Boosting

Statistical Machine Learning (ENGG*6600*08)

School of Engineering,
University of Guelph, ON, Canada

Course Instructor: Benyamin Ghojogh
Fall 2023

Definition

Definition

- Boosting is a **meta algorithm** which can be used with any model (classifier, regression, etc).
- For binary classification, for example, if we use boosting with a classifier even slightly better than flipping a coin, we will have a strong classifier (we will explain the reason later). Thus, we can say boosting makes the estimation or classification very strong. In other words, boosting addresses the question whether **a strong classifier can be obtained from a set of weak classifiers** [1, 2].

Definition

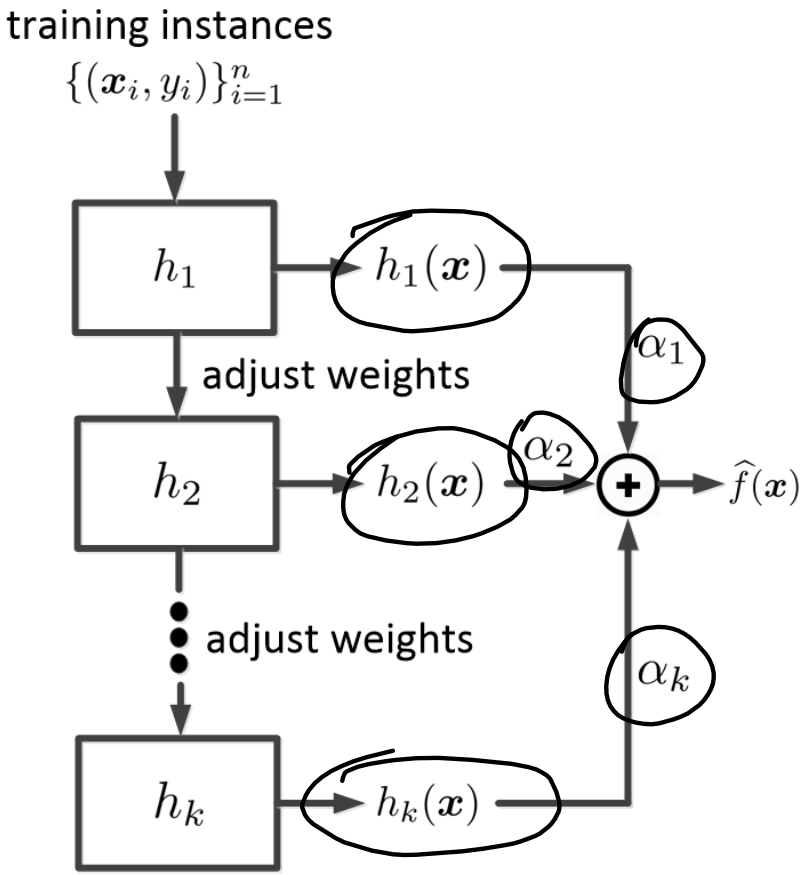
- The idea of boosting is to learn k models in a hierarchy where every model gives more attention (larger weight) to the instances misclassified (or estimated very badly) by the previous model.
- Finally, the overall estimation or classification is a weighted summation (average) of the k estimations. For an instance $\boldsymbol{x}$, we have:

$$\widehat{f}(\boldsymbol{x}) = \sum_{j=1}^{k} \alpha_j \, h_j(\boldsymbol{x}). \tag{1}$$

If the model is classifier, we should probably use sign function:

$$\widehat{f}(\boldsymbol{x}) = \text{sign}\Big(\sum_{j=1}^{k} \alpha_j h_j(\boldsymbol{x})\Big), \tag{2}$$

which is equivalent to **majority voting** among the trained classifiers.



Definition

- Different methods have been proposed for boosting, one of the most well-known ones is **AdaBoost (Adaptive Boosting)** (1996) [3].
- The algorithm of AdaBoost for binary classification is shown below.

Initialize $w_i = 1/n, \forall i \in \{1, \ldots, n\}$
for j *from* 1 *to* k **do**
  $h_j(\boldsymbol{x}) = \arg\min L_j$
  $\alpha_j = \log(\frac{1-L_j}{L_j})$
  $w_i = w_i \exp\left(\alpha_j \mathbb{I}(y_i \neq h_j(\boldsymbol{x}_i))\right)$

Algorithm : The AdaBoost Algorithm

- In this algorithm, L_j is the cost function minimized in the j-th model h_j, the $\mathbb{I}(.)$ is the indicator function which is one and zero if its condition is and is not satisfied, respectively, and w_i is the weight associated to the i-th instance for weighting it as the input to the next layer of boosting.
- Note that the cost in AdaBoost is:

$$L_j = \frac{\sum_{i=1}^{n} w_i \, \mathbb{I}(y_i \neq h_j(\boldsymbol{x}_i))}{\sum_{i=1}^{n} w_i}, \tag{3}$$

which makes sense because it gets larger if the observations of more instances are estimated incorrectly.

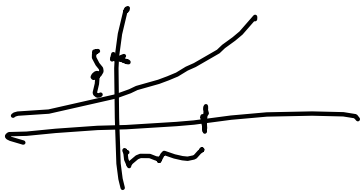
Definition

1 **Initialize** $w_i = 1/n, \forall i \in \{1, \ldots, n\}$
2 **for** j *from* 1 *to* k **do**
3 $\quad h_j(\boldsymbol{x}) = \arg\min L_j$
4 $\quad \alpha_j = \log(\frac{1-L_j}{L_j})$
5 $\quad w_i = w_i \exp\left(\alpha_j \mathbb{I}(y_i \neq h_j(\boldsymbol{x}_i))\right)$

Algorithm : The AdaBoost Algorithm

- Here, we can have several cases which help us understand the interpretation of the AdaBoost algorithm:
- If an instance is correctly classified, the $\mathbb{I}(y_i \neq h_j(\boldsymbol{x}_i))$ is zero and thus the w_i will be still w_i without any change. This makes sense because the correctly classified instance should not gain a significant weight in the next layer of boosting.

