Journal of Machine Learning Research 9 (2008) 157-164 Published 2/08

Response to Mease and Wyner, Evidence Contrary to the Statistical View of Boosting, *JMLR 9*:131–156, 2008

Kristin P. Bennett BENNEK@RPI.EDU
Department of Mathematical Sciences
Rensselear Polytechnic Institute
Troy, NY 12180, USA

Editor: Yoav Freund

1. Introduction

Mease and Wyner (MW) argue experimentally that the statistical view of boosting does not account for its success and that following the now conventional wisdom arising from this view in Friedman et al. (2000) does not necessarily lead to choices in the boosting algorithm that improve generalization. The authors did an excellent job of defining a set of experiments in which small changes in the boosting algorithm (such as changing the hypothesis space, loss function, and shrinkage) produce significant changes in generalization that were unintuitive given the statistical view of AdaBoost (Freund and Schapire, 1996) expressed in Friedman et al. (2000).

The authors state "The statistical view focuses only on one aspect of the algorithm - the optimization." But one can argue just the opposite, that some of the problems and surprises come from not enough of the optimization perspective instead of too much. Analyzing AdaBoost's performance as an optimization algorithm in terms of convergence rates and optimality conditions (measured on the training data) can be quite revealing. First, we observe that the experiments in MW make dramatic changes in the convergence rates of AdaBoost and that these convergence rates are closely associated with the margin of the classifier. AdaBoost may avoid overfitting for two completely different reasons. Sometimes the algorithm is converging so slowly that stopping at a large number of iterations is still early stopping. At other times, AdaBoost converges relatively quickly and is in essence "overtrained" way past reasonable measures of the optimality conditions. In this case the classifier has converged and is no longer changing much, so the classifier does not overfit. Indeed, some overtraining appears to help improve the classifier slightly. Second, we observe that AdaBoost cannot be trained forever. For the separable case, overtraining AdaBoost and LogitBoost will eventually produce numeric problems that can produce artifacts in the generalization error. In Experiments 3.3 and 4.3 in MW, LogitBoost was overtrained to the point of failure. The so called overfitting observed for LogitBoost was really an algorithmic issue that is quite fixable. If LogitBoost is stopped appropriately or another stepsize strategy is used, the results for LogitBoost are as good as or better than those for AdaBoost. More discussion of these results can be found below.

2. A Mathematical Programmer's View of AdaBoost

AdaBoost optimizes a linear combination of weak hypotheses with respect to the exponential loss. AdaBoost is a coordinate descent (CD) algorithm, that iteratively optimizes the problem with respect to one hypothesis at a time using column generation (Bennett et al., 2000). The weak learner seeks the hypothesis that maximizes the inner product with the function gradient (Mason et al., 2000). The convergence properties of such coordinate descent algorithms have been extensively studied in the mathematical programming community and a full analysis of relevant CD results and their extension to the boosting case can be found in Rätsch (2001).

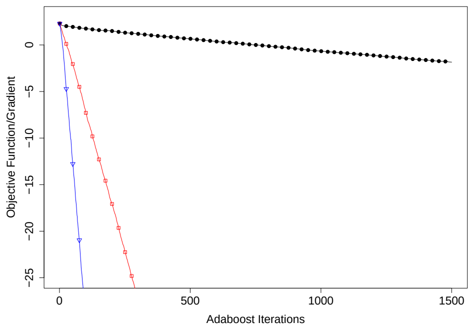
Thus from the mathematical programming perspective, we know AdaBoost inherits both the beneficial and potentially problematic properties of CD. We know from both the CD and original AdaBoost theoretical results that the AdaBoost *objective* converges linearly to the optimal objective. The simplicity of CD and its suitability for column generation make coordinate descent an attractive algorithm, but in practice coordinate descent is not widely used because it can be very slow and it has a tendency to cycle. CD guarantees that the objective function converges to the minimum but there is no guarantee that optimal hypothesis coefficients are attained, and cycling is possible. The AdaBoost loss function is particularly problematic since the exponential function is not strongly convex and the Hessian is rank deficient when the size of the hypothesis space exceeds the number of points. Overall, mathematical programming tells us that we can expect the AdaBoost objective value to converge linearly and the convergence rate to be slow, especially when cycling occurs. The paper on the dynamics of AdaBoost (Rudin et al., 2004) investigates this cycling behavior.

