Decision Tree Learning (DTL)

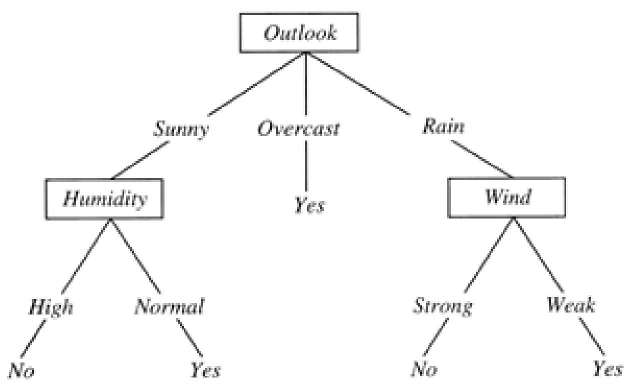
- decision tree representation

. Decision tree learning is a method for *approximating discrete-valued target functions* in which the learned function is represented by *a decision tree.*
The decision tree can be easily represented by *if-then rules* to improve human readability.

. In decision tree, *each internal node tests an attribute,*
each branch corresponds to attribute value, and
each leaf node assigns a classification.

. In general, decision tree represent *a disjunction of conjunctions of constraints* on the attribute values of instances.

example. $(Outlook = Sunny \wedge Humidity = Normal) \vee$
$(Outllook = Overcast) \vee$
$(Outlook = Rain \wedge Wind = Weak)$



- appropriate problems of DTL
  . instances described 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,
  spam-mail filtering, etc.

- learning algorithms of DTL
  CART (Friedman 1977; Breiman et al. 1984)
  ID3 (Quinlan, 1979, 1983), C4.5 (Quinlan, 1993)

- ID3 algorithm

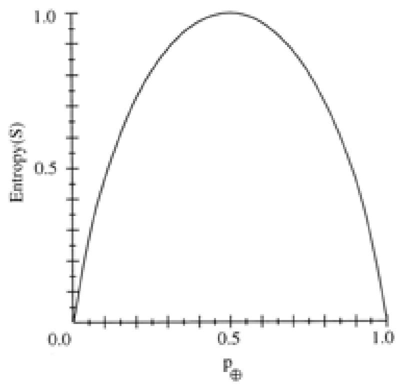
Step 1. Create a root node for the tree that
best classifies examples.
Step 2. Do the following procedure:
(1) $A \leftarrow$ the best decision attribute for the next node.
(2) assign A as decision attribute for the node.
(3) for each value of A, create new descendant of node.
(4) sort training examples to leaf node.
(5) If training examples perfectly classified, then stop.
else, iterate over new leaf nodes.

What is *the best decision attribute* for the root node and other nodes?

- entropy
  . S is a sample of training examples.
  . p_+ is the probability of positive examples in S.
  . p_- is the probability of negative examples in S.
  . entropy of S is described by

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

$Entropy(S)$ represents *the number of bits needed to encode class* (+ or -) of randomly drawn number of S (under the optimal shortest-length code).



More generally, if the target attribute can take on c different values,

$$Entropy(S) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i$$

- information gain

. Information gain describes *the expected reduction in entropy due to sorting on attribute* A, that is,

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

where $Values(A)$ represents the set of all possible values for attribute A and S_v represents the subset of S for which attribute A has value v.

Training examples for the target concept PlayTennis

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

$$
\begin{aligned}
Entropy(S) &= Entropy([9_+, 5_-]) \\
&= -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) \\
&= 0.940
\end{aligned}
$$

$$
\begin{aligned}
Gain(S, Wind) &= Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} Entropy(S_v) \\
&= Entropy(S) - \frac{8}{14} Entropy(S_{Weak}) - \frac{6}{14} Entropy(S_{Strong}) \\
&= 0.940 - \frac{8}{14} Entropy([6_+, 2_-]) - \frac{6}{14} Entropy([3_+, 3_-]) \\
&= 0.940 - \frac{8}{14} 0.811 - \frac{6}{14} 1.00 \\
&= 0.048
\end{aligned}
$$

