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

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Humidity</th>
<th>Wind</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sunny</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Overcast</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D3</td>
<td>Rain</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Overcast</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Normal</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D6</td>
<td>Overcast</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D9</td>
<td>Rain</td>
<td>Normal</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D10</td>
<td>Overcast</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D11</td>
<td>Overcast</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
</tbody>
</table>

New data:  
D13 Rain High Weak ?

ID3 algorithm

1. A ∈ the 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)  
    - impure (3 yes / 3 no)
  - can’t use P(“yes” | set)  
  - must be symmetric: 4 yes / 0 no as pure as 0 yes / 4 no

Entropy

- Entropy:  
  \[ H(S) = - p_{yes} \log_2 p_{yes} - p_{no} \log_2 p_{no} \]
  - S: subset of training examples
  - p_{yes}, p_{no}: % 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):
    \[ H(S) = - \frac{3}{6} \log_2 \frac{3}{6} - \frac{3}{6} \log_2 \frac{3}{6} = -\frac{1}{2} \log_2 \frac{1}{2} = 1 \text{ bits} \]
  - 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:
  \[ \text{Gain}(S, A) = H(S) - \sum_{v \in \text{possible values of } A} \frac{|S_v|}{|S|} \cdot H(S_v) \]
- Mutual Information
  - between attribute A and class labels of S
    \[ \text{Gain}(S, \text{Rain}) = H(S) - \sum_{v \in \text{possible values of } A} \frac{|S_v|}{|S|} \cdot H(S_v) \]
  - 4 yes / 2 no
  - 3 yes / 3 no
  - 2 yes / 4 no
  - 1 yes / 6 no
  - 6 yes / 5 no
  \[ H(S) = 0.46 \quad H(S_{\text{yes}}) = 0.7 \quad H(S_{\text{no}}) = 1 \]
  \[ \text{Gain}(S, \text{Rain}) = 0.46 - (0.46 \times 0.7 + 0.46 \times 1) = 0.046 \]

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

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: DIS Rain High Weak
- Use GainRatio:
  \[ \text{GainRatio}(S, A) = \frac{\text{Gain}(S, A)}{\text{SplitEntropy}(S, A)} \]
  - 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

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

Multi-class and Regression
- Multi-class classification:
  - predict most frequent class in the subset
  - entropy: \( H(S) = -\sum_{c} p_{c} \log_2 p_{c} \)
  - \( p_{c} \) = % 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: \( O(d) \)
- Cons:
  - only axis-aligned splits of data
  - greedy (may not find best tree)
  - exponentially many possible tree
Random Decision Forest

- Grow $K$ different decision trees:
  - pick a random subset $S_i$ of training examples
  - grow a full ID3 tree $T_i$ (no pruning):
    - when splitting: pick from $d << D$ random attributes
    - compute gain based on $S_i$ instead of full set
    - repeat for $r = 1 \ldots K$
- Given a new data point $X$:
  - classify $X$ using each of the trees $T_1 \ldots T_K$
  - 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 $\uparrow$ on validation set
- Fast, compact, interpretable