Note: A Derivation of Discrete AdaBoost

Boris Babenko
Department of Computer Science and Engineering
University of California, San Diego
bbabenko@ucsd.edu

The purpose of this note is to show the derivation of the Discrete AdaBoost algorithm. Several good references are available for this topic, and this is meant to be just another view of the same result. Most of this derivation, however, is based on the derivations shown in [3] and [2].

1 Discrete AdaBoost

The term "boosting" refers to the process of taking a "weak" learning algorithm and boosting its performance by training many classifiers and combining them somehow. Generally, the model for the final "strong" classifier is a weighted voting or linear combination of the weak classifiers. Formally, suppose we are given a training set $D = \{(x_1, y_1), (x_2, y_2)...(x_n, y_n)\}$, where $x_i \in \mathbb{R}^D$ and $y \in \{+1, -1\}$. Our weak classifiers are functions $h : x \rightarrow y$. The strong classifier has the following form:

$$H(x) = \sum_{k=1}^{T} \alpha_k h_k(x) \tag{1}$$

The problem is then how to choose the T weak classifiers h_k. Freund and Schapire introduced the AdaBoost algorithm (adaptive boosting) in [1], which has since become very popular. Many other flavors of this algorithm have been introduced, but in this note we will only consider the original algorithm, which we refer to as Discrete AdaBoost.

Generally, we would like to minimize the 1/0 loss. This loss function, however, is not differentiable and therefore difficult to work with. Instead we will minimize the exponential loss, which is an upper bound on 0/1 loss. The exponential loss function is

$$\mathcal{L}(H) = \sum_{i=1}^{n} \exp\Big(- y_i H(x_i)\Big) \tag{2}$$

Suppose we have the first t weak classifiers, and now we are interested in choosing h_{t+1} and the corresponding weight α_{t+1}. Let $H_t(x) = \sum_{k=1}^{t} \alpha_k h_k(x)$ be the strong classifier composed of the first t classifiers. We can pose the $t+1$ iteration of AdaBoost as the following optimization:

$$\begin{aligned}
(h_{t+1}, \alpha_{t+1}) &= \arg\min_{h,\alpha} \mathcal{L}(H_t + \alpha h) \\
&= \arg\min_{h,\alpha} \sum_{i=1}^{n} \exp\Big(- y_i (H_t(x_i) + \alpha h(x_i))\Big) \\
&= \arg\min_{h,\alpha} \sum_{i=1}^{n} w_t(i) \exp\Big(- y_i \alpha h(x_i)\Big)
\end{aligned}$$

Note that the factor $e^{-y_i H_t(x_i)}$ does not depend on the arguments of the min operation, so we can regard it as a constant, which we can think of as a weight for each point at iteration t which we will denote $w_t(i)$. Furthermore, let's assume that these instance weights have been normalized such that $\sum_i w_t(i) = 1$, since the

normalization factor is a constant factor and again would not affect the min operation. Now let's continue to manipulate the algebra:

$$\begin{aligned}
(h_{t+1}, \alpha_{t+1}) &= \arg\min_{h,\alpha} \sum_{i=1}^{n} w_t(i) \exp\Big(- y_i \alpha h(x_i) \Big) \\
&= \arg\min_{h,\alpha} e^{-\alpha} \sum_{i|y_i = h(x_i)} w_t(i) + e^{\alpha} \sum_{i|y_i \neq h(x_i)} w_t(i) \\
&= \arg\min_{h,\alpha} (e^{\alpha} - e^{-\alpha}) \sum_{i=1}^{n} w_t(i) \mathbf{1}(y_i \neq h(x_i)) + e^{-\alpha} \sum_{i=1}^{n} w_t(i) \\
&= \arg\min_{h,\alpha} (e^{\alpha} - e^{-\alpha}) \sum_{i=1}^{n} w_t(i) \mathbf{1}(y_i \neq h(x_i)) + e^{-\alpha}
\end{aligned}$$

From the above we see that $h_{t+1} = \arg\min_h \sum_{i=1}^{n} w_t(i) \mathbf{1}(y_i \neq h(x_i))$ since $(e^{\alpha} - e^{-\alpha})$ is a positive number. Now we need to solve for α_{t+1}. Let $\epsilon_t = \sum_{i=1}^{n} w_t(i) \mathbf{1}(y_i \neq h(x_i))$ be the weighted error of the classifier h on the instances with weights w_t. We must solve the following equation:

$$\alpha_{t+1} = \arg\min_{h,\alpha} (e^{\alpha} - e^{-\alpha}) \epsilon_t + e^{-\alpha}$$

We can do this with simple calculus. We take the derivative of the right hand side and set it equal to 0:

$$\begin{aligned}
\frac{\partial}{\partial \alpha} (e^{\alpha} - e^{-\alpha}) \epsilon_t + e^{-\alpha} &= e^{\alpha} \epsilon_t + e^{-\alpha} \epsilon_t - e^{-\alpha} \\
&= e^{2\alpha} \epsilon_t + \epsilon_t - 1
\end{aligned}$$

