Learning MDL-guided Decision Trees for Constructor-Based Languages

C. Ferri-Ramírez, J. Hernández-Orallo & M.J. Ramírez-Quintana

DSIC, Universitat Politècnica de València
Camí de Vera s/n, 46022 València, Spain.
Email: {cferri,jorallo,mramirez}@dsic.upv.es

11th International Conference on Inductive Logic Programming (ILP’2001)
Strasbourg, France, September 9th-11th

1 Work partially supported by CICYT under grant TIC 98-0445-C03-C1 and
Generalitat Valenciana under grant GV00-092-14.
Extending Decision Tree Learning

• Decision Tree Learning: methods such as CARS, ID3, C4.5/C5.0 and FOIL are amongst the most popular symbolic learning methods.
  
  o Induction is usually made in two phases:
    • building phase
    • post-pruning phase

• FOIL, TILDE and derivatives represent an extension to include relational patterns and even recursion.
  o However, constructor data-types must be flattened.

Learning from semi-structured data either requires ad-hoc methods or requires important re-processing for general methods (e.g. ILP), which converts data into an unnatural condition.
Constructor-Based Decision Trees

Defined over *Functional Logic Programs*:

- Facts are represented as equalities, where constructors can appear in any argument or even in the class:

\[
E^+ = \left\{ 
\begin{aligned}
e_1 : \text{member}(a, \lambda) &= \text{false} \\
e_2 : \text{member}(b, \text{ins}(\lambda, a)) &= \text{false} \\
e_3 : \text{member}(c, \lambda) &= \text{false} \\
e_4 : \text{member}(c, \text{ins}(\lambda, b)) &= \text{false} \\
e_5 : \text{member}(a, \text{ins}(\text{ins}(\lambda, b), d)) &= \text{false} \\
e_6 : \text{member}(a, \text{ins}(\text{ins}(\lambda, b), a)) &= \text{true} \\
e_7 : \text{member}(b, \text{ins}(\text{ins}(\lambda, b), a)) &= \text{true} \\
e_8 : \text{member}(c, \text{ins}(\text{ins}(\text{ins}(\lambda, b), a), c)) &= \text{true} \\
e_9 : \text{member}(a, \text{ins}(\text{ins}(\text{ins}(\lambda, b), a), b)) &= \text{true} \\
e_{10} : \text{member}(c, \text{ins}(\lambda, c)) &= \text{true}
\end{aligned}
\right\}
\]

- Hypotheses are represented as conditional functional logic rules:

(i) \( \text{member}(X, \lambda) = \text{false} \)
(ii) \( \text{member}(X, \text{ins}(Z, X)) = \text{true} \)
(iii) \( \text{member}(X, \text{ins}(L, W)) = \text{member}(X, L) \Leftrightarrow W \neq X \)
Constructor-Based Decision Trees

A functional logic program can be represented as a functional-logic tree:

- The root of the tree is a fully uninstantiated rule.
- Branches add instantiations (substitutions) to these variables.
- Recursive calls and background knowledge can appear as arguments or as the function result.

Selection criteria based on discrimination (GINI, Gain, Gain Ratio) are not applicable.
Descriptive MDL

Derived from Maximum A Posteriori (MAP) hypothesis and descriptional complexity ($K(\cdot)$).

$$h_{MAP} = \arg\max_{h \in H} P(h \mid E) = \arg\min_{h \in H} (K(h) + K(E \mid h))$$

- In predictive MDL: $K(E \mid h)$ just measures the information needed to code the function result.
- In descriptive MDL: $K(E \mid h)$ measures the information needed to code the arguments and the function result.

Several estimates are introduced for:
- $K(h)$: information needed to code a branch up to a node.
- $K(E \mid h)$: information needed to code the examples that fall under that branch, using the branch information.
Partitions

- Splits allowed:

<table>
<thead>
<tr>
<th>#</th>
<th>Partition on Attribute $X_i$ (Split)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$X_i = a_1 \mid X_i = a_2 \mid \ldots \mid X_i = a_k$</td>
</tr>
<tr>
<td>2</td>
<td>$X_i = c_0 \mid \ldots \mid X_i = c_k(Y_1, \ldots, Y_{km})$</td>
</tr>
<tr>
<td>3</td>
<td>$X_i &lt; t \mid X_i \geq t$ where $t$ is a threshold</td>
</tr>
<tr>
<td>4</td>
<td>$X_i = Y$ where $Y \in {X_1, \ldots, X_n}$ and $Y \neq X_i$</td>
</tr>
<tr>
<td>5</td>
<td>$X_i = a \mid X_i \neq a$</td>
</tr>
<tr>
<td>6</td>
<td>$a_1 = f(Y_1, \ldots, Y_n) \mid \ldots \mid a_n = f(Y_1, \ldots, Y_n)$</td>
</tr>
<tr>
<td></td>
<td>where $\exists! Y_i \in {X_1, \ldots, X_n}$</td>
</tr>
<tr>
<td>7</td>
<td>$X_i = f(Y_1, \ldots, Y_n)$</td>
</tr>
<tr>
<td>8</td>
<td>$a_1 = g(Y_1, \ldots, Y_n) \mid \ldots \mid a_n = g(Y_1, \ldots, Y_n)$</td>
</tr>
<tr>
<td></td>
<td>where $\exists! Y_i \in {X_1, \ldots, X_n}$ and $g \in \Sigma_B$</td>
</tr>
<tr>
<td>9</td>
<td>$X_i = g(Y_1, \ldots, Y_n)$ where $g \in \Sigma_B$</td>
</tr>
</tbody>
</table>

