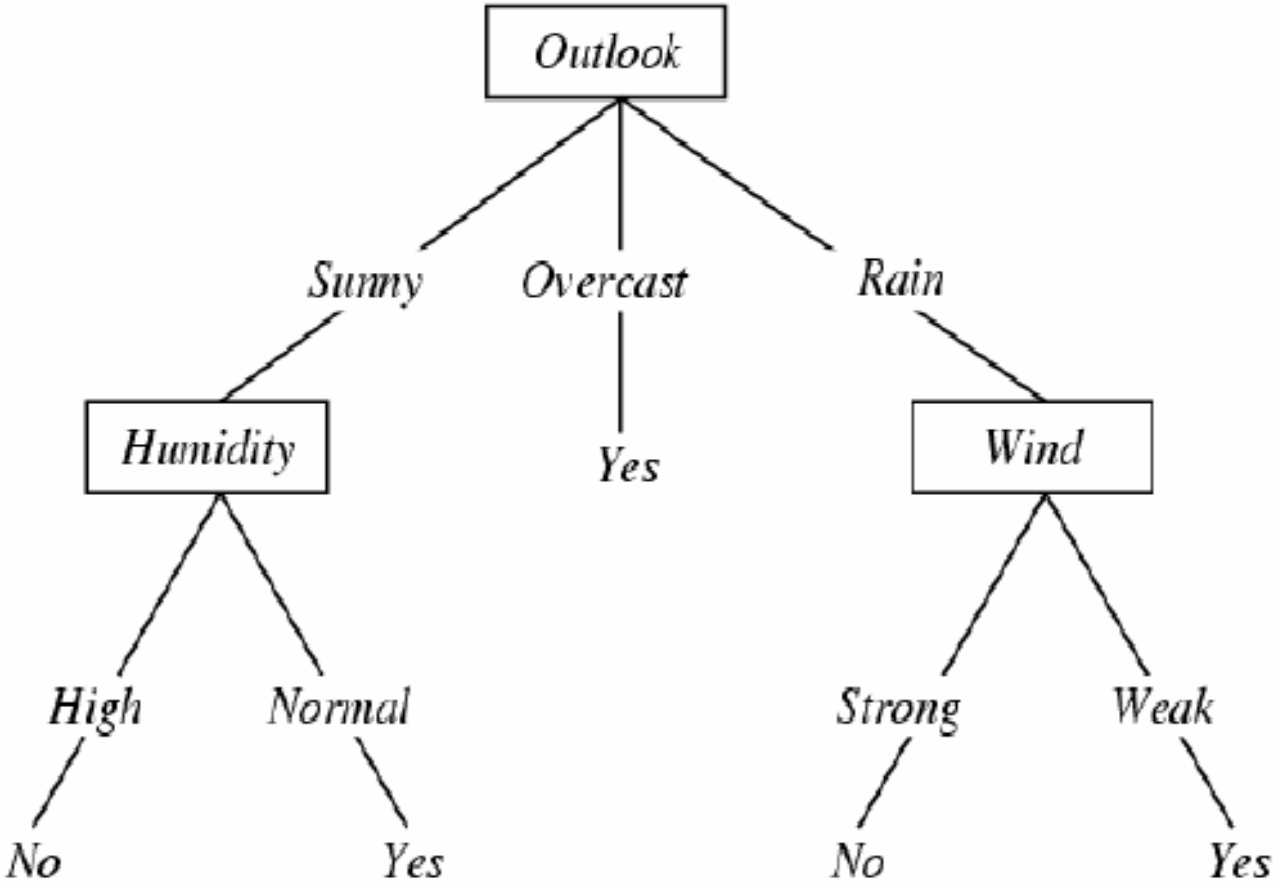


S_{sunny} = {D1,D2,D8,D9,D11}

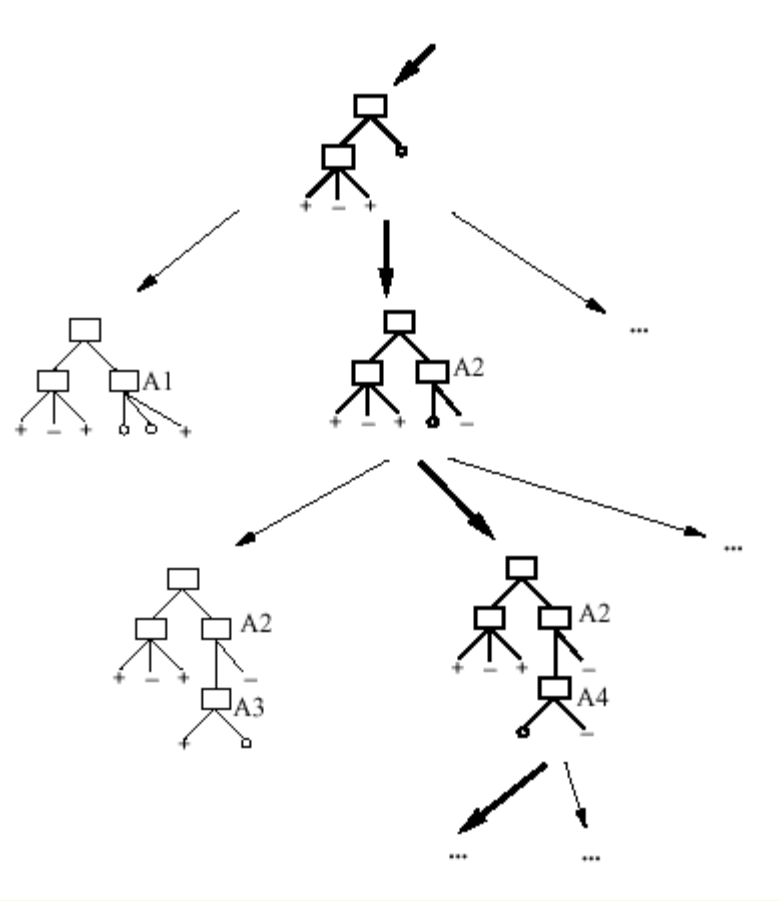
Gain (S_{sunny}, *Humidity*) = .970 − (3/5) 0.0 − (2/5) 0.0 = .970

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


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

Which Tree Should We Output?



- ID3 performs heuristic search through space of decision trees from simplest to increasingly complex, guided by the information gain.
- It stops at smallest acceptable tree.

