

Boosting

Boosting is part of ensemble techniques to improve prediction result and can be applied to 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)

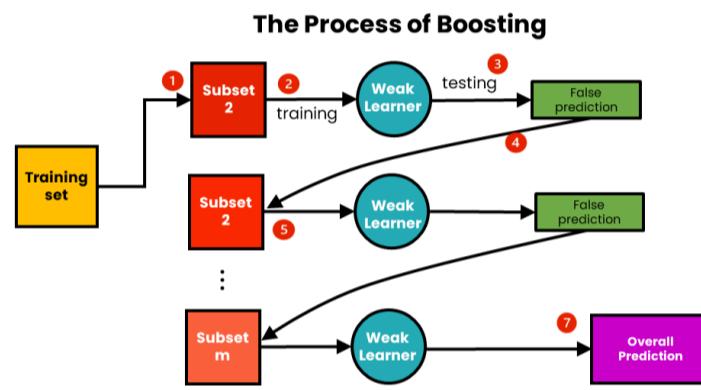
$$\text{error rate} \approx 0.45 - 4.9$$

Where :

- An error rate ≤ 0.3 is a **Strong Learner**
- An 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 :

1. Initialize the observation weights $w_i = 1/N$, $i = 1, 2, \dots, N$.
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 $w_i \leftarrow \frac{w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]}{Z_m}$, $i = 1, 2, \dots, N$
3. Output $G(x) = \text{sign}\left[\sum_{m=1}^M \alpha_m G_m(x)\right]$.

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

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

- 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

$$(d) \text{ Set } w_i \leftarrow \frac{w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]}{Z_m}, i = 1, 2, \dots, N$$

- 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 $m = 1 \dots M$

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

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

- 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):

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

Sharing the y_i on the terms we get :

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

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

$$\underset{i}{\operatorname{argmin}} \sum_i^n w_i^{(m)} \exp[-y_i \beta_m G_m(x_i)]$$

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

$$\operatorname{argmin} e^{-\beta} \sum_{i:y_i=G(x_i)} w_i + e^{\beta} \sum_{i:y_i \neq G(x_i)} w_i$$

Given that :

$$\sum_{i:y_i=G(x_i)} w_i = W - \sum_{i:y_i \neq G(x_i)} w_i$$

- With W being the **sum of all weights** for classified and misclassified samples

By substituting we get :

$$\begin{aligned} & \operatorname{argmin} e^{-\beta}(W - \sum_{i:y_i \neq G(x_i)} w_i) + e^{\beta} \sum_{i:y_i \neq G(x_i)} w_i \\ & \operatorname{argmin} e^{-\beta}W - e^{-\beta} \sum_{i:y_i \neq G(x_i)} w_i + e^{\beta} \sum_{i:y_i \neq G(x_i)} w_i \end{aligned}$$

Factoring $\sum_{y_i \neq G(x_i)} w_i$:

$$\operatorname{argmin} e^{-\beta}W + (e^{\beta} - e^{-\beta}) \sum_{i:y_i \neq G(x_i)} w_i$$

Remember that we are minimizing for (β_m, G_m) , first let's minimize for G_m

$$\operatorname{argmin} \sum_{i:y_i \neq G(x_i)} w_i$$

- Since $e^{-\beta}W$ is a constant
- $(e^{\beta} - e^{-\beta})$ can be neglected when minimizing

Can also be written :

$$\operatorname{argmin} \sum_i^N w_i^{(m)} I(y_i \neq G(x_i))$$

Thus :

$$G_m = \operatorname{argmin} \sum_i^N w_i^{(m)} I(y_i \neq G(x_i))$$

Now the weighted error will be : (notice that is the same as in the Ada boost algorithm above)

$$\operatorname{err}_m = \frac{\sum_i^N w_i^{(m)} I(y_i \neq G(x_i))}{W}$$

Which allow us to write the weights for the classified and misclassified samples as :

$$\sum_{i:y_i \neq G(x_i)} w_i = \operatorname{err}_m W$$

$$\sum_{i:y_i=G(x_i)} w_i = (1 - \operatorname{err}_m) W$$

Now plugging those into this expression to minimize β_m

$$\operatorname{argmin} e^{-\beta} \sum_{i:y_i=G(x_i)} w_i + e^{\beta} \sum_{i:y_i \neq G(x_i)} w_i$$

We get :

$$\operatorname{argmin} e^{-\beta}(1 - \operatorname{err}_m) W + e^{\beta} \operatorname{err}_m W$$

Taking the derivative w.r.t. β :

$$\frac{d}{d\beta} = -e^{-\beta}(1 - \operatorname{err}_m) W + e^{\beta} \operatorname{err}_m W = 0$$

$$\beta = \frac{1}{2} \log \left(\frac{1 - \text{err}_m}{\text{err}_m} \right)$$

Which is the same as the **Classifier weight** in Ada boost

Now we got at the last step in the forward stage-wise algorithm :

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

With the weight at a step m is :

$$w_i^{(m)} = e^{-y_i f_{m-1}(x_i)}$$

and the weight for the next step is :

$$\begin{aligned} w_i^{(m+1)} &= e^{-y_i(f_{m-1}(x) + \beta_m G_m(x_i))} = e^{-y_i f_{m-1}(x)} \cdot e^{-y_i \beta_m G_m(x_i)} \\ w_i^{(m+1)} &= w_i^{(m)} \cdot e^{-y_i \beta_m G_m(x_i)} \end{aligned}$$

Knowing that :

- When $y_i = G_m(x_i) \rightarrow y_i G_m(x_i) = 1$ and $I(y_i \neq G_m(x_i)) = 0$ (correct classification)
- When $y_i \neq G_m(x_i) \rightarrow y_i G_m(x_i) = -1$ and $I(y_i \neq G_m(x_i)) = 1$ (wrong classification)

Now we want a formula that gives:

- -1 when $I(y \neq G_m) = 0$ (we want -1 cause $-y_i G(x_i)_m$ so it result in 1)
- 1 when $I(y \neq G_m) = 1$

$$-y_i G(x_i)_m = 2 \cdot I(y_i \neq G(x_i)_m) - 1$$

Note : It might be a bit confusing but try write it and check

Now we plug the formula into the weight formula above :

$$\begin{aligned} w_i^{(m+1)} &= w_i^{(m)} \cdot e^{\beta_m (-y_i G_m(x_i))} \\ w_i^{(m+1)} &= w_i^{(m)} \cdot e^{2\beta_m I(y_i \neq G_m(x_i)) - 1} \\ w_i^{(m+1)} &= w_i^{(m)} \cdot e^{2\beta_m I(y_i \neq G_m(x_i))} \cdot e^{-\beta_m} \end{aligned}$$

With $\alpha = 2\beta$

$$w_i^{(m+1)} = w_i^{(m)} \cdot e^{\alpha I(y_i \neq G_m(x_i))} \cdot e^{-\beta_m}$$

Now we minimize for β we get :

$$w_i^{(m+1)} = w_i^{(m)} \cdot e^{\alpha I(y_i \neq G_m(x_i))}$$

- e^{β_m} is dropped since this value is multiplied by all the weights

Resulting in the same expression used in the **Ada Boost**:

$$(d) \text{ Set } w_i \leftarrow \frac{w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]}{Z_m}, i = 1, 2, \dots, N$$

Concluding that **Ada Boost**