

Rule-based Machine Learning Methods for Functional Prediction

Sholom M. Weiss and Nitin Indurkha

Journal of Artificial Intelligence Research 3 (1995)

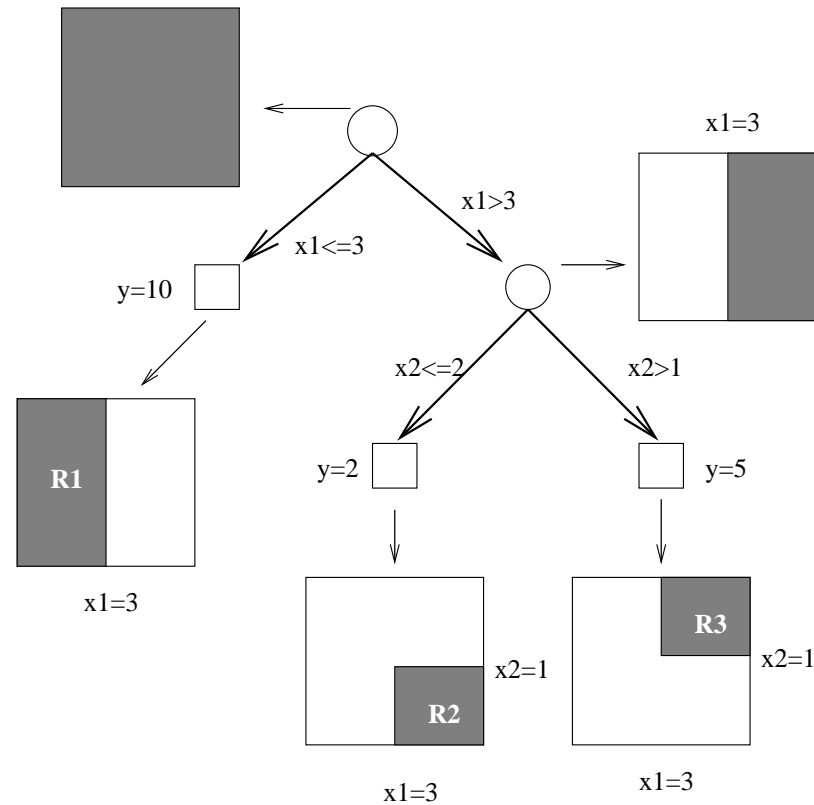
The Prediction Problems

- The General form of a prediction problem: Learn $f()$

$$y = f(x_1, x_2, \dots, x_n) + \epsilon \quad (1)$$

- f : predictor variables (\mathbf{x}) \rightarrow response variable (y).
- Two main types of prediction problems:
 - **Classification:** y is a (discrete) class label
 - **Regression:** $y \in \mathcal{R}$ is a continuous variable
- “*Regression = Classification among continuous classes*”

Regression Tree Example



$$\text{if } \mathbf{x} \in R_i \text{ then } f(\mathbf{x}) = k_i = \text{median}(y_j^i) \quad (2)$$

Regression Trees

- Dynamic feature selection (prefer relevant features near root)
- Explanatory capabilities (translated to DNF)

but ...

- Cannot represent simple (linear) functions compactly.
- RT is discrete and predicts continuous variables.
- Produces discrete regions (discontinuous at boundaries).

New method for inducing regression rules is proposed here

Regression by Rule Induction

- Regression Trees \equiv DNF \equiv Rule Induction Model
- A Rule Set is of the form:

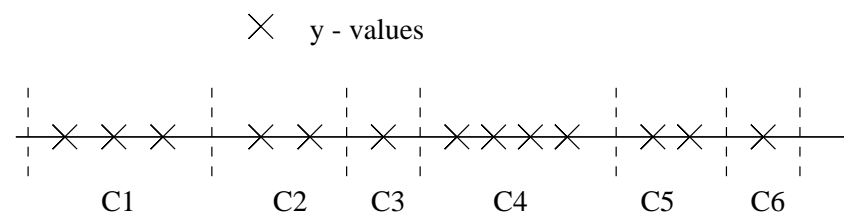
$$\{x_1 \leq 3 \rightarrow y = 10, x_2 \leq 1 \rightarrow y = 2, \text{ otherwise } y = 5\} \quad (3)$$

- More than one rules can fire at a time
- Use *decision list*: Pick first rule that triggers

$$\text{if } i < j \text{ and } \mathbf{x} \in R_i \ \& \ \mathbf{x} \in R_j \text{ then } f(\mathbf{x}) = k_i \quad (4)$$

