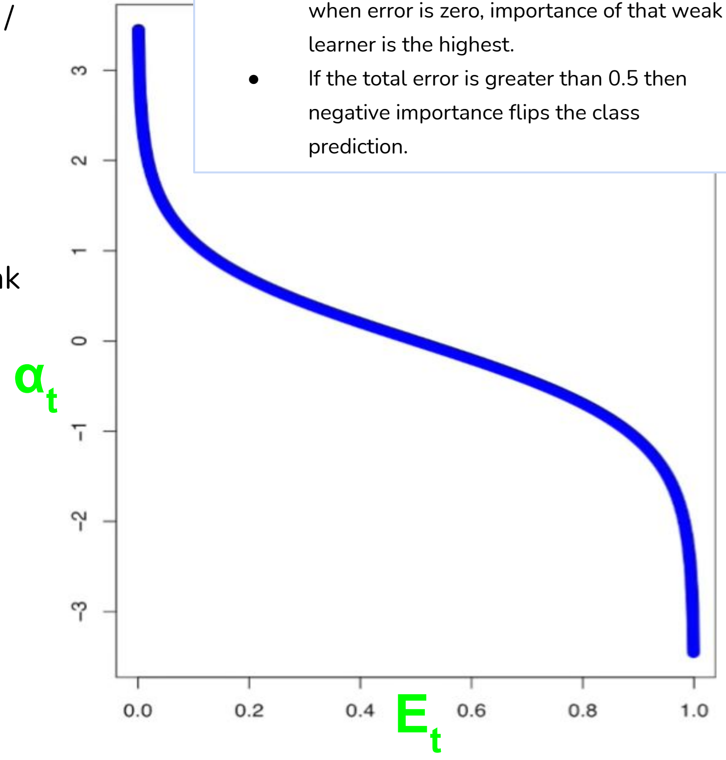
Summary - AdaBoost

1. Assign equal sample weights for each sample - sample weight = 1 / number of samples
2. We bootstrap the samples as per the weights assigned and build a weak learner on that sample
3. Once the weak learner is built, AdaBoost chooses the alpha, which measures the importance of it based on the error made by that weak learner $$\alpha_t = \frac{1}{2} log(\frac{1-\epsilon_t}{\epsilon_t})$$
4. Calculate the new sample weights for the next weak learner
   a. New sample weight for incorrect samples = sample weight * exp(alpha) / z_t
   b. New sample weight for correct samples = sample weight * exp(-alpha) / z_t
5. Create a bootstrapped dataset with the odds of each sample being chosen based on their new sample weights
6. Repeat the process n number of times
7. The final prediction is a weighted majority vote/average of all the weak learners



Importance VS Error of each learner

- As evident from the graph, importance of each weak learner decreases with the increase in error made by that learner, that is when error is zero, importance of that weak learner is the highest.
- If the total error is greater than 0.5 then negative importance flips the class prediction.

Summary AdaBoost

$$\alpha = (½) \log((1 - E_t) / E_t)$$

Incorrect => New weight = (old weight) * (e^{α}) / z

Correct => New weight = (old weight) * ($e^{-\alpha}$) / z

Gradient Boosting - Classification

Let's understand this with an example

Let's consider an example dataset, where X is assumed to be the age of people and Y is whether they like a particular movie or not.

X	Y	Pred
10	0	0.67
20	1	0.67
30	1	0.67
40	1	0.67
50	0	0.67
60	1	0.67

- Let's find log odds for Y =1
- log(4/2) = 0.69
- Let's find out the probability using the below formula :