Definition



Initialize $w_i = 1/n, \forall i \in \{1, \ldots, n\}$
for j *from* 1 *to* k **do**
  $h_j(\boldsymbol{x}) = \arg\min L_j$
  $\alpha_j = \log(\frac{1-L_j}{L_j})$
  $w_i = w_i \exp(\alpha_j \mathbb{I}(y_i \neq h_j(\boldsymbol{x}_i)))$

Algorithm : The AdaBoost Algorithm

$L_j > 0.5$

$\frac{1-L_j}{L_j} < 1$

- If an instance is misclassified, the $\mathbb{I}(y_i \neq h_j(\boldsymbol{x}_i))$ is one. In this case, we can have two sub-cases:
  - If the classifier which classified that instance was a bad classifier, its cost would be like flipping a coin, i.e., $L_j = 0.5$. Therefore, we will have $\alpha_j = \log(1) = 0$ and again the w_i will still be w_i without any change. This makes sense because we cannot trust the bad classifier whether the instance is correctly or incorrectly classified and thus we should not make any decision based on that.
  - If the classifier which classified that instance was a good classifier, then we have $L_j \neq 0.5$ and as we also have $\mathbb{I}(y_i \neq h_j(\boldsymbol{x}_i)) = 1$, the weight will change as $w_i := w_i \exp(\alpha_j)$. This also is intuitive because the previous model in the boosting was a good classifier and we can trust it and that good classifier could not classify the instance correctly. Therefore, we should notice that instance more in the next model in the boosting hierarchy.

Theory Based on Additive Models

Theory Based on Additive Models

- **Additive models** [4] can be used to explain why boosting works [5, 6]. In an additive model, we map the data as $\boldsymbol{x} \mapsto \phi_j(\boldsymbol{x}), \forall j \in \{1, \ldots, k\}$ and then add them using some weights β_j's:

$$\phi(\boldsymbol{x}) = \sum_{j=1}^{k} \beta_j \, \phi_j(\boldsymbol{x}). \tag{4}$$

- A well-known example of the additive model is Radial Basis Function (RBF) neural network (here with k hidden nodes) which uses Gaussian mappings [7, 8].
- Now, consider a cost function for an instance as:

$$L(y, h(\boldsymbol{x})) := \exp(-y \, h(\boldsymbol{x})), \tag{5}$$

where y is the observation or label for $\boldsymbol{x}$ and $h(\boldsymbol{x})$ is the model's estimation of y. This cost is intuitive because when the instance is misclassified, the signs of y and $h(\boldsymbol{x})$ will be different (so $y \, h(\boldsymbol{x}) < 0$) and the cost will be large, while in case of correct classification, the signs are similar (so $y \, h(\boldsymbol{x}) > 0$) and the cost is small.

- If we add up the cost over the n training instances, we have:

$$L_t(y, h(\boldsymbol{x})) := \sum_{i=1}^{n} \exp(-y_i \, h(\boldsymbol{x}_i)), \tag{6}$$

where L_t denotes the total cost.

Theory Based on Additive Models

- In Eq. (4), $\phi(\boldsymbol{x}) = \sum_{j=1}^{k} \beta_j \, \phi_j(\boldsymbol{x})$, if we rename the mapping to $h(\boldsymbol{x})$, which is the model used in boosting, we will have:

$$h(\boldsymbol{x}) = \sum_{j=1}^{k} \beta_j \, h_j(\boldsymbol{x}). \tag{7}$$

- We can write this expression as a **forward stage-wise additive model** [5, 6] in which we work with the models one by one where we add up the previously worked models:

$$f_{q-1}(\boldsymbol{x}) = \sum_{j=1}^{q-1} \beta_j \, h_j(\boldsymbol{x}), \tag{8}$$

$$f_q(\boldsymbol{x}) = f_{q-1}(\boldsymbol{x}) + \beta_q \, h_q(\boldsymbol{x}), \quad q \leq k, \tag{9}$$

where $h(\boldsymbol{x}) = f_k(\boldsymbol{x})$.

- Therefore, minimizing the cost, i.e., Eq. (6), $L_t(y, h(\boldsymbol{x})) := \sum_{i=1}^{n} \exp(-y_i \, h(\boldsymbol{x}_i))$, for the j-th model in the additive manner is:

$$\min_{\beta_j, h_j} \sum_{i=1}^{n} \exp\left(-y_i \left[f_{j-1}(\boldsymbol{x}_i) + \beta_j \, h_j(\boldsymbol{x}_i)\right]\right) = \min_{\beta_j, h_j} \sum_{i=1}^{n} \exp(-y_i \, f_{j-1}(\boldsymbol{x}_i)) \exp(-y_i \, \beta_j \, h_j(\boldsymbol{x}_i)).$$

Theory Based on Additive Models

- We had:

$$\min_{\beta_j, h_j} \sum_{i=1}^{n} \exp\left(-y_i \left[f_{j-1}(\boldsymbol{x}_i) + \beta_j\, h_j(\boldsymbol{x}_i)\right]\right) = \min_{\beta_j, h_j} \sum_{i=1}^{n} \exp(-y_i\, f_{j-1}(\boldsymbol{x}_i)) \exp(-y_i\, \beta_j\, h_j(\boldsymbol{x}_i)).$$

- The first term is a constant with respect to β_j and h_j so we name it by w_i:

$$w_i := \exp(-y_i\, f_{j-1}(\boldsymbol{x}_i)). \tag{10}$$

Thus:

$$\min_{\beta_j, h_j} \sum_{i=1}^{n} w_i \exp(-y_i\, \beta_j\, h_j(\boldsymbol{x}_i)).$$

