Searching for the Optimal Ordering of Classes in Rule Induction

Sezin Ata    Olcay Taner Yıldız

Department of Computer Engineering, Işık University, TR-34980, Istanbul, Turkey

ICPR 2012
Sezin Ata, Olcay Taner Yıldız

Searching for the Optimal Ordering of Classes in Rule Induction
Rule Induction

- A rule set is typically an ordered list of rules.
- A rule contains a conjunction of terms and a class code.
- The terms are of the form $x_i = v$, $x_i < \theta$ or $x_i \geq \theta$.
- Example rule (Iris dataset)
  
  **If** $F_3 < 1.9$ **and** $F_4 \geq 5.1$ **Then** iris-setosa
  **Else**
    **If** $F_3 < 4.7$ **Then** iris-versicolor
    **Else** iris-virginica
Motivation

a) Current Approach's Rule Set
   If \((x_1 < 18) && (x_2 < 6)\)
      Then Class = 1
   If \((x_1 > 9) && (x_2 < 8)\)
      Then Class = 1
   If \((x_1 < 9)\)
      Then Class = 2
   If \((x_1 > 18)\)
      Then Class = 2
   Else
      Class = 3

b) Another Ordering's Rule Set
   If \((x_1 < 9) && (x_2 > 6)\)
      Then Class = 2
   If \((x_1 > 18)\)
      Then Class = 2
   If \((x_2 < 8)\)
      Then Class = 1
   Else
      Class = 3
\begin{verbatim}
1    RS = {}
2    \textit{C}_i \text{ ordered in increasing prior probability}
3    \textbf{for} p = 1 \textbf{to} K - 1
4        Pos = C_p, Neg = C_{p+1}, \ldots, C_K
5        RS_p = {}
6    \textbf{while} D \text{ contains positive samples}
7        Divide D into Grow set G and Prune set P
8        r = GrowRule(G)
9        PruneRule(r, P)
10       \textbf{if} \text{ CalculateError}(r) > 0.5
11           \textbf{break}
12       \textbf{else}
13           RS_p = RS_p + r
14           Remove examples covered by r from D
15       \textbf{for} i = 1 \textbf{to} 2
16           OptimizeRuleset(RS_p, D)
17           SimplifyRuleset(RS_p, D)
18    RS = RS + RS_p
19    \textbf{return} RS
\end{verbatim}
Forward Ordering Search: Algorithm

- Views optimizing the ordering of classes in Ripper as a search in the state space of all possible orderings.
- Forward search algorithm starts from an initial state (heuristic ordering) and use exchange operators to generate candidate states (orderings).
- Let say we have the ordering $C_1 C_2 C_3 \ldots C_{K-1} C_K$. The exchange operator creates the following $K - 1$ candidate orderings: $C_2 C_1 C_3 \ldots C_{K-1} C_K$, $C_1 C_3 C_2 \ldots C_{K-1} C_K$, $C_1 C_2 C_4 \ldots C_{K-1} C_K$, $\ldots$, $C_1 C_2 C_3 \ldots C_K C_{K-1}$.
- Candidates are evaluated via $10 \times 10$-fold cross-validation.
- We stop the search when no candidate improves on the current best.
Forward Ordering Search: Example

Sezin Ata, Olcay Taner Yıldız

Searching for the Optimal Ordering of Classes in Rule Induction
The expected error of the Ripper trained with the ordering $\pi_i$, is the sum of $K(K - 1)/2$ pairwise expected errors of classes:

$$\hat{E}_{\pi_i} = \sum_{j=1}^{K} \sum_{k=1, [j] < \pi_i[k]}^{K} e_{jk}$$
Pairwise Error Approximation: Assumption

\[ E_{123} = e_{12} + e_{13} + e_{23} \]
Pairwise Error Approximation: Algorithm

- Run Ripper algorithm $N$ times with $N$ random orderings $\pi_i$ and get the test errors $E_{\pi_i}$.
- Minimize the total estimation error

$$E_t = \sum_{i=1}^{N} (E_{\pi_i} - \hat{E}_{\pi_i})^2$$

- Solve the following system of linear equations

$$\forall j, k \frac{\partial E_t}{\partial e_{jk}} = 0$$

- Solved $e_{jk}$’s, search all possible class orderings to get the optimal ordering.
**Datasets**

