Artificial Intelligence

Decision Trees

Andrea Torsello
Decision Trees

Complex decisions can often be expressed in terms of a series of questions:

What to do this Weekend?

• If my parents are visiting
  – We’ll go to the cinema

• If not
  – Then, if it’s sunny I’ll play tennis
  – But if it’s windy and I’m rich, I’ll go shopping
  – If it’s windy and I’m poor, I’ll go to the cinema
  – If it’s rainy, I’ll stay in
Decision Trees

These decisions can be expressed in terms of a tree.
Decision Trees

Decision trees can be written as
  • Horn clauses in first order logic

Read from the root to every tip
  • If this and this and this ... and this, then do this

In our example:
  • If no_parents and sunny_day, then play_tennis
  • no_parents \land sunny_day \rightarrow play_tennis
Decision Tree Learning

Decision tree can be seen as rules for performing a categorization

- E.g., “what kind of weekend will this be?”

Remember that we’re learning from examples

- Not turning thought processes into decision trees

We need examples put into categories

We also need attributes for the examples

- Attributes describe examples (background knowledge)
- Each attribute takes only a finite set of values
The ID3 Algorithm

The major question in decision tree learning
- Which nodes to put in which positions
- Including the root node and the leaf nodes

ID3 uses a measure called Information Gain
- Based on a notion of entropy
  - “Impurity in the data”
- Used to choose which node to put in next

Node with the highest information gain is chosen
- When there are no choices, a leaf node is put on
Entropy

For examples in a binary categorization
- Where $p_+$ is the proportion of positives

Given a set of examples, $S$
- And $p_-$ is the proportion of negatives

$$Entropy(S) = -p_+ \log_2(p_+) - p_- \log_2(p_-)$$

For examples in categorizations $c_1$ to $c_n$
- Where $p_n$ is the proportion of examples in $c_n$

$$Entropy(S) = \sum_{i=1}^{n} -p_i \log_2(p_i)$$
Entropy

Each category adds to the whole measure

When $\pi$ is near to 1

- (Nearly) all the examples are in this category
  - So it should score low for its bit of the entropy
- $\log_2(\pi)$ gets closer and closer to 0
  - And this part dominates the overall calculation
  - So the overall calculation comes to nearly 0 (which is good)

When $\pi$ is near to 0

- (Very) few examples are in this category
  - So it should score low for its bit of the entropy
- $\log_2(\pi)$ gets larger (more negative), but does not dominate
  - Hence overall calculation comes to nearly 0 (which is good)
Information Gain

Given set of examples $S$ and an attribute $A$

- $A$ can take values $v_1 \ldots v_m$
- Let $S_v = \{\text{examples which take value } v \text{ for attribute } A\}$

Calculate $\text{Gain}(S, A)$

- Estimates the reduction in entropy we get if we know the value of attribute $A$ for the examples in $S$

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$
Example Calculation of Information Gain

Suppose we have a set of examples

- \( S = \{s_1, s_2, s_3, s_4\} \)

- In a binary categorization
  - With one positive example and three negative examples
  - The positive example is \( s_1 \)

And Attribute A

- Which takes values \( v_1, v_2, v_3 \)

- \( s_1 \) takes value \( v_2 \) for A
- \( s_2 \) takes value \( v_2 \) for A
- \( s_3 \) takes value \( v_3 \) for A
- \( s_4 \) takes value \( v_1 \) for A
Recall that

- Entropy(S) = -p_+ \log_2(p_+) - p_- \log_2(p_-)

From binary categorisation, we know that

- p_+ = \frac{1}{4} and p_- = \frac{3}{4}

Hence

- Entropy(S) = -\left(\frac{1}{4}\right) \log_2\left(\frac{1}{4}\right) - \left(\frac{3}{4}\right) \log_2\left(\frac{3}{4}\right) = 0.811

Note that, by convention:

0*\log_2(0) is taken to be 0
Remember that

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

And that \( S_v = \{ \text{set of example with value V for A} \} \)

- So, \( S_v = \{s_4\}, \ S_v = \{s_1, s_2\}, \ S_v = \{s_3\} \)

Now, \( (|S_v|/|S|) \ast \text{Entropy}(S_v) \)

\[
= (1/4) \ast (-0/1)\log_2(0/1)-(1/1)\log_2(1/1)) \\
= (1/4) \ast (0 - (1)\log_2(1)) = (1/4)(0-0) = 0
\]

Similarly, \((|S_v|/|S|) = 0.5 \) and \((|S_v|/|S|) = 0 \)

So, we have:

- \( \text{Gain}(S,A) = 0.811 - (0+1/2+0) = 0.311 \)
The ID3 Algorithm

Given a set of examples, S
  • Described by a set of attributes $A_i$
  • Categorised into categories $c_j$

1. Choose the root node to be attribute $A$
   • Such that $A$ scores highest for information gain
     - Relative to $S$, i.e., $\text{gain}(S,A)$ is the highest over all attributes

2. For each value $v$ that $A$ can take
   • Draw a branch and label each with corresponding $v$
     - Then see the options in the next slide!
The ID3 Algorithm

For each branch you’ve just drawn (for value v)

