Derivation of AdaBoost as a Special Case of Forward Stagewise Additive Modelling

Matthias Feurer   Katharina Eggensperger   Frank Hutter

July 3, 2018

This explanation is based on Section 10.4 ("Exponential Loss and AdaBoost") from:

[1] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning*, Springer, 2nd edition, 2005, `http://statweb.stanford.edu/~tibs/ElemStatLearn/`.

We highly recommend Chapter 10 of Hastie et al. We only wrote this note to fill in the gory details for the derivation in Section 10.4.

AdaBoost combines several weak learners through a weighted majority vote to produce the final prediction (equation 10.1):

$$G(x) = sign(\sum_{m=1}^{M} \alpha_m G_m(x)) \tag{1}$$

Boosting is a special way to combine weak learners (basis functions) in an additive way. Section 10.2 provides other examples of this technique, for example neural networks, wavelets and decision trees.

Such additive models could optimally be fit by minimizing an averaged loss function over the training data (10.4):

$$\min_{\{\beta_m,\gamma_m\}_1^M} \sum_{i=1}^{N} \mathbb{L}(y, \sum_{m=1}^{M} \beta_m b(x_i;\gamma_m)) \tag{2}$$

However, since solving this combined minimization problem is often prohibitive, we resort to a technique called *Forward Stagewise Additive Modeling.* At each iteration m, this solves for the optimal next weak learner $b(x, \gamma_m)$ and the corresponding coefficient β_m (without changing the previous weak learners and coefficients) and adds the current expansion $f_{m-1}(x)$ to produce $f_m(x)$.

We will show that AdaBoost is a special case of forward stagewise additive modeling using the following exponential loss function (10.8):

$$\mathbb{L}(y, f(x)) = \exp(-y \times f(x)) \tag{3}$$

In each iteration, forward stagewise additive modeling must solve the following minimization problem:

$$\begin{align}
(\beta_m, G_m) &= arg \min_{\beta,G} \sum_{i=1}^{N} \exp(-y_i(f_{m-1}(x_i) + \beta G(x_i)) \tag{4}\\
&= arg \min_{\beta,G} \sum_{i=1}^{N} \exp(-y_i f_{m-1}(x_i) + (-y_i)\beta G(x_i)) \tag{5}\\
&= arg \min_{\beta,G} \sum_{i=1}^{N} \exp(-y_i f_{m-1}(x_i)) \cdot \exp(-y_i \beta G(x_i)) \tag{6}
\end{align}$$

Because y_i and f_{m-1} neither depend on β nor $G(x)$, we can write the term $\exp(-y_i f_{m-1}(x_i))$ as a weight, which is applied to each individual observation:

$$w_i^{(m)} = \exp(-y_i f_{m-1}(x_i)).$$

This leads us to the following equation (10.9):

$$(\beta_m, G_m) = arg \min_{\beta_m,G} \sum_{i=1}^{N} w_i^{(m)} \exp(-\beta y_i G(x_i)) \tag{7}$$

Weights depend on $f_{(m-1)}$, and so the individual weight values change with each iteration m. We can solve this equations in two steps:

1. find a model for $G_m(x)$.
2. find a value for β_m.

We first show that the optimal $G_m(x)$ does not depend on β. With $\beta > 0$, we split the right-hand side of the above equation into two terms and then simplify:

$$\begin{align}
G_m &= arg \min_{G} \sum_{y_i=G(x_i)} w_i^{(m)} \exp(-y_i \beta G(x_i)) + \sum_{y \neq G(x_i)} w_i^{(m)} \exp(-y_i \beta G(x_i)) \tag{8}\\
&= arg \min_{G} \sum_{y_i=G(x_i)} w_i^{(m)} \exp(-\beta) + \sum_{y \neq G(x_i)} w_i^{(m)} \exp(\beta) \tag{9}\\
&= arg \min_{G} \exp(-\beta) \sum_{y_i=G(x_i)} w_i^{(m)} + \exp(\beta) \sum_{y \neq G(x_i)} w_i^{(m)} \tag{10}
\end{align}$$

In the first sum y_i and $G(x_i)$ are equal, therefore $\exp(-y_i\beta G(x_i)) = \exp(-\beta)$. In the second sum they differ, changing the sign in the exponent: $\exp(-y_i\beta G(x_i)) = \exp(\beta)$. Hastie et al. rewrite this in equation 10.11 in a way which shows that only the weighted errors influence the choice:

$$G_m = arg\min_G \left((\exp(\beta) - \exp(-\beta)) \cdot \sum_{i=1}^{N} w_i^{(m)} \mathbb{I}(y_i \neq G(x_i)) + \exp(-\beta) \cdot \sum_{i=1}^{N} w_i^{(m)} \right) \tag{11}$$

Observe that the first sum resembles the second sum of the upper equation, but is multiplied with a lower factor. The second sum serves as a correction factor in the case of a wrong classification, or is equal to the first sum in the positive case.

