

Decision Trees - Derivations & Proofs

Paul F. Roysdon, Ph.D.

Contents

1 Mathematical Derivations & Proofs	1
1.1 Introduction	1
1.2 Data and Notation	1
1.3 Model Formulation	2
1.4 Impurity Measures (Classification) and Variance (Regression)	2
1.5 Impurity Reduction (Split Gain)	2
1.6 Optimizing Splits	3
1.7 Recursive Partitioning Algorithm (TDIDT)	3
1.8 Cost-Complexity Pruning (CART)	3
1.9 Weighted Samples and Class Imbalance	4
1.10 Computational Complexity	4
1.11 Summary of Variables and Their Dimensions	4

1 Mathematical Derivations & Proofs

1.1 Introduction

Decision trees are non-parametric models for classification and regression. A tree recursively partitions the feature space into regions and assigns a constant prediction per region. In classification, splits are chosen to produce *pure* nodes according to impurity measures such as **Gini** or **entropy**; in regression, splits reduce within-node variance. We derive impurity formulas, the split “gain”, and the recursive construction, while also presenting an ERM view, pruning, and complexity. All variables follow the notation of this section.

1.2 Data and Notation

Let the training dataset be

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n, \quad \mathbf{x}_i \in \mathbb{R}^d \text{ (column, } d \times 1), \\ y_i \in \begin{cases} \{1, 2, \dots, K\} \text{ (classification)} \\ \mathbb{R} \text{ (regression)}. \end{cases}$$

At any node (region) t , denote by $\mathcal{D}_t \subseteq \mathcal{D}$ the samples at t , with size $n_t = |\mathcal{D}_t|$. For classification, let

$$p_{k,t} = \frac{1}{n_t} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_t} I\{y_i = k\}, \quad k = 1, \dots, K.$$

Here, $I\{\cdot\}$ is the indicator function, and the dimensions $n, d, K \in \mathbb{N}$ (scalars); $\mathbf{x}_i \in \mathbb{R}^d$; y_i scalar; n_t scalar; $p_{k,t} \in [0, 1]$ dimensionless.

1.3 Model Formulation

Herein we build a piecewise-constant predictor and ERM. A binary tree partitions \mathbb{R}^d into leaf regions $\{R_\ell\}_{\ell=1}^L$ and predicts

$$f_T(\mathbf{x}) = \sum_{\ell=1}^L c_\ell \mathbf{1}\{\mathbf{x} \in R_\ell\}.$$

Given a loss $\ell(\hat{y}, y)$, empirical risk minimization (ERM) seeks

$$\hat{T} \in \arg \min_T \frac{1}{n} \sum_{i=1}^n \ell(f_T(\mathbf{x}_i), y_i).$$

Exactly optimizing over all trees is intractable, so we use greedy top-down induction with a node-wise surrogate (impurity) and set leaf predictions as loss minimizers:

$$\begin{aligned} c_\ell^* &\in \arg \min_{c \in \mathbb{R}} \sum_{(\mathbf{x}_i, y_i) \in R_\ell} \ell(c, y_i) \\ \Rightarrow \quad &\begin{cases} \text{classification: } c_\ell^* = \arg \max_k p_{k,\ell} \text{ (majority)} \\ \text{regression (squared): } c_\ell^* = \bar{y}_\ell \end{cases} \end{aligned}$$

1.4 Impurity Measures (Classification) and Variance (Regression)

Gini impurity. At node t ,

$$I_G(t) = 1 - \sum_{k=1}^K p_{k,t}^2.$$

Properties: $I_G(t) \in [0, 1 - \frac{1}{K}]$; $I_G(t) = 0$ iff $p_{k,t} = 1$ for some k . Binary case ($K = 2$, $p_{1,t} = p$): $I_G(t) = 1 - (p^2 + (1-p)^2) = 2p(1-p)$.

Entropy.

$$H(t) = - \sum_{k=1}^K p_{k,t} \log p_{k,t} \quad (\text{bits if } \log_2, \text{ nats if } \log_e).$$

Regression variance.

$$V(t) = \frac{1}{n_t} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_t} (y_i - \bar{y}_t)^2, \quad \bar{y}_t = \frac{1}{n_t} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_t} y_i.$$

1.5 Impurity Reduction (Split Gain)

Consider a candidate binary split of node t into children t_L (left) and t_R (right) with sizes n_L, n_R ($n_L + n_R = n_t$). Define the weighted post-split impurity, $\text{Imp}(\cdot)$

$$I_{\text{split}} = \frac{n_L}{n_t} \text{Imp}(t_L) + \frac{n_R}{n_t} \text{Imp}(t_R), \quad \text{Imp} \in \{I_G, H, V\}.$$

The impurity reduction (gain) is

$$\Delta \text{Imp} = \text{Imp}(t) - I_{\text{split}} = \text{Imp}(t) - \left(\frac{n_L}{n_t} \text{Imp}(t_L) + \frac{n_R}{n_t} \text{Imp}(t_R) \right).$$

A good split maximizes ΔImp .

