On Boosting and the Exponential Loss

Abraham J. Wyner
Statistics Dept.
Wharton School
University of Pennsylvania
Philadelphia, PA 19104
ajw@wharton.upenn.edu

Abstract

Boosting algorithms in general and AdaBoost in particular, initially baffled the statistical world by posing two questions: (1) Why is it that AdaBoost performs so well? and (2) What makes Boosting methods resistant to overfitting? In response to question (1) Freidman, Hastie and Tibshirani (2000) take a statistical view of Boosting by recasting it as a stagewise approach to the minimization of an exponential loss function by means of an additive model in a process similar to additive logistic regression. This characterization has since been well integrated in the statistics and computer science communities as the best statistical answer to question (1). In this paper, we argue that this well assimilated view is questionable and that perhaps Boosting's success has nothing to do with the minimization of an exponential criterion or indeed any optimization at all. Our argument rests on a constructive theorem that states that for any sequence of classifiers there exists a linear combination for which the exponential criterion equals one. Furthermore, we present a Boosting algorithm which performs empirically like AdaBoost while stabilizing the exponential loss to a constant.

1 ON THE EXPONENTIAL LOSS

The performance of Boosting surprised statisticians. AdaBoost (Freund and Schapire, 1996) works well-really well, on a huge variety of problems in various different contexts. Yet it achieves this success in defiance of several important statistical paradigms. Its program of model selection is haphazard, random and unregulated. It shrinks the training set error rate to zero, yet, paradoxically, it continues to improve its out of sample performance even after this mark is obtained.

Finally, It seems remarkably resisitant to overfitting in many if not all contexts, even though it never validates on hold out data, never estimates an out of sample error rate, or otherwise considers performance on anything other than the training data.

Friedman, Hastie and Tibshirani (Friedman, et. al 2000) take great strides to clear up the mystery of boosting to provide statisticians with a statistical view of the subject. The heart of their article is the recasting of boosting as a statistically familiar program for finding an additive model by means of a forward stagewise approximate optimization of an exponential criterion.

This explanation has been widely assimilated and has reappeared in the statistical literature (see for example Friedman et al. 2001) as well as in a plethora of computer science articles (for example Schapire's overview 1999). Further, there has been very interesting work connecting the optimization realized by AdaBoost to other optimizations (see Collins et. al (2000) for example).

1.1 STATISTICAL VIEWS OF BOOSTING

Let's review this argument. The statistical view in Friedman et al. 2000 begins by asking a statistical question: Given a space of classifiers G, consider linear combinations of the form

$$f(x) = \sum_{m=1}^{M} \alpha_m g_m(x) \text{ with } g_m \in G \text{ and } \alpha_m \in R.$$

How does one select g_m, α_m and M to optimize an exponential criterion $\exp(-yf(x))$? Here and throughout this discussion, we will consider x a vector of predictors, $y \in \{-1, 1\}$ a binary class label, and T a training set of N pairs $\{x_i, y_i\}$ for $i = 1, .., N$. Since global minimization is usually not possible, a forward stagewise optimization is proposed. Given

$$f_{m-1} = \sum_{i=1}^{m-1} \alpha_i g_i(x)$$

the goal is transformed to finding g_m and α_m that minimize

$$\sum_{i=1}^{N} \exp(-y_i(f_{m-1}(x_i) + \alpha_m g_m(x_i))).$$

In light of the defining property of the exponential, namely factorization , the previous problem is reduced to a *weighted* minimization. That is

$$\{\alpha_m, g_m\} = arg \min_{\alpha, g \in G} \sum_{i=1}^{N}$$

$$= arg \min_{\alpha, g \in G} \exp(-\alpha) \sum_{i=1}^{N} w_{m-1}(i)$$

$$+(e^{\alpha} - e^{-\alpha}) \sum_{i=1}^{N} w_{m-1}(i) 1\{y_i \neq g(x_i)\}$$

with

$$w_{m-1}(i) \exp(-y_i \alpha g(x_i)) / \sum_{i=1}^{N} \exp(-y_i f_{m-1}(x_i))$$

defined to be weights. The AdaBoost algorithm is recovered by noting that this minimum is found by letting

