Lets start from Adaboost & move towards Xgboost. We assumes that we are familiar with "Decision trees" & "Random forest".

1. Adaboost (Adaptive boosting)

⇒ Three main ideas:

1. In random forest each time we make a tree, we make full sized trees. (no predetermined maximum depth)

   In contrast trees made with Adaboost are usually just a node & two leaves.

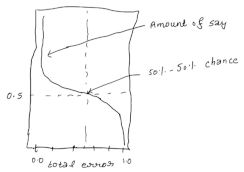
   ⇒ stumps can only use one variable to make a decision.
   "weak learners"

2. In contrast to Random forests, in forest of stumps made with AdaBoost, some stumps get more say in the final classification than others.

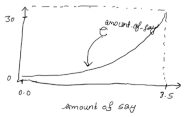
3. Similarly, order in which stumps are made is important. Unlike random forest where the trees order is not important.

   ➡ Reason that the first stump makes influences the reason the second stump makes ... ...

* Amount of say for a stump = $\frac{1}{2}\log\left(\frac{1-\text{total error}}{\text{total error}}\right)$



New sample weight = sample weight * $\exp^{(\text{amount of say})}$



* Strategy is to increase the sample weights of "incorrectly classified" samples and decrease the weights of "correctly classified" samples.

If we have a "weighted min function" then we use it with the sample weights, otherwise we use the sample weights to make a new dataset that reflects these weights.

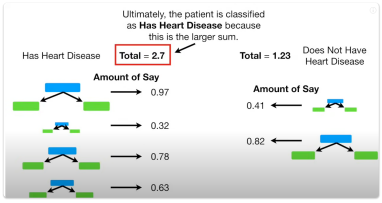


fig: Example of how AdaBoost makes predictions

2. Gradient Boosting (Regression):

1. Gradient Boost starts by making a single leaf, instead of a tree or stump. This leaf represents an initial guess for the weights of all of the samples.
   ↳ Average values.

2. Then gradient boost builds a tree, but still restricts the size of the tree (8 to 32 leaves generally).

3. Like Adaboost, Gradient boost scales the tree. However it scales all trees by same amount. Then gradient based builds another tree based on the errors made by previous tree & then it scales the tree ... ...

Prediction = [ ] + $\rho * T_1 + \rho * T_2 + \rho * T_3 + \dots \dots$

[ ] ↳ Average & initial value & guess

Where, ρ = learning rate & taking small step in right direction

$T_1, T_2, \dots T_3$ = Trees

** Maths **

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$ and a differentiable Loss function $L(y_i, F(x))$.

Step I: Initialize model with a constant value: $F_0(x) = \underset{\rho}{\text{argmin}} \sum_{i=1}^{n} L(y_i, \rho)$

Step II: For $m = 1$ to M:

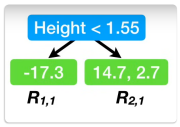
(A) Compute $r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x) = F_{m-1}(x)}$ for $i = 1, \dots, n$

(observed - predicted)
or
(Residual)

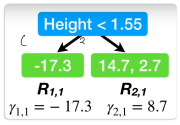
Height (m)	Favorite Color	Gender	Weight (kg)	$F_{1,1}$
1.6	Blue	Male	88	14.7
1.6	Green	Female	76	2.7
1.5	Blue	Female	56	-17.3

fig: Example of residual calculation for first tree (m=1).

(B) Fit a regression tree to the r_{im} values & create terminal regions R_{jm}, for $j = 1, \dots J_m$



(C) For $j = 1 \dots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$



(D) Update $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$

↳ learning rate [0, 1]

Step 3: Output $F_M(x)$

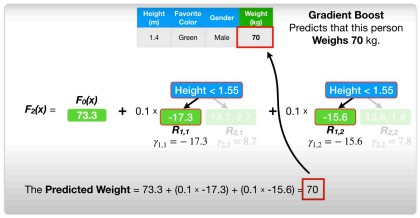


figure: Making prediction Example for M =)

classification using Gradient Boosting:

Input: Data $\{(x_i, y_i)\}_{i=1}^{n}$, and a differentiable Loss function $L(y_i, F(x))$.

