Learning MDL-guided Decision Trees for Constructor-Based Languages[1]

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.
    - *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.
    - *However, constructor data-types must be flattened.*

$$\Downarrow$$

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:

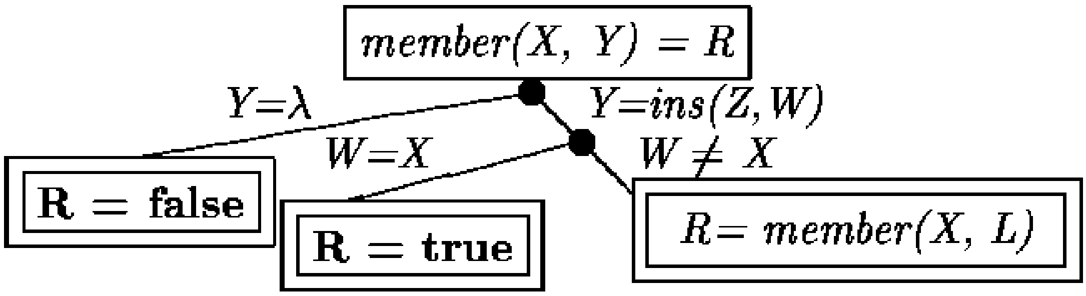
$$E^+ = \left\{ \begin{array}{l} e_1 : member(a, \lambda) = false \\ e_2 : member(b, ins(\lambda, a)) = false \\ e_3 : member(c, \lambda) = false \\ e_4 : member(c, ins(\lambda, b)) = false \\ e_5 : member(a, ins(ins(\lambda, b), d)) = false \\ e_6 : member(a, ins(ins(\lambda, b), a)) = true \\ e_7 :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

- Hypotheses are represented as conditional functional logic rules:

$$\begin{array}{ll} (i) & member(X, \lambda) = false \\ (ii) & member(X, ins(Z, X)) = true \\ (iii) & member(X, ins(L, W)) = member(X, L) \Leftarrow W \neq X \end{array}$$

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.

$$\Downarrow$$

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

Descriptive MDL

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

$$\boxed{h_{MAP} = \mathrm{argmax}_{h \in H} P(h \mid E) = \mathrm{argmin}_{h \in H} (K(h) + K(E \mid h))}$$

- 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:

#	Partition on Attribute X_i (Split)
1	$X_i = a_1 \mid X_i = a_2 \mid \ldots \mid X_i = a_k$
2	$X_i = c_0 \mid \ldots \mid X_i = c_k(Y_1, \ldots, Y_{k_m})$
3	$X_i < t \mid X_i \geq t^6$where t is a threshold
4	$X_i = Y$ where $Y \in \{X_1, \ldots, X_n\}$ and $Y \neq X_i$
5	$X_i = a \mid X_i \neq a^7$
6	$a_1 = f(Y_1, \ldots, Y_n) \mid \ldots \mid a_n = f(Y_1, \ldots, Y_n)$ where $\exists! Y_i \in \{X_1, \ldots, X_n\}$
7	$X_i = f(Y_1, \ldots, Y_n)$
8	$a_1 = g(Y_1, \ldots, Y_n) \mid \ldots \mid a_n = g(Y_1, \ldots, Y_n) \ldots$ where $\exists! Y_i \in \{X_1, \ldots, X_n\}$ and $g \in \Sigma_B$
9	$X_i = g(Y_1, \ldots, Y_n)$ where $g \in \Sigma_B$

- Expressiveness comparison:

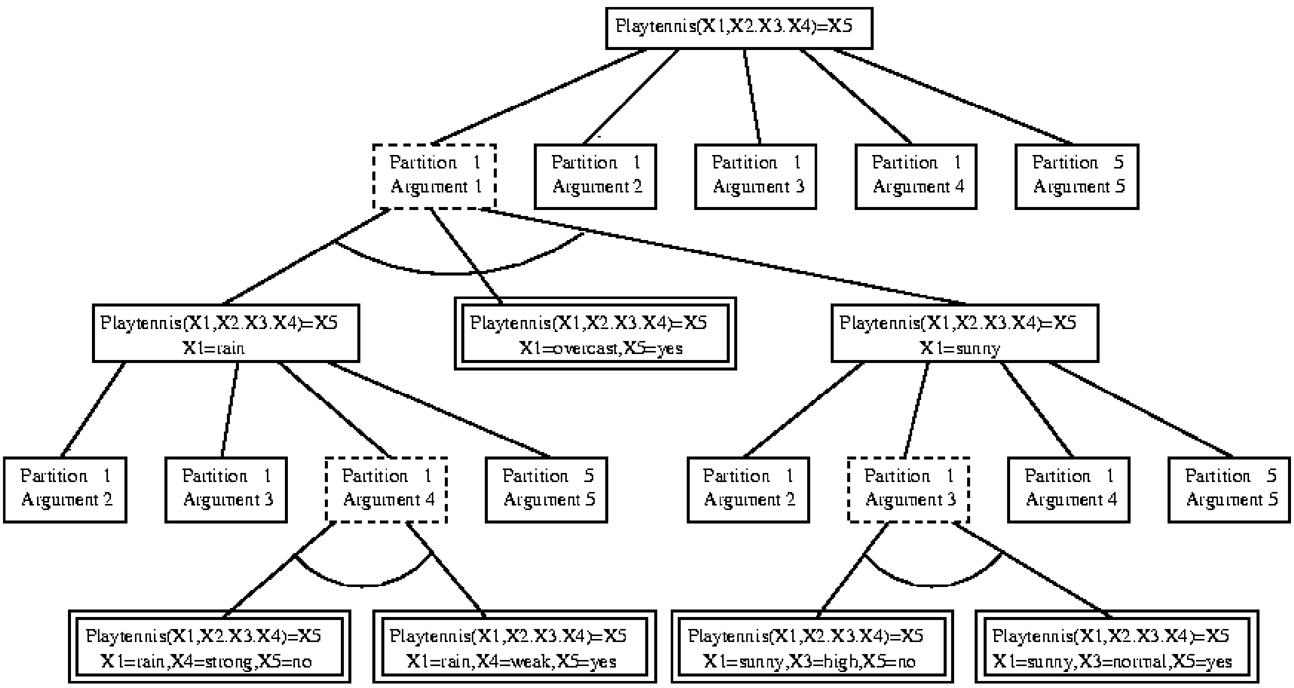
#	ID3	FOIL	CRG	CDTL
1	×	×	×	×
2	-	-	×	×
3	×	×	-	×
4	-	×	×	×
5	-	-	-	×
6	-	×	-	×
7	-	-	-	×
8	-	×	-	×
9	-	-	-	×

Constructing a Multitree

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

This constitutes a *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:

Numtree	1		10		100		1000	
Example	Rules	Accuracy	Rules	Accuracy	Rules	Accuracy	Rules	Accuracy
cars	140	86.57	119	86.69	119	86.69	110	87.50
house-votes	23	89.45	10	94.50	10	94.50	7	94.50
tic-tac-toe	111	75.99	101	71.59	94	75.78	73	77.87
nursery	517	93.00	440	94.86	440	94.86	345	93.87
monks1	9	100	5	100	5	100	5	100
monks2	44	62.50	38	64.35	28	63.19	26	62.04
monks3	21	94.44	9	97.22	9	97.22	9	97.22

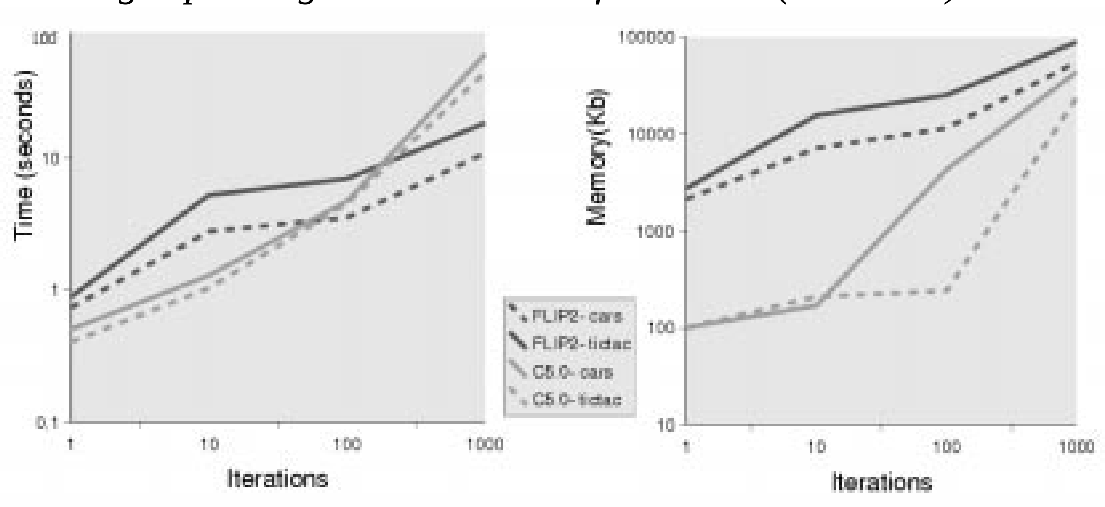
Fig. 2: Comparing CDTL (FLIP2) with other learning algorithms (from Clementine v. 5.2.1):

Example	FLIP2	C5.0	Rules	TrainNet
cars	87.5	88.54	85.88	95.49
house-votes	94.50	94.50	94.5	95.87
tic-tac-toe	77.87	80.38	77.45	79.96
nursery	93.87	95.99	95.73	96.74
monks1	100	87.90	100	88.71
monks2	62.04	65.05	65.74	99.77
monks3	97.22	97.22	94.44	96.06

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 cm^2
1 full = 21 x 29,7 = 623 cm^2
1 (títol) + 3 files de 3 + 1 conclusions