Proof of Impurity Reduction

Proof. At node t , the total impurity is given by $n_t I_G(t)$. After splitting, the total impurity in the children is $n_L I_G(L) + n_R I_G(R)$. The difference,

$$n_t I_G(t) - (n_L I_G(L) + n_R I_G(R)),$$

represents the total reduction in impurity. Dividing by n_t gives ΔI_G , the average reduction per sample. ■

Information-theoretic identity (classification). With $\text{Imp} = H$ and $S \in \{L, R\}$ the split outcome,

$$\begin{aligned} \Delta H &= H(Y \mid \mathbf{X} \in t) - \frac{n_L}{n_t} H(Y \mid \mathbf{X} \in t_L) - \frac{n_R}{n_t} H(Y \mid \mathbf{X} \in t_R) \\ &= I(Y; S \mid \mathbf{X} \in t). \end{aligned}$$

ANOVA identity (regression). Let $\bar{y}_t, \bar{y}_L, \bar{y}_R$ be means. Then

$$\begin{aligned} V(t) - \frac{n_L}{n_t} V(t_L) - \frac{n_R}{n_t} V(t_R) &= \frac{n_L}{n_t} (\bar{y}_L - \bar{y}_t)^2 + \frac{n_R}{n_t} (\bar{y}_R - \bar{y}_t)^2 \\ &= \frac{n_L n_R}{n_t^2} (\bar{y}_L - \bar{y}_R)^2, \end{aligned}$$

so variance reduction equals explained variance between children.

1.6 Optimizing Splits

Numeric features. Within node t , for feature j , sort samples by x_{ij} once; candidate thresholds are mid-points between consecutive *distinct* values (for classification, only where class changes). Scan all candidates in $O(n_t)$ using cumulative counts/sums after an $O(n_t \log n_t)$ sort.

Categorical features. For binary trees, a split corresponds to a bipartition $A \subset \{1, \dots, C\}$: $x_j \in A$ vs. $x_j \notin A$. For $K = 2$, order categories by class-1 proportion and scan $C - 1$ thresholds; for general K , consider heuristics (grouping rare categories) or multiway splits: $I_{\text{split}} = \sum_c \frac{|t_c|}{n_t} \text{Imp}(t_c)$.

1.7 Recursive Partitioning Algorithm (TDIDT)

1. **Start at root:** node t contains \mathcal{D} ($n_t = n$).
2. **At node t :**
 - (a) For each feature and candidate threshold/partition, compute ΔImp .
 - (b) Select the split with maximal ΔImp ; form t_L, t_R and partition \mathcal{D}_t .
3. **Stopping criteria:** stop if (i) $n_t < n_{\min}$, (ii) $\max \Delta \text{Imp} < \gamma_{\min}$, (iii) depth limit, or (iv) purity ($I_G = 0$).
4. **Leaf prediction:** majority class (classification) or mean (regression) over \mathcal{D}_t .

1.8 Cost-Complexity Pruning (CART)

Grow a large tree T then prune. For any subtree $U \subseteq T$, define penalized risk

$$\hat{R}_\alpha(U) = \frac{1}{n} \sum_{\ell \in \mathcal{L}(U)} \sum_{(\mathbf{x}_i, y_i) \in R_\ell} \ell(c_\ell^*, y_i) + \alpha |\mathcal{L}(U)|,$$

with $\alpha \geq 0$. For internal node t with subtree T_t , the weakest-link pruning threshold is

$$\alpha_t = \frac{\widehat{R}(t) - \widehat{R}(T_t)}{|\mathcal{L}(T_t)| - 1},$$

where $\widehat{R}(\cdot)$ omits the penalty. Iteratively prune the smallest α_t to get a nested sequence; select by validation or cross-validation.

1.9 Weighted Samples and Class Imbalance

With nonnegative sample weights $\{w_i\}$, replace counts by weighted counts everywhere:

$$p_{k,t} = \frac{\sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_t} w_i I\{y_i = k\}}{\sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_t} w_i}, \quad \bar{y}_t = \frac{\sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_t} w_i y_i}{\sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_t} w_i},$$

and weight the child proportions by sums of w_i in ΔImp .

1.10 Computational Complexity

At node t , numeric-feature processing costs $O(d n_t \log n_t)$ for sorting (reused down a path when possible) plus $O(d n_t)$ to scan thresholds. Balanced trees have depth $O(\log n)$.

1.11 Summary of Variables and Their Dimensions

- $\mathbf{x}_i \in \mathbb{R}^d$: i th feature vector (dimension $d \times 1$).
- $y_i \in \{1, \dots, K\}$ (classification) or $y_i \in \mathbb{R}$ (regression): target (scalar).
- n : number of samples; d : number of features; K : number of classes.
- $\mathcal{D}_t \subseteq \mathcal{D}$: samples at node t ; $n_t = |\mathcal{D}_t|$.
- $p_{k,t}$: class proportion in t (dimensionless).
- $I_G(t), H(t), V(t) \in \mathbb{R}$: Gini, entropy, variance at node t (scalars).
- t_L, t_R : child nodes; n_L, n_R their sizes; $\Delta\text{Imp} \in \mathbb{R}$: impurity reduction.
- R_ℓ : leaf region; c_ℓ : leaf prediction (majority class or mean).
- n_{\min}, γ_{\min} : stopping hyperparameters (scalars).
- α : cost-complexity pruning parameter (scalar).

1.12 Summary

From first-principles: specify data/notation; define impurity (Gini/entropy) or variance; select splits that maximize impurity reduction; grow trees greedily (TDIDT); regularize via stopping or cost-complexity pruning; and set leaf predictions as loss minimizers. All quantities are declared with dimensions to support clear implementation.