- If $S_v$ only contains examples in category c
  - Then put that category as a leaf node in the tree

- If $S_v$ is empty
  - Then find the default category (which contains the most examples from S)
  - Put this default category as a leaf node in the tree

- Otherwise
  - Remove A from attributes which can be put into nodes
  - Replace S with $S_v$
  - Find new attribute A scoring best for Gain(S, A)
  - Start again at part 2

Make sure you replace S with $S_v$
Attribute A scores highest for Gain(S,A)

Attribute B scores highest for Gain(Sw,B)

Leaf node category c

Sv must contain only examples in category c

Default leaf node d

Sw must have no examples taking value x for attribute B, and d must be the category containing the most members of Sw
<table>
<thead>
<tr>
<th>Weekend</th>
<th>Weather</th>
<th>Parents</th>
<th>Money</th>
<th>Decision (Category)</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1</td>
<td>Sunny</td>
<td>Yes</td>
<td>Rich</td>
<td>Cinema</td>
</tr>
<tr>
<td>W2</td>
<td>Sunny</td>
<td>No</td>
<td>Rich</td>
<td>Tennis</td>
</tr>
<tr>
<td>W3</td>
<td>Windy</td>
<td>Yes</td>
<td>Rich</td>
<td>Cinema</td>
</tr>
<tr>
<td>W4</td>
<td>Rainy</td>
<td>Yes</td>
<td>Poor</td>
<td>Cinema</td>
</tr>
<tr>
<td>W5</td>
<td>Rainy</td>
<td>No</td>
<td>Rich</td>
<td>Stay in</td>
</tr>
<tr>
<td>W6</td>
<td>Rainy</td>
<td>Yes</td>
<td>Poor</td>
<td>Cinema</td>
</tr>
<tr>
<td>W7</td>
<td>Windy</td>
<td>No</td>
<td>Poor</td>
<td>Cinema</td>
</tr>
<tr>
<td>W8</td>
<td>Windy</td>
<td>No</td>
<td>Rich</td>
<td>Shopping</td>
</tr>
<tr>
<td>W9</td>
<td>Windy</td>
<td>Yes</td>
<td>Rich</td>
<td>Cinema</td>
</tr>
<tr>
<td>W10</td>
<td>Sunny</td>
<td>No</td>
<td>Rich</td>
<td>Tennis</td>
</tr>
</tbody>
</table>
\[ S = \{W1, W2, ..., W10\} \]

Firstly, we need to calculate:
- Entropy\((S) = ... = 1.571\)

Next, we need to calculate information gain
- For all the attributes we currently have available (which is all of them at the moment)
- Gain\((S, \text{weather}) = ... = 0.7\)
- Gain\((S, \text{parents}) = ... = 0.61\)
- Gain\((S, \text{money}) = ... = 0.2816\)

Hence, the weather is the first attribute to split on
- Because this gives us the biggest information gain
So, this is the top of our tree:

Now, we look at each branch in turn

- In particular, we look at the examples with the attribute prescribed by the branch

$S_{\text{sunny}} = \{W1, W2, W10\}$

- Categorisations are cinema, tennis and tennis for $W1, W2$ and $W10$

- What does the algorithm say?
  - Set is neither empty, nor a single category
  - So we have to replace $S$ by $S_{\text{sunny}}$ and start again
Need to choose a new attribute to split on
• Cannot be weather, of course – we’ve already had that

So, calculate information gain again:
• $\text{Gain}(S_{\text{sunny}}, \text{parents}) = \ldots = 0.918$
• $\text{Gain}(S_{\text{sunny}}, \text{money}) = \ldots = 0$

Hence we choose to split on parents

<table>
<thead>
<tr>
<th>Weekend</th>
<th>Weather</th>
<th>Parents</th>
<th>Money</th>
<th>Decision</th>
</tr>
</thead>
<tbody>
<tr>
<td>W1</td>
<td>Sunny</td>
<td>Yes</td>
<td>Rich</td>
<td>Cinema</td>
</tr>
<tr>
<td>W2</td>
<td>Sunny</td>
<td>No</td>
<td>Rich</td>
<td>Tennis</td>
</tr>
<tr>
<td>W10</td>
<td>Sunny</td>
<td>No</td>
<td>Rich</td>
<td>Tennis</td>
</tr>
</tbody>
</table>
If it’s sunny and the parents have turned up
- Then, looking at the table in previous slide
- There’s only one answer: go to cinema

If it’s sunny and the parents haven’t turned up
- Then, again, there’s only one answer: play tennis

Hence our decision tree looks like this
Avoiding Overfitting

Decision trees can be learned to perfectly fit the data given
• This is probably overfitting
• The answer is a memorisation, rather than generalisation

Avoidance method 1:
• Stop growing the tree before it reaches perfection

Avoidance method 2:
• Grow to perfection, then prune it back afterwards
  – Most useful of two methods in practice

Decision tree algorithms are fairly robust to errors
• In the actual classifications
• In the attribute-value pairs
• In missing information
Decision Boundary

All the attributes used for categorization form an n-dimensional space

Decisions can be of the form $x_i > \text{constant}$

- Divides the space into axis aligned rectangles