AdaBoost

AdaBoost, which stands for ``**Ada**ptive **Boost**ing", is an ensemble learning algorithm that uses the boosting paradigm [1].

We will discuss AdaBoost for binary classification. That is, we assume that we are given a training set $S := (x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ where $\forall i, y_i \in \{-1, 1\}$ and a pool of hypothesis functions $\mathcal{H}$ from which we are to pick T hypotheses in order to form an ensemble H. H then makes a decision using the individual hypotheses $h_1, \dots, h_T$ in the ensemble as follows:

$$H(x) = \sum_{i=1}^{T} \alpha_i h_i(x) \quad (1)$$

That is, H uses a linear combination of the decisions of each of the h_i hypotheses in the ensemble. The AdaBoost algorithm sequentially chooses h_i from $\mathcal{H}$ and assigns this hypothesis a weight α_i. We let H_t be the classifier formed by the first t hypotheses. That is,

$$H_t(x) = \sum_{i=1}^{t} \alpha_i h_i(x)$$

$$= H_{t-1}(x) + \alpha_t h_t(x)$$

where $H_0(x) := 0$. That is, the empty ensemble will always output 0.

The idea behind the AdaBoost algorithm is that the t^{th} hypothesis will correct for the errors that the first $t-1$ hypotheses make on the training set. More specifically, after we select the first $t-1$ hypotheses, we determine which instances in S our $t-1$ hypotheses perform poorly on and make sure that the t^{th} hypothesis performs well on these instances. The pseudocode for AdaBoost is described in Algorithm 1. A high-level overview of the algorithm is described below:

1. Initialize a training set distribution

At each iteration $1, \dots, T$ of the AdaBoost algorithm , we define a probability distribution $\mathcal{D}$ over the training instances in S. We let $\mathcal{D}_t$ be the probability distribution at the t^{th} iteration and $\mathcal{D}_t(i)$ be the probability assigned to the i^{th} training instance, $(x_i, y_i) \in S$, according to $\mathcal{D}_t$. As the algorithm proceeds, each iteration will design $\mathcal{D}_t$ so that it assigns higher probability mass to instances that the first $t-1$ hypotheses performed poorly on. That is, the worse the performance on x_i, the higher will be $\mathcal{D}_t(i)$.

At the onset of the algorithm, we set $\mathcal{D}_1$ to be the uniform distribution over the instances. That is,

$$\forall i \in \{1, 2, \dots, n\}, \mathcal{D}_1(i) := \frac{1}{n}$$

Algorithm 1 AdaBoost for binary classification

Precondition: A training set $S := (x_1, y_1), \ldots, (x_n, y_n)$, hypothesis space $\mathcal{H}$, and number of iterations T.

for $i \in \{1, 2 \ldots, n\}$ **do**
  $\mathcal{D}_1(i) \leftarrow \frac{1}{n}$
end for
$H \leftarrow \emptyset$
for $t = 1, \ldots, T$ **do**
  $h_t \leftarrow \text{argmin}_{h \in \mathcal{H}} P_{i \sim \mathcal{D}_t}(h(x_i) \neq y_i)$ ▹ find good hypothesis on weighted training set
  $\epsilon_t \leftarrow P_{i \sim \mathcal{D}_t}(h_t(x_i) \neq y_i)$ ▹ compute hypothesis's error
  $\alpha_t \leftarrow \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$ ▹ compute hypothesis's weight
  $H \leftarrow H \cup \{(\alpha_t, h_t)\}$ ▹ add hypothesis to the ensemble
  for $i \in \{1, 2 \ldots, n\}$ **do** ▹ update training set distribution
    $\mathcal{D}_{t+1}(i) \leftarrow \frac{\mathcal{D}_t(i)\ e^{-\alpha_t y_i h_t(x_i)}}{\sum_{j=1}^n \mathcal{D}_t(j)\ e^{-\alpha_t y_j h_t(x_j)}}$
  end for
end for
return H

where n is the size of S.

2. Find a new hypothesis to add to the ensemble

At the t^{th} iteration, we search for a new hypothesis, h_t, that performs well on S assuming that instances are drawn from $\mathcal{D}_t$). By ``performs well", we mean that h_t should have a low expected 0-1 loss on S under $\mathcal{D}_t$. That is

$$\begin{aligned} h_t &:= \underset{h \in \mathcal{H}}{\text{argmin}}\ E_{i \sim \mathcal{D}_t}[\ell_{0-1}(h, x_i, y_i)] \\ &= \underset{h \in \mathcal{H}}{\text{argmin}}\ P_{i \sim \mathcal{D}_t}(y_i \neq h(x_i)) \end{aligned}$$

We call this expected loss the ``weighted loss" because the 0-1 loss is not computed on the instances in the training set directly, but rather on the *weighted* instances in the training set.

