Decision Trees

Example Decision Tree

Applications

- Credit-card companies and banks develop DT's to decide whether to grant a card or loan.
- Medical apps, e.g., given information about patients, decide which will benefit from a new drug.
- Many others.

Issues

- How do you use a decision tree?
  - I.e., given a record, how do you decide whether it is good or bad?
- How do you design decision trees?
  - Ad-hoc: decide yourself.
  - Training: algorithm to construct "best" DT from given data.
  - Hope future records match the training set.

Designing a Decision Tree

- Typically, we are given data consisting of a number of records, perhaps representing individuals.
- Each record has a value for each of several attributes.
  - Often binary attributes, e.g., "has dog."
  - Sometimes numeric, e.g. "age", or discrete, multiway, like "school attended."

Designing a Decision Tree 2

- Records are classified into "good" or "bad."
  - More generally: some number of outcomes.
- The goal is to make a small number of tests involving attributes to decide as best we can whether a record is good or bad.
Using a Decision Tree

◆ Given a record to classify, start at the root, and answer the question at the root for that record.
  ◆ E.g., is the record for a married person?
  ◆ Move next to the indicated child.
  ◆ Recursively, apply the DT rooted at that child, until we reach a decision.

Training Sets

◆ Decision-tree construction is today considered a type of “machine learning.”
◆ We are given a training set of example records, properly classified, with which to construct our decision tree.

Example

◆ Here is the data on which our example DT was based:

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>G</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>G</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>G</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>B</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>B</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>B</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>B</td>
</tr>
</tbody>
</table>

Binary Attributes

◆ When all attributes are binary, we can pick an attribute to place at the root by considering how nonrandom are the sets of records that go to each side.
◆ Branches correspond to the value of the chosen attribute.

Entropy: A Measure of Goodness

◆ Consider the pools of records on the “yes” and “no” sides.
◆ If fraction $p$ on on a side are “good,” the entropy of that side is $-(p \log_2 p + (1-p) \log_2 (1-p))$.
  $= p \log_2 (1/p) + (1-p) \log_2 (1/(1-p))$
◆ Pick attribute that minimizes maximum entropies of the sides.

Shape of Entropy Function
Intuition

◆ Entropy 1 = random behavior, no useful information.
◆ Low entropy = significant information.
  • At entropy = 0, we know exactly.
◆ Ideally, we find an attribute such that most of the “good’s” are on one side, and most of the “bad’s” are on the other.

Example

◆ Our Married, Home, Dog, Rating data:
  • 010G, 001G, 111G, 100G, 100B, 000B, 101B, 110B.
  • Married: 1/4 of Y is G; 1/4 of N is B.
  • Entropy = ((1/4) log₂ 4 + (3/4) log₂ (4/3)) = .81 on both sides.

Example, Continued

◆ 010G, 001G, 011G, 100G, 100B, 000B, 101B, 110B.
◆ Home: 1/3 of Y is B; 2/5 of N is G.
  • Entropy is (1/3) log₂ 3 + (2/3) log₂ (3/2) = .92 on Y side.
  • Entropy is (2/5) log₂ (5/2) + (3/5) log₂ (5/3) = .98 on N side.
  • Max = .98, greater than for Married.
◆ Dog is similar, so Married “wins.”

The “Training” Process

Handling Numeric Data

◆ While complicated tests at a node are permissible, e.g., “age = 30 or age ≤ 50 and age ≥ 42,” the simplest thing is to pick one breakpoint, and divide records by value < breakpoint and value > breakpoint.
◆ Rate an attribute and breakpoint by min-max entropy of the two sides.

Overfitting

◆ A major problem in designing decision trees is that one tends to create too many levels.
  • The number of records reaching a node is small, so significance is lost.
  • Extreme example: our data was generated by coin-flips; the tree is unlikely to reflect additional data that it would be used to classify.
Possible Solutions

1. Limit depth of tree so that each decision is based on a sufficiently large pool of training data.
2. Create several trees independently (needs randomness in choice of attribute).
   • Decision based on vote of D.T.’s.
   • Filters out irrelevant factors.