Data Mining: Practical Machine Learning Tools and Techniques

Slides for Chapter 4 of Data Mining by I. H. Witten and E. Frank

Decision Trees

- Strategy: top down
  - Recursive divide-and-conquer fashion
  - First: select attribute for root node
    Create branch for each possible attribute value
  - Then: split instances into subsets
    One for each branch extending from the node
  - Finally: repeat recursively for each branch, using only instances that reach the branch
- Stop if all instances have the same class

Which attribute to select?
Criterion for attribute selection

- Which is the best attribute?
  - Want to get the smallest tree
  - Heuristic: choose the attribute that produces the “purest” nodes
- Popular impurity criterion: information gain
  - Information gain increases with the average purity of the subsets
- Strategy: choose attribute that gives greatest information gain

Example: attribute *Outlook*

- **Outlook = Sunny:**
  \[ \text{info}[2,3]=\text{entropy}(2/5,3/5)=-2/5\log(2/5)-3/5\log(3/5)=0.971 \text{ bits} \]
- **Outlook = Overcast:**
  \[ \text{info}[4,0]=\text{entropy}(1,0)=-1\log(1)-0\log(0)=0 \text{ bits} \]
  \[ \text{Note: this is normally undefined.} \]
- **Outlook = Rainy:**
  \[ \text{info}[2,3]=\text{entropy}(3/5,2/5)=-3/5\log(3/5)-2/5\log(2/5)=0.971 \text{ bits} \]
- Expected information for attribute:
  \[ \text{info}[3,2],[4,0],[3,2]=(5/14)\times0.971+(4/14)\times0+(5/14)\times0.971=0.693 \text{ bits} \]

Computing information

- Measure information in *bits*
  - Given a probability distribution, the info required to predict an event is the distribution’s *entropy*
  - Entropy gives the information required in bits (can involve fractions of bits!)
- Formula for computing the entropy:
  \[ \text{entropy}(p_1,p_2,...,p_n)=-p_1\log p_1-p_2\log p_2...-p_n\log p_n \]

Computing information gain

- Information gain: information before splitting – information after splitting
  \[ \text{gain(Outlook)} = \text{info}[9,5] - \text{info}[2,3],[4,0],[3,2] \]
  \[ = 0.940 - 0.693 \]
  \[ = 0.247 \text{ bits} \]
- Information gain for attributes from weather data:
  \[ \text{gain(Outlook)} = 0.247 \text{ bits} \]
  \[ \text{gain(Temperature)} = 0.029 \text{ bits} \]
  \[ \text{gain(Humidity)} = 0.152 \text{ bits} \]
  \[ \text{gain(Windy)} = 0.048 \text{ bits} \]
Continuing to split

- gain($Temperature$) = 0.571 bits
- gain($Humidity$) = 0.971 bits
- gain($Windy$) = 0.020 bits

- Note: not all leaves need to be pure; sometimes identical instances have different classes
  ⇒ Splitting stops when data can't be split any further

Wishlist for a purity measure

- Properties we require from a purity measure:
  - When node is pure, measure should be zero
  - When impurity is maximal (i.e. all classes equally likely), measure should be maximal
  - Measure should obey multistage property (i.e. decisions can be made in several stages):
    \[
    \text{measure}(\{2,3,4\}) = \text{measure}(\{2,7\}) + \frac{7}{9} \times \text{measure}(\{3,4\})
    \]
  - Entropy is the only function that satisfies all three properties!

Properties of the entropy

- The multistage property:
  \[
  \text{entropy}(p, q, r) = \text{entropy}(p, q+r) + (q+r) \times \text{entropy}(\frac{q}{q+r}, \frac{r}{q+r})
  \]
- Simplification of computation:
  \[
  \text{info}(\{2,3,4\}) = -\frac{2}{9} \times \log(\frac{2}{9}) - \frac{3}{9} \times \log(\frac{3}{9}) - \frac{4}{9} \times \log(\frac{4}{9})
  = -2 \times \log 2 - 3 \times \log 3 - 4 \times \log 4 + 9 \times \log 9 / 9
  \]
- Note: instead of maximizing info gain we could just minimize information
Highly-branching attributes

