Discussion. Additive logistic regression: a statistical view of boosting, by Friedman, J., Hastie, T. and Tibshirani, R.

Peter Bühlmann
ETH Zürich, Switzerland

Bin Yu
Bell Laboratories, Murray Hill, Lucent Technologies and University of California at Berkeley

We congratulate Friedman, Hastie and Tibshirani (FHT) for their significant work connecting boosting and (a kind of) additive models. FHT provide a much-needed bridge between an important and effective machine learning procedure and traditional statistical modeling ideas. With this bridge, ideas can now flow easily in both directions so that a deeper and more thorough understanding of boosting will eventually emerge.

In this discussion, we would like to share our thoughts and reflections on boosting and related statistical ideas, inspired by FHT.

1 What makes boosting work in classification?

Let us consider the two-class problem in this section. In an elegant and convincing way, FHT bring Freund and Schapire's Discrete AdaBoosting to our home domain by rederiving its population version as a familiar Newton-like step in the optimization of a not-so-familiar exponential loss function serving as a surrogate for the 0-1 loss function. This surrogate is sensible since it is uniquely minimized by (half of) the log odds ratio of the conditional probability of $Y = 1$ given $X = x$. Under this light, it is not hard to find instances of statistical procedures sharing traces of similarities with boosting. For example, in parametric estimation, we indeed have been 'boosting' a consistent estimator to an efficient estimator by taking a Newton step in the optimization of the likelihood function; and indeed have been using estimating equations as sensible surrogates because the likelihood equations are intractable. Despite of these similarities, there are fundamental differences between boosting and these statistical procedures. As recounted by FHT, boosting as a conjecture was proposed as a theoretical learning question in the machine learning community and is now being viewed as a nonparametric stepwise fitting technique in an additive style. Its superb empirical performance in high-dimensional classification problems has very few, if any, rivals. Even though the concept of 'true' model does get used in the evaluation stage of boosting, the starting point of boosting is not a 'true model' as commonly done in statistics. Its starting point is a 'weak learner' and the question posed was how to make it better or 'boost' it. In hindsight, this is a natural and realistic approach in modern data analysis where the size of data sets could not be imagined in the time of our forebears such as Fisher or Neyman. Because of the complexity and scale of these problems, it is impossible to come up with an effective likelihood approach in one step. Often, if not always, a sensible procedure or a 'weak learner' is available, either ad-hoc or based on a simple approximation. Moreover, an evaluation criterion or a loss function is always available. The boosting question now becomes how to improve the weak learner in terms of this loss function. With a starting point and an objective function, a greedy approach for improvement is very natural. Numerical considerations of the greedy approach explain why boosting in terms of the *evaluating* 0-1 loss function might not be a good idea. Any derivative-based greedy algorithm such as Newton's method is not appropriate for this discontinuous loss function whose minimizer is not unique but a whole class. A surrogate, the exponential loss function, is used in AdaBoosting as the *implementing* objective function although the evaluation function remains the 0-1 loss. From a numerical point of view, the exponential function is an excellent function to apply Newton's method, because of its convexity and gradually changing derivatives. Furthermore, this is a relevant loss function to optimize since its minimizer is sensible for the classification problem. A second surrogate, proposed by FHT, is the binomial likelihood that as a function is very close to the exponential loss and has the same minimizer. FHT devise the Gentle Boosting algorithm based on the binomial surrogate and show a similar performance as those based on the exponential loss. A third surrogate, the squared loss function, is also discussed in FHT. The squared function is the best possible for Newton's method: convex and constant second derivatives. FHT report good performance but think that it is dominated by schemes based on either the exponential or the binomial likelihood loss functions and hint that the reason might be that the squared loss function loses its monotonicity when $yF(x) > 1$. It is well known that the unique minimizer of the squared loss is the conditional expectation of Y given $X = x$ which is naturally bounded between -1 and 1 since $|Y| = 1$. If this constraint is taken into account in the greedy search for the optimizer of the squared loss, then one should not wander out to the region where $yF(x) > 1$ but stay in the region where the squared loss is monotone. It is curious to know whether or not taking this constraint into account (obviously numerically more complicated) would bring the performance of squared loss boosting on par with those based on the exponential or binomial likelihood loss.

