Decision Trees on Real Problems

Must consider the following issues:

- Multi-class problems
- Alternative splitting criterion
- Noise in the data
- Real-valued attributes
- Missing values
- Attributes with costs

Applying Entropy to Multiple Classes

Thus far we have assumed that the target class is Boolean. More generally, the class can take on c values, then the entropy of S relative to this c-wise classification is defined as:

$$Entropy(S) = \sum_{i=1}^{c} -\frac{\mid S_i \mid}{\mid S \mid} \log_2(\frac{\mid S_i \mid}{\mid S \mid})$$

Where S_i is the proportion of S belonging to class i.

A Problem with Information Gain

- Biased toward attributes that have many possible values.
  Examples:
  Date attribute: 365 possible values
  Name attribute: 50,000 possible values

- Splits data into (possibly) perfectly classified (albeit small) partitions
- *Problem*: Not good class predictors.

Slide CS478–3

An Alternative Measure

GainRatio: penalizes attributes with many values by incorporating a term called $SplitInformation$.

$SplitInformation$ measures the entropy of the data with respect to the attribute values, not the class.

$$SplitInformation(S, A) \equiv -\sum_{i=1}^{|V|} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

where S_i is subset of S for which A has value $v_i \in V$

Slide CS478–4

$GainRatio$ uses $SplitInformation$ to discourage preference for these attributes.

$$GainRatio(S, A) \equiv \frac{Gain(S, A)}{SplitInformation(S, A)}$$

Slide CS478–5

A Problem with Gain Ratio

- $SplitInformation$ can be very small or even zero when $| S_i | \approx | S |$ for some S_i.
- In this case, $GainRatio$ becomes very large or even undefined, skewing the results.
- To avoid this problem, one approach is to compute the $Gain$ of each attribute. Then for those that have above average $Gain$, choose the best by applying the $GainRatio$ test. This is the approach used by C4.5.

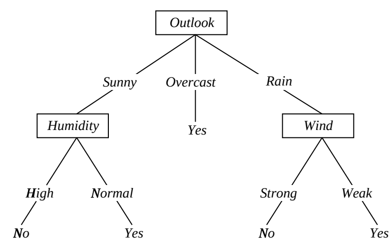
Slide CS478–6

Overfitting in Decision Trees

Consider adding noisy training example #15:

$$Sunny,\ Hot,\ Normal,\ Strong,\ PlayTennis = No$$

What effect on earlier tree?



Slide CS478–7

Overfitting

Overfitting occurs when the learned concept is too specific to the training data. Overfitting can occur for several reasons:

- Noise
- Not enough training examples

In one study of 5 learning tasks, overfitting decreased the accuracy of the decision trees by 10-25%.

Moral: Overfitting is a <u>real</u> problem!

Slide CS478–8

A precise definition

Consider error of hypothesis h over

- training data: $error_{train}(h)$
- entire distribution $\mathcal{D}$ of data: $error_{\mathcal{D}}(h)$

Hypothesis $h \in H$ **overfits** training data if there is an alternative hypothesis $h' \in H$ such that

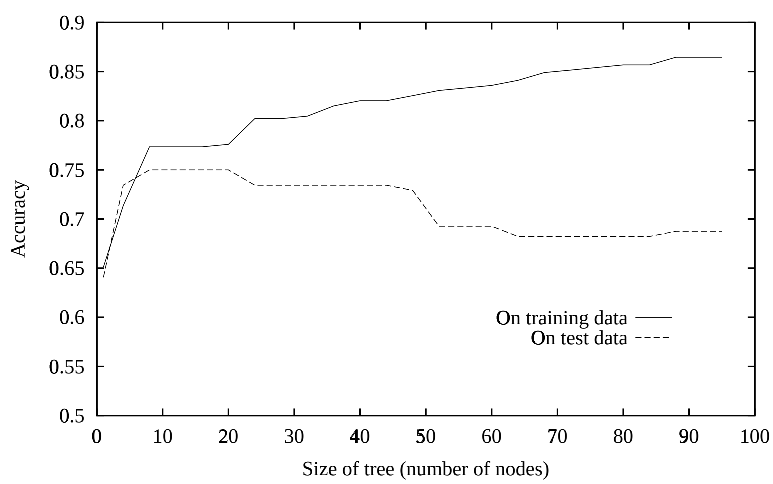
$$error_{train}(h) < error_{train}(h')$$

and

$$error_{\mathcal{D}}(h') < error_{\mathcal{D}}(h)$$

Slide CS478–9

Recognizing Overfitting



Slide CS478–10

Avoiding Overfitting

- **Prepruning:** Stop growing the tree when there is not enough data to make reliable decisions, or when the examples are acceptably homogeoeous
- **Postpruning:** Grow the full decision tree and then remove nodes for which there is not sufficient evidence.

