Class Notes on Xgboost

Dr. Subhasis Dasgupta, PhD

May 14, 2021

Gradient boosting machine (GBM) uses a sequence of base models to make the final prediction. Xgboost also builds the final model in the same manner. However, in case of GBM, any base model can be used (not just tree) but Xgboost uses trees only to build a powerful model. This class note is going to give you the basic mathematical background of how Xgboost works. I have used the same notation in this note as there in the original paper so that it becomes easier for the students to compare with the main text. For the full paper, visit `https://arxiv.org/pdf/1603.02754.pdf`.

Let a dataset D contains n samples and m features (or variables).

$D = \{x_i, y_i\}$ for $i = 1, 2, 3, ...$ and $x_i \in R^m$ whereas $y_i \in R^1$

$$\hat{y_i} = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i) \tag{0.1}$$

where K is the number of trees

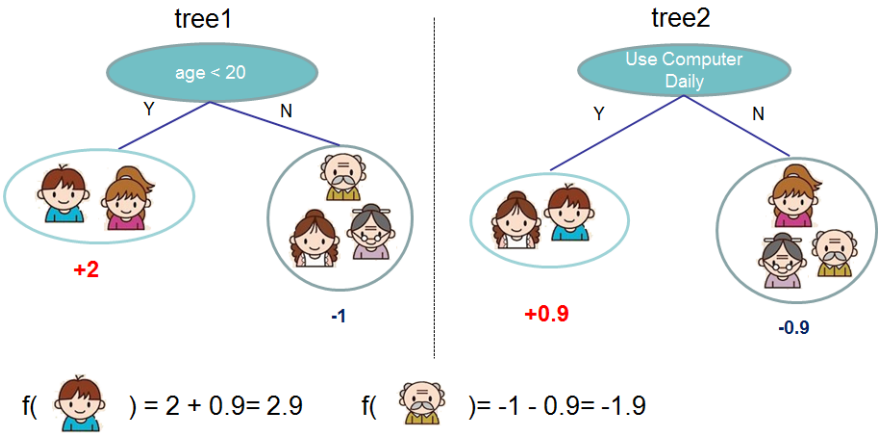
If each tree is to have a structure q, outcome of the tree $q(x)$ is given by the weight associated with the leaf where x is belonging to, i.e.

$$f(x) = w_{q(x)} \tag{0.2}$$

For each tree, there would certain number of leaves associated with it. Let us assume that the number of leaves associated with the tree is T. Xgboost uses regression tree to make the final prediction. The weight $w_{q(x)}$is a continuous numerical score which plays a critical role in prediction. For predicting the outcome of an example (x_i),

- Put the example x_i in the leaf of a tree $q(x)$ having a score $w_{q(x)}$, $j = 1, 2, 3, ...$ number of leaves in q
- Add the scores of the corresponding leaves for final prediction

Figure 0.1: Example of prediction process of boosted tree



As shown in figure 1, the two CART trees, based on their respective rules, will give final prediction based on the weights associated with the terminal leaves. In case of normal GBM with trees, these weights are found out without regularizing them. Hence, in case of regression, the weights are simply the average of all the $x_i$associated with the respective leaf. But, in case of Xgboost, the new tree which is added in the sequence is regularized. The loss function associated with Xgboost is

$$L(\phi) = \sum_i l(y_i, \hat{y}_i) + \sum_k \Omega(f_k) \tag{0.3}$$

$$where\ \Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2 \tag{0.4}$$

Without the regularization portion in the ls function, the algorithm falls back to traditional GBM algorithm. Let us assume that the loss at t^{th}tree is to be optimized and predicted value after $(t-1)$ (previously regularized) trees are already available. In this case, the loss function can be written as:

$$L^{(t)} = \sum_i l\left[y_i, \hat{y}_i^{(t-1)} + f_t(x_i)\right] + \Omega(f_t) \tag{0.5}$$

The above loss function is having a function as a parameter and gradient descent algorithm is incapable of optimizing a function whose parameter is a function. Hence, to apply gradient descent algorithm, the loss function can be approximated as the second order Taylor series as shown below:

$$L^{(t)} \approx \sum_i \left[l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)\right] + \Omega(f_t) \tag{0.6}$$