Now setting this equal to zero to solve for the minimum:

$$\begin{aligned}
e^{2\alpha} \epsilon_t + \epsilon_t - 1 &= 0 \\
e^{2\alpha} &= \frac{1 - \epsilon_t}{\epsilon_t} \\
\alpha_{t+1} &= \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t}
\end{aligned}$$

We are almost at the point where we can write down an algorithm compactly. The last thing we will simplify is the expression for the instance weights. Recall that we set $w_t(i) = e^{-y_i H_t(x_i)} / Z_t$ where Z_t is some normalization factor such that the weights sum to 1. We could therefore set the weights for the next iteration to $w_{t+1}(i) = e^{-y_i (H_t(x_i) + \alpha_{t+1} h_{t+1}(x_i))} / Z_{t+1}$ and normalize again, but there is much redundant computation here that can be simplified with some more algebra.

$$w_{t+1}(i) = \exp\Big(- y_i (H_t(x_i) + \alpha_{t+1} h_{t+1}(x_i)) \Big) / Z_{t+1} \tag{3}$$

$$= w_t(i) Z_t \exp\Big(- y_i \alpha_{t+1} h_{t+1}(x_i) \Big) / Z_{t+1} \tag{4}$$

Now we can use a simple trick: $-y h(x_i) = 2 \times \mathbf{1}(y \neq h(x_i)) - 1$ [1]:

$$w_{t+1}(i) = w_t(i) Z_t \exp\Big(2 \alpha_{t+1} \mathbf{1}(y_i \neq h_{t+1}(x_i)) \Big) e^{-\alpha} / Z_{t+1}$$

[1] Note that the derivation in [2] does not use this trick and proceeds without this substitution. Of course, the final result is the same, but it may be a point of confusion.

Since Z_t and $e^{-\alpha}$ are constants (and recall that constants do not affect the optimization in 3), we can simply absorb them into Z_{t+1}.

$$\begin{aligned} w_{t+1}(i) &= w_t(i) \exp\Big(2\alpha_{t+1}\mathbf{1}(y_i \neq h_{t+1}(x_i))\Big)/Z_{t+1} \\ &= w_t(i) \exp\Big(\log(\frac{1-\epsilon_t}{\epsilon_t})\mathbf{1}(y_i \neq h_{t+1}(x_i))\Big)/Z_{t+1} \end{aligned}$$

Some references leave the above equation as shown above, but we can simplify it even further if we look at the normalization factor Z_{t+1}[2]:

$$\begin{aligned} Z_{t+1} &= \sum_{i|y_i \neq h_{t+1}(x_i)} w_t(i)(\frac{1-\epsilon_t}{\epsilon_t}) + \sum_{i|y=h_{t+1}(x_i)} w_t(i) \\ &= \epsilon_t(\frac{1-\epsilon_t}{\epsilon_t}) + (1-\epsilon_t) \\ &= 2(1-\epsilon_t) \end{aligned}$$

Putting everything together, the final simple expression for updating the weights comes out to

$$w_{t+1}(i) = \begin{cases} \frac{w_t(i)}{2(1-\epsilon_t)} & \text{if } y_i = h_{t+1}(x_i) \\ \frac{w_t(i)}{2\epsilon_t} & \text{if } y_i \neq h_{t+1}(x_i) \end{cases}$$

These results lead us to a very compact and simple algorithm that is Discrete AdaBoost:

Algorithm 1 Discrete AdaBoost

Input: Training data D, number of iterations T, and an initial distribution $w_0(i)$ over the examples.
For $t = 1, ..., T$:

- Train a weak classifier $h_t = \arg\min_h \sum_{i=1}^n w_{t-1}(i)\mathbf{1}(y_i \neq h(x_i))$.
- Calculate the error of $h_t : \epsilon_t = \sum_{i=1}^n w_{t-1}(i)\mathbf{1}(y_i \neq h_t(x_i))$.
- Set $\alpha_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}$.
- Set $w_{t+1}(i) = \begin{cases} \frac{w_t(i)}{2(1-\epsilon_t)} & \text{if } y_i = h_{t+1}(x_i) \\ \frac{w_t(i)}{2\epsilon_t} & \text{if } y_i \neq h_{t+1}(x_i) \end{cases}$

Output: Strong classifier $H(x) = \sum_{k=1}^T \alpha_k h_k(x)$.

References

[1] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[2] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting (With discussion and a rejoinder by the authors). *Ann. Statist*, 28(2):337–407, 2000.

[3] T. Hastie, R. Tibshirani, J. Friedman, et al. *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2001.

[2]Again it is important to note that in [2] the normalization factor is slightly different because some of the algebra steps to get here are different.