

# Decision Tree Learning

March 18, 2009

**Acknowledgement:** Material derived from slides by:

Tom M. Mitchell, <http://www-2.cs.cmu.edu/~tom/mlbook.html>

Andrew W. Moore, <http://www.cs.cmu.edu/~awm/tutorials>

and Eibe Frank, <http://www.cs.waikato.ac.nz/ml/weka/>

## Aims

This lecture will enable you to describe decision tree learning, the use of entropy and the problem of overfitting. Following it you should be able to:

- define the decision tree representation
- list representation properties of data and models for which decision trees are appropriate
- reproduce the basic top-down algorithm for decision tree induction (TDIDT)
- define entropy in the context of learning a Boolean classifier from examples

## Aims

- describe the inductive bias of the basic TDIDT algorithm
- define overfitting of a training set by a hypothesis
- describe developments of the basic TDIDT algorithm: pruning, rule generation, numerical attributes, many-valued attributes, costs, missing values

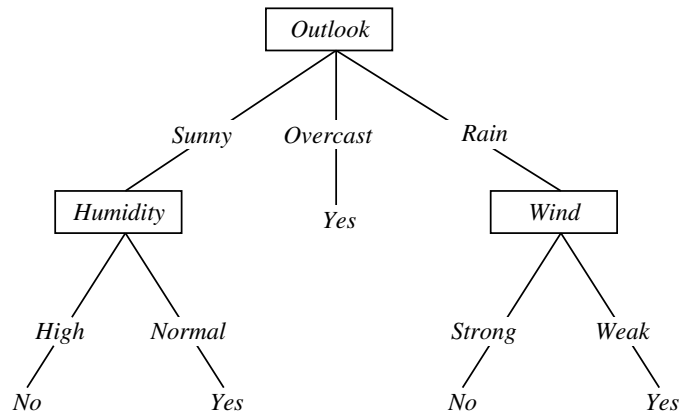
[Recommended reading: Mitchell, Chapter 3]

[Recommended exercises: 3.1, 3.2, 3.4(a,b)]

## Introduction

- Decision trees are the single most popular data mining tool
  - Easy to understand
  - Easy to implement
  - Easy to use
  - Computationally cheap
- There are some drawbacks, though ! (such as overfitting)
- They do *classification* : predict a categorical output from categorical and/or real inputs

## Decision Tree for *PlayTennis*



## A Tree to Predict C-Section Risk

Learned from medical records of 1000 women

Negative examples are C-sections

```

[833+,167-] .83+ .17-
Fetal_Presentation = 1: [822+,116-] .88+ .12-
| Previous_Csection = 0: [767+,81-] .90+ .10-
| | Primiparous = 0: [399+,13-] .97+ .03-
| | Primiparous = 1: [368+,68-] .84+ .16-
| | | Fetal_Distress = 0: [334+,47-] .88+ .12-
| | | | Birth_Weight < 3349: [201+,10.6-] .95+ .05-
| | | | Birth_Weight >= 3349: [133+,36.4-] .78+ .22-
| | | Fetal_Distress = 1: [34+,21-] .62+ .38-
| Previous_Csection = 1: [55+,35-] .61+ .39-
Fetal_Presentation = 2: [3+,29-] .11+ .89-
Fetal_Presentation = 3: [8+,22-] .27+ .73-
  
```

## Decision Trees

Decision tree representation:

- Each internal node tests an attribute
- Each branch corresponds to attribute value
- Each leaf node assigns a classification

How would we represent:

- $\wedge, \vee, \text{XOR}$
- $(A \wedge B) \vee (C \wedge \neg D \wedge E)$
- $M$  of  $N$

## Decision Trees

$X \wedge Y$

```

X = t:
| Y = t: true
| Y = f: no
X = f: no
  
```

$X \vee Y$

```

X = t: true
X = f:
| Y = t: true
| Y = f: no
  
```

2 of 3

```

X = t:
| Y = t: true
| Y = f:
| | Z = t: true
| | Z = f: false
X = f:
| Y = t:
| | Z = t: true
| | Z = f: false
| Y = f: false

```

So in general decision trees represent a *disjunction of conjunctions* of constraints on the attributes values of instances.

