



Developer Guide for Intel(R) Data Analytics Acceleration Library

Developer Guide

Version: 2020
Last Updated: 07/15/2020
Public Content

[Download as PDF](#)



Contents

Developer Guide for Intel® Data Analytics Acceleration Library 2020 Update 2

Introduction

Data Management

Algorithms

Algorithm Input

Algorithm Output



Details

Given n feature vectors $X = \{x_1 = (x_{11}, \dots, x_{1p}), \dots, x_n = (x_{n1}, \dots, x_{np})\}$ of np -dimensional feature vectors and n responses $Y = \{y_1, \dots, y_n\}$, the problem is to build a gradient boosted trees classification or regression model.

The tree ensemble model uses M additive functions to predict the output

$$\hat{y}_i = f(x) = \sum_{k=1}^M f_k(x_i), f_k \in F$$

where $F = \{f(x) = w_{q(x)}, q: R^p \rightarrow T, w \in R^T\}$ is the space of regression trees, T is the number of leaves in the tree, w is a leaf weights vector, w_i is a score on i -th leaf. $q(x)$ represents the structure of each tree that maps an observation to the corresponding leaf index.

Training procedure is an iterative functional gradient descent algorithm which minimizes objective function over function space by iteratively choosing a function (regression tree) that points in the negative gradient direction. The objective function is

$$L(f) = \sum_{i=1}^n l(y_i, f(x_i)) + \sum_{k=1}^M \Omega(f_k)$$

where $l(y_i, f(x_i))$ is a differentiable convex loss function that measures the difference between the prediction $f(x_i)$ and the response y_i , and $\Omega(f) = \gamma T + \frac{1}{2} \lambda \|w\|^2$ is a regularization term that penalizes the complexity of the model defined by the number of leaves T and the L2 norm of the weights $\|w\|^2$ for each tree, γ and λ are regularization parameters.

Training Stage

Library uses the second-order approximation of objective function

$$L^{(k)}(f) \approx \sum_{i=1}^n \left(g_i f_k(x_i) + \frac{1}{2} h_i f_k^2(x_i) \right) + \Omega(f_k), \text{ where } g_i = \frac{\partial l(y_i, \hat{y}_i^{(k-1)})}{\partial \hat{y}_i^{(k-1)}}, h_i = \frac{\partial^2 l(y_i, \hat{y}_i^{(k-1)})}{\partial^2 \hat{y}_i^{(k-1)}} \text{ and}$$

following algorithmic framework for the training stage.

Let $S = (X, Y)$ be the set of observations. Given the training parameters, such as the number of iterations M , loss function $l(f)$, regression tree training parameters, regularization parameters γ and λ , shrinkage (learning rate) parameter θ , the algorithm does the following:

- Find an initial guess $\hat{y}_i^{(0)}, i = 1, \dots, n$
 - For $k = 1, \dots, M$:
 - Update g_i and $h_i, i = 1, \dots, n$
 - Grow a regression tree $f_k \in F$ that minimizes the objective function $-\frac{1}{2} \sum_{j=1}^T \frac{G_j^2}{H_j + \lambda} + \gamma T$,
- where $G_j = \sum_{i \in I_j} g_i, H_j = \sum_{i \in I_j} h_i, I_j = \{i \mid q(x_i) = j\}, j = 1, \dots, T$

For general information on using the Model Builder class, see Training and Prediction

USA (English)   

. For details on using the Model Builder class for Gradient Boosted Trees Classification, see Usage of training alternative

Prediction Stage

Given a gradient boosted trees model and vectors x_1, \dots, x_r , the problem is to calculate the responses for those vectors. To solve the problem for each given query vector x_i , the algorithm finds the leaf node in a tree in the ensemble which gives the response by that tree. Resulting response is based on an aggregation of responses from all trees in the ensemble. For detailed definition, see description of a specific algorithm.

Split Calculation Mode

The library supports two split calculation modes:

- *exact* - all possible split values are examined when searching for the best split for a feature.
- *inexact* - continuous features are bucketed into discrete bins and the possible splits are restricted by the buckets borders only.

