1 Introduction

Decision tree learning is one of the most widely used and practical methods for inductive inference. It is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree.

![Decision Tree Diagram]

Figure 1: A decision tree for the concept PlayTennis.

2 Decision Tree Representation

Decision trees classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. In the decision tree representation:

- Each internal node tests an attribute
- Each branch corresponds to attribute value
• Each leaf node assigns a classification

How would we represent:

• $\land, \lor, \text{XOR}$

• $(A \land B) \lor (C \land \neg D \land E)$

• $M$ of $N$

Decision trees represent a disjunction of conjunctions.

• Each path from root to a leaf is a conjunction of attribute tests.

• The tree itself is a disjunction of these conjunctions.

Figure 1 illustrates a typical learned decision tree. It corresponds to the expression:

$$
( \text{Outlook} = \text{Sunny} \land \text{Humidity} = \text{Normal} ) \\
\lor \\
( \text{Outlook} = \text{Overcast} ) \\
\lor \\
( \text{Outlook} = \text{Rain} \land \text{Wind} = \text{Weak} )
$$

3 Appropriate Problems for Decision Tree Learning

Decision tree learning is generally best suited to problems with the following characteristics:

• Instances describable by attribute-value pairs.

• Target function is discrete valued.

• Disjunctive hypothesis may be required.

• Possibly noisy training data.

Some examples of problems that fit to these characteristics are:

• Medical or equipment diagnosis

• Credit risk analysis

• Modelling calendar scheduling preferences
4 The Basic Decision Tree Learning Algorithm

The basic decision tree learning algorithm, ID3, employs a top-down, greedy search through the space of possible decision trees, beginning with the question "which attribute should be tested at the root of the tree?". The algorithm is given below:

**ID3(Examples, Target-attribute, Attributes)**
/* Examples: The training examples; */
/* Target-attribute: The attribute whose value is to be predicted by the tree; */
/* Attributes: A list of other attributes that may be tested by the learned decision tree. */
/* Return a decision tree that correctly classifies the given Examples */

Step 1: Create a Root node for the tree
Step 2: If all Examples are positive, Return the single-node tree Root, with label = +
Step 3: If all Examples are negative, Return the single-node tree Root, with label = -
Step 4: If Attributes is empty, Return the single-node tree Root, with label = most common value of Target-attribute in Examples
Step 5: Otherwise Begin

- A ← the attribute from Attributes that best (i.e., highest information gain) classifies Examples;
- The decision attribute for Root ← A;
- For each possible value, \( v_i \), of A,
  - Add a new tree branch below Root, corresponding to the test A=\( v_i \);
  - Let Examples(\( v_i \)) be the subset of Examples that have value \( v_i \) for A;
  - If 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
      ID3(Examples(\( v_i \)), Target-attribute, Attributes- A ))

End
Return Root

We want to select the attribute that is most useful for classifying examples. In order to measure the worth of an attribute a statistical property is defined, information gain, which measures how well a given attribute separates the training examples according to their target classification.

4.1 Entropy

In order to define information gain precisely, we begin by defining a measure called entropy, that characterizes the (im)purity of an arbitrary collection of examples. That is, it measures the homogeneity of examples.
Entropy(S) ≡ −p⊕ log₂ p⊕ − p⊖ log₂ p⊖

S is a sample of training examples
p⊕ is the proportion of positive examples in S
p⊖ is the proportion of negative examples in S

4.2 Information Gain

Information gain is simply the expected reduction in entropy caused by partitioning the examples according to this attribute.

\[
Gain(S, A) ≡ \text{Entropy}(S) − \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)
\]

Values(A) is the set of all possible values for attribute A
S_v is the subset of S for which attribute A has value v

4.3 An Illustrative Example

To illustrate the operation of ID3, let’s consider the learning task represented by the below examples.

<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>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>
In the first step of the algorithm, the topmost node of the decision tree is created. In order to determine the attribute that should be tested first in the tree, the information gain for attributes (Outlook, Temperature, Humidity, and Wind) are determined. The computation of information gain for Humidity and Wind is shown below.

The information gain for all four attributes are:

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

Since Outlook attribute provides the best prediction, it is selected as the decision attribute for the root node.

In Figure-3, the partially learned decision tree resulting from the first step of ID3 is shown. The final decision tree learned by ID3 from the 14 training examples is shown in Figure-1.
Hypothesis Space Search in Decision Tree Learning

The hypothesis space searched by ID3 is the set of possible decision trees. ID3 performs a hill-climbing search through the space of possible decision trees from simplest to increasingly complex, guided by the information gain heuristic. This search is depicted in Figure-4.

Capabilities and limitations of ID3:

- Hypothesis space is complete!
  Target function is surely in there...

- Outputs a single hypothesis (which one?)
  Can’t play 20 questions...

Figure 3: The partially learned decision tree resulting from the first step of ID3.

5 Hypothesis Space Search in Decision Tree Learning

The hypothesis space searched by ID3 is the set of possible decision trees. ID3 performs a hill-climbing search through the space of possible decision trees from simplest to increasingly complex, guided by the information gain heuristic. This search is depicted in Figure-4.

Capabilities and limitations of ID3:

- Hypothesis space is complete!
  Target function is surely in there...

- Outputs a single hypothesis (which one?)
  Can’t play 20 questions...
6 Inductive Bias in Decision Tree Learning

Note $H$ is the power set of instances $X$.

→ Unbiased?
Not really...