Similarly,

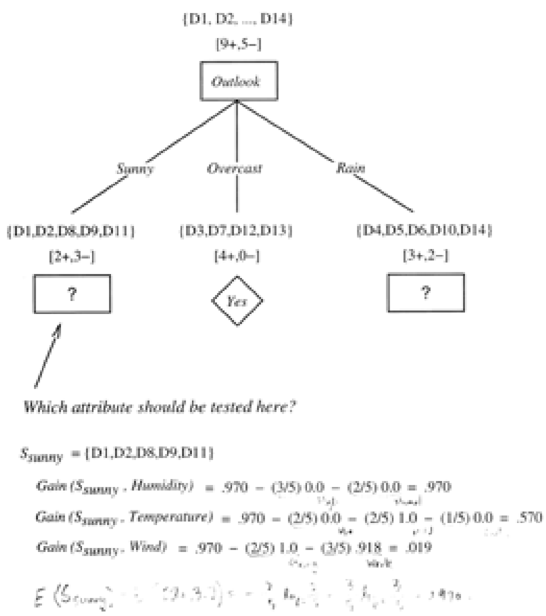
$Gain(S, Outlook) = 0.246$,

$Gain(S, Humidity) = 0.151$, and

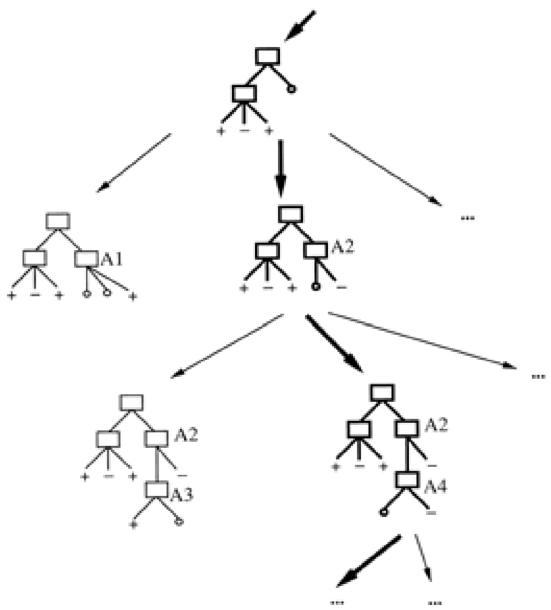
$Gain(S, Temperature) = 0.029$.

Therefore, the $Outlook$ attribute provides the best prediction of the target concept PlayTennis over training examples.

DTL by ID3



Hypothesis Space Search by ID3



- hypothesis space search by ID3
  . The hypothesis space H is *complete*, that is, the target function is surely in H.
  . ID3 generates *a single hypothesis*.
  . No backtracking, that is, ID3 generates *a locally optimal solution* corresponding to the decision tree.
  . Statistically-based search using the information gain - as a result, it is *robust to noisy data*.
- inductive bias in ID3
  . *preference for short trees* and for those with high information gain attributes near the root.
  cf. Occam's razor: preference to the shortest hypothesis that fits the data.

- overfitting of ID3

  . error of hypothesis h over training data T :
  $error_T(h) \equiv \Pr_{x \in T}[c(x) \neq h(x)]$
  . error of hypothesis h over entire distribution D of data:
  $error_D(h) \equiv \Pr_{x \in D}[c(x) \neq h(x)]$
  . The hypothesis $h \in H$ overfits training data T if there is an alternative hypothesis $h^{'} \in H$ such that
  $error_T(h) < error_T(h^{'})$ and
  $error_D(h) > error_D(h^{'})$.

- methods to avoid overfitting

. stop growing when data split not statistically significant.
. grow full tree, then post-prune.
. selecting the best tree:
  (1) measure performance over training data.
  (2) measure performance over separate validation set.
  (3) minimize the size of tree and the misclassification of tree.

. reduced-error pruning

  (1) split data into training and validation sets.
  (2) do the following procedure until further pruning is harmful:
      1) evaluate impact on validation set of pruning
         each node (plus those below it).
      2) greedily remove the one that most improves
         validation set accuracy.

. rule post-pruning (C4.5)

(1) grow the tree until the training data are fit
as well as possible.
(2) convert the tree to equivalent set of (if-then) rules.
(3) prune each rule that results in improving
its estimated accuracy.
(4) sort the pruned rules by their estimated accuracy.

- improving ID3

. continuous valued attributes: dynamically defining
new discrete-valued attributes that partition the continuous value
into a discrete set of intervals.
One of the methods is picking a threshold that produces
the greatest information gain.

. attributes with many values: so many possible values are
bounded to separate the training examples into very small
subsets which results in high information gain compare to
other training examples in large sunsets.

One of methods is to use gain ratio instead of information gain (Quinlan, 1986):

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

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

where S_1 through S_c are the subsets of examples resulting from partitioning S by the c-valued attribute A.

example. n examples and attribute A has 2 values.
Suppose we have 2 subsets and each subset has n/2 examples.

$$SplitInfo(S, A) = -(\frac{1}{2}\log_2\frac{1}{2} + \frac{1}{2}\log_2\frac{1}{2}) = 1$$

example. n examples and attribute A has n values.
Suppose we have n subsets and each subset has 1 example.

$$SplitInfo(S, A) = -\sum_{i=1}^{n} \frac{1}{n}\log_2\frac{1}{n} = \log_2 n$$

. attributes with costs: learn a consistent tree with low expected cost. Each attribute may have associated cost according to the learning task.

In this case, $Gain(S, A)$ can be replaced by

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

$$\frac{2^{Gain(S, A)} - 1}{(Cost(A)+1)^w}$$ (Nunez, 1988)

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

Reference: Machine Learning, chapter 3.