ML2 : Boosting

Ramesh Kestur

Topics

- Recap of Decision Trees and Random Forest
- Adaboost illustration with example
- Adaboost algorithm
- Code demo/walkthrough of Adaboost

- Compare and contrast Adaboost and Gradient Boost
- Gradient boost illustration
- Gradient boost algorithm description with walk through of example
- Code demo/walkthrough of Gradient boost

Decision trees

→ Multiple decision trees
→ Each tree is independent of the other

→ Some trees may be bigger than the others
→ There is no pre-determined maximum depth (Assuming they are not truncated)

AdaBoost

→ The trees are just a node and two leaves, called stumps

→ A forest of stumps

→ Stumps can use only one variable to make a decision and thus are called 'weak learners'

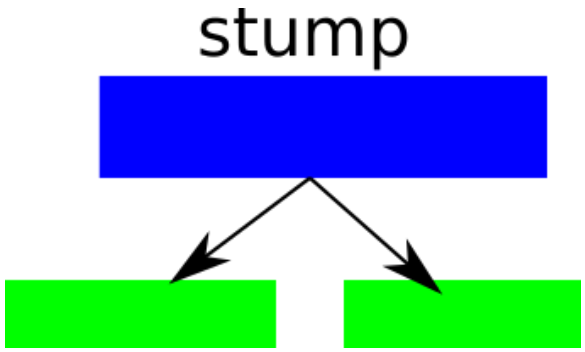
Random forest

- Each tree has an equal weight on the final classification
- Each decision tree is made independently of the others.
- Order is not important

Adaboost

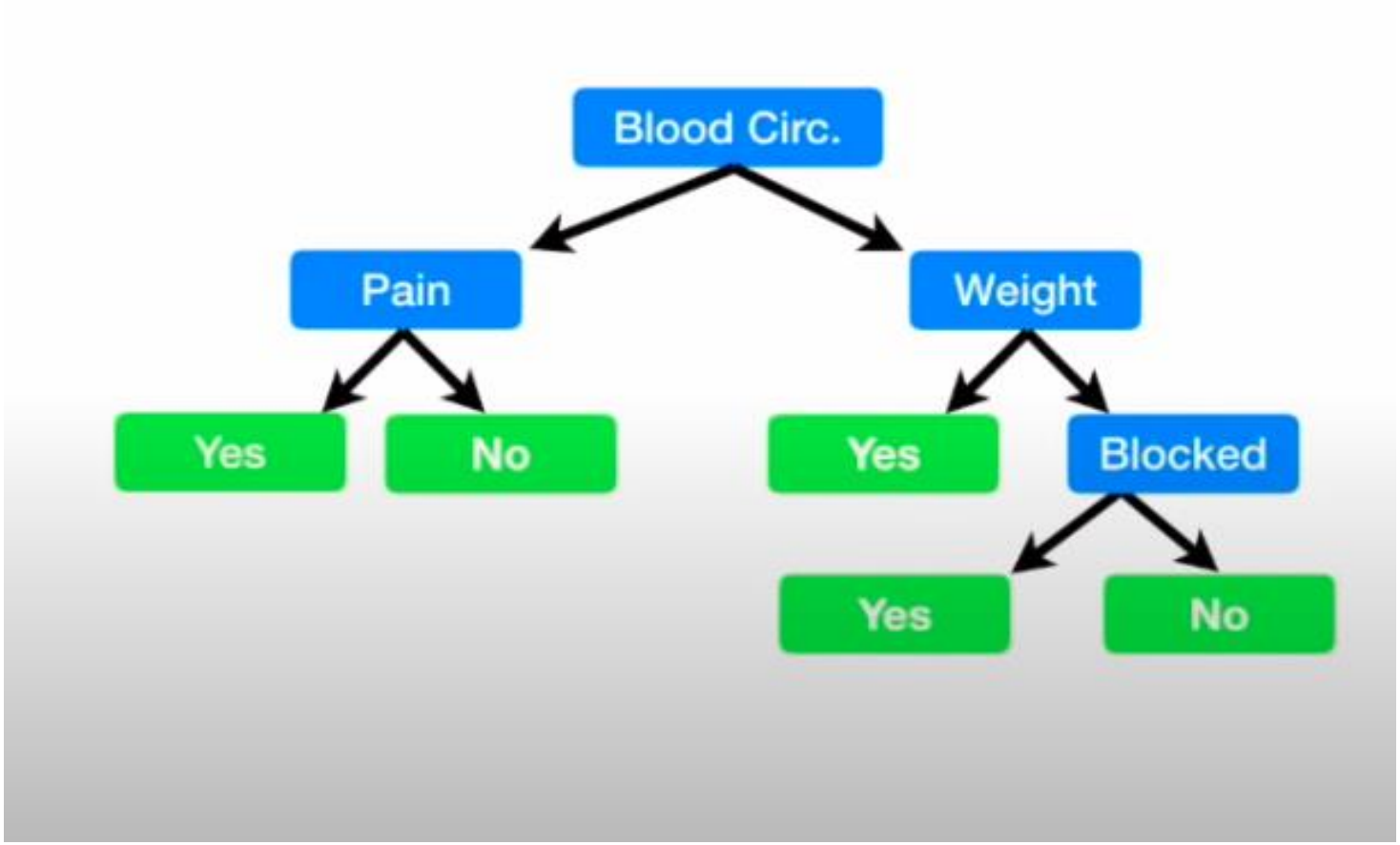
- Some stumps get more say in the final classification than others
- Order is important.
- The error that the first stump makes influences how the second stump is made

Illustration

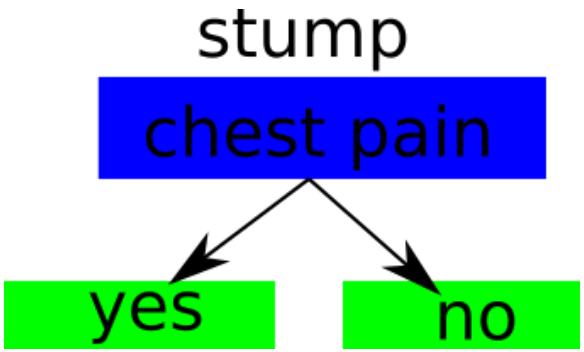


A tree with just one node and two leaves

Chest pain	Good blood circulation	Blocked arteries	Weight	Heart disease
No	No	No	60	No
Yes	Yes	Yes	81	Yes
Yes	Yes	No	95	No
Yes	No	Yes	75	Yes



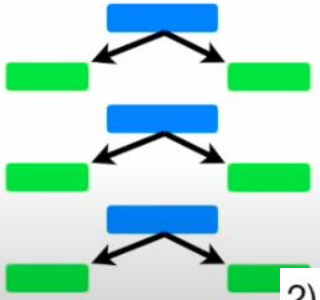
A full sized decision tree would take advantage of all variable to make a decision



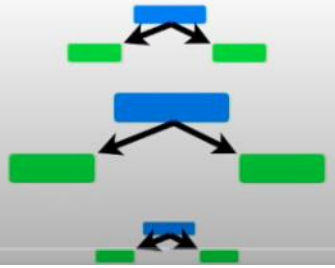
- A stump can use only one variable to make a decision
- Stumps are weak learners

In summary, the three ideas behind adaboost are

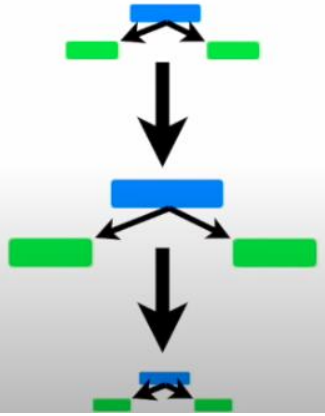
1) **AdaBoost** combines a lot of "weak learners" to make classifications. The weak learners are almost aways **stumps**.



2) Some **stumps** get more say in the classification than others.



3) Each **stump** is made by taking the previous **stump's** mistakes into account.