The MW experiments focus on the degenerate case in which the optimal objective value of the underlying exponential optimization problem is zero. LogitBoost and AdaBoost are functions of the form $min_\alpha J(f)\ s.t.\ f = Ha$ where H is the hypothesis space matrix containing all possible weak learners for that data set. In every case, there exists some linear combination of weak learners that classifies the points with no error, and therefore the objective can be driven to zero. The AdaBoost exponential loss function is $\sum \exp(-y_i f_i)$. The function space gradient is $\frac{\partial J(f)}{\partial f_i} = -y_i exp(-y_i f_i)$. Note the 1-norm of the function space gradient is the same as the objective, $\left\|\frac{\partial J(f)}{\partial f}\right\|_1 = \sum \exp(-y_i f(x_i))$ for two-class classification. The optimality condition is that the gradient with respect to α is zero, $\frac{\partial J(Ha)}{\partial \alpha} = H' \frac{\partial J(f)}{\partial f} = 0$. In theory, to check this gradient we need to know the weak learners for the full hypothesis space, H. But, for cases where the misclassification error is driven to 0, it is sufficient to monitor the gradient in function space. Fortunately, the norm of the function space gradient provides an upper bound on the norm of the true gradient since $\left\|H' \frac{\partial J(f)}{\partial f}\right\| \leq C \left\|\frac{\partial J(f)}{\partial f}\right\|$ for some fixed $C > 0$.

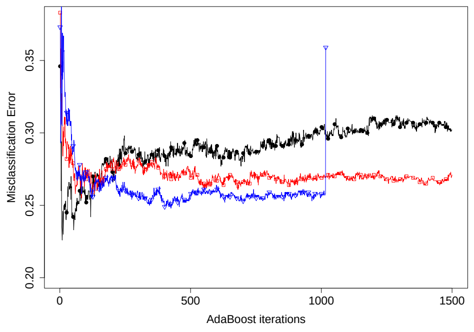
From a mathematical programming perspective, we are optimizing a degenerate, poorly-scaled problem for which the optimal objective value of 0 can only be achieved in the limit using a slower algorithm prone to cycling that may become numerically unstable. Clearly, convergence of the algorithm should be monitored closely. Yet, in most machine learning boosting papers, the focus is on generalization for a fixed number of iterations and rarely on optimization performance.

3. Convergence Rate of AdaBoost

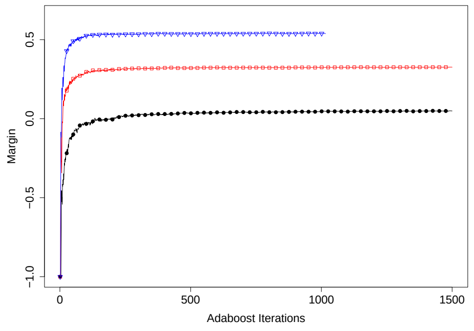
Let's examine the convergence rate and optimality conditions of AdaBoost in the MW experiments. Figure 1 contains three plots, one each for the log base 10 of the objective (or equivalently the 1-



(a) Objective/Gradient (log10)



