Lecture 3: Decision Trees

Cognitive Systems II - Machine Learning
WS 2005/2006

Part I: Basic Approaches of Concept Learning

ID3, Information Gain, Overfitting, Pruning
Decision Tree Representation

- classification of instances by sorting them down the tree from the root to some leaf node
  - node ≈ test of some attribute
  - branch ≈ one of the possible values for the attribute

- decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances
  - i.e., \((... \land ... \land ...) \lor (... \land ... \land ...) \lor ...

- equivalent to a set of if-then-rules
  - each branch represents one if-then-rule
    - if-part: conjunctions of attribute tests on the nodes
    - then-part: classification of the branch
This decision tree is equivalent to:

if \((\text{Outlook} = \text{Sunny}) \land (\text{Humidity} = \text{Normal})\) then \(Yes\);
if \((\text{Outlook} = \text{Overcast})\) then \(Yes\);
if \((\text{Outlook} = \text{Rain}) \land (\text{Wind} = \text{Weak})\) then \(Yes\);
Appropriate Problems

- Instances are represented by attribute-value pairs, e.g. (Temperature, Hot)

- Target function has discrete output values, e.g. yes or no (concept/classification learning)

- Disjunctive descriptions may be required

- Training data may contain errors

- Training data may contain missing attribute values

⇒ last three points make Decision Tree Learning more attractive than CANDIDATE-ELIMINATION
ID3

- learns decision trees by constructing them top-down
- employs a greedy search algorithm without backtracking through the space of all possible decision trees
  ⇒ finds the shortest (wrt path length) but not necessarily the best decision tree

key idea:
- selection of the next attribute according to a statistical measure
- all examples are considered at the same time (simultaneous covering)
- recursive application with reduction of selectable attributes until each training example can be classified unambiguously
**ID3 Algorithm for Concept Learning**

\[ ID3(\text{Examples, Target\_attribute, Attributes}) \]

- Create a *Root* for the tree
- If all examples are **positive**, Return single-node tree *Root*, with *label* = +
- If all examples are **negative**, Return single-node tree *Root*, with *label* = −
- If *Attributes* is empty, Return single-node tree *Root*, with *label* = most common value of *Target\_attribute* in *Examples*
- **otherwise**, Begin
  - *A* ← attribute in *Attributes* that best classifies *Examples*
  - decision attribute for *Root* ← *A*
  - For each possible value \( v_i \) of *A*
    - Add new branch below *Root* with *A* = \( v_i \)
    - Let *Examples\_{v_i}* be the subset of *Examples* with \( v_i \) for *A*
    - If *Examples\_{v_i} is empty*
      - Then add a leaf node with label = most common value of *Target\_attribute* in *Examples*
    - Else add ID3(*Examples\_{v_i}, Target\_Attribute, Attributes − \{A\})*
  - Return *Root*
The best classifier

central choice: Which attribute classifies the examples best?

ID3 uses the information gain

statistical measure that indicates how well a given attribute separates the training examples according to their target classification

\[
Gain(S, A) = Entropy(S) - \sum_{v \in \text{values}(A)} \frac{|S_v|}{|S|} \cdot Entropy(S_v)
\]

original entropy of S

relative entropy of S

interpretation:

- denotes the reduction in entropy caused by partitioning \( S \) according to \( A \)
- alternative: number of saved yes/no questions (i.e., bits)

\[\Rightarrow \text{attribute with } \max_A Gain(S, A) \text{ is selected!}\]
Entropy

- statistical measure from information theory that characterizes (im-)purity of an arbitrary collection of examples $S$

- for binary classification: $H(S) \equiv -p_\oplus \log_2 p_\oplus - p_\ominus \log_2 p_\ominus$

- for n-ary classification: $H(S) \equiv \sum_{i=1}^{n} -p_i \log_2 p_i$

- interpretation:
  - specification of the minimum number of bits of information needed to encode the classification of an arbitrary member of $S$
  - alternative: number of yes/no questions
Entropy