- Expressiveness comparison:

<table>
<thead>
<tr>
<th>#</th>
<th>ID3</th>
<th>FOIL</th>
<th>CRG</th>
<th>CDTL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>-</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>3</td>
<td>x</td>
<td>x</td>
<td>-</td>
<td>x</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>-</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>x</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>-</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>7</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>x</td>
</tr>
<tr>
<td>8</td>
<td>-</td>
<td>-</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>9</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>x</td>
</tr>
</tbody>
</table>
Constructing a Multitree

Once a solution is found (in a greedy way), the tree is further populated to find more solutions.

This constitutes a \textit{multitree}, more specifically an AND-OR tree.

Fig. 1: Complete AND/OR tree for the playtennis example
Selection Criteria

Several selection criteria are needed:

- **Node Selection Criterion**: from all the open nodes, the node with less description cost is selected first. This criterion is irrelevant.

- **Split Selection Criterion**: from all the possible partitions (splits), we select the split which minimises the cost of the split and the cost of describing the evidence under that split in one level.

- **Stopping/Pruning Criterion**: a node is closed when the class is consistent with all the examples falling under that node or the cost of coding the exceptions is less than following the branch.

- **Tree Selection Criterion** (multitree population): from all the unexplored splits, the one which is relatively costlier wrt. the best alternative one is selected (rival ratio).

- **Solution Selection Criterion**: from all the solutions in a multitree, the shortest one is selected (Occam’s razor).
Experiments (1/2)

Fig. 1: Rules and accuracy of CDTL for increasing number of solutions:

<table>
<thead>
<tr>
<th>Numtree</th>
<th>Rules</th>
<th>Accuracy</th>
<th>Rules</th>
<th>Accuracy</th>
<th>Rules</th>
<th>Accuracy</th>
<th>Rules</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>cars</td>
<td>140</td>
<td>86.57</td>
<td>119</td>
<td>86.69</td>
<td>119</td>
<td>86.69</td>
<td>110</td>
<td>87.50</td>
</tr>
<tr>
<td>house-votes</td>
<td>23</td>
<td>89.45</td>
<td>10</td>
<td>94.50</td>
<td>10</td>
<td>94.50</td>
<td>7</td>
<td>94.50</td>
</tr>
<tr>
<td>tic-tac-toe</td>
<td>111</td>
<td>75.99</td>
<td>101</td>
<td>71.59</td>
<td>94</td>
<td>75.78</td>
<td>73</td>
<td>77.87</td>
</tr>
<tr>
<td>nursery</td>
<td>517</td>
<td>93.00</td>
<td>440</td>
<td>94.86</td>
<td>440</td>
<td>94.86</td>
<td>345</td>
<td>93.87</td>
</tr>
<tr>
<td>monks1</td>
<td>9</td>
<td>100</td>
<td>5</td>
<td>100</td>
<td>5</td>
<td>100</td>
<td>5</td>
<td>100</td>
</tr>
<tr>
<td>monks2</td>
<td>44</td>
<td>62.50</td>
<td>38</td>
<td>64.35</td>
<td>28</td>
<td>63.19</td>
<td>26</td>
<td>62.04</td>
</tr>
<tr>
<td>monks3</td>
<td>21</td>
<td>94.44</td>
<td>9</td>
<td>97.22</td>
<td>9</td>
<td>97.22</td>
<td>9</td>
<td>97.22</td>
</tr>
</tbody>
</table>

Fig. 2:Comparing CDTL (FLIP2) with other learning algorithms (from Clementine v. 5.2.1):
Experiments (2/2)

The use of a multitree allows the generation of multiple solutions which share common parts, thus allowing a sublinear growing of resources:

*Fig. 3: Time and memory required by FLIP2 and C5.0 with boosting depending on the number of solutions (iterations):*
Conclusions and Future Work

* Unified framework: new splitting criterion, node selection criterion and tree-selection criterion all based on descriptive MDL. Resulting accuracy on first implemented system (FLIP2) is comparable to the most popular ML methods.

* Functional Logic Language: Extension for constructor-based data $\Rightarrow$ XML applications.

* The multi-tree allows an efficient structure for the generation of multiple solutions with sublinear growing time/memory.

Current and Future work:

- Implementation of all the possible partitions.
- Evaluation of different hypotheses combination techniques on the multitree: voting, boosting, etc.
POSTER: 200 x 100 cms = 20000 cm²
1 full = 21 x 29,7 = 623 cm²
1 (títol) + 3 files de 3 + 1 conclusions