Discussion

Peter Bühlmann and Bin Yu

October 15, 2007

We would like to thank the authors for their provocative view on boosting. Their view is built upon some "contrary" evidence based on a particular simulation model. In our discussion, we argue that the structure of the simulation model explains many aspects of the "contrary" evidence. We touch upon the issue of shrinkage or small step-sizes, and we conclude that the "statistical view" provides constructive insights for applying boosting in a highly successful way.

The gradient and "statistical" point of view

The gradient point of view of AdaBoost is, in our opinion, a great leap forward for understanding AdaBoost and deriving new variants of boosting now meaning much more than just AdaBoost. This view, which seems to be called the "statistical view" by Mease and Wyner (MW), has been pioneered by Breiman (1998, 1999), Friedman et al. (2000), Mason et al. (2000), Rätsch et al. (2001) and is not just a product of the statistics community. The gradient view of boosting allows transferring of the boosting methodology to many other contexts than just classification, see for example Meir and Rätsch (2003) or Bühlmann and Hothorn (2007) for an overview. We should also emphasize that the gradient view has never promised to explain everything about AdaBoost. Hence we are puzzled by the negative picture of this view painted in the paper under discussion: it differs greatly for most part from our experience and understanding of the statistical research on boosting. In particular, the MW paper seems to ignore simulation, real data and theoretical evidence about overfitting and early stopping (cf. Bartlett and Traskin (2007) regarding asymptotic theory for AdaBoost). We will discuss these issues in more details below.

The relevance of MW's counter-examples

The evidences in MW are simulated "counter-examples". It is questionable that they are representative of situations encountered in practice. More importantly, with one exception, evidence of differences shown contradicting the so-called "statistical view" are 1 or 2 % in error rate. One wonders how important or meaningful these differences are in practice, even though they might be statistically significant. In any real world situation, the model used is for sure wrong

and the approximation error of the model to the real situation could easily swallow these small differences in performance.

Furthermore, all the evaluation metrics in the MW paper are on statistical performance without any consideration of the computation involved or the meaning of the model derived. For large data problems, computation is an indispensable player and needs to be in the picture.

Additive decision boundary but non-additive logit-probabilities

MW's model (in Section 3) is additive for the decision boundary. In terms of conditional probabilities $p(x) = \mathbb{P}[Y = 1|X = x]$ on the logit-scale, $\text{logit}(p(x))$ is *not* an additive function in the (feature) components of x.

Since the population minimizer of (gradient) AdaBoost or also of LogitBoost equals

$$F_{\text{pop}}(x) = 0.5 \cdot \text{logit}(p(x)) = 0.5 \cdot \log\left(\frac{p(x)}{1 - p(x)}\right),$$

a (boosting) estimate will be good if it involves an effective parameterization. We believe that this is a central insight, which has been pioneered by Breiman (1998, 1999), Friedman et al. (2000) and which has been further developed by more recent asymptotic results on boosting. In the MW model, $F_{\text{pop}}(x)$ is non-additive in x while boosting with stumps yields an estimate $\hat{f}(x)$ which is additive in x. We think that this is the main reason why some of the figures in MW lead to "contrary" evidence: with our model, as illustrated below, the comparison of stumps versus larger trees for weak learners is always in favor of stumps, i.e. stumps yield better performance and larger trees are more heavily overfitting which is the opposite finding to Figures 1, 2, and 11 in MW. MW's model in Section 4 involves only a single component of x and hence it is additive also on the logit-scale for the probabilities. But our own model described below does not confirm MW's statement that their findings "do not depend on a particular simulation model".

Other issues in MW concerning "contrary" evidence cannot be easily explained by the nature of the model.

Figure 3 intends to show that LogitBoost is worse than AdaBoost. The MW finding might seem relevant at 1000 iterations. But one doesn't need to go that far for both methods by early stopping. 100 or so iterations seems enough for stumps and 400 for 8-node trees. The performance difference is then less than 1%. Thus, having some computation savings in mind, early stopped LogitBoost is preferable.

Figure 4 tries to make the point that early stopping could hurt to lose about 1% performance when the total Bayes error is 20% and there is no structure to be learned. However, the 1000 iteration model undoubt-fully gives the wrong impression that something is there, while the early stopped model gives the correct impression that not much is to be learned. Hence we think early stopping is not hurting here. In addition, the starting value of boosting matters but this issue is ignored in standard AdaBoost. A (gradient) boosting algorithm should be started with $F_{init} = F_0 \equiv 0.5 \log(\hat{p}/(1 - \hat{p}))$ where $\hat{p}$ is the empirical frequency of $Y = 1$, cf.

