RETIS: Regression Tree Induction System

Employing Linear Regression in Regression Tree Leaves

Based on Aram Karalić (1992)

ECAI'92 Proceedings, pp. 440–441

1 Introduction

Regression trees, similar to classification trees, are used for learning relationships between attributes and continuous target variables. Traditional regression tree algorithms like CART (Breiman et al., 1984) assign constant values (typically the mean) to leaf nodes. The RETIS (Regression Tree Induction System) algorithm, introduced by Aram Karalić at the 10th European Conference on Artificial Intelligence (ECAI'92), modifies this approach by employing multivariate linear regression models in the leaves instead of constants.

1.1 Motivation

The key insight behind RETIS is that data well-predicted by a linear model can have large variance. Using variance reduction alone as a splitting criterion may be suboptimal when the underlying relationship is locally linear. By fitting linear models in the leaves, RETIS can:

- Capture local linear trends and attribute interactions more effectively
- Produce smaller, more interpretable trees
- Achieve lower prediction errors compared to constant-leaf regression trees
- Better represent piecewise-linear functions

1.2 Comparison with CART

While CART uses the average target value in each leaf, RETIS uses all available attributes to form a complete linear regression model in each leaf. This fundamental difference affects both tree construction and pruning strategies.

2 Algorithm Description

2.1 Tree Structure

A RETIS tree has the following components:

- **Internal nodes**: Contain attribute tests (typically binary splits for continuous attributes)
- **Leaf nodes**: Contain multivariate linear regression models of the form:

  $$\hat{y} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n \tag{1}$$

  where β_i are coefficients estimated using least squares regression on the training examples reaching that leaf

2.2 Splitting Criterion

The splitting criterion in RETIS differs fundamentally from variance-based methods. Instead of measuring variance reduction, RETIS evaluates splits based on the **mean squared error (MSE) of the linear models** fitted to the resulting subsets.

For a candidate split dividing examples into left subset I_L and right subset I_R:

$$\text{MSE}_{\text{split}} = \frac{1}{|I_L|}\sum_{i \in I_L}(y_i - \hat{f}_L(x_i))^2 + \frac{1}{|I_R|}\sum_{i \in I_R}(y_i - \hat{f}_R(x_i))^2 \tag{2}$$

where $\hat{f}_L$ and $\hat{f}_R$ are the optimal linear regression models for the left and right subsets.

Key principle: The split that minimizes the combined MSE of linear models in both child nodes is selected. This differs from CART, which minimizes variance without accounting for linear fit quality.

2.3 Tree Construction Process

The algorithm follows a standard top-down recursive partitioning approach:

1. **Initialize**: Start with all training examples at the root node
2. **Splitting phase**: For each node:
   - For each continuous attribute x_j:
     - Consider candidate split points (e.g., midpoints between consecutive sorted values)
     - For each split point s, partition examples into $x_j \leq s$ (left) and $x_j > s$ (right)
     - Fit linear regression models to both subsets
     - Calculate the combined MSE
   - For discrete attributes: Consider all possible binary partitions
   - Select the attribute and split point yielding minimum MSE
3. **Stopping criteria**: Stop splitting if:
   - Node contains fewer than a minimum number of examples
   - MSE reduction is below a threshold
   - Maximum tree depth is reached
4. **Leaf assignment**: When a node becomes a leaf, fit a final multivariate linear regression model using all examples in that node
5. **Recurse**: Apply the process recursively to child nodes

2.4 Pruning Strategy

RETIS employs a Bayesian pruning approach based on the m-estimate method, adapted from decision tree pruning techniques. The pruning process:

1. First grows an oversized tree
2. Estimates the error of each internal node on unseen examples using the m-estimate (Bayesian approach)
3. Compares the estimated error of each internal node with the combined error of its subtrees
4. Prunes subtrees when the parent node's error estimate is lower than the combined error of its children

The m-parameter in the m-estimate has intuitive meaning and can be set through cross-validation. This approach provides a principled way to balance model complexity against prediction accuracy.

Note: Because linear models provide better local fits, RETIS typically produces shallower trees than CART, naturally reducing the need for aggressive pruning.

3 Computational Complexity

3.1 Time Complexity

The computational cost of RETIS is higher than standard regression trees because each split evaluation requires:

- **Linear system solving**: Fitting least-squares regression for each candidate split
- **Multiple evaluations**: Testing all attributes and split points

For a dataset with N examples, M attributes, and average K candidate splits per attribute:

- Sorting values per attribute: $O(N \log N)$
- Fitting linear model per candidate: $O(NM^2)$ for M predictors
- Total per node: $O(KM \cdot NM^2) = O(KM^3N)$
- Worst-case overall: $O(N^2M^4)$ for deep trees

This is substantially more expensive than variance-based trees which only compute means.

