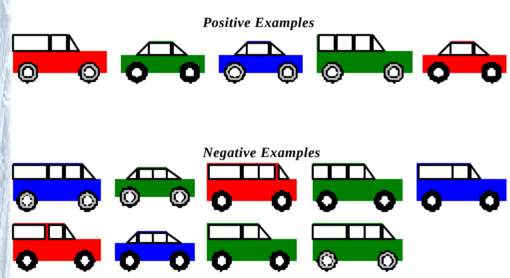
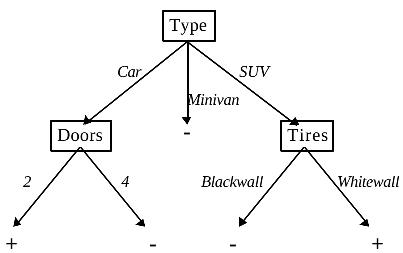
Decision Trees

- Decision tree representation
- ID3 learning algorithm
- Entropy, Information gain
- Overfitting

Another Example Problem



A Decision Tree



Decision Trees

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

How would you represent:
- $\wedge, \vee$, XOR
- $(A \wedge B) \vee (C \wedge \neg D \wedge E)$
- M of N

When to Consider Decision Trees

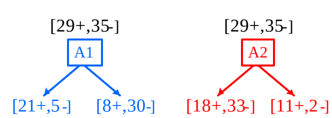
- Instances describable by attribute-value pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy training data

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

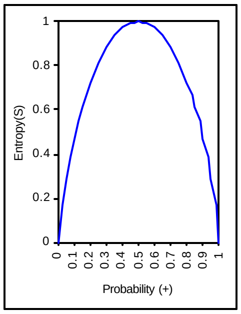
Top-Down Induction of Decision Trees

Main loop:
1. A = the "best" decision attribute for next *node*
2. Assign A as decision attribute for *node*
3. For each value of A, create descendant of *node*
4. Divide training examples among child nodes
5. If training examples perfectly classified, STOP
   Else iterate over new leaf nodes

Which attribute is best?



Entropy



- S = sample of training examples
- p_+ = proportion of positive examples in S
- p_- = proportion of negative examples in S
- Entropy measures the impurity of S

$$Entropy(S) \equiv -p_+ \log_2 p_+ - p_- \log_2 p_-$$

Entropy

Entropy(S) = expected number of bits need to encode class (+ or -) of randomly drawn member of *S* (using an 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 + or - of random member of S:

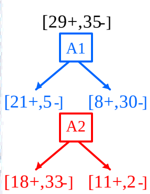
$$-p_+ \log_2 p_+ - p_- \log_2 p_-$$

$$Entropy(S) \equiv -p_+ \log_2 p_+ - p_- \log_2 p_-$$

Information Gain

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

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



$$Entropy([29+,35-]) = -\frac{29}{64}\log_2(\frac{29}{64}) - \frac{35}{64}\log_2\frac{35}{64} = 0.994$$

$$Entropy([21+,5-]) = -\frac{21}{26}\log_2(\frac{21}{26}) - \frac{5}{26}\log_2\frac{5}{26} = 0.706$$

$$Entropy([8+,30-]) = 0.742$$

$$Gain(S,A1) = 0.994 - (\frac{26}{64}Entropy([21+,5-]) + \frac{38}{64}Entropy([8+,30-])) = 0.266$$

$$Entropy([18+,33-]) = 0.937$$

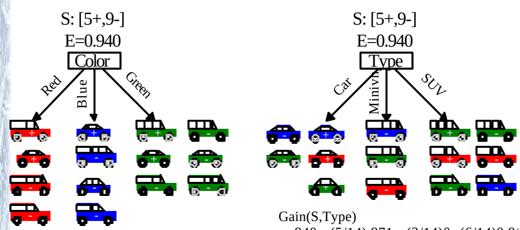
$$Entropy([11+,2-]) = 0.619$$

$$Gain(S,A2) = 0.121$$

Car Examples

Color	Type	Doors	Tires	Class
Red	SUV	2	Whitewall	+
Blue	Minivan	4	Whitewall	-
Green	Car	4	Whitewall	-
Red	Minivan	4	Blackwall	-
Green	Car	2	Blackwall	+
Green	SUV	4	Blackwall	-
Blue	SUV	2	Blackwall	-
Blue	Car	2	Whitewall	+
Red	SUV	2	Blackwall	-
Blue	Car	4	Blackwall	-
Green	SUV	4	Whitewall	+
Red	Car	2	Blackwall	+
Green	SUV	2	Blackwall	-
Green	Minivan	4	Whitewall	-

