Boosting:

Or How to Make a Silk Purse Out of a Sow’s Ear

Harrison B. Prosper

Florida State University, Tallahassee, Florida 32306

(Dated: April 16, 2005)

Abstract

Boosting is a relatively new technique, recently introduced into high energy physics, from the field of machine learning. It is a general method of creating an ensemble of classifiers that collectively achieve classification error rates arbitrarily lower than that of any one classifier, at least on the training sample, provided that each classifier performs at least slightly better than random guessing. In this note, I give a brief, accessible, introduction to boosting.
I. INTRODUCTION

Event discrimination, or classification, is an important step in any realistic analysis. In this note, I consider the canonical task of assigning events to one of two classes, signal \((S)\) or background \((B)\), on the basis of a vector of discriminating variables \(x\), for each event, and a function \(F(x)\), called variously a classifier, discriminant, hypothesis, or learner. The classifier \(F(x)\) can always be designed so that signal events tend to yield positive values, while background events tend to yield negative values. Presumably, the goal of event classification is to classify events with the smallest possible error rate. To that end, consider the function

\[
F(x) = g(B(x)), \tag{I.1}
\]

where \(g(\cdot)\) is any one-to-one function and

\[
B(x) = \frac{p(S|x)}{p(B|x)}, \tag{I.2}
\]

\[
= \frac{p(x|S)p(S)}{p(x|B)p(B)}, \tag{I.3}
\]

is the Bayes Discriminant, with \(p(x|S)\) and \(p(x|B)\) the signal and background probability densities, respectively, and \(p(S)/p(B)\) the signal to background ratio. Equations (I.1) and (I.2) is the complete solution to the problem of constructing an optimal classifier; that is, one with the smallest possible error rate. Any classifier that does as well as the Bayes discriminant is said to achieve the Bayes Limit. Two points deserve particular attention: no classifier, however sophisticated, can do better than the Bayes limit and any that does reach the Bayes limit must be isomorphic to the Bayes discriminant. Consequently, if one has found a method that reaches the Bayes limit there is no point looking for something better, except perhaps in the hope of finding a faster method that performs as well.

The standard approach to classification is to try, as it were, to cut a good classifier from a single cloth [2]. However, in the field of machine learning (see for example, Ref. [3]) a different strategy is under active investigation: instead of trying to construct a good classifier directly, one makes do with an ensemble of classifiers, each of which need perform only slightly better...
than a classifier based on the results of random coin tosses. The advantage of doing this, of course, is that weak classifiers are easy to find. The key idea is the realization that an ensemble of weak classifiers can be made to perform much better than any individual. This is the basic idea behind boosting.

II. BOOSTING

Boosting is a general technique that can be applied to any classifier. See, for example, Ref. [1], which describes the first application of boosting in high energy physics using decision trees as the underlying classifiers. Consider a training sample $T[y, x, w]$ in which the $n^{\text{th}}$ event is characterized by a label $y_n$ with value $+1$ for signal and $-1$ for background, a vector of discriminant variables $x_n$ and an event weight $w_n$. The most general boosting algorithm can be written as

$$T_1 = \text{initialize}().$$

For $k$ in $1, \ldots, K$

$$\alpha_k, f_k = \text{train}(T_k),$$
$$T_{k+1} = \text{modify}(T_k).$$

$$F \equiv h(f_1, \ldots, f_K),$$

where $f_k$ are the individual (weak) classifiers, associated with weights $\alpha_k$, and $h(\cdot)$ is some function thereof, usually, a weighted sum, so that

$$F(x) = \sum_{k=1}^{K} \alpha_k f_k(x, T_k). \quad (\text{II.1})$$

The crucial thing to note is that each classifier $f_k$ depends, in general, on a different modification of the training data $T$. The clever part is to create modifications that yield reduced classification error rates, on the training sample, when using the ensemble-based classifier $F$. One of the most successful algorithms in this regard, AdaBoost, is due to Freund and Schapire [4] and is the focus of the rest of this note.
III. ADABOOST

In AdaBoost, the steps initialize, train and modify proceed as follows:

1. initialize

   Initialize the training sample $T_1 = T[y, x, w]$ with the event weights, $w_{1,n}$, normalized so that $\sum_{n=1}^{N} w_{1,n} = 1$.

2. train

   Train a classifier using the training sample $T_k(y, x, w_k)$ and a method that makes use of event weights. Associate an (optimal) weight $\alpha_k$ with classifier $f_k$.

3. modify

   Modify the training sample by updating the weights as follows:
   
   \[ w_{k+1,n} = w_{k,n} \frac{e^{-\alpha_k f_k,n y_n/2}}{Z_k}, \]  
   
   (III.1)

   where the normalization factor
   
   \[ Z_k = \sum_{n=1}^{N} w_{k,n} e^{-\alpha_k f_k,n y_n/2}, \]  
   
   (III.2)

   ensures that the weights $w_{k+1,n}$ sum to unity.

