Decision Tree Learning

Learning decision trees

Problem: decide whether to wait for a table at a restaurant, based on the following attributes:
- Alternate: is there an alternative restaurant nearby?
- Bar: is there a comfortable bar area to wait in?
- Fri/Sat: is today Friday or Saturday?
- Hungry: are we hungry?
- Patrons: number of people in the restaurant (None, Some, Full)
- Rain: is it raining outside?
- Reservation: have we made a reservation?
- Type: kind of restaurant (French, Italian, Thai, Burger)
- WaitEst: estimated waiting time (0-10, 10-30, 30-60, >60)

Attribute-based representations
- Examples described by attribute values (Boolean, discrete, continuous)
- Example situations where I will/won’t wait for a table:

Decision trees
- Decision trees: a form of representation for hypotheses (classification rules)
- Examples: the “true” tree for deciding whether to wait:

Expressiveness
- Decision trees can express any function of the input attributes.
- E.g., for Boolean functions, truth table row → path to leaf:

Decision tree learning
- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose “most significant” attribute as root of (sub)tree

Expressiveness
- Trivially, there is a consistent decision tree for any training set with one path to leaf for each example but it probably won’t generalize to new examples
- Prefer to find compact decision trees
Choosing an attribute

- Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"

- Patrons? is a better choice
- Need a measure of quality for an attribute

Information theory (cont.)

- Consider a question that splits the set of numbers 0,..., 1023 into two sets.
- The average information left after a split of N numbers into two sets of size \( n \) and \( p \):
  \[ \text{Entropy}(n,N) + \text{Entropy}(p,N) \]
- The average information provided by a question:
  \[ \text{log} N 	imes (\text{Entropy}(n,N) + \text{Entropy}(p,N)) \]
- The definition for a multinomial distribution:
  \[ \text{Entropy}(p_1,\ldots,p_N) = \text{log} N \sum_i p_i \log p_i \]

Digression: information theory

- I am thinking of an integer between 0 and 1,023. You want to guess it using the fewest number of questions.
- Most of us would ask "is it between 0 and 512?"
- This is a good strategy because it provides the most information about the unknown number.
- It provides the first binary digit of the number.
- Initially you need to obtain \( \text{log}_2(1024) = 10 \) bits of information. After the first question you only need \( \text{log}_2(512) = 9 \) bits.

Example: triangles and squares

<table>
<thead>
<tr>
<th>#</th>
<th>Color</th>
<th>Outline</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>green</td>
<td>thick</td>
<td>yes</td>
</tr>
<tr>
<td>2</td>
<td>green</td>
<td>dashed</td>
<td>yes</td>
</tr>
<tr>
<td>3</td>
<td>yellow</td>
<td>dashed</td>
<td>yes</td>
</tr>
<tr>
<td>4</td>
<td>red</td>
<td>solid</td>
<td>yes</td>
</tr>
<tr>
<td>5</td>
<td>red</td>
<td>solid</td>
<td>yes</td>
</tr>
<tr>
<td>6</td>
<td>red</td>
<td>solid</td>
<td>yes</td>
</tr>
<tr>
<td>7</td>
<td>green</td>
<td>solid</td>
<td>yes</td>
</tr>
<tr>
<td>8</td>
<td>green</td>
<td>solid</td>
<td>yes</td>
</tr>
<tr>
<td>9</td>
<td>yellow</td>
<td>solid</td>
<td>yes</td>
</tr>
<tr>
<td>10</td>
<td>red</td>
<td>solid</td>
<td>yes</td>
</tr>
<tr>
<td>11</td>
<td>green</td>
<td>solid</td>
<td>yes</td>
</tr>
<tr>
<td>12</td>
<td>green</td>
<td>solid</td>
<td>yes</td>
</tr>
<tr>
<td>13</td>
<td>yellow</td>
<td>solid</td>
<td>yes</td>
</tr>
<tr>
<td>14</td>
<td>red</td>
<td>dashed</td>
<td>yes</td>
</tr>
</tbody>
</table>

Entropy

- Graph of the entropy function for a binomial distribution:
- Maximal when \( p = 1/2 \).