- Problematic: attributes with a large number of values (extreme case: ID code)
- Subsets are more likely to be pure if there is a large number of values
  $\Rightarrow$ Information gain is biased towards choosing attributes with a large number of values
  $\Rightarrow$ This may result in overfitting (selection of an attribute that is non-optimal for prediction)
- Another problem: fragmentation

<table>
<thead>
<tr>
<th>ID code</th>
<th>Outlook</th>
<th>Temp.</th>
<th>Humidity</th>
<th>Windy</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>False</td>
<td>No</td>
</tr>
<tr>
<td>B</td>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>False</td>
<td>No</td>
</tr>
<tr>
<td>C</td>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>D</td>
<td>Rainy</td>
<td>Mild</td>
<td>High</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>E</td>
<td>Rainy</td>
<td>Cool</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>F</td>
<td>Rainy</td>
<td>Cool</td>
<td>Normal</td>
<td>True</td>
<td>No</td>
</tr>
<tr>
<td>G</td>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>H</td>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>False</td>
<td>No</td>
</tr>
<tr>
<td>I</td>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>J</td>
<td>Rainy</td>
<td>Mild</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>K</td>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>L</td>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>True</td>
<td>Yes</td>
</tr>
<tr>
<td>M</td>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>N</td>
<td>Rainy</td>
<td>Mild</td>
<td>High</td>
<td>True</td>
<td>No</td>
</tr>
</tbody>
</table>

Tree stump for **ID code** attribute

- Entropy of split:
  $\text{info}(\text{ID code})=\text{info}([0,1]) + \text{info}([0,1]) + \ldots + \text{info}([0,1]) = 0 \text{ bits}$
  $\Rightarrow$ Information gain is maximal for ID code (namely 0.940 bits)

Gain ratio

- **Gain ratio**: a modification of the information gain that reduces its bias
- Gain ratio takes number and size of branches into account when choosing an attribute
  - It corrects the information gain by taking the intrinsic information of a split into account
- Intrinsic information: entropy of distribution of instances into branches (i.e. how much info do we need to tell which branch an instance belongs to)
Computing the gain ratio

- Example: intrinsic information for ID code
  \[\text{info}([1,1,...,1])=14\times(-1/14\times\log(1/14))=3.807\text{bits}\]
- Value of attribute decreases as intrinsic information gets larger
- Definition of gain ratio:
  \[\text{gain ratio}(\text{attribute}) = \frac{\text{gain}(\text{attribute})}{\text{intrinsic info}(\text{attribute})}\]
- Example:
  \[\text{gain ratio}(\text{ID code}) = \frac{0.940}{3.807} = 0.246\]

Gain ratios for weather data

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
</tr>
</thead>
<tbody>
<tr>
<td>Info:</td>
<td>0.693</td>
</tr>
<tr>
<td>Gain:</td>
<td>0.940-0.693</td>
</tr>
<tr>
<td>Split info: info([5,4,5])</td>
<td>1.577</td>
</tr>
<tr>
<td>Gain ratio: 0.247/1.577</td>
<td>0.157</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Humidity</th>
<th>Windy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Info:</td>
<td>0.788</td>
</tr>
<tr>
<td>Gain:</td>
<td>0.940-0.788</td>
</tr>
<tr>
<td>Split info: info([7,7])</td>
<td>1.000</td>
</tr>
<tr>
<td>Gain ratio: 0.152/1</td>
<td>0.152</td>
</tr>
</tbody>
</table>

More on the gain ratio

- “Outlook” still comes out top
- However: “ID code” has greater gain ratio
  - Standard fix: ad hoc test to prevent splitting on that type of attribute
- Problem with gain ratio: it may overcompensate
  - May choose an attribute just because its intrinsic information is very low
  - Standard fix: only consider attributes with greater than average information gain

Discussion

- Top-down induction of decision trees: ID3, algorithm developed by Ross Quinlan
  - Gain ratio just one modification of this basic algorithm
  - \(\Rightarrow\) C4.5: deals with numeric attributes, missing values, noisy data
- Similar approach: CART
- There are many other attribute selection criteria!
  (But little difference in accuracy of result)
Covering algorithms

- Convert decision tree into a rule set
  - Straightforward, but rule set overly complex
  - More effective conversions are not trivial
- Instead, can generate rule set directly
  - for each class in turn find rule set that covers all instances in it (excluding instances not in the class)
- Called a covering approach:
  - at each stage a rule is identified that “covers” some of the instances

