Gradient Boosted Decision Tree

JrPhy

Introduction

- We’ve known the Adaboost and Random Forest(RF). Adaboost is a algorithm with weak base learning algorithm + **optimal re-weighting factor** + linear aggregation α. And RF is a algorithm uses decision tree by bootstrapping and form many decision trees.
- Now we want to combine them, but RF just aggregate those trees, without weighting factor, so here we need a **weighted** decision tree.

Adaboost decision tree

- Both adaboost and decision tree needs bootstrapping. In the BAGging algorithm, the weights represents how many bootstrap-sampled copies in the dataset. In the randomized base algorithm, the weights represents the proportion. So we don’t modify the decision tree algorithm.
- So that we can tell it the information of weights by sampling.
- So Adaboost decision tree = Adaboost + sampling + decision tree

Adaboost decision tree

- In adaboost, the weight is $\alpha = \ln\sqrt{\frac{(1-t)}{t}}$, t is weighted error rate.

- If the decision tree is **fully-grown**, then E_{in} = 0, so $E^{u}{}_{in}$ = 0, too, so that t = 0, then a is infinity,
- So we don't need the fully-grown tree, and just needs **some** sample instead of all data.
- Adaboost decision tree = Adaboost + sampling + pruned decision tree
- A pruned method is to limit the height of the tree.

Weights of adaboost

- In adaboost, the next score is from the previous.

$$u_n^{(t+1)} = \begin{cases} u_n^{(t)} \times \sqrt{\left(1-t\right)/t}, \text{correct} \\ u_n^{(t)} / \sqrt{\left(1-t\right)/t}, \text{incorrect} \end{cases} \rightarrow \quad \alpha = \ln\sqrt{\frac{(1-t)}{t}} \rightarrow \sqrt{\frac{(1-t)}{t}} = e^{\alpha}$$

$$= u_n^{(t)} \sqrt{\left(1-t\right)/t}^{\,-y_n g_i(x_n)} = u_n^{(t)} \exp(-y_n \alpha_i g_i(x_n))$$

$$u_n^{(N+1)} = u_n^{(1)} \prod_{i=1}^{N} \exp(-y_n \alpha_i g_i(x_n)) = \frac{1}{N} \exp(-y_n \sum_{i=1}^{N} \alpha_i g_i(x_n)) = \frac{1}{N} \exp(-y_n \times \text{voting score})$$

- $G(x_n) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i g_i(x_n)\right)$ is the voting score.

Weights of adaboost

- The voting score is like the margin of hard-margin SVM, we want the large margin, means the positive and large.
- Voting score is positive and large → exp(–y_n × voting score) small
- → $u^{(N+1)}n$ small
- So that here we want to minimize the

$$\sum_{n=1}^{M} u_n^{(N+1)} = \frac{1}{N}\sum_{n=1}^{M} \exp\left(-y_n \sum_{i=1}^{N} \alpha_i g_i(x_i)\right)$$

- Here we use gradient descent algorithm to find the minimum.

Weights of adaboost

- Gradient descent: $f(u+\eta v) \sim f(u) + \eta v f(u)$, $\|v\| = 1$. the following $v = h(x)$

$$\frac{1}{N}\sum_{n=1}^{M}\exp\left(-y_n\sum_{i=1}^{N}\alpha_i g_i(x_i)\right) \to \boxed{\frac{1}{N}\sum_{n=1}^{M}\exp\left(-y_n\left(\sum_{i=1}^{N}\alpha_i g_i(x_i)\right.\right.}\left.\left.+\eta v\right)\right)$$

- $u^{(N)}{}_n$

$$= \sum_{n=1}^{M} u_n^{(N)} \exp(-y_n \eta v) \approx \sum_{n=1}^{M} u_n^{(N)} (1 - y_n \eta v)$$

-

taylor expansion

$$= \sum_{n=1}^{M} u_n^{(N)} + \sum_{n=1}^{M} u_n^{(N)} (-y_n \eta v)$$

Optimize

$$E_{in}^{u} = \sum_{n=1}^{M} u_n^{(N)} + \eta \sum_{n=1}^{M} u_n^{(N)} \left(-y_n h(x_n)\right)$$

- So that we want to minimize the last term.

$$\sum_{n=1}^{M} u_n^{(N)} \left(-y_n \eta h(x_n)\right) = \sum_{n=1}^{M} u_n^{(N)} \times \begin{cases} 1, & y_n = h(x_n) \\ -1, & y_n \neq h(x_n) \end{cases}$$

$$= -\sum_{n=1}^{M} u_n^{(N)} + \sum_{n=1}^{M} u_n^{(N)} \times \begin{cases} 0, & y_n = h(x_n) \\ 2, & y_n \neq h(x_n) \end{cases} = -\sum_{n=1}^{M} u_n^{(N)} + 2E_{in}^{u}(h)N$$

- In adaboost, the base algorithm minimizes E_{in}. It finds $h = g_i$ for gradient descent.

