Decision Trees
When to Consider Using 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
- Risk analysis
  - Credit, loans
  - Insurance
  - Consumer fraud
  - Employee fraud
- Modeling calendar scheduling preferences (predicting quality of candidate time)

Instances Usually Represented Using Discrete Valued Attributes

- Typical types
  - Nominal (\{red, yellow, green\})
  - Quantized (\{low, medium, high\})
- Handling numerical values
  - Discretization, a form of vector quantization (e.g., histogramming)
  - Using thresholds for splitting nodes
Decision Trees and Decision Boundaries

- Decision tree divides the Instance Space into Axis-Parallel Rectangles

![Decision Tree Diagram]

Machine Learning
Algorithm *Build-DT*(Examples, Attributes)

IF all examples have the same label THEN
   RETURN (leaf node with label)
ELSE IF set of attributes is empty THEN
   RETURN (leaf with majority label)
ELSE
   Choose best attribute $A$ as root
   FOR each value $\nu$ of $A$
      Create a branch out of the root for the condition $A = \nu$
      IF $\{x \in \text{Examples}: x.A = \nu\} = \emptyset$ THEN
         RETURN (leaf with majority label)
      ELSE
         *Build-DT*(\{$x \in \text{Examples}: x.A = \nu\}, Attributes \sim \{A\})

But Which Attribute Is Best?

Machine Learning
Applicability of Decision Trees

Assumptions in Previous Algorithm

- **Discrete output**
  - Real-valued outputs are possible
  - Regression trees [Breiman et al., 1984]

- **Discrete input**
  - Quantization methods
  - *Inequalities* at nodes instead of equality tests (see rectangle example)

Scaling Up

- Critical in knowledge discovery and database mining (KDD) from very large databases (VLDB)
- Good news: efficient algorithms exist for processing many *examples*
- Bad news: much harder when there are too many *attributes*

Other Desired Tolerances

- Noisy data (classification noise ≡ incorrect labels; attribute noise ≡ inaccurate or imprecise data)
- Missing attribute values
Choosing the “Best” Root Attribute

- **Objective**
  - Construct a decision tree that is as small as possible (Occam’s Razor)
  - Subject to: consistency with labels on training data

- **Obstacles**
  - Finding the minimal consistent hypothesis (i.e., decision tree) is NP-hard
  - Recursive algorithm (*Build-DT*)
    - A greedy heuristic search for a simple tree
    - Cannot guarantee optimality

- **Main Decision: Next Attribute to Condition On**
  - Want: attributes that split examples into sets that are relatively pure in one label
  - Result: closer to a leaf node
  - Most popular heuristic
    - Developed by J. R. Quinlan
    - Based on information gain
    - Used in *ID3* algorithm
A Measure of Uncertainty

- The Quantity
  - Purity: how close a set of instances is to having just one label
  - Impurity (disorder): how close it is to total uncertainty over labels

- The Measure: Entropy
  - Directly proportional to impurity, uncertainty, irregularity, surprise
  - Inversely proportional to purity, certainty, regularity, redundancy

Example

- For simplicity, assume $H = \{0, 1\}$, distributed according to $P(y)$
  - Can have (more than 2) discrete class labels
  - Continuous random variables: differential entropy

- Optimal purity for $y$ either
  - $P(y = 0) = 1, P(y = 1) = 0$
  - $P(y = 1) = 1, P(y = 0) = 0$

- What is the least pure probability distribution?
  - $P(y = 0) = 0.5, P(y = 1) = 0.5$
  - Correlates with maximum impurity/uncertainty/irregularity/surprise
  - Property of entropy: concave function (“concave downward”)
Entropy: Information Theoretic Definition

☐ Components

- $D$: a set of examples $\{<x_1, c(x_1)>, <x_2, c(x_2)>, ..., <x_m, c(x_m)>)\}$
- $p_+ = P(c(x) = +)$, $p_- = P(c(x) = -)$

☐ Definition

- $H$ is defined over a probability density function $\rho$
- $D$ contains examples whose frequency of + and - labels indicates $\rho_+$ and $\rho_-$ for the observed data
- The entropy of $D$ relative to $c$ is:
  $$H(D) \equiv -\rho_+ \log_b(\rho_+) - \rho_- \log_b(\rho_-)$$

☐ What Units is $H$ Measured In?

