### AdaBoost

**AdaBoost**, which stands for "Adaptive Boosting", 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 \( S := (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n) \) 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 \( h_1, \ldots, h_T \) in the ensemble as follows:

\[
H(x) = \sum_{i=1}^{T} \alpha_i h_i(x)
\]

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 \( \alpha_i \). We let \( H_t \) be the classifier formed by the first \( t \) hypotheses. That is,

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

\[
= H_{t-1}(x) + \alpha_i h_i(x)
\]

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 \( 1, \ldots, T \) of the AdaBoost algorithm, we define a probability distribution \( D \) over the training instances in \( S \). We let \( D_t \) be the probability distribution at the \( t \)th iteration and \( D_t(i) \) be the probability assigned to the \( i \)th training instance, \( (x_i, y_i) \in S \), according to \( D_t \). As the algorithm proceeds, each iteration will design \( D_t \) 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 \( D_t(i) \).

At the onset of the algorithm, we set \( D_1 \) to be the uniform distribution over the instances. That is,

\[
\forall i \in \{1, 2, \ldots, n\}, D_1(i) := \frac{1}{n}
\]
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 $T$.

1. for $i \in \{1, 2, \ldots, n\}$ do
2. $D_1(i) \leftarrow \frac{1}{n}$
3. end for
4. $H \leftarrow \emptyset$
5. for $t = 1, \ldots, T$ do
6. $h_t \leftarrow \text{argmin}_{h \in \mathcal{H}} P_{t-\mathcal{D}_t}(h(x) \neq y) \quad \triangleright$ find good hypothesis on weighted training set
7. $\epsilon_t \leftarrow P_{t-\mathcal{D}_t}(h_t(x) \neq y_t) \quad \triangleright$ compute hypothesis's error
8. $\alpha_t \leftarrow \frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right) \quad \triangleright$ compute hypothesis's weight
9. $H \leftarrow H \cup \{(\alpha_t, h_t)\} \quad \triangleright$ add hypothesis to the ensemble
10. for $i \in \{1, 2, \ldots, n\}$ do
11. $D_{t+1}(i) \leftarrow \frac{D_t(i) e^{-\alpha_t h_t(x)}}{\sum_{j=1}^n D_t(j) e^{-\alpha_t h_t(x_j)}} \quad \triangleright$ update training set distribution
12. end for
13. end for
14. 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

$$h_t := \text{argmin}_{h \in \mathcal{H}} E_{t-\mathcal{D}_t}[\ell_{0-1}(h, x, y)]$$
$$= \text{argmin}_{h \in \mathcal{H}} P_{t-\mathcal{D}_t}(y \neq h(x))$$

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 $\alpha_t$ based on its performance. More specifically, we give it the weight

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

where

$$ \epsilon_t := P_{i-D_t}(y_i \neq h_t(x_i)) $$

We will soon explain the theoretical justification of this precise weight assignment, but intuitively we see that the higher $\epsilon_t$, the larger will be the denominator and the smaller the numerator in $\frac{1}{2} \ln \left( \frac{1 - \epsilon_t}{\epsilon_t} \right)$. Thus, if the new hypothesis, $h_t$, has a high error, $\epsilon_t$, 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 $D_{t+1}$ as follows:

$$ D_{t+1}(i) := \frac{D_t(i) e^{-\alpha_y h_t(x_i)}}{\sum_{j=1}^{n} D_t(j) e^{-\alpha_y h_t(x_j)}} $$

We will soon explain a theoretical justification for this precise probability assignment, but for now we can gain an intuitive understanding. Note the term $e^{-\alpha_y h_t(x_i)}$. If $h_t(x_i) = y_i$, then $y_i h_t(x_i) = 1$ which means that $e^{-\alpha_y h_t(x_i)} = e^{-\alpha_t}$. If, on the other hand, $h_t(x_i) \neq y_i$, then $y_i h_t(x_i) = -1$ which means that $e^{-\alpha_y h_t(x_i)} = e^{\alpha_t}$. Thus, we see that $e^{-\alpha_y h_t(x_i)}$ 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**:

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

© Matthew Bernstein 2017
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 $\alpha_t$ 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,

$$h_t = \arg\min_{h \in \mathcal{H}} L_S(H_{t-1} + Ch)$$

where

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

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

$$\alpha_t := \arg\min_{\alpha} L_S(H_{t-1} + \alpha h_t)$$

\[\hfill\]