Regression as classification

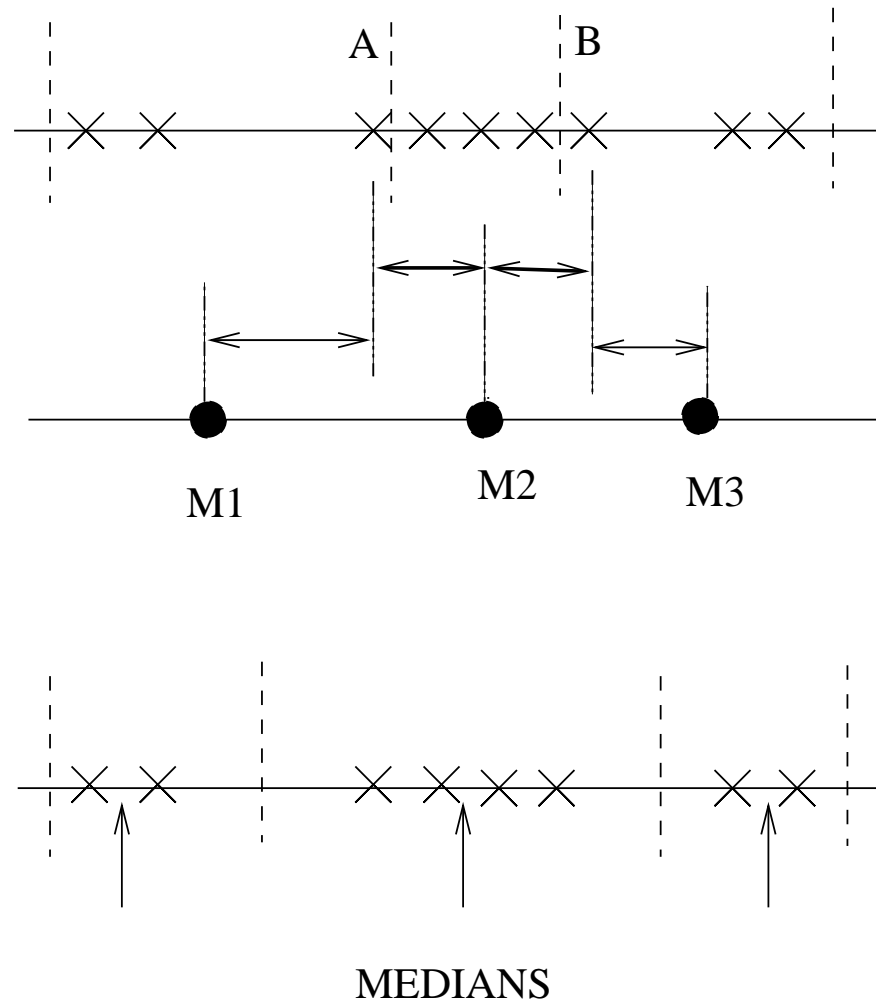
- Given training data $\mathcal{X} \{ \mathbf{x}_j, y_j \}_{j=1}^N$
- Sort all y_j ($i < j \rightarrow y_i < y_j$)
- Make use of natural ordering in $y \in \mathcal{R}$
- Divide \mathcal{R} into regions with approx. equal values
- Each region C_i is a class now.



Generating Pseudo Classes

1. Input: $\mathcal{Y} = \{y_j\}$ = sorted set of N real numbers, $k = \#$ classes
2. Initialize: $\forall C_i, C_i = \text{next } \frac{N}{k}$ samples from \mathcal{Y} ; $t = 0$
3. Compute Error $E(t) = \sum_{i=1}^k \frac{1}{|C_i|} \sum_{y \in C_i} (y - \text{mean}(C_i))^2$
4. For each $y_j \in \mathcal{Y}$ do
5. When $y_j \in C_i$
6. if $\Delta[y_j, \text{mean}(C_{i-1})] < \Delta[y_j, \text{mean}(C_i)] \rightarrow$ Move y_j to C_{i-1}
7. if $\Delta[y_j, \text{mean}(C_{i+1})] < \Delta[y_j, \text{mean}(C_i)] \rightarrow$ Move y_j to C_{i+1}
8. $t = t + 1$; Compute $E(t)$ (using 3); if $E(t) < E(t - 1)$ go to 4

Generating Pseudo Classes (Example)



Covering Rule Set

1. Covering rule set for $\mathcal{X}' = \{\mathbf{x}_i, C_i\}_{i=1}^N$
2. Each class C_i has an associated median(y_i) (classes are sorted)
3. For $C_i = C_1$ to C_{k-1} do
 - create X_i^+ = all examples in C_i ,
 - create X_i^- = all examples in C_{i+1}, \dots, C_k
 - RS_i = Rule set (decision tree) for 2 class problem
 $\langle X_i^+, X_i^- \rangle$

Pruning

- Necessary for over-fitting and better generalization
- Tree pruning same for classification and regression
- Rule set pruning is far more difficult
- Rules or part of rules (single conjunction) are removed
- A sequence of pruned rule sets are obtained

Improvements

- Use linear regression instead of single value in a region
- Use Non-linear techniques like k -NN in a region:

$$y_{knn}(\mathbf{x}) = \frac{1}{K} \sum_{k=1}^K y_k \text{ for } K \text{ nearest neighbors of } \mathbf{x} \quad (5)$$

- Feature selection of DT lessens the “curse” for k -NN
- **Model Combination** using over K models $\{M_k\}_{k=1}^K$:

$$y(\mathbf{x}) = \sum_{k=1}^K w_k M_k(\mathbf{x}) \quad (6)$$

- $M_k = k$ -NN with different K or different size (pruned) DT or RS

Results and Conclusion

- Rule set outperforms regression trees
- Combination of Rule with k -NN is better than both separately
- Rule set : more compact due to overlap, hard to prune
- Improvement: instead of rules like $x_3 < 2 \rightarrow y = 4$ one can use $\mathbf{w}_i^T x < 1 \rightarrow y = 10$: rule extraction from NN
- Interesting combination of k -NN and decision trees