- Instances describable by attribute–value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data

Examples:

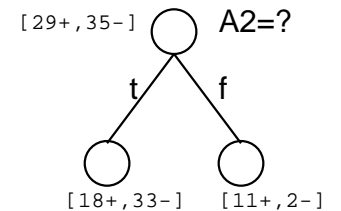
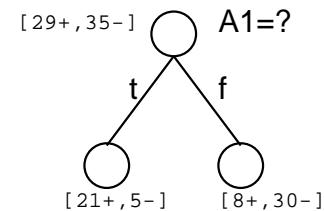
- Equipment or medical diagnosis
- Credit risk analysis
- Modeling calendar scheduling preferences

## Top-Down Induction of Decision Trees (TDIDT or ID3)

Main loop:

1.  $A \leftarrow$  the “best” decision attribute for next *node*
2. Assign  $A$  as decision attribute for *node*
3. For each value of  $A$ , create new descendant of *node*
4. Sort training examples to leaf nodes
5. If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes

## Which attribute is best?



## Bits

You are watching a set of independent random samples of  $X$   
You observe that  $X$  has four possible values

$$P(X = A) = \frac{1}{4} \quad P(X = B) = \frac{1}{4} \quad P(X = C) = \frac{1}{4} \quad P(X = D) = \frac{1}{4}$$

So you might see: BAACBADCDADDDA...

You transmit data over a binary serial link. You can *encode* each reading with two bits (e.g. A = 00, B = 01, C = 10, D = 11)

0100001001001110110011111100...

## Fewer Bits

Someone tells you that the probabilities are not equal

$$P(X = A) = \frac{1}{2} \quad P(X = B) = \frac{1}{4} \quad P(X = C) = \frac{1}{8} \quad P(X = D) = \frac{1}{8}$$

It's possible ...

... to invent a *coding* for your transmission that only uses 1.75 bits on average per symbol. How ?

## Fewer Bits

Someone tells you that the probabilities are not equal

$$P(X = A) = \frac{1}{2} \quad P(X = B) = \frac{1}{4} \quad P(X = C) = \frac{1}{8} \quad P(X = D) = \frac{1}{8}$$

It's possible ...

... to invent a *coding* for your transmission that only uses 1.75 bits per symbol on average. How ?

A	0
B	10
C	110
D	111

(This is just one of several ways)

## Fewer Bits

Suppose there are three equally likely values

$$P(X = A) = \frac{1}{3} \quad P(X = B) = \frac{1}{3} \quad P(X = C) = \frac{1}{3}$$

Here's a naïve coding, costing 2 bits per symbol

A	00
B	01
C	10

Can you think of a coding that would need only 1.6 bits per symbol on average ?

## Fewer Bits

Suppose there are three equally likely values

$$P(X = A) = \frac{1}{3} \quad P(X = B) = \frac{1}{3} \quad P(X = C) = \frac{1}{3}$$

Using the same approach as before, we can get a coding costing 1.6 bits per symbol on average ...

A	0
B	10
C	11

This gives us, on average  $\frac{1}{3} \times 1$  bit for A and  $2 \times \frac{1}{3} \times 2$  bits for B and C, which equals  $\frac{5}{3} \approx 1.6$  bits.

Is this the best we can do ?

## Fewer Bits

Suppose there are three equally likely values

$$P(X = A) = \frac{1}{3} \quad P(X = B) = \frac{1}{3} \quad P(X = C) = \frac{1}{3}$$

From information theory, the optimal number of bits to encode a symbol with probability  $p$  is  $-\log_2 p$  ...

So the best we can do for this case is  $-\log_2 \frac{1}{3}$  bits for each of A, B and C, or 1.5849625007211563 bits per symbol

## General Case

Suppose  $X$  can have one of  $m$  values ...  $V_1, V_2, \dots, V_m$

$$P(X = V_1) = p_1 \quad P(X = V_2) = p_2 \quad \dots \quad P(X = V_m) = p_m$$

What's the smallest possible number of bits, on average, per symbol, needed to transmit a stream of symbols drawn from  $X$ 's distribution ? It's

