Slides for “Data Mining”
by
I. H. Witten and E. Frank
Simplicity first

- Simple algorithms often work very well!
- There are many kinds of simple structure, eg:
  - One attribute does all the work
  - All attributes contribute equally & independently
  - A weighted linear combination might do
  - Instance-based: use a few prototypes
  - Use simple logical rules
- Success of method depends on the domain
Inferring rudimentary rules

- 1R: learns a 1-level decision tree
  - I.e., rules that all test one particular attribute

- Basic version
  - One branch for each value
  - Each branch assigns most frequent class
  - Error rate: proportion of instances that don’t belong to the majority class of their corresponding branch
  - Choose attribute with lowest error rate

*(assumes nominal attributes)*
Pseudo-code for 1R

For each attribute,
  For each value of the attribute, make a rule as follows:
    count how often each class appears
    find the most frequent class
    make the rule assign that class to this attribute-value
  Calculate the error rate of the rules
Choose the rules with the smallest error rate

Note: “missing” is treated as a separate attribute value
# Evaluating the weather attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Rules</th>
<th>Errors</th>
<th>Total errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outlook</td>
<td>Sunny → No</td>
<td>2/5</td>
<td>4/14</td>
</tr>
<tr>
<td>Overcast</td>
<td>Overcast → Yes</td>
<td>0/4</td>
<td></td>
</tr>
<tr>
<td>Rainy</td>
<td>Rainy → Yes</td>
<td>2/5</td>
<td></td>
</tr>
<tr>
<td>Temp</td>
<td>Hot → No*</td>
<td>2/4</td>
<td>5/14</td>
</tr>
<tr>
<td>Humidity</td>
<td>High → No</td>
<td>3/7</td>
<td>4/14</td>
</tr>
<tr>
<td>Windy</td>
<td>False → Yes</td>
<td>2/8</td>
<td>5/14</td>
</tr>
<tr>
<td></td>
<td>True → No*</td>
<td>3/6</td>
<td></td>
</tr>
</tbody>
</table>

* indicates a tie
Dealing with numeric attributes

- Discretize numeric attributes
- Divide each attribute’s range into intervals
  - Sort instances according to attribute’s values
  - Place breakpoints where the class changes (the majority class)
- This minimizes the total error
- Example:

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Windy</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>85</td>
<td>85</td>
<td>False</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>80</td>
<td>90</td>
<td>True</td>
<td>No</td>
</tr>
<tr>
<td>Overcast</td>
<td>83</td>
<td>86</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>Rainy</td>
<td>75</td>
<td>80</td>
<td>False</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Humidity</th>
<th>Temperature</th>
<th>Outlook</th>
<th>Windy</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>75</td>
<td>Sunny</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>85</td>
<td>75</td>
<td>Sunny</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>80</td>
<td>75</td>
<td>Rainy</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>80</td>
<td>75</td>
<td>Rainy</td>
<td>False</td>
<td>Yes</td>
</tr>
<tr>
<td>80</td>
<td>75</td>
<td>Rainy</td>
<td>False</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Temperature</th>
<th>Humidity</th>
<th>Outlook</th>
<th>Windy</th>
<th>Play</th>
</tr>
</thead>
<tbody>
<tr>
<td>85</td>
<td>85</td>
<td>Sunny</td>
<td>False</td>
<td>No</td>
</tr>
<tr>
<td>85</td>
<td>85</td>
<td>Sunny</td>
<td>False</td>
<td>No</td>
</tr>
<tr>
<td>86</td>
<td>False</td>
<td>Overcast</td>
<td>True</td>
<td>No</td>
</tr>
<tr>
<td>80</td>
<td>False</td>
<td>Rainy</td>
<td>True</td>
<td>No</td>
</tr>
<tr>
<td>80</td>
<td>False</td>
<td>Rainy</td>
<td>True</td>
<td>No</td>
</tr>
<tr>
<td>80</td>
<td>False</td>
<td>Rainy</td>
<td>True</td>
<td>No</td>
</tr>
<tr>
<td>80</td>
<td>False</td>
<td>Rainy</td>
<td>True</td>
<td>No</td>
</tr>
<tr>
<td>80</td>
<td>False</td>
<td>Rainy</td>
<td>True</td>
<td>No</td>
</tr>
</tbody>
</table>

64 65 68 69 70 71 72 72 75 75 80 81 83 85
Yes  No  Yes Yes Yes No No Yes Yes Yes No Yes Yes No