Optimize

- So in adaboost, the error is $E_{ada} = \sum_{n=1}^{M} u_n^{(N)} \exp\left(-y_n \eta g_i(x_n)\right)$
- In the summation, we know
- For correct: $y_n = g_i(x_n) \rightarrow u^{(N)}{}_{\mathrm{n}}\exp(-\eta)$
- For incorrect: $y_n \neq g_i(x_n) \rightarrow u^{(N)}{}_{\mathrm{n}}\exp(+\eta)$

$$E_{ada} = \sum_{n=1}^{M} u_n^{(N)} \left(\left(1-t\right)\exp(-\eta) + t\exp(+\eta)\right)$$

- Then take differentiate on η, we can find a optimize $\eta*$ in gradient descent. And

$$\eta^* = \ln\sqrt{\frac{1-t}{t}} = \alpha$$

Optimize

- So in adaboost, it finds g_i to approximate h, then fix y_n and h to find a optimize $\eta*$, and apply it in gradient descent, this method of gradient descent is called **steepest gradient descent**.
- The original gradient descent tune η by user, so you may try many η to find the optimize $\eta*$, but in steepest gradient descent, the $\eta*$ is determined by the condition.
- In the adaboost with binary output, which we minimize is

$$\min_{\eta}\left(\min_{h}\frac{1}{N}\sum_{n=1}^{M}\exp\left(-y_n\left(\sum_{i=1}^{N}\alpha_i g_i\left(x_i\right)+\eta h\left(x_i\right)\right)\right)\right)$$

GradientBoost with square error

- Here we want to use generalized error function, like square error, or every error can solve by gradient descent.

$$\min_{\eta}\left(\min_{h}\frac{1}{N}\sum_{n=1}^{M}err\left(\sum_{i=1}^{N}\alpha_i g_i\left(x_i\right)+\eta h\left(x_i\right),y_n\right)\right)$$

- Here we use square error as example, $err(s, y) = (s - y)^2$

$$\min_{h}\frac{1}{N}\sum_{n=1}^{M}err\left(\sum_{i=1}^{N}\alpha_i g_i\left(x_i\right)+\eta h\left(x_i\right),y_n\right)$$

$$\approx\min_{h}\frac{1}{N}\sum_{n=1}^{M}err(s_n,y_n)+\frac{1}{N}\sum_{i=1}^{N}\eta h\left(x_i\right)\frac{\partial}{\partial s}err(s_n,y_n)\Big|_{s=s_n}$$

GradientBoost with square error

$$\min_h \frac{1}{N}\sum_{n=1}^{M} err\left(\sum_{i=1}^{N}\alpha_i g_i\left(x_i\right)+\eta h\left(x_i\right), y_n\right)$$

$$\approx \min_h\left(\frac{1}{N}\sum_{n=1}^{M} err(s_n, y_n)+\frac{1}{N}\sum_{i=1}^{N}\eta h\left(x_i\right)\frac{\partial}{\partial s} err(s_n, y_n)\Big|_{s=s_n}\right)$$

$$=\min_h\left(\frac{1}{N}\sum_{n=1}^{M} err(s_n, y_n)+\frac{\eta}{N}\sum_{i=1}^{N} h\left(x_i\right)\times 2\left(s_n - y_n\right)\right)$$

- The 1st term is constant, so we just need to minimize the 2nd term. If *h*(*x*) is without constrain, then *h*(*x*) can be negative infinity, so we should set some constrain on *h*(*x*).

GradientBoost with square error

- Here we add $(h(x))^2$ as the constrain

$$\frac{\eta}{N}\sum_{i=1}^{N} h(x_i)\times 2(s_n - y_n) \rightarrow \frac{\eta}{N}\sum_{i=1}^{N}\left(h(x_i)\times 2(s_n - y_n) + \left(h(x_i)\right)^2\right)$$

$$= \frac{\eta}{N}\sum_{i=1}^{N}\left(\left(h(x_i) - (y_n - s_n)\right)^2 + \text{constant}\right)$$

- So we want to solve the square error regression on $\{(x_n, y_n - s_n)\}$, it finds the $g_j = h$ by regression with residuals.

GradientBoost with square error

- After finding h, then find the optimize η.

$$\min_{\eta}\left(\min_{h}\frac{1}{N}\sum_{n=1}^{M}err\left(\sum_{i=1}^{N}\alpha_i g_i\left(x_i\right)+\eta h\left(x_i\right),y_n\right)\right)=\min_{\eta}\left(\frac{1}{N}\sum_{n=1}^{M}\left(s_n-y_n+\eta g_j(x_n)\right)^2\right)$$

$$=\min_{\eta}\left(\frac{1}{N}\sum_{n=1}^{M}\left(-\left(s_n-y_n\right)-\eta g_j(x_n)\right)^2\right)=\min_{\eta}\left(\frac{1}{N}\sum_{n=1}^{M}\left(\left(y_n-s_n\right)-\eta g_j(x_n)\right)^2\right)$$

- it’s a linear regression problem, the target function is $g_j(x_n)$, $(y_n - s_n)$ is the residual, η is the step size in GradientBoost.

Algorithm

Gradient Boosted Decision Tree (GBDT)

$s_1 = s_2 = \ldots = s_N = 0$

for $t = 1, 2, \ldots, T$

1. obtain g_t by $\mathcal{A}(\{(\mathbf{x}_n, y_n - s_n)\})$ where $\mathcal{A}$ is a (squared-error) regression algorithm
—**how about sampled and pruned C&RT?**
2. compute $\alpha_t = \text{OneVarLinearRegression}(\{(g_t(\mathbf{x}_n), y_n - s_n)\})$
3. update $s_n \leftarrow s_n + \alpha_t g_t(\mathbf{x}_n)$

return $G(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t g_t(\mathbf{x})$

Hsuan Tien Lin, mltech/211_handout