Decision Tree Learning

TM Chapter 3, RN Section 18.3

Outline

- Decision tree representation
- ID3 learning algorithm
- Entropy & information gain
- Overfitting

Why Study Decision Tree (DT) Learning?

	Concept Learning	DT Learning
Target fn/concept	Binary outputs	Discrete outputs
Training data	Noise-free	Robust to noise
Hypothesis space	Restricted (hard bias)	Complete, expressive
Search strategy	Complete: version space Refine search per example	Incomplete: prefer shorter tree (soft bias) Refine search using all examples No backtracking
Exploit structure	General to specific ordering	Simple to complex ordering

DT Learning for *Wait* for a Table

Decide whether to wait for a table at a restaurant (i.e., output or target concept *Wait*) based on the following input attributes:

1. *Alternate*: is there an alternative restaurant nearby?
2. *Bar*: is there a comfortable bar area to wait in?
3. *Fri/Sat*: is today Friday or Saturday?
4. *Hungry*: are we hungry?
5. *Patrons*: number of people in the restaurant (None, Some, Full)
6. *Price*: price range ($, $$, $$$)
7. *Raining*: is it raining outside?
8. *Reservation*: have we made a reservation?
9. *Type*: kind of restaurant (French, Italian, Thai, Burger)
10. *WaitEstimate*: estimated waiting time in minutes (0-10, 10-30, 30-60, >60)

DT Learning for *Wait* for a Table

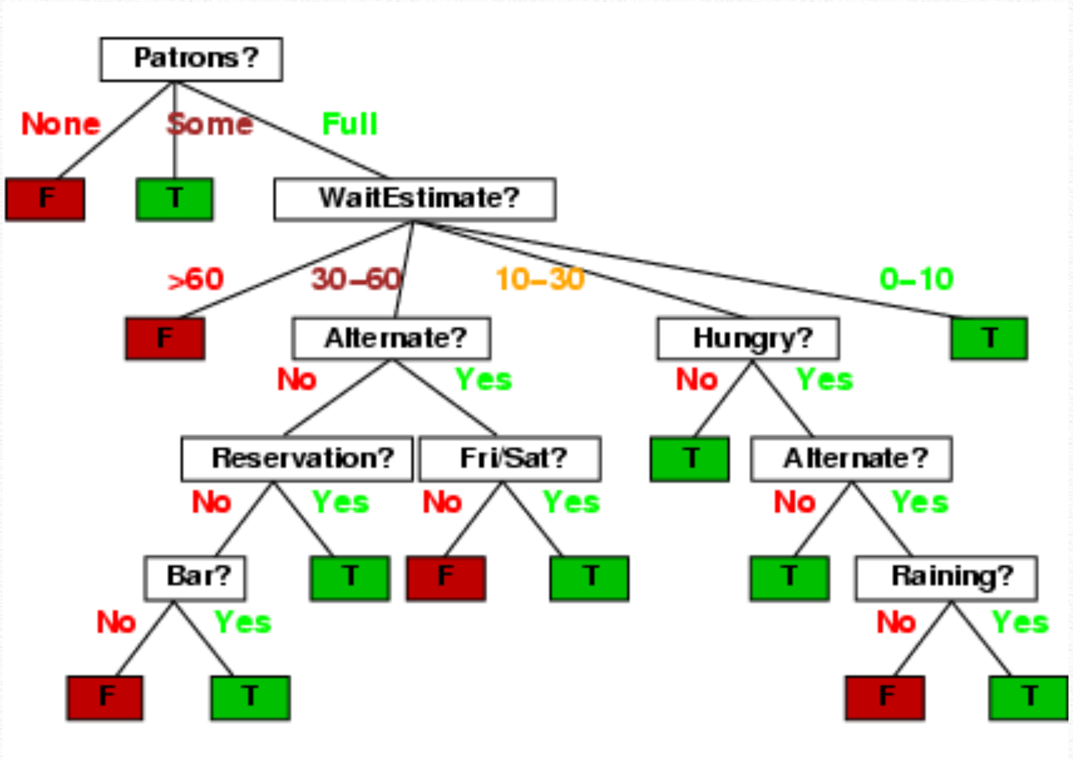
- Input instance of training examples is described by input attribute values (Boolean, discrete, continuous)
- These training examples show situations where I will/won't wait for a table:

Example	Attributes										Target
	Alt	Bar	Fri	Hun	Pat	$Price$	$Rain$	Res	$Type$	Est	$Wait$
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

- Classification of training examples is +ve (T) or –ve (F)

Decision Trees

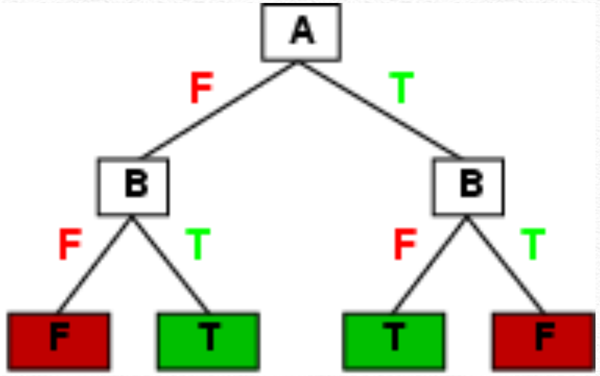
- Another possible representation for hypotheses
- Here is the “true” decision tree for deciding whether to wait for a table:



Expressive Power

- Decision trees can express any function of the input attributes
- e.g., for Boolean functions, truth table row → path to leaf:

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F

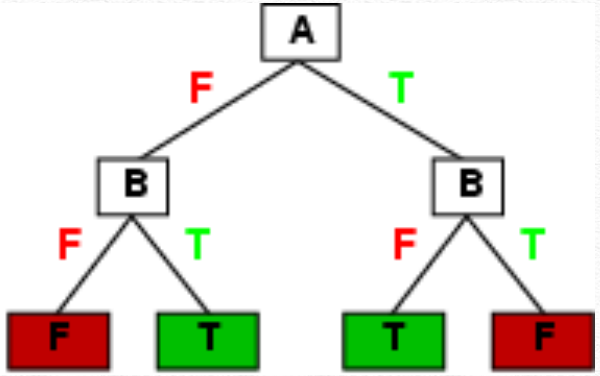


- Trivially, there is a consistent decision tree for any training set with one path to leaf for each example but it probably won't generalize well to classify unobserved input instances
- Prefer to find compact decision trees

Expressive Power

- A Boolean decision tree can be expressed in disjunctive normal form
- e.g., A xor B ⇔ $(\neg A \wedge B) \vee (A \wedge \neg B)$

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



- Target concept C ⇔ ($Path_1 \vee Path_2 \vee \ldots$) where each *Path* is a conjunction of attribute-value tests required to follow that path leading to a leaf with value *true*
- e.g., *Path* = (*Patrons* = *Full* ∧ *WaitEstimate* = 0-10)

Hypothesis/Search Space

Number of distinct binary decision trees with m Boolean attributes

= number of Boolean-valued functions

= number of distinct truth tables with 2^m rows

$= 2^{2^m}$

e.g., with 6 Boolean attributes, there are more than 18,446,744,073,709,551,616 trees