Although the three surrogates mentioned so far have sensible minimizers, they are only qualitative approximations to the 0-1 loss function. They are actually quite bad quantitative approximations (cf. Figure 2 of FHT). But the population minimizers of the expected 0-1 loss function include those of exponential, binomial likelihood and the squared loss. The first question then simply becomes:

(1) Which surrogate (or implementing loss function for boosting) to use so that the boosting *estimate* best approximates its corresponding population minimizer? This (difficult) question might involve numerical as well as statistical/stochastic efficiency issues.

Now let us step back from these successful (implementing) surrogates to look at the original 0-1 loss function. From the quantitative approximation point of view, an obvious surrogate is a smoothed version of the 0-1 loss function $1 - \Phi(yF(x)/\sigma)$ where Φ is the cdf of $\mathcal{N}(0, 1)$ and σ is a tuning parameter to control the approximation of such a smoothed version with the original 0-1 loss function. This makes the numerical greedy approach possible. If one follows the exact steps (when c is fixed) to this objective function as in the derivation of the population version of Discrete AdaBoost (Result 1 in FHT), one gets a boosting algorithm with a *different reweighting* scheme having weights

$$\varphi(yF(x)/\sigma) = \varphi(F(x)/\sigma),$$

because $y^2 = 1$ and $\varphi(\cdot)$, the standard normal density, is symmetric.

These weights ignore the values of y or avoid reality check and concentrate more and more on the cases where the previous classifier F is unsure about itself. Since this is the opposite of the boosting reweighting philosophy, one might expect that it would not work well. The smoothed 0-1 loss function is not convex (or concave) and its derivatives change rapidly, especially for a small tuning parameter σ or a closer approximation to the 0-1 loss function. For such functions, the quadratic approximation is not accurate around zero and Newton's method easily over-shoots to miss the true optimizer. Hence this recipe should not work from a numerical point of view.

(2) The question is, however, for this smoothed 0-1 loss: will a more suitable numerical optimization method such as trust region (cf. [2]) lead to a sensible boosting procedure having a different weighting philosophy, presumably with higher computational cost?

As argued above, boosting algorithms benefit greatly from a smart choice of a surrogate implementing objective function from both statistical and numerical points of view. FHT (section 4.4) also point out that Gentle Boosting has an edge because of its numerical stability of the derivative calculation of the binomial likelihood function. However, boosting's most powerful advocate is its stunning success on real data sets and its 'mysterious' resistance to overfitting in most cases in classification. The summary in Table 2 of FHT on real data set results suggests that the performance of a particular variant of boosting depends on the interaction among the choice of the weak learner, the underlying problem and the effectiveness of the Newton method as a numerical approximation to the surrogate loss function. Figuring out how these factors interact should bring us a deeper understanding of boosting and might shed light on its resistance to overfitting.

We explore this interaction in the rest of this discussion. In the context of boosting L_2 regression, we compare boosting with another ensemble scheme, bagging [Breiman, 1996], for which we have gained some understanding recently [1]. Different weak learners are looked at, the overfitting issue is touched upon, and a bag-boosting scheme is proposed. In the last section, we return to classification to emphasize the importance of the choice of the weak learner and to make the point that the resistance to overfitting of boosting is probably due to the 0-1 evaluation loss function.

2 Boosting and bagging with L_2 loss for regression

Boosting as explained and analyzed by FHT is a general method: a greedy forward stagewise technique to fit a model of additive style by minimizing a loss function. This view opens the door for boosting in other contexts than classification (although L_2 loss is also an appropriate surrogate as described in the previous section). For L_2 regression, the boosting algorithm works as follows:

(a) Set $F_0 \equiv 0$ (or another more sensible starting value).

(b) For $m = 1, 2, \ldots, M$, fit the function estimator $f_m(\cdot)$ which is parameterized as $f_m(x) = \beta b(x, \gamma)$ (as in FHT):

$$\beta_m, \gamma_m = \text{argmin}_{\beta,\gamma} \sum_{i=1}^{n} (Y_i - F_{m-1}(X_i) - \beta b(X_i; \gamma))^2.$$

Set $F_m(\cdot) = F_{m-1}(\cdot) + f_m(\cdot)$.