Numerous studies indicate that the function $F(x)$ (Eq. (II.1)), constructed in this way, works remarkably well (see for example, Ref. [1]). Why this is so, is discussed next.

We first note that the sign of $F(x)$ can be used as a classifier: if $F(x) > 0$, we classify the event as a signal, otherwise as a background. Therefore, since $y_n F(x_n) > 0$ for a correct classification and negative for a wrong one, the classification error rate $\epsilon$, on the training sample, can be expressed as

\[ \epsilon = \sum_{n=1}^{N} w_{1,n} 1[y_n F(x_n) \leq 0], \]  

(III.3)
where the indicator function \( I[X] = 1 \) if the statement \( X \) is true, zero otherwise. Unfortunately, the direct minimization of \( \epsilon \) is generally not feasible in a reasonable amount of time. Instead one tries to minimize a cost function \( C \) that is a suitable proxy for the error rate. To that end, consider the function \( \exp(-y_n F_n/2) \), where \( F_n \equiv F(x_n) \). This function is \( \geq 1 \) whenever \( y_n F_n \leq 0 \) and \( < 1 \) otherwise. This property leads to the inequality

\[
\epsilon \leq \sum_{n=1}^{N} w_{1,n} e^{-y_n F_n/2} I[y_n F_n \leq 0],
\]

(III.4)

In the limit of an infinite training sample, \( N \to \infty \), we may write

\[
\epsilon \leq \mathbb{E}[e^{-yF(x)/2}],
\]

(III.5)

where \( \mathbb{E}[\cdot] \) is the expectation operator with respect to the probability density function of \( x \). We conclude that the minimization of the training error \( \epsilon \) can be done indirectly by minimizing the expectation value of the function \( \exp(-yF(x)/2) \). I shall return to this point. But first, we must determine how to find \( \alpha_k \).

From the recursive definition, Eq. (III.1), of the event weights \( w_{k,n} \), we can write

\[
w_{K+1,n} = w_{1,n} \prod_{k=1}^{K} \frac{e^{-\alpha_k f_{k,n} y_n/2}}{Z_k} = w_{1,n} \frac{e^{-\sum_{k=1}^{K} \alpha_k f_{k,n} y_n/2}}{\prod_{k=1}^{K} Z_k} = w_{1,n} e^{-y_n F_n/2} \frac{1}{\prod_{k=1}^{K} Z_k},
\]

(III.6)

But, since by construction, \( \sum_{n=1}^{N} w_{K+1,n} = 1 \),

\[
\sum_{n=1}^{N} w_{1,n} e^{-y_n F_n/2} = \prod_{k=1}^{K} Z_k,
\]

(III.7)

and, therefore, from Eq. (III.4),

\[
\epsilon \leq \prod_{k=1}^{K} Z_k.
\]

(III.8)
Thus, we can minimize the training error $\epsilon$ indirectly by minimizing the function $Z = \prod_{k=1}^{K} Z_k$. For a given sequence of classifiers $f_1, \ldots, f_K$, $Z$ can be minimized by taking its derivative
\[
\frac{\partial Z}{\partial \alpha_k} = -\frac{1}{2} \left( \prod_{j \neq k} Z_j \right) \sum_{n=1}^{N} w_{k,n} f_{k,n} y_n e^{-\alpha_k f_{k,n} y_n / 2},
\]
with respect to each classifier weight $\alpha_k$ and setting it to zero; that is, by solving
\[
\sum_{n=1}^{N} w_{k,n} f_{k,n} y_n e^{-\alpha_k f_{k,n} y_n / 2} = 0,
\]
usually, numerically. If, however, the values of the classifiers $f_k$ are restricted to the binary set $\{+1, -1\}$, Eq. (III.10) can be written as
\[
e^{-\alpha_k / 2} \sum_{n=1}^{N} w_{k,n} I[f_{k,n} y_n > 0] - e^{+\alpha_k / 2} \sum_{n=1}^{N} w_{k,n} I[f_{k,n} y_n \leq 0] = 0,
\]
which allows for an exact solution. If we define the weighted training error rate $\epsilon_k$, associated with the $k$th classifier $f_k$, by
\[
\epsilon_k \equiv \sum_{n=1}^{N} w_{k,n} I[f_{k,n} y_n \leq 0],
\]
we can write Eq. (III.11) as
\[
e^{-\alpha_k / 2}(1 - \epsilon_k) - e^{+\alpha_k / 2} \epsilon_k = 0,
\]
from which it follows that
\[
\alpha_k = \ln \left( \frac{1 - \epsilon_k}{\epsilon_k} \right).
\]
When inserted into Eq. (III.2) the above yields
\[
Z_k = 2 \sqrt{\epsilon_k (1 - \epsilon_k)},
\]
and from Eq. (III.8) the bound
\[
\epsilon \leq \prod_{k=1}^{K} 2 \sqrt{\epsilon_k (1 - \epsilon_k)},
\]
on the error rate for the training sample $\mathcal{T}$. Freund and Schapire [4] consider classifiers for which $\epsilon_k = \frac{1}{2} - \gamma_k$, with $\gamma_k > 0$. The latter quantity, called the weak edge, is a measure of how much better a classifier is than one based on random guessing. Clearly, a useful classifier is one whose weak edge is always positive. In terms of the weak edges, the training error bound can be written as