DECISION-TREE-LEARNING

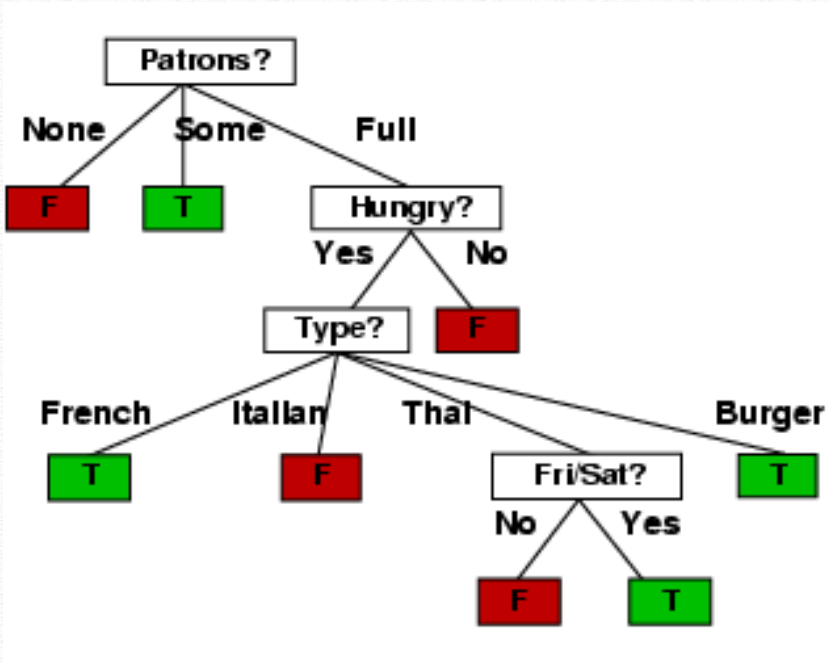
- Aim. Find a small tree consistent with the training examples
- Idea. Greedily choose "most important" attribute as root of (sub)tree

function DECISION-TREE-LEARNING(*examples*, *attributes*, *parent_examples*) **returns** tree

```
if examples is empty then return PLURALITY-VALUE(parent_examples)
else if all examples have the same classification then return the classification
else if attributes is empty then return PLURALITY-VALUE(examples)
else
    A ← argmax_{a ∈ attributes} IMPORTANCE(a, examples)
    tree ← a new decision tree with root test A
    for each value v_k of A do
        exs ← {e : e ∈ examples and e.A = v_k}
        subtree ← DECISION-TREE-LEARNING(exs, attributes − A, examples)
        add a branch to tree with label (A = v_k) and subtree subtree
    return tree
```

Learned Decision Tree

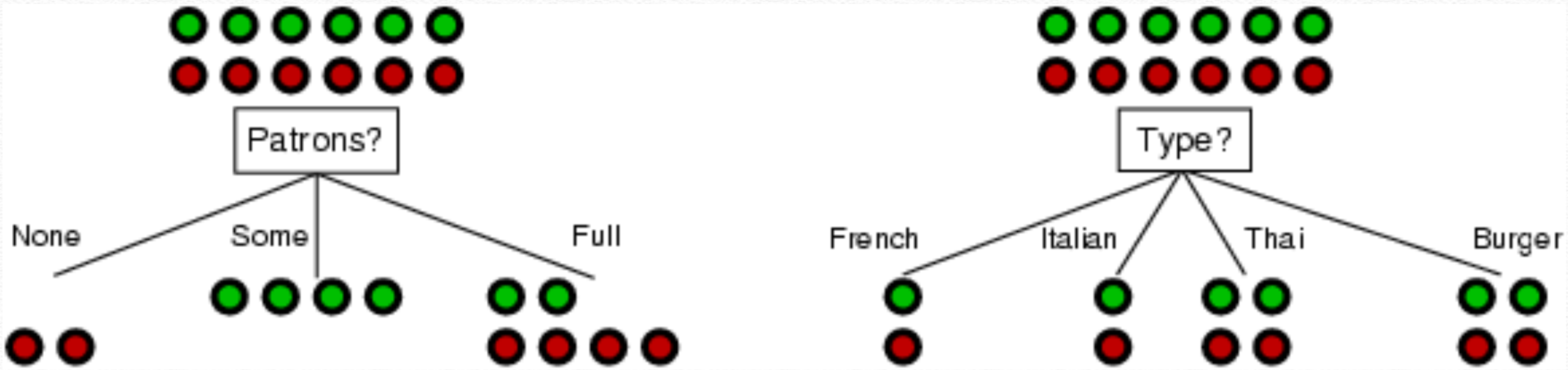
- Decision tree learned from the 12 examples:



- Substantially simpler than “true” decision tree -- a more complex hypothesis isn’t justified by small amount of data

Choosing "Most Important" Attribute

Intuition. A good attribute splits the examples into subsets that are (ideally) "all +ve" or "all −ve" (i.e., classified exactly)



Patrons? is a better choice

Using Information Theory

- To implement the IMPORTANCE function in the DECISION-TREE-LEARNING algorithm, use entropy to measure uncertainty of classification
- Entropy measures uncertainty of r.v. $C \in \{c_1, \ldots, c_k\}$:

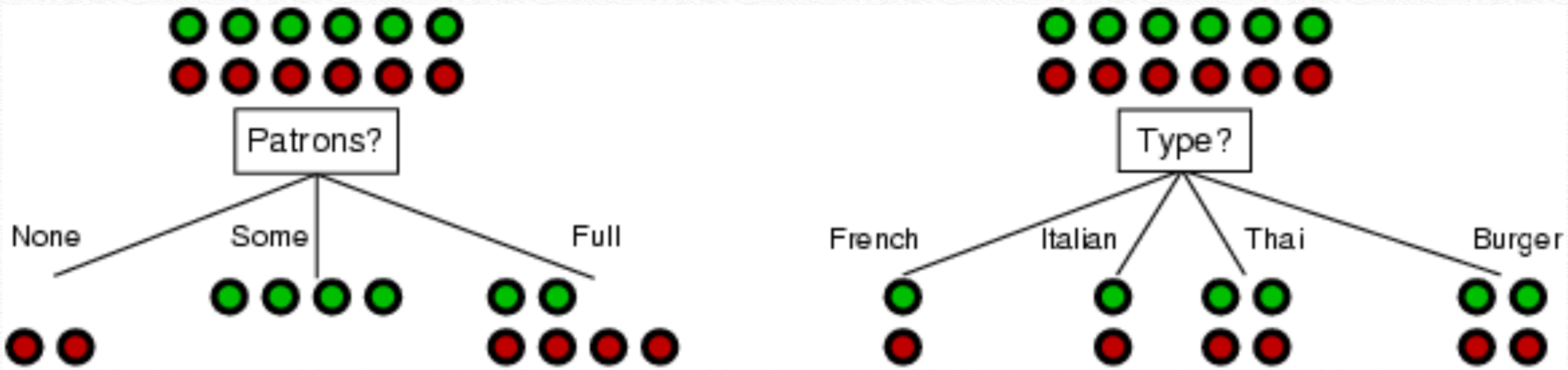
$$H(C) = -\sum_{i=1}^{k} P(c_i) \log_2 P(c_i) \ .$$

- Define $B(q)$ as entropy of Boolean r.v. that is true with probability q : $B(q) = -(q \log_2 q + (1-q) \log_2 (1-q))$.
- For a training set containing p +ve examples and n −ve examples, entropy of target concept C on this set is

$$H(C) = B\left(\frac{p}{p+n}\right) = -\frac{p}{p+n} \log_2 \frac{p}{p+n} - \frac{n}{p+n} \log_2 \frac{n}{p+n} \ .$$

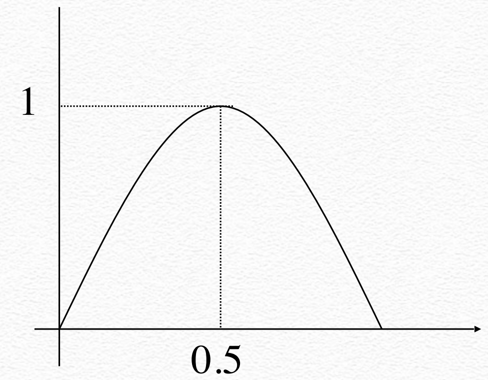
Uncertainty of Classification

- If $p = n \neq 0$, then $H(C) = B(1/2) = 1$ (maximum uncertainty)
- If $(p \neq 0, n = 0)$ or $(p = 0, n \neq 0)$, then $H(C) = 0$ (no uncertainty)
- If $p = 2, n = 4$, $H(C) = B(2/6) \in (0, 1)$ (some uncertainty)



Entropy Curve

- For $p/(p+n)$ between 0 & 1, the 2-class entropy $B(p/(p+n))$ is
  - 0 when $p/(p+n)$ is 0
  - 1 when $p/(p+n)$ is 0.5
  - 0 when $p/(p+n)$ is 1
  - monotonically increasing between 0 and 0.5
  - monotonically decreasing between 0.5 and 1