$$P = \frac{e^{log(odds)}}{1+e^{log(odds)}}$$

- P(Y = 1) = (e^(0.69))/(1 + e^(0.69)) = 0.67

$P(Y=1) = \frac{4}{6}$

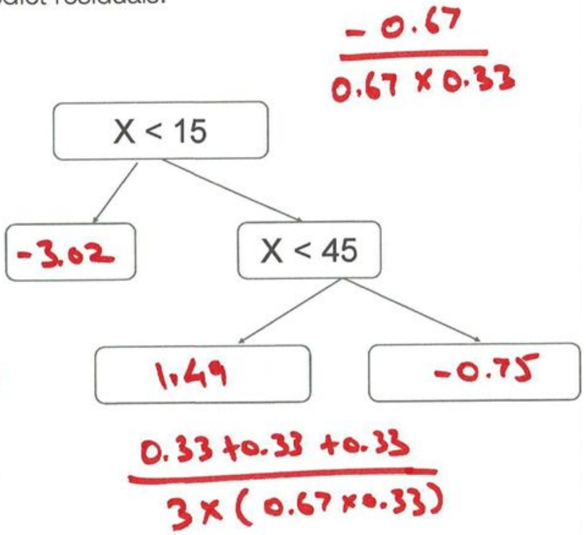
$\rightarrow \boxed{m_1} \rightarrow$

Gradient Boosting - Classification

Let's calculate residuals for our predictions

We will build a tree with max_depth = 2 to predict residuals.

X	Y	Residual (observed-predicted probability)
10	0	-0.67
20	1	0.33
30	1	0.33
40	1	0.33
50	0	-0.67
60	1	0.33



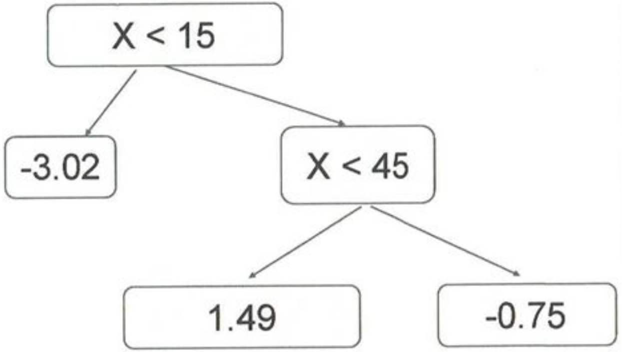
Gradient Boosting - Classification

Transformation formula

We can calculate the output value of each leaf using the following formula:

$$\frac{\sum Residual}{\sum[PreviousProb * (1-PreviousProb)]}$$

X	Y	Residual
10	0	-0.67
20	1	0.33
30	1	0.33
40	1	0.33
50	0	-0.67
60	1	0.33



Gradient Boosting - Classification

Great Learning
POWER AHEAD

Update Predictions

We'll update our predictions using below formula with a learning rate of 0.8

$$OldTree + LearningRate * NewTree$$

$0.69 + (0.8)(-3.02)$

X	Y	Residual	Y1 = Previous(log odds) + (LR * New values)	Output1 = New predictions (e^Y1) / (1 + e^Y1)
10	0	-0.67	-1.72	0.15
20	1	0.33	1.88	0.87
30	1	0.33	1.88	0.87
40	1	0.33	1.88	0.87
50	0	-0.67	0.09	0.52
60	1	0.33	0.09	0.52

$P = \frac{e^{\square}}{1+e^{\square}}$

R

-0.15
0.13
0.13
0.13
-0.52
0.48

$0.69 + (0.8)(-0.75)$

$0.69 + (0.8)(1.49)$

Gradient Boosting - Classification

$y = a + b x$

$Log (p / (1-p)) = a + b x$

$$P = e^{\boxed{}} / (1 + e^{\boxed{}}) = (P / 1-P) / (1+ (P / 1-P))$$

$$= (P / (1-P)) / (1 / (1 - P))$$

$$= P$$

Gradient Boosting - Classification Loss Function

$$\log(\text{odds}) = \log(P_i / (1-P_i))$$

$$\log(\text{likelihood}) = y_i \log(p_i) + (1 - y_i) \log(1 - p_i)$$

$$d / d(\underline{\log(\text{odds})})\ [\ y_i\ \underline{\log(\text{odds})} - \log(1 + e^{\underline{\log(\text{odds})}})\ )$$

$$y_i - e^{\log(\text{odds})} / (1 + e^{\log(\text{odds})}) = y_i - p_i$$

Gradient Boosting - Regression Loss Function

$\boxed{(\tfrac{1}{2})\,(y-y_0)^2}$ **Loss function**

$\downarrow$

Negative gradient $\longrightarrow$ $(\,y - y_0\,)$