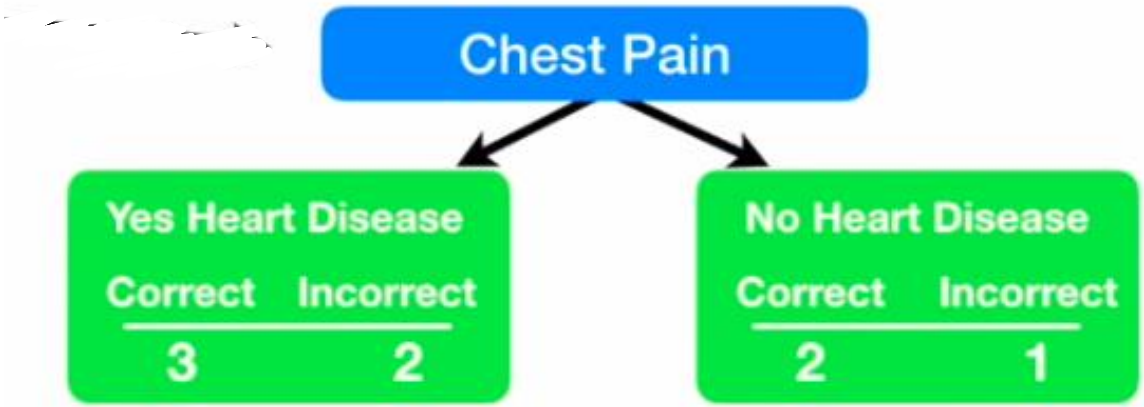


Creating a decision tree using Adaboost

Chest pain	Blocked arteries	Patient Weight	Heart disease
Yes	Yes	93	Yes
No	Yes	82	Yes
Yes	No	95	Yes
Yes	Yes	76	Yes
No	Yes	71	No
No	Yes	57	No
Yes	No	77	No
Yes	Yes	78	No

Creating a decision tree using adaboost

Chest pain	Blocked arteries	Patient Weight	Heart disease	Sample weight
Yes	Yes	93	Yes	1/8
No	Yes	82	Yes	1/8
Yes	No	95	Yes	1/8
Yes	Yes	76	Yes	1/8
No	Yes	71	No	1/8
No	Yes	57	No	1/8
Yes	No	77	No	1/8
Yes	Yes	78	No	1/8



$$1-\left(\frac{3}{5}\right)^2-\left(\frac{2}{5}\right)^2 = 1-\frac{9}{25}-\frac{4}{25}$$

$$= 1-\frac{13}{25} = \frac{25-13}{25}$$

$$= \frac{12}{25} = 0.52$$

$$1-\left(\frac{2}{3}\right)^2-\left(\frac{1}{3}\right)^2 = 1-\frac{4}{9}-\frac{1}{9}$$

$$= 1-\frac{5}{9}$$

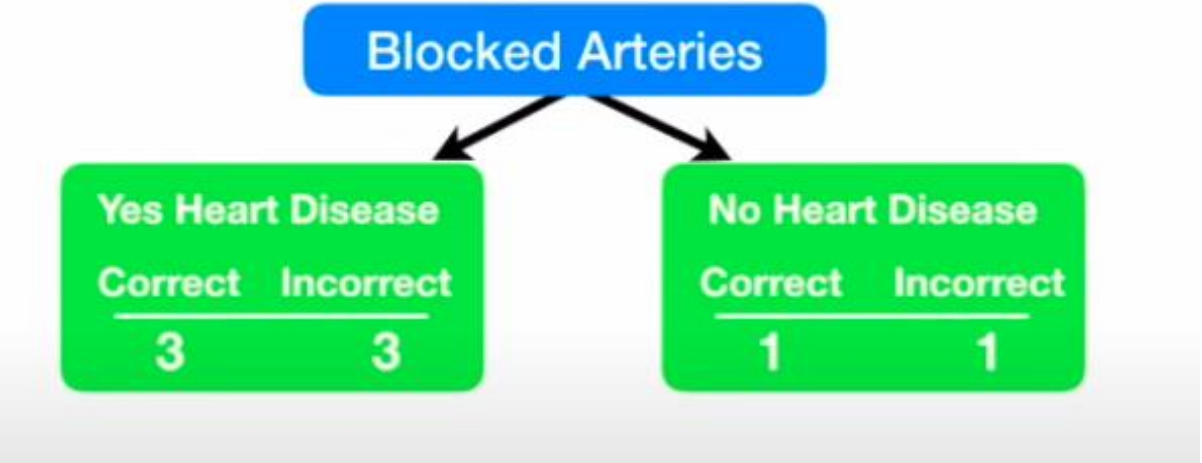
$$= 0.44$$

$$\frac{5}{8}\times 0.52 + \frac{3}{8}\times 0.44$$

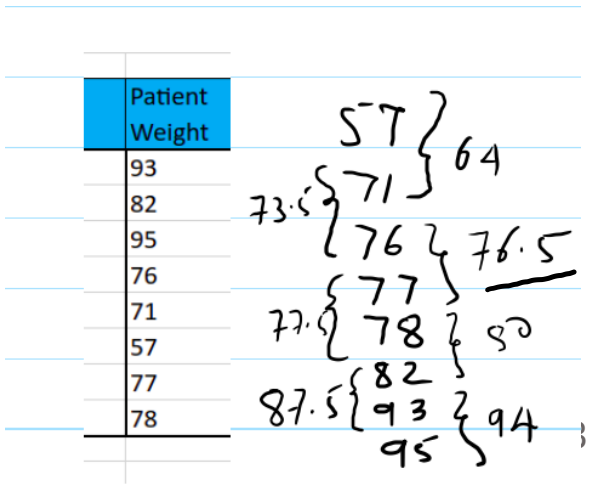
$$= .325 + .165$$

$$= 0.49$$

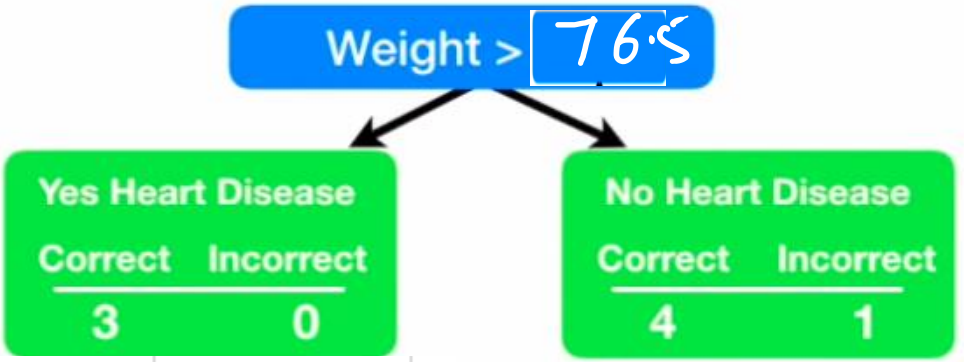
Gini index



Gini Index = 0.5



Gini Index = 0.2



Chest pain	Blocked arteries	Patient Weight	Heart disease	Sample weight
Yes	Yes	93	Yes	1/8
No	Yes	82	Yes	1/8
Yes	No	95	Yes	1/8
Yes	Yes	76	Yes	1/8
No	Yes	71	No	1/8
No	Yes	57	No	1/8
Yes	No	77	No	1/8
Yes	Yes	78	No	1/8

The total error for a stump is the sum of weights of the incorrectly classified samples