$$g_m = arg \min_{g \in G} \sum_{i=1}^{N} w_{m-1} 1\{y_i \neq g(x_i)\}$$

while assuming that every $g \in G$ maps to $\{-1, 1\}$. Taking a single derivative leads to the solution

$$g_m = arg \min_{g \in G} \sum_{i=1}^{N} w_{m-1}(i) 1\{y_i \neq g(x_i)\}$$

and

$$\alpha_m = \frac{1}{2} \log \frac{(1 - \epsilon_m)}{\epsilon_m},$$

where

$$\epsilon_m = \sum_{i=1}^{N} w_{m-1}(i) 1\{y_i \neq g_m(i)\}$$

is the weighted error rate on the training set by classifier g_m. At the next stage, the weights are recomputed by an updating procedure where each correctly classifier training example gets down weighted by the factor $\exp(-\alpha_m)$ and each incorrect example gets up weighted by the factor $\exp(-\alpha)$. This is AdaBoost.

Schapire and Freund (see [4]) arrive at the same solution and calculation but in a different context. Since one can construct a binary classifier by taking the sign of a linear combination of real values functions, the training set error rate $\epsilon(M)$ of an M term linear combination f_M, can be re-written in the following form:

$$\epsilon(M) = \sum_{i=1}^{N} 1\{y_i f(x_i) < 0\}.$$

Freund and Schapire observe that

$$\sum_{i=1}^{M} 1\{y_i f_M(x_i) < 0\} \leq \sum_{i=1}^{M} \exp(y_i f(x_i)).$$

Thus, a stagewise minimization of the left hand side (*training set error rate*) is bounded by a stagewise minimization of the right hand side (the exponential) and the resulting algorithm is AdaBoost.

In summary, the exponential function's identification with Adaboost is uncontested. Nevertheless, minimization of the exponential loss is sensible only in light of the *population* version of the algorithm for which it is proven (see Friedman et al. 2001) that optimization of the exponential criterion results in estimates of (half) the log odds ratio. Thus, it would appear that taking the sign of such an estimator (as Adaboost does) would result in the optimal Bayes classifer.

In summary, we have recounted the connection between stagewise optimization of the exponential loss and Adaboost. Unanswered is the question of whether optimization of the exponential on *training data* will result in *population* optimization as well.

2 STABILIZING THE LOSS

From the perspective of the exponential loss AdaBoost should be a disasterous overfit, since it never validates or otherwise considers hold out samples. Furthermore, it computes the loss exclusively on the data while ignoring any consideration of its expectation over the population. In more traditional statistical contexts this ignorance may not be problematic. For example, the construction of a linear regression fits a function to the data (the least squares line) which is proven to be good only in expectation (the conditional expectation is the population minimizer). In small parameter regression contexts this is rarely a concern since the function space is small and a fit to the data is likely to be a good fit out of sample. Not so in our context, where not only are the base classifiers often strong learners, but the resulting linear combination of classifiers are almost always sufficiently rich enough to separate the training data, even in high dimensions. To wit, Yali Amit has observed (in private communication) that the loss can increase exponentially on a hold out sample while decaying exponentially on the training data. Breiman (see his discussion on [1]) has expressed his misgivings with this explaination as well.

In this paper I will argue that Breiman's intuition was correct: that minimization of an exponential criterion has less to do with the success of Adaboost than it appears. My case rests on one simple theorem and an empirical observation. We start with the theorem.

Consider, as before, a set of binary classifiers G, such that every $g \in G$ is a mapping of vectors x in some p-dimensional feature space to the binary variable $y \in \{-1, 1\}$.

Theorem A: Let $g_m \in G$ from $m = 1$ to M be any arbitrary sequence of classifiers. Let $T = \{x_i, y_i\}_{i=1}^N$ be a training set of N observations. Starting with $f_0 = 0$ recursively define $f_m = \sum_{i=1}^m \alpha_i g_i$ and $\alpha_m = \beta \log \frac{1-\epsilon_m}{\epsilon_m}$ where

$$\epsilon_m = \sum_{i=1}^N \frac{1}{N} \exp(-y_i f_{m-1}(x_i)) 1\{y_i \neq g_m(x_i)\}.$$

It follows that for all M

$$\sum_{i=1}^N \frac{1}{N} \exp(\frac{1}{\beta} y_i f_M(x_i)) = 1.$$

Note: The value of ϵ_m denotes the weighted error rate of g_m on training set T using the accumulated exponential loss at each observation x_i by the linear combination f_{m-1}.

