

# HARP DAAL Update

## Distributed Gradient Boosting Tree

Bo Peng

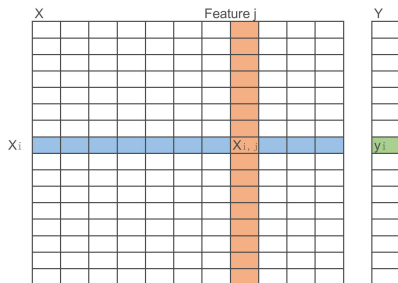
Digital Science Center  
Indiana University

*[pengb@indiana.edu](mailto:pengb@indiana.edu)*

August 22, 2018

- 1 Background
  - GBT Algorithm
- 2 Parallel and Distributed Implementation
  - I. Shared-Memory
  - II. Distributed
- 3 Future Work
  - Topics for the Next Step

# Regression

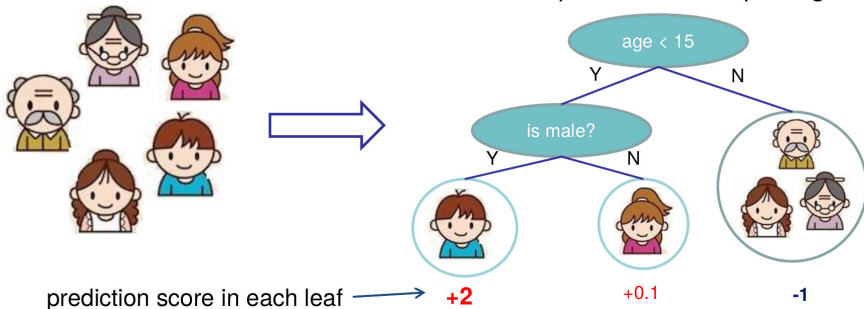


- Problem: find  $\hat{y}_i = \phi(x_i)$  that minimize regularized objective  $\mathcal{L}(\phi) = \sum_{i=1}^n \ell(\hat{y}_i, y_i) + \Omega(\phi)$
- Loss function, for example  $\ell(\hat{y}_i, y_i) = (\hat{y}_i - y_i)^2$
- Data:  
Input:  $n$  samples  $x_i$  each as a  $m$  dimensional vector,  
Response:  $n$  responses  $y_i$

# Regression Tree

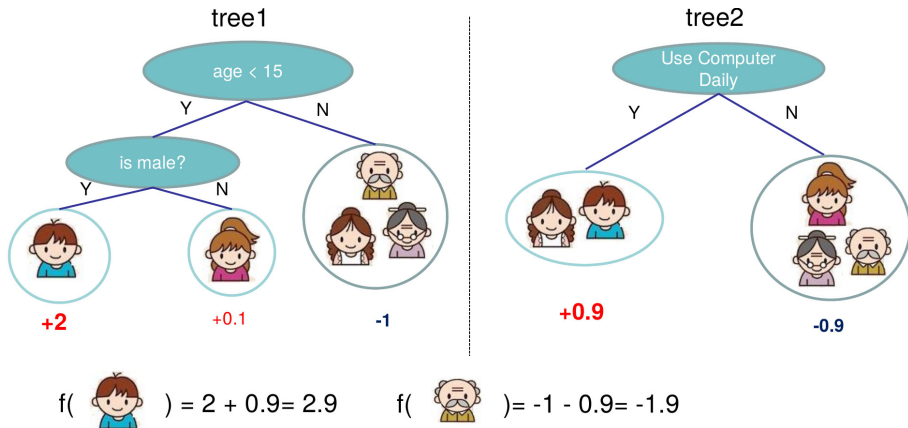
Input: age, gender, occupation, ...