\textbf{Theorem 1} \textit{The choice of $h_t$ under AdaBoost,}

$$h_t := \arg\min_{h \in \mathcal{H}} P_{l-D}(y_i \neq h(x_i))$$

\textit{minimizes the exponential-loss of $H_t$ over the training set. That is, given an arbitrary constant $C$,}

$$h_t = \arg\min_{h \in \mathcal{H}} L_S(H_{t-1} + Ch)$$

\[\hfill\]
Proof:

\[ h_t = \arg \min_{h \in H} L_S (H_{t-1} + Ch) \]
\[ = \arg \min_{h \in H} \frac{1}{n} \sum_{i=1}^{n} e^{-\gamma[H_{t-1}(x_i) + Ch(x_i)]} \]
\[ = \arg \min_{h \in H} \frac{1}{n} \sum_{i=1}^{n} e^{-\gamma H_{t-1}(x_i)} e^{-\gamma Ch(x_i)} \]
\[ = \arg \min_{h \in H} \frac{1}{n} \sum_{i=1}^{n} W_{t,i} e^{-\gamma Ch(x_i)} \]
\[ \text{let } W_{t,i} := e^{-\gamma H_{t-1}(x_i)} \]
\[ = \arg \min_{h \in H} \sum_{i=1}^{n} W_{t,i} e^{-\gamma Ch(x_i)} \]
\[ = \arg \min_{h \in H} \left\{ \sum_{i : h(x_i) = y_i} W_{t,i} e^{-C} + \sum_{i : h(x_i) \neq y_i} W_{t,i} e^{C} \right\} \]
\[ \text{split the summation} \]
\[ = \arg \min_{h \in H} \left\{ \sum_{i=1}^{n} W_{t,i} e^{-C} - \sum_{i : h(x_i) \neq y_i} W_{t,i} e^{-C} + \sum_{i : h(x_i) \neq y_i} W_{t,i} e^{C} \right\} \]
\[ = \arg \min_{h \in H} \left\{ \sum_{i=1}^{n} W_{t,i} e^{-C} + \sum_{i : h(x_i) \neq y_i} W_{t,i} (e^{C} - e^{-C}) \right\} \]
\[ = \arg \min_{h \in H} \left\{ K + \sum_{i : h(x_i) \neq y_i} W_{t,i} (e^{C} - e^{-C}) \right\} \]
\[ K := \sum_{i=1}^{n} W_i e^{-\alpha_i} \text{ is a constant} \]
\[ = \arg \min_{h \in H} \left\{ (e^{C} - e^{-C}) \sum_{i : h(x_i) \neq y_i} W_{t,i} \right\} \]
\[ = \arg \min_{h \in H} \sum_{i : h(x_i) \neq y_i} W_{t,i} \]
\[ = \arg \min_{h \in H} \frac{1}{\sum_{j=1}^{n} W_{j,i}} \sum_{i : h(x_i) \neq y_i} W_{t,i} \]
\[ \frac{1}{\sum_{j=1}^{n} W_{j,i}} \text{ is a constant} \]
\[ = \arg \min_{h \in H} \sum_{i : h(x_i) \neq y_i} \sum_{j=1}^{n} W_{t,j} \]
\[ = \arg \min_{h \in H} P_{i \sim \mathcal{D}_h}(y_i \neq h(x_i)) \]
\[ \text{See Lemma 1} \]
Lemma 1

\[ P_{t|D_i}(y_i \neq h(x_i)) = \sum_{i, h(x_i) \neq y_i} \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}} \]

where

\[ w_{t,j} := e^{-y_i H_0(x_i)} \]

Proof:

First, we show that

\[ D_t(i) = \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}} \tag{4} \]

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

\[
\frac{w_{1,i}}{\sum_{j=1}^{n} w_{1,j}} = \frac{e^{-y_i H_0(x_i)}}{\sum_{j=1}^{n} e^{-y_j H_0(x_j)}} = \frac{1}{n} \quad \text{because } H_0(x_i) = 0
\]

\[ = D_1(i) \text{ for all } i \]