→ Since sum of weights add up to 1 the error is in the range of 0 to 1

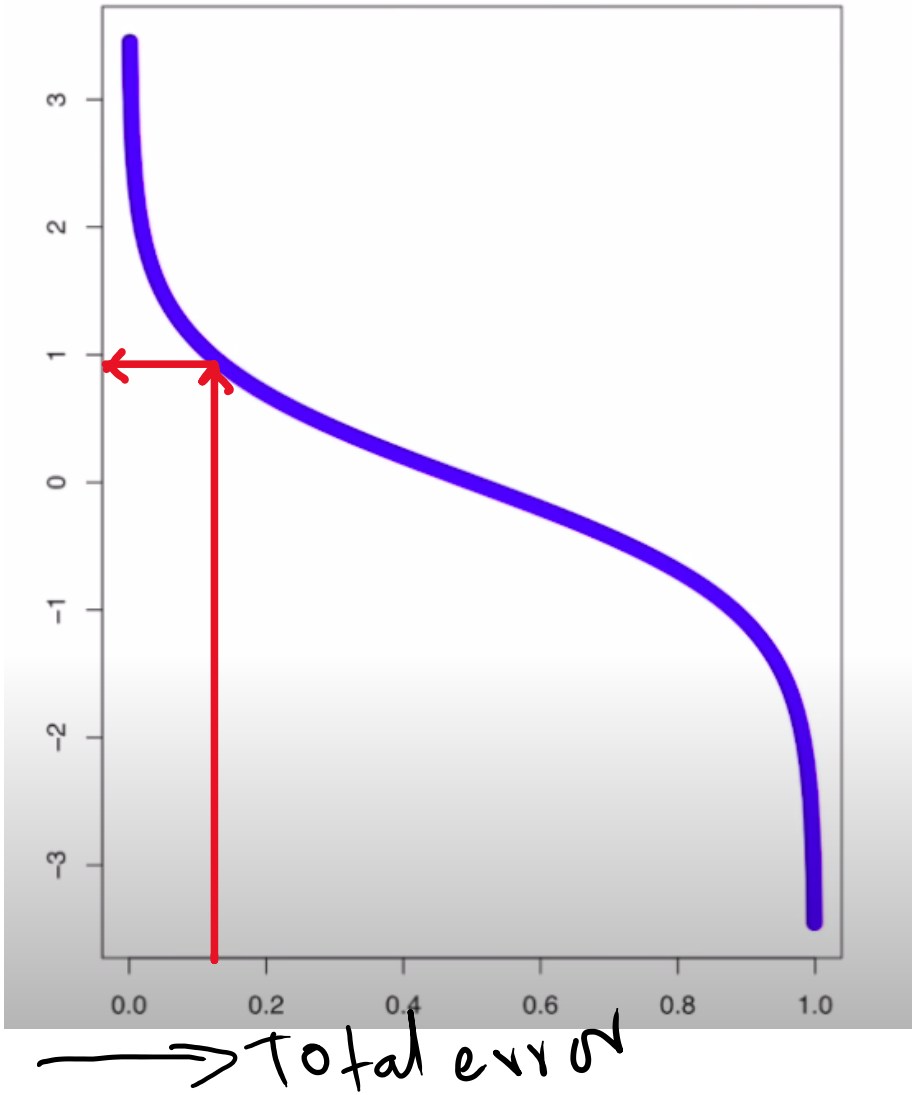
→ Total error is used to determine the amount of say of a stump in the final classification

→ Amount of say $= \frac{1}{2} \log \left(\frac{1 - \text{Total error}}{\text{total error}} \right)$

$$\frac{1}{2} \log \left(\frac{1 - \frac{1}{8}}{\frac{1}{8}} \right)$$

$$= \frac{1}{2} \log \left(\frac{\frac{7}{8}}{\frac{1}{8}} \right)$$

$$= \frac{1}{2} \log 7 = 0.97$$



Amount of Say

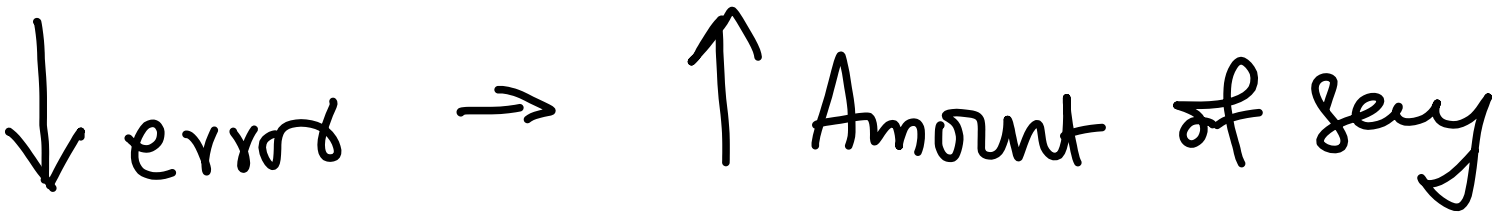
$$= \frac{1}{2}\log\left[\frac{1 - \text{Total Error}}{\text{Total error}}\right]$$

$$= \frac{1}{2}\log\left[\frac{1 - \frac{1}{8}}{\frac{1}{8}}\right]$$

$$= \frac{1}{2}\log\left[\frac{7/8}{1/8}\right]$$

$$= \frac{1}{2}\log 7 = 0.9$$

Now updating the sample weights.....



Now updating the sample weights.....

Chest pain	Blocked arteries	Patient Weight	Heart disease	Sample weight
Yes	Yes	93	Yes	1/8
No	Yes	82	Yes	1/8
Yes	No	95	Yes	1/8
Yes	Yes	76	Yes	1/8
No	Yes	71	No	1/8
No	Yes	57	No	1/8
Yes	No	77	No	1/8
Yes	Yes	78	No	1/8

↑ weight of incorrectly classified samples

↓ wt of correctly classified samples

Now updating the sample weights.....

For incorrectly classified samples:

New Sample weight = Sample wt $\times e^{\text{amount of say}}$

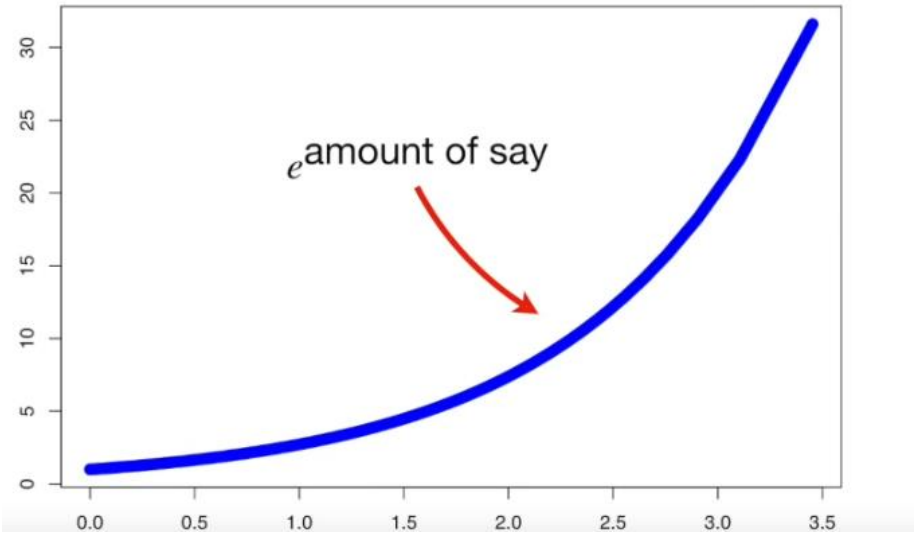
$= 1/8 \times e^{0.97}$

$= \frac{1}{8} \times 2.64$

$= 0.33$

Now updating the sample weights.....

Incorrectly classified samples

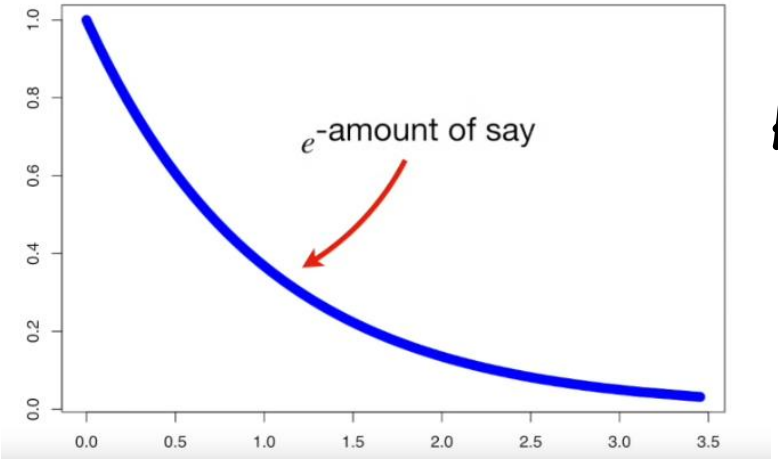


Amount of say →

Now updating the sample weights.....