Does the person like computer games



- Regression Tree:  $f(x) = w_{q(x)}$  where  $q : \mathbb{R}^m \rightarrow T$   
 $q$  is the tree structure, map  $x$  to leaf nodes ( $T$  is number of leaves)  
 $w$  is leaf weights
- How to learn  $f(x)$  ?  
Greedy algorithm, **findBestSplit()** according to a **score function** at each internal node

# Boosting



- Boosting:  $\hat{y}_i = \phi(x_i) = \sum_{k=1}^K f_k(x_i)$
- additive model: prediction is the sum score of all the trees

# GBT: Gradient Boosting Tree

- stage-wise adding new tree:

$$\mathcal{L}^{(t)} = \sum_{i=1}^n \ell(\hat{y}_i^{(t-1)} + f_t(x_i), y_i) + \Omega(f_t)$$

- by second order approximation

$$\mathcal{L}^{(t)} \approx \sum_{i=1}^n [\ell(\hat{y}_i^{(t-1)}, y_i) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)] + \Omega(f_t)$$

where;

$$g_i = \partial_{\hat{y}_i^{(t-1)}} \ell(\hat{y}_i^{(t-1)}, y_i)$$

$$h_i = \partial_{\hat{y}_i^{(t-1)}}^2 \ell(\hat{y}_i^{(t-1)}, y_i)$$

- optimal weight  $w_j^*$  and value for leaf  $j$  should be

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

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

- Example

- square loss:  $\ell(\hat{y}_i, y_i) = (\hat{y}_i - y_i)^2$   
 $g_i = \partial_{\hat{y}_i^{(t-1)}} \ell(\hat{y}_i^{(t-1)}, y_i) = 2(\hat{y}_i^{(t-1)} - y_i)$   
 $h_i = 2$

- intuitively optimal weight  $w_j^*$  is just a kind of **average residual**

$$w_j^* = - \frac{\sum_{i \in I_j} g_i}{\sum_{i \in I_j} h_i + \lambda} = \frac{2}{2 + \lambda} \frac{(y_i - \hat{y}_i^{(t-1)})}{|I_j|}$$

- Important data structure on leaf  $j$

- $G_j = \sum_{i \in I_j} g_i$
- $H_j = \sum_{i \in I_j} h_i$
- $S(L, R) = \frac{G_{L_L}^2}{H_{L_L} + \lambda} + \frac{G_{L_R}^2}{H_{L_R} + \lambda}$  score function to get best split (impurity, more general the loss reduction)

# GBT Algorithm

**Algorithm 1:** Gradient Boosted Regression Tree

**input** : dataset  $D = (x_i, y_i)_{i=1}^n$ ,  
parameter  $\lambda, \alpha, m$

**output:**  $m$  trees  $f(x) = w_q(x)$

**begin**

Initialize()

**for**  $t=1$  to  $m$  **do**

    // BuildTree( $\{(x_i, y_i)\}$ )

$(w, q) =$

$\arg \min_{f_t} \sum_{i=1}^n [\ell(\hat{y}_i^{(t-1)} + \alpha f_t(x_i), y_i)] + \Omega(f_t)$

    // additive update

$f_t(x) = f_{t-1}(x) + \alpha f_t(x)$

**Algorithm 2:** Greedy Split Finding

**input** :  $I$ , instance set of current node;  $d$ , feature dimension

**output:** split at the position with max score

1 **begin**

2      $score = 0$

3      $G = \sum_{i \in I} g_i$

4      $H = \sum_{i \in I} h_i$

5     **for**  $k = 1$  to  $d$  **do**

6          $G_L = 0; H_L = 0$

7         **for**  $j$  in sorted( $I$ , by  $X_{jk}$ ) **do**

8              $G_L = G_L + g_j; H_L = H_L + h_j$

9              $G_R = G - G_L; H_R = H - H_L$

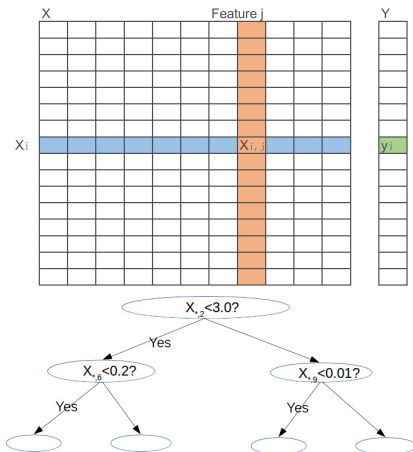
10              $score = \max(score, \frac{G_L^2}{H_L + \lambda} + \frac{G_R^2}{H_R + \lambda})$



- basic idea on
  - boosting; function approximation target on the residual (gradient in general)
  - decision tree; weak learner easy to understand and build
- a general framework
  - supporting wide range of loss functions and regularizations
  - the score function to build tree is derived from a wide range of objective functions
- features
  - auto feature selection (kind of)
  - easy to deal with missing values, category values

