# Ensemble Learning

Fish

2019 年 6 月 25 日

**Content**

## 1 the relation between decision tree and random forest

1. the decision treee of random forest take samplt independently each other .
2. sample weighting :A

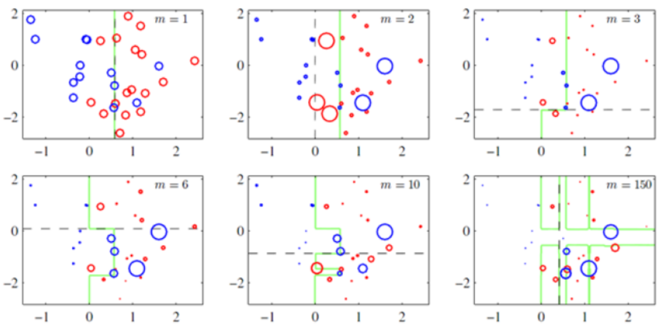

Fig 1: samplt weighting

## 2 boosting

### 2.1 concept

1. boosting is a technique, adapted to solve regression and classfication problem. it produce a weak predict model (like decision tree), and weghting cumulation to summary model, if the produce of weak model is depend on the gradient direction fo loss function every step, it is named gradient boostion.
2. gradient boosting althgorithm make sure that a target loss function firstly, its definitin domain is the collection of weak function (basic fuction) all that are feasibility; it get be close to local minimal value choosing a basic function of negative gradient direction iteratly. this opinion about gradient boosting in function domain has a deep influence on machine learning.
3. the meaning of boosting in theory:

   if a problem exit weak classifiter, then it could obtain strong classifiter by taking the method of boosting.

### 2.2 boosting althgorithm

1. form the any training sample $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$ given the input vetor x and output variable y, the target is that find approximate functin $\hat{F}(\vec{x})$, make loss value of loss function $L(y, F(x))$ be least.

2. typical definition of $L(y, F(x))$ is:

$$L(y, F(\vec{x})) = \frac{1}{2}(y - F(\vec{x}))^2 \tag{1}$$

$$L(y, F(\vec{x})) = |y - F(\vec{x})| \tag{2}$$

3. assume that optimal function is $F^*(\vec{x})$, as follow:

$$F^*(\vec{x}) = \arg\min_F E_{(x,y)}\left[L(y, F(\vec{x}))\right] \tag{3}$$

4. assume that $F(x)$ is the weghting sum of a collection of basic function $f_i(x)$

$$F(\vec{x}) = \sum_{i=1}^{M} \gamma_i f_i(x) + const \tag{4}$$

### 2.3 derivation of boosting althgorithm

1. find optimal solution $F(x)$ by gradient boostin althgorithm, let the excetion fo loss function over the train set be least. the method as follow:

2. firstly, given const function $F_0(x)$ :

$$F_0(\vec{x}) = \arg\min_\gamma \sum_{i=1}^{n} L(y_i, \gamma) \tag{5}$$

3. getting $F_m(x)$ by developing greedy thinking :

$$F_m(\vec{x}) = F_{m-1}(\vec{x}) + \arg\min_{f \in H} \sum_{i=1}^{n} L(y_i, F_{m-1}(\vec{x}_i) + f(\vec{x}_i)) \tag{6}$$

4. greedy method still is difficult when select optimal basic function every times.

   (a) to be approximate calculate using gradient descent method

   (b) bring sample to basic function $f(x)$, get $f(x_1), f(x_2), ..., f(x_n)$, then L degrade into the vector $L(y_1, f(x_1))$, $L(y_2, f(x_2)), ..., L(y_n, f(x_n))$

$$F_m(\vec{x}) = F_{m-1}(\vec{x}) - \gamma_m \sum_{i=1}^{n} \nabla_f L(y_i, F_{m-1}(\vec{x}_i)) \tag{7}$$

