GBT cheat sheet

The term boosting tree corresponds to an aggregation of tree models ("tree") adaptive to each other ("boosting"). Why do we speak of gradient boosting trees?

Theory :

We want to build a model h_M such as :

$$h_M(x) = h_{M-1}(x) + \alpha . \delta_M(x) = \sum_{t=1}^{M} \alpha . \delta_t(x) \quad (1)$$

with the objective of minimizing $E(L(h_M(X), Y))$ with La continuous, differentiable and strictly convex cost function and δ a tree model.

Let $\{(x_i, y_i)\}_{i=1,...,n}$ be realization of the couple (X, Y) and let $h_{m-1}(x)$ be posed.

Knowing that $h_m(x) = h_{m-1}(x) + \alpha . \delta_m(x)$ with α a constant, we are looking for $\delta_m(x)$ a tree model such as :

$$\sum_{i=1}^{n} L(y_i, h_m(x_i)) < \sum_{i=1}^{n} L(y_i, h_{m-1}(x_i)) \quad (2)$$

which is equivalent to :

$$\sum_{i=1}^{n} L(y_i, h_{m-1}(x_i) + \alpha . \delta_m(x)) < \sum_{i=1}^{n} L(y_i, h_{m-1}(x_i)) \quad (3)$$

$x \rightarrow L(y, x)$ being $L2$ and strictly convex, we have (This can be easily demonstrated with a development of taylor to order 1) :

$$L(y, x - h . \nabla_x L(y, x)) < L(y, x) \qquad \forall x \neq x_{min} \text{ and } h \text{ small enough} \quad (4)$$

By replacing$x$ by $h_{m-1}(x_i)$ and h by αwe obtain :

$$\sum_{i=1}^{n} L(y_i, h_{m-1}(x_i) + \alpha . g_i) < \sum_{i=1}^{n} L(y_i, h_{m-1}(x_i)) \quad (5)$$

with

$$g_i = - \nabla_{h_{m-1}(x_i)} L(y_i, h_{m-1}(x_i)) \qquad \text{for } x \neq x_{min} \text{ and } \alpha \text{ small enough.}$$

The g_i are called negative gradient or residuals. As the g_i are dependant from the y_i, we need to approximate them from the x_i observations with a tree model within the meaning of the $L2$ norm to keep true the inequality (5). So we just have to build a regression tree δ_m from the x_i observations to fit the negative gradient g_i.

So we can therefore develop an algorithm where we initialize $h_0(x)$for example by the average of the realizations y_i in a regression problem, and then we fit at each step k the regression trees $\delta_k$on the negative gradients $g_i = -\nabla_{h_{k-1}(x_i)} L(y_i, h_{k-1}(x_i))$. We will choose a fairly small α and the strict convexity property of L ensures the convergence of the algorithm towards a global minimum as long as the inequality (5) can be held valid by the approximation of the negative gradient from the regression trees δ_k.

Algorithm :

Input : α small, $h_0(x) = c$ (we can take $argmin_c \sum_{i=1}^{n} L(y_i, c)$)

for k = 1 **to** m **do** :

- compute the negative gradients $g_i = -\nabla_{h_{k-1}(x_i)} L(y_i, h_{k-1}(x_i))$, $\quad i = 1, ..., n$
- fit $\delta_k(x)$ on the $(g_i)_{i=1,...,n}$
- compute $h_k(x_i) = h_{k-1}(x_i) + \alpha . \delta_k$, $\quad i = 1, ..., n$

done

It is possible to optimize the descent step α by resolving the one dimensional optimization problem :

$$\alpha_k = argmin_\alpha \sum_{i=1}^{n} L(y_i, h_{m-1}(x_i) + \alpha \, . \, \delta_m(x))$$

This is the algorithm describe in wikipedia page :

Input: training set $\{(x_i, y_i)\}_{i=1}^{n}$, a differentiable loss function $L(y, F(x))$, number of iterations M.

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \underset{\gamma}{\arg\min} \sum_{i=1}^{n} L(y_i, \gamma).$$

2. For m = 1 to M:
    1. Compute so-called *pseudo-residuals*:

    $$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

    2. Fit a base learner (or weak learner, e.g. tree) $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$.
    3. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

    $$\gamma_m = \underset{\gamma}{\arg\min} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

    4. Update the model:

    $$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Output $F_M(x)$.

Go further

To avoid overfitting, XGBoost proposes a penalization of trees, and starting from a Taylor expansion of order 2 of (2) we end up with an approximate solution of original trees.