XGBoost (Extreme Gradient Boosting)

Additive model $f(x)$ is base learner, estimation is $\hat{y}$

$$\hat{y} = \sum_{m=1}^{M} f_m(x), \quad f_m \in F \quad ①$$

The target function

$$\mathcal{L}^{(m)} = \sum_{i=1}^{N} L(y_i, \hat{y}_i^{(m-1)} + f_m(x_i)) + \Omega(f_m) \quad ②$$

where Ω is the regularization term and GBDT doesn't have:

$$\Omega(f) = \gamma J + \frac{1}{2}\lambda \sum_{j=1}^{J} b_j^2$$

where J is the number of leaf nodes and b_j is the value

Ω aims at restricting the complexity of tree, which means decrease the number of leaves and values on the leaves.

b_j larger, learner faster $\Rightarrow$ overfitting.

Second-order taylor expansion of $L^{(m)}$ on $\hat{y}_i^{(m-1)}$

$$L^{(m)} \simeq \sum_{i=1}^{N} [L(y_i, \hat{y}_i^{(m-1)}) - g_i f_m(x_i) + \frac{1}{2} h_i f_m^2(x_i)] + \gamma J + \frac{1}{2}\lambda \sum_{j=1}^{J} b_j^2 \quad ③$$

where $g_i = \frac{\partial L(y_i, \hat{y}_i^{(m-1)})}{\partial \hat{y}^{(m-1)}}$, $h_i = \frac{\partial^2 L(y_i, \hat{y}_i^{(m-1)})}{(\partial \hat{y}^{(m-1)})^2}$

DT splits feature space into different isolated area and each sample belongs to one of the regions. Single DT:

$f(x) = \sum_{j=1}^{J} b_j I(x \in R_j)$. If we sum all samples up, $\sum_{i=1}^{N} f(x_i) = \sum_{j=1}^{J} \sum_{x \in R_j} b_j$

Plug it to ③ and remove the constant term $L(y_i, \hat{y}_i^{(m-1)})$

$$\mathcal{L}^{(m)} = \sum_{i=1}^{N} [g_i f_m(x_i) + \frac{1}{2} h_i f_m^2(x_i)] + \gamma J + \frac{1}{2}\lambda \sum_{j=1}^{J} b_j^2$$

$$= \sum_{j=1}^{J} [\sum_{x_i \in R_j} g_j b_j + \frac{1}{2}(\sum_{x_i \in R_j} h_i + \lambda) b_j^2] + \gamma J$$

$$= \sum_{j=1}^{J} (G_j b_j + \frac{1}{2}(H_j + \lambda) b_j^2) + \gamma J \quad ④$$

where $G_j = \sum_{x_i \in R_j} g_j$, $H_j = \sum_{x_i \in R_j} h_i$

New gain calculation method

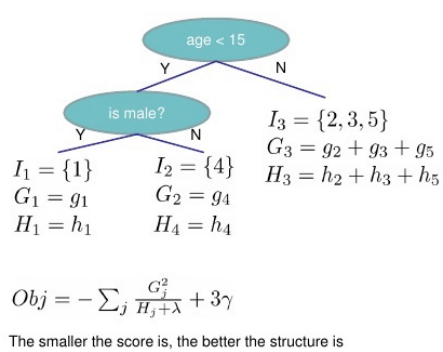
If the structure of a tree is confirmed, compute the deriv of ④ and optimal value:

$$b_j^* = -\frac{G_j}{H_j + \lambda}$$

Plug it to ④:

$$\mathcal{L}^{(m)} = -\frac{1}{2} \sum_{j=1}^{J} \frac{G_j^2}{H_j + \lambda} + \gamma J \quad ⑤$$

It can be thought as score and match different loss func. (can be derivable in first-order and second-order)



$$Gain = \frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda} - \gamma$$

Gain larger ⇒ split the tree.

Pseudocode:

① initialize $f_0(x)$

② for $m=1$ to M:

(a) compute 1st deriv and 2nd deriv of loss func over each sample:

$$g_i = \frac{\partial L(y_i, \hat{y}_i^{(m-1)})}{\partial \hat{y}^{(m-1)}}, \quad h_i = \frac{\partial^2 L(y_i, \hat{y}_i^{(m-1)})}{(\partial \hat{y}^{(m-1)})^2} \quad i=1,2,\cdots,N$$

(b) recursively use Algorithm 2 to generate a DT $f_m(x)$

(c) add DT to models: $\hat{y}_i^{(m)} = \hat{y}_i^{(m-1)} + f_m(x)$

Algorithm 2: Exact Greedy Algorithm for Split Finding

Input: I, instance set of current node

Input: d, feature dimension

Gain = 0

$G = \sum_{i\in I} g_i$, $H = \sum_{i\in I} h_i$

For $k = 1$ **to** m **do:**

$G_L = 0$, $H_L = 0$

For j *in sorted(I, by* x_{jk}*)* **do:**

$G_L = G_L + g_j$, $H_L = H_L + h_j$

$G_R = G - G_L$, $H_R = H - H_L$

score = max(score, $\frac{G_L^2}{H_L+\delta} + \frac{G_R^2}{H_R+\delta} - \frac{G^2}{H+\delta}$)

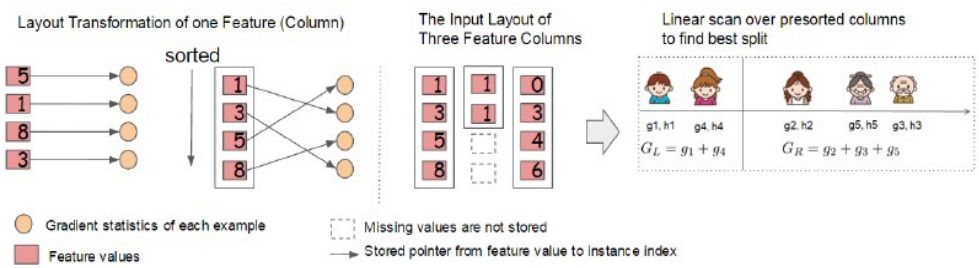
End

End

Output: Split with max score

Column Block for Parallel Learning.

It's hard to split the continuous value for DT. because we have many values. So we can sort the samples by feature values and sequentially compute the Gain to choose the split point. saved as block and each col stored in ascending order.



To improve training speed.

Algorithm 2: Approximate Algorithm for Split Finding

for $k = 1$ ***to*** m **do**
  Propose $S_k = \{s_{k1}, s_{k2}, \cdots s_{kl}\}$ by percentiles on feature k.
  Proposal can be done per tree (global), or per split(local).
end
for $k = 1$ ***to*** m **do**
  $G_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} g_j$
  $H_{kv} \leftarrow= \sum_{j \in \{j | s_{k,v} \geq \mathbf{x}_{jk} > s_{k,v-1}\}} h_j$
end
Follow same step as in previous section to find max score only among proposed splits.