5. among above equation, the weight value $\gamma$ is the step length of gradient descent, solve optimal step length using linear searching:

$$\gamma_m = \arg\min_\gamma \sum_{i=1}^{n} L\left(y_i, F_{m-1}(\vec{x}_i) - \gamma \cdot \nabla_f L(y_i, F_{m-1}(\vec{x}_i)\right) \tag{8}$$

### 2.4 pseudo code of boosting algorithm

1. given model be const initially:

$$F_0(\vec{x}) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma) \tag{9}$$

2. for $m = 1$ to $M$:

   (a) calculate pseudo residuals: for $i = 1, 2, ..., n$

   $$r_{im} = \left[ \frac{\partial L(y_i, F(\vec{x}_i))}{\partial F(\vec{x}_i)} \right]_{F(\vec{x}) - F_{m-1}(\vec{x})} \tag{10}$$

   (b) calculate basic function $f_m(x)$ of fitting residuals using data $\{(\vec{x}_i, r_{im})\}_{i=1}^{n}$

   (c) calclulate step length

   $$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(\vec{x}_i) - \gamma \cdot f_m(\vec{x}_i)\right) \tag{11}$$

   - optimal problem about one dimension

   (d) update model

   $$F_m(\vec{x}) = F_{m-1}(\vec{x}) - \gamma_m f_m(\vec{x}_i) \tag{12}$$

## 3 gradient boosting decision tree GBDT

### 3.1 concept and algorithm theory

1. the typical basic function of gradient boosting is decision tree (CART in particular)

2. calculate decision tree $t_m(x)$ due to pseudo residuals data an mth gradient boosting. make the number of leaf node for tree $t_m(x)$ be J, then tree $t_m(x)$ devide the input space into J intersect area $R_{1m}, R_{2m}, ..., R_{Jm}$, and decision tree $t_m(x)$ could give assured prediction in every area. using indicated tag $I(x)$, for input x, $t_m(x)$ is:

$$t_m(\vec{x}) = \sum_{j=1}^{J} b_{jm} I(\vec{x} \in R_{jm}) \tag{13}$$

3. among, $b_{jm}$ is the predict value that sample x in area $R_{jm}$

4. minimize loss function using linear searching to calculate learning rate

$$F_m(\vec{x}) = F_{m-1}(\vec{x}) + \gamma \cdot t_m(\vec{x}_i) \tag{14}$$

$$\gamma = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, F_{m-1}(\vec{x}_i) + \gamma \cdot t_m(\vec{x}_i)) \tag{15}$$

5. what's more: calculate step length differentiatly for every area of tree, then combine the coefficient $bjm$ to step length. then :

$$F_m(\vec{x}) = F_{m-1}(\vec{x}) + \sum_{j=1}^{J} \gamma_{jm} I(\vec{x} \in R_{jm}) \tag{16}$$

$$\gamma_{jm} = \arg\min_{\gamma} \sum_{\vec{x}_i \in R_{jm}} L(y_i, F_{m-1}(\vec{x}_i) + \gamma \cdot t_m(\vec{x}_i)) \tag{17}$$

### 3.2 set parameter and regularization

1. decline the generalization ability fitting training dataset over highly, need to regularization technique to decline over fitting.
   (a) increase penalty party for complex model, like: the complex degree is positive with leaf node number or the square fo predicted value of leaf node.
   (b) be used to pruning of decision tree
2. the number of leaf node control the number of layer of tree, select $4 \leq J \leq 8$ in generally.
3. leaf node contain least the number of sample
   (a) avoid occur over small leaf node, decline predicting variance.
4. the number of iteration times M about gradient boosting
   (a) decline the loss value of training dataset by increase M, but exit the risk over fitting
   (b) cross validation

### 3.3 summary of GBDT

1. function estimate is treated as a problem in function space not parameter space originally. but phased additive expansion and gradient decline transfer function estimate into parameter estimate.

2. loss function is least square error、absolute error and so on, and is classfication problem; but error function is changed to more class Logistic likelihood function, and become a classification function.

3. make target function divide into the weighted sum of any basic function, is common technique method: here could see its shaddow nenural network、radial base function、Fourier / wavelet transform、SVM.

