Gradient Boosting Trees - Derivations & Proofs

Paul F. Roysdon, Ph.D.

Contents

1 Mathematical Derivations & Proofs

1.1 Introduction

Gradient Boosting builds a strong predictor by *additively* fitting weak learners (typically regression trees) to reduce a chosen empirical loss. At each stage, the new tree approximates the *negative functional gradient* of the loss at current predictions (the *pseudo-residuals*); optionally, a second-order (Newton) step improves efficiency and split scoring. We derive both views, state per-leaf optimal updates, and give an algorithm with all variables and dimensions.

1.2 Data and Notation

Let the training set be

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}, \qquad \mathbf{x}_i \in \mathbb{R}^d \text{ (column)}, \quad y_i \in \mathbb{R} \text{ (scalar, regression)}.$$

An additive model with M boosting stages is

$$F(\mathbf{x}) \;=\; F_0(\mathbf{x}) + \sum_{m=1}^{M} f_m(\mathbf{x}), \qquad f_m : \mathbb{R}^d \to \mathbb{R}. \tag{1}$$

We minimize the empirical risk for a differentiable loss $L : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_{\geq 0}$:

$$\min_{F} \sum_{i=1}^{n} L\big(y_i,\, F(\mathbf{x}_i)\big).$$

Dimensions/properties. $F(\mathbf{x}), F_0(\mathbf{x}), f_m(\mathbf{x}) \in \mathbb{R}$; $n, d \in \mathbb{N}$.

1.3 Model Formulation and Functional Gradients

Gradient Boosting constructs the model in a stage-wise fashion. Suppose that at iteration $m-1$ we have built a model:

$$F_{m-1}(\mathbf{x}) = F_0(\mathbf{x}) + \sum_{j=1}^{m-1} f_j(\mathbf{x}).$$

At the mth iteration, we wish to update the model:

$$F_m(\mathbf{x}) \;=\; F_{m-1}(\mathbf{x}) + \nu\, f_m(\mathbf{x}). \tag{2}$$

where $\nu \in (0, 1]$ is a *learning rate* (a scalar) that controls the contribution of each new tree.

First-order (gradient) view. Define the *pseudo-residuals* for $i = 1, \ldots, n$:

$$r_{im} \;=\; -\;\left.\frac{\partial L\big(y_i,\, F(\mathbf{x}_i)\big)}{\partial F(\mathbf{x}_i)}\right|_{F(\mathbf{x}_i)=F_{m-1}(\mathbf{x}_i)} \;\in\; \mathbb{R}. \tag{3}$$

Fit a regression tree f_m to $\{(\mathbf{x}_i, r_{im})\}$ (least squares fit), then (optionally) refine each terminal region by a one-dimensional line search (below), where $r_{im} \in \mathbb{R}$ is a scalar for each $i = 1, \ldots, n$, and the pseudo-residuals represent the direction in which the prediction $F_{m-1}(\mathbf{x}_i)$ should be adjusted to reduce the loss.

Second-order (Newton) view. Let

$$g_i \;=\; \left.\frac{\partial L}{\partial F(\mathbf{x}_i)}\right|_{F_{m-1}}, \qquad h_i \;=\; \left.\frac{\partial^2 L}{\partial F(\mathbf{x}_i)^2}\right|_{F_{m-1}} \quad (\; h_i \geq 0\;).$$

A quadratic Taylor surrogate around F_{m-1} yields leafwise closed forms and principled split scores.

1.4 Tree Base Learner and Per-Leaf Updates

Let the stage-m tree partition $\mathbb{R}^d$ into J_m disjoint regions $\{R_{jm}\}_{j=1}^{J_m}$ and predict a constant in each region:

$$f_m(\mathbf{x}) \;=\; \sum_{j=1}^{J_m} \gamma_{jm}\, I\big(\mathbf{x} \in R_{jm}\big), \qquad \gamma_{jm} \in \mathbb{R}. \tag{4}$$

Region-wise optimal updates (general loss). Given regions $\{R_{jm}\}$ (found by growing the tree on residuals/working responses), choose each leaf value by a 1D minimization:

$$\gamma_{jm} \;=\; \arg\min_{\gamma \in \mathbb{R}} \sum_{\mathbf{x}_i \in R_{jm}} L\big(y_i,\; F_{m-1}(\mathbf{x}_i) + \gamma\big). \tag{5}$$

Squared-error loss. For $L(y, F) = \frac{1}{2}(y - F)^2$, $r_{im} = y_i - F_{m-1}(\mathbf{x}_i)$ and

$$\gamma_{jm} \;=\; \frac{1}{|R_{jm}|} \sum_{\mathbf{x}_i \in R_{jm}} r_{im}. \tag{6}$$