- As in binary AdaBoost, we have ± 1 for y_i and h_j, we can say:

$$\min_{\beta_j, h_j} \exp(-\beta_j) \sum_{i=1}^{n} w_i\, \mathbb{I}(y_i = h_j(\boldsymbol{x}_i)) + \exp(\beta_j) \sum_{i=1}^{n} w_i\, \mathbb{I}(y_i \neq h_j(\boldsymbol{x}_i))$$

$$\overset{(a)}{=} \min_{\beta_j, h_j} \exp(-\beta_j) \sum_{i=1}^{n} w_i - \exp(-\beta_j) \sum_{i=1}^{n} w_i\, \mathbb{I}(y_i \neq h_j(\boldsymbol{x}_i)) + \exp(\beta_j) \sum_{i=1}^{n} w_i\, \mathbb{I}(y_i \neq h_j(\boldsymbol{x}_i)),$$

where (a) is because:

$$\sum_{i=1}^{n} w_i\, \mathbb{I}(y_i = h_j(\boldsymbol{x}_i)) = \sum_{i=1}^{n} w_i - \sum_{i=1}^{n} w_i\, \mathbb{I}(y_i \neq h_j(\boldsymbol{x}_i)).$$

Theory Based on Additive Models

- We had:

$$\min_{\beta_j, h_j} \exp(-\beta_j) \sum_{i=1}^{n} w_i - \exp(-\beta_j) \sum_{i=1}^{n} w_i \mathbb{I}(y_i \neq h_j(\boldsymbol{x}_i)) + \exp(\beta_j) \sum_{i=1}^{n} w_i \mathbb{I}(y_i \neq h_j(\boldsymbol{x}_i)).$$

- For the sake of minimization, we take the derivative:

$$\frac{\partial L_t}{\partial \beta_j} = -\exp(-\beta_j) \sum_{i=1}^{n} w_i + \exp(-\beta_j) \sum_{i=1}^{n} w_i \mathbb{I}(y_i \neq h_j(\boldsymbol{x}_i)) + \exp(\beta_j) \sum_{i=1}^{n} w_i \mathbb{I}(y_i \neq h_j(\boldsymbol{x}_i)) \stackrel{\text{set}}{=} 0,$$

which gives:

$$\begin{aligned}
&\implies (\exp(-\beta_j) + \exp(\beta_j)) \times \frac{\sum_{i=1}^{n} w_i \mathbb{I}(y_i \neq h_j(\boldsymbol{x}_i))}{\sum_{i=1}^{n} w_i} = \exp(-\beta_j) \\
&\stackrel{(3)}{\implies} (\exp(-\beta_j) + \exp(\beta_j)) L_j = \exp(-\beta_j) \\
&\implies L_j = \frac{\exp(-\beta_j)}{\exp(-\beta_j) + \exp(\beta_j)} \implies \exp(2\beta_j) = \frac{1 - L_j}{L_j} \implies 2\beta_j = \log(\frac{1 - L_j}{L_j}) \\
&\stackrel{(a)}{\implies} \alpha_j = 2\beta_j,
\end{aligned} \tag{11}$$

where (a) is because of the formula of α_j in the Algorithm.

Theory Based on Additive Models

- According to Eqs. (8), (9), and (10),

$$\begin{cases} f_{q-1}(\boldsymbol{x}) = \sum_{j=1}^{q-1} \beta_j \, h_j(\boldsymbol{x}), \\ f_q(\boldsymbol{x}) = f_{q-1}(\boldsymbol{x}) + \beta_q \, h_q(\boldsymbol{x}), \quad q \leq k, \\ w_i := \exp(-y_i \, f_{j-1}(\boldsymbol{x}_i)). \end{cases}$$

we have:

$$w_i := w_i \, \exp(-y_i \, \beta_j \, h_j(\boldsymbol{x}_i)). \tag{12}$$

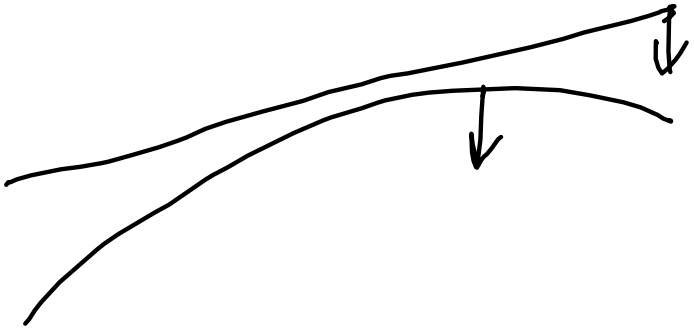
As we have $y_i \, h_j(\boldsymbol{x}_i) = \pm 1$, we can say:

$$-y_i \, h_j(\boldsymbol{x}_i) = 2 \, \mathbb{I}(y_i \neq h_j(\boldsymbol{x}_i)) - 1. \tag{13}$$

According Eqs. (11), $\alpha_j = 2\beta_j$, (12), and (13), we have:

$$w_i := w_i \, \exp\left(\alpha_j \, \mathbb{I}(y_i \neq h(\boldsymbol{x}_i))\right) \exp(-\beta_j), \tag{14}$$

which is equivalent to the formula of w_i in the Algorithm with a factor of $\exp(-\beta_j)$. This factor does not have impact on whether the instance is correctly classified or not.