Theorem A implies that *any* sequence of classifiers can be combined linearly while holding the exponential loss on the data constant. We exploit this fact to produce a *good* linear combination of classifiers that is indifferent to the exponential criterion. To this end we construct the following algorithm:

Beta-Boosting: Let $w_0 = \frac{1}{N}$. Then for $m = 1$ to M do:

1. $g_m = \text{argmin}_{g \in G} \sum_{i=1}^N w_{m-1} 1\{y_i \neq g(x_i)\}$
2. Let ϵ_m be the weighted error rate achieved by g_m
3. $\alpha_m = \frac{1}{\beta} \log \frac{1-\epsilon_m}{\epsilon_m}$.
4. $w_m(i) = w_{m-1}(i) \exp(-y_i g_m(x_i) \alpha_m)$.
5. Renormalize $w_m(i)$.

Output final classifier

$$F(x) = sign[\sum_{m=1}^M \beta \alpha_m g_m(x)].$$

Observe that Beta-Boosting is almost Adaboost. The main difference is that we have generalized the factor of $1/2$ from the definition of α_m which results in an updating scheme whose weights are the *beta* powers of Adaboost's weights. Theorem A applies in the special case of $\beta = 1$ (which we call "square-boosting" (since its weights are the squares of Adaboost's), for which it follows that at every stage the weights always sum to 1 (so normalization is not required) and that the exponential criterion remains constant at 1 regardless of the number of iterations. Beta boosting can be viewed in the context of optimization theory as well. The optimal program with respect to the exponential is $\beta = 1/2$. Square boosting is a case of Successive Over Relaxation (SOR). In this particular case, however, the doubling of the step size leads not only to over-relaxation but excessive relaxation. Square-Boosting is thus not an optimization.

Like Adaboost, the SquareBoost algorithm is a stagewise scheme based on modular reweighting for constructing a linear combination of classifiers fitted on perturbed training sets. Adaboost will drive the exponential criterion to zero exponentially quickly, while Squareboosting keeps the criterion constant at one. More importantly, the criterion is identically one for *all* binary weak learners, not necessarily the one that achieves the lowest weighted training error. Accordingly, we can expect that SquareBoosting should fail to Boost.

Proof of Theorem A: Let $f_0 = 0$ we proceed by induction. For $m = 0$ the result is trivial. Now, given $f_{m-1} = \sum_{i=1}^n \alpha_i g_i$. Let $w_{m-1}(i) = \exp(-y_i f_{m-1}(x_i))/sum_{i=1}^N \exp_{-y_i f_{m-1}(x_i)}$. Using the fact that $y_i g_m(x_i)$ is -1 if $g_m(x_i) \neq y_i$ and 1 otherwise, we have

$$\sum_{i=1}^N \exp(-y_i(f_{m-1} + \alpha_m g_m(x_i))) =$$

$$\sum_{i=1}^n w_{m-1}(i) \exp(-y_i \alpha_m g_m(x_i))$$

$$= \sum_{i=1}^N w_{m-1}(i) \exp(\alpha_m) 1\{y_i \neq g_m(x_i)\} +$$

$$\sum_{i=1}^N w_{m-1}(i) \exp(-\alpha_m)[1 - 1\{y_i = g_m(x_i)\}]$$

$$= \frac{1-\epsilon_m}{\epsilon_m}\sum_{i=1}^{N} w_m(i)1\{y_i = g_m(x_i)\}+$$

$$\frac{\epsilon_m}{1-\epsilon_m}\sum_{i=1}^{N} w_{m-1}(i)[1 - 1\{y_i = g_m(x_i)\}]$$

$$= \frac{1-\epsilon_m}{\epsilon_m}\epsilon_m + \frac{\epsilon_m}{1-\epsilon_m}(1-\epsilon_m)$$

$$= 1 - \epsilon_m + \epsilon_m$$

$$= 1$$

Thus the exponential criterion equals one for all M and for *arbitrary* choice of classifiers g_m, which prove the theorem. Note: If the initial weights are normalized then the sum of the weights will sum to one and renormalization will not be necessary.