Occam's razor: prefer the simplest hypothesis that fits the data

Ockham's razor

Why simple trees should be preferred?

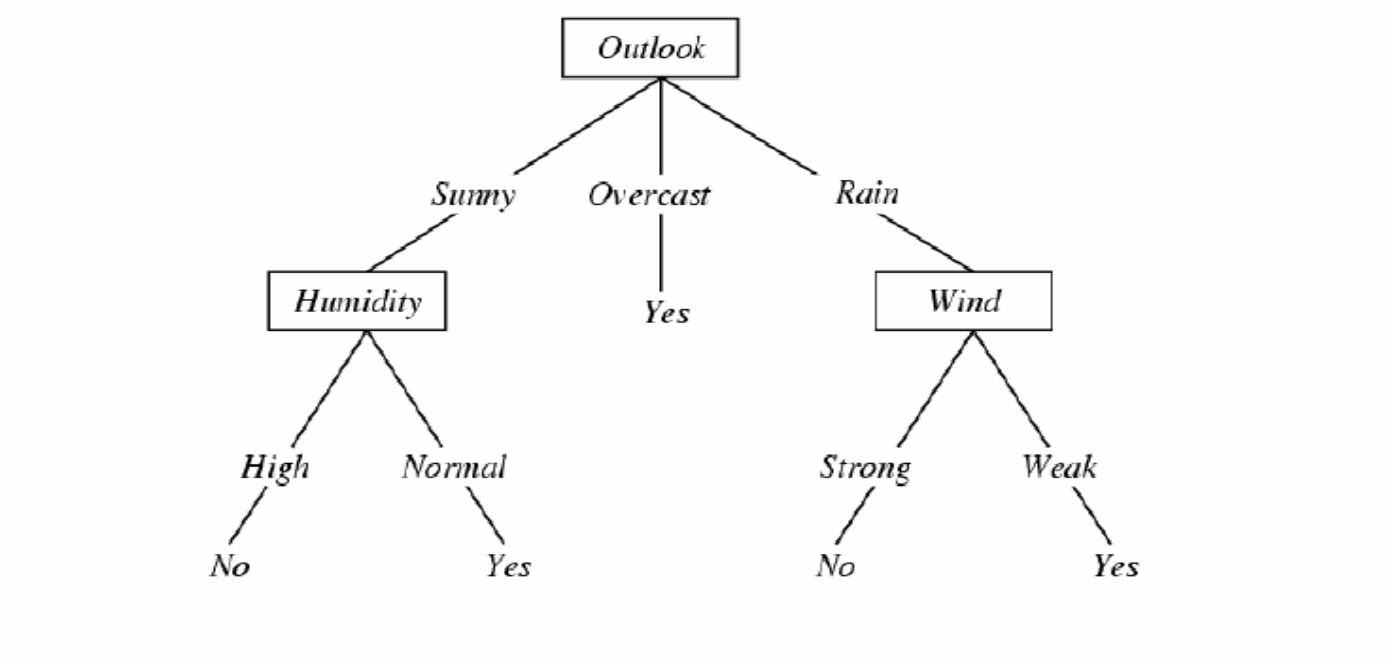
1. The number of simple hypothesis that may accidentally fit the data is small, so chances that simple hypothesis uncover some interesting knowledge about the data are larger.
2. Simpler trees do not partition the feature space into too many small boxes, and thus may generalize better, while complex trees may end up with a separate box for each training data sample.