- **Minimum** of $H(S)$
  - for minimal impurity $\rightarrow$ point distribution
  - $H(S) = 0$

- **Maximum** of $H(S)$
  - for maximal impurity $\rightarrow$ uniform distribution
  - for binary classification: $H(S) = 1$
  - for n-ary classification: $H(S) = \log_2 n$
# Illustrative Example

**example days**

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temp.</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>D7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>D9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>D13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>D14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>
Illustrative Example

entropy of $S$

\[ S = \{D1, ..., D14\} = [9+, 5-] \]
\[ H(S) = -\frac{9}{14} \cdot \log_2 \frac{9}{14} - \frac{5}{14} \cdot \log_2 \frac{5}{14} = 0.940 \]

information gain (e.g. $Wind$)

\[ S_{Weak} = \{D1, D3, D4, D5, D8, D9, D10, D13\} = [6+, 2-] \]
\[ S_{Strong} = \{D2, D6, D7, D11, D12, D4\} = [3+, 3-] \]

\[
\text{Gain}(S, Wind) = H(S) - \sum_{v \in Wind} \frac{|S_v|}{|S|} \cdot H(S_v)
\]
\[
= H(S) - \frac{8}{14} \cdot H(S_{Weak}) - \frac{6}{14} \cdot H(S_{Strong})
\]
\[
= 0.940 - \frac{8}{14} \cdot 0.811 - \frac{6}{14} \cdot 1.000
\]
\[
= 0.048
\]
Illustrative Example

Which attribute is the best classifier?

\[
\text{Gain (S, Humidity)} = 0.940 - \frac{7}{14} \cdot 0.985 - \frac{7}{14} \cdot 0.592 = 0.151
\]

\[
\text{Gain (S, Wind)} = 0.940 - \frac{8}{14} \cdot 0.811 - \frac{6}{14} \cdot 1.00 = 0.048
\]
Illustrative Example

- informations gains for the four attributes:

  \[
  \begin{align*}
  Gain(S, \text{Outlook}) &= 0.246 \\
  Gain(S, \text{Humidity}) &= 0.151 \\
  Gain(S, \text{Wind}) &= 0.048 \\
  Gain(S, \text{Temperature}) &= 0.029 \\
  \end{align*}
  \]

  ⇒ \textit{Outlook} is selected as best classifier and is therefore \textit{Root} of the tree

  ⇒ now branches are created below the root for each possible value

  - because every example for which \textit{Outlook} = \textit{Overcast} is positive, this node becomes a leaf node with the classification \textit{Yes}

  - the other descendants are still ambiguous (\( H(S) \neq 0 \))

  - hence, the decision tree has to be further elaborated below these nodes
The diagram illustrates a decision tree with the following attributes:

- **Outlook**
  - Sunny
  - Overcast
  - Rain

The decision tree is structured as follows:

- **Outlook**
  - **Sunny**
    - {D1, D2, D8, D9, D11}
    - [2+, 3-]
    - Next attribute is **Humidity**
        - $G_{\text{Gain}}(S_{\text{Sunny}}, \text{Humidity}) = 0.970 - (3/5) 0.0 - (2/5) 0.0 = 0.970$
  - **Overcast**
    - {D3, D7, D12, D13}
    - [4+, 0-]
    - Next attribute is **Temperature**
        - $G_{\text{Gain}}(S_{\text{Sunny}}, \text{Temperature}) = 0.970 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = 0.570$
  - **Rain**
    - {D4, D5, D6, D10, D14}
    - [3+, 2-]
    - Next attribute is **Wind**
        - $G_{\text{Gain}}(S_{\text{Sunny}}, \text{Wind}) = 0.970 - (2/5) 1.0 - (3/5) 0.918 = 0.019$

**Which attribute should be tested here?**

$S_{\text{Sunny}} = \{D1, D2, D8, D9, D11\}$
Illustrative Example

Resulting decision tree

```
   Outlook
     /    \
  Sunny  Overcast
       /   /    \
  No  Yes  Rain
       |     |
       |     |
  Humidity
     /   \
  High  Normal
       /   \
  No   Yes
  Wind
     /   \
  Strong  Weak
       /   /    \
  No  Yes   Yes
```
Hypothesis Space Search

