Section 18.3 Learning Decision Trees

CS4811 - Artificial Intelligence

Nilufer Onder
Department of Computer Science
Michigan Technological University
Outline

Attribute-based representations

Decision tree learning as a search problem

A greedy algorithm
Decision trees

- A decision tree allows a classification of an object by testing its values for certain properties.
- An example is the 20 questions game. A player asks questions to an answerer and tries to guess the object that the answerer chose at the beginning of the game.
- The objective of decision tree learning is to learn a tree of questions which determines class membership at the leaf of each branch.
- There was an online example at the following address: http://myacquire.com/aiinc/whalewatcher/
Possible decision tree

- See flukes?
  - Yes
    - See dorsal fin?
      - Yes
        - Size?
          - Vlg
            - Blue whale
          - Med
            - Blow forward?
              - Yes
                - Sperm whale
              - No
                - Humpback whale
      - No
  - No (see next page)

- Size med?
  - Yes
    - Blows?
      - 1
        - Gray whale
      - 2
        - Right whale
  - No
    - Size?
      - Ig
      - Vsm
      - Bowhead whale
      - Narwhal whale
Possible decision tree (cont’d)

- **see flukes?**
  - yes
  - (see previous page)
  - no
- **see dorsal fin?**
  - no
  - yes
    - **blow?**
      - yes
      - **size?**
        - lg
        - dorsal fin and blow visible at the same time?
          - yes
            - sei whale
          - no
            - fin whale
        - sm
            - dorsal fin tall and pointed?
              - yes
                - killer whale
              - no
                - northern bottlenose whale
What might the original data look like?

<table>
<thead>
<tr>
<th>Place</th>
<th>Time</th>
<th>Group</th>
<th>Fluke</th>
<th>Dorsal fin</th>
<th>Dorsal shape</th>
<th>Size</th>
<th>Blow</th>
<th>...</th>
<th>Blow fwd</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kaikora</td>
<td>17:00</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>small triang.</td>
<td>Very large</td>
<td>Yes</td>
<td></td>
<td>No</td>
<td>Blue whale</td>
</tr>
<tr>
<td>Kaikora</td>
<td>7:00</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>small triang.</td>
<td>Very large</td>
<td>Yes</td>
<td></td>
<td>No</td>
<td>Blue whale</td>
</tr>
<tr>
<td>Kaikora</td>
<td>8:00</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>small triang.</td>
<td>Very large</td>
<td>Yes</td>
<td></td>
<td>No</td>
<td>Blue whale</td>
</tr>
<tr>
<td>Kaikora</td>
<td>9:00</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>squat triang.</td>
<td>Medium</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Sperm whale</td>
</tr>
<tr>
<td>Cape Cod</td>
<td>18:00</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Irregular</td>
<td>Medium</td>
<td>Yes</td>
<td></td>
<td>No</td>
<td>Hump-back whale</td>
</tr>
<tr>
<td>Cape Cod</td>
<td>20:00</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Irregular</td>
<td>Medium</td>
<td>Yes</td>
<td></td>
<td>No</td>
<td>Hump-back whale</td>
</tr>
<tr>
<td>Newb. Port</td>
<td>18:00</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Curved</td>
<td>Large</td>
<td>Yes</td>
<td></td>
<td>No</td>
<td>Fin whale</td>
</tr>
<tr>
<td>Cape Cod</td>
<td>6:00</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>None</td>
<td>Medium</td>
<td>Yes</td>
<td></td>
<td>No</td>
<td>Right whale</td>
</tr>
</tbody>
</table>
The search problem

This is an *attribute-based representation* where examples are described by *attribute values* (Boolean, discrete, continuous, etc.)

*Classification* of examples is positive (T) or negative (F).

Given a table of observable properties, search for a decision tree that

- correctly represents the data
  (for now, assume that the data is noise-free)
- is as small as possible

What does the search tree look like?
Predicate as a decision tree

The predicate $\text{CONCEPT}(x) \iff A(x) \land (\neg B(x) \lor C(x))$ can be represented by the following decision tree:

Example:
A mushroom is poisonous iff it is yellow and small, or yellow, big and spotted
- $x$ is a mushroom
- $\text{CONCEPT} = \text{POISONOUS}$
- $A = \text{YELLOW}$
- $B = \text{BIG}$
- $C = \text{SPOTTED}$
- $D = \text{FUNNEL-CAP}$
- $E = \text{BULKY}$
The training set

<table>
<thead>
<tr>
<th>Ex. #</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>CONCEPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>2</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>3</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>4</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>5</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>6</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>7</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>8</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>9</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>10</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>11</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>12</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>13</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>
Possible decision tree