## 4 XGBoost

### 4.1 sencond derivation information

1. target function:

$$J(f_t) = \sum_{i=1}^{n} L(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) + C \tag{18}$$

2. according to Taylor developing equation:

$$f(x + \Delta x) \approx f(x) + f'(x)\Delta x + \frac{1}{2}f''(x)\Delta x^2 \tag{19}$$

3. define,

$$g_i \overset{def}{==} \frac{\partial L(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}} \tag{20}$$

$$h_i \overset{def}{==} \frac{\partial^2 L(y_i, \hat{y}_i^{(t-1)})}{\partial \hat{y}_i^{(t-1)}} \tag{21}$$

4. get:

$$J(f_t) \approx \sum_{i=1}^{n} \left[ L(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i)) + \frac{1}{2} h_i f_t^2(x_i) \right] + \Omega(f_t) + C \tag{22}$$

### 4.2 describe of decision tree

1. classfy or regress the sample using decision tree, the thining process from root node to leaf node, the predict value of sample that belong to alike leaf node is equal.

2. assume that the number of leaf node of decision tree is T, weight value of every leaf node is $\vec{w} = (w_1, w_2, ..., w_T)$, the learning process of decision tree, also explain building process how to use feature devided and getting weight value.

3. sample x is belong to leaf node q, define function : $f_t(x) = w_{q(x)}$

   - the kernel of decision tree is "tree structure" and "leaf weight value"

### 4.3 definition of regularization $\Omega(f_t)$

1. the complex degree of decision depend on the number of leaf node and leaf weight value, like using weighting between the number of all leaf node and square sum of leaf weight value

$$\Omega(f_t) = \gamma \cdot T_t + \lambda \cdot \frac{1}{2} \sum_{j=1}^{T} w_j^2 \tag{23}$$

$T$ :leaf number $w_j^2$ : square sum of weight

### 4.4 calculation and simplify of loss function

1. calculation

$$\begin{aligned}
J(f_t) &\approx \sum_{i=1}^{n}\left[L(y_i,\hat{y}_i^{(t-1)}) + g_i f_t(x_i)) + \frac{1}{2}h_i f_t^2(x_i)\right] + \Omega(f_t) + C \\
&= \sum_{i=1}^{n}\left[g_i f_t(x_i) + \frac{1}{2}h_i f_t^2(x_i)\right] + \Omega(f_t) + C \\
&= \sum_{i=1}^{n}\left[g_i w_{q(x_i)} + \frac{1}{2}h_t w_{q(x_i)}^2\right] + \gamma\cdot T + \lambda\cdot\frac{1}{2}\sum_{j=1}^{T} w_j^2 + C \\
&= \sum_{j=1}^{T}\left[\left(\sum_{i\in I_j} g_i\right) w_j + \frac{1}{2}\left(\sum_{i\in I_j} h_t\right) w_j^2\right] + \gamma\cdot T + \lambda\cdot\frac{1}{2}\sum_{j=1}^{T} w_j^2 + C \\
&= \sum_{j=1}^{T}\left[\left(\sum_{i\in I_j} g_i\right) w_j + \frac{1}{2}\left(\sum_{i\in I_j} h_t + \lambda\right) w_j^2\right] + \gamma\cdot T + C
\end{aligned} \tag{24}$$

2. simplify target function

   (a) for : $J(f_t) = \sum\limits_{j=1}^{T}\left[\left(\sum\limits_{i\in I_j} g_i\right) w_j + \frac{1}{2}\left(\sum\limits_{i\in I_j} h_t + \lambda\right) w_j^2\right] + \gamma\cdot T + C$

   (b) define $G_j \overset{def}{=\!=} \sum\limits_{i\in I_j} g_i,\ H_j \overset{def}{=\!=} \sum\limits_{i\in i_j} h_i$

   (c) then, $J(f_t) = \sum\limits_{j=1}^{T}\left[G_j w_j + \frac{1}{2}(H_j+\lambda)w_j^2\right] + \gamma\cdot T + C$