Here g_i and h_i are the gradient and hessian of the loss function at the i^{th} sample. In the above equation $l(y_i, \hat{y}_i^{(t-1)})$ is constant with respect to the current tree f_t. Hence, from optimization point of view, this term is not at all significant and can be dropped without loss of accuracy. Hence, the new loss function, which is to be optimized is:

$$\tilde{L}^t = \sum_i \Big[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \Big] + \gamma T + \frac{1}{2}\lambda \|w\|^2 \tag{0.7}$$

It is to be understood that prediction of a tree $f(x_i) = w_j$where w_j is the weight associated with the j^{th}leaf of the tree where the sample x_i is belonging to based on the rules of the tree. Moreover, each leaf will have multiple samples and hence a set I_j can be considered which stores indices of the training samples lying in the j^{th} leaf as $I_j = \{i | q(x_i) = j\}$. (Remember that $q(x)$ denotes the structure of the tree generating the rules). With this, equation no (0.7) can be written as:

$$\begin{aligned}
\tilde{L}^t &= \sum_i \Big[g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i) \Big] + \gamma T + \frac{1}{2}\lambda \|w\|^2 \\
&= \sum_{j=1}^{T} \Big[\Big(\sum_{i \in I_j} g_i \Big) w_j + \Big(\frac{1}{2} \sum_{i \in I_j} h_i \Big) w_j^2 \Big] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2 \\
&= \sum_{j=1}^{T} \Big[\Big(\sum_{i \in I_j} g_i \Big) w_j + \frac{1}{2} \Big(\sum_{i \in I_j} h_i + \lambda \Big) w_j^2 \Big] + \gamma T
\end{aligned}$$

The parameter which is to be optimized for lowest loss is found out by differentiating $\tilde{L}^t$with respect to w_j and equating it to zero. Hence,

$$\begin{aligned}
\frac{\partial \tilde{L}^t}{\partial w_j} &= 0 \\
\Big(\sum_{i \in I_j} g_i \Big) + \Big(\sum_{i \in I_j} h_i + \lambda \Big) w_j &= 0
\end{aligned}$$

$$w_j^* = - \frac{\Big(\sum_{i \in I_j} g_i \Big)}{\Big(\sum_{i \in I_j} h_i + \lambda \Big)}$$

This is how weight of each leaf is decided. Putting the value of w_j^* in the loss function we get the lowest loss function as:

$$\tilde{L}^{t*} = -\frac{1}{2}\sum_{j=1}^{T}\frac{\Big(\sum_{i\in I_j} g_i\Big)^2}{\Big(\sum_{i\in I_j} h_i + \lambda\Big)} + \gamma T \tag{0.8}$$

The above expression can be considered as the splitting criteria in Xgboost. Xgboost does binary splitting of nodes and hence, after each split, initial samples (I) there would be left hand side node (with samples I_L)and right hand side node (with samples I_R) such that $I = I_L \cup I_R$. The best split is evaluated for which the following formula gives the maximum loss reduction.

$$L_{split} = \frac{1}{2}\Big[\frac{\Big(\sum_{i\in I_L} g_i\Big)^2}{\Big(\sum_{i\in I_L} h_i + \lambda\Big)} + \frac{\Big(\sum_{i\in I_R} g_i\Big)^2}{\Big(\sum_{i\in I_R} h_i + \lambda\Big)} - \frac{\Big(\sum_{i\in I} g_i\Big)^2}{\Big(\sum_{i\in I} h + \lambda\Big)}\Big] - \gamma \tag{0.9}$$

Xgboost, in essence, does multi-criteria optimization and hence, it is quite complex compared to simple GBM. It also utilizes significant amount of computer resources for calculating gradients and hessian matrices. It has the capability to handle sparse matrices and data sets having missing values. It simply focuses on the non missing entries and put the missing values either into the left node or right node such that the gain is maximized. There are many other aspect of Xgboost which are beyond the scope of this note. Interested students can visit the original paper by visiting `https://arxiv.org/pdf/1603.02754.pdf`