(b) Test Error

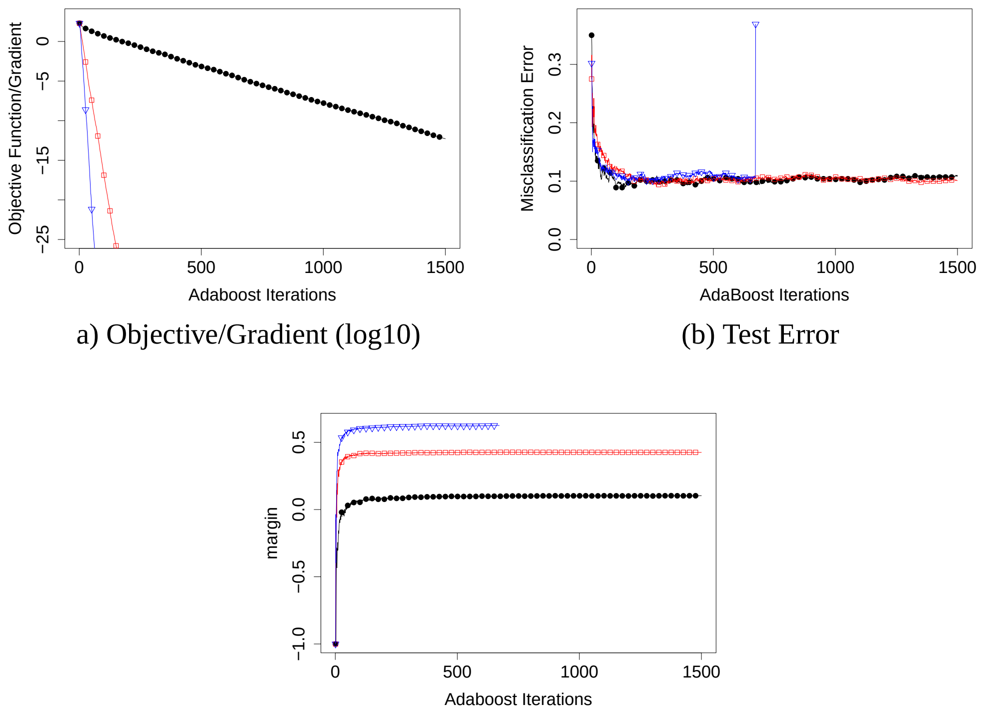


(c) Margin

Figure 1: 1 trial of Experiment 3.1 (10% Bayes Error) for AdaBoost + stumps (black, dots), AdaBoost + 8-node trees (red, squares), and AdaBoost + 16-node trees (blue, triangles).

norm of the gradient), the testing error, and the training margin ($\frac{\min_i(y_i f_i)}{\sum_m \alpha_m}$) for 1500 iterations for the first trial of the experiment with 10% Bayes error in section 3.1 of MW. The graphs contain results for AdaBoost with stumps (black, dots), AdaBoost with 8-node trees (red, squares) and AdaBoost with 16-node trees (blue, triangles). Observe that the loss function and gradient are driven to zero for all three hypothesis spaces and that the remarkably different convergence rates are inversely proportional to the size of the trees being boosted. The results for AdaBoost with 16-node trees end at 1017 iterations because the objective becomes less than 10^{-322}, so a divide-by-zero error occurs. In general, AdaBoost with bigger trees achieves bigger margins and obtains better generalization. AdaBoost with stumps converges incredibly slowly and arguably should be run for more than 1500 iterations if early stopping is not desired.

Figure 2 contains the same three graphs for the first trial for the experiment with no Bayes Error in section 3.2. The objective/gradient and margin graphs are qualitatively similar for experiments 3.1 and 3.2. Note that the 16 node tree Adaboost algorithm underflows at 673 iterations. The testing error graph for experiment 3.2 is quite different. The performance for AdaBoost with stumps is much improved and now competitive or better than AdaBoost with 8 or 16 node trees. Here the margin results do not predict which type of boosted trees will generalize best. MW found at least one simple example in which margins don't work well.



a) Objective/Gradient (log10) (b) Test Error

(c) Margin

Figure 2: 1 trial of Experiment 3.2 (0 Bayes Error) for AdaBoost + stumps (black, dots), AdaBoost + 8-node trees (red, squares), and AdaBoost + 16-node trees (blue, triangles).

Figure 3 shows the results for AdaBoost with 8-node trees and AdaBoost with 8-node trees restricted to 15 nodes for the first trial of experiment 3.6. Here we see that restricting the trees slows convergence, decreases margins, and increases error.

Figure 4 shows the results for AdaBoost with 8-node trees with no shrinkage (red, squares), .1 shrinkage (purple, dots), and .5 shrinkage (blue, triangles) for the first trial of experiment 3.3 (10 % Bayes error). Here we see that shrinkage can speed up or slow down the convergence rates. For .1 shrinkage compared to no shrinkage, the convergence rate was slower, the margin smaller, and the test error larger. For .5 shrinkage, the convergence rate was faster and the margin was larger than for the .1 shrinkage case. If the .5 shrinkage algorithm is terminated at using reasonable stopping criteria, the performance is quite comparable with the no shrinkage case, and improved over the .1 shrinkage case.

We present the following conjectures based on observations of this and other MW experiments for the separable case and leave fuller investigation to later work.

