Decision Tree Learning

- Widely used, practical method for inductive inference
- Approximates discrete-valued target functions as trees
- Robust to noisy data and capable of learning disjunctive expressions
- A family of decision tree learning algorithms includes ID3, ASSISTANT and C4.5
- Use a completely expressive hypothesis space
- Inductive bias is a preference for small trees over large trees
Introduction

- Learned function is represented as a decision tree
- Learned trees can also be re-represented as sets of *if-then* rules to improve human readability

![Decision Tree Diagram]

Decision Tree Representation

- Decision trees classify instances
  - by sorting them down from the root to the leaf node,
  - which provides the classification of the instance.
- Each node in the tree specifies a test of some *attribute* of the instance.
- Each branch descending from that node corresponds to one of the possible values of this attribute.
Examples for Decision Tree

<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>

Learned Decision Tree for PlayTennis

![Decision Tree Diagram](image-url)
Decision Trees represent disjunction of conjunctions

- Decision tree represents
  
  \[(\text{outlook} = \text{“sunny”} \land \text{humidity} = \text{“normal”}) \lor (\text{outlook} = \text{“overcast”}) \lor (\text{outlook} = \text{“rain”} \land \text{wind} = \text{“weak”})\]

Appropriate Problems for Decision Tree Learning (1)

- Instances are represented by attribute-value pairs
  
  - each attribute takes on a small no of disjoint possible values, eg, hot, mild, cold
  - extensions allow real-valued variables as well, eg temperature

- The target function has discrete output values
  
  - eg, Boolean classification (yes or no)
  - easily extended to multiple-valued functions
  - can be extended to real-valued outputs as well
Appropriate Problems for Decision Tree Learning (2)

- Disjunctive descriptions may be required
  - naturally represent disjunctive expressions

- The training data may contain errors
  - robust to errors in classifications and in attribute values

- The training data may contain missing attribute values
  - eg, humidity value is known only for some training examples

Appropriate Problems for Decision Tree Learning (3)

- Practical problems that fit these characteristics are:
  - learning to classify
    - medical patients by their disease
    - equipment malfunctions by their cause
    - loan applications by by likelihood of defaults on payments
The Basic Decision Tree Learning Algorithm (ID3)

- Top-down, greedy search (no backtracking) through space of possible decision trees
- Begins with the question
  - “which attribute should be tested at the root of the tree?”
- Answer
  - evaluate each attribute to see how it alone classifies training examples
- Best attribute is used as root node
  - descendant of root node is created for each possible value of this attribute

Which Attribute Is Best for the Classifier?

- Select attribute that is most useful for classification
- ID3 uses Information gain as a quantitative measure of an attribute
- Information Gain: A statistical property that measures how well a given attribute separates the training examples according to their target classification.
ID3 Notation

Attribute A that best classifies the examples (Target Attribute for ID3)

ID3 Algorithm to learn boolean-valued functions

ID3 (Examples, Target_attribute, Attributes)

Examples are the training examples. Target_attribute is the attribute (or feature) whose value is to be predicted by the tree. Attributes is a list of other attributes that may be tested by the learned decision tree. Returns a decision tree (actually the root node of the tree) that correctly classifies the given Examples.

- Create a Root node for the tree
- If all Examples are positive, Return the single-node tree Root, with label +=
- If all Examples are negative, Return the single-node tree Root, with label =-
- If Attributes is empty, Return the single-node tree Root, with label = the most common value of Target_attribute in Examples
- ©Note that we will return the name of a feature at this point
Summary of the ID3 Algorithm, continued

*Otherwise Begin
  • A ← the attribute from Attributes that best* classifies Examples
  • The decision attribute (feature) for Root ← A
  • For each possible value \( v_i \) of A,
    • Add a new tree branch below Root, corresponding to test \( A = v_i \)
    • Let \( \text{Examples}_{v_i} \) the subset of Examples that have value \( v_i \) for A
    • If \( \text{Examples}_{v_i} \) is empty′
      • Then below this new branch, add a leaf node with label = most common value of Target_attribute in Examples
    • Else, below this new branch add the subtree
      \[ \text{ID3(Examples}_{v_i}, \text{Target_attribute, Attributes - \{A\})} \]
  • End
  • Return Root

* The best attribute is the one with the highest information gain, as defined in

\[
G_{\text{Ent}}(S, A) \triangleq \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_{v}|}{|S|} \text{Entropy}(S_{v})
\]

Entropy as a Measure of Homogeneity of Examples

• Information Gain is defined in terms of Entropy
  • expected reduction in entropy caused by partitioning the examples according to this attribute

• Entropy: Characterizes the (im)purity of an arbitrary collection of examples

• Given a collection S of positive and negative examples, entropy of S relative to boolean classification is.

\[
\text{Entropy}(S) \triangleq -p_+ \log_2 p_+ - p_- \log_2 p_-
\]

Where \( p_+ \) is proportion of positive examples and \( p_- \) is proportion of negative examples
Entropy

- Illustration:
  - S is a collection of 14 examples with 9 positive and 5 negative examples