Upper Bound on the Generalization Error of Boosting

Upper Bound on the Generalization Error of Boosting

- There is an **upper bound on the generalization error** of boosting [9]. In binary boosting, we have ± 1 for y_i and also the sign of $\widehat{f}(\boldsymbol{x}_i)$ is important; therefore, $y_i\,\widehat{f}(\boldsymbol{x}_i) < 0$ means that we have error for estimating the i-th instance. Thus, for an error, we have:

$$y_i\,\widehat{f}(\boldsymbol{x}_i) \leq \theta, \tag{15}$$

for a $\theta > 0$. Recall the Eq. (1):

$$\widehat{f}(\boldsymbol{x}) = \sum_{j=1}^{k} \alpha_j\, h_j(\boldsymbol{x}).$$

- We can normalize this equation because the sign of it is important:

$$\widehat{f}(\boldsymbol{x}_i) = \frac{\sum_{j=1}^{k} \alpha_j\, h_j(\boldsymbol{x}_i)}{\sum_{j=1}^{k} \alpha_j}. \tag{16}$$

- According to Eqs. (15) and (16), we have:

$$y_i\,\widehat{f}(\boldsymbol{x}_i) \leq \theta \iff y_i \sum_{j=1}^{k} \alpha_j\, h_j(\boldsymbol{x}_i) \leq \theta \sum_{j=1}^{k} \alpha_j$$

$$\iff -y_i \sum_{j=1}^{k} \alpha_j\, h_j(\boldsymbol{x}_i) + \theta \sum_{j=1}^{k} \alpha_j \geq 0 \iff \exp\Big(-y_i \sum_{j=1}^{k} \alpha_j\, h_j(\boldsymbol{x}_i) + \theta \sum_{j=1}^{k} \alpha_j\Big) \geq 1.$$

Upper Bound on the Generalization Error of Boosting

- We found:

$$y_i \widehat{f}(\boldsymbol{x}_i) \leq \theta \iff \exp\Big(- y_i \sum_{j=1}^{k} \alpha_j \, h_j(\boldsymbol{x}_i) + \theta \sum_{j=1}^{k} \alpha_j \Big) \geq 1.$$

- Therefore, in terms of probability, we have:

$$\mathbb{P}(y_i \widehat{f}(\boldsymbol{x}_i) \leq \theta) = \mathbb{P}\Big(\exp\Big(- y_i \sum_{j=1}^{k} \alpha_j \, h_j(\boldsymbol{x}_i) + \theta \sum_{j=1}^{k} \alpha_j \Big) \geq 1 \Big). \tag{17}$$

- According to the Markov's inequality which is (for $a > 0$ and a random variable X):

$$\mathbb{P}(X \geq a) \leq \frac{\mathbb{E}(X)}{a}, \tag{18}$$

and Eq. (17), we have (take $a = 1$ and the exponential term as X in Markov's inequality):

$$\mathbb{P}(y_i \widehat{f}(\boldsymbol{x}_i) \leq \theta) = \mathbb{P}\Big(\exp\Big(- y_i \sum_{j=1}^{k} \alpha_j \, h_j(\boldsymbol{x}_i) + \theta \sum_{j=1}^{k} \alpha_j \Big) \geq 1 \Big)$$

$$\overset{(18)}{\leq} \mathbb{E}\Big(\exp\Big(- y_i \sum_{j=1}^{k} \alpha_j \, h_j(\boldsymbol{x}_i) + \theta \sum_{j=1}^{k} \alpha_j \Big) \Big)$$

Upper Bound on the Generalization Error of Boosting

- Continuing:

$$\mathbb{P}(y_i \widehat{f}(\boldsymbol{x}_i) \leq \theta) = \mathbb{P}\Big(\exp\big(- y_i \sum_{j=1}^{k} \alpha_j \, h_j(\boldsymbol{x}_i) + \theta \sum_{j=1}^{k} \alpha_j \big) \geq 1 \Big)$$

$$\overset{(18)}{\leq} \mathbb{E}\Big(\exp\big(- y_i \sum_{j=1}^{k} \alpha_j \, h_j(\boldsymbol{x}_i) + \theta \sum_{j=1}^{k} \alpha_j \big) \Big)$$

$$\overset{(a)}{=} \exp\big(\theta \sum_{j=1}^{k} \alpha_j\big) \, \mathbb{E}\Big(\exp\big(- y_i \sum_{j=1}^{k} \alpha_j \, h_j(\boldsymbol{x}_i) \big) \Big)$$

$$\overset{(b)}{=} \frac{1}{n} \exp\big(\theta \sum_{j=1}^{k} \alpha_j\big) \sum_{i=1}^{n} \exp\big(- y_i \sum_{j=1}^{k} \alpha_j \, h_j(\boldsymbol{x}_i) \big), \tag{19}$$

where (a) is because the expectation is with respect to the data, i.e., $\boldsymbol{x}_i$ and y_i and (b) is according to the definition of expectation.

by Monte Carlo approximation.

Upper Bound on the Generalization Error of Boosting

- Recall the formula of w_i in the Algorithm:

$$w_i^{(j+1)} = w_i^{(j)} \exp\left(\alpha_j \mathbb{I}(y_i \neq h_j(\boldsymbol{x}_i))\right),$$

where j denotes the iteration index. It can be restated as:

$$w_i^{(j+1)} = w_i^{(j)} \exp\left(-y_i \alpha_j h_j(\boldsymbol{x}_i)\right),$$

because $y_i = \pm 1$ and $h_j(\boldsymbol{x}_i) = \pm 1$.