3. Assign the new hypothesis a weight

Once we compute h_t, we assign h_t a weight α_t based on its performance. More specifically, we give it the weight

$$\alpha_t := \frac{1}{2} \ln\left(\frac{1 - \epsilon_t}{\epsilon_t}\right) \tag{2}$$

where

$$\epsilon_t := P_{i \sim \mathcal{D}_t}(y_i \neq h_t(x_i))$$

. We will soon explain the theoretical justification of this precise weight assignment, but intuitively we see that the the higher ϵ_t, the the larger will be the denominator and the smaller the numerator in $\frac{1-\epsilon_t}{\epsilon_t}$ thus, the smaller will be $\frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$. Thus, if the new hypothesis, h_t, has a high error, ϵ_t, then we assign this hypothesis a smaller weight. That is, h_t will contribute less to the output of ensemble H.

4. Recompute the training set distribution

Once the new hypothesis is added to the ensemble, we recompute the training set distribution to assign each instance a probability proportional to how well the current ensemble H_t performs on the training set. We compute $\mathcal{D}_{t+1}$ as follows:

$$\mathcal{D}_{t+1}(i) := \frac{\mathcal{D}_t(i)\ e^{-\alpha_t y_i h_t(x_i)}}{\sum_{j=1}^{n} \mathcal{D}_t(j)\ e^{-\alpha_t y_j h_t(x_j)}} \tag{3}$$

We will soon explain a theoretical justification for this precise probability assignment, but for now we can gain an intuitive understanding. Note the term $e^{-\alpha_t y_i h_t(x_i)}$. If $h_t(x_i) = y_i$, then $y_i h_t(x_i) = 1$ which means that $e^{-\alpha_t y_i h_t(x_i)} = e^{-\alpha_t}$. If, on the other hand, $h_t(x_i) \neq y_i$, then $y_i h_t(x_i) = -1$ which means that $e^{-\alpha_t y_i h_t(x_i)} = e^{\alpha_t}$. Thus, we see that $e^{-\alpha_t y_i h_t(x_i)}$ is smaller if the hypothesis's prediction agrees with the true value. That is, we assign higher probability to the i^{th} instance if h_t was wrong on x_i.

Repeat steps 2 through 4

Repeat steps 2 through 4 for $T - 1$ more iterations.

Derivation of AdaBoost from first principles

The AdaBoost algorithm can be viewed as an algorithm that searches for hypotheses of the form of Equation 1 in order to minimize the empirical loss under the **exponential loss function**:

$$\ell_{\text{exp}}(h, x, y) := e^{-yh(x)}$$

We note that there are many ways in which one might search for a hypothesis of the form of Equation 1 in order to minimize the exponential loss function. The AdaBoost algorithm performs this minimization using a sequential procedure such that, at iteration t, we are given H_{t-1} and our goal is to produce

$$H_t = H_{t-1} + \alpha_t h_t$$

where the new h_t and α_t minimizes the exponential loss of H_t on the training data. Theorem 1 shows that AdaBoost's choice of h_t minimizes the exponential loss of H_t over the training data. That is,

$$h_t = \operatorname*{argmin}_{h \in \mathcal{H}} L_S(H_{t-1} + Ch)$$

where

$$L_S(H_{t-1} + Ch) := \frac{1}{n} \sum_{i=1}^{n} \ell_{\text{exp}}(H_{t-1} + Ch, x, y)$$

and C is an arbitrary constant. Theorem 2 shows that once h_t is chosen, AdaBoost's choice of α_t then further minimizes the exponential loss of H_t over the training set. That is,

$$\alpha_t := \operatorname*{argmin}_{\alpha} L_S(H_{t-1} + \alpha h_t)$$

.

Theorem 1 *The choice of h_t under AdaBoost,*

$$h_t := \operatorname*{argmin}_{h \in \mathcal{H}} P_{i \sim \mathcal{D}_t}(y_i \neq h(x_i))$$

, minimizes the exponential-loss of H_t over the training set. That is, given an arbitrary constant C,

$$h_t = \operatorname*{argmin}_{h \in \mathcal{H}} L_S(H_{t-1} + Ch)$$

.