- Depends on the base $b$ of the log (bits for $b = 2$, nats for $b = e$, etc.)
- A single bit is required to encode each example in the worst case ($\rho_+ = 0.5$)
- If there is less uncertainty (e.g., $\rho_+ = 0.8$), we can use less than 1 bit each
Information Gain: Information Theoretic Definition

- **Partitioning on Attribute Values**
  - Recall: a partition of \( D \) is a collection of disjoint subsets whose union is \( D \)
  - Goal: measure the uncertainty removed by splitting on the value of attribute \( A \)

- **Definition**
  - The information gain of \( D \) relative to attribute \( A \) is the expected reduction in entropy due to splitting ("sorting") on \( A \):

\[
\text{Gain}(D, A) \equiv H(D) - \sum_{v \in \text{values}(A)} \frac{|D_v|}{|D|} \cdot H(D_v)
\]

where \( D_v = \{ x \in D. x.A = v \} \), the set of examples in \( D \) where attribute \( A \) has value \( v \)

- Idea: partition on \( A \); scale entropy to the size of each subset \( D_v \)

- **Which Attribute Is Best?**

```
[29+, 35-] [29+, 35-]
A1
True False
[21+, 5-] [8+, 30-]

[21+, 5-] [18+, 33-]
A2
True False
[11+, 2-]
```
An Illustrative Example

Training Examples for Concept *PlayTennis*

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th><em>PlayTennis</em>**?*</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Light</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Light</td>
<td>No</td>
</tr>
<tr>
<td>9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>

*ID3* Build-DT using Gain(*)

How Will *ID3* Construct A Decision Tree?
Constructing A Decision Tree for *PlayTennis* using *ID3*\cite{1}

### Selecting The Root Attribute

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Light</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Light</td>
<td>No</td>
</tr>
<tr>
<td>9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>

#### Prior (unconditioned) distribution: 9+, 5-

- $H(D) = -\frac{9}{14} \log_2 \left(\frac{9}{14}\right) - \frac{5}{14} \log_2 \left(\frac{5}{14}\right)$ \text{bits} = 0.94 bits
- $H(D, \text{Humidity}=\text{High}) = -\frac{3}{7} \log_2 \left(\frac{3}{7}\right) - \frac{4}{7} \log_2 \left(\frac{4}{7}\right)$ \text{bits} = 0.985 bits
- $H(D, \text{Humidity}=\text{Normal}) = -\frac{6}{7} \log_2 \left(\frac{6}{7}\right) - \frac{1}{7} \log_2 \left(\frac{1}{7}\right)$ \text{bits} = 0.592 bits
- $\text{Gain}(D, \text{Humidity}) = 0.94 - \left(\frac{7}{14}\right) \times 0.985 - \left(\frac{7}{14}\right) \times 0.592 = 0.151$ bits
- Similarly, $\text{Gain}(D, \text{Wind}) = 0.94 - \left(\frac{8}{14}\right) \times 0.811 + \left(\frac{6}{14}\right) \times 1.0 = 0.048$ bits

\[
\text{Gain}(D, A) \equiv H(D) - \sum_{v \in \text{values}(A)} \left(\frac{D_v}{|D|} \times H(D_v)\right)
\]
Constructing A Decision Tree for *PlayTennis* using *ID3* [2]

### Selecting The Root Attribute

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Light</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Light</td>
<td>No</td>
</tr>
<tr>
<td>9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>

- \( Gain(D, \text{Humidity}) = 0.151 \text{ bits} \)
- \( Gain(D, \text{Wind}) = 0.048 \text{ bits} \)
- \( Gain(D, \text{Temperature}) = 0.029 \text{ bits} \)
- \( Gain(D, \text{Outlook}) = 0.246 \text{ bits} \)

### Selecting The Next Attribute (Root of Subtree)

- Continue until every example is included in path or purity = 100%
- What does purity = 100% mean?
- Can \( Gain(D, A) < 0 \)?
Selecting The Next Attribute (Root of Subtree)

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Light</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Light</td>
<td>No</td>
</tr>
<tr>
<td>9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>