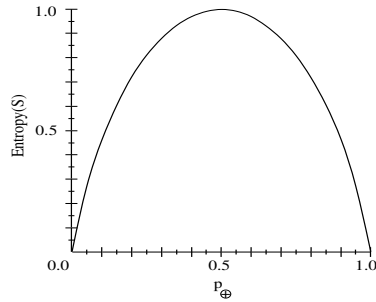
$$\begin{aligned}
H(X) &= -p_1 \log_2 p_1 - p_2 \log_2 p_2 - \dots - p_m \log_2 p_m \\
&= -\sum_{j=1}^m p_j \log_2 p_j
\end{aligned}$$

$H(X)$  = the *entropy* of  $X$

## General Case

"High entropy" means  $X$  is very uniform and boring  
"Low entropy" means  $X$  is very varied and interesting

## Entropy



Where:

$S$  is a sample of training examples

$p_{\oplus}$  is the proportion of positive examples in  $S$

$p_{\ominus}$  is the proportion of negative examples in  $S$

## Entropy

Entropy measures the “impurity” of  $S$

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

A “pure” sample is one in which all examples are of the same class.

## Entropy

$Entropy(S)$  = expected number of bits needed to encode class ( $\oplus$  or  $\ominus$ ) of randomly drawn member of  $S$  (under the optimal, shortest-length code)

Why ?

Information theory: optimal length code assigns  $-\log_2 p$  bits to message having probability  $p$ .

So, expected number of bits to encode  $\oplus$  or  $\ominus$  of random member of  $S$ :

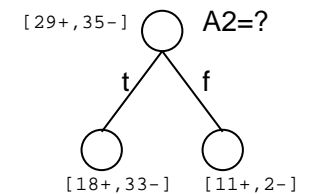
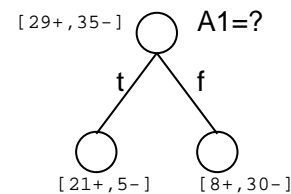
$$p_{\oplus}(-\log_2 p_{\oplus}) + p_{\ominus}(-\log_2 p_{\ominus})$$

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

## Information Gain

$Gain(S, A)$  = expected reduction in entropy due to sorting on  $A$

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$



### Information Gain

$$\begin{aligned}
 \text{Gain}(S, A1) &= \text{Entropy}(S) - \left( \frac{|S_t|}{|S|} \text{Entropy}(S_t) + \frac{|S_f|}{|S|} \text{Entropy}(S_f) \right) \\
 &= 0.9936 - \\
 &\quad \left( \left( \frac{26}{64} \left( -\frac{21}{26} \log_2 \left( \frac{21}{26} \right) - \frac{5}{26} \log_2 \left( \frac{5}{26} \right) \right) \right) + \right. \\
 &\quad \left. \left( \frac{38}{64} \left( -\frac{8}{38} \log_2 \left( \frac{8}{38} \right) - \frac{30}{38} \log_2 \left( \frac{30}{38} \right) \right) \right) \right) \\
 &= 0.9936 - ( 0.2869 + 0.4408 ) \\
 &= 0.2658
 \end{aligned}$$

### Information Gain

$$\begin{aligned}
 \text{Gain}(S, A2) &= 0.9936 - ( 0.7464 + 0.0828 ) \\
 &= 0.1643
 \end{aligned}$$

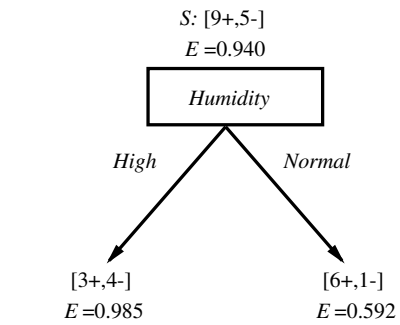
### Information Gain

So we choose A1, since it gives a larger expected reduction in entropy.

