Boosting II

Extreme Gradient Boosting

Introduction

Boosting

- Class of ensemble methods which combine **sequential** prediction models
- Adaptive approach with focus on "difficult observations"
- Different flavors exist
  - AdaBoost
  - Gradient Boosting Machines (GBM)
  - ...
- Can be applied to different (weak) base learners
  - Boosting trees
  - ...

Gradient Boosting

Starting point: A decision tree...

$$T(x;\Theta) = \sum_{j=1}^{J} \gamma_j I(x \in R_j)$$

with tree parameters $\Theta = \{R_j, \gamma_j\}$.

Goal of tree construction:

$$\Theta = \arg\min_{\Theta} \sum_{j=1}^{J} \sum_{x_i \in R_j} L(y_i, \gamma_i)$$

$\rightarrow$ Boosting: Minimizing loss over a *sequence* of trees

Gradient Boosting

Boosting: Reduce the loss given $f_{m-1}(x_i)$

$$\Theta = \arg\min_{\Theta_m} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + T(x_i; \Theta_m))$$

$\rightarrow$ Focus on pseudo-residuals for the ith obs on iteration m

$$r_{im} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f=f_{m-1}}$$

$\rightarrow$ Fit regression tree to pseudo-residuals/ negative gradients

$$\tilde{\Theta}_m = \arg\min_{\Theta} \sum_{i=1}^{N} (-g_{im} - T(x_i; \Theta))^2$$

Boosting for regression

Algorithm 1: Gradient Boosting for regression

Set number of trees M;
Set interaction depth D;
Set shrinkage parameter λ;
Use $\bar{y}$ as initial prediction;
for $m = 1$ *to* M **do**
    compute residuals based on current predictions;
    assign data to root node, using the residuals as the outcome;
    while *current tree depth* $< D$ **do**
        tree growing process;
    end
    compute the predicted values of the current tree;
    add the shrinked new predictions to the previous predicted values;
end

Extreme Gradient Boosting

XGBoost

- Widely used (and competitive) in ML challenges
- Introduces regularization and a modified splitting criterion
- Scalable due to various algorithmic optimizations
  - Sparsity-aware split finding
  - Multicore processing
- Trees as base learners (`xgbtree`), or linear models (`xgblinear`)

→ Chen and Guestrin 2016

Extreme Gradient Boosting

The XGBoost ensemble

$$\hat{y}_i = \sum_{k=1}^{K} f_k(x_i), f_k \in \mathcal{F}$$

with K functions f of the set of all possible trees $\mathcal{F}$

Regularized objective function

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} L(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)$$

$$\Omega(f) = \gamma T + \frac{1}{2}\lambda||w||^2$$

with number of leaves T, vector of leaf scores w, regularization parameters γ, λ

Extreme Gradient Boosting

Objective of a sequence of XGBoost trees

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

Optimization via second-order approximation

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} (g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)) + \Omega(f_t)$$

with first and second order gradient statistics g_i, h_i

$\rightarrow$ Tree quality score

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

Extreme Gradient Boosting

Objective of a sequence of XGBoost trees

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

Optimization via second-order approximation

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} (g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)) + \Omega(f_t)$$

with first and second order gradient statistics g_i, h_i

$\rightarrow$ Tree quality score

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

Extreme Gradient Boosting

Objective of a sequence of XGBoost trees

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

Optimization via second-order approximation

$$\tilde{\mathcal{L}}^{(t)} = \sum_{i=1}^{n} (g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)) + \Omega(f_t)$$

with first and second order gradient statistics g_i, h_i

$\rightarrow$ Tree quality score

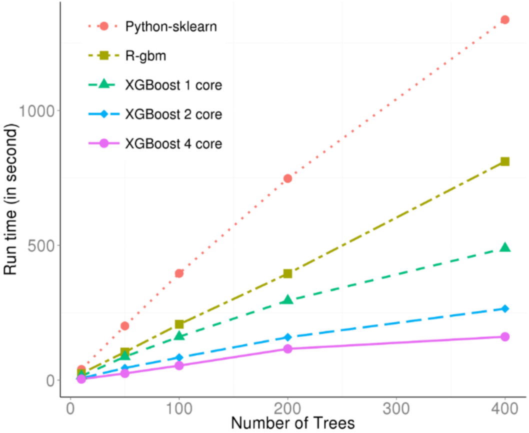
$$\tilde{\mathcal{L}}^{(t)}(q) = -\frac{1}{2} \sum_{j=1}^{T} \frac{(\sum_{i \in I_j} g_i)^2}{\sum_{i \in I_j} h_i + \lambda} + \gamma T$$

XGBoost Tuning

- `nrounds`
  - Number of trees
- `eta`
  - Shrinkage, learning rate
- `max_depth`
  - Maximum tree depth
- `subsample`
  - Subsample ratio of observations
- `gamma`
  - Minimum loss reduction required to split
- `colsample_bytree`
  - Subsample ratio of columns
- https://xgboost.readthedocs.io/en/latest/parameter.html

Scalability

Figure: Speed comparison[1]



[1] https://www.r-bloggers.com/parallel-computation-with-r-and-xgboost/