3.2 Space Complexity

- Store regression coefficients (M coefficients per leaf)
- Typically fewer leaves than CART due to better fits
- Space: $O(LM)$ where L is the number of leaves

3.3 Practical Mitigations

To reduce computational burden:

- Use quantile-based sampling of split points (e.g., test only deciles)
- Employ heuristics for discrete attribute grouping
- Apply early stopping based on minimum improvement thresholds
- Use efficient linear algebra libraries for regression fitting

4 Key Advantages

1. **Better approximation**: Linear models capture local trends more accurately than constants
2. **Smaller trees**: Fewer splits needed due to more expressive leaf models
3. **Interpretability**: Both tree structure and linear coefficients are interpretable
4. **Attribute importance**: Coefficient magnitudes indicate variable relevance within regions
5. **Lower prediction error**: Experimental results show reduced classification errors

5 Limitations and Considerations

- **Computational cost**: Significantly slower than CART due to repeated regression fitting
- **Overfitting risk**: Linear models in small leaves may overfit; requires careful pruning
- **Multicollinearity**: Linear models may be unstable if predictors are highly correlated
- **Sample size**: Each leaf needs sufficient examples to fit stable linear models (typically $> M \times 5$)
- **Categorical variables**: High-cardinality categoricals increase complexity exponentially

6 Relationship to Other Methods

RETIS belongs to the family of **model trees**—regression trees with models (rather than constants) in leaves:

- **CART** (1984): Uses constant predictions (means) in leaves
- **RETIS** (1992): Uses linear models with all attributes in leaves
- **M5** (Quinlan, 1992): Uses linear models but prunes them with heuristics to manage complexity
- **M5'** (Wang & Witten, 1997): Enhanced version of M5
- **GUIDE** (Loh, 2002): Unbiased variable selection methods
- **SECRET** (Dobra & Gehrke, 2002): Scalable linear regression trees

The key distinguishing feature of RETIS is its use of MSE of linear models as the splitting criterion, and its use of all attributes in the linear models.

7 Experimental Results

According to Karalič's original work, RETIS was evaluated on artificial and real-world datasets. The results demonstrated that:

- The modification of using linear leaves is **beneficial**
- RETIS leads to **smaller classification errors** of induced regression trees
- Trees are generally **smaller and more interpretable** than CART trees
- Performance improvements are especially notable on datasets with:
  - Strong attribute interactions
  - Piecewise-linear target functions
  - Locally linear relationships

8 Implementation Considerations

When implementing RETIS, consider:

1. **Minimum leaf size**: Set to at least $5M$ to $10M$ examples (where M is the number of attributes) to ensure stable linear model fitting
2. **Regularization**: Consider ridge regression or LASSO in leaves to handle multicollinearity and prevent overfitting

3. **Split point selection**: Use strategic sampling (e.g., percentiles) rather than exhaustive search
4. **Numerical stability**: Handle ill-conditioned matrices when fitting linear models; may need to fall back to constant models in problematic leaves
5. **Cross-validation**: Tune the m-parameter for pruning using cross-validation
6. **Standardization**: Consider standardizing attributes to improve numerical stability and coefficient interpretation

9 Conclusion

RETIS represents an important early contribution to model tree methods. By replacing constant predictions with linear regression models, it achieves better predictive accuracy while maintaining interpretability. The algorithm's core innovation—using MSE of linear models as a splitting criterion—enables the discovery of locally linear relationships in data.

While computationally more expensive than CART, RETIS and its descendants (M5, M5', etc.) have proven valuable for applications requiring both accuracy and interpretability, particularly in knowledge discovery and data mining contexts.

10 References

1. Karalić, A. (1992). Employing Linear Regression in Regression Tree Leaves. In B. Neumann (Ed.), *Proceedings of the 10th European Conference on Artificial Intelligence (ECAI'92)*, pp. 440–441. Vienna, Austria. John Wiley & Sons, Inc.
2. Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth.
3. Quinlan, J. R. (1992). Learning with Continuous Classes. In *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence*, pp. 343–348. World Scientific.
4. Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.
5. Wang, Y., & Witten, I. H. (1997). Induction of Model Trees for Predicting Continuous Classes. In *Proceedings of the Poster Papers of the European Conference on Machine Learning*. University of Economics, Faculty of Informatics and Statistics, Prague.
6. Torgo, L. (1997). Functional Models for Regression Tree Leaves. In *Proceedings of the 14th International Conference on Machine Learning*, pp. 385–393. Morgan Kaufmann.