- Entropy of S relative to the Boolean classification:
  - Entropy \((9+, 5-) = -(\frac{9}{14}\log_2\frac{9}{14}) - (\frac{5}{14}\log_2\frac{5}{14})\)  
    \[\approx 0.940\]
  - Entropy is zero if all members of S belong to the same class
Entropy for multi-valued target function

If the target attribute can take on $c$ different values, the entropy of $S$ relative to this $c$-wise classification is

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

Information Gain Measures the Expected Reduction in Entropy

- Entropy measures the impurity of a collection
- Information gain $Gain(S, A)$ of attribute $A$ is the reduction in entropy caused by partitioning the examples according to this attribute

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

- where $Values(A)$ is the set of all possible values for attribute $A$ and $S_v$ is the subset of $S$ for which attribute $A$ has value $v$
Training Examples for Target Concept

*PlayTennis*

<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>

Stepping through ID3 for the example

- Top Level (with S={D1,...,D14})
  - Gain(S, Outlook) = 0.246
  - Gain(S, Humidity) = 0.151
  - Gain(S, Wind) = 0.048
  - Gain(S, Temperature) = 0.029

- Example computation of Gain for Wind
  - Values(Wind) = Weak, Strong
  - S = [9+, 5-]
  - $S_{weak} = [6+, 2-]$, $S_{strong} = [3+, 3-]$
  - Gain(S, Wind) = 0.940 - (8/14)0.811 - (6/14)1.00
    = 0.048
Sample calculations for Humidity and Wind

\[
\begin{align*}
S: & [9+5] \\
E & = 0.940
\end{align*}
\]

Humidity

\[
\begin{align*}
[3,+1] & \\
E & = 0.985
\end{align*}
\]

\[
\begin{align*}
[6,+1] & \\
E & = 0.592
\end{align*}
\]

Wind

\[
\begin{align*}
[6,+2] & \\
E & = 0.811
\end{align*}
\]

\[
\begin{align*}
[3,+3] & \\
E & = 1.00
\end{align*}
\]

Gain (S, Humidity)

\[
= 9.40 - (7/14) \times 0.985 - (7/14) \times 0.592 = 0.151
\]

Gain (S, Wind)

\[
= 9.40 - (8/14) \times 0.811 - (6/14) \times 1.0 = 0.48
\]

The Partially Learned Decision Tree

\[
\begin{align*}
(D1, D2, \ldots, D14) & \\
[9+3] & \\
\text{Sunny} & \\
\text{Overcast} & \\
\text{Rain} &
\end{align*}
\]

\[
\begin{align*}
(D1,D2,D3,D9,D11) & \\
[2,+3] & \\
\text{Yes} & \\
\text{No} &
\end{align*}
\]

\[
\begin{align*}
(D6,D8,D10,D14) & \\
[3,+2] & \\
\text{Yes} & \\
\text{No} &
\end{align*}
\]

Which attribute should be tested here?

\[
\begin{align*}
S_{\text{Sunny}} & = (D1,D2,D3,D9,D11) \\
\text{Gain} (S_{\text{Sunny}}, \text{Humidity}) & = 9.70 - (3/5) \times 0.0 - (2/5) \times 0.0 = 9.70 \\
\text{Gain} (S_{\text{Sunny}}, \text{Temperature}) & = 9.70 - (2/5) \times 0.0 - (2/5) \times 1.0 - (1/5) \times 0.0 = 0.70 \\
\text{Gain} (S_{\text{Sunny}}, \text{Wind}) & = 9.70 - (2/5) \times 1.0 - (3/5) \times 0.18 = 0.19
\end{align*}
\]
Hypothesis Space Search in Decision Tree Learning

- ID3 searches a hypothesis space for one that fits training examples
- ID3 performs hill-climbing considering progressively more elaborate hypotheses (to find a decision tree that correctly classifies the training data)
- Hill climbing is guided by evaluation function which is the gain measure

Hypothesis Space Search by ID3

Information gain heuristic guides search of hypothesis space by ID3
Capabilities and Limitations of ID3

- Hypothesis space is a complete space of all discrete valued functions
- Cannot determine how many alternative trees are consistent with training data (follows from maintaining a single current hypothesis)
- ID3 in its pure form performs no backtracking (usual risks of hill-climbing - converges to local optimum)
- ID3 uses all training examples at each step in to make statistically based decisions regarding how to refine its current hypothesis
  - more robust than Find-S and Candidate Elimination which are incrementally-based

Inductive Bias in ID3

- What is the policy by which ID3 generalizes from observed training examples to classify unseen instances?
- Basis for choosing one consistent hypothesis over others
Inductive Bias in Decision Tree Learning

- Approximate inductive bias of ID3: Shorter trees are preferred over larger trees.
  - Breadth First Search ID3 (BFS-ID3) searches all trees of depth 1, all trees of depth 2, etc and produces same tree as ID3

- A closer approximation to the inductive bias of ID3: Shorter trees are preferred over longer trees. Trees that place high information gain attributes close to the root are preferred over those that do not.