$H \approx$ complete space of finite discrete functions, relative to the available attributes (i.e. all possible decision trees)

- capabilities and limitations:
  - returns just one single consistent hypothesis
  - performs greedy search (i.e., $\max_A Gain(S, A)$)
  - susceptible to the usual risks of hill-climbing without backtracking
  - uses all training examples at each step $\Rightarrow$ simultaneous covering
Inductive Bias

- as mentioned above, ID3 searches
  - complete space of possible hypotheses, but not completely
  ⇒ Preference Bias

- Inductive bias: Shorter trees are preferred to longer trees. Trees that place high information gain attributes close to the root are also preferred.

- Why prefer shorter hypotheses?
  - Occam’s Razor: Prefer the simplest hypothesis that fits the data! (aka W. Ockham)
  - see Minimum Description Length Principle (Bayesian Learning)
  - e.g., if there are two decision trees, one with 500 nodes and another with 5 nodes, the second one should be preferred
  ⇒ better chance to avoid overfitting
Given a hypothesis space $H$, a hypothesis $h \in H$ is said to **overfit** the training data if there exists some alternative hypothesis $h' \in H$, such that $h$ has smaller error than $h'$ over the training data, but $h'$ has smaller error than $h$ over the entire distribution of instances.
Overfitting

- reasons for overfitting:
  - noise in the data
  - number of training examples is too small to produce a representative sample of the target function

- how to avoid overfitting:
  - stop the tree growing earlier, before it reaches the point where it perfectly classifies the training data
  - allow overfitting and then post-prune the tree (more successful in practice!)

- how to determine the perfect tree size:
  - separate validation set to evaluate utility of post-pruning
  - apply statistical test to estimate whether expanding (or pruning) produces an improvement
Reduced Error Pruning

- each of the decision nodes is considered to be a candidate for pruning

- **Pruning** a decision node consists of removing the subtree rooted at the node, making it a leaf node and assigning the most common classification of the training examples affiliated with that node

- nodes are removed only if the resulting tree performs **not worse** than the original tree over the validation set

- pruning starts with the node whose removal most increases accuracy and continues until further pruning is harmful
Reduced Error Pruning

- effect of reduced error pruning:

- any node added to coincidental regularities in the training set is likely to be pruned
Rule Post-Pruning

- rule post-pruning involves the following steps:
  1. Infer the decision tree from the training set (Overfitting allowed!)
  2. Convert the tree into a set of rules
  3. Prune each rule by removing any preconditions that result in improving its estimated accuracy
  4. Sort the pruned rules by their estimated accuracy

- one method to estimate rule accuracy is to use a separate validation set

- pruning rules is more precise than pruning the tree itself
Alternative measures

- natural bias in information gain favors attributes with many values over those with few values

- e.g. attribute *Date*
  - very large number of values (e.g. March 21, 2005)
  - inserted in the above example, it would have the highest information gain, because it perfectly separates the training data
  - but the classification of unseen examples would be impossible

- alternative measure: *GainRatio*

  \[
  \text{GainRatio}(S, A) = \frac{\text{Gain}(S, A)}{\text{SplitInformation}(S, A)}
  \]

  \[
  \text{SplitInformation}(S, A) \equiv - \sum_{i=1}^{n} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}
  \]

  \(\text{SplitInformation}(S, A)\) is sensitive to how broadly and uniformly *A* splits *S* (entropy of *S* with respect to the values of *A*)

  \(\Rightarrow\) *GainRatio* penalizes attributes such as *Date*
Summary

- practical and intuitively understandable method for concept learning
- able to learn disjunctive, discrete-valued concepts
- noise in the data is allowed

**ID3** is a simultaneous covering algorithm based on information gain that performs a greedy top-down search through the space of possible decision trees

**Inductive Bias**: Short trees are preferred (Ockham’s Razor)

overfitting is an important issue and can be reduced by pruning