Mathematical Foundations of CART (1984): A Production Implementation

Implementation Guide

December 6, 2025

1 Introduction

This document describes the mathematical foundations of Classification and Regression Trees (CART) as originally defined by Breiman, Friedman, Olshen, and Stone in their seminal 1984 work. Our implementation follows these principles exactly, distinguishing it from modern approximations.

2 Tree Structure

A binary tree T consists of nodes $t \in T$. Each node is either:

- A **terminal node** (leaf): $t \in \tilde{T}$, where $\tilde{T}$ denotes the set of leaves
- An **internal node**: has two children t_L (left) and t_R (right)

The tree defines a partition of the feature space $\mathcal{X}$ into regions $\{R_t : t \in \tilde{T}\}$.

3 Splitting Rules

3.1 Numerical Features

For a numerical feature X_j, a split at node t is defined by a threshold c:

$$t_L = \{x : X_j(x) \leq c\} \tag{1}$$
$$t_R = \{x : X_j(x) > c\} \tag{2}$$

3.2 Categorical Features (True CART Methodology)

For a categorical feature X_j with domain $\mathcal{C} = \{c_1, c_2, \ldots, c_k\}$, a split is defined by a **subset** $S \subset \mathcal{C}$:

$$t_L = \{x : X_j(x) \in S\} \tag{3}$$
$$t_R = \{x : X_j(x) \notin S\} \tag{4}$$

This is fundamentally different from one-hot encoding. Instead of creating k binary features, CART finds the *optimal binary partition* of the category set.

3.2.1 Efficient Categorical Splitting

For binary classification with classes $\{0, 1\}$, there exists an optimal ordering trick:

1. For each category c_i, compute $\bar{y}_i = \mathbb{E}[Y|X_j = c_i]$
2. Sort categories by $\bar{y}_i$: $c_{(1)}, c_{(2)}, \ldots, c_{(k)}$
3. Only consider splits of the form $S = \{c_{(1)}, \ldots, c_{(m)}\}$ for $m = 1, \ldots, k-1$

This reduces complexity from $O(2^k)$ to $O(k \log k)$.

4 Impurity Measures

4.1 Classification: Gini Impurity

For a node t with K classes, let $p(j|t)$ be the proportion of class j samples at node t. The Gini impurity is:

$$i(t) = 1 - \sum_{j=1}^{K} p^2(j|t) \tag{5}$$

Properties:

- $i(t) = 0$ when node is pure (all samples same class)
- $i(t)$ is maximum when classes are uniformly distributed
- $i(t) \in [0, 1 - 1/K]$

4.2 Regression: Mean Squared Error

For regression, the impurity at node t is:

$$i(t) = \frac{1}{N_t} \sum_{x_i \in t} (y_i - \bar{y}_t)^2 \tag{6}$$

where N_t is the number of samples at t and $\bar{y}_t = \frac{1}{N_t} \sum_{x_i \in t} y_i$.

5 Split Selection

The best split at node t maximizes the **impurity decrease**:

$$\Delta i(s, t) = i(t) - p_L i(t_L) - p_R i(t_R) \tag{7}$$

where:

- s is a candidate split
- $p_L = N_{t_L}/N_t$ and $p_R = N_{t_R}/N_t$
- N_{t_L}, N_{t_R} are sample counts in children

We select:

$$s^* = \arg\max_{s \in S_t} \Delta i(s, t) \tag{8}$$

where S_t is the set of all possible splits at node t.

6 Cost-Complexity Pruning

6.1 The Cost-Complexity Measure

For a tree T, define the cost-complexity measure:

$$R_\alpha(T) = R(T) + \alpha|\tilde{T}| \tag{9}$$

where:

- $R(T) = \sum_{t \in \tilde{T}} r(t) \cdot p(t)$ is the error of tree T
- $r(t)$ is the error rate at leaf t
- $p(t) = N_t/N$ is the proportion of samples at t
- $|\tilde{T}|$ is the number of leaves
- $\alpha \geq 0$ is the complexity parameter

6.2 Weakest Link Pruning

For each internal node t, let T_t denote the subtree rooted at t. Define:

$$g(t) = \frac{R(t) - R(T_t)}{|\tilde{T}_t| - 1} \tag{10}$$

This represents the *effective* α at which pruning node t becomes favorable.

Algorithm: Weakest Link Pruning

Initialize $T_0 = T_{\max}$ (fully grown tree)
for $k = 0, 1, 2, \ldots$ **do**
  Find node $\tilde{t}_k$ with minimum $g(t)$ among internal nodes of T_k
  Set $\alpha_{k+1} = g(\tilde{t}_k)$
  Prune $\tilde{t}_k$ to create T_{k+1}
  if T_{k+1} is root only **then**
    Break
  end if
end for

This produces a nested sequence:

$$T_{\max} \supset T_1 \supset T_2 \supset \cdots \supset \{\text{root}\} \tag{11}$$

6.3 Selecting Optimal α

Use cross-validation or a validation set to select α^*:

$$\alpha^* = \arg\min_{\alpha} \text{Error}_{\text{validation}}(T(\alpha)) \tag{12}$$

where $T(\alpha)$ is the pruned tree for complexity α.

7 Why Subset Splitting is Superior

7.1 Information Preservation

One-hot encoding creates k binary features from a k-category feature. Each binary split can only separate one category from the others, requiring up to $k-1$ splits to achieve any partition.

Subset splitting can achieve *any* binary partition in a single split, preserving the full information of the categorical variable.

7.2 Tree Depth

Example: Consider a 4-category feature: $\{A, B, C, D\}$

- **One-hot encoding:** To achieve partition $\{A, C\}$ vs $\{B, D\}$ requires multiple splits over depth 2-3
- **Subset splitting:** Achieved in single split at depth 1

7.3 Statistical Efficiency

Subset splitting maintains the categorical nature of the variable, leading to:

- More interpretable trees
- Better sample efficiency (fewer splits needed)
- Natural handling of category similarities

8 Prediction

8.1 Classification

For a leaf node t, predict the majority class:

$$\hat{y}(x) = \arg\max_{j} p(j|t) \quad \text{for } x \in t \tag{13}$$

8.2 Regression

For a leaf node t, predict the mean:

$$\hat{y}(x) = \bar{y}_t = \frac{1}{N_t} \sum_{x_i \in t} y_i \quad \text{for } x \in t \tag{14}$$

9 Implementation Considerations

9.1 Computational Complexity

- **Numerical splits:** $O(n \log n)$ per feature (sorting)
- **Categorical splits (binary):** $O(k \log k)$ using ordering trick
- **Categorical splits (general):** $O(2^k)$ or use greedy heuristic
- **Tree growing:** $O(n \cdot p \cdot \log n)$ for n samples, p features
- **Pruning:** $O(|T|^2)$ for tree with $|T|$ nodes

9.2 Numerical Stability

When computing impurity:

- Check for division by zero when $N_t = 0$
- Use numerical stable formulas for variance computations
- Handle edge cases where all samples have same feature value

10 Conclusion

This implementation follows the original CART methodology rigorously, providing:

1. True categorical handling via subset splitting
2. Cost-complexity pruning for optimal tree selection
3. Transparent, interpretable tree structures
4. Educational value for understanding the algorithm

The key insight is that CART treats categorical variables as first-class citizens, not as sets of binary indicators. This leads to more efficient, interpretable, and accurate models.