(c) Output the function estimator

$$F_M(\cdot) = \sum_{m=1}^{M} f_m(\cdot).$$

This algorithm is indicated by FHT [formula (6)] and was also given by Friedman (1999). It does not involve any 'reweighting'. The weak learner $b(\cdot; \gamma)$ is fitted in the mth step to the current residuals $Y_i - F_{m-1}(X_i)$. There is no need for a surrogate loss function in this case since the evaluating L_2 loss is the best possible for Newton's method and the quadratic approximation is exact. We now discuss the issue of choosing the weak learner. This discussion is continued in section 3 for the case of classification.

Linear weak learners.

It is well-known that boosting's cousin bagging does not change any linear procedures. It can be shown easily that it is the same with boosting in L_2 regression. When the weak learner is linear $b(x, \gamma) = x^T \gamma$ and $f_m(x) = x^T \gamma_m$, the following two statements hold.

(a) For L_2 boosting in regression as described earlier,

$$\begin{aligned} f_m(\cdot) &\equiv 0 \text{ for all } m = 2, 3, \ldots, M, \\ F_M(\cdot) &= \text{ least squares linear regression predictor for all } M \geq 1. \end{aligned}$$

(b) For LogitBoost as presented by FHT, with $p_M(x) = \frac{\exp(F_M(x))}{exp(F_M(x)) + exp(-F_M(x))}$

$p_M(x)$ converges to the corresponding MLE in logistic linear regression as $M \to \infty$,

provided that standard conditions for convergence of the MLE hold.

Stumps as the weak learner

We now investigate the effect of boosting in a concrete simple linear model:

$$Y_i = 2 + 3X_i + \varepsilon_i, \; i = 1, \ldots, n, \tag{1}$$

with ε_i's i.i.d. $\sim \mathcal{N}(0, 1)$, X_i's i.i.d. $\sim$ Uniform([0, 1]) and all X_i's independent from all ε_i's. The target is the true function $F_{true}(x) = 2 + 3x, \; x \in [0, 1]$.

We use stumps (with 1-dimensional covariate space) as weak learners and run the L_2 boosting algorithm for regression from above. For a typical sample, the boosting stumps estimate is displayed in Figure 1: it is still a rather crude and erratic approximation for the target. Figure 1 uses 6 boosting iterations which is optimal on average with respect to MSE as indicated in Figure 2. Based on 50 independent simulations of model (1), we have estimated bias, variance and hence also mean squared error as a function of the number of boosting iterations. The result is displayed in Figure 2.

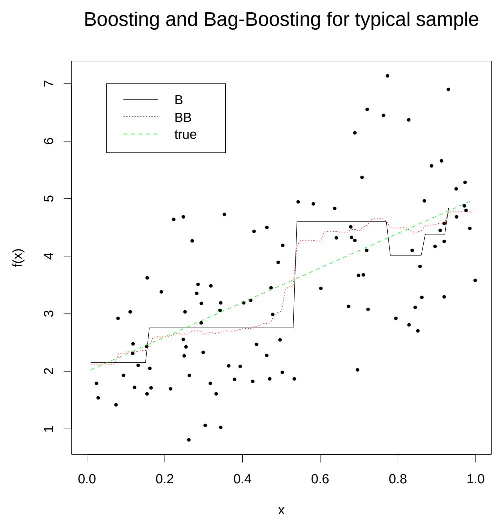

Figure 1: Boosting stumps [B, black] and bag-boosting stumps [BB, red] estimate based on a typical sample [dots] from (1). The target function is indicated in green. Boosting is done with 6 and bag-boosting with 4 iterations.

(3) Interestingly, there is a clear indication for overfitting, starting after 6 iterations already. We see here a simple example in L_2 boosting for regression where overfitting occurs easily in contrast to classification; similar phenomena are reported in Friedman (1999).

From our own work [1] we know that stumps evaluated at x have high variances, for x in a whole region of the covariate space. From an asymptotic point of view, this region is 'centered around' the true optimal split point for a stump and has 'substantial' size of order $O(n^{-1/3})$. That is, stumps do have high variances even in low dimensions as in this simple case (with only three parameters) as long as one is looking at the 'right scale' of order $O(n^{-1/3})$: such a high variance presumably propagates when combining stumps in boosting. This observation is the starting point for another boosting machine to be described next.

