XGBoost / Gradient boosting

Quick note

Abdelwahid Benslimane

wahid.benslimane@gmail.com

XGBoost stands for **Extreme Gradient Boosting**, and as its name indicates it is a method which relies on the gradient boosting of which it improves the performances in terms of computation speed. Among its main features we find parallelization, distributed computing and cache optimization, which allows to process very large datasets.

Let's move on to gradient boosting.

Gradient boosting is a sequential training technique method that consists in adding (iteratively) a new contribution to a model in order to decrease the "residual" with respect to the target variable y_i, i.e. the part that could not be taken into account by the previous model. We are in fact looking for a linear mixture model. Gradient boosting can be used for both classification and regression.

Let's have a look at the structure of the method in more detail.

At iteration m the model $f_{m-1}(x)$ is already built and we seek to add a contribution $h_m(x)$ so that $f_m(x) = f_{m-1}(x) + h_m(x)$ and $f_{m-1}(x_i) + h_m(x_i) = y_i$. We therefore want the values $h_m(x_i)$ approximate well the residuals $y_i - f_{m-1}(x_i)$: the principle is thus to reduce the differences between the current predictions and the values y_i.

Let us assume that we use a quadratic loss; at the beginning of step m we have the following result:

$$L(y, f_{m-1}(x)) = \frac{1}{2}\sum_{i=1}^{n}(y_i - f_{m-1}(x_i))^2.$$

Let's calculate the derivative of the loss function with respect to f_{m-1}:

$$\frac{\delta L(y, f_{m-1}(x))}{\delta f_{m-1}(x)} = -\sum_{i=1}^{n}(y_i - f_{m-1}(xi)).$$

In the case of the quadratic loss, we find that the residuals are given by the inverse of the gradient of the loss function :

residual at step m for the example i from the dataset = $r_{mi} = y_i - f_{m-1}(x_i) = -\frac{\delta L(y_i, f_{m-1}(x_i))}{\delta f_{m-1}(x_i)}, \;\; i = 1, \ldots, n.$

Then we learn a weak classifier $h_m(x)$ (a decision tree typically) on the training set $\{x_i, r_{mi}\}_{i=1}^{n}$ and update the model:

$$f_m(x) = f_{m-1}(x) + \gamma_m h_m(x).$$

The process thus iteratively searches for the model $f(x)$ by a gradient descent that minimizes the loss function $L(\cdot)$. The parameter γ_m is given by :

$$\gamma_m = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, f_{m-1}(x_i) + \gamma h_m(x_i)).$$

The algorithm starts with a constant function $f_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$ and continues during M iterations. Of course, if the loss function is not quadratic, the r_{mi} are no longer the residuals with respect to the target, but the principle remains the same: the weak learner h_m "pulls" the model $f_{m-1}(\cdot)$ in a direction that minimizes the objective function $L(\cdots)$. The process remains the same and in this case, to keep the same terminology, the r_{mi} are called pseudo-residuals.