• Preference for short trees over larger trees, and for those with high information gain attributes near the root
• Bias is a *preference* for some hypotheses, rather than a *restriction* of hypothesis space \( H \)

• Occam’s razor prefer the shortest hypothesis that fits the data

6.1 Why Prefer Short Hypothesis?

Is ID3’s inductive bias favoring shorter decision trees a sound basis for generalizing beyond the training data?

**Occam’s razor** : Prefer the simplest hypothesis that fits the data.

Why prefer short hypotheses?

**Arguments in favor:**

• Fewer short hypotheses than long hypotheses
  
  – a short hypothesis that fits data is unlikely to be a coincidence
  
  – a long hypothesis that fits data might be a coincidence

**Arguments opposed:**

• There are many ways to define small sets of hypotheses
  
  e.g., all trees with a prime number of nodes that use attributes beginning with ”Z”

• What’s so special about small sets based on size of hypothesis?

7 Issues in Decision Tree Learning

Practical issues in learning decision trees include:

• Determining how deeply to grow the decision tree.

• Handling continues attributes.

• Choosing an appropriate attribute selection measure.

• Handling training data with missing attribute values.

• Handling attributes with differing costs.

• Improving computational efficiency.
7.1 Avoiding 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, overfit the training examples!!

Consider error of hypothesis $h$ over

- training data: $\text{error}_{\text{train}}(h)$
- entire distribution $D$ of data: $\text{error}_{\text{D}}(h)$

Hypothesis $h \in H$ OVERFITS training data if there is an alternative hypothesis $h' \in H$ such that

$$\text{error}_{\text{train}}(h) < \text{error}_{\text{train}}(h')$$
and
$$\text{error}_{\text{D}}(h) > \text{error}_{\text{D}}(h')$$

Consider adding a noisy training example #15 to the training examples we considered before:

Sunny, Hot, Normal, Strong, PlayTennis = No

What effect does it have on the earlier tree?

ID3 outputs a decision tree ($h$) that is more complex than the original tree from Figure-1($h'$). $h$ fits the training examples perfectly, whereas the simpler $h'$ will not.

Figure-5 illustrates the impact of overfitting in a typical application of decision tree learning.

How can we avoid overfitting?

- stop growing when data split not statistically significant
- grow full tree, then post-prune

How to select "best" tree?

- Measure performance over training data
- Measure performance over separate validation data set
- MDL: minimize $\text{size}(\text{tree}) + \text{size}(\text{misclassifications(tree)})$
7.1.1 Reduced Error Pruning

How exactly might we use a validation set to prevent overfitting? One approach is reduced-error pruning. Pruning a node consists of:

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

Reduced-error pruning produces the smallest version of the most accurate subtree. The impact of reduced-error pruning on the accuracy of the decision tree is illustrated in Figure-6.

- Training set: learn the tree
- Validation set: prune the tree
- Test set: estimate future classification accuracy

Split data into training and validation set.

Do until further pruning is harmful:

1. Evaluate impact on validation set of pruning each possible node (plus those below it)
2. Greedily remove the one that most improves validation set accuracy
7.1.2 Rule Post-Pruning

Rule-post pruning involves the following steps:

1. Infer the decision tree from the training set.
2. Convert the tree to equivalent set of rules.
3. Prune each rule independently of others.
4. Sort final rules into desired sequence for use.

In rule post-pruning, one rule is generated for each leaf node in the tree. For example, the two leftmost path of the tree in Figure-1 is translated into the rules:

IF (Outlook = Sunny) ∧ (Humidity = High) THEN PlayTennis = No

IF (Outlook = Sunny) ∧ (Humidity = Normal) THEN PlayTennis = Yes

Advantages of converting decision tree to rules before pruning:

- allows distinguishing among the different contexts in which a decision node is used
- removes the distinction between attribute tests that occur near the root of the tree and those that occur near the leaves
- improves readability

Figure 6: Effect of reduced-error pruning in decision tree learning.
7.2 Incorporating Continuous-Valued Attributes

Create a discrete attribute to test continuous values:

- Temperature = 82.5
- (Temperature > 72.3) = t,f

<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>

7.3 Attributes with Many Values

Problem:
- If attribute has many values, Gain will select it.
- Imagine using Date = Jun-3-1996 as attribute.

One approach: use GainRatio instead.

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

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

where \(S_i\) is subset of \(S\) for which \(A\) has value \(v_i\).

7.4 Unknown Attribute Values

What if some examples missing values of \(A\)?

Use training example anyway, sort through tree:

- If node \(n\) tests \(A\), assign most common value of \(A\) among other examples sorted to node \(n\).
- assign most common value of \(A\) among other examples with same target value.
- assign probability \(p_i\) to each possible value \(v_i\) of \(A\)
  - assign fraction \(p_i\) of example to each descendant in tree.

Classify new examples in the same fashion.
7.5 Attributes with Costs

Consider

- medical diagnosis, BloodTest has cost $150
- robotics, Width-from-1ft has cost 23 sec.

How to learn a consistent tree with minimum expected cost?

One approach: replace gain by

- Tan and Schlimmer (1990)
  \[
  \frac{\text{Gain}^2(S, A)}{\text{Cost}(A)}
  \]

- Nunez (1988)
  \[
  \frac{2^{\text{Gain}(S,A)} - 1}{(\text{Cost}(A) + 1)^{w}}
  \]

where \( w \in [0,1] \) determines importance of cost.

8 References

6. Online. A complete, java implementation of the decision-tree algorithm: http://www.cogs.susx.ac.uk/users/christ/crs/sai/lec15.html