Entropy

- 5 triangles
- 9 squares
- class probabilities
  \[ p(\square) = \frac{9}{14} \]
  \[ p(\triangle) = \frac{5}{14} \]
- entropy
  \[ I = -\frac{9}{14} \text{log}_2 \frac{9}{14} - \frac{5}{14} \text{log}_2 \frac{5}{14} = 0.940 \text{ bits} \]
Entropy after partitioning

\[ H(\text{red}) = -\frac{2}{3} \log_2 \frac{2}{3} - \frac{1}{3} \log_2 \frac{1}{3} = 0.971 \text{ bits} \]
\[ H(\text{green}) = -\frac{2}{3} \log_2 \frac{2}{3} - \frac{1}{3} \log_2 \frac{1}{3} = 0.971 \text{ bits} \]
\[ H(\text{yellow}) = \frac{4}{9} \log_2 \frac{4}{9} - 0 \cdot \log_2 \frac{1}{9} = 0.0 \text{ bits} \]

\[ \text{Entropy after partitioning} \]
\[ H(\text{red}) = \frac{3}{11} \text{ bits} \]
\[ H(\text{green}) = \frac{4}{11} \text{ bits} \]
\[ H(\text{yellow}) = 0.0 \text{ bits} \]
\[ H(\text{Color}) = \sum p(v) H(v) = \frac{5}{11} \cdot 0.971 + \frac{5}{11} \cdot 0.971 + \frac{1}{11} \cdot 0 = 0.694 \text{ bits} \]

Information Gain

\[ IG(\text{Color}) = I - H(\text{Color}) = I - \sum p(v) H(v) = 0.940 - 0.694 = 0.246 \text{ bits} \]

\[ IG(\text{Outline}) = 0.971 - 0 = 0.971 \text{ bits} \]
\[ IG(\text{Dot}) = 0.971 - 0 = 0.971 \text{ bits} \]

Attributes
- IG(Color) = 0.246
- IG(Outline) = 0.151
- IG(Dot) = 0.048

Heuristic: attribute with the highest gain is chosen for making a split

Example

Gain(Outline) = 0.971 - 0 = 0.971 bits
Gain(Dot) = 0.971 - 0 = 0.971 bits

Example (cont.)
Example (cont.)

Decision Tree

Issue with IG

- IG favors attributes with many values
- Such attribute splits S to many subsets, and if these are small, they will tend to be pure anyway
- One way to rectify this is through a corrected measure of information gain ratio:
  \[
  \text{GainRatio}(A) = \frac{\text{IG}(A)}{\text{IntrinsicInfo}(A)}
  \]

\[\text{IntrinsicInfo}(A) \text{ is the entropy associated with the distribution of the attribute: } \sum_{i=1}^{k} p_i \log p_i, \text{ where } p_i \text{ is the probability of observing the } i^{th} \text{ value of } A.\]

Information Gain and Information Gain Ratio

| A         | |v(A)| | Gain(A) | GainRatio(A) |
|-----------|-------|------|---------|-------------|
| Color     | 3     | 0.247| 0.156   |
| Outline   | 2     | 0.152| 0.152   |
| Dot       | 2     | 0.048| 0.049   |

The Gini Index

- Another sensible measure of impurity (i and j are classes)
  \[
  \text{Gini} = \sum_{i \neq j} p(i)p(j)
  \]
- After applying a split on attribute A, the resulting Gini index is
  \[
  \text{Gini}(A) = \sum_{v} p(v) \sum_{i \neq j} p(i|v)p(j|v)
  \]
- Gini can be interpreted as expected error rate