The bag-boosting machine with bagged weak learners.

Bagging is known to smooth the weak learner such as stumps and thus achieve a variance reduction. The precise (asymptotic) smoothing function is given in [1] which also characterizes its variance reduction effect in simple yet canonical cases. The high variance of a stump is reduced up to a factor of size 2 to 3, while leaving the bias approximately unchanged. This implies that bagging is also effective for a low-dimensional predictor such as stumps. Hence, we combine bagging with boosting, we call it bag-boosting: for the weak learner in boosting, just replace the stump (or a more general tree) by the bagged stump (or bagged tree). This is a very natural idea and has been thought about by a couple of researchers although we have not seen anything in writing. The resulted fit from bag-boosting is shown in Figures 1 and 2. Note that performance for bagging alone is given by bag-boosting with one boost. Instead of a classical bagging step, we

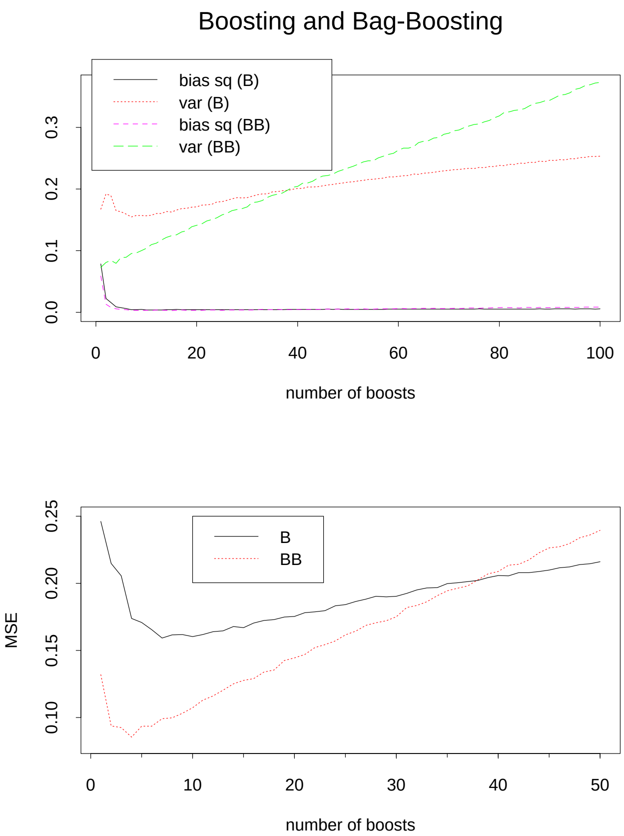

Figure 2: Boosting stumps and bag-boosting stumps for model (1). Top: squared bias and variance of boosting [B, black and red] and bag-boosting [BB, purple and green]. Bottom: mean squared error for boosting [B, black] and bag-boosting [BB,red].

actually use 'subbagging' or bagging on subsamples of size $[n/2]$ (resampling without replacement) which is computationally cheaper while still being as accurate as bagging [1]. The visual improvement of bag-boosting in Figure 1 can be explained as that a bagged stump is a smooth rather than a step function [1]. Its variance and MSE improvements are impressively described by Figure 2, provided that we know roughly where to stop with boosting iterations. Figure 2 illustrates that the more efficient base learner (namely the bagged stump) has a faster increase in variance with the number of boosting iterations: it would be interesting to know whether this is a more general fact.

(4) Thus bag-boosting has a potential to improve upon boosting with stumps or larger decision trees. Its drawback is the higher, although still feasible computational cost.

3 Back to classification

Many issues remain. For example, the convergence issue is not touched by FHT. In particular, for a fixed sample size, there are two issues: the (non-)convergence of the boosting algorithm and whether it is better to stop iterations before having approximately converged, if the algorithm actually would converge. The latter issue is known in fitting complex parametric models such as neural networks where a possible regularization is given by stopping before convergence. There is another convergence issue of the next level when the sample size tends to infinity.