For correct samples – Amount of say

New sample wt = Sample wt x e



Higher amount of say
↓ by a smaller amount

Lower amount of say
↓ by a higher amount

Now updating the sample weights.....

$$\text{New sample wt} = \text{Sample wt} \times e^{-\text{Amount of say}}$$

$$= \frac{1}{8} \times e^{-0.97} = 0.3$$

$$= \frac{1}{8} \times 0.33 = 0.05$$

Re-Sampling the original dataset using the sample weights

Chest pain	Blocked arteries	Patient Weight	Heart disease	Sample weight	New sample weight	Normalized new sample weight
Yes	Yes	93	Yes	1/8	0.05	0.07
No	Yes	82	Yes	1/8	0.05	0.07
Yes	No	95	Yes	1/8	0.05	0.07
Yes	Yes	76	Yes	1/8	0.33	0.49
No	Yes	71	No	1/8	0.05	0.07
No	Yes	57	No	1/8	0.05	0.07
Yes	No	77	No	1/8	0.05	0.07
Yes	Yes	78	No	1/8	0.05	0.07

✓ Divide each wt by sum of weights ÷ by .68

Re-Sampling the original dataset using the sample weights

Chest pain	Blocked arteries	Patient Weight	Heart disease	Sample weight	New sample weight	Normalized new sample weight	
Yes	Yes	93	Yes	1/8	0.05	0.07	0.07
No	Yes	82	Yes	1/8	0.05	0.07	0.14
Yes	No	95	Yes	1/8	0.05	0.07	0.21
Yes	Yes	76	Yes	1/8	0.33	0.49	0.7
No	Yes	71	No	1/8	0.05	0.07	0.77
No	Yes	57	No	1/8	0.05	0.07	0.84
Yes	No	77	No	1/8	0.05	0.07	0.91
Yes	Yes	78	No	1/8	0.05	0.07	0.98

For Ex: Let the random numbers be .72, .42, .83, .51, .63, .43 .85 and .96

Re sampling the original dataset using the sample weights

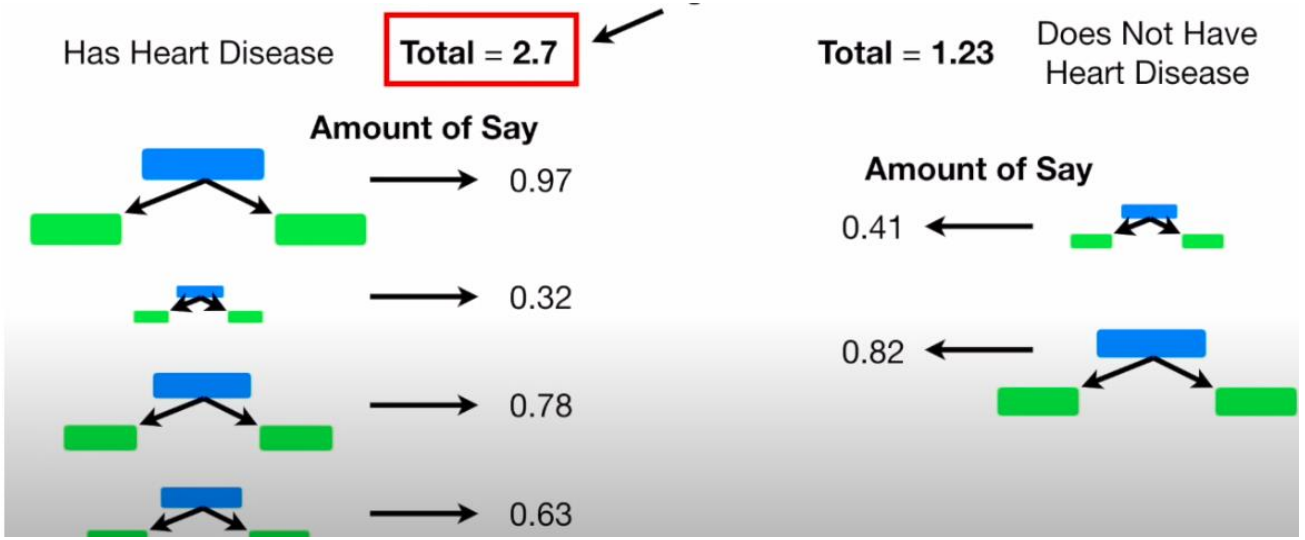
Chest pain	Blocked arteries	Patient Weight	Heart disease	Sample weight
No	Yes	71	No	1/8
Yes	Yes	76	Yes	1/8
No	Yes	57	No	1/8
Yes	Yes	76	Yes	1/8
Yes	Yes	76	Yes	1/8
Yes	Yes	76	Yes	1/8
Yes	No	77	No	1/8
Yes	Yes	78	No	1/8

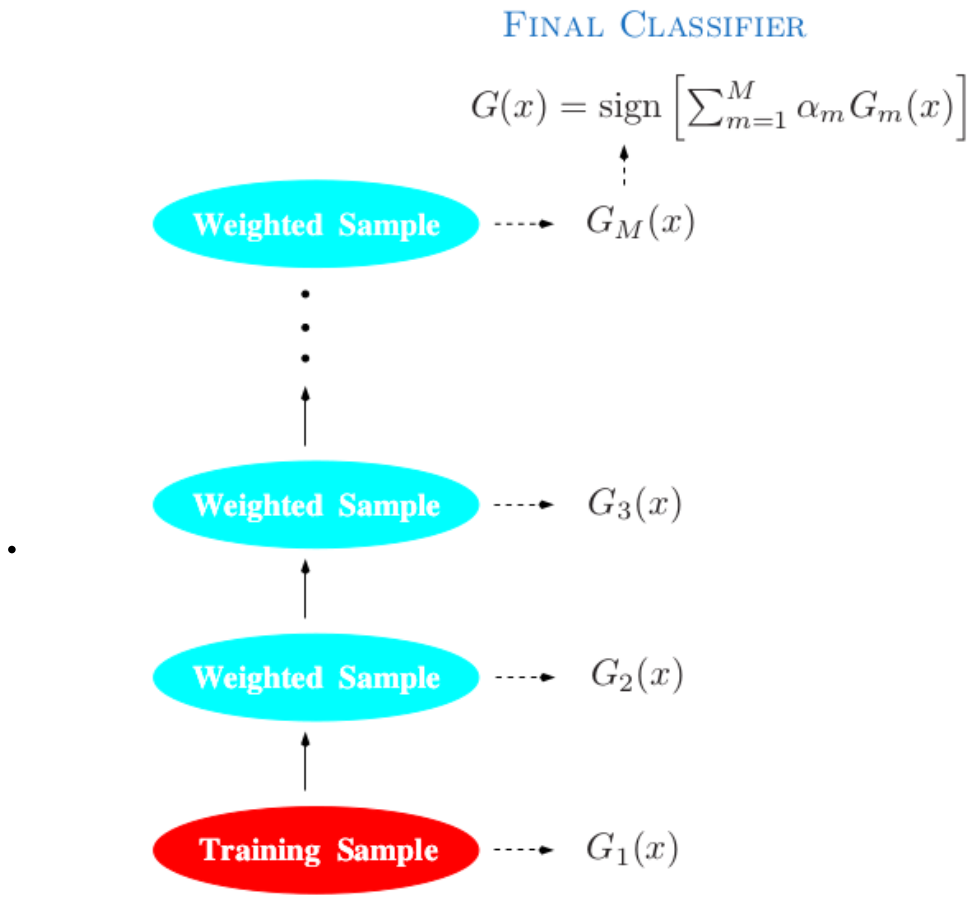
Now - Start from the beginning for this dataset. Start on the next stump

