ID3 Bias: Prefer Short Hypotheses

but why?

Occam's Razor: Prefer the simplest hypothesis

Argument in favor:

- There are fewer short hypotheses than long
- A short hypothesis that fits training data is unlikely to be coincidence
- A long hypothesis that fits data might be

Argument opposed:

- There are many small sets of hypotheses
- E.g., All trees that use attributes beginning with "M"
- What's special about small sets that are short?

Some Difficulties with DT Learning

- When should we stop growing the tree?
  - to avoid overfitting the training data
- What about attributes with continuous values?
  - e.g., temperature. T > Const
- What about discrete attributes with many values?
  - e.g., Birthday
- What if attribute values are sometimes unknown?
  - e.g., CatScanResult
- What if attributes have different costs?
  - e.g., Bruised? vs. CatScanResult

Overfitting

A decision tree, d, is said to *overfit* the training data if there exists some tree d', such that d has smaller error than d' over the training examples, but d' has a smaller error than d over the entire distribution of instances.

How can this happen?

- Noisy training data
- Even noise-free data can be misleading

Approaches to Avoid Overfitting

When to address it:

- Change termination criterion, to stop earlier
- Allow tree to overfit, then post-prune

What criterion to use:

- Optimize performance on separate test data
- Include only statistically significant tree splits
- Minimize (Size[tree] + Size[misclassified examples])

Train and Test Approach

Train on some data, test on other

(doesn't involve assumptions about distributions)

Reduced error pruning:

1. Build decision tree from training set
2. Consider each node (subtree) a pruning candidate
3. Greedily prune nodes that gain information on *test* set

Finds smallest tree among those most accurate on test set

(Why not just prune using the training set?)

Estimating Error Rate

How can we estimate future performance of h?

(performance on training data gives optimistic estimate)

Train and test estimation:

- Divide available examples into training set, test set
- Train on training set
- Measure error rate on test set

How well will measured error estimate *true* error?

Estimating Error Rates

(Assuming boolean concept, independent examples)

Given n samples, e errors

- *Observed* error rate, r = e/n
- What does this tell us about *true* error rate?

Estimating proportions from sample of size n:

- sd = SQRT[r(1-r) / n]
- 95% confidence interval = 1.96 sd

With 95% confidence, the *true* error rate is r plus or minus 1.96 SQRT[r(1-r) / n]

P% confidence interval:

- With probability P, true error rate is within this interval

Rule Post-Pruning

1. Infer a decision tree
2. Convert tree to rules, one per leaf node
3. Prune (generalize) each rule:
    - Remove preconditions that improve estimated accuracy
4. Filter out poor rules, sort remaining into optimal sequence

Why rules?

- Allows pruning a test for one rule, but retaining for another
- Avoids distinction between tests near root vs. near leaves
- More understandable to humans?

Rule Post-Pruning in C4.5

Assuming data is scarce, can't afford train and test, so

1. Train on all available data
2. Obtain rule observed-error-rate over all data
3. Estimate rule's true-error-rate to be
   - Observed-error-rate + standard-deviation

Suspicious from a theoretical statistics viewpoint (pessimistic, biased estimate)

Successful in practice, according to [Quinlan 93]

Continuous-Valued Attributes

Problem: need a descendant per value...

Solution: generate features dynamically (Temperature > C)

Temperature:	40	48	60	72	80	90
Play-Tennis?:	No	No	Yes	Yes	Yes	No

Which values for C need be considered?

See also learning Oblique Decision Trees [Murthy 94]

Other Measures for Selecting Attributes

Problem: Information Gain favors attributes with many values

Approaches:

- Create boolean features: A = Apr-26-1993
- Replace *inf gain* by different selection measure

Gain Ratio

see C4.5 [Quinlan 93]

Intuition: penalize broad, uniform splits

$$Gain\text{–}Ratio(S,A) = Gain(S,A)/Split\text{–}Inf(S,A)$$

where

$$Split\text{–}Inf(S,A) = -\sum_{i=1}^{n} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

here S_i is subset of S with $A = value_i$

n completely partitioned examples $\rightarrow$ $Split\text{–}Inf = log_2 n$

n examples split in half $\rightarrow$ $Split\text{–}Inf = 1$

Missing Attributes

Suppose sometimes missing attribute (e.g., *Cat-Scan-Result*)

Approach: Assign a value for the missing attribute

- Based on most common value of examples at that tree node
- Or assign a *distribution*, and sort example fractionally through tree

Different Attribute Costs

If different costs, prefer trees with low cost, high accuracy

Heuristic: add a cost term to selection measure

- E.g., [Information-Gain^2/Cost]
- Used in medical domains [Nunez91], Robotic perception [Tan90]

Decision Trees: Summary

- Practical method for learning discrete-valued functions, from noisy data
- Greedy algorithm for growing tree, followed by post-pruning
- Searches *complete* hypothesis space, with preference for small trees
- Methods to avoid overfitting the data are necessary
- Many extensions to basic algorithm
- Still debate on alternative selection measures, pruning methods