Why the Trees in XGBoost are Better!

Step 1: build a loss function that minimize both Bias and Variance

total loss function

$$\mathcal{L}(\phi) = \sum_i \overbrace{l(y_i, \hat{y}_i)}^{\text{Minimize the Bias}} + \sum_k \overbrace{\Omega(f_k)}^{\text{Minimize the Variance}}$$

Sum over the samples

Sum over the trees

Number of leaves

Prediction at each leaf

where $\Omega(f) = \underbrace{\gamma T}_{\text{Push the number of leaves to be lower}} + \frac{1}{2}\lambda \underbrace{\|w\|^2}_{\text{Push the predictions to be smaller}}$

Step 2: Approximate the loss function with a 2nd order Taylor expansion

At the t^{th} boosting iteration

prediction target

Features

$$\underbrace{l(y_i, \hat{y}_i)}_{\text{sample level loss function}} = l(y_i, \underbrace{\hat{y}_i^{(t-1)}}_{\text{prediction of the target at the (t-1)-th iteration}} + \underbrace{f_t(\mathbf{x}_i)}_{\text{prediction of the new tree}})$$

Second order Taylor approximation

Hessian

$$l(y_i, \hat{y}_i) \simeq l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(\mathbf{x}_i)) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i))$$

Gradient

$$g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$$ Gradient

$$h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}_i^{(t-1)})$$ Hessian

TheAiEdge.io

Example: the cross-entropy loss function

```
loss = y * np.log(y_hat) + (1 - y) * np.log(1 - y_hat)
grad = -y / y_hat + (1 - y) / (1 - y_hat)
hess = y / (y_hat ** 2) + (1 - y) / ((1 - y_hat) ** 2)
```

Step 3: Just use the loss function depending on the new tree

$$\mathcal{L}^{(t)} \simeq \sum_{i=1}^{n}\left[\underbrace{l(y_i, \hat{y}_i^{(t-1)})}_{\text{Does not depends on the new tree}} + \underbrace{g_i f_t(\mathbf{x}_i))}_{\text{Depends on the new tree}} + \underbrace{\frac{1}{2} h_i f_t^2(\mathbf{x}_i)}_{\text{Depends on the new tree}}\right] + \underbrace{\Omega(f_t)}_{\text{Depends on the new tree}}$$

$$\Rightarrow \tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n}\left[g_i f_t(\mathbf{x}_i)) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)\right] + \Omega(f_t)$$

Step 4: Rearrange summation from samples to leaves and leaf instances

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n}\left[g_i f_t(\mathbf{x}_i)) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)\right] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2$$

Sum on samples

prediction at leaf *j*

$$= \sum_{j=1}^{T}\left[(\sum_{i \in I_j} g_i) w_j + \frac{1}{2}(\sum_{i \in I_j} h_i + \lambda) w_j^2\right] + \gamma T$$

Sum on leafs

Sum on instances in each leaf

Step 5: Find optimal weights by setting the derivative to zero

Find the minimum $\partial_{w_j} \tilde{\mathcal{L}}^{(t)} = 0$

Solution $w_j^* = \dfrac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda}$

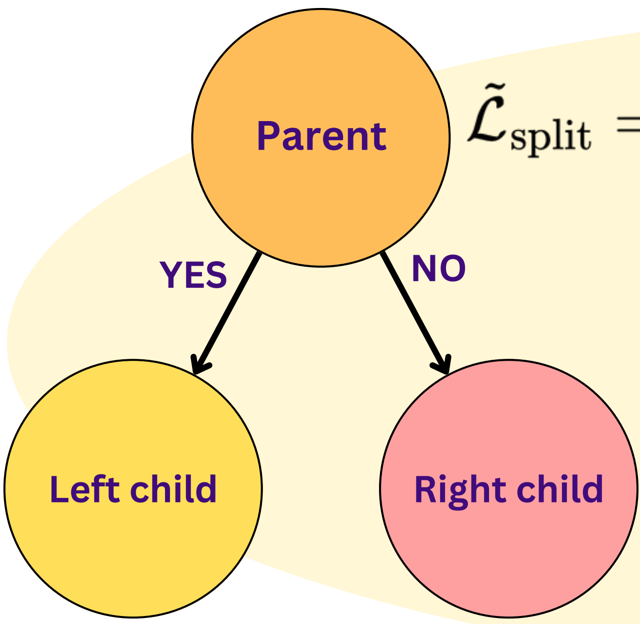
TheAiEdge.io

Step 6: Replace in the loss function

Minimum value of the Loss

$$\tilde{\mathcal{L}}^{(t)} = -\frac{1}{2}\sum_{j=1}^{T}\frac{\left(\sum_{i\in I_j} g_i\right)^2}{\sum_{i\in I_j} h_i + \lambda} + \gamma T$$