- Convention: $\log (0/a) = 0$
- $Gain(D_{Sunny, Humidity}) = 0.97 - (3/5) \times 0 - (2/5) \times 0 = 0.97$ bits
- $Gain(D_{Sunny, Wind}) = 0.97 - (2/5) \times 1 - (3/5) \times 0.92 = 0.02$ bits
- $Gain(D_{Sunny, Temperature}) = 0.57$ bits

Top-Down Induction

- For discrete-valued attributes, terminates in $O(r)$ splits
- Makes at most one pass through data set at each level (why?)
Constructing A Decision Tree for \textit{PlayTennis} using \textit{ID3}[4]

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>\textit{PlayTennis}?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Light</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Light</td>
<td>No</td>
</tr>
<tr>
<td>9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>

\begin{itemize}
  \item \textit{Outlook}: \{Sunny, Overcast, Rain\}
  \item \textit{Temperature}: \{Hot, Mild\}
  \item \textit{Humidity}: \{High, Normal\}
  \item \textit{Wind}: \{Light, Strong\}
\end{itemize}
Hypothesis Space Search by \textit{ID3}

- **Search Problem**
  - Conduct a search of the \textit{space of decision trees}, which can represent all possible discrete functions.
  - Objective: to find the best decision tree (minimal consistent tree).
  - Obstacle: finding this tree is \textit{NP}-hard.
  - Tradeoff
    - Use \textit{heuristic} (figure of merit that guides search).
    - Use greedy algorithm (gradient “descent”) without backtracking.

- **Search Space and Search Strategy of ID3**
  - Hypothesis space of ID3 contains a \textit{complete space} of finite discrete valued functions.
  - ID3 maintains only a single current hypothesis as it searches through the space of decision trees.
    - Cannot represent all consistent hypothesis $\rightarrow$ no incremental algorithm.
  - ID3 uses greedy algorithm.
    - Locally Optimal.
  - ID3 uses all training data at each step in the search to make statistically based decisions.
    - Robust to noisy data.
Types of Biases

- **Preference (Search) Bias**
  - Put priority on choosing hypothesis
  - Encoded in *learning algorithm* \(\rightarrow\) Compare: *search heuristic*
  - *For ex.* ID3 has this type of bias.

- **Language Bias**
  - Put restriction on the set of hypotheses considered
  - Encoded in *knowledge (hypothesis) representation*
  - Compare: *restriction of search space*
  - *For ex.* Candidate-Elimination has this type of bias.

Which Bias is better?

- Preference bias is more desirable.
  - Because, the learner works within a complete space that is assured to contain the unknown concept.

Inductive Bias of ID3

- Shorter trees are preferred over longer trees.
  - Occam’s razor: *Prefer the simplest hypothesis that fits the data.*

- Trees that place high information gain attributes close to the root are preferred over those that do not.
Over-fitting in Decision Trees

Recall: Induced Tree

Boolean Decision Tree for Concept PlayTennis

Noisy Training Example

- How shall the DT be revised (incremental learning)?
- Example 15: <Sunny, Hot, Normal, Strong, ->
  - Example is noisy because the correct label is +
  - Previously constructed tree misclassifies it
- New hypothesis $h' = \mathcal{T}'$ is expected to perform worse than $h = \mathcal{T}$

Machine Learning
Overfitting in Inductive Learning

Definition: Hypothesis $h$ overfits training data set $D$ if there exists an alternative hypothesis $h'$ such that $\text{error}_D(h) < \text{error}_D(h')$ but $\text{error}_{\text{test}}(h) > \text{error}_{\text{test}}(h')$

- Causes
  - decisions based on too little data or noisy data

How can we avoid over-fitting?

- Prevention
  - Stop training (growing) before it reaches the point that overfits.
  - Select attributes that are relevant (i.e., will be useful in the decision tree)
  - requires some predictive measure of relevance

- Avoidance
  - Allow to over-fit, then improve the generalization capability of the tree.
  - Holding out a validation set (test set)

- Detection and Recovery
  - Letting the problem happen, detecting when it does, recovering afterward
  - Build model, remove (prune) elements that contribute to overfitting

How to Select “Best” Tree?

- Training and Validation Set
  - Use a separate set of examples (distinct from the training set) for test.

- Statistical Test
  - Use all data for training, but apply the statistical test to estimate the over-fitting.