$$\epsilon \leq \prod_{k=1}^{K} \sqrt{1 - 4\gamma_k^2},$$

$$\leq (1 - 4\gamma^2)^{K/2},$$

$$\rightarrow e^{-2K\gamma^2}, \quad K \rightarrow \infty. \tag{III.17}$$

The last couple of steps follow if $\forall k, \gamma_k > \gamma > 0$ and if the ensemble of classifiers is large enough.

Equation (III.17) implies the rather extraordinary bottom line:

- provided that the ensemble is large enough, and every classifier therein does at least marginally better than random guessing, the error rate on the training sample falls exponentially to zero as the sample size grows to infinity.

This result, it seems to me, shows that a sequential construction of $F(x)$ is not actually necessary. What matters is that the weak edges have a guaranteed bound away from zero, that is, $\gamma_k > \gamma > 0$. In principle, therefore, any large ensemble of, possibly disparate, classifiers would work provided that each has a weak edge $> \gamma$. Another point is worth making and it is this: if, in fact, the weak edges are bounded away from zero, the boosting algorithm of Freund and Schapire will eventually fit the finite training data exactly [5], that is, over-fit. Consequently, the error rate on test samples must necessarily reach a non-zero minimum for some ensemble size and, perhaps, rise thereafter. So although, in practice, the AdaBoost algorithm seems resistant to over-fitting, the result in Eq. (III.17) suggests that at some point AdaBoost must over-fit. Therefore, the sensible strategy is to keep augmenting the ensemble of classifiers, but at each step validate $F(x)$ on a test sample and stop just before the test error starts to rise or before one runs out of computer memory!
IV. PROBABILISTIC INTERPRETATION

The minimization of the AdaBoost cost function in Eq. (III.5) leads to an elegant probabilistic interpretation [6], which is sketched here. The AdaBoost cost function can be written as

$$C = \int p(x, y) e^{-yF(x)/2} dx dy,$$  \hspace{1cm} (IV.1)

where the density $p(x, y) = p(x, y = +1|S)p(S) + p(x, y = -1|B)p(B)$. The signal density can be factorized further: $p(x, y = +1|S)p(S) = p(y = +1|x)p(x|S)p(S)$ and likewise for the background density. Note that the probability for $y = +1$ is 1 for $x \in S$ and zero otherwise and similarly for the background. Therefore, the integration over $y$ is immediate and yields

$$C = \int p(x|S)p(S)e^{-F/2} dx + \int p(x|B)p(B)e^{+F/2} dx. \hspace{1cm} (IV.2)$$

To minimize $C$ we must take its functional derivative

$$\frac{\delta C}{\delta F} = -\frac{1}{2} \int p(x|S)p(S)e^{-F/2} dx + \frac{1}{2} \int p(x|B)p(B)e^{+F/2} dx,$$

$$= \frac{1}{2} \int [-p(S|x)e^{-F/2} + p(B|x)e^{+F/2}]p(x) dx, \hspace{1cm} (IV.3)$$

with respect to $F$ and set it to zero, if possible. In the last step we have availed ourselves of Bayes’ theorem $p(A|x) = p(x|A)p(A)/p(x)$. If it is possible for the functional derivative to be zero, which requires the function $F$ to be sufficiently malleable, then, since $p(x) > 0$, $\forall x$, the only way to get zero, in Eq. (IV.3), is if the term in brackets is zero $\forall x$, that is, if

$$p(S|x) = \frac{1}{1 + \exp(-F(x))}. \hspace{1cm} (IV.4)$$

We see, explicitly, that, provided we have enough training data and provided the function $F$ is flexible enough, $F$ is just a Bayes discriminant in disguise, as it must be if it reaches the Bayes limit. Moreover, Eq. (IV.4) has the same logistic form as a feed-forward neural network [2], with output bounded to the interval $[0,1]$. 
SUMMARY AND CONCLUSIONS

Boosting is a general technique for marshaling an ensemble of weak classifiers into a single effective one. In principle, it can be applied to any classifier. However, if a classifier is already at, or is close to, the Bayes limit, boosting, or any other classifier enhancer—however sophisticated, is unlikely to be effective since the Bayes limit cannot be breached. If on the other hand the classifiers have weak edges that skirt dangerously close to zero, boosting may become impractical if the ensemble size grows prohibitively large.

The message, as always, is to try it and see!

REFERENCES

[7] In the machine learning literature there is no factor of 2. I’ve introduced it to render the subsequent mathematics a bit tidier.