Newton (second-order) leaf update and split score. Let $G_{jm} = \sum_{\mathbf{x}_i \in R_{jm}} g_i$ and $H_{jm} = \sum_{\mathbf{x}_i \in R_{jm}} h_i$. With ℓ_2 leaf-value regularization $\lambda \geq 0$ and an (optional) penalty $\Gamma \geq 0$ per leaf, the surrogate objective per tree is

$$\sum_{j=1}^{J_m} \Big(G_{jm}\gamma_{jm} + \tfrac{1}{2}(H_{jm} + \lambda)\gamma_{jm}^2 \Big) + \Gamma J_m,$$

minimized at

$$\boxed{\gamma_{jm}^\star \;=\; -\frac{G_{jm}}{H_{jm} + \lambda}}. \tag{7}$$

For a candidate split of a node with (G, H) into children (G_L, H_L) and (G_R, H_R), the *gain* is

$$\text{Gain} \;=\; \frac{1}{2}\left(\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{G^2}{H + \lambda} \right) \;-\; \Gamma, \tag{8}$$

which guides greedy tree construction. (Use Eqn. (5) / Eqn. (6) for first-order boosting.)

1.5 Initialization, Learning Rate, and Special Cases

Initialization. Choose the constant $F_0 \in \mathbb{R}$ that minimizes the loss:

$$F_0 \in \arg\min_{c \in \mathbb{R}} \sum_{i=1}^{n} L(y_i, c) \quad \big(\text{mean of } y_i \text{ for squared error}\big).$$

Learning rate (shrinkage). Apply Eqn. (2) with $\nu \in (0, 1]$ to temper each stages contribution.

Binary logistic (optional classification). For $y_i \in \{0, 1\}$ and $L(y, F) = -\,y \log p - (1 - y) \log(1 - p)$ with $p = \sigma(F)$, $g_i = p_{m-1}(\mathbf{x}_i) - y_i$, $h_i = p_{m-1}(\mathbf{x}_i)(1 - p_{m-1}(\mathbf{x}_i))$, and the Newton leaf formula Eqn. (7) applies; initialize $F_0 = \log\big(\bar{y}/(1 - \bar{y})\big)$.

1.6 Algorithm (Gradient/Newton Boosting with Trees)

1. **Input:** data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$; differentiable loss L; learning rate $\nu \in (0, 1]$; number of stages M; tree size J_m (or max depth); (optional) λ, Γ.
2. **Initialize** $F_0 \in \arg\min_c \sum_i L(y_i, c)$.
3. **For** $m = 1, 2, \ldots, M$:
   (a) *First-order:* compute residuals r_{im} by Eqn. (3) and fit f_m to $(\mathbf{x}_i, r_{im})$ (least squares). *Or Newton:* compute g_i, h_i, grow f_m using gains Eqn. (8), and set leaves by Eqn. (7).
   (b) Given regions $\{R_{jm}\}$, refine leaf values via Eqn. (5) (or Eqn. (7)/Eqn. (6)).
   (c) Update $F_m(\mathbf{x}) \leftarrow F_{m-1}(\mathbf{x}) + \nu\, f_m(\mathbf{x})$.
4. **Output:** $F(\mathbf{x}) = F_M(\mathbf{x})$.

1.7 Summary of Variables and Their Dimensions

- $\mathbf{x}_i \in \mathbb{R}^d$: feature vector (dimension $d \times 1$); $y_i \in \mathbb{R}$: response.
- $F(\mathbf{x}), F_m(\mathbf{x}), F_0(\mathbf{x}) \in \mathbb{R}$; $f_m(\mathbf{x}) \in \mathbb{R}$; $M \in \mathbb{N}$; $\nu \in (0, 1]$.
- $r_{im} \in \mathbb{R}$: pseudo-residual (first-order); $g_i, h_i \in \mathbb{R}$: gradient/Hessian (second-order).
- Tree at stage m: $J_m \in \mathbb{N}$ leaves; regions $R_{jm} \subset \mathbb{R}^d$; leaf values $\gamma_{jm} \in \mathbb{R}$.
- Newton aggregation: $G_{jm} = \sum_{\mathbf{x}_i \in R_{jm}} g_i \in \mathbb{R}$, $H_{jm} = \sum_{\mathbf{x}_i \in R_{jm}} h_i \in \mathbb{R}$; regularization $\lambda, \Gamma \in \mathbb{R}_{\geq 0}$.

1.8 Summary

From first principles: (i) define pseudo-residuals Eqn. (3) and fit a tree to them; (ii) set per-leaf updates by Eqn. (5) (mean residual for squared error Eqn. (6)); (iii) optionally use Newton updates Eqn. (7), which also induce the split gain Eqn. (8); (iv) update the additive model with shrinkage Eqn. (2). All quantities are declared with dimensions, and the framework extends directly to logistic/softmax losses by substituting g_i, h_i.