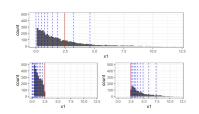
Introduction to Machine Learning

Boosting Gradient Boosting: Deep Dive XGBoost Optimization





Learning goals

- Understand details of the regularized risk in XGBoost
- Understand approximation of loss used in optimization
- Understand split finding algorithm

XGBoost uses a risk function with 3 regularization terms:

$$\mathcal{R}_{\text{reg}}^{[m]} = \sum_{i=1}^{n} L\left(y^{(i)}, f^{[m-1]}(\mathbf{x}^{(i)}) + b^{[m]}(\mathbf{x}^{(i)})\right) + \lambda_1 J_1(b^{[m]}) + \lambda_2 J_2(b^{[m]}) + \lambda_3 J_3(b^{[m]}),$$



with $J_1(b^{[m]}) = T^{[m]}$ the number of leaves in the tree to penalize tree depth.

 $J_2(b^{[m]}) = \|\mathbf{c}^{[m]}\|_2^2$ and $J_3(b^{[m]}) = \|\mathbf{c}^{[m]}\|_1$ are L2 and L1 penalties of the terminal region values $c_t^{[m]}, t = 1, \dots, T^{[m]}$.

We define $J(b^{[m]}) := \lambda_1 J_1(b^{[m]}) + \lambda_2 J_2(b^{[m]}) + \lambda_3 J_3(b^{[m]})$.

To approximate the loss in iteration m, a second-order Taylor expansion around $f^{[m-1]}(\mathbf{x})$ is computed:

$$L(y, f^{[m-1]}(\mathbf{x}) + b^{[m]}(\mathbf{x})) \approx$$

$$L(y, f^{[m-1]}(\mathbf{x})) + g^{[m]}(\mathbf{x})b^{[m]}(\mathbf{x}) + \frac{1}{2}h^{[m]}(\mathbf{x})b^{[m]}(\mathbf{x})^{2},$$

with gradient

$$g^{[m]}(\mathbf{x}) = \frac{\partial L(y, f^{[m-1]}(\mathbf{x}))}{\partial f^{[m-1]}(\mathbf{x})}$$

and Hessian

$$h^{[m]}(\mathbf{x}) = \frac{\partial^2 L(y, f^{[m-1]}(\mathbf{x}))}{\partial f^{[m-1]}(\mathbf{x})^2}.$$

Note: $g^{[m]}(\mathbf{x})$ are the negative pseudo-residuals $-\tilde{r}^{[m]}$ we use in standard gradient boosting to determine the direction of the update.



Since $L(y, f^{[m-1]}(\mathbf{x}))$ is constant, the optimization simplifies to

$$\mathcal{R}_{\text{reg}}^{[m]} = \sum_{i=1}^{n} g^{[m]}(\mathbf{x}^{(i)}) b^{[m]}(\mathbf{x}^{(i)}) + \frac{1}{2} h^{[m]}(\mathbf{x}^{(i)}) b^{[m]}(\mathbf{x}^{(i)})^{2} + J(b^{[m]}) + const$$

$$\propto \sum_{t=1}^{T^{[m]}} \sum_{\mathbf{x}^{(i)} \in R_{t}^{[m]}} g^{[m]}(\mathbf{x}^{(i)}) c_{t}^{[m]} + \frac{1}{2} h^{[m]}(\mathbf{x}^{(i)}) (c_{t}^{[m]})^{2} + J(b^{[m]})$$

$$= \sum_{t=1}^{T^{[m]}} G_{t}^{[m]} c_{t}^{[m]} + \frac{1}{2} H_{t}^{[m]}(c_{t}^{[m]})^{2} + J(b^{[m]}).$$

Where $G_t^{[m]}$ and $H_t^{[m]}$ are the accumulated gradient and Hessian values in terminal node t.



Expanding $J(b^{[m]})$:

$$egin{aligned} \mathcal{R}_{\mathsf{reg}}^{[m]} &= \sum_{t=1}^{T^{[m]}} \left(G_t^{[m]} c_t^{[m]} + rac{1}{2} H_t^{[m]} (c_t^{[m]})^2 + rac{1}{2} \lambda_2 (c_t^{[m]})^2 + \lambda_3 |c_t^{[m]}|
ight) + \lambda_1 T^{[m]} \ &= \sum_{t=1}^{T^{[m]}} \left(G_t^{[m]} c_t^{[m]} + rac{1}{2} (H_t^{[m]} + \lambda_2) (c_t^{[m]})^2 + \lambda_3 |c_t^{[m]}|
ight) + \lambda_1 T^{[m]}. \end{aligned}$$



Note: The factor $\frac{1}{2}$ is added to the L2 regularization to simplify the notation as shown in the second step. This does not impact estimation since we can just define $\lambda_2=2\tilde{\lambda}_2$.

Computing the derivative for a terminal node constant value $c_t^{[m]}$ yields

$$\frac{\partial \mathcal{R}_{\text{reg}}^{[m]}}{\partial c_t^{[m]}} = \left(G_t^{[m]} + \operatorname{sign}\left(c_t^m\right)\lambda_3\right) + \left(H_t^{[m]} + \lambda_2\right)c_t^m.$$

The optimal constants $\hat{c}_1^{[m]},\ldots,\hat{c}_{T^{[m]}}^{[m]}$ can then be calculated as



$$\hat{c}_t^{[m]} = -\frac{t_{\lambda_3}\left(G_t^{[m]}\right)}{H_t^{[m]} + \lambda_2}, t = 1, \dots T^{[m]},$$

with

$$t_{\lambda_3}(x) = egin{cases} x + \lambda_3 & ext{for } x < -\lambda_3 \ 0 & ext{for } |x| \le \lambda_3 \ x - \lambda_3 & ext{for } x > \lambda_3. \end{cases}$$

LOSS MINIMIZATION - SPLIT FINDING

To evaluate the performance of a candidate split that divides the instances in region $R_t^{[m]}$ into a left and right node we use the **risk reduction** achieved by that split:

$$ilde{\mathcal{S}}_{LR} = rac{1}{2} \left[rac{t_{\lambda_3} \left(G_{tL}^{[m]}
ight)^2}{H_{tL}^{[m]} + \lambda_2} + rac{t_{\lambda_3} \left(G_{tR}^{[m]}
ight)^2}{H_{tR}^{[m]} + \lambda_2} - rac{t_{\lambda_3} \left(G_t^{[m]}
ight)^2}{H_t^{[m]} + \lambda_2}
ight] - \lambda_1,$$

where the subscripts *L* and *R* denote the left and right leaves after the split.



LOSS MINIMIZATION - SPLIT FINDING /2

Algorithm (Exact) Algorithm for split finding

- 1: Input I: instance set of current node
- 2: Input p: dimension of feature space
- 3: $gain \leftarrow 0$
- 4: $G \leftarrow \sum_{i \in I} g(\mathbf{x}^{(i)}), H \leftarrow \sum_{i \in I} h(\mathbf{x}^{(i)})$
- 5: for $j = 1 \rightarrow p$ do
- 6: $G_L \leftarrow 0, H_L \leftarrow 0$
- 7: **for** i in sorted(I, by x_i) **do**
- 8: $G_L \leftarrow G_L + g(\mathbf{x}^{(i)}), H_L \leftarrow H_L + h(\mathbf{x}^{(i)})$
- 9: $G_R \leftarrow G G_L, H_R \leftarrow H H_L$
- 10: compute \tilde{S}_{LR}
- 11: end for
- 12: end for
- 13: **Output** Split with maximal \tilde{S}_{LR}

