Abstract

In this work, I synthesize some papers regarding to gradient boosting and related methods like xgboost, a powerful machine learning technique for efficiently solving some classic problems with state of the art performance.

Theory

Consider a training set of N pairs of data \((x_i, y_i)\) for \(1 \leq i \leq N\). Our aim is to find a function \(F\) such that it minimizes

\[
E_{\mathbf{y}} L(y, F(x))
\]

where the loss function \(L\) represents the quality of the prediction.

As it is proposed in [1], we can restructure the space of functions for \(F\) by using a sum of functions \(h(x,a_i)\). Estimating the joint distribution of \(x\) and \(y\) only with the training set, the following approximation of (1)

\[
\sum_{i=1}^{N} L(y_i, M_{m=1} \beta_m h(x_i; a_m))
\]

Now, using the “greedy-stagewise” strategy proposed in [1], which is taking \(F_0(x)\) as the best constant predictor that minimizes \(\sum_{i=1}^{N} L(y_i, \beta)\), and defining for \(m = 1, 2, \ldots, M\)

\[
(\beta_m, a_m) = \arg \min_{\beta_m, a_m} \sum_{i=1}^{N} L(y_i, F_0(x_i) + \beta h(x_i; a_i))
\]

take \(F_m(x) = F_{m-1}(x) + \beta_m h(x, a_m)\). In equation (2), since \(y_i\) are fixed, the predictor vector \(F_0(x), \ldots, F_{m-1}(x)\) determines the value of the loss function. Then, using the gradient descent strategy we want to move in the direction of the gradient vector. This define the gradient boosting algorithm (1). Using trees, we have \(F_m(x) = F_{m-1}(x) + \sum_{j=1}^{2^l} \gamma_j m_{l} \varepsilon_{l}\) where \(\gamma_j m_{l} \varepsilon_{l}\) it implies

\[
F_m(x) = F_{m-1}(x) + \sum_{j=1}^{2^l} \gamma_j m_{l} \varepsilon_{l}
\]

And we have then J functions to add, every one of them can be maximized individually, as we can see in algorithm (2). The stochastic version of this algorithm uses only a proportion of the data for the calculations at each iteration, see [2].

XGBoost is a regularized version of gradient boosting, all the details of its development are in [3]. Let the tree \(\gamma_j m_{l}\) be composed by \(J\) leaves and \(\lambda|w_m|\) to growth. Now, let \(I_j\) \((i \in \{0, 1\})\) and

\[
\gamma_i = \frac{\partial L(y, \mathbf{F}(\mathbf{x}))}{\partial \mathbf{F}(\mathbf{x})} \mathbf{F}(\mathbf{x}) = \mathbf{F}_{\mathbf{m}-1}(\mathbf{x})
\]

\[
H_l = \sum_{j=1}^{2^l} \gamma_j m_{l} \varepsilon_{l}
\]

\[
G_l = \sum_{j=1}^{2^l} \gamma_j m_{l} \varepsilon_{l}
\]

\[
\gamma_j m_{l} \varepsilon_{l}
\]

Applications

From [4]

\[
L_{\text{rank}}(\mathbf{y}_0, \mathbf{F}(\mathbf{x})) = \sum_{i=1}^{N} (F(x_i) - y_i)
\]

Pairwise approach.

\[
L_i^{\text{pair}}(\mathbf{y}_0, \mathbf{F}(\mathbf{x})) = \sum_{i,j \in \text{pairs}} (F(x_i) - F(x_j))
\]

Listwise approach.

\[
L_i^{\text{list}}(\mathbf{F}(\mathbf{x}) = \sum_{i=1}^{N-1} [-F(x_i) + \log \sum_{y \in \text{pairs}} \exp(F(x_i))] = \sum_{i=1}^{N} (F(x_i) - y_i)
\]

XGBoost library

gboost is a library available in R, some useful parameters are:

- booster: The default is gbtree, boosting using trees.
- eta: It’s the shrinkage rate \(\eta\) (or \(\nu\) in notation of 2).
- gamma: Penalization for the number of leaves.
- lambda: Penalization of the sum of weights of the leafs \(\lambda\).
- subsample: Proportion of the data used for adjusting the model at each iteration. If subsample is less than 1 we have stochastic gradient boosting.

Some loss functions are:

- reg: squarederror for Least-squares regression
- binary:logistic obtains the probabilities for the two-class logistic regression and classification
- multi: softmax for multiclass classification. It returns the most probable class.
- multi:softprob similar to softmax, it returns the matrix of probabilities.
- rank:pairwise for ranking using the pairwise approach.

As referred in [5] xgboost can automatically do parallel computation on Windows and Linux and supports customized objective functions. A full list of parameters can be found in [6].

Applications

From [3]

Least-squares regression:

\[
L(y, F) = \frac{(y - F)^2}{2}
\]

Least-squares deviation regression:

\[
L(y, F) = \frac{|y - F|}{2}
\]

L1 regression:

\[
L(y, F) = \frac{1}{2} (y - F)^2 + \frac{1}{2} \sum_{i=1}^{N} (|y_i - F_i| - \delta)
\]

Two-class logistic regression and classification:

\[
L(y, F) = \max(1 + e^{-yF}, 0)
\]

Multi-class logistic regression and classification:

\[
L(y, F(x)) = -\sum_{k=1}^{K} y_k \log p_k(x)
\]

References


Last accessed July 30, 2019

See full report (in spanish) at: https://www.angeldomingo.com