Bühlmann and Hothorn (2007). That is, boosting would (try to) improve upon the MLE from the "pure noise" model. Then, it is expected - and we checked this using gradient LogitBoost on the unbalanced example corresponding to Figure 4 in MW - that boosting will overfit from the beginning because the underlying structure is pure noise. The same idea could be applied to AdaBoost as well: in contrast, standard AdaBoost and MW start with the naive value $F_{init} = F_0 \equiv 0$.

Shrinkage and small step-sizes: another dimension for regularization

MW makes some claims about additional shrinking using small step-sizes. As we understand Friedman (2001), he *never* intended to say that a shrinkage factor would avoid overfitting. Instead, he argued that introducing a shrinkage factor may improve the performance. Later, Efron et al. (2004) made the connection, in the setting of linear models, that boosting with an infinitesimally small shrinkage step is equivalent to the Lasso under some conditions, and for general situations, Zhao and Yu (2007) showed that appropriate backward steps need to be added to boosting to get Lasso. This intriguing connection shows again that the shrinkage factor cannot eliminate overfitting. All what it achieves is a different, usually more powerful solution path (with a new regularization dimension through the step-size) than without shrinkage.

Our own findings with an additive model for logit-probabilities

Now we devise our own simulation model to clarify some issues regarding overfitting, choice of weak learner and the estimation of probabilities via boosting. Arguably, as emphasized above, examples should not be over-interpreted. However, in view of many reported findings similar to what we show here, we feel that our examples are rather "representative" and we are reporting major instead of slight differences.

Our model is in the spirit of MW but on the logit-scale:

$$\begin{aligned} \text{logit}(p(x)) &= 4\sum_{j=1}^{5}(x_j - 0.5) \\ Y &\sim \text{Bernoulli}(p(x)), \end{aligned}$$

where $p(x) = \mathbb{P}[Y|X = x]$. This model has Bayes error rate approximately equal to 0.1 (as in the MW paper). We use this model as it is additive on the *logit-scale* for the probabilities since the population minimizer of (gradient) AdaBoost and (gradient) LogitBoost is $0.5\text{logit}(p(x))$. We use $n = 100$, $d = 20$ (i.e. 15 ineffective features), x as in MW and we show the results for one representative example with test set of size 2000. We skipped the repetition step over many realizations from the model: again, we think that one realization is representative and it mimics somewhat better the situation of analyzing one real data set.

We consider the misclassification test error, the surrogate loss test error (e.g. the test set

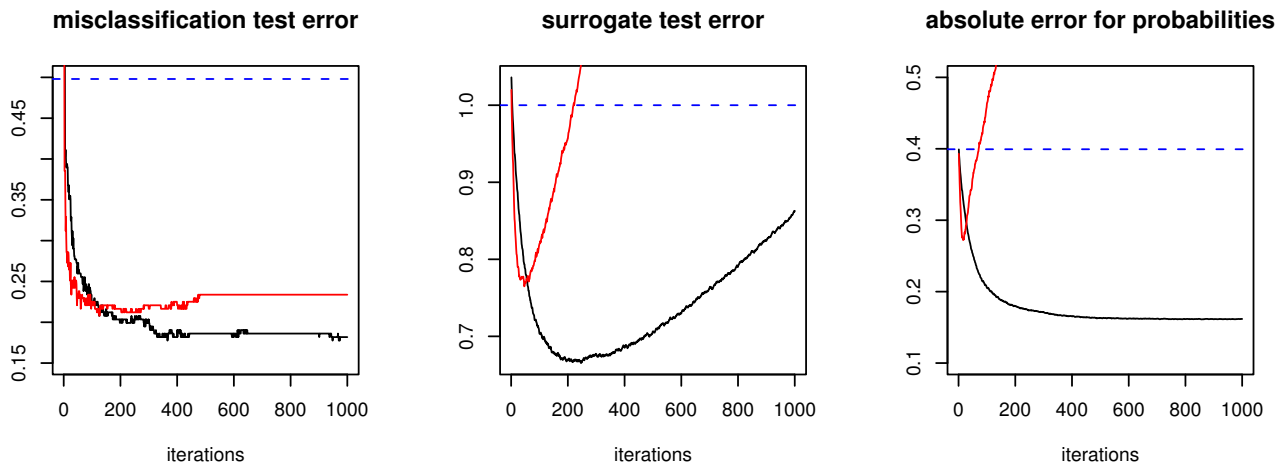

Figure 1: Gradient boosting with exponential loss (gradient AdaBoost). Left panel: Test set misclassification error; Middle panel: test set surrogate loss; Right panel: test set absolute error for probabilities. Black: stumps; Red: larger tree; Blue dashed line: naive estimator.