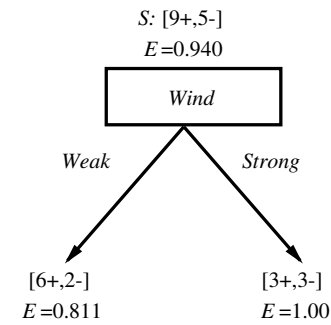
### Training Examples

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

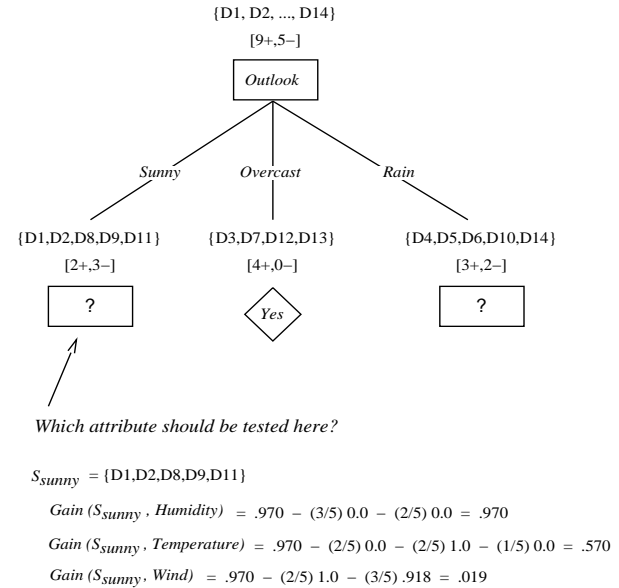
## Which attribute is the best classifier?



$Gain(S, Humidity)$   
 $= .940 - (7/14).985 - (7/14).592$   
 $= .151$



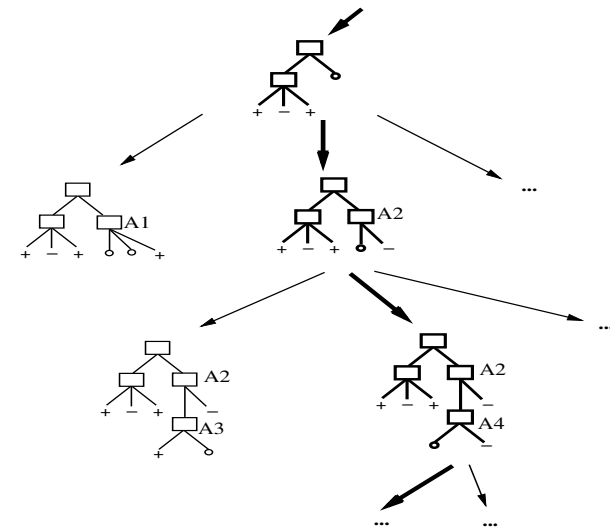
$Gain(S, Wind)$   
 $= .940 - (8/14).811 - (6/14)1.0$   
 $= .048$



## Brief History of Decision Tree Learning Algorithms

- late 1950's – Bruner et al. in psychology work on modelling *concept acquisition*
- early 1960s – Hunt et al. in computer science work on *Concept Learning Systems (CLS)*
- late 1970s – Quinlan's *Iterative Dichotomizer 3 (ID3)* based on CLS is efficient at learning on then-large data sets
- early 1990s – ID3 adds features, develops into C4.5, becomes the "default" machine learning algorithm
- late 1990s – C5.0, commercial version of C4.5 (available from SPSS and [www.rulequest.com](http://www.rulequest.com))

## Hypothesis Space Search by ID3



## Hypothesis Space Search by ID3

- Hypothesis space is complete! (contains all finite discrete-valued functions w.r.t attributes)
  - Target function surely in there...
- Outputs a single hypothesis (which one?)
  - Can't play 20 questions...
- No back tracking
  - Local minima...
- Statistically-based search choices
  - Robust to noisy data...
- Inductive bias: approx "prefer shortest tree"

## Inductive Bias in ID3

Note  $H$  is the power set of instances  $X$

→ Unbiased?

Not really...

- Preference for short trees, and for those with high information gain attributes near the root
- Bias is a *preference* for some hypotheses, rather than a *restriction* of hypothesis space  $H$
- an incomplete search of a complete hypothesis space *versus* a complete search of an incomplete hypothesis space (as in learning conjunctive concepts)
- Occam's razor: prefer the shortest hypothesis that fits the data

## Occam's Razor

William of Ockham (c. 1287-1347)

*Entities should not be multiplied beyond necessity*

Why prefer short hypotheses?

