Decision trees and rule-based classifiers
Decision tree: An example

- Idea: Ask a sequence of questions (as in the ‘20 questions’ game) to infer the class
Decision tree: A second example

There can be many different trees that all work equally well!
Decision tree: Structure

Structure of the tree:

- A single root node with no incoming edges, and zero or more outgoing edges
- Internal nodes, each of which has exactly one incoming edge and two or more outgoing edges
- Leaf or terminal nodes, each of which has exactly one incoming edge and no outgoing edges

Node contents:

- Each terminal node is assigned a prediction (here, for simplicity: a definite class label).
- Each non-terminal node defines a test, with the outgoing edges representing the various possible results of the test (here, for simplicity: a test only involves a single attribute)
Notation: In this figure \( x \) and \( y \) are two continuous-valued attributes (i.e. \( y \) is not the class label in this figure!)

Decision boundary consists of parts which all are parallel to the axes because each decision depends only on a single attribute
Learning a decision tree from data: General idea

- Simple idea: Recursively divide up the space into pieces which are as \textit{pure} as possible.
Learning a decision tree from data: General idea

- Simple idea: Recursively divide up the space into pieces which are as pure as possible
Learning a decision tree from data: General idea

- Simple idea: Recursively divide up the space into pieces which are as *pure* as possible.
Learning a decision tree from data: General idea

- Simple idea: Recursively divide up the space into pieces which are as *pure* as possible
Learning a decision tree from data: General idea

- Simple idea: Recursively divide up the space into pieces which are as *pure* as possible
Learning a decision tree from data: Hunt’s algorithm

- **Notation:** Let $D_t$ denote the set of records corresponding to node $t$. For the root node, $D_t$ is the set of all training data.

- **Hunt’s algorithm:**
  1. If all the records in $D_t$ belong to the same class $y_t$, then $t$ is a leaf node labeled as $y_t$
  2. If $D_t$ contains records that belong to more than one class, *select an attribute test condition* that partitions the records into smaller subsets. Create a child node for each outcome and distribute the records in $D_t$ to the children. Apply the algorithm recursively to each child node.

   - If $D_t$ is an empty set $\Rightarrow$ use majority vote among parent records
   - All records in $D_t$ are identical, but labels not $\Rightarrow$ use majority vote

This general method is also known as *TDIDT* (Top-Down Induction of Decision Trees).
Attribute test conditions

- Binary attributes: yes / no (two children only)

- Nominal (categorical) attributes with $L$ states:
  - Multiway split ($L$ children)
  - Binary split (2 children, any of the $2^{L-1} - 1$ ways of splitting)

- Ordinal attributes with $L$ states:
  - Multiway or binary split
  - Must respect the ordering (only combine contiguous values)

- Continuous attributes:
  - Multiway or binary split
  - Defined using breakpoints
Impurity measures

How do we measure whether a subset of the records is ‘pure’ or ‘impure’?

> Denote by $p(i \mid t)$ the fraction of records belonging to class $i$ of all the records at node $t$.

> Impurity measures:

\[
\begin{align*}
\text{Entropy}(t) &= - \sum_{i=0}^{K-1} p(i \mid t) \log p(i \mid t) \\
\text{Gini}(t) &= 1 - \sum_{i=0}^{K-1} p(i \mid t)^2 \\
\text{Classification error}(t) &= 1 - \max_i p(i \mid t)
\end{align*}
\]

where $K$ is the total number of classes.
Impurity measures: Binary classification

Qualitatively, the three measures agree. However, some differences in the selection of test attributes do occur.
Selecting the best split

- Test all valid splits

- Select the split which maximizes the gain $\Delta$:

\[
\Delta = I(\text{parent}) - \sum_{j=1}^{k} \frac{N(v_j)}{N} I(v_j),
\]

where

- $I(\cdot)$ is the impurity of a given node
- $N$ is the total number of records of the parent
- $N(v_j)$ is the number of records of child $v_j$

- Note: Essentially just minimizing a weighted sum of the impurities of the children in the split
Computational issues:

- Binary attributes: Just one split to test per attribute
- Discrete attributes: Finite number of possible splits to test per attribute
- Continuous attributes: In principle, any value $x \in R$ could be chosen as a breakpoint, but we only need to test midpoints between adjacent points. Computationally feasible by first sorting (complexity $O(N \log N)$) the records, then stepping through in order while updating the necessary statistics.
Example: Web Robot Detection