Prepruning: easier and more intuitive
Postpruning: generally works better in practice

Evaluation Methods for Pruning

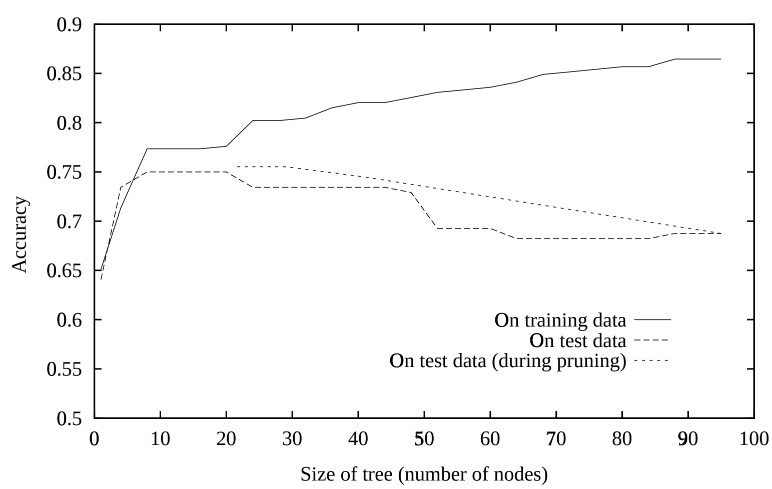
- **Validation Methods:** Reserve some portion of the training data as a *validation set*. Two common methods are:
  - Use a single training set and a single validation set.
  - *Cross-validation:* Divide the training set into N partitions. Do N experiments: each partition is used once as the validation set, and the other N-1 partitions are used as the training set.
- **Statistical Analyses:** Use statistical tests to estimate whether expanding/pruning a node is likely to produce an improvement beyond the training data.

Reduced-Error Pruning

- Split data into *training* and *validation* set.
- Build a full decision tree from the training set.
- Do until further pruning is harmful (decreases accuracy on the validation set):
  - For each non-leaf node N:
    - ∗ Temporarily prune the subtree rooted by N and replace it with a leaf node labelled with the majority class.
    - ∗ Test the accuracy of the pruned tree on the validation set.
  - Greedily remove the subtree that results in the greatest improvement in accuracy on the *validation* set.

Slide CS478–13

Effect of Reduced-Error Pruning



Slide CS478–14

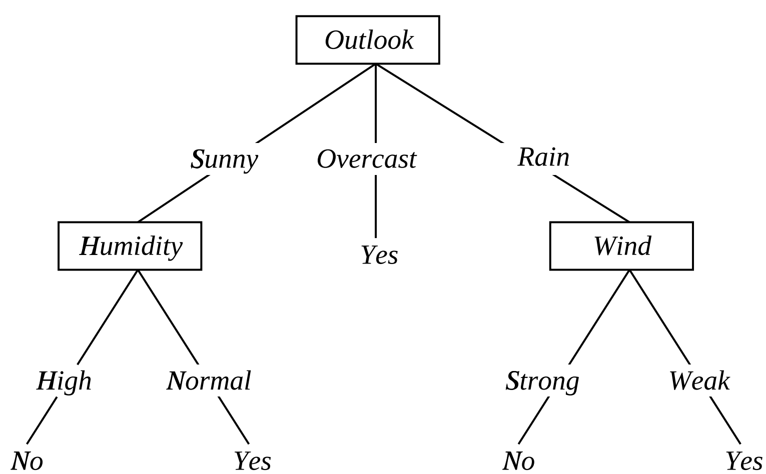
Rule Post-Pruning

Perhaps most frequently used pruning method (e.g. C4.5).

1. Split data into training and validation sets.
2. Build a full decision tree from the training set.
3. Convert tree to an equivalent set of rules.
4. Prune (generalize) each rule by removing preconditions.
5. Sort pruned rules based on estimated accuracy. Use them in this order to classify new instances.

Slide CS478–15

Converting A Tree to Rules



Slide CS478–16

IF $(Outlook = Sunny) \wedge (Humidity = High)$
THEN $PlayTennis = No$

IF $(Outlook = Sunny) \wedge (Humidity = Normal)$
THEN $PlayTennis = Yes$

IF $(Outlook = Overcast)$
THEN $PlayTennis = Yes \ldots$

Slide CS478–17

Discretizing Continuous-Valued Attributes

- Idea: dynamically define a set of discrete values that are candidates for partitioning the examples.
- For a continuous feature A, each discretized value will be a binary attribute of the form $(A < THRESHOLD)$.
- These dynamically generated attributes can then compete with all other (discrete) attributes when building the decision tree.

Sort the examples according to their values for A.
For each ordered pair X_i, X_{i+1} in the sorted list,
If the category of X_i and X_{i+1} are different,
Then use the midpoint between their values as a candidate threshold.

Slide CS478–18

An Example

Value:	10	15	21	28	32	40	50
Class:	No	Yes	Yes	No	Yes	Yes	No

Unknown Attribute Values

1. Assign the most common value for the attribute among the training examples that reached the same node in the decision tree.

2. Assign the most common value for the attribute among the training examples with the same class c_i that reached the same node in the decision tree.

3. Push the example down the decision tree in fractions, probabilistically. The fractions are based on the proportion of examples at the node that have each attribute value.

Attributes with Costs

- Introduce a cost term into attribute selection measure:

$$\frac{Gain^2(S,A)}{Cost(A)}.$$

Slide CS478–21

Strengths of decision trees

- Easy to generate; simple algorithm.
- Easy to read small trees; can be converted to rule set.
- Decision trees are highly expressive.
- Relatively fast to construct; classification is very fast.
- Can achieve good performance on many tasks.
- A wide variety of problems can be recast as classification problems.

Slide CS478–22

Weaknesses of decision trees

- Not always sufficient to learn complex concepts.
- Can be hard to understand.
- Some problems with continuously-valued attributes or classes may not be easily discretized.
- Methods for handling missing attribute values are somewhat clumsy.