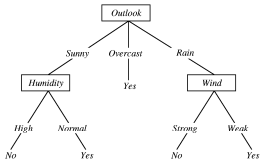

Decision Tree Learning

Chapter 18.3

Decision Trees

- One of the most widely used and practical methods for inductive inference
- Approximates discrete-valued functions (including disjunctions)
- Can be used for classification

Decision Tree



```

graph TD
    Outlook[Outlook] -- Sunny --> Humidity[Humidity]
    Outlook -- Overcast --> Yes[Yes]
    Outlook -- Rain --> Wind[Wind]
    Humidity -- High --> No1[No]
    Humidity -- Normal --> Yes1[Yes]
    Wind -- Strong --> No2[No]
    Wind -- Weak --> Yes2[Yes]
  
```

- A decision tree can represent a disjunction of conjunctions of constraints on the attribute values of instances.
 - Each path corresponds to a conjunction
 - The tree itself corresponds to a disjunction

If (O=Sunny AND H=Normal) OR (O=Overcast) OR (O=Rain AND W=Weak) then YES

Top-Down Induction of Decision Trees

```

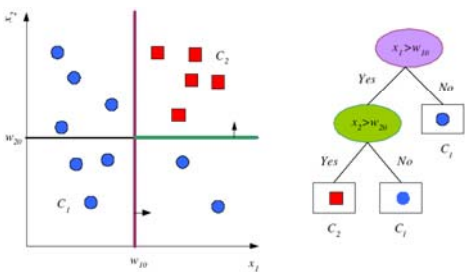
function DTL(examples, attributes, default) returns a decision tree
  if examples is empty then return default
  else if all examples have the same classification then return the classification
  else if attributes is empty then return MODE(examples)
  else
    best ← CHOOSE-ATTRIBUTE(attributes, examples)
    tree ← a new decision tree with root test best
    for each value vi of best do
      examplesi ← {elements of examples with best = vi}
      subtree ← DTL(examplesi, attributes - best, MODE(examplesi))
      add a branch to tree with label vi and subtree subtree
    return tree
  
```

Decision tree representation

- Each internal node corresponds to a test
- Each branch corresponds to a result of the test
- Each leaf node assigns a classification

Once the tree is trained, a new instance is classified by starting at the root and following the path as dictated by the test results for this instance.

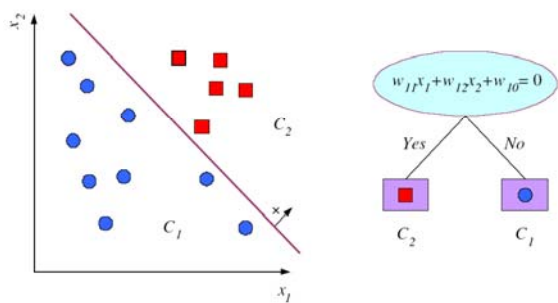
Tree Uses Nodes, and Leaves



Divide and Conquer

- Internal decision nodes
 - Univariate: Uses a single attribute, x_j
 - Numeric x_j : Binary split : $x_j > w_m$
 - Discrete x_j : n -way split for n possible values
 - Multivariate: Uses all attributes, \mathbf{x}
- Leaves
 - Classification: Class labels, or proportions
 - Regression: Numeric; r average, or local fit
- The learning algorithm is greedy; find the best split recursively

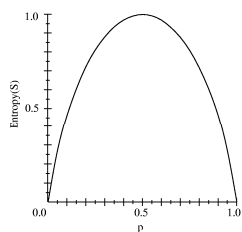
Multivariate Trees



*** Entropy**

- “Measure of uncertainty”
- “Expected number of bits to resolve uncertainty”
- Suppose $\Pr\{X = 0\} = 1/8$
 - If other events are equally likely, the number of events is 8. To indicate one out of so many events, one needs $\lg 8$ bits.
- Consider a binary random variable X s.t. $\Pr\{X = 0\} = 0.1$.
 - The expected number of bits: $0.1 \times \lg \frac{1}{0.1} + (1-0.1) \times \lg \frac{1}{(1-0.1)}$
- In general, if a random variable X has c values with prob. p_c :
 - The expected number of bits: $H = \sum_{i=1}^c p_i \lg \frac{1}{p_i} = -\sum_{i=1}^c p_i \lg p_i$