3 PERFORMANCE

SquareBoost, by construction, will find a linear combination of classifiers by a stagewise process that is indifferent to the exponential criterion. This sets it apart from LogitBoost, GentleBoost (see FHT, 2000) and Mart (Friedman, 2001). Yet the surprising fact is that SquareBoosting is apparently as successful a classifer as Adaboost. We present SquareBoost tested on examples from the UCI Irvine repository using C4.5 as the base classifier. In all of them, Squareboosting exhibits the properties of AdaBoost: excellent performance (relative to C4.5), training set error rates that reach zero, and test set error rates that continue to drop even after the training error has reached zero. In this draft we include three examples: CAR, SATIMAGE, and SEGMENTATION (details of the simulation appear in Table 1). Other examples (not shown) reveal the same behavior. In all three examples we examine the performance of the algorithm as measured by test set error rate. We then plot the error rate as a function of the number of iterations. Three features clearly stand out:

1. In the examples, the test set error rate of SquareBoost jumps up after the second iteraration, but then descends well below the C4.5 error rate. This is likely due to the fact that SquareBoost's reweighting stategy is more aggressive than AdaBoost in the early stages.

2. The test set error rate of SquareBoost is essentially equivalent to that of AdaBoost after 225 iterations.

3. Performance of SquareBoost continues to improve for many iterations after the training set error rate is zero.

Table 1: Description of three data sets from the UC Irvine Repository

Dataset	Dataset size	Training set size	Attributes Cont/Discr	Classes
car	1728	1000	0/6	4
satimage	4435	2000	36/0	6
segmentation	2310	1155	19/0	7

4. The performance of AdaBoost and SquareBoosting after 225 iterations is statistically equivalent. AdaBoost slightly outperforms SquareBoost in Satimage (Figure 1), SquareBoost outperforms AdaBoost in Car (Figure 3) and the two are nearly the same in Segmentation (Figure 2)

SquareBoosting is actually a special case of Beta-Booting with $\beta = 1$, that is weights that are simply the squares of AdaBoost's weights. This choice is interesting since it seems to perform as well as AdaBoost while stabilizing the exponential criterion. We now consider two other special cases: "root" Boosting, that is Beta-Boosting for $\beta = 1/4$ (weights that are the square roots of AdaBoost) and Quad-Boosting which uses weights that are the fourth powers of AdaBoost. Qualitatively, Root boosting is akin to GentleBoost since the step size in the "optimization" is halved, in many examples a much more gradual fitting to the training set is observed, unfortunately the overall performance is many example (see for example Figure 4) tends to be quite poor relative to AdaBoost. Quad boosting is too aggressive. In all examples considered (See Figure 5, other examples not shown) its performance is weakest relative to the other examples, although is may be possible to improve its performance by means of a regularization algorithm.

4 CONCLUSIONS

We have proved that there exists a stagewise forward program for constructing a linear combination of classifiers that is indifferent to the exponential criterion. Nevertheless, this algorithm is nearly AdaBoost in definition and in practice. This forces a reconsideration of the connection between Boosting algorithms and statistical optimization. Another view suggests that despite the difference in their respective treatment of the exponential criterion, both algorithms are actually programs for sequentially perturbing the data, fitting classifiers and averaging the result. The averaging is smoothing and that prevents overfit (see Krieger et al 2001). A closer look at other stagewise programs for finding a linear combination, for example GentleBoost and LogitBoost, reveal the same underlying structure of perturbation and averaging. What we have accomplished in this paper is a mathematical justification for the lingering doubt among some in the Boosting community that optimization is an explanation for AdaBoost.

