Introductory Applied Machine Learning

Decision Trees

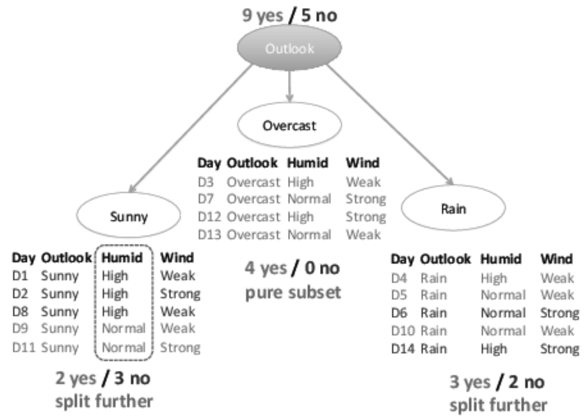
Victor Lavrenko and Nigel Goddard
School of Informatics

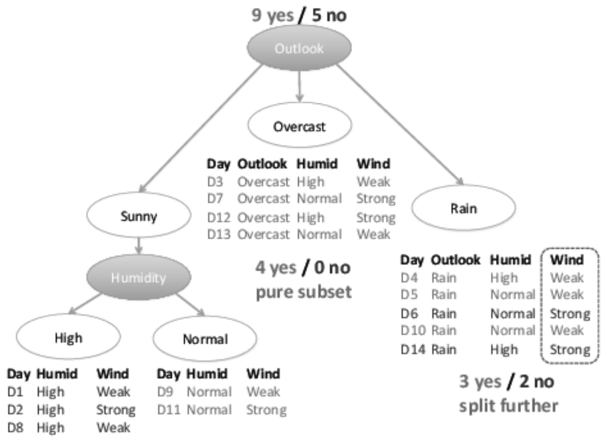
Predict if John will play tennis

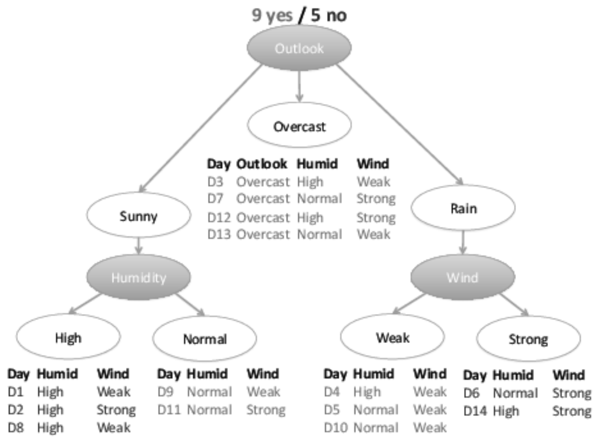
- Hard to guess
- Try to *understand* when John plays
- Divide & conquer:
  - split into subsets
  - are they pure? (all yes or all no)
  - if yes: stop
  - if not: repeat
- See which subset new data falls into

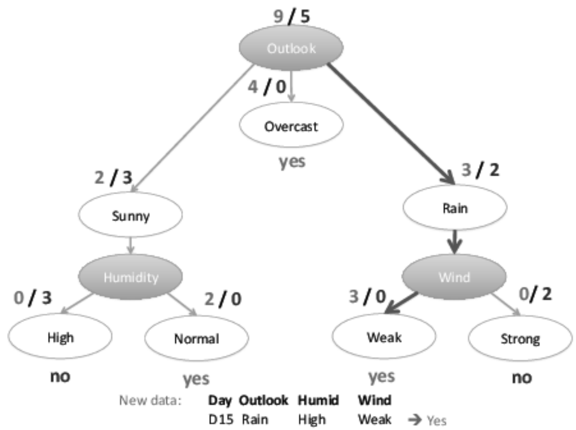
Training examples: 9 yes / 5 no

Day	Outlook	Humidity	Wind	Play
D1	Sunny	High	Weak	No
D2	Sunny	High	Strong	No
D3	Overcast	High	Weak	Yes
D4	Rain	High	Weak	Yes
D5	Rain	Normal	Weak	Yes
D6	Rain	Normal	Strong	No
D7	Overcast	Normal	Strong	Yes
D8	Sunny	High	Weak	No
D9	Sunny	Normal	Weak	Yes
D10	Rain	Normal	Weak	Yes
D11	Sunny	Normal	Strong	Yes
D12	Overcast	High	Strong	Yes
D13	Overcast	Normal	Weak	Yes
D14	Rain	High	Strong	No
New data:				
D15	Rain	High	Weak	?