Overfitting in DT

- Consider adding noisy example

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

- How it effects the earlier tree?



- Noise in data or small number of training examples lead to overfitting

Avoiding overfitting

How to avoid overfitting?

- Stop growing earlier, before it perfectly classifies the training data.
- Grow full tree, then post-prune the tree.

How to select “best” tree:

- Measure performance over separate validation data set.
- Trade complexity for accuracy: minimize

size(tree)+size(misclassifications(tree))

- …

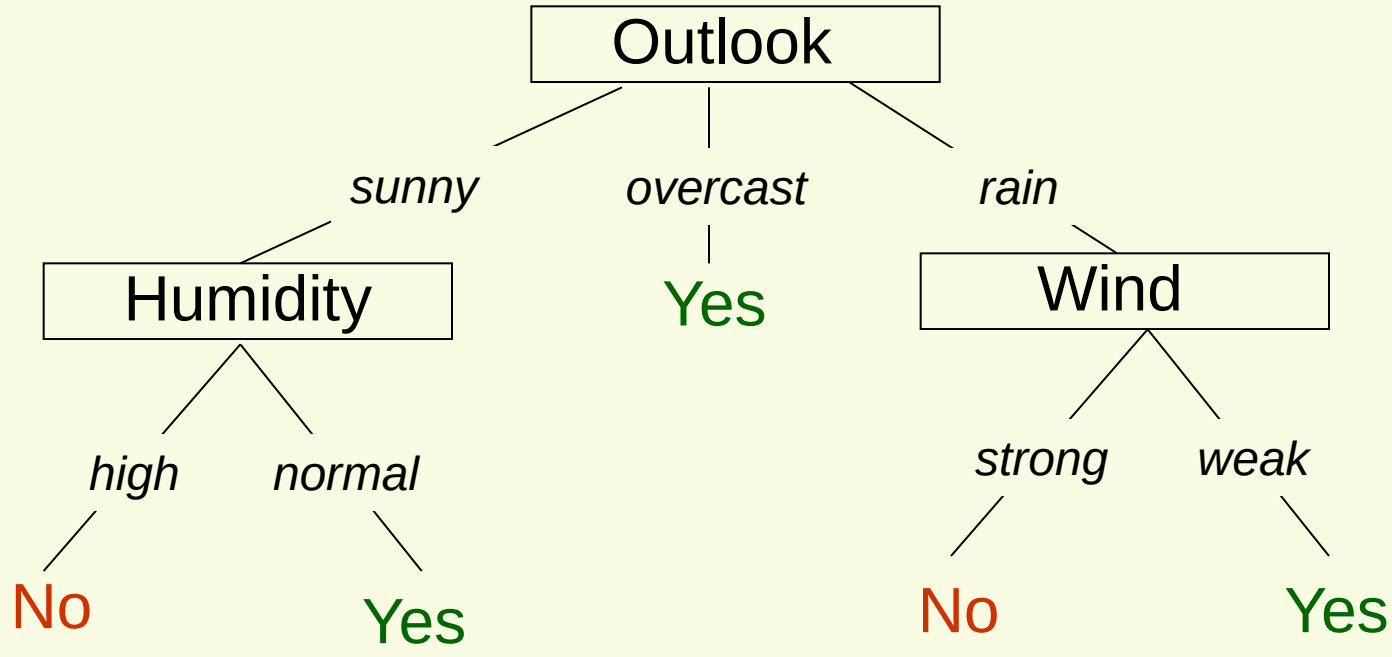
DT Pruning

- One way to deal with overfitting in decision trees is to ***prune*** the tree
- This means to make it smaller by removing nodes that don't turn out to be helpful
- Use a greedy hill-climbing search to prune a tree:
  - for each interior (non-leaf) node *n*
    - remove the subtree at *n*
    - replace it with a leaf node marked + or -, according to the majority of the training examples that fall into that subtree.
    - evaluate the performance of the new decision tree on the *validation set*
    - store the pruned tree with the best performance
  - repeat until no improvement is made from removing any interior node
  - if a pruned tree has the same performance as the current tree, then use the pruned tree.

Rule Post-Pruning (used in C4.5)

- Grow the tree until the training data is fit as well as possible.
- Convert the learned tree into an equivalent set of rules, by creating one rule per each path.
- Eliminate unnecessary rule antecedents to simplify the rules.
  - Rules with only one antecedent cannot be further simplified, so we only consider those with two or more.
  - To simplify a rule, eliminate antecedents that have no effect on the conclusion reached by the rule.
- Sort final rules into desired sequence for use.

Converting Tree to Rules



IF (*Outlook=Sunny)* AND *(Humidity=High*)

THEN *PlayTennis=No*

IF (*Outlook=Sunny)* AND *(Humidity=Normal*)

THEN *PlayTennis=Yes*