average of $\exp(-y\hat{f})$ for AdaBoost) and the absolute error for probabilities

$$\frac{1}{2000}\sum_{i=1}^{2000}|\hat{p}(X_i) - p(X_i)|,$$

where averaging is over the test set.

All our computations have been done with MW's code for AdaBoost and the `R`-package `mboost` from Bühlmann and Hothorn (2007): we used stumps and larger trees as weak learners. By the way, MW's code is *not* implementing 8 node trees but trees which have on average about 6-8 terminal nodes (during the boosting iterations for this model). The results are displayed in Figures 1- 3. A comparison is also made to the naive estimator with $\hat{p}(x) \equiv 0.5$.

From this very limited experiment we find all facts that we view as important and typical for boosting:

1. Overfitting can be a severe issue when considering the test surrogate loss or for estimating conditional probabilities. In fact, overfitting is seen clearly for all three methods, that is gradient AdaBoost, LogitBoost and AdaBoost. In addition, the misclassification loss is much more insensitive with respect to overfitting. This has been pointed out very clearly in Bühlmann and Yu (2000)) and in the rejoinder of Friedman et al. (2000).

2. Estimating conditional probabilities is quite reasonable when stopping early: as in point 1 above, we see very clearly that early stopping is absolutely crucial for all three meth-

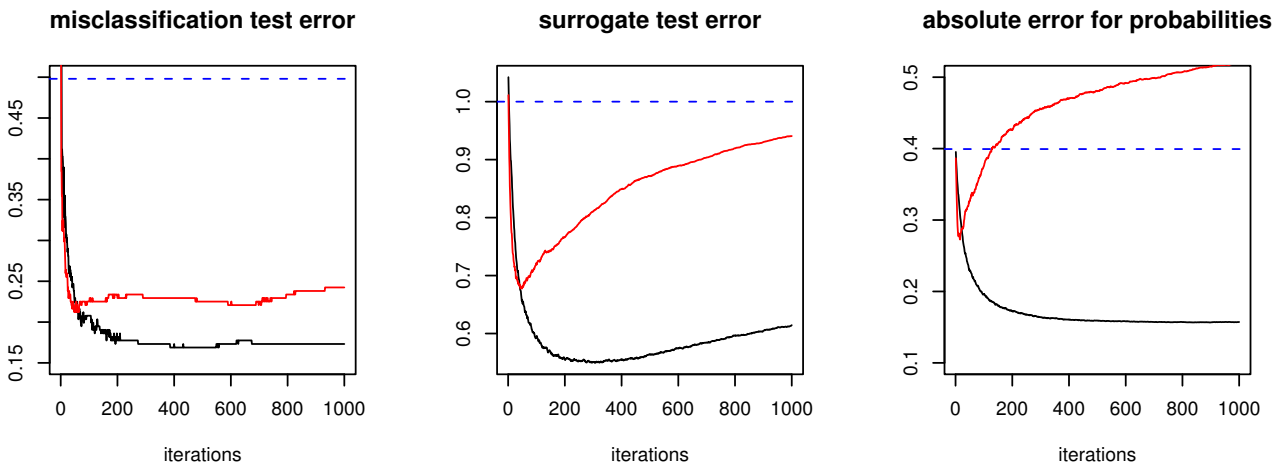

Figure 2: Gradient boosting with Binomial log-likelihood (gradient LogitBoost). Left panel: Test set misclassification error; Middle panel: test set surrogate loss; Right panel: test set absolute error for probabilities. Black: stumps; Red: larger tree; Blue dashed line: naive estimator.

