Ensemble Methods (Adaboost and GBDT)

Prof.Mingkui Tan

South China University of Technology
Southern Artificial Intelligence Laboratory(SAIL)

November 7, 2017

Content

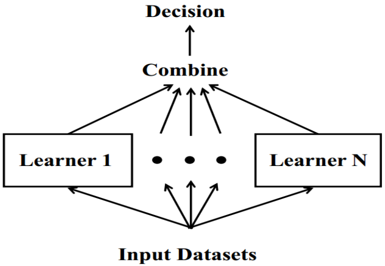
1. Introduction of Boosting Method
2. Adaboost
3. GBDT

Contents

1. Introduction of Boosting Method
2. Adaboost
3. GBDT

Ensemble Learning

- Ensemble learning: Combine numerous weak learners to a strong learner
- Main methods: Boosting, Bagging



Boosting Method

Algorithm 1: Adaboost

Input: $D = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$, **where** $\mathbf{x}_i \in X, y_i \in \{-1, 1\}$
Initialize: Sample weight distribution $D_1 = \frac{1}{n}$

```
Train a base learner h_1(x) with D_1
for m=2,3,...,M do
    Update the sample distribution D_m, to make the wrong
      predictive samples more important
    Train a new base learner h_m(x) with D_m
end
```

Output: $H(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m h_m(\mathbf{x})$

Contents

1 Introduction of Boosting Method

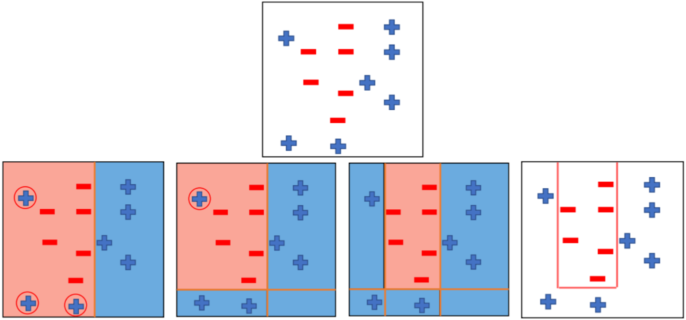
2 Adaboost

3 GBDT

Adaboost

How to train the base leaner?

Make the wrong predictive samples more important, and handle it in next round:



Adaboost

Sample weight updating formula

$$w_{m+1}(i) = \frac{w_m(i)}{z_m} e^{-\alpha_m y_i h_m(\mathbf{x}_i)}$$

$z_m = \sum_{i=1}^{n} w_m(i) e^{-\alpha_m y_i h_m(\mathbf{x}_i)}$ is normalization term, makes $w_m(i)$ become probability distributions

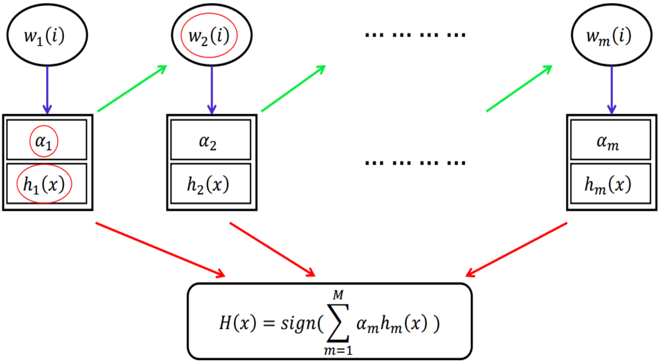
$$w_{m+1}(i) = \begin{cases} \dfrac{w_m(i)}{z_m} e^{-\alpha_m} & \text{for right predictive sample} \\ \dfrac{w_m(i)}{z_m} e^{\alpha_m} & \text{for wrong predictive sample} \end{cases}$$

so in next round, $\frac{w_{wrong}(i)}{w_{right}(i)} = e^{2\alpha_m} = \dfrac{1-\epsilon_m}{\epsilon_m}$ and $\epsilon_m < 0.5$, wrong samples will be more important

Adaboost

How to combine the base learner?

Every iteration generates a new base learner $h_m(\mathbf{x})$ and its importance score α_m



Adaboost

Evaluate the performance of the base learner

- Base learner

$$h_m(\mathbf{x}) : \mathbf{x} \mapsto \{-1, 1\}$$

- Error rate

$$\epsilon_m = p(h_m(\mathbf{x}_i) \neq y_i) = \sum_{i=1}^{n} w_m(i)\mathbb{I}(h_m(\mathbf{x}_i) \neq y_i)$$

ϵ_m¡0.5, or the performance of Adaboost is weaker than random classfication.

Adaboost

Importance score of base learner

Make the base learner with lower ϵ_m more important

$$\alpha_m = \frac{1}{2} \log \frac{1 - \epsilon_m}{\epsilon_m}$$

Adaboost

Additive model

- Final learner

$$H(\mathbf{x}) = \text{sign}(\sum_{m=1}^{M} \alpha_m h_m(\mathbf{x}))$$

Note: $h_m(\mathbf{x}) = \text{sign}(\mathbf{w}^\top \mathbf{x})$ is a nonlinear function, so the Adaboost can deal with nonlinear problem

Algorithm

Algorithm 2: Adaboost

Input: $D = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$, **where** $\mathbf{x}_i \in X, y_i \in \{-1, 1\}$
Initialize: Sample distribution w_m
Base learner: $\mathcal{L}$

```
w_1(i) = 1/n
for m=1,2,...,M do
    h_m(x) = L(D, w_m)
    ε_m = Σ_{i=1}^{n} w_m(i) I(h_m(x_i) ≠ y_i)
    if ε_m > 0.5 then
        break
    end
    α_m = 1/2 log((1 - ε_m)/ε_m)
    w_{m+1}(i) = (w_m(i)/z_m) e^{-α_m y_i h_m(x_i)}, where i = 1, 2, ..., n and
      z_m = Σ_{i=1}^{n} w_m(i) e^{-α_m y_i h_m(x_i)}
end
```