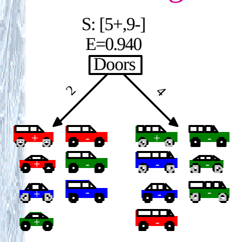
Selecting Root Attribute



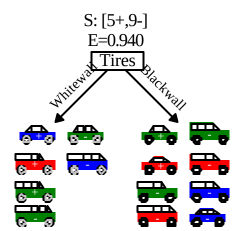
Gain(S,Color)
= .940 - (4/14)1.0 - (4/14).811 -(6/14).918
= 0.029

Gain(S,Type)
= .940 - (5/14).971 - (3/14)0 -(6/14)0.91
= 0.200

Selecting Root Attribute (cont)



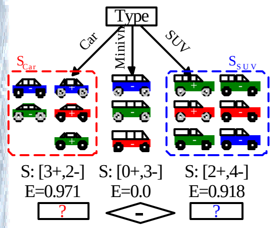
Gain(S,Doors)
= .940 - (7/14)0.985 - (7/14)0.592
= 0.152



Gain(S,Type)
= .940 - (6/14)1.0 - (8/14).811
= 0.048

Best attribute: *Type* (Gain = 0.200)

Selecting Next Attribute



Gain(S_{Car},Color) =
.971-(1/5)0.0-(2/5)1.0-(2/5)1.0=.171

Gain(S_{Car},Doors)=
.971-(3/5)0.0-(2/5)0.0=.971

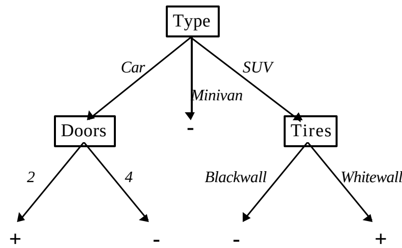
Gain(S_{Car},Tires)=
.971-(2/5)1.0-(3/5).918= .020

Gain(S_{SUV},Color)=
.918-(2/6)1.0-(1/6)0.0-(3/6).918= .126

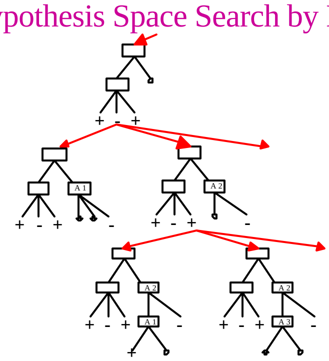
Gain(S_{SUV},Doors)=
.918-(4/6).811-(2/6)1.0= .044

Gain(S_{SUV},Tires)=
.918-(2/6)0.0-(4/6)0.0=.918

Resulting Tree



Hypothesis Space Search by ID3



Hypothesis Space Search by ID3

- Hypothesis space is complete!
  - Target function is in there (but will we find it?)
- Outputs a single hypothesis (which one?)
  - Cannot play 20 questions
- No back tracing
  - Local minima possible
- Statistically-based search choices
  - Robust to noisy data
- Inductive bias: approximately "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*
- Occam's razor: prefer the shortest hypothesis that fits the data

Occam's Razor

Why prefer short hypotheses?

Argument in favor:

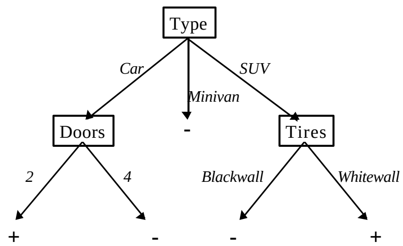
- Fewer short hypotheses than long hypotheses
- short hyp. that fits data unlikely to be coincidence
- long hyp. that fits data more likely to be coincidence

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 is so special about small sets based on size of hypothesis?