ID3 algorithm

- Split (node, {examples}):
  1. A ← the $\boxed{\text{best attribute}}$ for splitting the {examples}
  2. Decision attribute for this node ← A
  3. For each value of A, create new child node
  4. Split training {examples} to child nodes
  5. For each child node / subset:
     - if subset is pure: STOP
     - else: Split (child_node, {subset})
- Ross Quinlan (ID3: 1986), (C4.5: 1993)
- Breimanetal (CaRT: 1984) from statistics

Which attribute to split on?

- Want to measure "purity" of the split
  - more certain about Yes/No after the split
    - pure set (4 yes / 0 no) => completely certain (100%)
    - impure (3 yes / 3 no) => completely uncertain (50%)
  - can't use P("yes" | set):
    - must be symmetric: 4 yes / 0 no as pure as 0 yes / 4 no

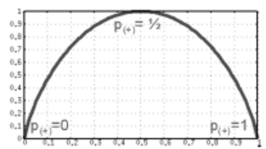
Entropy

- Entropy: $H(S) = - p_{(+)} \log_2 p_{(+)} - p_{(-)} \log_2 p_{(-)}$ bits
  - S … subset of training examples
  - $p_{(+)}$ / $p_{(-)}$ … % of positive / negative examples in S
- Interpretation: assume item X belongs to S
  - how many bits need to tell if X positive or negative
- impure (3 yes / 3 no):

$$H(S) = -\frac{3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6} = 1 \text{ bits}$$

- pure set (4 yes / 0 no):

$$H(S) = -\frac{4}{4}\log_2\frac{4}{4} - \frac{0}{4}\log_2\frac{0}{4} = 0 \text{ bits}$$



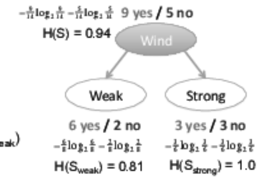
Information Gain

- Want many items in pure sets
- Expected drop in entropy after split:

$$Gain(S,A) = H(S) - \sum_{V \in Values(A)} \frac{|S_V|}{|S|} H(S_V)$$

V … possible values of A
S … set of examples {X}
S_V … subset where X_A = V

- Mutual Information
  - between attribute A and class labels of S



Gain (S, Wind)
$= H(S) - {}^{8}/_{14} H(S_{weak}) - {}^{6}/_{14} H(S_{weak})$
$= 0.94 - {}^{8}/_{14} * 0.81 - {}^{6}/_{14} * 1.0$
$= 0.049$

Overfitting in Decision Trees

- Can always classify training examples perfectly
  - keep splitting until each node contains 1 example
  - singleton = pure
- Doesn't work on new data

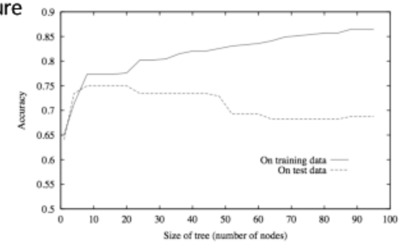


Figure credit: Tom Mitchell, 1997

Avoid overfitting

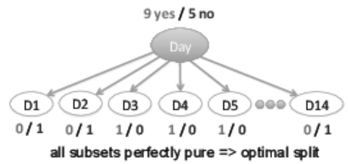
- Stop splitting when not statistically significant
- Grow, then post-prune
  - based on validation set
- Sub-tree replacement pruning (WF 6.1)
  - for each node:
    - pretend remove node + all children from the tree
    - measure performance on validation set
  - remove node that results in greatest improvement
  - repeat until further pruning is harmful