- *The convergence rate of AdaBoost is dependent on the space spanned by the weak learner and larger hypothesis spaces converge faster.* The weak learners produced by stumps are a subset of those from the 8-node decision tree which are in turn a subset of those produced by the 16-node decision tree. The bigger the decision tree, the better the weak learner can match

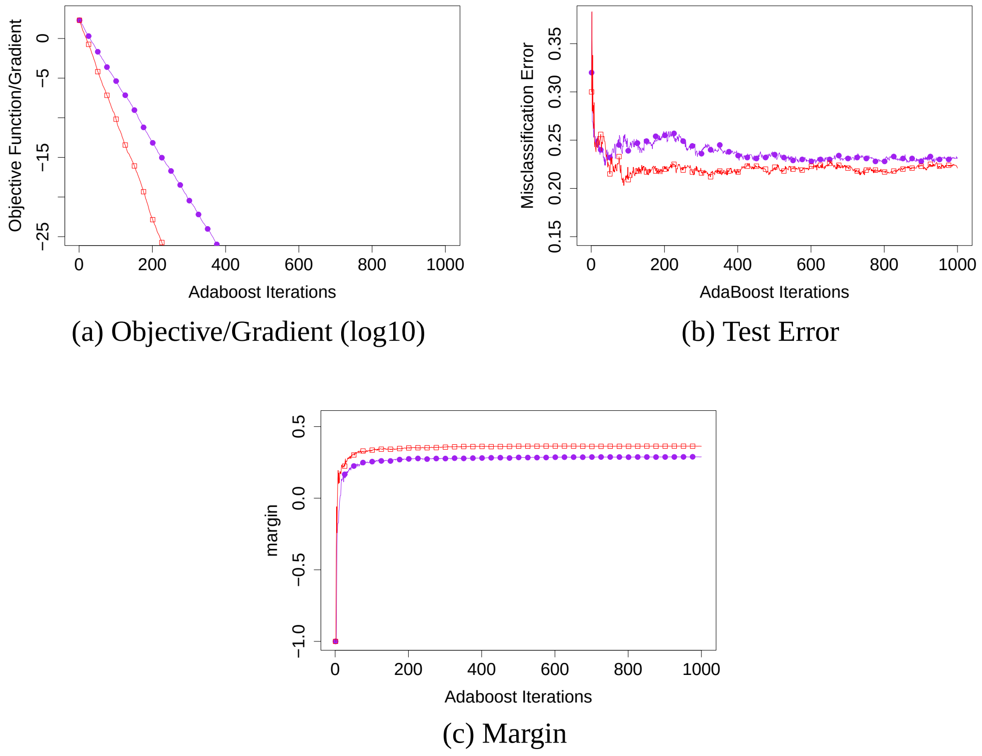


Figure 3: 1 trial of Experiment 3.6 (10% Bayes Error) for AdaBoost + 8-node trees (red, squares), and AdaBoost + 8-node trees restricted to at least 15 observations in terminal nodes (purple, dots).

the gradient at each iteration (as reflected by weighted misclassification error). So AdaBoost can obtain a better decrease in the objective value. This conjecture is also supported by the fact that in Figure 3's run of experiment 3.6, decreasing the hypothesis space by restricting the terminal node size, also reduced the convergence rate.

- *For a fixed separable problem, faster convergence rates of AdaBoost can result in larger margins.* AdaBoost is known to approximately optimize the margin as measured by the 1-norm (Rosset et al., 2004; Schapire et al., 1998). The objective decreases the numerator of the margin and the iterations increase the denominator, so getting a smaller objective quicker creates a better margin. Bigger hypothesis spaces allow bigger steps resulting in larger margins. This finding is also supported by the fact that when shrinkage is used to change the convergence rate, the resulting margins changed as well (see Figure 4). For problems with no training error, we expect larger margins to translate to better generalization rates. But MW's experiments 3.2 and 4.2 show that this is not always the case. Figure 2 shows the margin for 1 run of Experiment 3.2. So MW are quite right in their conclusion that there is more to the