- Define the measure of complexity
  - Halting the grow when this measure is minimized.
Two Basic Approaches

- **Pre-pruning (avoidance):** stop growing tree at some point during construction when it is determined that there is not enough data to make reliable choices
- **Post-pruning (recovery):** grow the full tree and then remove nodes that seem not to have sufficient evidence
Reduced-Error Pruning

- Post-Pruning, Cross-Validation Approach
- Split Data into Training and Validation Sets
- Function $Prune(T, node)$
  - Remove the subtree rooted at $node$
  - Make $node$ a leaf (with majority label of associated examples)
- Algorithm $Reduced-Error-Pruning(D)$
  - Partition $D$ into $D_{train}$ (training / “growing”), $D_{validation}$ (validation / “pruning”)
  - Build complete tree $T$ using ID3 on $D_{train}$
  - UNTIL accuracy on $D_{validation}$ decreases DO
    - FOR each non-leaf node $candidate$ in $T$
      - $Temp[candidate] ← Prune(T, candidate)$
      - $Accuracy[candidate] ← Test(Temp[candidate], D_{validation})$
    - $T ← T' ∈ Temp$ with best value of $Accuracy$ (best increase; greedy)
  - RETURN (pruned) $T$

- Drawbacks
  - Data is limited

---

**Machine Learning**
Rule Post-Pruning

- **Frequently Used Method**
  - Popular anti-overfitting method; perhaps most popular pruning method
  - Variant used in C4.5, an outgrowth of ID3

- **Algorithm Rule-Post-Pruning(D)**
  - Infer from D (using ID3 - grow until D is fit as well as possible (allow overfitting)
  - Convert into equivalent set of rules (one for each root-to-leaf path)
  - Prune (generalize) each rule *independently* by deleting any preconditions whose deletion improves its estimated accuracy
  - Sort the pruned rules
    - Sort by their estimated accuracy
    - Apply them in sequence on D

- **Rule Syntax**
  - LHS: precondition (conjunctive formula over attribute equality tests)
  - RHS: class label

- **Example**
  - IF (Outlook = Sunny) ∧ (Humidity = High) THEN PlayTennis = No
  - IF (Outlook = Sunny) ∧ (Humidity = Normal) THEN PlayTennis = Yes

Machine Learning
Pruning Algorithms

- Reduced Error Pruning
- Pessimistic Error Pruning
- Minimum Error Pruning
- Critical Value Pruning
- Cost-Complexity Pruning

Reading


Two Methods for Handling Continuous Attributes

- Discretization (e.g., histogramming)
  - Break real-valued attributes into ranges *in advance*
  - e.g., \{ *high* ≡ Temp > 35° C, *med* ≡ 10° C < Temp ≤ 35° C, *low* ≡ Temp ≤ 10° C \}

- Using thresholds for splitting nodes
  - e.g., \( A \leq a \) produces subsets \( A \leq a \) and \( A > a \)
  - *Information gain is calculated the same way* as for discrete splits

How to Find the Split with Highest Gain?

- FOR each continuous attribute \( A \)
  - Divide examples \( \{ x \in D \} \) according to \( x.A \)
  - FOR each ordered pair of values (\( l, u \)) of \( A \) with different labels
    - Evaluate gain of *mid-point* as a possible threshold, i.e., \( D_{A \leq (l+u)/2} \) \( D_{A > (l+u)/2} \)

- Example
  - \( A \equiv \text{Length} \): 10 15 21 28 32 40 50
  - Class: - + + - + + -
  - Check thresholds: \( \text{Length} \leq 12.5? \) \( \leq 24.5? \) \( \leq 30? \) \( \leq 45? \)
Missing Data: Unknown Attribute Values

Problem: What If Some Examples Missing Values of $A$?