The Gini Index: example

- For class \( C = \{1, 2, 3\} \):
  \[
  p(C) = \frac{9}{14}
  \]
  \[
  p(D) = \frac{5}{14}
  \]

\[
\text{Gini} = \sum_{i \neq j} p(i)p(j) = \frac{9}{14} \times \frac{5}{14} = 0.230
\]
The Gini Index: example (cont.)

\[
Gini(A) = \sum_{i} \sum_{j} p(i)p(j)
\]

\[
Gini(\text{Color}) = \frac{5}{14} \times \left( \frac{2}{5} \times \frac{1}{5} + \frac{5}{7} \times \frac{3}{5} \right) + \frac{4}{14} \times \frac{1}{5} \times \frac{1}{5} = 0.171
\]

The Gini Gain Index

\[
Gini(\text{Color}) = \frac{9}{14} \times \frac{5}{14} = 0.230
\]

\[
GiniGain(\text{Color}) = 0.230 - 0.171 = 0.058
\]

Three Impurity Measures

<table>
<thead>
<tr>
<th>A</th>
<th>Gain(A)</th>
<th>GainRatio(A)</th>
<th>GiniGain(A)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td>0.247</td>
<td>0.156</td>
<td>0.058</td>
</tr>
<tr>
<td>Outline</td>
<td>0.152</td>
<td>0.152</td>
<td>0.046</td>
</tr>
<tr>
<td>Det</td>
<td>0.048</td>
<td>0.049</td>
<td>0.025</td>
</tr>
</tbody>
</table>

Back to the restaurant example

- Decision tree learned from the 12 examples:

- Substantially simpler than "true" tree - a more complex hypothesis isn’t justified by small amount of data

Continuous variables

- If variables are continuous we can bin them.
- Alternative: learn a simple classifier on a single dimension, e.g. find decision point, classifying all data to the left in one class and all data to the right in the other

\[
\text{attribute } i \quad \text{threshold} \quad +1 \quad -1
\]

\[
\text{So, we choose an attribute, a sign } +/\text{ and and a threshold. This determines a half space to be } +1 \text{ and the other half } -1.
\]

When to Stop?

- If we keep splitting until perfect classification we might over-fit.
- Heuristics:
  - Stop when splitting criterion is below some threshold
  - Stop when number of examples in each leaf node is below some threshold
- Alternative: prune the tree; potentially better than stopped splitting, since split may be useful at a later point.
Comments on decision trees

- Fast training (complexity?)
- Established commercial software (e.g. CART, C4.5, C5.0)
- Users like them because they produce rules which can supposedly be interpreted (but decision trees are very sensitive with respect to training data)
- Able to handle missing data (how?)

When are decision trees useful?

- Limited accuracy for problems with many variables. Why?

Classification by committee

- An example of a committee classifier: a classifier that bases its prediction on a set of classifiers.
- Output: a majority vote
- If the errors made by the individual classifiers are independent the committee will perform well.

Random Forests

- A committee algorithm that combines decision trees (Breiman, 2001)
- To train a tree in a random forest:
  - Choose a training set by picking \( n \) examples with replacement from the \( n \) training examples (a bootstrap sample).
  - For each node of the tree, randomly choose \( m \) variables on which to base variable choice at that node.

Advantages of random forests

- State of the art performance
- Works well for high dimensional data.
- Estimates the importance of variables in determining classification.
- It generates an estimate of the generalization error as the forest building progresses.
-_handles missing data well.
- Scalable to large datasets.

Bagging

- The idea behind random forests can be applied to any classifier and is called bagging:
  - Choose M bootstrap samples.
  - Train M different classifiers on these bootstrap samples.
  - For a new query, average or take a majority vote.
- Other committee methods: Boosting (Freund & Schapire, 1995)
Classification: summary

- Classifiers we learned:
  - Nearest neighbors
  - Decision trees
  - Random forests
  - Naïve Bayes

- How do we choose which classifier to use?
- How do we choose model parameters?