Historical

- Ross Quinlan, since late 1970s
- Basic ideas in ID3: [Quinlan, 1986]
- Refined to become C4.5 [Quinlan, 1993]
- C5.0 a commercial (dark!) product
Rule list vs. tree representation

- Consider rules: paths to leaves
- Distinguishes tests' "contexts"
Information gain

\[ H(T) = \sum_i -p_i \log_2 p_i \quad \text{where } p_i = \frac{|C_i|}{|T|} \]

\[ \text{InfoGain}(T,A) = H(T) - \sum_{v \in A} \frac{|T_v|}{|T|} H(T_v) \]
Compensating for ‘broad’ attributes

• InfoGain has bias towards attributes with many values

\[ \text{SplitInfo}(T, A) \equiv - \sum_{v \in A} \frac{|T_v|}{|T|} \log_2 \frac{|T_v|}{|T|} \]

• H(T,Attribute) vs. H(T,Classification)

• Penalizes high-cardinality attributes
  • Eg: full “shattering” of \( n \) instances: SplitInfo = \( \log_2 n \)
  • Eg: Binary variable that divides evenly: SplitInfo = 1

\[ \text{GainRatio}(T, A) \equiv \frac{\text{InfoGain}(T, A)}{\text{SplitInfo}(T, A)} \]
Missing values

- Treat as its own pseudo-value
- Replace with most typical (frequent)
  - Across entire Train set
  - wrt/ subset of instances classified $C_i$ by node
- Consider all “fractional” probabilities
  - InfoGain only needs probabilities
  - Propagates down branches
  - Also useful for classification of unknown variables after training complete
**DecisionTree search**

- Complete: all sets of discrete \( (A=v) \) hypotheses representable as decision trees
- Single hypothesis considered
- Simple -> complex construction order
- Gradient search, InfoGain as objective function
- No backtracking (in ID3)
- **Batch learner**
  - Statistical use of entire training ensemble, vs. incremental changes to each instance
- Sensitive to noise, overfitting
### Complexity

- $n$ instances, $m$ attributes
- All $n$ instances at every depth
- Assume tree has $\log n$ depth
- Assume attributes considered at each node
- $O(mn \log n)$
## Estimating accuracy (C4.5)

- When Test data not available?!  
- Consider node’s **majority** class as potential leaf  
- Error: all other instances  
- Compare to **binomial** (?!?) distribution which generates same number of errors  
- Pessimistically use (one-tailed) upper confidence interval  
  - vs. typical symmetric confidence range  
  - *[W&F, p. 165]*
Pruning

- Reduced error pruning
- Rule post-pruning (C4.5)
Reduced error pruning

- Replace subtree with leaf
- Associate most common class to it
- Remove node which maximizes accuracy on Test
Rule post-pruning (C4.5)

- Consider rules (paths to leaves) vs. tree
- Test all generalizations of all rules
- Keep any that improve accuracy
- Consider rules in order of their accuracy!
Continuos variables

- Discretize by picking threshold
- Sort instances according to continuous variable
- Look for changes in class labels

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>24</td>
<td>34</td>
<td>35</td>
<td>37</td>
<td>41</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>+</td>
<td>-</td>
<td>+</td>
<td>-</td>
</tr>
</tbody>
</table>

t=29  t=39
WEKA/J48

- Default confidence = 0.25
- Minimum instances/leaf = 2
- Reduced-error pruning
- Subtree raising?
- Attribute order can matter
- Tree output
Attribute order can matter

From: "Eibe Frank" <eibe@cs.waikato.ac.nz>
To: <wekalist@list.scms.waikato.ac.nz>
Subject: RE: [Wekalist] j48 and attribute ordering
Date: Sun, 5 Aug 2001 13:53:44 +1200

> From tpederse@d.umn.edu Sun Aug 05 04:45:03 2001
> The only possible explanation I have come up with is
> that perhaps a "tie-breaking" mechanism in J48 is
> based on the order in which attributes are defined. In
> other words, if two attributes are considered equally
> good at partitioning the training data at a particular
> point in the learning process, then is the one that is
> selected for inclusion in the tree based on its "position"
> in the arff file?

That's correct. If several attributes are considered equally good by the selection criterion, the one that is listed first in the arff file will be chosen. The same thing happens in the original C version of C4.5.

Cheers,
Eibe
FEATURE = value: class (correct/incorrect)