- It is not harmful to AdaBoost if we use the normalized weights:

$$w_i^{(j+1)} = \frac{w_i^{(j)} \exp\left(-y_i \alpha_j h_j(\boldsymbol{x}_i)\right)}{z_j}, \tag{20}$$

where:

$$z_j := \sum_{i=1}^{n} w_i^{(j)} \exp\left(-y_i \alpha_j h_j(\boldsymbol{x}_i)\right). \tag{21}$$

Upper Bound on the Generalization Error of Boosting

- We had:

$$w_i^{(j+1)} = \frac{w_i^{(j)} \exp\left(-y_i \alpha_j h_j(\boldsymbol{x}_i)\right)}{z_j},$$

- Considering that $w_i^{(1)} = 1/n$, we can have recursive expression for the weights:

$$\begin{aligned}
w_i^{(k+1)} &= \frac{w_i^{(k)} \exp\left(-y_i \alpha_k h_k(\boldsymbol{x}_i)\right)}{z_k} \\
&= w_i^{(1)} \times \frac{1}{z_k \times \cdots \times z_1} \times \exp\left(-y_i \alpha_k h_k(\boldsymbol{x}_i)\right) \times \cdots \times \exp\left(-y_i \alpha_1 h_1(\boldsymbol{x}_i)\right) \\
&= \frac{1}{n} \times \frac{1}{\prod_{j=1}^{k} z_j} \times \prod_{j=1}^{k} \exp\left(-y_i \alpha_j h_j(\boldsymbol{x}_i)\right) = \frac{1}{n} \times \frac{1}{\prod_{j=1}^{k} z_j} \times \exp\left(-y_i \sum_{j=1}^{k} \alpha_j h_j(\boldsymbol{x}_i)\right) \\
\implies & \frac{1}{n} \exp\left(-y_i \sum_{j=1}^{k} \alpha_j h_j(\boldsymbol{x}_i)\right) = \left(\prod_{j=1}^{k} z_j\right) \sum_{i=1}^{n} w_i^{(k+1)}. \qquad (22)
\end{aligned}$$

Upper Bound on the Generalization Error of Boosting

- We found (Eq. (22)):

$$\frac{1}{n}\exp\left(-y_i \sum_{j=1}^{k} \alpha_j \, h_j(\boldsymbol{x}_i)\right) = \left(\prod_{j=1}^{k} z_j\right) \sum_{i=1}^{n} w_i^{(k+1)}.$$

- We continue the Eq. (19):

$$\mathbb{P}(y_i \, \widehat{f}(\boldsymbol{x}_i) \leq \theta) \leq \frac{1}{n} \exp\left(\theta \sum_{j=1}^{k} \alpha_j\right) \sum_{i=1}^{n} \exp\left(-y_i \sum_{j=1}^{k} \alpha_j \, h_j(\boldsymbol{x}_i)\right)$$

$$\overset{(22)}{=} \exp\left(\theta \sum_{j=1}^{k} \alpha_j\right) \left(\prod_{j=1}^{k} z_j\right) \sum_{i=1}^{n} w_i^{(k+1)}.$$

Upper Bound on the Generalization Error of Boosting

- We found:

$$\mathbb{P}(y_i \widehat{f}(\boldsymbol{x}_i) \leq \theta) \leq \exp\left(\theta \sum_{j=1}^{k} \alpha_j\right) \left(\prod_{j=1}^{k} z_j\right) \sum_{i=1}^{n} w_i^{(k+1)}.$$

- According to Eqs. (20) and (21),

$$\begin{cases} w_i^{(j+1)} = \dfrac{w_i^{(j)} \exp\left(-y_i \alpha_j h_j(\boldsymbol{x}_i)\right)}{z_j}, \\ z_j := \displaystyle\sum_{i=1}^{n} w_i^{(j)} \exp\left(-y_i \alpha_j h_j(\boldsymbol{x}_i)\right), \end{cases}$$

we have:

$$\sum_{i=1}^{n} w_i^{(j+1)} = \frac{\sum_{i=1}^{n} w_i^{(j)} \exp\left(-y_i \alpha_j h_j(\boldsymbol{x}_i)\right)}{\sum_{i=1}^{n} w_i^{(j)} \exp\left(-y_i \alpha_j h_j(\boldsymbol{x}_i)\right)} = 1.$$

Therefore:

$$\mathbb{P}(y_i \widehat{f}(\boldsymbol{x}_i) \leq \theta) \leq \exp\left(\theta \sum_{j=1}^{k} \alpha_j\right) \left(\prod_{j=1}^{k} z_j\right). \tag{23}$$

Upper Bound on the Generalization Error of Boosting

- On the other hand, according to Eq. (21), $z_j := \sum_{i=1}^{n} w_i^{(j)} \exp\left(-y_i\, \alpha_j\, h_j(\boldsymbol{x}_i)\right)$, we have:

$$
\begin{aligned}
z_j &= \sum_{i=1}^{n} w_i^{(j)} \exp\left(-y_i\, \alpha_j\, h_j(\boldsymbol{x}_i)\right) \\
&= \sum_{i=1}^{n} w_i^{(j)} \exp(-\alpha_j)\, \mathbb{I}(y_i = h_j(\boldsymbol{x}_i)) + \sum_{i=1}^{n} w_i^{(j)} \exp(\alpha_j)\, \mathbb{I}(y_i \neq h_j(\boldsymbol{x}_i)) \\
&= \exp(-\alpha_j) \sum_{i=1}^{n} w_i^{(j)} \mathbb{I}(y_i = h_j(\boldsymbol{x}_i)) + \exp(\alpha_j) \sum_{i=1}^{n} w_i^{(j)} \mathbb{I}(y_i \neq h_j(\boldsymbol{x}_i)). \qquad (24)
\end{aligned}
$$