Restriction Biases and Preference Biases

- ID3 searches a complete hypothesis space (i.e., one capable of expressing any finite discrete-valued function).
  - Its inductive bias is a preference for certain hypotheses
    - Referred to as preference bias or search bias

- The version space CANDIDATE-ELIMINATION algorithm searches an incomplete hypothesis space (i.e., one that can express only a subset of the potentially teachable concepts).
  - Its inductive bias is in the form of a restriction on the set of hypotheses considered
    - Referred to as restriction bias or language bias.
Why Prefer Short Hypotheses?

- ID3 has an inductive bias for favoring shorter decision trees
- Occam’s Razor: Prefer the simplest hypothesis that fits the data.
- There are fewer short hypotheses than long ones
  - less likely to find a short hypothesis that coincidentally fits the data
  - many long hypotheses fail to generalize subsequently
  - prefer 5-node tree to fit 20 examples than 500 node tree
  - polynomial versus linear fit of noisy data

Issues in Decision Tree Learning

- How deeply to grow the decision tree
- Handling continuous attributes
- Choosing an appropriate attribute selection measure
- Handling training data with
  - missing attribute values
  - differing costs
- Improving computational efficiency
Avoiding Overfitting the Data

• Must be careful to avoid overfitting the data
  • when there is noise in the data or
  • when the number of training examples is too small to produce a representative sample of the true target function.

Overfitting the Data

Definition: 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 examples, but $h'$ has a smaller error than $h$ over the entire distribution of instances.
Overfitting in Decision Tree Learning

![Graph showing overfitting in decision tree learning.](image)

Learning which patients have a form of diabetes

Approaches To Prevent Overfitting

- Approaches that stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data.
- Approaches that allow the tree to overfit the data and then post-prune the tree.
How To Determine the Correct Size

1. Training and Validation: Use a separate set of examples, distinct from the training examples, to evaluate the utility of post-pruning nodes from the tree.
2. Use all available data for training, but apply a statistical test to estimate whether expanding (or pruning) a particular node is likely to produce an improvement.
3. Use an explicit measure of the complexity for encoding the training examples and the decision tree, halting growth when this encoding size is minimized.

Training and Validation

- Training Set: used to form learned hypotheses
- Validation Set:
  - used to evaluate the accuracy of this hypothesis over subsequent data
  - also, evaluate impact of pruning hypothesis
- Philosophy:
  - Validation set is unlikely to exhibit same random fluctuations as Training set
  - check against overfitting
- Typically validation set is one half size of training set
Reduced Error Pruning

- Reduced-error Pruning: Consider each node of the decision nodes in the tree to be candidates for pruning
- Pruning a decision tree consists of
  - removing a subtree rooted at the node
  - making it a leaf node
  - assigning it the most common classification of the training examples affiliated with that node.
- Nodes are removed only if the resulting pruned tree performs no worse than the original over the validation set.

Effect of Reduced-Error Pruning
Rule Post-Pruning

- Infer the decision tree from the training set, growing the tree until the training data is fit as well as possible and allowing the overfitting to occur.
- Convert the learned tree into an equivalent set of rules by creating 1 rule for each path from the root node to the leaf node.
- Prune (generalize) each rule by removing any preconditions that result in improving its estimated accuracy.
- Sort the pruned rules by their estimated accuracy and consider them in this sequence when classifying subsequent sequences.

Incorporating Continuous-Valued Attributes

<table>
<thead>
<tr>
<th>Temperature</th>
<th>40</th>
<th>48</th>
<th>60</th>
<th>72</th>
<th>80</th>
<th>90</th>
</tr>
</thead>
<tbody>
<tr>
<td>PlayTennis</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>
Data Mining (Decision Tree Algorithm)  
DCS 802, Spring 2002

(a)

(b)
### Alternative Measures for Selecting Attributes

\[
\text{SplitInformation}(S, A) \equiv -\sum_{i=1}^{c} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}
\]
Alternative Measures for Selecting Attributes,
continued

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

Handling Training Examples with Missing Attribute Values

- Strategies
  - assign it the value that is most common among training examples at node \( n \).
  - assign a probability to each of the possible values of \( A \) rather than simply assigning the most common value to \( A(x) \).
Handling Attributes with Differing Costs

- Strategies
  - replace information gain attribute selection measure by
    \[ \frac{Gain^2(S, A)}{Cost(A)} \]
  - use attribute selection measure
    \[ \frac{2^{Gain(S, A)} - 1}{(Cost(A) + 1)^w} \]

Summary

- Decision tree learning provides a practical method for concept learning and for learning discrete-valued functions.
- ID3 searches a complete hypothesis space (i.e., the space of decision trees can represent any discrete-valued function defined over discrete-valued instances).
- The inductive bias implicit in ID3 includes a \textit{preference} for smaller trees.
Summary, continued

- Overfitting the training data is an important issue in decision tree learning.
- A large variety of extensions to the basic ID3 algorithm has been developed by different researchers.