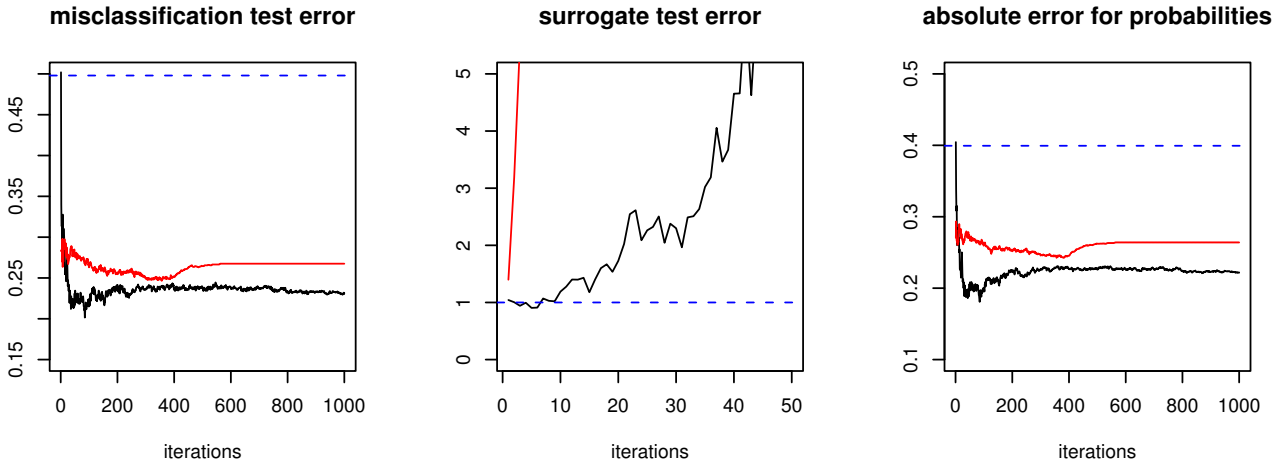

Figure 3: AdaBoost (as in MW). Left panel: Test set misclassification error; Middle panel: test set surrogate loss; Right panel: test set absolute error for probabilities. Black: stumps; Red: larger tree; Blue dashed line: naive estimator. More details are described in point 4 of our summary of findings.

ods. And LogitBoost with early stopping gives the best misclassification error and best probability estimate among the three.

3. Regarding the weak learner, larger trees are worse than stumps for our model where the conditional probability function is additive on the logit scale. The "statistical view" reveals the model behind AdaBoost and LogitBoost: we have to consider the logit-scale (the MW model is *not* additive in terms of the logit of conditional probabilities; note that for the decision boundary the scale doesn't matter while it does play a role for conditional probabilities).

   Larger trees do overfit more heavily for probability estimation or with respect to surrogate test loss. For non-additive models (for probabilities on the logit-scale), the overfitting will kick in later for large trees as the the underlying model requires a more complex fit to balance approximation ("bias") and stochastic error ("variance").

4. Somewhat more in line with the MW paper, the original AdaBoost has less a tendency to overfit than the gradient boosting version. The reason why AdaBoost with the larger tree in Figure 3 is staying constant after a while is due to the fact that the algorithm gets "stuck": it alternates back and forth and hence, the amount of overfitting is limited. At this stage of alternating behavior the estimated conditional class probabilities are very much concentrated around either zero or one (not shown but similar to Fig. 18 in MW), i.e. overfitting has kicked in severely. We are not convinced that this "getting stuck" property of the algorithm is desirable, despite the consequence that a bound on overfitting is then obviously in action. The surrogate loss function in AdaBoost explodes much earlier (w.r.t. boosting iterations) and one needs to implement an upper bound in the program in order to avoid NA values (MW's code needs this little some small modification here!).

Our general understanding about Boosting and it's success

Instead of going through all issues in MW, we choose instead to repeat several general understandings about boosting which were incorrectly questioned by the paper under discussion:

A. Overfitting does matter, and it is a function of the both the "bias" and "variance". Large trees do not overfit heavily in terms of classification error because:
   (i) the misclassification loss is very insensitive to overfitting (see Bühlmann and Yu (2000) and the rejoinder of Friedman et al. (2000));
   (ii) larger trees are not as "complex" as the number of nodes in them indicates since they are fitted in a greedy fashion (e.g. 8-node trees fitted by boosting are not 4 times as complex as stumps with two nodes).

   Most probably, the difference between plain vanilla AdaBoost and a gradient version of AdaBoost (as in MW) will *not* play a crucial role in terms of overfitting behavior; but

gradient-based boosting seems somewhat more exposed to overfitting while AdaBoost can get stuck which naturally limits the amount of overfitting (on a single data-set).

B. Early stopping, particularly for probability estimation, is very important (because of overfitting) and brings computational savings. The supporting theory is given in, for example, Zhang and Yu (2005), Bühlmann (2006), Bartlett and Traskin (2007) and Bissantz et al. (2007).