7
The problem of overfitting

- This procedure is very sensitive to noise
  - One instance with an incorrect class label will probably produce a separate interval
- Also: *time stamp* attribute will have zero errors
- Simple solution: *enforce minimum number of instances in majority class per interval*
- Example (with min = 3):

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|64|65|68|69|70|71|72|72|75|75|80|81|83|85|
|Yes |No |Yes|Yes|Yes | |No|No|Yes |Yes |Yes | |No |Yes|Yes |No |

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|64|65|68|69|70|71|72|72|75|75|80|81|83|85|
|Yes |No |Yes|Yes|Yes | |No|No|Yes |Yes |Yes | |No |Yes|Yes |No |
With overfitting avoidance

Resulting rule set:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Rules</th>
<th>Errors</th>
<th>Total errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outlook</td>
<td>Sunny → No</td>
<td>2/5</td>
<td>4/14</td>
</tr>
<tr>
<td></td>
<td>Overcast → Yes</td>
<td>0/4</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Rainy → Yes</td>
<td>2/5</td>
<td></td>
</tr>
<tr>
<td>Temperature</td>
<td>≤ 77.5 → Yes</td>
<td>3/10</td>
<td>5/14</td>
</tr>
<tr>
<td></td>
<td>&gt; 77.5 → No*</td>
<td>2/4</td>
<td></td>
</tr>
<tr>
<td>Humidity</td>
<td>≤ 82.5 → Yes</td>
<td>1/7</td>
<td>3/14</td>
</tr>
<tr>
<td></td>
<td>&gt; 82.5 and ≤ 95.5 → No</td>
<td>2/6</td>
<td></td>
</tr>
<tr>
<td></td>
<td>&gt; 95.5 → Yes</td>
<td>0/1</td>
<td></td>
</tr>
<tr>
<td>Windy</td>
<td>False → Yes</td>
<td>2/8</td>
<td>5/14</td>
</tr>
<tr>
<td></td>
<td>True → No*</td>
<td>3/6</td>
<td></td>
</tr>
</tbody>
</table>
Discussion of 1R

- 1R was described in a paper by Holte (1993)
  - Contains an experimental evaluation on 16 datasets (using cross-validation so that results were representative of performance on future data)
  - Minimum number of instances was set to 6 after some experimentation
  - 1R’s simple rules performed not much worse than much more complex decision trees

- Simplicity first pays off!

Very Simple Classification Rules Perform Well on Most Commonly Used Datasets

Robert C. Holte, Computer Science Department, University of Ottawa
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 true
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
  - $t - p$ number of errors made by rule
  - $\Rightarrow$ 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:
  If ?
  then recommendation = hard

- Possible tests:

  Age = Young 2/8
  Age = Pre-presbyopic 1/8
  Age = Presbyopic 1/8
  Spectacle prescription = Myope 3/12
  Spectacle prescription = Hypermetrope 1/12
  Astigmatism = no 0/12
  Astigmatism = yes 4/12
  Tear production rate = Reduced 0/12
  Tear production rate = Normal 4/12
Modified rule and resulting data

Rule with best test added:

If astigmatism = yes
then recommendation = 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>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>hard</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>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>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>Normal</td>
<td>None</td>
</tr>
</tbody>
</table>
Further refinement

- **Current state:**

  ```
  If astigmatism = yes
  and ?
  then recommendation = hard
  ```

- **Possible tests:**

  - Age = Young 2/4
  - Age = Pre-presbyopic 1/4
  - Age = Presbyopic 1/4
  - Spectacle prescription = Myope 3/6
  - Spectacle prescription = Hypermetrope 1/6
  - Tear production rate = Reduced 0/6
  - Tear production rate = Normal 4/6
Modified rule and resulting data

- Rule with best test added:

  If astigmatism = yes
  and tear production rate = normal
  then recommendation = 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:**

  If astigmatism = yes
  and tear production rate = normal
  and ?
  then recommendation = hard

- **Possible tests:**

  Age = Young 2/2
  Age = Pre-presbyopic 1/2
  Age = Presbyopic 1/2
  Spectacle prescription = Myope 3/3
  Spectacle prescription = Hypermetrope 1/3

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

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

- **Second rule for recommending “hard lenses”:**
  (built from instances not covered by first rule)
  
  \[
  \text{If age = young and astigmatism = yes} \\
  \text{and tear production rate = normal} \\
  \text{then recommendation = hard}
  \]

- **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