→ This creates a forest of stumps

→ classify unseen samples on the forest of stumps

→



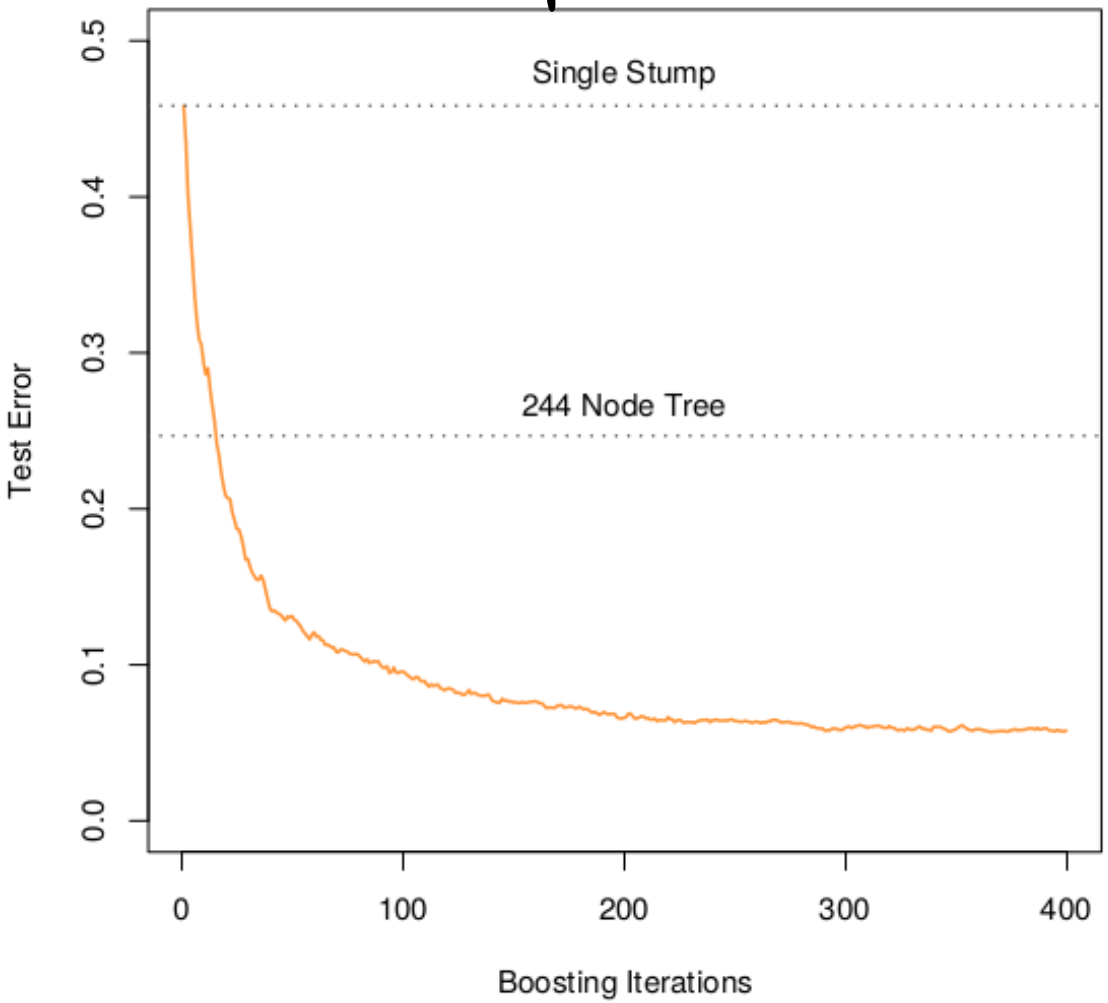


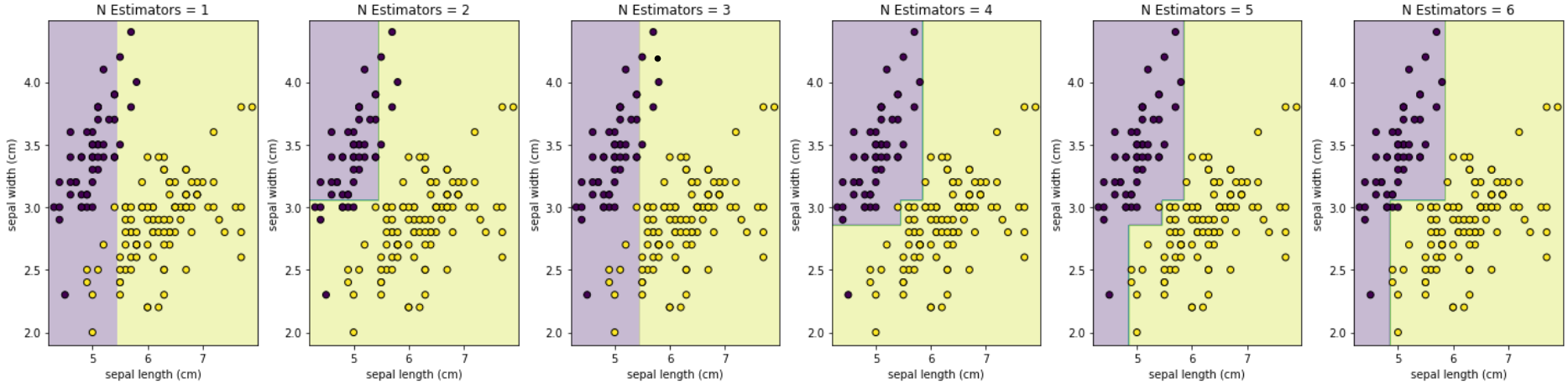
Schematic of Adaboost algorithm

Adaboost algorithm

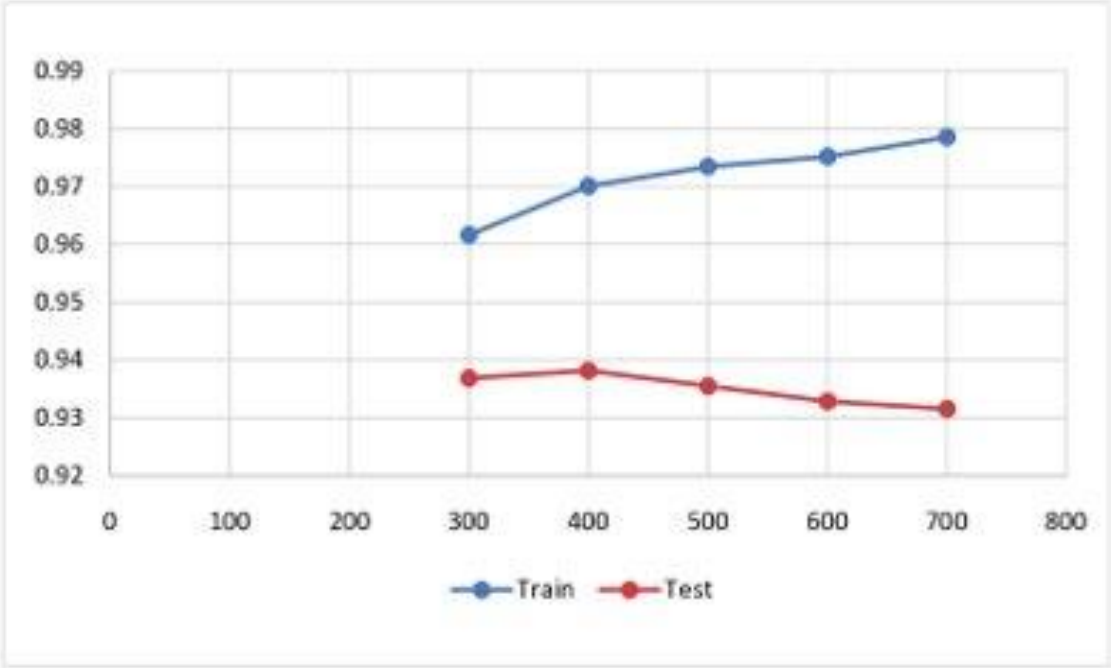
1. Initialize the observation weights $w_i = 1/N,\ i = 1, 2, \ldots, N$.
2. For $m = 1$ to M:
   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.
   (b) Compute
   $$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$
   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))],\ i = 1, 2, \ldots, N$.
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

Comparison


Single Stump
244 Node Tree
Test Error
0.0
0.1
0.2
0.3
0.4
0.5
0
100
200
300
400
Boosting Iterations


N Estimators = 1
N Estimators = 2
N Estimators = 3
N Estimators = 4
N Estimators = 5
N Estimators = 6
sepal width (cm)
sepal length (cm)
4.0
3.5
3.0
2.5
2.0
5
6
7

The number of learners.