- Recall Eq. (20) for w_i^{j+1}, i.e., $w_i^{(j+1)} = \frac{w_i^{(j)} \exp\left(-y_i\, \alpha_j\, h_j(\boldsymbol{x}_i)\right)}{z_j}$. This is in the range $[0, 1]$ and its summation over error cases can be considered as the probability of error:

$$
\sum_{i=1}^{n} w_i^{(j)} \mathbb{I}(y_i \neq h_j(\boldsymbol{x}_i)) = \mathbb{P}(y_i \neq h_j(\boldsymbol{x}_i)) \overset{(a)}{=} L_j, \qquad (25)
$$

where (a) is because the Eq. (3), $L_j = \frac{\sum_{i=1}^{n} w_i\, \mathbb{I}(y_i \neq h_j(\boldsymbol{x}_i))}{\sum_{i=1}^{n} w_i}$, is the cost which is the probability of error. Therefore, the Eq. (24) becomes:

$$
z_j = \exp(-\alpha_j)\,(1 - L_j) + \exp(\alpha_j)\, L_j.
$$

Upper Bound on the Generalization Error of Boosting

- Recall the formula of α_j in the Algorithm, i.e., $\alpha_j = \log(\frac{1-L_j}{L_j})$. Scaling it is not harmful to AdaBoost:

$$\alpha_j = \frac{1}{2}\log(\frac{1-L_j}{L_j}). \tag{26}$$

- Therefore, we can have:

$$\begin{aligned}
z_j &= \exp(-\alpha_j)(1-L_j) + \exp(\alpha_j) L_j \\
&= \exp(-\frac{1}{2}\log(\frac{1-L_j}{L_j}))(1-L_j) + \exp(\frac{1}{2}\log(\frac{1-L_j}{L_j})) L_j \\
&= \exp(\log(\sqrt{\frac{L_j}{1-L_j}}))(1-L_j) + \exp(\log(\sqrt{\frac{1-L_j}{L_j}})) L_j \\
&= \sqrt{\frac{L_j}{1-L_j}}(1-L_j) + \sqrt{\frac{1-L_j}{L_j}} L_j = \sqrt{L_j(1-L_j)} + \sqrt{L_j(1-L_j)} \\
\Longrightarrow\ z_j &= 2\sqrt{L_j(1-L_j)}.
\end{aligned} \tag{27}$$

Upper Bound on the Generalization Error of Boosting

- We found Eqs. (26), (27), and (23):

$$\alpha_j = \frac{1}{2}\log\left(\frac{1-L_j}{L_j}\right), \quad z_j = 2\sqrt{L_j(1-L_j)}, \quad \mathbb{P}(y_i\,\widehat{f}(\boldsymbol{x}_i) \leq \theta) \leq \exp\left(\theta \sum_{j=1}^{k} \alpha_j\right)\left(\prod_{j=1}^{k} z_j\right).$$

- Plugging Eqs. (26) and (27) in Eq. (23) gives:

$$\mathbb{P}(y_i\,\widehat{f}(\boldsymbol{x}_i) \leq \theta) \leq \exp\left(\frac{1}{2}\theta \sum_{j=1}^{k} \log\left(\frac{1-L_j}{L_j}\right)\right)\left(2^k \prod_{j=1}^{k} \sqrt{L_j(1-L_j)}\right)$$

$$= 2^k \exp\left(\sum_{j=1}^{k} \log\left(\left(\frac{1-L_j}{L_j}\right)^{\theta/2}\right)\right) \prod_{j=1}^{k} \sqrt{L_j(1-L_j)}$$

$$= 2^k \prod_{j=1}^{k} \exp\left(\log\left(\left(\frac{1-L_j}{L_j}\right)^{\theta/2}\right)\right) \prod_{j=1}^{k} \sqrt{L_j(1-L_j)}$$

$$= 2^k \prod_{j=1}^{k} \left(\frac{1-L_j}{L_j}\right)^{\theta/2} \prod_{j=1}^{k} \sqrt{L_j(1-L_j)} = 2^k \prod_{j=1}^{k} \sqrt{\left(\frac{1-L_j}{L_j}\right)^{\theta} L_j(1-L_j)},$$

which simplifies to the upper bound on the generalization error of AdaBoost [9]:

$$\mathbb{P}\left(y_i\,\widehat{f}(\boldsymbol{x}_i) \leq \theta\right) \leq 2^k \prod_{j=1}^{k} \sqrt{L_j^{1-\theta}(1-L_j)^{1+\theta}}. \tag{28}$$

Upper Bound on the Generalization Error of Boosting

- We found the upper bound on the generalization error of AdaBoost [9]:

$$\mathbb{P}\big(y_i\, \widehat{f}(\boldsymbol{x}_i) \leq \theta\big) \leq 2^k \prod_{j=1}^{k} \sqrt{L_j^{1-\theta}(1-L_j)^{1+\theta}},$$

where $\mathbb{P}\big(y_i\, \widehat{f}(\boldsymbol{x}_i) \leq \theta\big)$ is the probability that the generalization (true) error for the i-th instance is less than $\theta > 0$.

- According to Eq. (3), $L_j = \frac{\sum_{i=1}^{n} w_i\, \mathbb{I}(y_i \neq h_j(\boldsymbol{x}_i))}{\sum_{i=1}^{n} w_i}$, we have $L_j \in [0,1]$. If we have:

$$L_j \leq 0.5 - \xi, \tag{29}$$

where $\xi \in (0, 0.5)$, the Eq. (28) becomes:

$$\mathbb{P}\big(y_i\, \widehat{f}(\boldsymbol{x}_i) \leq \theta\big) \leq 2^k \prod_{j=1}^{k} \sqrt{L_j^{1-\theta}(1-L_j)^{1+\theta}} = \prod_{j=1}^{k} 2\sqrt{L_j^{1-\theta}(1-L_j)^{1+\theta}}$$

$$= \prod_{j=1}^{k} \sqrt{2^2 L_j^{1-\theta}(1-L_j)^{1+\theta}} = \prod_{j=1}^{k} \sqrt{2^{1-\theta} L_j^{1-\theta}\, 2^{1+\theta}(1-L_j)^{1+\theta}}$$