   (d) derivate partially for w, getting

$$\frac{\partial J(f_t)}{\partial w_j} = G_j + (H_j+\lambda)w_j \overset{令}{=\!=} 0 \tag{25}$$

$$\Rightarrow w_j = -\frac{G_j}{H_j+\lambda} \tag{26}$$

   (e) bring back to target function, getting

$$J(f_t) = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_J^2}{H_j+\lambda} + \gamma\cdot T \tag{27}$$

### 4.5 example

1. what

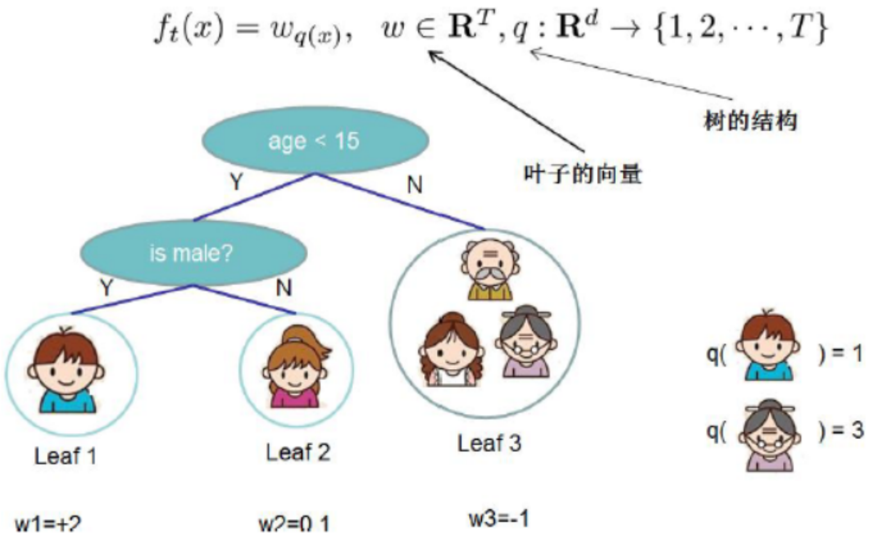

Fig 2: XGBoost example

2. penalty part

$$\Omega = \gamma \cdot 3 + \frac{1}{2} \cdot \lambda \cdot (4 + 0.01 + 1) \tag{28}$$

3. calculation

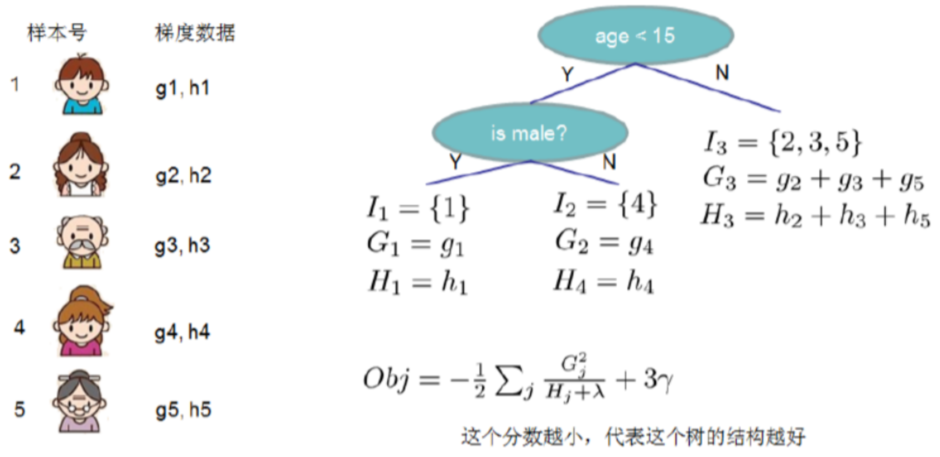

Fig 3: XGBoost example calculation

4. how to bulid the structure of decision tree $J(f_t) = -\frac{1}{2}\sum_{j=1}^{T} \frac{G_j^2}{H_j+\lambda} + \gamma \cdot T$

   (a) equestion: for current node, how to divide subtree?