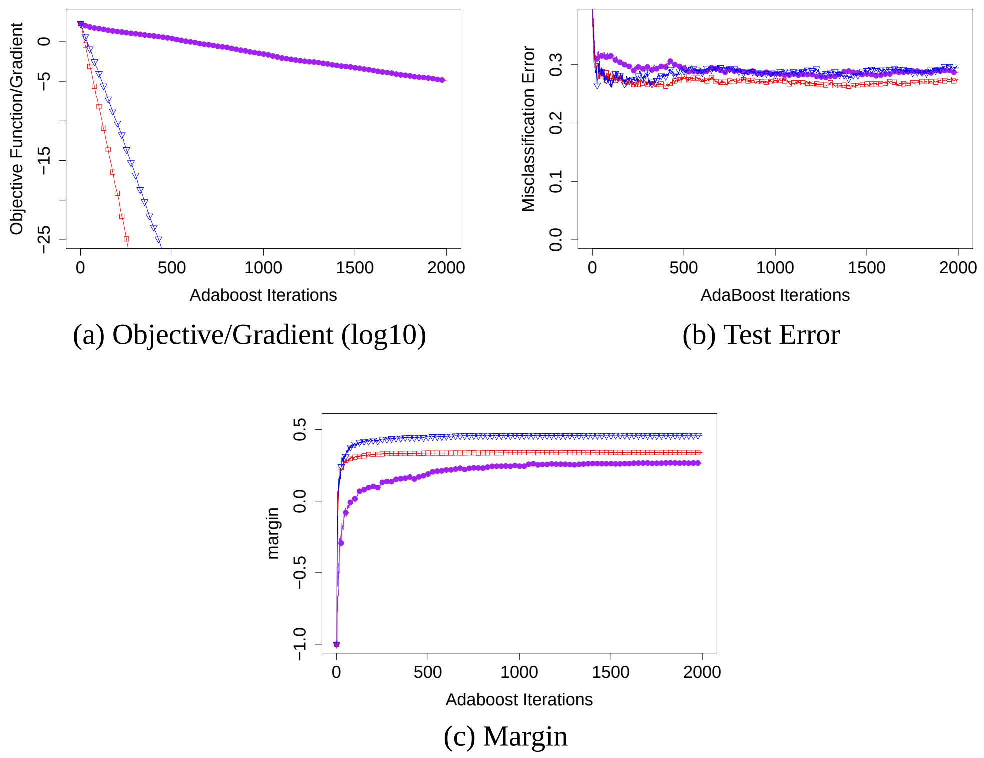


Figure 4: 1 trial of Experiment 3.7 (10% Bayes Error) for AdaBoost + 8-node trees (red, squares), AdaBoost + 8-node trees with .5 shrinkage (blue, triangles) and AdaBoost + 8-node trees with .1 shrinkage (purple, dots).

generalization of AdaBoost then just optimizing the loss. Adding consideration of the margin is not enough either.

- *For slowly converging problems, AdaBoost will frequently be regularized by early stopping.* In experiments 3.1 and 3.2, AdaBoost with stumps is overfitting and the early stopping in MW at 1000 iterations helps the generalization error. For this specific experiment, the slow convergence is a result of cycling. For the first trial in experiments 3.1 and 3.2, AdaBoost with stumps only generated 158 and 156 distinct weak learners in 1000 iterations respectively. The weak learners generated by AdaBoost with 8-node trees and 16-node trees were distinct except for 2. By cycling through relatively few weak learners, AdaBoost with stumps strongly weights a few trees. This appears to be bad for generalization in experiment 3.1 and good for generalization for the no noise case in experiment 3.2.

- *For more rapidly converging problems, AdaBoost will converge and enter an overtraining phase.* For the larger tree cases, the objective and margins converge rapidly. Typically one would halt an optimization algorithm when the gradient became near 0. In the MW experiments, AdaBoost with 8-node trees is overtrained past the point where one would normally halt an optimization algorithm based on gradient criteria (Gill et al.). AdaBoost doesn't overfit in this overtraining phase because it has converged and only very small changes are being made. Perhaps the overtraining phase contributes to the robustness of AdaBoost, since AdaBoost is performing the self-averaging discussed in MW and acting more like bagging. In the MW experiments, AdaBoost achieves better generalization when trained to an extraordinarily high degree of accuracy, a fact contrary to the usual loose convergence criteria used in support vector machines (Bennett and Parrado-Hernández, 2006). But care must be taken to halt the boosting algorithm before the overtraining produces numeric problems due to finite precision problems. As shown in Figure 1, AdaBoost with 16-node trees underflows at 1017 iterations for the 10% Bayes error case and at 673 iterations for the 0 error case. AdaBoost with 8-node trees also underflows eventually as well.