Information Gain

- A chosen attribute A divides the training set E into subsets E_1, …, E_d corresponding to the d distinct values of A. Each subset E_i has p_i +ve and n_i −ve examples.

$$H(C|A) = \sum_{i=1}^{d} \frac{p_i + n_i}{p + n} \, B\left(\frac{p_i}{p_i + n_i}\right).$$

- Information gain of target concept C from the attribute test on A is the expected reduction in entropy:

$$Gain(C, A) = \boxed{B\left(\frac{p}{p+n}\right)} - \boxed{H(C|A)}$$

- Choose the attribute A with the largest *Gain*

Entropy $H(C)$ of this node

Expected remaining entropy after testing A

Information Gain

- For the training set, $p = n = 6$, $H(Wait) = B(6/12) = 1$
- Consider the attributes *Patrons* and *Type* (and others too):

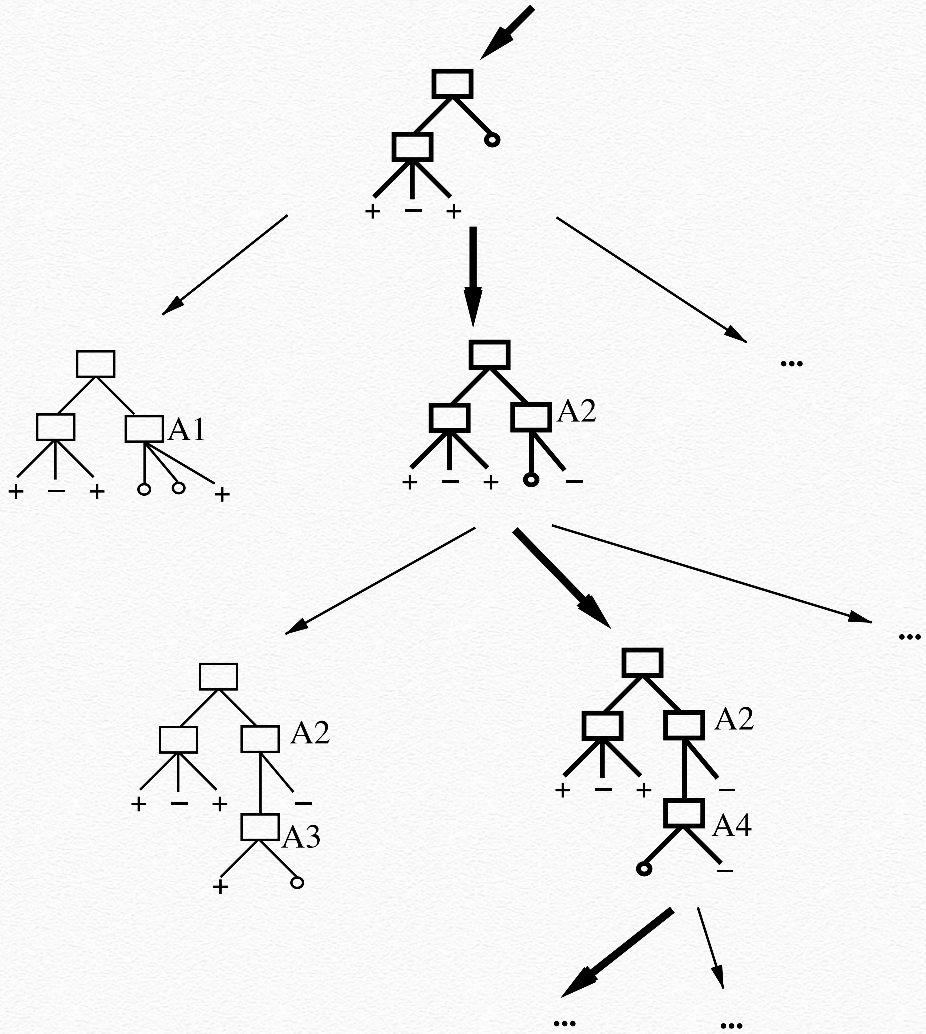
$$Gain(Wait, Patrons) = 1 - \left[\frac{2}{12}B\left(\frac{0}{2}\right) + \frac{4}{12}B\left(\frac{4}{4}\right) + \frac{6}{12}B\left(\frac{2}{6}\right)\right] = 0.541 \text{ bits}$$

$$Gain(Wait, Type) = 1 - \left[\frac{2}{12}B\left(\frac{1}{2}\right) + \frac{2}{12}B\left(\frac{1}{2}\right) + \frac{4}{12}B\left(\frac{2}{4}\right) + \frac{4}{12}B\left(\frac{2}{4}\right)\right] = 0 \text{ bits}$$