<table>
<thead>
<tr>
<th>Session</th>
<th>IP Address</th>
<th>Timestamp</th>
<th>Request Method</th>
<th>Requested Web Page</th>
<th>Protocol</th>
<th>Status</th>
<th>Number of Bytes</th>
<th>Referrer</th>
<th>User Agent</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>160.11.11.11</td>
<td>08/Aug/2004 10:15:21</td>
<td>GET</td>
<td><a href="http://www.cs.umn.edu/~kumar">http://www.cs.umn.edu/~kumar</a></td>
<td>HTTP/1.1</td>
<td>200</td>
<td>6424</td>
<td></td>
<td>Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)</td>
</tr>
<tr>
<td>1</td>
<td>160.11.11.11</td>
<td>08/Aug/2004 10:15:34</td>
<td>GET</td>
<td><a href="http://www.cs.umn.edu/~kumar/MINDS">http://www.cs.umn.edu/~kumar/MINDS</a></td>
<td>HTTP/1.1</td>
<td>200</td>
<td>41378</td>
<td><a href="http://www.cs.umn.edu/~kumar">http://www.cs.umn.edu/~kumar</a></td>
<td>Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)</td>
</tr>
<tr>
<td>1</td>
<td>160.11.11.11</td>
<td>08/Aug/2004 10:15:41</td>
<td>GET</td>
<td><a href="http://www.cs.umn.edu/~kumar/MINDS/MINDS_papers.htm">http://www.cs.umn.edu/~kumar/MINDS/MINDS_papers.htm</a></td>
<td>HTTP/1.1</td>
<td>200</td>
<td>1018516</td>
<td><a href="http://www.cs.umn.edu/~kumar/MINDS">http://www.cs.umn.edu/~kumar/MINDS</a></td>
<td>Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)</td>
</tr>
<tr>
<td>1</td>
<td>160.11.11.11</td>
<td>08/Aug/2004 10:16:11</td>
<td>GET</td>
<td><a href="http://www.cs.umn.edu/~kumar/papers/papers.html">http://www.cs.umn.edu/~kumar/papers/papers.html</a></td>
<td>HTTP/1.1</td>
<td>200</td>
<td>7463</td>
<td><a href="http://www.cs.umn.edu/~kumar">http://www.cs.umn.edu/~kumar</a></td>
<td>Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.0)</td>
</tr>
<tr>
<td>2</td>
<td>35.9.2.2</td>
<td>08/Aug/2004 10:16:15</td>
<td>GET</td>
<td><a href="http://www.cs.umn.edu/~steinbac">http://www.cs.umn.edu/~steinbac</a></td>
<td>HTTP/1.0</td>
<td>200</td>
<td>3149</td>
<td></td>
<td>Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.7) Gecko/20040616</td>
</tr>
</tbody>
</table>

(a) Example of a Web server log.

(b) Graph of a Web session.

(c) Derived attributes for Web robot detection.
Example: Web Robot Detection

- Resulting decision tree:

  (more details in the book)

![Decision Tree Diagram]

Figure 4.18. Decision tree model for Web robot detection.
Characteristics of Decision Tree Induction

- **Nonparametric** approach:
  - Can in the limit approximate *any* decision boundary to arbitrary precision
  
  \[ \Rightarrow \text{Approaches optimal performance (i.e. Bayesian classifier with known distributions) in the infinite sample limit} \]

  ...but requires *regularization* to avoid overlearning.
Characteristics of Decision Tree Induction

- **Nonparametric** approach:
  - Can in the limit approximate *any* decision boundary to arbitrary precision

  ⇒ Approaches optimal performance (i.e. Bayesian classifier with known distributions) in the infinite sample limit

  ...but requires *regularization* to avoid overlearning.
Characteristics of Decision Tree Induction

- **Nonparametric** approach:
  - Can in the limit approximate *any* decision boundary to arbitrary precision
  - \(\Rightarrow\) Approaches optimal performance (i.e. Bayesian classifier with known distributions) in the infinite sample limit
  - ...but requires *regularization* to avoid overlearning.
Characteristics of Decision Tree Induction

- **Nonparametric** approach:
  - Can in the limit approximate *any* decision boundary to arbitrary precision
  - \( \Rightarrow \) Approaches optimal performance (i.e. Bayesian classifier with known distributions) in the infinite sample limit
  - ... but requires *regularization* to avoid overlearning.

