Boosting

Boosting is part of ensemble techniques to improve prediction result and can be applied th any machine learning model , It tackle the high bias unlike Bagging which focuses on averaging multiple high variance models.

Weak Learners : a model is consider a weak learner if it's error rate is slightly less than randomly guessing (very high bias/underfit)

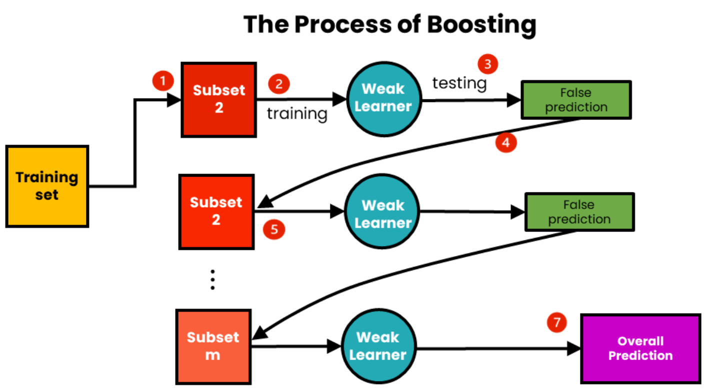
$$\text{error rate} \approx 0.45 - 4.9$$

Where :

- An $\text{error rate} \leq 0.3$ is a **Strong Learner**
- An $\text{error rate} = 0.5$ is a **Useless Learner** (randomly guessing)

Boosting idea :

Before Showing Formulas, The main idea of **Boosting** is **Sequentially** using those weak learners to get a **Strong Learner** by :



Now we Focus on **misclassified data** of the weak learner by:

- Increase the weights of misclassified data $\rightarrow$ Weighted Training
- Update the weights
- Train the next **Weak Learner** on the updated weights data
  - So now the rows that the first weak learner got wrong got their weight increase which means the next weak learner will focus on them

Important note : Weights in this context means importance of **Samples/rows** so a weight w_1 is the importance score of a single observation(row) x_1, Don't get it confused with coefficients and weights in Linear Regression , Neural Nets...

AdaBoost (Adaptive Boosting)

One of the most widely used **boosting** techniques, let's represent the algorithm steps and discuss it :

Algorithm AdaBoost.

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 $\alpha_m = \log\big((1 - \text{err}_m)/\text{err}_m\big)$.

   (d) Set $w_i \leftarrow w_i \cdot \exp\big[\alpha_m \cdot I(y_i \neq G_m(x_i))\big],\ i = 1, 2, \ldots, N$.

3. Output $G(x) = \text{sign}\Big[\sum_{m=1}^{M} \alpha_m G_m(x)\Big]$.

So After fitting the **Binary Classifier** $G_m(x)$, we compute the error rate associated with that model.

$$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}$$

- Here **misclassified** data $I(y_i \neq G_m(x))$ get multiplied by their weight so they contribute more to the error rate

$$\alpha_m = \log\big((1 - \text{err}_m)/\text{err}_m\big).$$

- After it we compute α_m which is the **classifier** G_m weight also can think of it as **voting power
  - Models with **error rate** has higher α and their votes are more of **importance**, which we need in the last step

$$w_i \leftarrow w_i \cdot \exp\big[\alpha_m \cdot I(y_i \neq G_m(x_i))\big],\ i = 1, 2, \ldots, N.$$

- Update the weights(importance) for each misclassified observation(row), the misclassified rows weights are multiplied by $\exp[\alpha_m]$

$$G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$$

- The final step is **Majority Vote**, Since it's a **binary classification** setting it's either $[-1, 1]$ each weak learner $G_m(x)$ is multiplied by "how important they are" **Classifier weight** α_m

Forward Stage-wise additive modeling

It's the framework behind boosting and here we will proof starting from the base algorithm to **Ada boost** , The main idea is instead of training the full **complex** model , we sequentially add smaller models .

Forward Stagewise additive modeling algorithm

1. Initialize $f_0(x) = 0$

2. for $m = 1 \ldots M$

   $(a).$ Compute$(\beta_m, h_m(x)) = \text{argmin} \sum_i L(y_i, f_{m-1}(x_i) + \beta_m h_m(x))$

   $(b).$ Set $f_m(x) = f_{m-1}(x) + \beta_m h_m(x)$

- With $h(x)$ being a basis function, see Basis Functions and β the coefficient for the model
- $L()$ is the loss function

This algorithm can be used in any model from linear regression to Tress and neural nets, in the case of **Ada boost** we use the **exponential loss** as a loss function (reason will be stated later) and basis function $h(x) = G(x)$ which are weak learners defined above.

The exponential loss function is :

$$L(y, f(x)) = e^{-yf(x)}$$

Applying the **forward stage-wise** (a):

$$(\beta_m, G_m) = \text{argmin} \sum_i^n \exp\Big[y_i(f_{m-1}(x) + \beta_m G_m(x_i)\Big]$$

Sharing the y_i on the terms we get :

$$\text{argmin} \sum_i^n \exp[y_i f_{m-1}(x_i)] \; \exp[y_i \beta_m G_m(x_i)]$$

- Where $w_i = \exp[y_i f_{m-1}(x_i)]$ from exponential loss.

$$\text{argmin} \sum_i^n w_i \exp[y_i \beta_m G_m(x_i)]$$

Since $G_m(x) = \{-1, 1\}$ we get the following :

$$\text{argmin } e^{-\beta}$$