Variable Importance

There are five main types of variable importance measures:

- **Weight** represents the number of nodes that split samples by the given feature:

$$\text{weight}_i = \sum_{j=0}^{n-1} \sum_{k=0}^{m(j)-1} [\text{featureSplit}_{jk} = i]$$

where featureSplit_{jk} is the number of features that is used to split samples in the k^{th} node of the j^{th} tree

- **Total cover** represents the number of samples passing through a node that splits the entire set by the given feature:

$$\text{totalCover}_i = \sum_{j=0}^{n-1} \sum_{k=0}^{m(j)-1} [\text{featureSplit}_{jk} = i] \times \text{samples}_{jk}$$

where

- featureSplit_{jk} is the number of features used to split samples in the k^{th} node of the j^{th} tree
- samples_{jk} is the number of samples passing through the k^{th} node of the j^{th} tree
- **Cover** represents the average number of samples passing through one split of the given feature:
$$\text{cover}_i = \frac{\text{totalCover}_i}{\text{weight}_i}$$
- **Total gain** is the reduction of the loss function of the given feature in the entire model:



- Assign an optimal weight $w_j^* = \frac{G_j}{H_j + \lambda}$ to the leaf $j, j = 1, \dots, J$
- Apply shrinkage parameter θ to the tree leafs and add the tree to the model
- Update $\hat{y}_i^{(k)}$

The algorithm for growing the tree:

- Generate a bootstrap training set if required (stochastic gradient boosting) as follows: select randomly without replacement $N = f * n$ observations, where f is a fraction of observations used for training of one tree.
- Start from the tree with depth 0.
- For each leaf node in the tree:
 - Choose a subset of feature for split finding if required (stochastic gradient boosting).
 - Find the best split that maximizes the gain:

$$\frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda} - \frac{(G_L + G_R)^2}{H_L + H_R + \lambda} - \gamma$$

- Stop when a termination criterion is met.

For more details, see [Chen2016

].

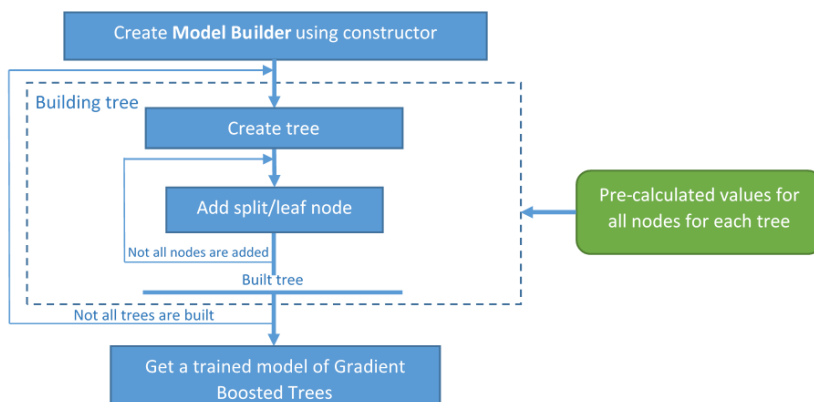
The library supports the following termination criteria when growing the tree:

- *Minimal number of observations in a leaf node.* Node t is not processed if the subset of observations is smaller than the predefined value. Splits that produce nodes with the number of observations smaller than that value are not allowed.
- *Maximal tree depth.* Node t is not processed, if its depth in the tree reached the predefined value.
- *Minimal split loss.* Node t is not processed, if the best possible split is smaller than parameter γ .

Training Alternative

If you already have a set of precomputed values for nodes in each tree, you can use the Model Builder class to get a trained Intel DAAL Gradient Boosted Trees Classification model based on the external model you have.

The following schema illustrates the use of the Model Builder class for Gradient Boosted Trees Classification:



$$\text{totalGain}_i = \sum_{j=0}^{n-1} \sum_{k=0}^{m(j)-1} \left[\text{featureSplit}_{jk} = i \right] \times \Delta \text{impurity}_{jk}$$

where

- featureSplit_{jk} is the number of features used to split samples in the k^{th} node of the j^{th} tree
- $\Delta \text{impurity}_{jk} = \text{impurityLeft}_{jk} + \text{impurityRight}_{jk} - \text{impurity}_{jk}$
- impurityLeft_{jk} is impurity of left child of the k^{th} node of the j^{th} tree
- $\text{impurityRight}_{jk}$ is impurity of right child of the k^{th} node of the j^{th} tree
- impurity_{jk} is impurity of the k^{th} node of the j^{th} tree
- **Gain** is the reduction of the loss function by one split for the given feature:

$$\text{gain}_i = \frac{\text{totalGain}_i}{\text{weight}_i}$$

Gradient Boosted Trees

Batch Processing

Product and Performance Information

¹ Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice. Notice revision #20110804




Company Information

Our Commitment

Communities

Investor Relations

 Contact Us

USA (English)   

Newsroom

Jobs



© Intel Corporation

[Terms of Use](#)

[*Trademarks](#)

[Privacy](#)

[Cookies](#)

[Supply Chain Transparency](#)

[Site Map](#)