*** Entropy of a binary variable**



$$H(p) = -p \lg p - (1-p) \lg (1-p)$$

Information gain

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

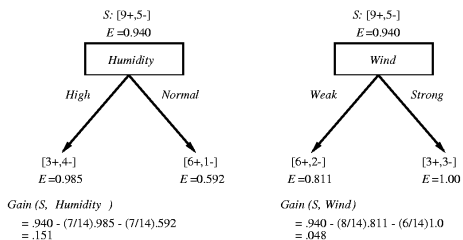
$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Training Examples

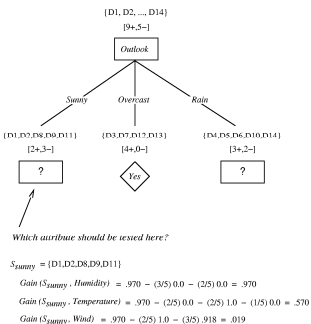
Day	Outlook	Temperature	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal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

Selecting the Next Attribute

Which attribute is the best classifier?

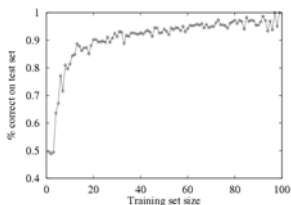


Partially learned tree



Performance measurement

- How do we know that $h \approx f$?
 - Use theorems of computational/statistical learning theory
 - Try h on a new test set of examples
(use same distribution over example space as training set)
- Learning curve = % correct on test set as a function of training set size



Why Learning Works

- There is a theoretic foundation: Computational Learning Theory.
- The underlying principle: Any hypothesis that is consistent with a sufficiently large set of training examples is unlikely to be seriously wrong: it must be **Probably Approximately Correct (PAC)**.
- The **Stationarity** Assumption: The training and test sets are drawn randomly from the same population of examples using the same probability distribution.

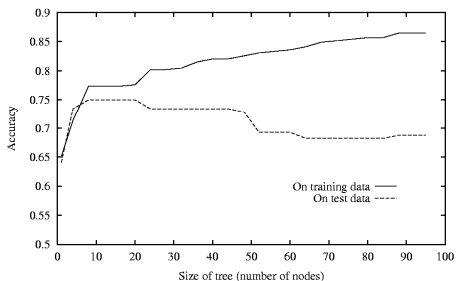
Occam's razor

- Prefer the simplest hypothesis that fits the data
- Support 1
 - Possibly because shorter hypotheses have better generalization ability
- Support 2
 - The number of short hypotheses are small, and therefore it is less likely a coincidence if data fit a short hypothesis

Over fitting in Decision Trees

- Why "over"-fitting?
 - A model can become more complex than the true target function(concept) when it tries to satisfy noisy data as well.
- Definition of overfitting
 - A hypothesis is said to overfit the training data if there exists some other hypothesis that has larger error over the training data but smaller error over the entire instances.

Over fitting in Decision Trees



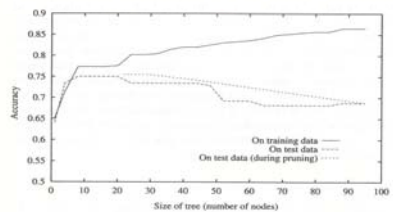
Avoiding over-fitting the data

- How can we avoid overfitting? There are 2 approaches:
 - stop growing the tree before it perfectly classifies the training data
 - grow full tree, then post-prune
 - Reduced error pruning
 - Rule post-pruning
 - The 2nd approach is found more useful in practice.