Example: generating a rule

If $x > 1.2$ then class = a
If $x > 1.2$ and $y > 2.6$ then class = a
If $x > 1.2$ then class = a

- Possible rule set for class “b”:
  - If $x \leq 1.2$ then class = b
  - If $x > 1.2$ and $y \leq 2.6$ then class = b
- Could add more rules, get “perfect” rule set

Rules vs. trees

Corresponding decision tree:
(produces exactly the same predictions)

- But: rule sets can be more perspicuous when decision trees suffer from replicated subtrees
- Also: in multiclass situations, covering algorithm concentrates on one class at a time whereas decision tree learner takes all classes into account

Simple covering algorithm

- Generates a rule by adding tests that maximize rule’s accuracy
- Similar to situation in decision trees: problem of selecting an attribute to split on
  - But: decision tree inducer maximizes overall purity
- Each new test reduces rule’s coverage:
Selecting a test

• Goal: maximize accuracy
  • \( t \) total number of instances covered by rule
  • \( p \) positive examples of the class covered by rule
  ⇒ Select test that maximizes the ratio \( p/t \)

• We are finished when \( p/t = 1 \) or the set of instances can’t be split any further

Example: contact lens data

• Rule we seek:
  \[
  \text{If } \begin{align*}
  \text{age} = \text{Young} \\
  \text{age} = \text{Pre-presbyopic} \\
  \text{age} = \text{Presbyopic} \\
  \text{spectacle prescription} = \text{Myope} \\
  \text{spectacle prescription} = \text{Hypermetrope} \\
  \text{astigmatism} = \text{yes} \\
  \text{tear production rate} = \text{Reduced} \\
  \text{tear production rate} = \text{Normal}
  \end{align*}
  \text{then recommendation} = \text{hard}
  \]

• Possible tests:

<table>
<thead>
<tr>
<th>Age</th>
<th>Spectacle prescription</th>
<th>Astigmatism</th>
<th>Tear production rate</th>
<th>Recommended lenses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Young</td>
<td>Myope</td>
<td>Yes</td>
<td>Reduced</td>
<td>None</td>
</tr>
<tr>
<td>Young</td>
<td>Myope</td>
<td>Yes</td>
<td>Normal</td>
<td>Hard</td>
</tr>
<tr>
<td>Young</td>
<td>Hypermetrope</td>
<td>Yes</td>
<td>Reduced</td>
<td>None</td>
</tr>
<tr>
<td>Young</td>
<td>Hypermetrope</td>
<td>Yes</td>
<td>Normal</td>
<td>None</td>
</tr>
<tr>
<td>Pre-presbyopic</td>
<td>Myope</td>
<td>Yes</td>
<td>Reduced</td>
<td>None</td>
</tr>
<tr>
<td>Pre-presbyopic</td>
<td>Myope</td>
<td>Yes</td>
<td>Normal</td>
<td>Hard</td>
</tr>
<tr>
<td>Pre-presbyopic</td>
<td>Hypermetrope</td>
<td>Yes</td>
<td>Reduced</td>
<td>None</td>
</tr>
<tr>
<td>Presbyopic</td>
<td>Myope</td>
<td>Yes</td>
<td>Reduced</td>
<td>None</td>
</tr>
<tr>
<td>Presbyopic</td>
<td>Myope</td>
<td>Yes</td>
<td>Normal</td>
<td>Hard</td>
</tr>
<tr>
<td>Presbyopic</td>
<td>Hypermetrope</td>
<td>Yes</td>
<td>Reduced</td>
<td>None</td>
</tr>
<tr>
<td>Presbyopic</td>
<td>Hypermetrope</td>
<td>Yes</td>
<td>Reduced</td>
<td>None</td>
</tr>
</tbody>
</table>

Further refinement

• Current state:
  \[
  \text{If astigmatism} = \text{yes} \\
  \text{and } \begin{align*}
  \text{age} = \text{Pre-presbyopic} \\
  \text{age} = \text{Presbyopic} \\
  \text{spectacle prescription} = \text{Myope} \\
  \text{spectacle prescription} = \text{Hypermetrope} \\
  \text{tear production rate} = \text{Reduced} \\
  \text{tear production rate} = \text{Normal}
  \end{align*}
  \text{then recommendation} = \text{hard}
  \]