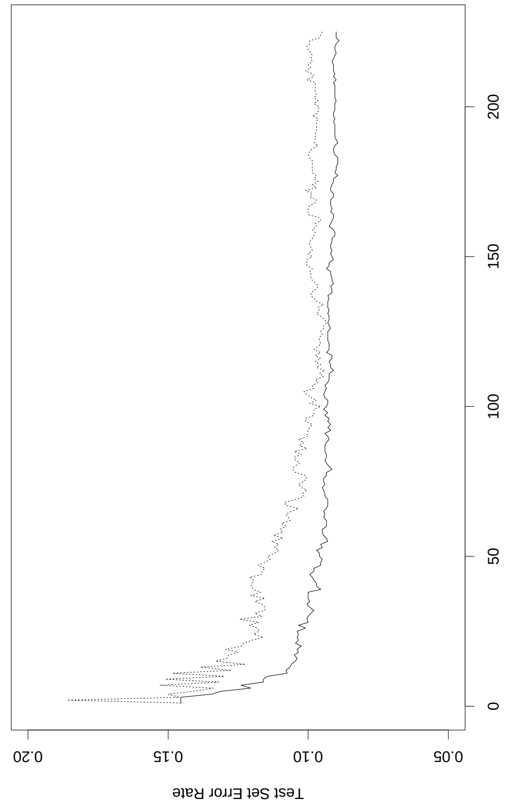

Figure 1: Comparison of SquareBoost (dashed line) and AdaBoost (solid line) applied to Satimage dataset

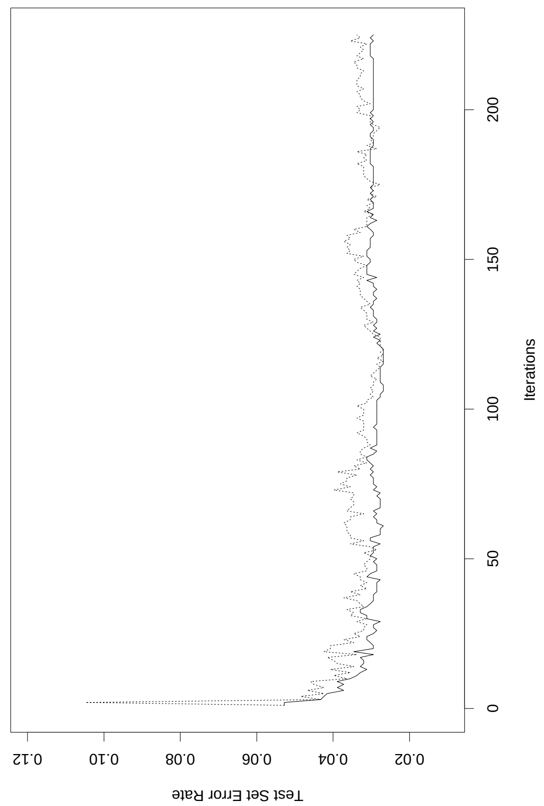

Figure 2: Comparison of SquareBoost (dashed line) and Adaboost(solid line) applied to Segmentation dataset

Acknowledgements

The author would like to acknowledge Andreas Bujas, Yoram Singer and Abba Krieger for many helpful conversations and discussions.

References

L. Breiman (2000), Discussion on Statistical View of Boosting, *The Annals of Statistics*, Vol. 28, No. 2.

Michael Collins, Robert E. Schapire, Yoram Singer, Logistic Regression, AdaBoost and Bregman Distances (2000) Computational Learing Theory.

Y.Freund and R.E. Schapire. (1996) Experiments with a new Boosting Algorithm. Machine Learning: Proceedings of teh Thirteenth International Conference 148-156, Morgan Kaufman, San Francisco.

T. Hastie, R. Tibshirani, and J. Friedman (2000). A Statistical View of Boosting, *The Annals of Statistics* 2000, Vol. 28,No. 2.

T. Hastie, R. Tibshirani, and J. Friedman. (2001) The Elements of Statistical Learning: data mining, inference and prediction. Springer, 2001.

A. Kreiger, C. Long and A.Wyner (2001) Boosting Noisy Data, Proceedings of International Conference on Machine Learning, Morgan Kaufman, July, 2001 .

Robert E. Schapire. (1999) Theoretical views of boosting and applications. In Tenth International Conference on Algorithmic Learning Theory, 1999.