Argument in favour:

- Fewer short hypotheses than long hypotheses
- a short hyp that fits data unlikely to be coincidence
- a long hyp that fits data might be coincidence

## Occam's Razor

Argument opposed:

- There are many ways to define small sets of hypotheses
  - e.g., all trees with a prime number of nodes that use attributes beginning with "Z"
- What's so special about small sets based on *size* of hypothesis??

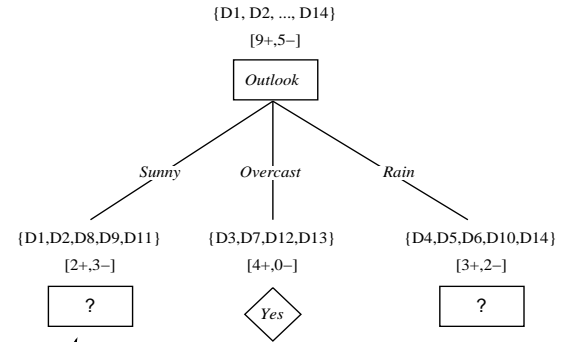
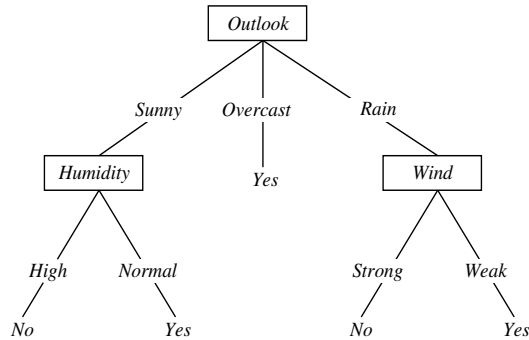
We will come back to this topic again.

## Overfitting in Decision Tree Learning

Consider adding noisy training example #15:

*Sunny, Hot, Normal, Strong, PlayTennis = No*

What effect on earlier tree?



Which attribute should be tested here?

$$S_{\text{sunny}} = \{D1, D2, D8, D9, D11\}$$

$$\text{Gain}(S_{\text{sunny}}, \text{Humidity}) = .970 - (3/5) 0.0 - (2/5) 0.0 = .970$$

$$\text{Gain}(S_{\text{sunny}}, \text{Temperature}) = .970 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = .570$$

$$\text{Gain}(S_{\text{sunny}}, \text{Wind}) = .970 - (2/5) 1.0 - (3/5) .918 = .019$$

## Overfitting in General

Consider error of hypothesis  $h$  over

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

### Definition

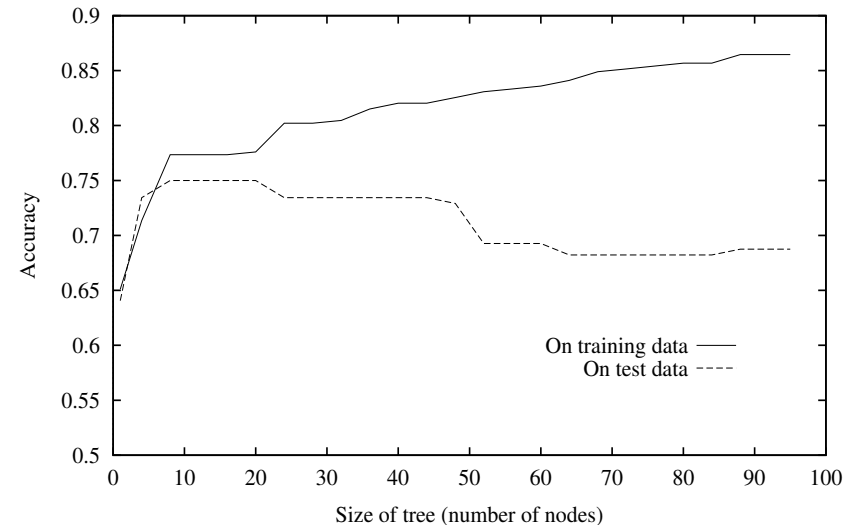
Hypothesis  $h \in H$  **overfits** training data if there is an alternative hypothesis  $h' \in H$  such that

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

and

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

## Overfitting in Decision Tree Learning



## Avoiding Overfitting

How can we avoid overfitting? **Pruning**

- **pre-pruning** stop growing when data split not statistically significant
- **post-pruning** grow full tree, then remove sub-trees which are overfitting

Post-pruning avoids problem of “early stopping”

How to select “best” tree:

- Measure performance over training data ?
- Measure performance over separate validation data set ?
- MDL: minimize  $size(tree) + size(misclassifications(tree))$  ?