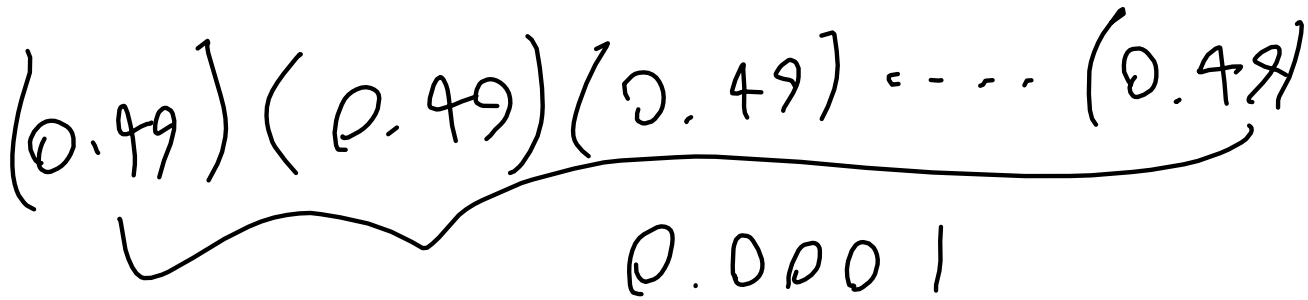
$$= \prod_{j=1}^{k} \sqrt{(2L_j)^{1-\theta}(2(1-L_j))^{1+\theta}} \overset{(29)}{\leq} \left(\sqrt{(1-2\xi)^{1-\theta}(1+2\xi)^{1+\theta}}\right)^k. \tag{30}$$

Upper Bound on the Generalization Error of Boosting

- We found:

$$\mathbb{P}\big(y_i\, \widehat{f}(\boldsymbol{x}_i)\big) \leq \Big(\sqrt{(1-2\xi)^{1-\theta}(1+2\xi)^{1+\theta}}\Big)^k,$$

  which is a very good upper bound because if $\theta < \xi$, we have $\sqrt{(1-2\xi)^{1-\theta}(1+2\xi)^{1+\theta}} < 1$; thus, the probability of error, $\mathbb{P}\big(y_i\, \widehat{f}(\boldsymbol{x}_i) \leq \theta\big)$, decreases **exponentially** with k which is the number of models used in boosting.
- This shows that boosting helps us reduce the generalization error and thus helps us avoid overfitting. In other words, because of the bound on generalization error, boosting overfits very hardly.
- If ξ is a very small positive number, the $L_j \leq 0.5 - \xi$ is a little smaller than 0.5, i.e., $L_j \lessapprox 0.5$. As we are discussing binary classification in boosting, $L_j = 0.5$ means random classification by flipping a coin. Therefore, for having the great bound of Eq. (30), **having weak base models (a little better than random decision) suffices**. This shows the effectiveness of boosting.
- Note that a very small ξ means a very small θ because of $\theta < \xi$; therefore, it means a very small probability of error because of $\mathbb{P}\big(y_i\, \widehat{f}(\boldsymbol{x}_i) \leq \theta\big)$.
- It is noteworthy that both boosting and bagging can be seen as **ensemble learning** [10] (or majority voting) methods which use **model averaging** [11, 12] and are very effective in learning theory.
- Moreover, both boosting and bagging reduce the variance of estimation [13, 9], especially for the models with high variance of estimation such as trees [14].
- In the above, we analyzed boosting for **binary** classification. A similar discussion can be done for **multi-class** classification in boosting and find an upper bound on the generalization error (see the appendix in [9] for more details).



Boosting as Maximum Margin Classifier

Boosting as Maximum Margin Classifier

- In another perspective, the found upper bound for boosting shows that boosting can be seen as a method to increase (maximize) the margins of training error which results in a good generalization error [15]. This phenomenon is the base for the theory of Support Vector Machines (SVM) [16, 17].
- In the following, we analyze the analogy between maximum margin classifier (i.e., SVM) and boosting [9]. In addition to [9], some more discussions exist for upper bound and margin of boosting [18, 19] to which we refer the interested readers.
- Assume we have training instances $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ where $y_i \in \{-1, +1\}$ for binary classification. The two classes may not be linearly separable. In order to handle this case, we map the data to higher dimensional feature space using kernels [20, 21], hoping that they become linearly separable in the feature space. Assume $\boldsymbol{h}(\boldsymbol{x})$ is a vector which non-linearly maps data to the feature space. Considering $\boldsymbol{\alpha}$ as the vector of optimization variables, the optimization problem [22] in SVM is [17, 9]:

$$\left[\underset{\boldsymbol{\alpha}}{\text{maximize}} \quad \underset{\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}}{\text{minimize}} \quad \frac{y_i \left(\boldsymbol{\alpha}^\top \boldsymbol{h}(\boldsymbol{x}_i)\right)}{||\boldsymbol{\alpha}||_2}. \right. \tag{31}$$

Note that $y_i = \pm 1$ and $\boldsymbol{\alpha}^\top \boldsymbol{h}(\boldsymbol{x}_i) \gtrless 0$; therefore, the sign of $y_i \left(\boldsymbol{\alpha}^\top \boldsymbol{h}(\boldsymbol{x}_i)\right)$ determines the class of the i-th instance.