General Structure

- **Task:** classification, discriminative
- **Model structure:** decision tree
- **Score function**
  - information gain at each node
  - preference for short trees
  - preference for high-gain attributes near the root
- **Optimization / search** method
  - greedy search from simple to complex
  - guided by information gain
- Book: sections 3.2, 3.3, 4.3

Problems with Information Gain

- Biased towards attributes with many values
- Won't work for new data: D15 Rain High Weak



- Use GainRatio:

$$SplitEntropy(S,A) = -\sum_{V \in Values(A)} \frac{|S_V|}{|S|} \log \frac{|S_V|}{|S|}$$

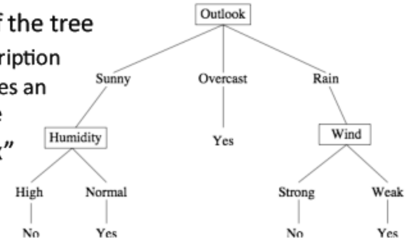
A … candidate attribute
V … possible values of A
S … set of examples {X}
S_V … subset where X_A = V

$$GainRatio(S,A) = \frac{Gain(S,A)}{SplitEntropy(S,A)}$$

penalizes attributes with many values

Trees are interpretable

- Read rules off the tree
  - concise description of what makes an item positive
- No "black box"
  - important for users



Rule: (Outlook = Overcast) ∨
(Outlook = Rain ∧ Wind = Weak) ∨
(Outlook = Sunny ∧ Humidity = Normal)
logical formula in DNF (disjunctive normal form)

Figure credit: Tom Mitchell, 1997

Continuous Attributes

- Dealing with continuous-valued attributes:
  - create a split: (Temperature > 72.3) = True,False
- Threshold can be optimized (WF 6.1)

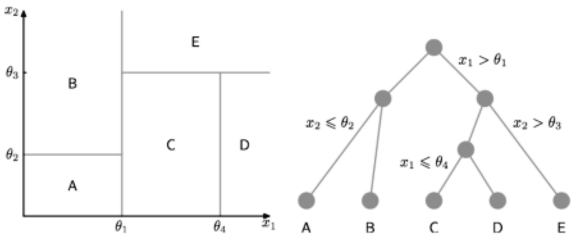


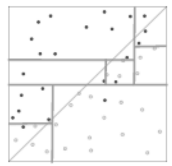
Figure credit: Chris Bishop, PRML

Multi-class and Regression

- Multi-class classification:
  - predict most frequent class in the subset
  - entropy: $H(S) = - \Sigma_c \, p_{(c)} \log_2 p_{(c)}$
  - $p_{(c)}$ … % of examples of class c in S
- Regression:
  - predicted output = average of the training examples in the subset
  - requires a different definition of entropy
  - can use linear regression at the leaves (WF 6.5)

Pros and Cons

- Pros:
  - interpretable: humans can understand decisions
  - easily handles irrelevant attributes (Gain = 0)
  - can handle missing data (WF 6.1)
  - very compact: #nodes << D after pruning
  - very fast at testing time: O(depth)
- Cons:
  - only axis-aligned splits of data
  - greedy (may not find best tree)
    - exponentially many possible trees



Random Decision Forest

- Grow *K* different decision trees:
  - pick a random subset S_r of training examples
  - grow a full ID3 tree T_r (no prunning):
    - when splitting: pick from $d << D$ random attributes
    - compute gain based on S_r instead of full set
  - repeat for $r = 1 \dots K$
- Given a new data point *X*:
  - classify *X* using each of the trees $T_1 \dots T_K$
  - use majority vote: class predicted most often
- State-of-the-art performance in many domains

Summary

- ID3: grows decision tree from the root down
  - greedily selects next best attribute (using Gain)
  - entropy: how uncertain we are of Yes/No in a set
  - Gain: reduction in uncertainty following a split
- Searches a complete hypothesis space
  - prefers smaller trees, high gain at the root
- Overfitting addressed by post-pruning
  - prune nodes, while accuracy ⇧ on validation set
- Fast, compact, interpretable