- Often, values not available for all attributes during training or testing
- Example: medical diagnosis
  - $<\text{Fever}= \text{true}, \text{Blood-Pressure}= \text{normal}, \ldots, \text{Blood-Test}= ?, \ldots>$
  - Sometimes values truly unknown, sometimes low priority (or cost too high)
- Missing values in learning versus classification
  - **Training**: evaluate $Gain(D, A)$ where for some $x \in D$, a value for $A$ is not given
  - **Testing**: classify a new example $x$ without knowing the value of $A$

Solutions: Incorporating a *Guess* into Calculation of $Gain(D, A)$

<table>
<thead>
<tr>
<th>Day</th>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>PlayTennis?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Light</td>
<td>No</td>
</tr>
<tr>
<td>2</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>3</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>7</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>Sunny</td>
<td>Mild</td>
<td>???</td>
<td>Light</td>
<td>No</td>
</tr>
<tr>
<td>9</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>10</td>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>11</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>12</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>13</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Light</td>
<td>Yes</td>
</tr>
<tr>
<td>14</td>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>
A regression tree is constructed in almost the same manner as classification tree, except that appropriate *impurity* measure for regression is used.

Let $X_m$ be a subset of $X$ reaching node $m$.

In regression tree, the goodness of a split is measured by the mean square error from the estimated value.

**Reading**

Decision Trees

- **Univariate Trees**
  - In *univariate trees*, the test at each internal node just uses only one of input attributes.

- **Multivariate Trees**
  - In *multivariate trees*, the test at each internal node can use all input attributes.
  - For example: Consider a data set with numerical attributes.
    - The test can be made using the weighted linear combination of some input attributes.
  - For example, if \( X = (x_1, x_2) \) be the input attributes. Let \( f(X) = w_0 + w_1 x_1 + w_2 x_2 \) can be used for test at an internal node. Such as \( f(x) > 0 \).

- **Reading**
Incremental Learning of Decision Trees

- ID3 cannot be trained incrementally.
- ID4, ID5, ID5R are samples of incremental induction of decision trees.

Reading
Attributes with Many Values

- **Problem**
  - If attribute has many values, \( Gain(\cdot) \) will select it (why?)
  - Imagine using \( Date = 06/03/1996 \) as an attribute!

- **One Approach: Use** \( GainRatio \) **instead of** \( Gain \)

\[
Gain(D, A) = H(D) - \sum_{v \in values(A)} \left( \frac{|D_v|}{|D|} \cdot H(D_v) \right)
\]

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

\[
SplitInformation(D, A) = -\sum_{v \in values(A)} \left( \frac{|D_v|}{|D|} \cdot \log \frac{|D_v|}{|D|} \right)
\]

- \( SplitInformation \) directly proportional to \( c = |values(A)| \)
- i.e., penalizes attributes with more values
  - e.g., suppose \( c_1 = c_{Date} = n \) and \( c_2 = 2 \)
  - \( SplitInformation(A_1) = \log(n) \), \( SplitInformation(A_2) = 1 \)
  - If \( Gain(D, A_1) = Gain(D, A_2) \), \( GainRatio(D, A_1) \ll GainRatio(D, A_2) \)
- Thus, \textit{preference bias} (for lower branch factor) expressed via \( GainRatio(\cdot) \)
Alternative Attribute Selection : Gini Index [1]

- If a data set $D$ contains examples from $n$ classes, gini index, $\text{gini}(D)$ is defined as
  \[
  \text{gini} \ (D) = 1 - \sum_{j=1}^{n} p_j^2
  \]
  where $p_j$ is the relative frequency of class $j$ in $D$.

- If a data set $D$ is split on $A$ into two subsets $D_1$ and $D_2$, the gini index $\text{gini}(D)$ is defined as
  \[
  \text{gini}_A(D) = \frac{|D_1|}{|D|} \text{gini} \ (D_1) + \frac{|D_2|}{|D|} \text{gini} \ (D_2)
  \]

- Reduction in Impurity:
  \[
  \Delta \text{gini}(A) = \text{gini}(D) - \text{gini}_A(D)
  \]

- The attribute provides the smallest $\text{gini}_{\text{split}}(D)$ (or the largest reduction in impurity) is chosen to split the node (need to enumerate all the possible splitting points for each attribute).
Handling Attributes With Different Costs

In some learning tasks the instance attributes may have associated costs.

- We prefer decision trees that use low cost attributes where possible, relying on high cost attributes only when needed to produce reliable classifications.

**Solutions**

- Extended ID3

\[
\frac{Gain(S, A)}{Cost(A)}
\]

- Tan and Schlimmer

\[
\frac{Gain^2(S, A)}{Cost(A)}
\]

- Nunez

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

- where \(w \in [0, 1]\) is a constant
References