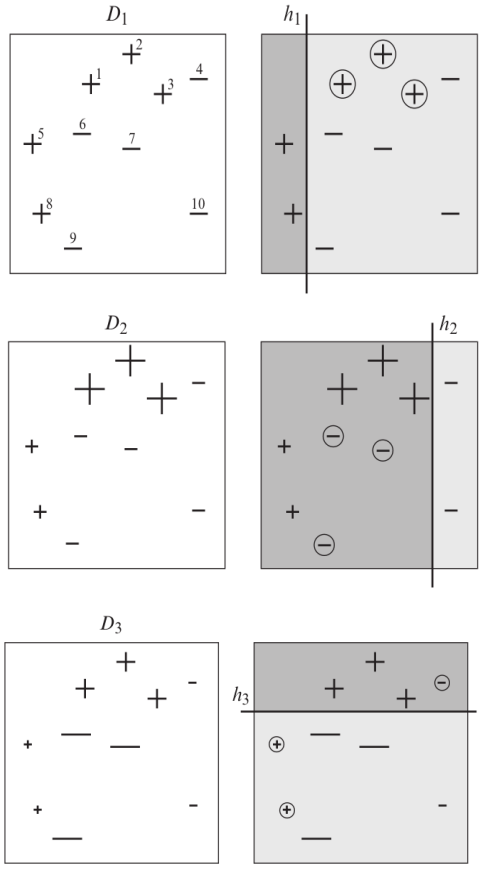


Comparing Adaboost and Random forest

- Adaboost typically provides more accurate predictions than Random Forest. However, Adaboost is more prone to to overfitting than Random Forest.
- Adaboost reduces bias where as random forest reduces variance
- Adaboost is simpler to implement a compared to random forest since they have relatively lesser no of hyper parameters

Boosting

- A series of models are constructed and with each new model iteration, the weights of the misclassified data in the previous model are increased.
- This redistribution of weights helps the algorithm identify the parameters that it needs to focus on to improve its performance.



AdaBoost illustration
No of samples = 10
Each row depicts one round, for t = 1, 2, 3. The left box in each row represents the distribution D_t , with the size of each example scaled in proportion to its weight under that distribution. Each box on the right shows the weak hypothesis h_t , where darker shading indicates the region of the domain predicted to be positive. Examples that are misclassified by h_t have been circled.

Boosting , foundations and algorithms ; Freund and Scapire

Gradient Boosting

Comparing and contrasting Adaboost and Gradient boost

Adaboost	Gradient boost
Uses stumps	Uses fixed trees
Builds stumps based on the error of the previous stump	Builds trees based on the errors of the previous tree
Changes the weight of the data samples	Scales the tree

Example

- Gradient boost starts with a single leaf instead of a tree or a stump
- This leaf represents the initial guess for weights of all the samples
- When trying to predict a continuuous variable, the first initial prediction is usually the average values
- The gradient boost builds a tree. These trees are fixed size trees
- Like Adaboost this tree is based on the errors of the previous predictions
- Although Gradient boost uses a tree unlike stumps in Adaboost, the size of the tree is still restricted, based on the number of leaves in the tree. Typically the number of leaves is between 8 and 32

Gradient boost regression example

Example

Height (m)	Favorite color	Gender	Weight(kgs)
1.6	Blue	Male	88
1.6	Green	Female	76
1.5	Blue	Female	56
1.8	Red	Male	73
1.5	Green	Male	77
1.4	Blue	Female	57

Average weight = 71.2

Next build a tree based on the error

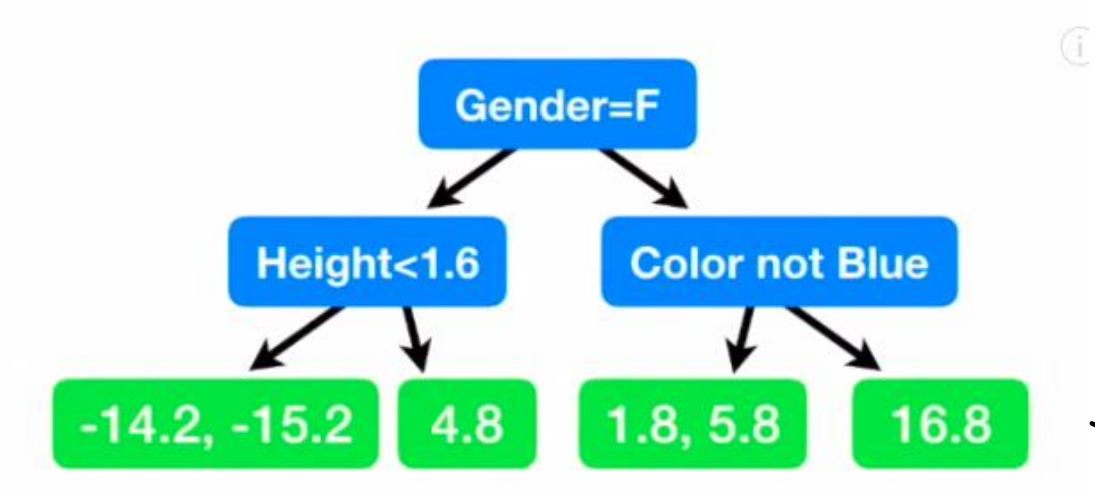
Observed weight - Predicted wt

Example

Height (m)	Favorite color	Gender	Weight(kgs)	Residual
1.6	Blue	Male	88	16.8
1.6	Green	Female	76	4.8
1.5	Blue	Female	56	-15.2
1.8	Red	Male	73	1.8
1.5	Green	Male	77	5.8
1.4	Blue	Female	57	-14.2

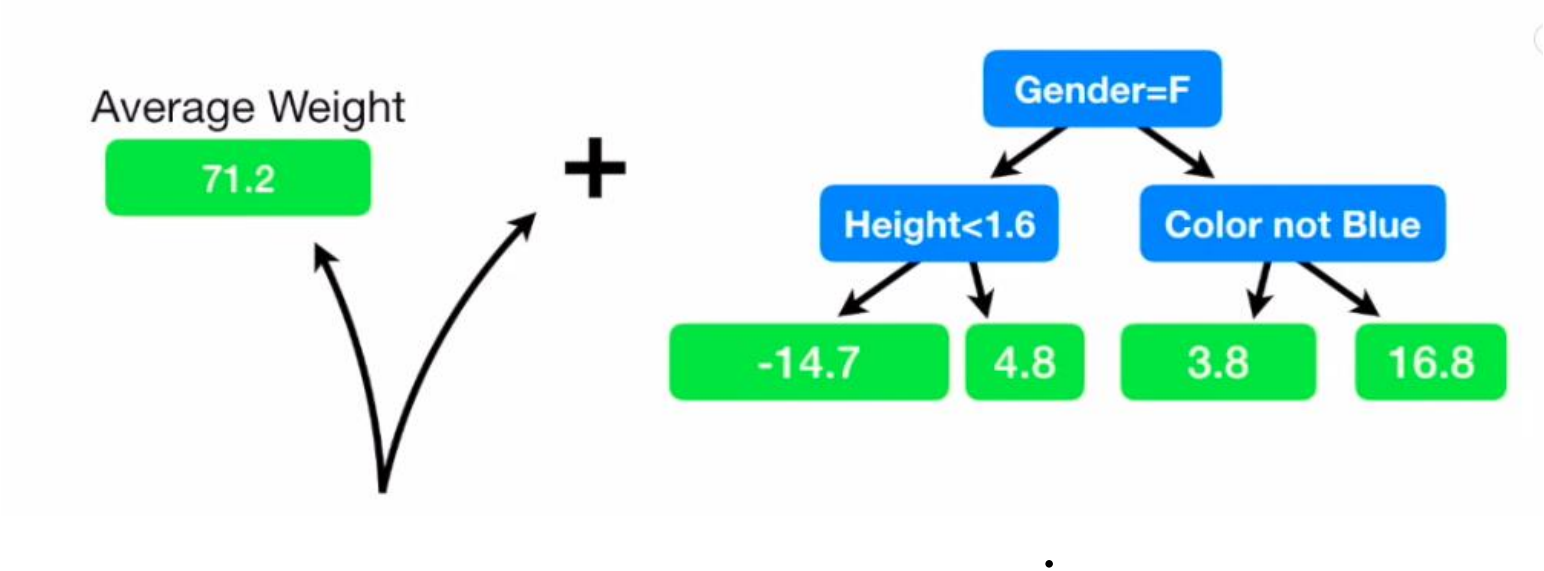
→ Now, building a tree using the given three features to build a tree. But please note the prediction is now on the Residuals