- *Patrons* has the highest *Gain* of all attributes and so is chosen by the DECISION-TREE-LEARNING algorithm as the root
- Recursively choose attributes for children nodes

Hypothesis Space Search

- DECISION-TREE-LEARNING is guided by IMPORTANCE function: information gain heuristic to search thru the space of DTs from simplest to increasingly complex
- Number of distinct DTs with m Boolean attributes $\gg 2^{2^m}$?



Inductive Bias of DECISION-TREE-LEARNING

Approximate inductive bias of DECISION-TREE-LEARNING. (a) Shorter trees are preferred. (b) Trees that place high information gain attributes close to the root are preferred.

- If only (a) is considered, it is exactly the approximate inductive bias of breadth first search for the shortest consistent DT, which can be prohibitively expensive

- Bias is a preference for some hypotheses, rather than a restriction of hypothesis space. Which one is more desirable?

- Occam's razor: Prefer shortest/simplest hypothesis that fits the data

Occam's Razor

Why prefer short/simple hypotheses?

Argument in favor:

- Fewer short hypotheses than long hypotheses
  - Short/simple hypothesis that fits data unlikely to be coincidence
  - Long/complex hypothesis that fits data may be coincidence

Argument opposed:

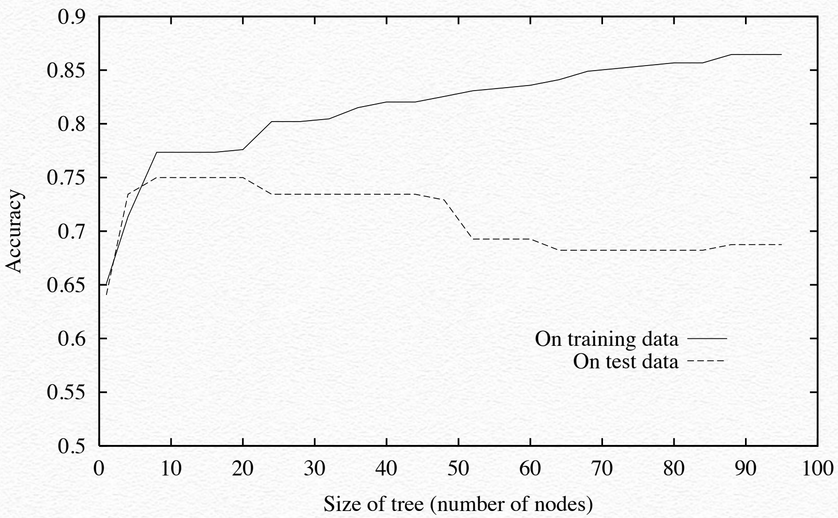
- Many ways to define small sets of hypotheses (e.g., all trees with a prime number of nodes that use attributes beginning with "Z")
- Small sets of short/simple hypothesis can be obtained using different hypothesis representations

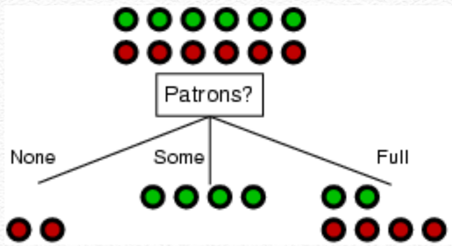
Overfitting

Definition. Hypothesis $h \in H$ **overfits** the set D of training examples iff

$$\exists h' \in H \setminus \{h\} \quad (error_D(h) < error_D(h')) \wedge (error_{D_X}(h) > error_{D_X}(h'))$$

where $error_D(h)$ and $error_{D_X}(h)$ denote errors of h over D and set D_X of examples corresponding to instance space X, respectively





Overfitting in DT learning

New erroneous/noisy training example with *Patrons* = *Some* and *Wait* = *No*?