## Avoiding Overfitting

Pre-pruning

- Usually based on statistical significance test
- Stops growing the tree when there is no statistically significant association between any attribute and the class at a particular node
- Most popular test: chi-squared test
- ID3: chi-squared test plus information gain
  - Only statistically significant attributes were allowed to be selected by information gain procedure

## Avoiding Overfitting

Early stopping

- Pre-pruning may suffer from early stopping: may stop the growth of tree prematurely
- Classic example: XOR/Parity-problem
  - No individual attribute exhibits a significant association with the class
  - Target structure only visible in fully expanded tree
  - Prepruning won't expand the root node
- But: XOR-type problems not common in practice
- And: pre-pruning faster than post-pruning

## Avoiding Overfitting

Post-pruning

- Builds full tree first and prunes it afterwards
  - Attribute interactions are visible in fully-grown tree
- Problem: identification of subtrees and nodes that are due to chance effects
- Two main pruning operations:
  - Subtree replacement
  - Subtree raising
- Possible strategies: error estimation, significance testing, MDL principle
- We examine two methods: Reduced-error Pruning and Error-based Pruning

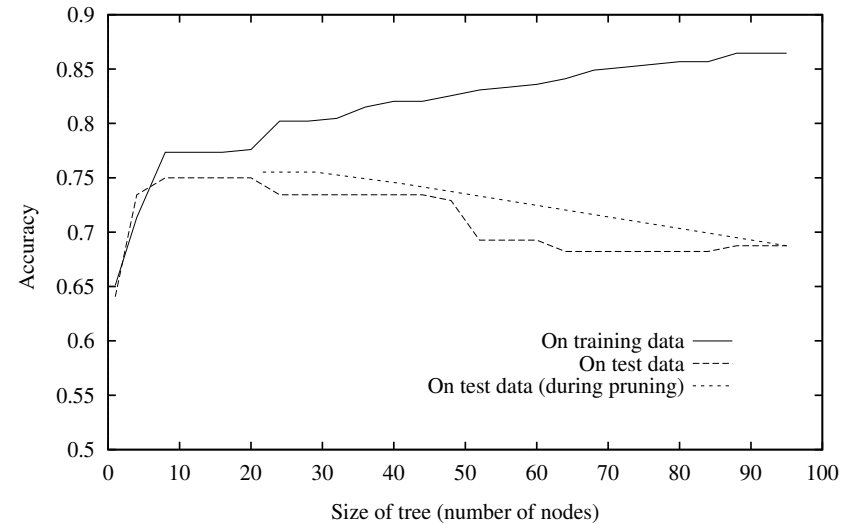
## Reduced-Error Pruning

Split data into *training* and *validation* set

Do until further pruning is harmful:

1. Evaluate impact on *validation* set of pruning each possible node (plus those below it)
  2. Greedily remove the one that most improves *validation* set accuracy
- **Good** produces smallest version of most accurate subtree
  - **Bad** reduces effective size of training set

## Effect of Reduced-Error Pruning



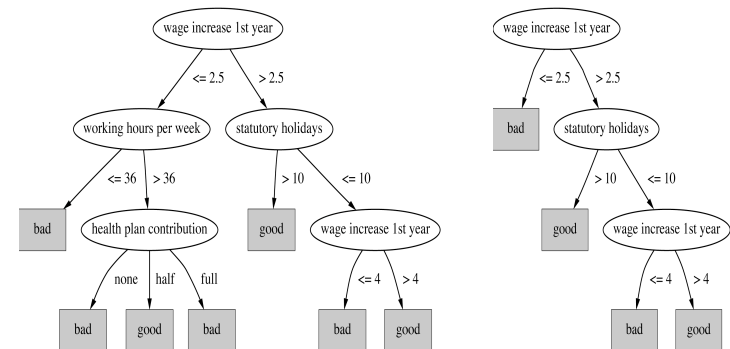
## Error-based pruning (C4.5)