(5) A more important question is how much FHT's population story tells about the data driven algorithms. The population quantities are always approximated: the accuracy depends on the choice of weak learners (see also our remarks about linear weak learners, stumps and bagged stumps in section 2) and the data-generating distributions. Finally, the hope is that the errors do not propagate with the boosting iterations!

Particularly, even with the connection made by FHT, it is hard to know whether one should use a simple learner like stumps or a more complex one such as trees with more terminal nodes.

In the boosting literature which is almost exclusively on classification, decision trees are the most popular weak learners. We do agree that trees are generally attractive, but it is worth pointing out that other (nonlinear) weak learners may exhibit good performance as well. Even when using trees as weak learners, the choice of the number of terminal nodes (e.g. with best-first induction) in the learners does matter. Figure 1 in FHT can be misleading for a superficial reader. It wrongly suggests that the number of terminal nodes in the learner is irrelevant when having sufficient computing power for performing many boosting steps; and that with stumps, Discrete AdaBoost is outperformed by Real AdaBoost consistently. The careful reader will exploit a more subtle sensitivity with respect to choosing the learner in section 6 of FHT (e.g. Figure 4 [top left and top right]) and results on real data sets (Table 2).

FHT distinguish between additive and non-additive decision boundaries. They argue convincingly that in the first case, choosing stumps as weak learners is better than using a larger tree which is expected to over-estimate non-existing interaction terms. If we knew that the decision boundary could be well approximated by an additive function, we would feel more comfortable by fitting an additive model (with backfitting) as in Hastie and Tibshirani (1990), rather than boosting stumps. Or in other words, if the conjecture by FHT [section 11] 'that boundaries involving very high order interactions will rarely be encountered in practice' (whatever 'high' means here) is really relevant, there would not be any good reason to prefer boosting over the established and popular GAM backfitting estimate.

Boosting's resistance to overfitting has been one of its major virtues in comparison with other nonparametric techniques. Overfitting in boosting algorithms in the sense of fitting a large model can happen in two ways. The first is having a large model fitted as the weak learner and the other having to run the algorithm for many iterations. The most overfitted model comes from a large/complex weak learned in combination with a long iteration in boosting. FHT touch on this overfitting issue in its concluding section by listing three plausible explanations. As emphasized in section 1 of this discussion, the use of the (implementing) surrogate exponential loss function is crucial from the numerical point of view and is thus partially responsible for the success of boosting. However, the evaluation in a classification problem has always been the 0-1 loss.

(6) This divorce of the implementation and the evaluation loss function does not exist in L_2 boosting where we saw overfitting easily in section 2. We concur with the last point of FHT in section 11 that the 0-1 loss is very robust against overfitting and we further argue that an evaluation based on the exponential implementing loss does show strong overfitting. This conjecture is supported by the breast cancer data set below. Moreover, we believe that 'parts of the real world' are non-additive. Under this belief and knowing boosting's resistance to overfitting, it can be very valuable to use best-first inducted trees with *more than two* terminal nodes. This is confirmed also by our analysis of the breast cancer data below.

Analysis of breast cancer data.

We partially re-do the analysis of FHT for the breast cancer data and thereby add insights on bag-boosting and overfitting. The table below and Figure 3 give comparative results to the ones in FHT. We use 'subagging' for computational efficiency and it has a performance similar to bagging [1].

Misclassification rates (in percentages) for the breast cancer data
original tree (unpruned; the program from R): 5.643

Subagging	4.812
Boosting stumps	4.029 (with optimal 97 boosts)
Bag-boosting stumps	3.652 (with optimal 47 boosts)
Boosting large tree	2.929 (with optimal 117 boosts)
Bag-boosting large tree	2.768 (with optimal 11 boosts)

The misclassification rates (called test error in FHT) are estimated by averaging over 100 random partitions of the data set into a 90% training and a 10% testing, while FHT use a 5-fold CV. So our results differ slightly from FHT (also because presumably we are using different tree algorithms), but we believe ours have better accuracies. Boosting and its variants are done with Real AdaBoost, and the trees are run with the program