Proof:

$$
\begin{aligned}
h_t =& \operatorname*{argmin}_{h \in \mathcal{H}} L_S(H_{t-1} + Ch) \\
&= \operatorname*{argmin}_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} e^{-y_i [H_{t-1}(x_i) + Ch(x_i)]} \\
&= \operatorname*{argmin}_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} e^{-y_i H_{t-1}(x_i)} e^{-yCh(x_i)} \\
&= \operatorname*{argmin}_{h \in \mathcal{H}} \frac{1}{n} \sum_{i=1}^{n} w_{t,i} e^{-yCh(x_i)} && \text{let } w_{t,i} := e^{-y_i H_{t-1}(x_i)} \\
&= \operatorname*{argmin}_{h \in \mathcal{H}} \sum_{i=1}^{n} w_{t,i} e^{-yCh_t(x_i)} \\
&= \operatorname*{argmin}_{h \in \mathcal{H}} \left\{ \sum_{i:h(x_i)=y_i} w_{t,i} e^{-C} + \sum_{i:h(x_i)\neq y_i} w_{t,i} e^{C} \right\} && \text{split the summation} \\
&= \operatorname*{argmin}_{h \in \mathcal{H}} \left\{ \left(\sum_{i=1}^{n} w_{t,i} e^{-C} - \sum_{i:h(x_i)\neq y_i} w_{t,i} e^{-C} \right) + \sum_{i:h(x_i)\neq y_i} w_{t,i} e^{\alpha_t} \right\} \\
&= \operatorname*{argmin}_{h \in \mathcal{H}} \left\{ \sum_{i=1}^{n} w_{t,i} e^{-C} + \sum_{i:h(x_i)\neq y_i} w_{t,i} (e^{C} - e^{-C}) \right\} \\
&= \operatorname*{argmin}_{h \in \mathcal{H}} \left\{ K + \sum_{i:h(x_i)\neq y_i} w_{t,i} (e^{C} - e^{-C}) \right\} && K := \sum_{i=1}^{n} w_i e^{-\alpha_t} \text{ is a constant} \\
&= \operatorname*{argmin}_{h \in \mathcal{H}} \left\{ (e^{C} - e^{-C}) \sum_{i:h(x_i)\neq y_i} w_{t,i} \right\} \\
&= \operatorname*{argmin}_{h \in \mathcal{H}} \sum_{i:h(x_i)\neq y_i} w_{t,i} \\
&= \operatorname*{argmin}_{h \in \mathcal{H}} \frac{1}{\sum_{j=1}^{n} w_{t,j}} \sum_{i:h(x_i)\neq y_i} w_{t,i} && \frac{1}{\sum_{j=1}^{n} w_{t,j}} \text{ is a constant} \\
&= \operatorname*{argmin}_{h \in \mathcal{H}} \sum_{i:h(x_i)\neq y_i} \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}} \\
&= \operatorname*{argmin}_{h \in \mathcal{H}} P_{i \sim \mathcal{D}_t}(y_i \neq h(x_i)) && \text{See Lemma 1}
\end{aligned}
$$

□

Lemma 1

$$P_{i\sim\mathcal{D}_t}(y_i \neq h(x_i)) = \sum_{i:h(x_i)\neq y_i} \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}}$$

where

$$w_{t,i} := e^{-y_i H_{t-1}(x_i)}$$

Proof:

First, we show that

$$\mathcal{D}_t(i) = \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}} \tag{4}$$

We show this fact by induction. First, we prove the base case:

$$\begin{aligned}\frac{w_{1,i}}{\sum_{j=1}^n w_{1,j}} &= \frac{e^{-y_i H_0(x_i)}}{\sum_{j=1}^n e^{-y_j H_0(x_j)}} \\ &= \frac{1}{n} && \text{because } H_0(x_i) = 0 \\ &= \mathcal{D}_1(i) \text{ for all } i\end{aligned}$$

Next, we need to prove the inductive step. That is, we prove that

$$\mathcal{D}_t(i) = \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}} \implies \mathcal{D}_{t+1}(i) = \frac{w_{t+1,i}}{\sum_{j=1}^n w_{t+1,j}}$$

This is proven as follows:

$$\begin{aligned}
\mathcal{D}_{t+1}(i) &:= \frac{\mathcal{D}_t(i)\ e^{-\alpha_t y_i h_t(x_i)}}{\sum_{j=1}^n \mathcal{D}_t(j)\ e^{-\alpha_t y_j h_t(x_j)}} && \text{by Equation 3} \\
&= \frac{\frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}}\ e^{-\alpha_t y_i h_t(x_i)}}{\sum_{j=1}^n \frac{w_{t,j}}{\sum_{k=1}^n w_{t,k}}\ e^{-\alpha_t y_j h_t(x_j)}} && \text{by the inductive hypothesis} \\
&= \frac{\frac{e^{-y_i H_{t-1}(x_i)}}{\sum_{j=1}^n e^{-y_j H_{t-1}(x_j)}}\ e^{-\alpha_t y_i h_t(x_i)}}{\sum_{j=1}^n \frac{e^{-y_j H_{t-1}(x_j)}}{\sum_{k=1}^n e^{-y_k H_{t-1}(x_k)}}\ e^{-\alpha_t y_j h_t(x_j)}} && \text{by the fact that } w_{t,i} := e^{-y_i H_{t-1}(x_i)} \\
&= \frac{\frac{1}{\sum_{j=1}^n e^{-y_j H_{t-1}(x_j)}}\ e^{-y_i H_{t-1}(x_i)}\ e^{-\alpha_t y_i h_t(x_i)}}{\frac{1}{\sum_{k=1}^n e^{-y_k H_{t-1}(x_k)}} \sum_{j=1}^n\ e^{-y_j H_{t-1}(x_j)}\ e^{-\alpha_t y_j h_t(x_j)}} \\
&= \frac{e^{-y_i H_{t-1}(x_i) - \alpha_t y_i h_t(x_i)}}{\sum_{j=1}^n\ e^{-y_j H_{t-1}(x_j) - \alpha_t y_j h_t(x_j)}} \\
&= \frac{e^{-y_i H_t(x_i)}}{\sum_{j=1}^n\ e^{-y_j H_t(x_j)}} \\
&= \frac{w_{t+1,i}}{\sum_{j=1}^n w_{t+1,j}}
\end{aligned}$$