→ As usual the root node is decided based on the Gini Index.



→ The size of the tree is restricted by the no of leaves In this example it is 4

Replace residuals by average values



Perform prediction on first sample

Height (m)	Favorite color	Gender	Weight(kgs)
1.6	Blue	Male	88

$= 71.2 + (0.1)\,16.8$

$= 72.9$

New residual $= 88 - 72.9 = 15.1$

Prediction and residual for the second sample

Height (m)	Favorite color	Gender	Weight(kgs)
1.6	Green	Female	76

Prediction

$= 71.2 + 0.1 \times (4.8)$

$= 71.2 + .48$

$= 71.68$

New residual: $76 - 71.68$

$= 4.2$

New residual

Height (m)	Favorite color	Gender	Weight(kgs)	Residual	Residual
1.6	Blue	Male	88	16.8	15.1
1.6	Green	Female	76	4.8	4.2
1.5	Blue	Female	56	-15.2	-13.7
1.8	Red	Male	73	1.8	1.4
1.5	Green	Male	77	5.8	5.4
1.4	Blue	Female	57	-14.2	-12.7

actual

Predictions for Avg = 71.2

Predicted value =

Avg value + (.1 x Tree1) + (.1 x Tree2) + ...

The Gradient Boost Algorithm

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: $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

Step 2: 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 $i = 1,\ldots,n$

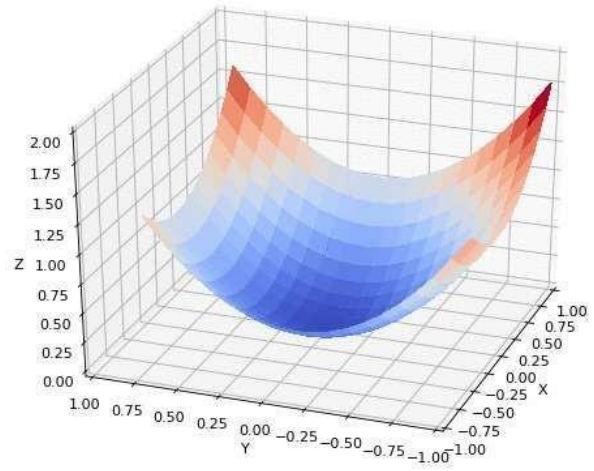
(B) Fit a regression tree to the r_{im} values and create terminal regions R_{jm}, for $j = 1 \ldots J_m$

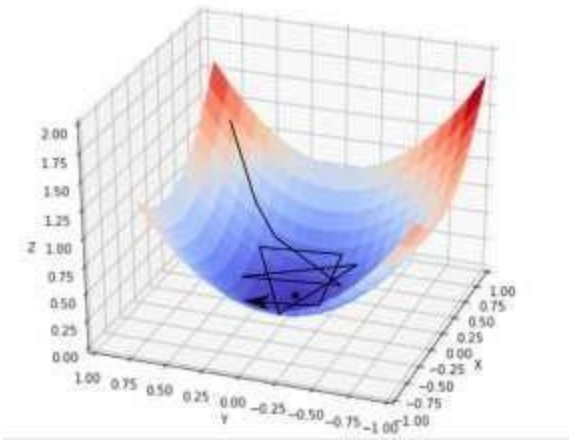
(C) For $j = 1 \ldots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$

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

Step 3: Output $F_M(x)$

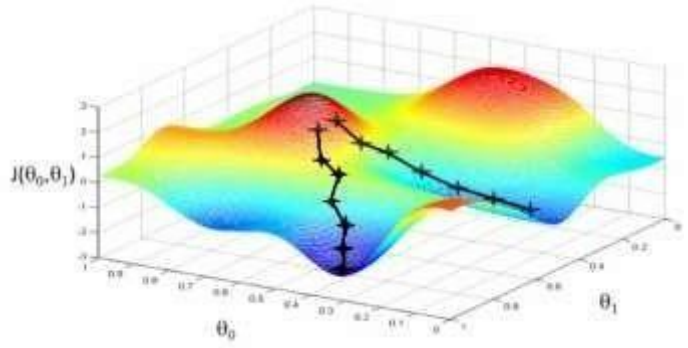
gradient descent





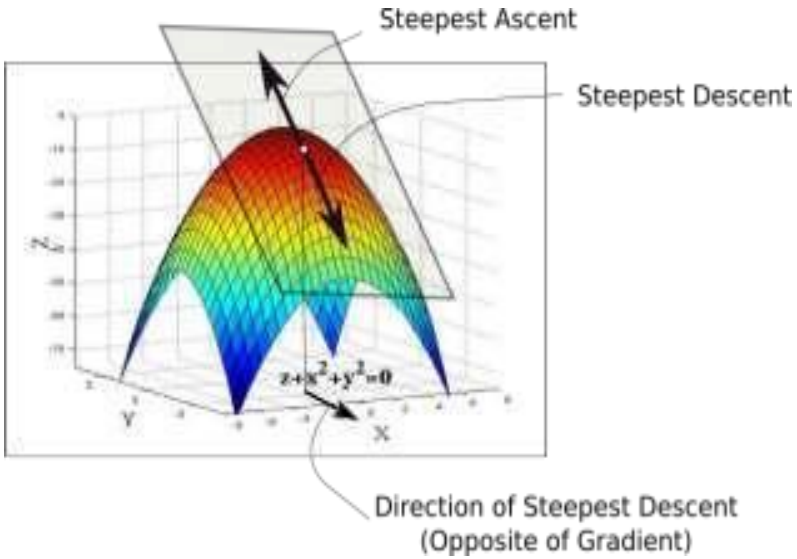
In practice, we might never *exactly* reach the minima, but we keep oscillating in a flat region in close vicinity of the minima. As we oscillate our this region, the loss is almost the minimum we can achieve, and doesn't change much as we just keep bouncing around the actual minimum.

Often, we stop our iterations when the loss values haven't improved in a pre-decided number, say, 10, or 20 iterations. When such a thing happens, we say our training has converged, or convergence has taken place.



While the direction of the gradient tells us which direction has the steepest ascent, it's magnitude tells us how steep the steepest ascent/descent is. So, atthe minima, where the contour is almost flat, you would expect the gradient to be almost zero. In fact, it's precisely zero for the point of minima.