Step 1: Initialize model with a constant value: $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$.
↓
log(odds)

Note: log(odds) = $\log\left(\frac{\text{Probability of samples in "yes" class}}{\text{Prob. of samples in "No" class}}\right)$

if "p" is the probability for "Yes" class then,

log(odds) = $\log\left(\frac{p}{1-p}\right)$

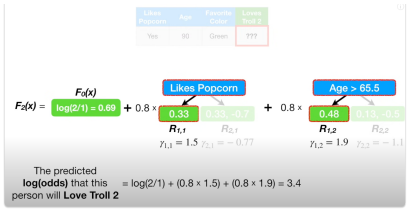
Step II: for m = 1 to M:

(A) Compute $r_{im} = \left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x) = F_{m-1}(x)}$ for $i = 1, \dots n$

(B) Fit a regression tree to the r_{im} values & create terminal regions R_{jm}, for $j = 1, \dots J_m$.

(C) For $j = 1 \dots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$

(D) Update $F_m(x) = F_{m-1}(x) + \nu \sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})$



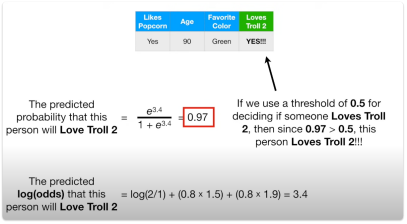


fig: Example showing classification using gradient boosting

Important considerations for classification using gradient boosting.

- Log (likelihood of the observed data given the prediction)

$$= \sum_{i=1}^{N} y_i \times \log(p) + (1-y_i) \times \log(1-p)$$

where, p = predicted probability

The goal is to maximize this log-likelihood function.
Or, minimize -log-likelihood function.

On simplification we get:

$$\text{Loss-function } (L) = -\text{observed} * \log(\text{odds}) + \log\left(1 + e^{\log(\text{odds})}\right)$$

$$* \log(\text{odds}) = \log\left(\frac{p}{1-p}\right)$$

$$* \text{observed} = y$$

Also, $\frac{\partial L}{\partial \log(\text{odds})} = -\text{observed} + p = -y + p$

- In step (C) where we need to determine

$$\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$$

for single region R_{11},

$$\gamma_{11} = \underset{\gamma}{\operatorname{argmin}} -y_i \times [F_{m-1}(x_i) + \gamma] + \log\left(1 + e^{F_{m-1}(x_i) + \gamma}\right)$$

Solving for optimal gamma:

Approximate the loss function: Taylor polynomial

$$L(y_i, F_{m-1}(x_i) + \gamma) \simeq L(y_i, F_{m-1}(x_i)) + \frac{\partial}{\partial F()}(y_i, F_{m-1}(x_i))\gamma + \frac{1}{2}\frac{\partial^2}{\partial F()^2}(y_i, F_{m-1}(x_i))\gamma^2$$

So, $\frac{\partial L(y_i, F_{m-1}(x_i) + \gamma)}{\partial \gamma} = \frac{\partial}{\partial F()}(y_i, F_{m-1}(x_i)) + \frac{\partial^2}{\partial F()^2}(y_i, F_{m-1}(x_i))\gamma$

Now, solving for γ with $\frac{\partial L}{\partial \gamma} = 0$ we get,

$$\gamma = \frac{-\frac{d}{dF()}(y_i, F_{m-1}(x_i))}{\frac{d^2}{dF()^2}(y_i, F_{m-1}(x_i))}$$

Or, $\gamma = \frac{\text{observed} - p}{p \times (1-p)}$

Or, $\gamma = \frac{\text{Residual}}{p \times (1-p)}$

For multiple samples in a leaf:

$$\frac{\sum \text{Residuals}_i}{\sum_{i=0}^{n} p_i \times (1-p_i)}$$