Now that we have a strategy for finding G_m, we turn to finding β_m. Setting the partial derivative of 10.9 with respect to β to zero and solving for β, we obtain:

$$\frac{\partial(\sum_{i=1}^{N} w_i^{(m)} \exp(-\beta y_i G(x_i)))}{\partial\beta} = 0 \tag{12}$$

$$\Rightarrow \frac{\partial(\exp(-\beta)\sum_{y=G(x)} w_i^{(m)} + \exp(\beta)\sum_{y\neq G(x)} w_i^{(m)})}{\partial\beta} = 0 \tag{13}$$

$$\Rightarrow -\exp(-\beta)\sum_{y=G(x)} w_i^{(m)} + \exp(\beta)\sum_{y\neq G(x)} w_i^{(m)} = 0 \tag{14}$$

$$\Rightarrow \exp(\beta)\sum_{y\neq G(x)} w_i^{(m)} = \exp(-\beta)\sum_{y=G(x)} w_i^{(m)}$$

$$\Rightarrow \log(\exp(\beta)\sum_{y\neq G(x)} w_i^{(m)}) = \log(\exp(-\beta)\sum_{y=G(x)} w_i^{(m)})$$

$$\Rightarrow \log(\exp(\beta)) + \log(\sum_{y\neq G(x)} w_i^{(m)}) = \log(\exp(-\beta)) + \log(\sum_{y=G(x)} w_i^{(m)}))$$

$$\Rightarrow \beta + \log(\sum_{y\neq G(x)} w_i^{(m)}) = -\beta + \log(\sum_{y=G(x)} w_i^{(m)}))$$

$$\Rightarrow 2\beta = \log(\sum_{y=G(x)} w_i^{(m)})) - \log(\sum_{y\neq G(x)} w_i^{(m)})$$

$$\Rightarrow \beta = \frac{1}{2}\log\frac{\sum_{y=G(x)} w_i^{(m)}}{\sum_{y\neq G(x)} w_i^{(m)}}$$

Now that we have an expression for β, we want to reformulate it, depending only on the weighted classification errors:

$$\Rightarrow \beta = \frac{1}{2}\log\frac{\sum_{y=G(x)} w_i^{(m)} + \sum_{y\neq G(x)} w_i^{(m)} - \sum_{y\neq G(x)} w_i^{(m)}}{\sum_{y\neq G(x)} w_i^{(m)}} \tag{15}$$

$$\Rightarrow \beta = \frac{1}{2}\log\frac{\sum_{i=1}^{N} w_i^{(m)} - \sum_{y\neq G(x)} w_i^{(m)}}{\sum_{y\neq G(x)} w_i^{(m)}} \tag{16}$$

$$\Rightarrow \beta = \frac{1}{2}\log\frac{(\sum_{i=1}^{N} w_i^{(m)} - \sum_{y\neq G(x)} w_i^{(m)}) \cdot \frac{1}{\sum_{i=1}^{N} w_i^{(m)}}}{\sum_{y\neq G(x)} w_i^{(m)} \cdot \frac{1}{\sum_{i=1}^{N} w_i^{(m)}}} \tag{17}$$

$$\tag{18}$$

By defining $\epsilon = \frac{\sum_{y\neq G(x)} w_i^{(m)})}{\sum_{i=1}^{N} w_i^{(m)}}$, we get that β_m (β at step m) is:

$$\beta_m = \frac{1}{2}\log\frac{1-\epsilon}{\epsilon} \tag{19}$$

The last missing piece is the weight update rule. By using the update rule of forward stagewise additive modelling:

$$f_m(x) = m_{(m-1)}(x) + \beta_m G_m(x) \qquad | -y \tag{20}$$

$$-yf_m(x) = -ym_{(m-1)}(x) + -y\beta_m G_m(x) \qquad | \exp() \tag{21}$$

$$\exp(-yf_m(x)) = \exp(-ym_{(m-1)}(x) + (-y\beta_m G_m(x))) \tag{22}$$

$$\exp(-yf_m(x)) = \exp(-ym_{(m-1)}(x)) \cdot \exp(-y\beta_m G_m(x)) \tag{23}$$

and the definition $w_i^{(m)} = \exp(-y_i f_{(m-1)}(x_i))$ we obtain:

$$w_i^{(m+1)} = w_i^{(m)} \cdot \exp(-\beta_m y_i G_m(x_i)). \tag{24}$$

Using the fact that $-y_i G_m(x_i) = 2 \cdot \mathbb{I}(y_i \neq G_m(x_i)) - 1$, this becomes

$$w_i^{(m+1)} = w_i^{(m)} \cdot \exp(\alpha_m \mathbb{I}(y_i \neq G_m(x_i))) \cdot \exp(-\beta_m), \tag{25}$$

where $\alpha_m = 2\beta_m$ and $\exp(-\beta_m)$ is a constant which multiplies all weights and thus has no effect.

Now we have all parts together for the AdaBoost.M1 algorithm:

Algorithm 1: AdaBoost.M1

```
1. Initialize the observation weights $w_i = \frac{1}{N}$, $i = 1, 2, \ldots, N$.
2. For $m = 1$ to $M$:
   a) Fit a classifier $G_m(x)$ to the training data using weights $w_i$.
   b) Compute $\epsilon = \frac{\sum_{y \neq G(x)} w_i^{(m)})}{\sum_{i=1}^{N} w_i^{(m)}}$.
   c) Compute $\alpha = \log \frac{1-\epsilon}{\epsilon}$.
   d) Set $w_i \leftarrow w_i \cdot \exp(\alpha_m \cdot \mathbb{I}(y_i \neq G_m(x_i))), i = 1, 2, \ldots, N$
Output $G(x) = \text{sign}(\sum_{m=1}^{M} \alpha_m G_m(x))$
```