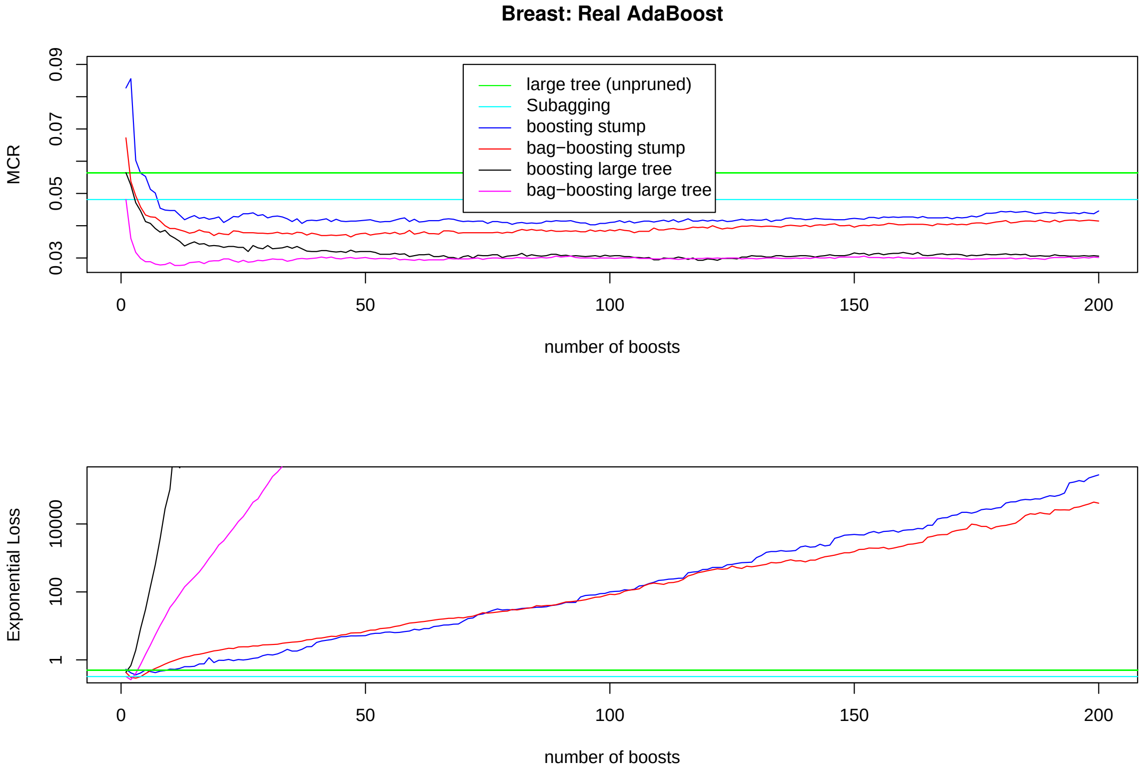

Figure 3: Different weak learners in Real AdaBoost for breast cancer data [section 7 in FHT]. Top: misclassification rates. Bottom: expected exponential losses $\mathbb{E}[\exp(-YF(X))]$ on the logarithmic scale.

from R. Bag-boosting large trees gives the best result followed by boosting large trees, bag-boosting stumps, boosting stumps and then bagging. Bag-boosting only needs few iterations and brings non-trivial improvement over boosting alone, at the expense of a more computationally costly weak learner. Note that Figure 3 (bottom panel) shows strong overfitting with respect to $\mathbb{E}[\exp(-YF(X))]$!

(7) For this particular data set and with respect to misclassification rate, there is hardly any overfitting. It suggests that there is no loss in using large trees, if there is never any substantial overfitting. Boosting stumps on the other hand restricts to additive models. With respect to the misclassification rate, shall we always boost with large trees as weak learners?

Acknowledgments. Bin Yu would like to thank Mark Hansen for many stimulating discussions and helpful comments on the draft, and Margaret Wright for helpful discussions on optimization. Special thanks are due to Josef Vogt for the results on the breast cancer data.

References

[1] Bühlmann, P. and Yu, B. (2000). Explaining Bagging. Preprint.

[2] Gill, E.P., Murray, W. and Wright, M.H. (1981). *Practical optimization.* Academic Press, New York.