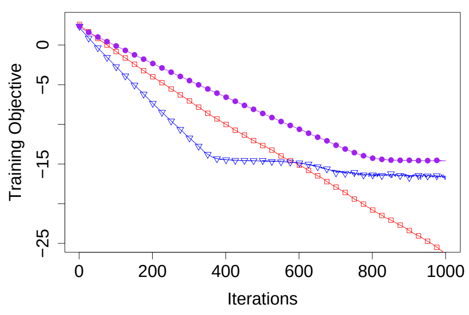
4. LogitBoost versus AdaBoost

Experiments 3.4 and 4.4 compare LogitBoost and AdaBoost and conclude LogitBoost overfits. Tracking the convergence of LogitBoost shows that this is not quite the case. We show our results repeating experiment 4.4 exactly as in the paper for AdaBoost and LogitBoost. Recall LogitBoost differs from AdaBoost in two ways. First, it uses the logistic loss instead of the exponential loss and second, it uses a Newton step instead of an exact step size. The Newton step for logistic loss works out to be 1/2 at each iteration. AdaBoost's stepsize is adaptive. The CD convergence results do not apply directly to LogitBoost as implemented in the paper because of the Newton step.

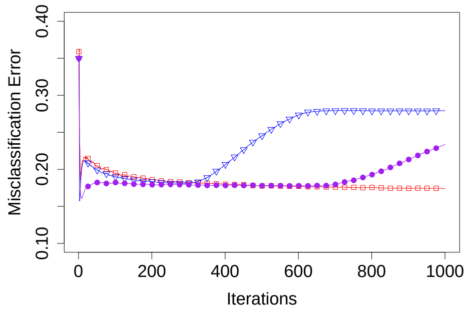
Figure 4 shows the average objective and misclassification results for 100 trials with 8-node trees. Note that at about 375 iterations, LogitBoost fails to obtain a decrease in the objective because the Newton step is too large when the objective is very small. From that point, the testing error declines. LogitBoost with shrinkage converges more slowly, so it can go more iterations before the step size fails. Once the objective becomes too small, the stepsize fails and the generalization performance of LogitBoost decreases remarkably. The LogitBoost objective is still small and continues to decrease slightly, but the self-averaging properties observed in AdaBoost in the overtraining phase are lost. Note that up until it missteps, LogitBoost is very competitive with AdaBoost. If LogitBoost and AdaBoost were halted at the same high degree of accuracy (e.g., 10^{-8}), there is no evidence of overfitting.

5. Conclusion

MW are correct is saying that optimization provides only part of the picture because optimization tells us nothing about generalization. Mathematical programming theory tells us that more well-posed boosting problems with well-conditioned loss functions (like the hinge loss) and explicit regularization in the objective should produce boosting algorithms with better behavior from an optimization perspective. But AdaBoost's ill-conditioning appears to be one of the secrets of its success. More investigation is needed comparing Adaboost with its regularized counterparts. Certainly machine learning researchers should mind their optimization theory and track the convergence of their algorithms. Optimality conditions should be used to halt and compare boosting algorithms instead of fixed iteration limits.



(a) Objective (log10)



(b) Test Error

Figure 5: Average of 100 trials of Experiment 4.3 (10% Bayes Error) for AdaBoost (red, squares), LogitBoost (blue, triangles), and LogitBoost with .5 shrinkage (purple, circles) for 8-node trees.

References

K.P. Bennett and E. Parrado-Hernández. The interplay of optimization and machine learning research. *Journal of Machine Learning Research*, 7:1265–1281, 2006.

K.P. Bennett, A. Demiriz, and J. Shawe-Taylor. A column generation algorithm for boosting. *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 65–72, 2000.

Y. Freund and R.E. Schapire. Experiments with a new boosting algorithm. *Machine Learning: Proceedings of the Thirteenth International Conference*, 148:156, 1996.

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.

P.E. Gill, W. Murray, and M.H. Wright. *Practical Optimization*. Academic Press, London and New York.

L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. *Advances in Neural Information Processing Systems*, 12:512–518, 2000.

G. Rätsch. *Robust Boosting via Convex Optimization Theory and Applications*. PhD thesis, Universitat Potsdam, 2001.

S. Rosset, J. Zhu, and T. Hastie. Boosting as a regularized path to a maximum margin classifier. *Journal of Machine Learning Research*, 5:941–973, 2004.

C. Rudin, I. Daubechies, and R.E. Schapire. The dynamics of AdaBoost: Cyclic behavior and convergence of margins. *Journal of Machine Learning Research*, 5:1557–1595, 2004.

R.E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, 1998.