(b) answer: use for reference the solution fo ID3/C4.5/CART, using greedy method:

   i. for a feasible division, calculate the $j(f)$ after dividing

   ii. for all feasible division, choose the segementation dot making $J(f)$ decline least

5. enumerate feasible segemetation dot, select division of largest gain, continue same operation, until statify threshold or get pure node

$$Gain(\phi) = \frac{1}{2}\left[\frac{G_L^2}{H_L+\lambda} + \frac{G_R^2}{H_R+\lambda} - \frac{(G_L+G_R)^2}{H_L+H_R+\lambda}\right] \tag{29}$$

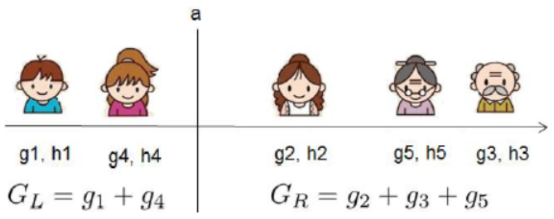

Fig 4: XGBoost example calculation result

## 5 Adaboosting

### 5.1 thinking

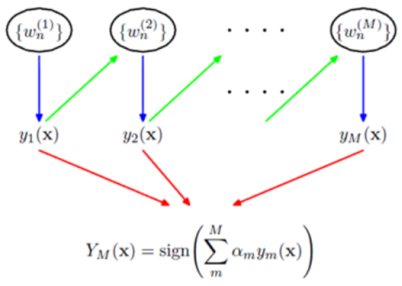

Fig 5: boosting thinking

### 5.2 algorithm

1. assume training dataset $T = \{(x_1, y_1), (x_2, y_2), ..., (x_N, y_N))\}$

2. initialize the weight distribution of training dataset

$$D_1 = (w_{11}, w_{12}, ..., w_{1i}, ..., w_{1N}), \quad w_{1i} = \frac{1}{N}, \quad i = 1, 2, ..., N \tag{30}$$

3. study algorithm using traning dataset that weight distribution is like $D_m$, get basic classfier

$$G_m(x) : \chi \to \{-1, +1\} \tag{31}$$

4. calculate the classfication error rate that $G_m(x)$ trained in training dataset

$$e_m = P(G_m(x_i) \neq y_i) = \sum_{i=1}^{N} w_{mi} I(G_m(x_i) \neq y_i) \tag{32}$$

5. calculate the coefficient of $G_m(x)$

$$\alpha_m = \frac{1}{2} \log \frac{1 - e_m}{e_m} \tag{33}$$

6. update the weight distribution of training dataset

$$D_{m+1} = (w_{m+1,1}, w_{m+1,2}, ..., w_{m+1,t}, ..., w_{m+1,N}) \tag{34}$$

$$w_{m+1,i} = \frac{w_{mi}}{Z_m} \text{exp}(-\alpha_m y_i G_m(x_i)), \quad i = 1, 2, .., N \tag{35}$$

7. here, $Z_m$ is standardization factor

$$Z_m = \sum_{i=1}^{N} w_{mi} \text{exp}(-\alpha_m y_i G_m(x_i)) \tag{36}$$

   - its purpose is only making $D_{m+1}$ become a probability distribution

$$w_{m+1,i} = \frac{w_{mi}}{Z_m} \text{exp}(-\alpha_m y_i G_m(x_i)) \tag{37}$$

$$\Rightarrow Z_m w_{m+1,i} = w_{mi} \text{exp}(-\alpha_m y_i G_m(x_i)) \tag{38}$$

$$\Rightarrow Z_1 w_{2,i} = w_{1i} \text{exp}(-\alpha_1 y_i G_1(x_i)) \tag{39}$$

8. build linear group of basic classfier

$$f(x) = \sum_{m=1}^{M} \alpha_m G_m(x) \tag{40}$$

9. final classfier

$$G(x) = sign(f(x)) = sign\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right) \tag{41}$$