Quinlan (1993) describes the successor to ID3 – C4.5

- many extensions – see below
- post-pruning using training set
- includes sub-tree replacement and sub-tree raising
- also: pruning by converting tree to rules
- commercial version – C5.0 – is widely used
  - go to RuleQuest.com

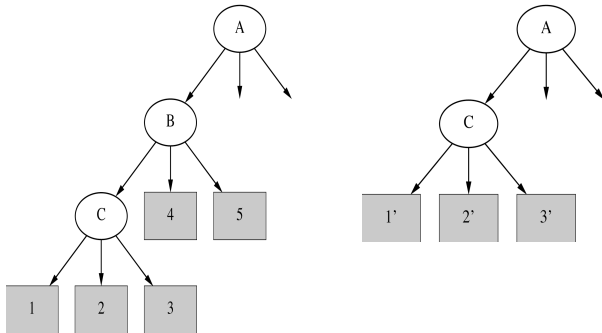
## Sub-tree replacement

Bottom-up: tree is considered for replacement once all its sub-trees have been considered



## Sub-tree raising

Deletes node and redistributes instances – more complicated, slow



COMP9417: March 18, 2009

Decision Tree Learning: Slide 48

## Error-based pruning

Goal is to improve estimate of error on unseen data using all and only data from training set

- Pruning operation is performed if this does not increase the estimated error
- C4.5's method: using upper limit of 25% confidence interval derived from the training data
  - Standard Bernoulli-process-based method
  - **Note:** statistically motivated but not statistically valid
  - **But:** works well in practice !

COMP9417: March 18, 2009

Decision Tree Learning: Slide 49

## Error-based pruning

- Error estimate for subtree is weighted sum of error estimates for all its leaves
- Error estimate for a node:

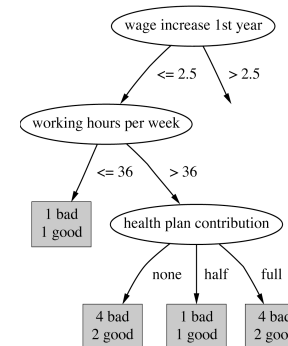
$$e = \frac{f + \frac{z^2}{2N} + z \sqrt{\frac{f}{N} - \frac{f^2}{N} + \frac{z^2}{4N^2}}}{1 + \frac{z^2}{N}}$$

- If  $c = 25\%$  then  $z = 0.69$  (from normal distribution)
- $f$  is the error on the training data
- $N$  is the number of instances covered by the leaf

COMP9417: March 18, 2009

Decision Tree Learning: Slide 50

## Error-based pruning



- health plan contribution: node measures  $f = 0.36, e = 0.46$
- sub-tree measures:
  - none:  $f = 0.33, e = 0.47$
  - half:  $f = 0.5, e = 0.72$
  - full:  $f = 0.33, e = 0.47$
- sub-trees combined 6 : 2 : 6 gives 0.51
- sub-trees estimated to give *greater* error so prune away

COMP9417: March 18, 2009

Decision Tree Learning: Slide 51

## Rule Post-Pruning

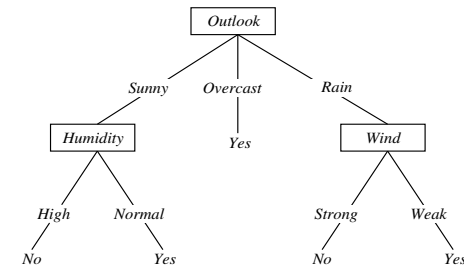
This method was introduced in Quinlan's C4.5

1. Convert tree to equivalent set of rules
2. Prune each rule independently of others
3. Sort final rules into desired sequence for use

**For:** simpler classifiers, people prefer rules to trees

**Against:** may not scale well, slow for large trees & datasets

## Converting A Tree to Rules



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

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

...