- Ok, but how to determine the optimal size of a tree?
 - Use validation examples to evaluate the effect of pruning (stopping)
 - Use a statistical test to estimate the effect of pruning (stopping)
 - ...

Reduced error pruning

- Examine each decision node to see if pruning decreases the tree's performance over the evaluation data.
- "Pruning" here means replacing a subtree with a leaf with the most common classification in the subtree.

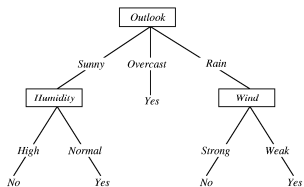


Rule post-pruning

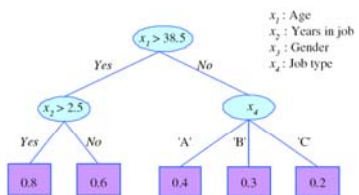
- Algorithm
 - Build a complete decision tree.
 - Convert the tree to set of rules.
 - Prune each rule:
 - Remove any preconditions if any improvement in accuracy
 - Sort the pruned rules by accuracy and use them in that order.

- This is the most frequently used method

- IF (*Outlook = Sunny*) ^ (*Humidity = High*)
THEN *PlayTennis = No*
- IF (*Outlook = Sunny*) ^ (*Humidity = Normal*)
THEN *PlayTennis = Yes*
- ...



Rule Extraction from Trees



- R1: IF (age > 38.5) AND (years-in-job > 2.5) THEN $y = 0.8$
- R2: IF (age > 38.5) AND (years-in-job ≤ 2.5) THEN $y = 0.6$
- R3: IF (age ≤ 38.5) AND (job-type='A') THEN $y = 0.4$
- R4: IF (age ≤ 38.5) AND (job-type='B') THEN $y = 0.3$
- R5: IF (age ≤ 38.5) AND (job-type='C') THEN $y = 0.2$

Split Information?

- Which is better?
 - In terms of information gain
 - In terms of gain ratio

Diagram illustrating two splits, A1 and A2, based on 100 examples.

Split A1 results in two branches: 40 positive / 40 negative and 20 positive / 20 negative.

Split A2 results in two main branches: 10 positive and 10 negative, with multiple unlabeled branches indicated by ellipses.

Attributes with Many Values

- One way to penalize such attributes is to use the following alternative measure:

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

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

Entropy of the attribute A:
Experimentally determined by the training samples

Handling training examples with missing attribute values

- What if an example x is missing the value an attribute A?
- Simple solution:
 - Use the most common value among examples at node n.
 - Or use the most common value among examples at node n that have classification c(x).
- More complex, probabilistic approach
 - Assign a probability to each of the possible values of A based on the observed frequencies of the various values of A
 - Then, propagate examples down the tree with these probabilities.
 - The same probabilities can be used in classification of new instances

Handling attributes with differing costs

- Sometimes, some attribute values are more expensive or difficult to prepare.
 - medical diagnosis, BloodTest has cost \$150
- In practice, it may be desired to postpone acquisition of such attribute values until they become necessary.
- To this purpose, one may modify the attribute selection measure to penalize expensive attributes.

- Tan and Schlimmer (1990) $\frac{Gain^2(S, A)}{Cost(A)}$
- Nunez (1988) $\frac{2^{Gain(S,A)} - 1}{(Cost(A) + 1)^w}, w \in [0, 1]$

Summary

- Learning needed for unknown environments, lazy designers
- Learning agent = performance element + learning element
- For supervised learning, the aim is to find a simple hypothesis approximately consistent with training examples
- Decision tree learning using information gain
- Learning performance = prediction accuracy measured on test set

Basic Procedures

1. Collect randomly a large set of examples
2. Choose randomly a subset of the examples as **training set**
3. Apply the learning algorithm to the training set, generating a hypothesis h.
4. Measure the percentage of examples in the whole set that are correctly classified by h.
5. Repeat steps 1-4 for different sizes of training sets if the performance is not satisfactory.