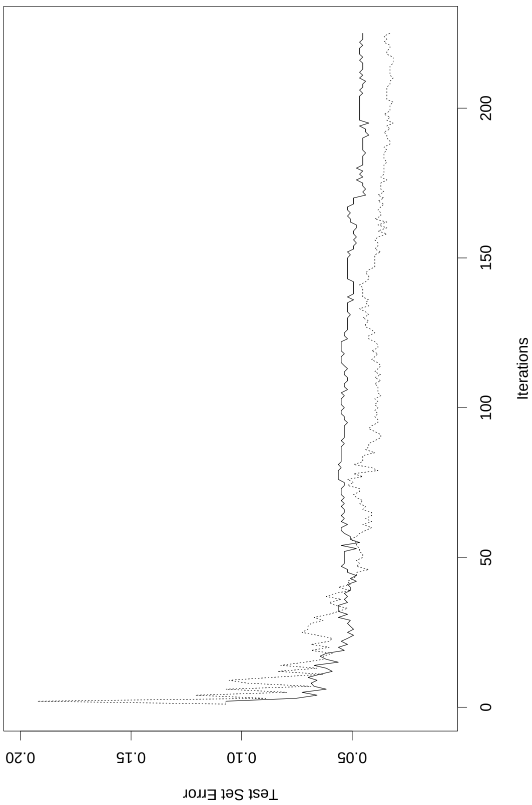

Figure 3: Comparison of SquareBoost (dashed line) and Adaboost (solid line) applied to Car dataset

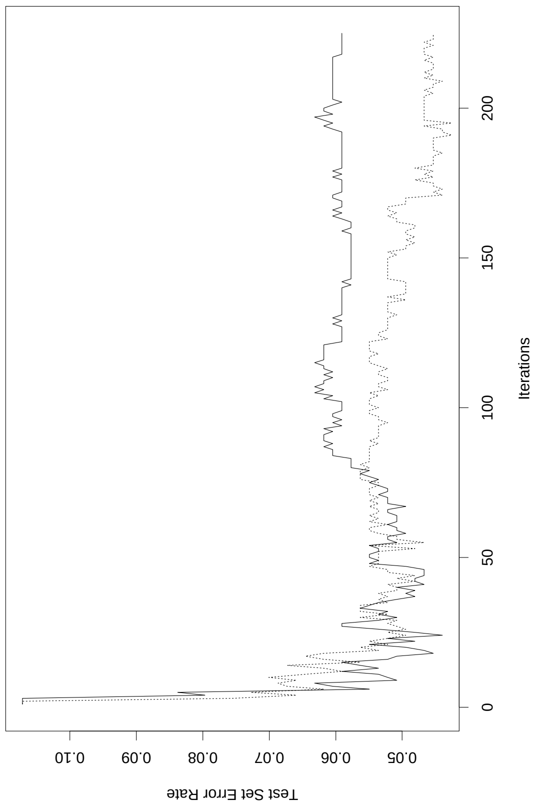

Figure 4: Comparison of RootBoost (Solid line) and Adaboost (Dashed line) applied to Car dataset

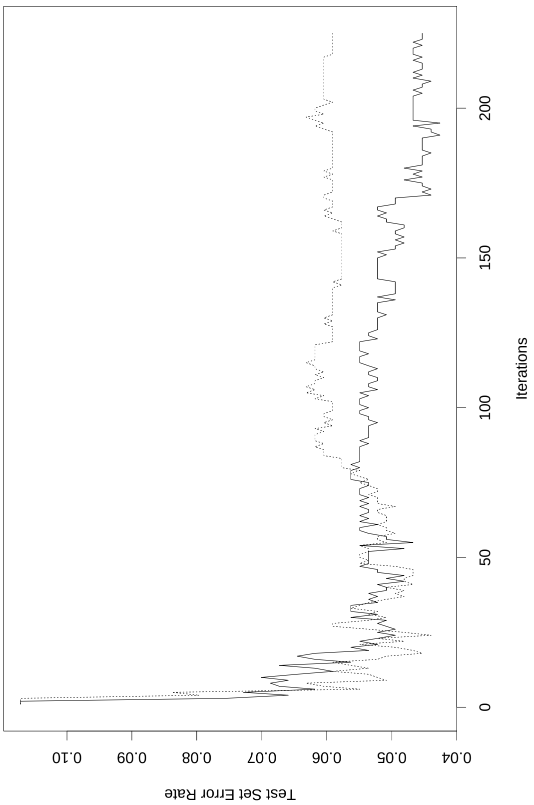

Figure 5: Comparison of QuadBoost (Dashed line) and Adaboost (solid line) applied to Car dataset