The best splits in the trees are the ones that decrease the loss



$$\tilde{\mathcal{L}}_{\text{split}} = \frac{1}{2}\left[\underbrace{\frac{\left(\sum_{i\in I_L} g_i\right)^2}{\sum_{i\in I_L} h_i + \lambda}}_{\text{Left child Loss}} + \underbrace{\frac{\left(\sum_{i\in I_R} g_i\right)^2}{\sum_{i\in I_R} h_i + \lambda}}_{\text{Right child Loss}} - \underbrace{\frac{\left(\sum_{i\in I} g_i\right)^2}{\sum_{i\in I} h_i + \lambda}}_{\text{Parent Loss}}\right] - \gamma$$

Step 7: Iterate and choose the splits that minimize the loss

Training data

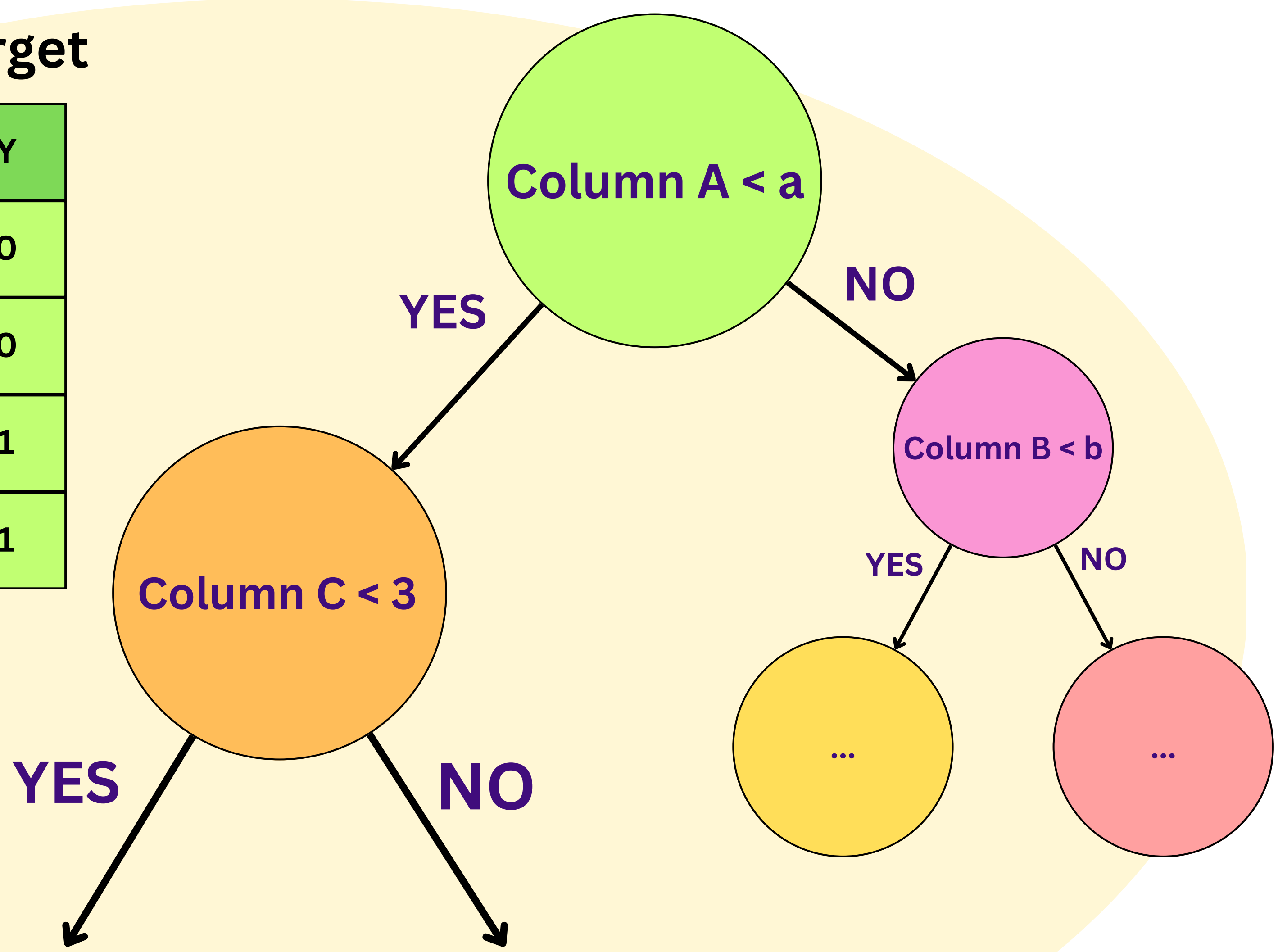
A	B	C	D
		0	
		2	
		3	
		6	

Split 2

Sort column

Target

Y
0
0
1
1



$$\arg\max_{\text{split}} \mathcal{L}_{\text{left}}^{(t)} + \mathcal{L}_{\text{right}}^{(t)} - \mathcal{L}_{\text{parent}}^{(t)}$$