## Rules from Trees (Rule Post-Pruning)

Rules can be simpler than trees but just as accurate, e.g., in C4.5Rules:

- path from root to leaf in (unpruned) tree forms a *rule*
  - i.e., tree forms a *set of rules*
- can simplify rules independently by deleting conditions
  - i.e., rules can be generalized while maintaining accuracy
- greedy rule simplification algorithm
  - drop the condition giving lowest estimated error (as for pruning)
  - continue while estimated error does not increase

## Rules from Trees

Select a “good” subset of rules within a class (C4.5Rules):

- goal: remove rules not useful in terms of accuracy
- find a subset of rules which minimises an *MDL* criterion
- trade-off accuracy and complexity of rule-set
- stochastic search using simulated annealing

Sets of rules can be ordered by class (C4.5Rules):

- order classes by increasing chance of making *false positive* errors
- set as a default the class with the most training instances not covered by any rule

## Continuous Valued Attributes

Decision trees originated for **discrete** attributes only. Now: continuous attributes.

Can create a discrete attribute to test continuous value:

- $Temperature = 82.5$
- $(Temperature > 72.3) \in \{t, f\}$
- Usual method: continuous attributes have a binary split
- Note:
  - discrete attributes – one split exhausts all values
  - continuous attributes – can have many splits in a tree

## Continuous Valued Attributes

- Splits evaluated on all possible split points
- More computation:  $n - 1$  possible splits for  $n$  values of an attribute in training set
- Fayyad (1991)
  - sort examples on continuous attribute
  - find midway boundaries where class changes, e.g. for  $Temperature$   $\frac{(48+60)}{2}$  and  $\frac{(80+90)}{2}$
- Choose best split point by info gain (or evaluation of choice)
- Note: C4.5 uses actual values in data

<i>Temperature:</i>	40	48	60	72	80	90
<i>PlayTennis:</i>	No	No	Yes	Yes	Yes	No

## Attributes with Many Values

Problem:

- If attribute has many values,  $Gain$  will select it
- Why? more likely to split instances into “pure” subsets
- Imagine using  $Date = Jun\_3\_1996$  as attribute
- High gain on training set, useless for prediction

## Attributes with Many Values

One approach: use  $GainRatio$  instead

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

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

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

## Attributes with Many Values

Why does this help ?

- sensitive to how broadly and uniformly attribute splits instances
- actually the entropy of  $S$  w.r.t. values of  $A$
- therefore higher for many-valued attributes, especially if mostly uniformly distributed across possible values

## Attributes with Costs

Consider

- medical diagnosis, *BloodTest* has cost \$150
- robotics, *Width\_from\_1ft* has cost 23 sec.

How to learn a consistent tree with low expected cost?

## Attributes with Costs

One approach: replace gain by

- Tan and Schlimmer (1990)

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

- Nunez (1988)

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

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

## Attributes with Costs

Key idea: evaluate gain *relative to* cost, so prefer decision trees using lower-cost attributes.

More recently

- Domingos (1999) – MetaCost, a meta-learning wrapper approach
- uses ensemble learning method to estimate probabilities
- decision-theoretic approach

General problem: class costs, instance costs, ...

SEE5 / C5.0 can use costs ...

## Unknown Attribute Values

What if some examples missing values of  $A$ ?

Use training example anyway, sort through tree. Here are 3 possible approaches

- If node  $n$  tests  $A$ , assign most common value of  $A$  among other examples sorted to node  $n$
- assign most common value of  $A$  among other examples with same target value
- assign probability  $p_i$  to each possible value  $v_i$  of  $A$ 
  - assign fraction  $p_i$  of example to each descendant in tree

Note: need to classify new (unseen) examples in same fashion

## Windowing

Early implementations – training sets too large for memory

As a solution ID3 implemented *windowing*:

1. select subset of instances – the *window*
2. construct decision tree from all instances in the window
3. use tree to classify training instances *not* in window
4. if all instances correctly classified then halt, else
5. add selected misclassified instances to the window
6. go to step 2

Windowing retained in C4.5 because it can lead to *more accurate* trees. Related to *ensemble learning*.

## Summary

- Decision tree learning is a practical method for concept learning and other classifier learning tasks
- TDIDT family descended from ID3 searches complete hypothesis space - the hypothesis is there, somewhere...
- Uses a search or *preference* bias, search for optimal tree is not tractable
- Overfitting is inevitable with an expressive hypothesis space and noisy data, so pruning is important
- Decades of research into extensions and refinements of the general approach
- The “default” machine learning method, illustrates many general issues
- Can be updated with use of “ensemble” methods