<table>
<thead>
<tr>
<th>Ex. #</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>CONCEPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>2</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>3</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>4</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>5</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>6</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>7</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>8</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>9</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>10</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>11</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>12</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>13</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>
Smaller decision tree

CONCEPT ⇔
(D ∧ (¬E v A)) v
(C ∧ (B v ((E ∧ ¬A) v A)))

CONCEPT ⇔ A ∧ (¬B v C)

KIS bias → Build smallest decision tree

Computationally intractable problem → greedy algorithm
The distribution of the training set is:

True: 6, 7, 8, 9, 10, 13
False: 1, 2, 3, 4, 5, 11, 12
The distribution of training set is:

True: 6, 7, 8, 9, 10, 13
False: 1, 2, 3, 4, 5, 11, 12

Without testing any observable predicate, we could report that CONCEPT is False (majority rule) with an estimated probability of error $P(E) = 6/13$
The distribution of training set is:

True: 6, 7, 8, 9, 10, 13
False: 1, 2, 3, 4, 5, 11, 12

Without testing any observable predicate, we could report that CONCEPT is False (majority rule) with an estimated probability of error $P(E) = \frac{6}{13}$.

Assuming that we will only include one observable predicate in the decision tree, which predicate should we test to minimize the probability of error?
How to compute the probability of error (1)

If we test only A, we will report that CONCEPT is True if A is True (majority rule) and False otherwise.

The estimated probability of error is:
$$\Pr(E) = \frac{8}{13} \times \frac{2}{8} + \frac{5}{13} \times \frac{0}{5} = \frac{2}{13}$$

8/13 is the probability of getting True for A, and 2/8 is the probability that the report was incorrect (we are always reporting True for the concept).
How to compute the probability of error (2)

If we test only A, we will report that CONCEPT is True if A is True (majority rule) and False otherwise.

The estimated probability of error is:
Pr(E) = \((8/13) \times (2/8) + (5/13) \times (0/5) = 2/13\)

5/13 is the probability of getting False for A, and 0 is the probability that the report was incorrect (we are always reporting False for the concept).
Assume it’s A

If we test only A, we will report that CONCEPT is True if A is True (majority rule) and False otherwise.

The estimated probability of error is:
\[ \Pr(E) = \frac{8}{13} \times \frac{2}{8} + \frac{5}{8} \times 0 = \frac{2}{13} \]
Assume it’s B

If we test only B, we will report that CONCEPT is False if B is True and True otherwise.

The estimated probability of error is:
\[ \text{Pr}(E) = \frac{6}{13} \times \frac{2}{6} + \frac{7}{13} \times \frac{3}{7} = \frac{5}{13} \]
Assume it’s C

If we test only C, we will report that CONCEPT is True if C is True and False otherwise.

The estimated probability of error is:
\[ Pr(E) = \frac{8}{13} \times \frac{3}{8} + \frac{5}{13} \times \frac{1}{5} = \frac{4}{13} \]
Assume it’s D

If we test only D, we will report that CONCEPT is True if D is True and False otherwise.

The estimated probability of error is:
\[ \Pr(E) = \left(\frac{5}{13}\right) \times \left(\frac{2}{5}\right) + \left(\frac{8}{13}\right) \times \left(\frac{3}{8}\right) = \frac{5}{13} \]
Assume it’s E

If we test only E we will report that CONCEPT is False, independent of the outcome

The estimated probability of error is:
\[ \Pr(E) = \frac{8}{13} \times \frac{4}{8} + \frac{5}{13} \times \frac{2}{5} = \frac{6}{13} \]
Probability of error for each

- If A: 2/13
- If B: 5/13
- If C: 4/13
- If D: 5/13
- If E: 6/13

So, the best predicate to test is A
Choice of second predicate

The majority rule gives the probability of error $Pr(E|A) = 1/8$ and $Pr(E) = 1/13$
Choice of third predicate

True:

False: 11,12

7
Final Tree

\[ L \equiv \text{CONCEPT} \iff A \land (C \lor \neg B) \]
The decision tree learning algorithm

function \textsc{Decision-Tree-Learning} \( (\text{examples}, \text{attributes}, \text{parent-examples}) \)
returns a tree

if \( \text{examples} \) is empty then
    return \textsc{Plurality-Value}(\text{parent-examples})
else if all \( \text{examples} \) have the same classification then
    return the classification
else if \( \text{attributes} \) is empty then
    return \textsc{Plurality-Value}(\text{examples})
else
    \( A \leftarrow \arg\max_{a \in \text{attributes}} \text{Importance}(a, \text{examples}) \)
    \( \text{tree} \leftarrow \) a new decision tree with root test \( A \)
    for each value \( v_k \) of \( A \) do
        \( \text{exs} \leftarrow \{ e : e \in \text{examples} \text{ and } e.A = v_k \} \)
        \( \text{subtree} \leftarrow \textsc{Decision-Tree-Learning}(\text{exs}, \text{attributes-}A, \text{examples}) \)
        add a branch to \( \text{tree} \) with label \( (A = v_k) \) and subtree \( \text{subtree} \)
    return \( \text{tree} \)
Notes on the algorithm