Gradient descent is an optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient



Predicted value

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: $F_0(x) = \underset{\gamma}{\operatorname{argmin}} \sum_{i=1}^{n} L(y_i, \gamma)$

Average value = 73.3

	Height (m)	Favorite color	Gender	Weight(kgs)	
x_1	1.6	Blue	Male	88	y_1
x_2	1.6	Green	Female	76	y_2
x_3	1.5	Blue	Female	56	y_3

$n=3$

r = residual

No of trees

Derivative of loss function

Step 2: 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 $i = 1,\ldots,n$

i = sample no

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

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

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

Step 2: 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 $i = 1,\ldots,n$

Loss function = $\frac{1}{2}$ (observed - predicted)^2

$y = \frac{1}{2}(o-p)^2$

Differentiating w.r.t P, find dy/dp

Use chainrule of differentiation

let $u = o - p$ $\quad y = \frac{1}{2}(u)^2$

$\frac{du}{dp} = -1$ $\quad \frac{dy}{du} = \frac{2}{2}(u)$ $\quad \because \frac{d}{dx}(x^n) = nx^{n-1}$

$\frac{dy}{dp} = \frac{dy}{du} \times \frac{du}{dp} = u(-1)$ $\quad -0$

Substituting for value of u in 0

$\frac{dy}{dp} = -(o-p) = -(\text{observed} - \text{predicted})$

Height (m)	Favorite color	Gender	Weight(kgs)	r(i,1)
1.6	Blue	Male	88	14.7
1.6	Green	Female	76	2.7
1.5	Blue	Female	56	-17.3

$r(1,1) = 88 - 73.3 = 14.7$

$r(2,1)$

$r(3,1)$

Step 2: 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 $i = 1,\ldots,n$

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

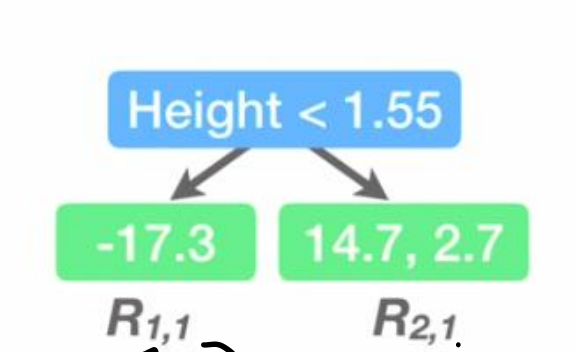
(C) For $j = 1\ldots J_m$ compute $\gamma_{jm} = \underset{\gamma}{\operatorname{argmin}} \sum_{x_i \in R_{ij}} L(y_i, F_{m-1}(x_i) + \gamma)$

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

Build a regression tree to predict the residuals instead of the wts

Step 3: Output $F_M(x)$

Build the tree using the variables Height, favorite color and gender. Again Select root node using Gini index



Tree no

Node no

Step 2: 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 $i = 1,\ldots,n$

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

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

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

Step 3: Output $F_M(x)$

Find a predicted value that min sum

o/p value for each leaf

previous prediction

Identity function

(C) → Determine o/p value for each leaf

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

Height (m)	Favorite color	Gender	Weight(kgs)	r(i,1)
1.6	Blue	Male	88	14.7
1.6	Green	Female	76	2.7
1.5	Blue	Female	56	-17.3

y_3

$$\gamma_{(1,1)} = \underset{\gamma}{\operatorname{argmin}} \frac{1}{2}\left[y_3 - \left(F_{m-1}(x_3) + \gamma\right)\right]^2$$

$$= \underset{\gamma}{\operatorname{argmin}} \frac{1}{2}\left[56 - 73.3 - \gamma\right]^2$$

$$= \underset{\gamma}{\operatorname{argmin}} \frac{1}{2}\left[-17.3 - \gamma\right]^2 \quad -(1)$$

Need to find the value of γ that minimizes (1)
so differentiate and equate to zero

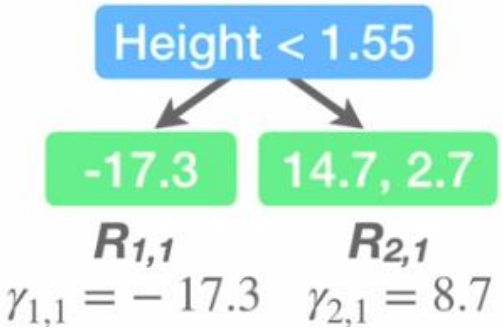
$$\frac{d}{d\gamma} \frac{1}{2}\left[-17.3 - \gamma\right]^2$$

$$= \frac{2}{2}\left[-17.3 - \gamma\right](-1)$$

$$= 17.3 + \gamma = 0$$

$$\gamma = -17.3$$

$$\therefore \gamma_{(1,1)} = -17.3$$



Similarly $\gamma_{(2,1)} = 8.7$

Step 2: 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 $i = 1,\ldots,n$

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

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

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

Learning rate

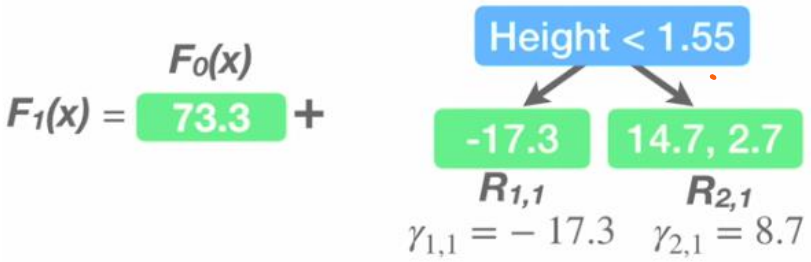
Identity function

(D) Make a new prediction for each sample

Add up the o/p values $\gamma_{j,m}$ for all the leaves that sample x_i is a part of

Step 3: Output $F_M(x)$

Height (m)	Favorite color	Gender	Weight(kgs)	F0(x)	F1(x)
1.6	Blue	Male	88	73.3	74.2
1.6	Green	Female	76	73.3	74.2
1.5	Blue	Female	56	73.3	71.6



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

$F_1(x_1) = 73.3 + 0.1\,(8.7)$

$= 74.2$

$F_1(x_2) = 73.3 + 0.1\,(8.7)$

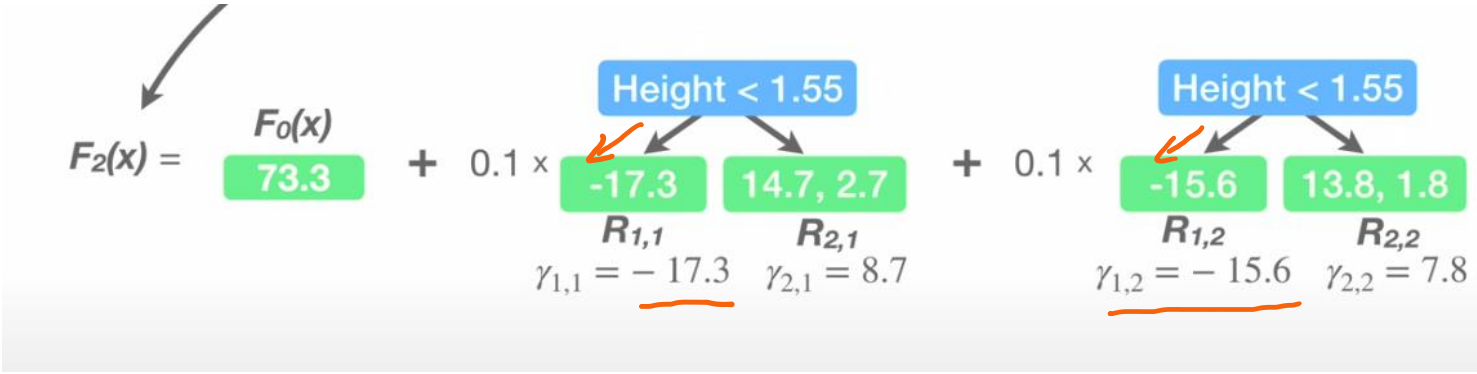
$= 74.2$

$F_1(x_3) = 73.3 + 0.1\,(-17.3)$

$= 71.6$

Step 3: Output $F_M(x)$

Perform prediction on unseen sample



Height (m)	Favorite color	Gender	Weight(kgs)
1.4	Green	Male	????

← 70

Predicted weight = $73.3 + 0.1(-17.3) + 0.1(-15.6)$

$= 70$