Boosting as Maximum Margin Classifier

- Eq. (31) was:

$$\underset{\boldsymbol{\alpha}}{\text{maximize}} \quad \underset{\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n}{\text{minimize}} \quad \frac{y_i \left(\boldsymbol{\alpha}^\top \boldsymbol{h}(\boldsymbol{x}_i)\right)}{||\boldsymbol{\alpha}||_2}.$$

- On the other hand, the Eq. (16), $\widehat{f}(\boldsymbol{x}_i) = \frac{\sum_{j=1}^k \alpha_j \, h_j(\boldsymbol{x}_i)}{\sum_{j=1}^k \alpha_j}$, can be written in a vector form:

$$\widehat{f}(\boldsymbol{x}_i) = \frac{\sum_{j=1}^k \alpha_j \, h_j(\boldsymbol{x}_i)}{\sum_{j=1}^k \alpha_j} = \frac{\boldsymbol{\alpha}^\top \boldsymbol{h}(\boldsymbol{x}_i)}{||\boldsymbol{\alpha}||_1}, \tag{32}$$

  where $\boldsymbol{h}(\boldsymbol{x}_i) = [h_1(\boldsymbol{x}_i), \dots, h_k(\boldsymbol{x}_i)]^\top$ and $\boldsymbol{\alpha} = [\alpha_1, \dots, \alpha_k]^\top$. Note that here, $h(\boldsymbol{x}_i) = \pm 1$ and α_j is obtained from Eq. (26), $\alpha_j = \frac{1}{2} \log(\frac{1-L_j}{L_j})$, or the formula of α_j in the Algorithm.

- The similarity between the Eq. (32) and the cost function in Eq. (31) shows that **boosting can be seen as maximizing the margin of classification** resulting in a **good generalization error** [9].
- In other words, finding a linear combination in the high dimensional feature space having a large margin between the training instances of the classes is performed in the two methods.
- Note that a slight difference is the **type of norm** which is interpretable because the mapping to feature space in boosting is only to $h(\boldsymbol{x}_i) = \pm 1$ while in SVM, it can be any number where the sign is important. Therefore, ℓ_1 and ℓ_2 norms are suitable in boosting and SVM, respectively [9].

Boosting as Maximum Margin Classifier

- Another connection between SVM (maximum margin classifier) and boosting is that some of the training instances are found to be most important instances, called **support vectors** [17]. In boosting, also, weighting the training instances can be seen as selecting some **informative** models [23] which can be analogous to support vectors.

Acknowledgment

- For more information on boosting in machine learning, see the book: Trevor Hastie, Robert Tibshirani, Jerome H. Friedman, Jerome H. Friedman. "The elements of statistical learning: data mining, inference, and prediction". Vol. 2. New York: springer, 2009 [24].
- Some slides are based on our tutorial paper: "The Theory Behind Overfitting, Cross Validation, Regularization, Bagging, and Boosting: Tutorial" [25]
- Some slides of this slide deck are inspired by teachings of Prof. Ali Ghodsi at University of Waterloo, Department of Statistics and Prof. Hoda Mohammadzade at Sharif University of Technology, Department of Electrical Engineering.

References

[1] M. Kearns, "Thoughts on hypothesis boosting," *Technical Report, Machine Learning class project*, pp. 1–9, 1988.

[2] M. Kearns and L. Valiant, "Cryptographic limitations on learning boolean formulae and finite automata," *Journal of the ACM (JACM)*, vol. 41, no. 1, pp. 67–95, 1994.

[3] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proceedings of the Thirteenth International Conference on Machine Learning*, pp. 148â–156, Morgan Kaufman, San Francisco, 1996.

[4] T. J. Hastie and R. Tibshirani, "Generalized additive models," *Statistical Science*, vol. 1, no. 3, pp. 297–318, 1986.

[5] J. Friedman, T. Hastie, and R. Tibshirani, "Additive logistic regression: a statistical view of boosting," *The annals of statistics*, vol. 28, no. 2, pp. 337–407, 2000.

[6] R. Rojas, "Adaboost and the super bowl of classifiers: a tutorial introduction to adaptive boosting," *Freie University, Berlin, Technical Report*, 2009.

[7] D. S. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Systems*, vol. 2, pp. 321–355, 1988.

[8] F. Schwenker, H. A. Kestler, and G. Palm, "Three learning phases for radial-basis-function networks," *Neural networks*, vol. 14, no. 4-5, pp. 439–458, 2001.

References (cont.)

[9] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, "Boosting the margin: A new explanation for the effectiveness of voting methods," *The annals of statistics*, vol. 26, no. 5, pp. 1651–1686, 1998.

[10] R. Polikar, "Ensemble learning," in *Ensemble machine learning*, pp. 1–34, Springer, 2012.

[11] J. A. Hoeting, D. Madigan, A. E. Raftery, and C. T. Volinsky, "Bayesian model averaging: a tutorial," *Statistical science*, pp. 382–401, 1999.

[12] G. Claeskens and N. L. Hjort, *Model selection and model averaging*. Cambridge Books, Cambridge University Press, 2008.

[13] L. Breiman, "Arcing classifier (with discussion and a rejoinder by the author)," *The annals of statistics*, vol. 26, no. 3, pp. 801–849, 1998.

[14] J. R. Quinlan, "Bagging, boosting, and c4.5," in *AAAI/IAAI Conference*, vol. 1, pp. 725–730, 1996.

[15] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, pp. 144–152, ACM, 1992.

[16] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

References (cont.)

[17] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[18] L. Wang, M. Sugiyama, C. Yang, Z.-H. Zhou, and J. Feng, "On the margin explanation of boosting algorithms," in *COLT*, pp. 479–490, Citeseer, 2008.

[19] W. Gao and Z.-H. Zhou, "On the doubt about margin explanation of boosting," *Artificial Intelligence*, vol. 203, pp. 1–18, 2013.

[20] B. Scholkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*.
MIT press, 2001.

[21] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *The annals of statistics*, pp. 1171–1220, 2008.

[22] S. Boyd and L. Vandenberghe, *Convex optimization*.
Cambridge university press, 2004.

[23] Y. Freund, "Boosting a weak learning algorithm by majority," *Information and computation*, vol. 121, no. 2, pp. 256–285, 1995.

[24] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, vol. 2.
Springer, 2009.

References (cont.)

[25] B. Ghojogh and M. Crowley, "The theory behind overfitting, cross validation, regularization, bagging, and boosting: tutorial," *arXiv preprint arXiv:1905.12787*, 2019.