<table>
<thead>
<tr>
<th>Dataset</th>
<th># of attributes</th>
<th># of classes</th>
<th>Sample size</th>
</tr>
</thead>
<tbody>
<tr>
<td>led7</td>
<td>7</td>
<td>7</td>
<td>3200</td>
</tr>
<tr>
<td>ocr</td>
<td>256</td>
<td>10</td>
<td>600</td>
</tr>
<tr>
<td>optdigits</td>
<td>64</td>
<td>10</td>
<td>3823</td>
</tr>
<tr>
<td>pendigits</td>
<td>16</td>
<td>10</td>
<td>7494</td>
</tr>
<tr>
<td>segment</td>
<td>19</td>
<td>7</td>
<td>2310</td>
</tr>
<tr>
<td>shuttle</td>
<td>9</td>
<td>7</td>
<td>58000</td>
</tr>
<tr>
<td>winequality</td>
<td>11</td>
<td>7</td>
<td>6497</td>
</tr>
<tr>
<td>yeast</td>
<td>8</td>
<td>10</td>
<td>1484</td>
</tr>
</tbody>
</table>
## Results: Error rates

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Ripper</th>
<th>FOS</th>
<th>PEA</th>
</tr>
</thead>
<tbody>
<tr>
<td>led7</td>
<td>31.83 ± 0.23</td>
<td>29.67 ± 0.23</td>
<td>28.93 ± 0.21</td>
</tr>
<tr>
<td>ocr</td>
<td>26.61 ± 0.58</td>
<td>24.73 ± 0.59</td>
<td>22.08 ± 0.54</td>
</tr>
<tr>
<td>opt</td>
<td>10.96 ± 0.14</td>
<td>10.55 ± 0.15</td>
<td>8.57 ± 0.12</td>
</tr>
<tr>
<td>pen</td>
<td>5.32 ± 0.07</td>
<td>4.87 ± 0.08</td>
<td>4.49 ± 0.08</td>
</tr>
<tr>
<td>seg</td>
<td>6.54 ± 0.17</td>
<td>4.38 ± 0.12</td>
<td>5.03 ± 0.14</td>
</tr>
<tr>
<td>shu</td>
<td>0.04 ± 0.00</td>
<td>0.03 ± 0.00</td>
<td>0.01 ± 0.00</td>
</tr>
<tr>
<td>wine</td>
<td>46.32 ± 0.16</td>
<td>46.23 ± 0.13</td>
<td>56.64 ± 0.23</td>
</tr>
<tr>
<td>yea</td>
<td>43.09 ± 0.35</td>
<td>42.39 ± 0.40</td>
<td>43.38 ± 0.36</td>
</tr>
</tbody>
</table>
# Results: Complexity of FOS

<table>
<thead>
<tr>
<th>Dataset</th>
<th># of Orderings</th>
</tr>
</thead>
<tbody>
<tr>
<td>led7</td>
<td>40</td>
</tr>
<tr>
<td>ocr</td>
<td>18</td>
</tr>
<tr>
<td>opt</td>
<td>25</td>
</tr>
<tr>
<td>pen</td>
<td>41</td>
</tr>
<tr>
<td>seg</td>
<td>22</td>
</tr>
<tr>
<td>shu</td>
<td>12</td>
</tr>
<tr>
<td>wine</td>
<td>17</td>
</tr>
<tr>
<td>yea</td>
<td>25</td>
</tr>
</tbody>
</table>
## Results: Average Estimation Error of PEA

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$E_t$</th>
</tr>
</thead>
<tbody>
<tr>
<td>led7</td>
<td>0.31</td>
</tr>
<tr>
<td>ocr</td>
<td>0.63</td>
</tr>
<tr>
<td>opt</td>
<td>0.20</td>
</tr>
<tr>
<td>pen</td>
<td>0.08</td>
</tr>
<tr>
<td>seg</td>
<td>0.24</td>
</tr>
<tr>
<td>shu</td>
<td>1.86</td>
</tr>
<tr>
<td>wine</td>
<td>3.41</td>
</tr>
<tr>
<td>yea</td>
<td>2.67</td>
</tr>
</tbody>
</table>
Summary

- Current heuristic approach that orders the classes in a dataset according to their sample sizes, usually does not give the most accurate classification.
- FOS starts with the ordering the heuristic provides and searches for better orderings by swapping consecutive classes.
- PEA transforms the ordering search problem into an optimization problem and uses the solution of the optimization problem to extract the optimal ordering.