![Diagram of decision tree boundaries](image)
Characteristics of Decision Tree Induction

- **Nonparametric** approach:
  - Can in the limit approximate *any* decision boundary to arbitrary precision

  ⇒ Approaches optimal performance (i.e. Bayesian classifier with known distributions) in the infinite sample limit

...but requires *regularization* to avoid overlearning.
Characteristics of Decision Tree Induction

- **Nonparametric** approach:
  - Can in the limit approximate *any* decision boundary to arbitrary precision

  ⇒ Approaches optimal performance (i.e. Bayesian classifier with known distributions) in the infinite sample limit

  ...but requires *regularization* to avoid overlearning.
Characteristics of Decision Tree Induction

- **Nonparametric** approach:
  - Can in the limit approximate *any* decision boundary to arbitrary precision

  ⇒ Approaches optimal performance (i.e. Bayesian classifier with known distributions) in the infinite sample limit

  ...but requires *regularization* to avoid overlearning.
Characteristics of Decision Tree Induction

- **Nonparametric** approach:
  - Can in the limit approximate *any* decision boundary to arbitrary precision
  
  ⇒ Approaches optimal performance (i.e. Bayesian classifier with known distributions) in the infinite sample limit
  
  ...but requires *regularization* to avoid overlearning.
Characteristics of Decision Tree Induction

- **Nonparametric** approach:
  - Can in the limit approximate *any* decision boundary to arbitrary precision
  
  ⇒ Approaches optimal performance (i.e. Bayesian classifier with known distributions) in the infinite sample limit
  
  ...but requires *regularization* to avoid overlearning.
Characteristics of Decision Tree Induction (cont.)

- Local, greedy learning to find a reasonable solution in reasonable time (as opposed to finding a globally optimal solution)

- Relatively easy to interpret (by experts or regular users of the system)
  - cf. ”Why was I recommended this?” on amazon.com (helps build confidence in the system)
Characteristics of Decision Tree Induction (cont.)

- Data fragmentation problem:
  - The nodes far down the tree are based on a very small fraction of the data, even only on a few data points ⇒ typically not very reliable information
  - Example: Divide any of the existing leaves into purer partitions:

```
x_1 > 2.8
x_2 > 1.2
x_2 > 2.0
x_1 > 4.3
```
Rule-based classifier

- e.g. Mac OS X ‘Mail’ application:
Rule-based classifier: Example

Example:

- $r_1$: ('webmail') $\cap$ ('give password') $\rightarrow$ spam
- $r_2$: ('important') $\cap$ (sender = teacher) $\rightarrow$ not spam
- $r_3$: ('viagra') $\cap$ ('extra strength') $\rightarrow$ spam
- $r_4$: ('millions') $\cap$ ('netflix challenge') $\rightarrow$ not spam
- $r_5$: ('you have won') $\rightarrow$ spam

Idea:

'Many local classification models make up one global classifier'

If we can find reliable rules of the form $r_1, \ldots, r_n$ we can build a working classifier
Rule-based classifier: Definition

▲ Rule set \( R = \{r_1, \ldots, r_n\} \)

▲ Each rule \( r_i \) is of the form

\[
 r_i : (\text{Condition}_i) \rightarrow y_i, \tag{19}
\]

where the left-hand-side is the \textit{rule antecedent} (a logical expression that evaluates to true or false depending on the datapoint) and the right-hand-side is the \textit{rule consequence}, i.e. a prediction (here, for simplicity, one of the classes).

▲ A rule \( r_i \) is said to \textit{cover} a record if its rule antecedent evaluates to true. Conversely, the record is said to \textit{trigger} the rule.
Properties of a single rule $r_i$

- The *coverage* of a rule is the fraction of records (in the training data) which it covers.
- The *accuracy* of a rule is the fraction of covered records for which the rule consequence equals the true class.