$w_1(i) = \frac{1}{n}$

for m=1,2,...,M **do**

- $h_m(x) = \mathcal{L}(D, w_m)$
- $\epsilon_m = \sum_{i=1}^{n} w_m(i)\mathbb{I}(h_m(\mathbf{x}_i) \neq y_i)$
- **if** $\epsilon_m > 0.5$ **then**
  - **break**
- **end**
- $\alpha_m = \frac{1}{2}\log\frac{1-\epsilon_m}{\epsilon_m}$
- $w_{m+1}(i) = \frac{w_m(i)}{z_m} e^{-\alpha_m y_i h_m(\mathbf{x}_i)}$,**where** $i = 1, 2, ..., n$ **and** $z_m = \sum_{i=1}^{n} w_m(i) e^{-\alpha_m y_i h_m(\mathbf{x}_i)}$

end

Output: $H(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m h_m(\mathbf{x})$

Contents

1. Introduction of Boosting Method
2. Adaboost
3. GBDT

Gradient Boosting Decision Trees

GBDT is a decision tree algorithm with iteration

Example: What is the difference between regression tree and GBDT?

Suppose: There are 4 peoples A, B, C and D, whose age are 14, 16, 24, 26 respectively.

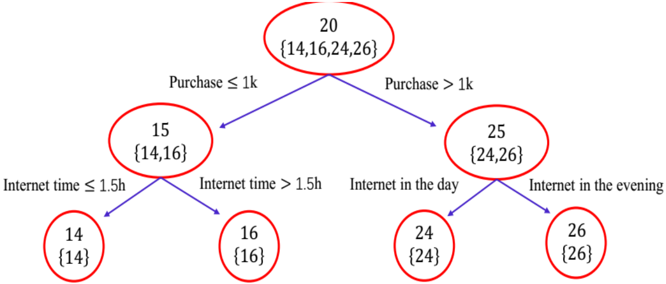
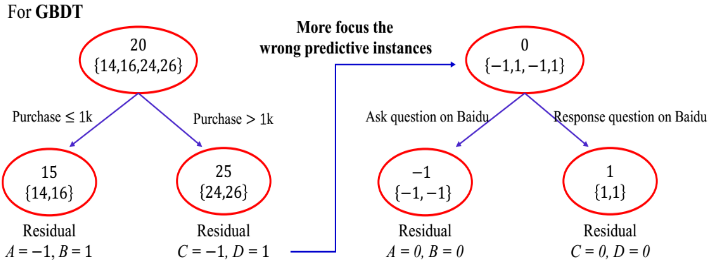


Figure: Single regression tree

Gradient Boosting Decision Trees

- The key of GBDT is that trees learn all the results and residuals of all trees before.
- The residual is the difference of predictive value and real value, so the predictive value is the sum of all results of trees.



- So,
  A=15+(-1)=14 B=15+1=16 C=25+(-1)=24 D=25+1=26

Gradient Boosting Decision Trees

Question

Q1:when results of these two algorithms are same, Why do we choose GBDT?

- The motivation of this algorithm is that every calculation of residual is to increase the weight of wrong predictive samples,and the residual of right predictive sample is zero.
- So in the next iteration, model can concentratively address these wrong predictive samples. Another function is to prevent over fitting.

Gradient Boosting Decision Trees

Question

Q2: Where does this algorithm reflect gradient boosting?

- In algorithm, residual is the gradient descent direction, which is the derivation of mean square error(MSE). Actually, MSE is the loss function of CART regression tree.

Setting	Loss Function	$-\partial L(y_i, f(x_i))/\partial f(x_i)$
Regression	$\frac{1}{2}[y_i - f(x_i)]^2$	$y_i - f(x_i)$
Regression	$\|y_i - f(x_i)\|$	$\text{sign}[y_i - f(x_i)]$
Regression	Huber	$y_i - f(x_i)$ for $\|y_i - f(x_i)\| \leq \delta_m$ $\delta_m \text{sign}[y_i - f(x_i)]$ for $\|y_i - f(x_i)\| > \delta_m$ where $\delta_m = \alpha$th-quantile$\{\|y_i - f(x_i)\|\}$
Classification	Deviance	kth component: $I(y_i = \mathcal{G}_k) - p_k(x_i)$

Algorithm

Algorithm 3: GBDT

Input: $D = \{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_n, y_n)\}$, **where** $\mathbf{x}_i \in X, y_i \in \{-1, 1\}$

Initialize: $f_0(x) = \arg\min_\mu \sum_{i=1}^n L(y_i, \mu)$

for m=1,2,...,M **do**

    for i=1,2,...,n **do**

        $r_{im} = -\left[\frac{\partial L(y_i, f_{m-1}(\mathbf{x}_i))}{\partial f_{m-1}(\mathbf{x}_i)}\right]$

        Fit a regression tree to targets r_{im} giving terminal regions $R_{jm}, j = 1, 2, ..., J_m$

    end

    for j=1,2,...,J_m **do**

        $\mu_{jm} = \arg\min_\mu \sum_{\mathbf{x}_i \in R_{jm}} L(y_i, f_{m-1}(\mathbf{x}_i) + \mu), j = 1, 2, ..., J_m$

        Update $f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \sum_{j=1}^{J_m} \mu_{jm} \mathbb{I}(\mathbf{x} \in R_{jm})$

    end

end

Output: $\hat{f}(\mathbf{x}) = f_M(\mathbf{x})$

THANK YOU!