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

\[ D_t(i) = \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}} \Rightarrow D_{t+1}(i) = \frac{w_{t+1,i}}{\sum_{j=1}^{n} w_{t+1,j}} \]
This is proven as follows:

\[ D_{t+1}(i) := \frac{D_t(i) e^{-\alpha_i y_i h_t(x_i)}}{\sum_{j=1}^{n} D_t(j) e^{-\alpha_i y_i h_t(x_j)}} \]

by Equation 3

\[
= \frac{\sum_{j=1}^{n} w_{t,j} e^{-\alpha_i y_i h_t(x_j)}}{\sum_{j=1}^{n} w_{t,j}} e^{-\alpha_i y_i h_t(x_i)}
\]

by the inductive hypothesis

\[
= \frac{\sum_{j=1}^{n} w_{t,j} e^{-y_j H_{t-1}(x_j)}}{\sum_{j=1}^{n} w_{t,j}} e^{-\alpha_i y_i h_t(x_i)}
\]

by the fact that \( w_{t,i} := e^{-y_i H_{t-1}(x_i)} \)

\[
= \frac{\sum_{j=1}^{n} e^{-y_j H_{t-1}(x_j)} \sum_{j=1}^{n} e^{-\alpha_i y_i h_t(x_j)}}{\sum_{j=1}^{n} e^{-y_j H_{t-1}(x_j)} - \alpha_i y_i h_t(x_j)}
\]

Now that we have proven Equation 4, it follows that

\[
\sum_{i \in h(x_i) \neq y_i} \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}} = \sum_{i \in h(x_i) \neq y_i} D_t(x_i)
\]

\[
= P_{i \sim D_t}(y_i \neq h_t(x_i))
\]

□

**Theorem 2** The choice of \( \alpha_t \) under AdaBoost,

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

where

\[
\epsilon_t := P_{i \sim D_t}(y_i \neq h_t(x_i))
\]
minimizes the exponential-loss of \( H_t \) over the training set. That is,
\[
\alpha_t = \arg\min_{\alpha} L_S (H_{t-1} + \alpha h_t)
\]

Proof:

Our goal is to solve
\[
\alpha_t := \arg\min_{\alpha} L_S (H_{t-1} + \alpha h_t)
\]

\[
= \arg\min_{\alpha} \left\{ \left( \sum_{i: h(x_i) \neq y_i} w_{t,i} \right) e^\alpha + \left( \sum_{i: h(x_i) = y_i} w_{t,i} \right) e^{-\alpha} \right\}
\]

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

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

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

\[
\Rightarrow e^{2\alpha} = \frac{\sum_{i: h(x_i) \neq y_i} w_{t,i}}{\sum_{i: h(x_i) = y_i} w_{t,i}}
\]

\[
\Rightarrow 2\alpha = \ln \left( \frac{\sum_{i: h(x_i) \neq y_i} w_{t,i}}{\sum_{i: h(x_i) = y_i} w_{t,i}} \right)
\]

\[
\Rightarrow \alpha = \frac{1}{2} \ln \left( \frac{\sum_{i: h(x_i) = y_i} w_{t,i}}{\sum_{i: h(x_i) \neq y_i} w_{t,i}} \right)
\]

\[
\Rightarrow \alpha = \frac{1}{2} \ln \left( \frac{\sum_{i=1}^n w_{t,i} - \sum_{i: h(x_i) \neq y_i} w_{t,i}}{\sum_{i: h(x_i) \neq y_i} w_{t,i}} \right)
\]

\[
\Rightarrow \alpha = \frac{1}{2} \ln \left( \frac{\frac{1}{\sum_{i=1}^n w_{t,i}} \sum_{i=1}^n w_{t,i} - \sum_{i: h(x_i) \neq y_i} w_{t,i}}{\sum_{i: h(x_i) \neq y_i} w_{t,i}} \right)
\]

\[
\Rightarrow \alpha = \frac{1}{2} \ln \left( \frac{1 - \sum_{i: h(x_i) \neq y_i} w_{t,i}}{\sum_{i=1}^n w_{t,i} \sum_{i: h(x_i) \neq y_i} w_{t,i}} \right)
\]

\[
\Rightarrow \alpha = \frac{1}{2} \ln \left( \frac{1}{e_t} \right)
\]
Bibliography