Overfitting in Decision Trees

Consider adding a noisy training example:
<Green,SUV,2,Blackwall> +
What happens to decision tree below?



Overfitting

Consider error of hypothesis h over

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

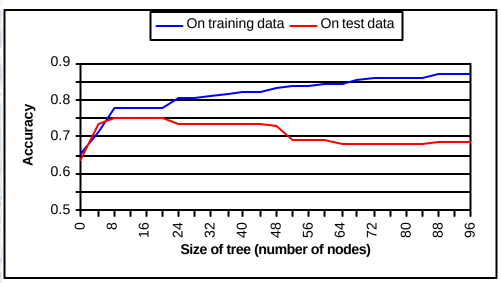
Hypothesis $h \in H$ **overfits** the training data if there is an alternative hypothesis $h' \in H$ such that

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

and

$$error_D(h) > error_D(h')$$

Overfitting in Decision Tree Learning



Avoiding Overfitting

How can we avoid overfitting?

- stop growing when data split not statistically significant
- grow full tree, the post-prune

How to select "best" tree:

- Measure performance over training data
- Measure performance over separate validation set (examples from the training set that are put aside)
- MDL: minimize

  size(tree) + size(misclassifications(tree))

Reduced-Error Pruning

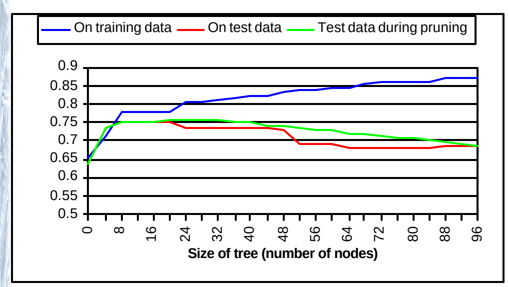
Split data into *training* and *validation* set

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

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

Effect of Reduced-Error Pruning



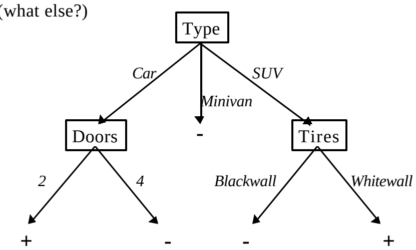
Rule Post-Pruning

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

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

Converting a Tree to Rules

IF (Type=Car) AND (Doors=2) THEN +
IF (Type=SUV) AND (Tires=Whitewall) THEN +
IF (Type=Minivan) THEN -
… (what else?)



Continuous Valued Attributes

Create one (or more) corresponding discrete attributes based on continuous

- (EngineSize = 325) = true or false
- (EngineSize <= 330) = t or f (330 is "split" point)

How to pick best "split" point?

1. Sort continuous data
2. Look at points where class differs between two values
3. Pick the split point with the best gain

Why this one?

EngineSize:	285	290	295	310	330	330	345	360
Class:	-	-	+	+	-	+	+	+

Attributes with Many Values

Problem:

- If attribute has many values, *Gain* will select it
- Imagine if cars had *PurchaseDate* feature - likely all would be different

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 Costs

Consider

- medical diagnosis, *BloodTest* has cost $150
- robotics, *Width_from_1ft* has cost 23 second

How to learn consistent tree with low expected cost?

Approaches: replace gain by

Tan and Schlimmer (1990) $\frac{Gain^2(S,A)}{Cost(A)}$

Nunez (1988)

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

Unknown Attribute Values

What if some examples missing values of *A*?

"?" in C4.5 data sets

Use training example anyway, sort through tree

- 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

Classify new examples in same fashion

Decision Tree Summary

- simple (easily understood), powerful (accurate)
- highly expressive (complete hypothesis space)
- bias: restrictive
  - search based on information gain (defined using entropy)
  - favors short hypotheses, high gain attributes near root
- issues:
  - overfitting
    - avoiding: stopping early, pruning
    - pruning: how to judge, what to prune (tree, rules, etc.)

Decision Tree Summary (cont)

- issues (cont):
  - attribute issues
    - continuous valued attributes
    - attributes with lots of values
    - attributes with costs
    - unknown values
- effective for discrete valued target functions
- handles noise