• Possible tests:

<table>
<thead>
<tr>
<th>Age</th>
<th>Spectacle prescription</th>
<th>Astigmatism</th>
<th>Tear production rate</th>
<th>Recommended lenses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Young</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Young</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Young</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Young</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pre-presbyopic</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pre-presbyopic</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Presbyopic</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Presbyopic</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**Modified rule and resulting data**

- Rule with best test added:
  
  \[
  \text{If } \text{astigmatism} = \text{yes} \\
  \text{and tear production rate} = \text{normal} \\
  \text{then recommendation} = \text{hard}
  \]

- Instances covered by modified rule:

<table>
<thead>
<tr>
<th>Age</th>
<th>Spectacle prescription</th>
<th>Astigmatism</th>
<th>Tear production rate</th>
<th>Recommended lenses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Young</td>
<td>Myope</td>
<td>Yes</td>
<td>Normal</td>
<td>Hard</td>
</tr>
<tr>
<td>Young</td>
<td>Hypermetrope</td>
<td>Yes</td>
<td>Normal</td>
<td>Hard</td>
</tr>
<tr>
<td>Pre-presbyopic</td>
<td>Myope</td>
<td>Yes</td>
<td>Normal</td>
<td>Hard</td>
</tr>
<tr>
<td>Pre-presbyopic</td>
<td>Hypermetrope</td>
<td>Yes</td>
<td>Normal</td>
<td>None</td>
</tr>
<tr>
<td>Presbyopic</td>
<td>Myope</td>
<td>Yes</td>
<td>Normal</td>
<td>Hard</td>
</tr>
<tr>
<td>Presbyopic</td>
<td>Hypermetrope</td>
<td>Yes</td>
<td>Normal</td>
<td>None</td>
</tr>
</tbody>
</table>

**Further refinement**

- Current state:
  
  \[
  \text{If } \text{astigmatism} = \text{yes} \\
  \text{and tear production rate} = \text{normal} \\
  \text{and } \text{spectacle prescription}\? \\
  \text{then recommendation} = \text{hard}
  \]

- Possible tests:
  
  - Age = Young
  - Age = Pre-presbyopic
  - Age = Presbyopic
  - Spectacle prescription = Myope
  - Spectacle prescription = Hypermetrope

- Tie between the first and the fourth test
  
  - We choose the one with greater coverage

**The result**

- Final rule:
  
  \[
  \text{If } \text{astigmatism} = \text{yes} \\
  \text{and tear production rate} = \text{normal} \\
  \text{and spectacle prescription} = \text{myope} \\
  \text{then recommendation} = \text{hard}
  \]

- Second rule for recommending “hard lenses”:
  
  (built from instances not covered by first rule)

  \[
  \text{If } \text{age} = \text{young} \text{ and astigmatism} = \text{yes} \\
  \text{and tear production rate} = \text{normal} \\
  \text{then recommendation} = \text{hard}
  \]

- These two rules cover all “hard lenses”:
  
  - Process is repeated with other two classes

**Pseudo-code for PRISM**

```
For each class C
  Initialize E to the instance set
  While E contains instances in class C
    Create a rule R with an empty left-hand side that predicts class C
    Until R is perfect (or there are no more attributes to use) do
      For each attribute A not mentioned in R, and each value v,
        Consider adding the condition A = v to the left-hand side of R
        Select A and v to maximize the accuracy p/t
          (break ties by choosing the condition with the largest p)
        Add A = v to R
        Remove the instances covered by R from E
```
Rules vs. decision lists

- PRISM with outer loop removed generates a decision list for one class
  - Subsequent rules are designed for rules that are not covered by previous rules
  - But: order doesn’t matter because all rules predict the same class
- Outer loop considers all classes separately
  - No order dependence implied
- Problems: overlapping rules, default rule required

Separate and conquer

- Methods like PRISM (for dealing with one class) are separate-and-conquer algorithms:
  - First, identify a useful rule
  - Then, separate out all the instances it covers
  - Finally, “conquer” the remaining instances
- Difference to divide-and-conquer methods:
  - Subset covered by rule doesn’t need to be explored any further