- 1 Background
  - GBT Algorithm
- 2 Parallel and Distributed Implementation
  - I. Shared-Memory
  - II. Distributed
- 3 Future Work
  - Topics for the Next Step

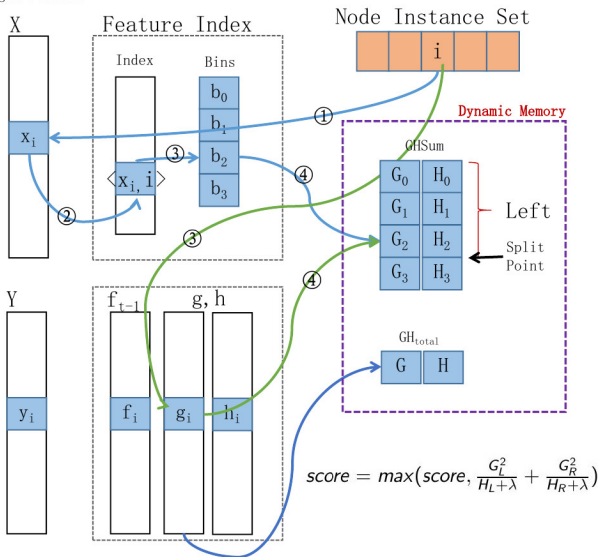
# Parallelism in a Shared-Memory Setting



- Boosting is a sequential process, building trees one by one. (multiclass categorization have multiple boosting processes)
- Parallelism in building one tree
  - **parallelFeatures:** `findBestSplit()` works on features independently
  - **parallelNodes:** same level of nodes in a tree works independently
  - **vectorization:** lots of  $\sum$  operations,  $G_j, H_j$

# Issues - Non-continuous Mem Access

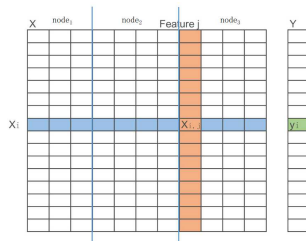
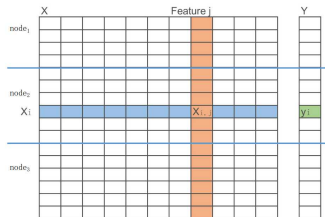
Single Feature



- **Bins** is a split candidates proposal  
 $B = b_0, b_1, \dots, b_k$ ,  
in general  
 $|B| \ll n$
- with Bins, the findBestSplit timecomplexity drops from  $\mathcal{O}(n \log n)$  to  $\mathcal{O}(n \log b)$

- 1 Background
  - GBT Algorithm
- 2 Parallel and Distributed Implementation
  - I. Shared-Memory
  - II. Distributed
- 3 Future Work
  - Topics for the Next Step

# Data Partition



- Partition by rows (samples)
  - need global communication to build **Bins**, once if use global static Bins.
  - need global communication to build **GHSum** for each feature in `findBestSplit()` → `allreduce(GHSum)`
- Partition by columns (features)
  - **Bins** and **GHSum** are all local, no communication
  - need global communication to select the best feature in `findBestSplit()` → `allreduce(maxscore)`

# Issue - Overhead of Synchronization & Communication

- Overhead of communication
  - well designed Bins works as model compression, reducing the data volume of communication
  - pipeline
- Overhead of synchronization
  - load imbalance is the killer of synchronization operation
  - allreduce give chance to [model rotation](#)
  - stochastic GBT give chance to [timer control](#)

# Outline

- 1 Background
  - GBT Algorithm
- 2 Parallel and Distributed Implementation
  - I. Shared-Memory
  - II. Distributed
- 3 Future Work
  - Topics for the Next Step



# Topics for the Next Step

- memory access optimization in shared memory system
- load imbalance issue in distributed system
- different Bins design
- dataset with sparsity and high dimensionality

# References



T. Chen and C. Guestrin, Xgboost: A scalable tree boosting system, in Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, 2016, pp. 785–794..



S. Tyree, K. Q. Weinberger, K. Agrawal, and J. Paykin, Parallel boosted regression trees for web search ranking, in Proceedings of the 20th international conference on World wide web, 2011, pp. 387–396.

# The End