C. Estimating probability via boosting is often quite reasonable. It is essential though to tune the boosting algorithm appropriately: a good choice is to do early stopping with respect to the log-likelihood test score (see next point regarding surrogate and evaluating loss).

D. It is important to distinguish between surrogate loss (implementing loss) and loss (evaluating loss) function. For example, there is no surprise that it can happen with AdaBoost that the training misclassification error is zero while the test set misclassification still decreases: the explanation is that the surrogate loss (exponential loss) can still be far away from

The usage of boosting as we have advocated in our works, and this is very much in line with Friedman et al. (2000) and their subsequent works, has proven to be very competitive and successful in applications. Gao et al. (2006) describe a successful application of boosting to a language transliteration problem. Lutz (2006) has won the performance prediction challenge of the world congress in computational intelligence in 2006 (WCCI 2006): he was using early-stopped LogitBoost with stumps. Part of his success is probably due to careful choice of choosing the stopping iteration: according to personal communication (he has been a former PhD student of the first author of this discussion), he stopped before reaching the minimal value of a cross-validation scheme. In summary, he did not take any of the findings from MW into account (he didn't know the paper at that time, of course). Maybe his success is more convincing evidence that LogitBoost with (i) its "natural" loss function for a binary classification problem, and using (ii) early stopping, (iii) simple weak learners and (iv) a small step size (i.e. shrinkage factor) often works surprisingly well. Other references about successful applications of gradient-based boosting can be found in Bühlmann and Hothorn (2007) which includes the `R` package `mboost` (standing for model-based boosting) for numerous application areas ranging from classification, regression, generalized regression to survival analysis.

References

[1] Bartlett, P.L. and Traskin, M. (2007). Adaboost is consistent. In Advances in Neural Information Processing Systems **19** (eds. B. Schölkopf, J. Platt, and T. Hoffman), pp. 105-112. MIT Press.

[2] Bissantz, N., Hohage, T., Munk, A. and Ruymgaart, F. (2007). Convergence rates of general regularization methods forstatistical inverse problems and applications. To appear in SIAM Journal of Numerical Analysis.

[3] Breiman, L. (1998). Arcing classifiers. Annals of Statistics **26**, 801–824.

[4] Breiman, L. (1999). Prediction games & arcing algorithms. Neural Computation **11**, 1493–1517.

[5] Bühlmann, P. (2006). Boosting for high-dimensional linear models. Annals of Statistics **34**, 559–583.

[6] Bühlmann, P. and Hothorn, T. (2007). Boosting algorithms: regularization, prediction and model fitting (with discussion). To appear in Statistical Science.

[7] Bühlmann, P. and Yu, B. (2000). Invited Discussion on "Additive logistic regression: a statistical view of boosting (Friedman, Hastie and Tibshirani)". Annals of Statistics **28**, 377–386.

[8] Efron, B., Hastie, T., Johnstone, I. and Tibshirani, R. (2004). Least angle regression (with discussion). Annals of Statistics **32**, 407–451.

[9] Friedman, J.H. (2001). Greedy function approximation: a gradient boosting machine. Annals of Statistics **29**, 1189–1232.

[10] Friedman, J.H., Hastie, T. and Tibshirani, R. (2000). Additive logistic regression: a statistical view of boosting (with discussion). Annals of Statistics **28**, 337–407.

[11] Gao, J., Suzuki, H. and Yu, B. (2006). Approximation Lasso methods for language modeling. Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the ACL, pp. 225–232, Sydney.

[12] Lutz, R.W. (2006). LogitBoost with trees applied to the WCCI 2006 performance prediction challenge datasets. Proceedings of the International Joint Conference on Neural Networks (IJCNN 2006)).

[13] Meir, R. and Rätsch, G. (2003). An introduction to boosting and leveraging. In Advanced Lectures on Machine Learning (eds. S. Mendelson and A. Smola). Lecture Notes in Computer Science, Springer.

[14] Rätsch, G., Onoda, T. and Müller, K.R. (2001). Soft margins for AdaBoost. Machine Learning **42**, 287–320.

[15] Zhang, T. and Yu, B. (2005). Boosting with early stopping: convergence and consistency. Annals of Statistics **33**, 1538–1579.

[16] Zhao, P. and Yu, B. (2007). Stagewise Lasso. To appear in Journal of Machine Learning Research.