For instance, if the rule

$$r_5: \text{"you have won"} \rightarrow \text{spam}$$

has a coverage of 0.1 and an accuracy of 0.85, it means that 10% of all emails received included the phrase ‘you have won’, and out of all those emails 85% were truly spam and 15% were non-spam.
Properties of the rule set $R$ as a whole

- The rules in rule set $R$ are *mutually exclusive* if no two rules can be triggered by any single record.

- The rules in rule set $R$ are *exhaustive* if at least one rule is triggered by any record.

(Note that we are here considering *any possible record*, not just the records in the training dataset.)

- Together, the two properties imply that any record can be classified into a unique class. However, not all rule sets $R$ have these properties.
Example

The rules:

\[ r_1: \text{('webmail') } \cap \text{('give password')} \rightarrow \text{spam} \]
\[ r_2: \text{('important') } \cap \text{(sender = teacher)} \rightarrow \text{not spam} \]
\[ r_3: \text{('viagra') } \cap \text{('extra strength')} \rightarrow \text{spam} \]
\[ r_4: \text{('millions') } \cap \text{('netflix challenge')} \rightarrow \text{not spam} \]
\[ r_5: \text{('you have won')} \rightarrow \text{spam} \]

are not mutually exclusive because the email: “you have won the netflix challenge, please come to the bank tomorrow to collect your millions” triggers both rule \( r_4 \) and \( r_5 \).

They are also not exhaustive because the email “hello from an old friend” is not covered by any rule.
Rule $r_1$: $(0.2 \leq x_1 \leq 2.2) \cap (1.3 \leq x_2 \leq 4.2) \rightarrow \text{red}$ has a coverage of $8/40 = 0.2$ and an accuracy of $7/8 = 0.875$.

The rules in rule set $R = \{r_1, r_2, r_3\}$ are mutually exclusive because the rectangles defining the rules are non-overlapping, but they are not exhaustive because there are areas of the space not covered by any of the rectangles.
Default rule

- A simple way to handle a non-exhaustive rule set is to add a *default* rule which has an empty antecedent (always true):

  \[ r_d: () \rightarrow y_d \]

- Note that since this rule is always triggered the resulting rule set is necessarily *not* mutually exclusive (as long as any other rule is sometimes triggered)

  \[ \Rightarrow \] Some mechanism is needed to handle non-mutually exclusive cases (see next slide)
Ordering or voting

- When the rule set is not mutually exclusive, conflicts need to be handled by
  - Rule *ordering*: Instead of an unordered collection of rules, the order is taken to be important. Hence, a new record is classified by the *first* rule that it triggers.
  - *Voting*: Each triggered rule votes for its consequence class. (The votes can also be weighed by the rule’s accuracy.)

- Note that an ordered list of rules (a *decision list*) behaves quite a bit like a decision tree. In fact, it is easy to show that a decision tree can be written in the form of an ordered list of rules, and vice versa.
Sequential covering algorithm

Learning a rule-based classifier can be performed by identifying good rules and then removing the covered records from the training set:

(i) Original Data

(ii) Step 1

(iii) Step 2

(iv) Step 3
Selecting the next rule

- We want to select a rule that has both high coverage and high accuracy. (Of course, typically this is a trade-off which has to be made.)

- Searching for such a rule:
  - Exhaustive search of all possible rules infeasible for all but the very smallest of problems
  - ‘General-to-specific’ greedy search: Start from an empty antecedent, add conditions one at a time to improve accuracy (but take care to still have decent coverage as well)
  - ‘Specific-to-general’ greedy search: Start from a random record, then generalize by removing conditions one-by-one to obtain better coverage while not sacrificing too much in accuracy
Characteristics of rule-based classifiers

- Expressiveness similar to that of decision trees
  - Rectilinear partitioning (though more complex decision boundaries with unordered rule sets and voting)
  - Ordered rule sets can be written as decision trees and vice versa
  - Similar also in terms of the underlying principles of learning the models
  - Non-parametric, need for regularization

- Can produce descriptive models
  - Easy to interpret (when the number of rules is not too large)
Decision trees and rule-based classifiers: Summary

- Can be easy to understand/interpret by experts/users (for small models)

- Classification generally very fast (worst case is linear in depth of decision tree or rule list length)

- Non-parametric method (can approximate optimal classifier for any underlying distribution) in the large sample limit (but may require a very large number of rules)

- Need to regularize / find a good stopping criterion when learning to avoid overfitting