- Notice that the “probability of error” calculations boil down to summing up the “minority numbers” and dividing by the total number of examples in that category. This is due to fraction cancellations. Probability of error is:

\[
\frac{\text{minority 1} + \text{minority 2} + \ldots}{\text{total number of examples in this category}}
\]

- After an attribute is selected take only the examples that have the attribute as labelled on the branch.
What happens if there is noise in the training set?

Consider a very small but inconsistent data set:

<table>
<thead>
<tr>
<th>A</th>
<th>classification</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

![Decision Tree Diagram]
Issues in learning decision trees

- If data for some attribute is missing and is hard to obtain, it might be possible to *extrapolate* or use unknown.
- If some attributes have continuous values, *groupings* might be used.
- If the data set is too large, one might use *bagging* to select a sample from the training set. Or, one can use *boosting* to assign a weight showing importance to each instance. Or, one can *divide the sample set into subsets* and train on one, and test on others.
How large is the hypothesis space?

How many decision trees with $n$ Boolean attributes?

$= \text{number of Boolean functions}$

$= \text{number of distinct truth tables with } 2^n \text{ rows.}$

$= 2^{2^n}$
Using “probability of error”

- The “probability of error” is based on a measure of the quantity of information that is contained in the truth value of an observable attribute.
- It shows how predictable the classification is after getting information about an attribute.
- The lower the probability of error, the higher the predictability.
- The attribute with the minimal probability of error yields the maximum predictability. That is what we chose A at the root of the decision tree.
Using information theory

- **Entropy** gives information about unpredictability.
- The scale is to use 1 bit to answer a Boolean question with prior $< 0.5, 0.5 >$. This is least predictability (highest unpredictability).
- Information answers questions: the more clueless we are about the answer initially, the more information is contained in the answer. i.e., we have a *gain* after getting an answer about attribute A.
- We select the attribute with the highest gain.
- Let $p$ be the number of positive examples, and $n$ the number of negative examples. Entropy$(p, n)$ is defined as

\[-p \log_2 p - n \log_2 n\]
Information gain

- **Gain(A)** is the expected reduction on entropy after getting an answer on attribute A.
- Let \( p_i \) be the number of positive examples when the answer to A is \( i \), and \( n_i \) be the number of negative examples when the answer to A is \( i \).
- Assuming two possible answers, Gain(A) is defined as

\[
\text{entropy}(p, n) - \frac{p_1 + n_1}{p + n} \text{entropy}(p_1, n_1) - \frac{p_2 + n_2}{p + n} \text{entropy}(p_2, n_2)
\]
Example

Assuming two possible answers, Gain(A) is defined as

\[
\text{entropy}(p, n) - \frac{p_1 + n_1}{p + n}\text{entropy}(p_1, n_1) - \frac{p_2 + n_2}{p + n}\text{entropy}(p_2, n_2)
\]

Initially there are 6 positive and 7 negative examples. Entropy(6,7) = 0.9957

There are 6 positive and 2 negative examples for A being true and 0 positive and 5 negative example for A being false. Therefore the gain is

\[
0.9957 - \frac{8}{13} \times \text{entropy}(6, 2) - \frac{5}{13} \times \text{entropy}(5, 0) =
\]

\[
0.9957 - \frac{8}{13} \times 0.8113 - \frac{5}{13} \times 0 = 0.4965
\]
The gain values are:
A: 0.4992
B: 0.0414
C: 0.1307
D: 0.0349
E: 0.0069
Summary

- Decision tree learning is a *supervised learning* paradigm.
- The *hypothesis* is a decision tree.
- The greedy algorithm uses *information gain* to decide which attribute should be placed at each node of the tree.
- Due to the greedy approach, the decision tree might not be optimal but the algorithm is fast.
- If the data set is complete and not noisy, then the learned decision tree will be accurate.
Sources for the slides

- AIMA textbook (3rd edition)
- AIMA slides: http://aima.cs.berkeley.edu/
- Jean-Claude Latombe’s CS121 slides http://robotics.stanford.edu/~latombe/cs121 (Accessed prior to 2009)