Limited data: Small no. of training examples with *Patrons* = *None*?

Overfitting

How to avoid overfitting?

- Stop growing DT when expanding a node is not statistically significant
- Allow DT to grow and overfit the data, then post-prune it

How to select "best" DT?

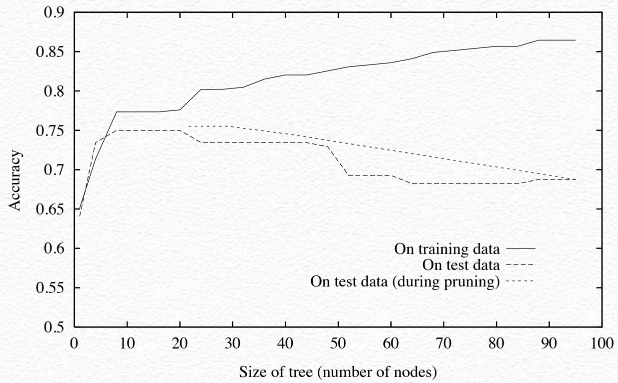
- Measure performance over training examples/data
- Measure performance over a separate validation dataset
- MDL: minimize $size(tree)$ & $size(misclassifications(tree))$

Reduced-Error Pruning

Partition data into training and validation sets

Do until further pruning is harmful:

1. Evaluate impact on validation set of pruning each possible node (i.e., removing subtree rooted at it)
2. Greedily remove the one that most improves validation set accuracy



- Produces smallest version of most accurate subtree
- What if data is limited?

Rule Post-Pruning

- Convert learned DT to an equivalent set of rules by creating one rule for each path from the root to a leaf

IF $(Patrons = Full) \wedge (Hungry = Yes) \wedge (Type = Thai) \wedge (Fri/Sat = No)$

THEN $Wait = No$

- Prune (generalize) each rule by removing any precondition that improves its estimated accuracy
- Sort pruned rules by estimated accuracy into desired sequence for use when classifying unobserved input instances

Continuous-Valued Attributes

Define a discrete-valued input attribute to partition the values of a continuous input attribute into a discrete set of intervals for testing:

WaitEstimate: estimated waiting time in minutes (0-10, 10-30, 30-60, >60)

Attributes with Many Values

Problem. *Gain* will select attribute with many values (e.g., *Date*)

Solution. Use *GainRatio* instead:

$$GainRatio(C, A) = \frac{Gain(C, A)}{SplitInformation(C, A)}$$

$$SplitInformation(C, A) = -\sum_{i=1}^{d} \frac{|E_i|}{|E|} \log_2 \frac{|E_i|}{|E|}$$

Recall (page 16) that a chosen attribute A divides the training set E into subsets $E_1, \ldots, E_d$ corresponding to the d distinct values of A

Attributes with Differing Costs

Problem. How to learn a consistent DT with low expected cost?
(e.g., medical diagnosis: *Temperature*, *Biopsy*, *BloodTest*, *Pulse*)

Solution. Replace *Gain* by

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

$$\frac{2^{Gain(C,A)} - 1}{(Cost(A) + 1)^w}$$

where $w \in [0, 1]$ determines importance of cost

Missing Attribute Values

Problem. What if some examples are missing values of A?

Solution. Use training example anyway & sort through DT

- If node n tests A, then assign most common value of A among other examples sorted to node n
- Assign most common value of A among other examples sorted to node n with same value of output/target concept
- Assign probability p_i to each possible value of A
  - Assign fraction p_i of example to each descendant in DT

Classify new unobserved input instances with missing attribute values in same manner

Summary

- Decision tree learning uses information gain
- Decision tree learning differs from concept learning in its discrete-valued output/target concept, robustness to noisy data, complete & expressive hypothesis space, incomplete search strategy, soft preference bias, search refinement with all examples, simple to complex ordering of hypotheses (see table on page 3)
- Overfitting arises due to noisy and limited training data and can be avoided by post-pruning
- Extensions to DECISION-TREE-LEARNING include attributes with continuous, missing, many values and differing costs