Now that we have proven Equation 4, it follows that

$$\begin{aligned}
\sum_{i:h(x_i)\neq y_i} \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}} &= \sum_{i:h(x_i)\neq y_i} \mathcal{D}_t(x_i) \\
&= P_{i\sim\mathcal{D}_t}(y_i \neq h_t(x_i))
\end{aligned}$$

□

Theorem 2 *The choice of α_t under AdaBoost,*

$$\alpha_t := \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

where

$$\epsilon_t := P_{i\sim\mathcal{D}_t}(y_i \neq h_t(x_i))$$

, minimizes the exponential-loss of H_t over the training set. That is,

$$\alpha_t = \underset{\alpha}{argmin}\ L_S(H_{t-1} + \alpha h_t)$$

.

Proof:

Our goal is to solve

$$\begin{aligned}\alpha_t &:= \underset{\alpha}{\text{argmin}}\ L_S(H_{t-1} + \alpha h_t) \\ &= \underset{\alpha}{\text{argmin}}\ \left\{\left(\sum_{i:h(x_i)\neq y_i} w_{t,i}\right)e^{\alpha} + \left(\sum_{i:h(x_i)=y_i} w_{t,i}\right)e^{-\alpha}\right\}\end{aligned}$$

To do so, set the derivative of the function in the argmin to zero and solve for α (the function is convex, though we don't prove it here):

$$\frac{d}{d\alpha}\left\{\left(\sum_{i:h(x_i)\neq y_i} w_{t,i}\right)e^{\alpha} + \left(\sum_{i:h(x_i)=y_i} w_{t,i}\right)e^{-\alpha}\right\} = 0$$

$$\implies \left(\sum_{i:h(x_i)\neq y_i} w_{t,i}\right)e^{\alpha} - \left(\sum_{i:h(x_i)=y_i} w_{t,i}\right)e^{-\alpha} = 0$$

$$\begin{aligned}&\implies e^{2\alpha} = -\frac{\sum_{i:h(x_i)\neq y_i} w_{t,i}}{\sum_{i:h(x_i)=y_i} w_{t,i}} \\ &\implies 2\alpha = \ln\left(-\frac{\sum_{i:h(x_i)\neq y_i} w_{t,i}}{\sum_{i:h(x_i)=y_i} w_{t,i}}\right) \\ &\implies \alpha = \frac{1}{2}\ln\left(\frac{\sum_{i:h(x_i)=y_i} w_{t,i}}{\sum_{i:h(x_i)\neq y_i} w_{t,i}}\right) \\ &\implies \alpha = \frac{1}{2}\ln\left(\frac{\sum_{i=1}^n w_{t,i} - \sum_{i:h(x_i)\neq y_i} w_{t,i}}{\sum_{i:h(x_i)\neq y_i} w_{t,i}}\right) \\ &\implies \alpha = \frac{1}{2}\ln\left(\frac{\frac{1}{\sum_{i=1}^n w_{t,i}}}{\frac{1}{\sum_{i=1}^n w_{t,i}}}\frac{\sum_{i=1}^n w_{t,i} - \sum_{i:h(x_i)\neq y_i} w_{t,i}}{\sum_{i:h(x_i)\neq y_i} w_{t,i}}\right) \\ &\implies \alpha = \frac{1}{2}\ln\left(\frac{1 - \frac{\sum_{i:h(x_i)\neq y_i} w_{t,i}}{\sum_{i=1}^n w_{t,i}}}{\frac{\sum_{i:h(x_i)\neq y_i} w_{t,i}}{\sum_{i=1}^n w_{t,i}}}\right) \\ &\implies \alpha = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)\end{aligned}$$

☐

Bibliography

[1] Y. Freund and R.E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 1996.