The State of Boosting *

Greg Ridgeway
Department of Statistics
University of Washington
Seattle, WA 98195-4322
greg@stat.washington.edu

Abstract

In many problem domains, combining the predictions of several models often results in a model with improved predictive performance. Boosting is one such method that has shown great promise. On the applied side, empirical studies have shown that combining models using boosting methods produces more accurate classification and regression models. These methods are extendible to the exponential family as well as proportional hazards regression models. This article shows that boosting, which is still new to statistics, is widely applicable.

I will introduce boosting, discuss the current state of boosting, and show how these methods connect to more standard statistical practice.

1 Introduction

The trend toward model mixing had a resurgence in economics (Bates and Granger 1969), has increased in the machine learning community, and is partially accepted in the statistics community. Researchers in these fields often had different goals in mind as they developed their methodology. Breiman (1996) specifically wanted to reduce the variance of prediction models and proposed *bagging*. Bayesian model averaging (Madigan et al. 1996, Hoeting et al. 1999) hoped to alleviate the problems associated with model selection and account for the uncertainty involved in model search. Other methods like stacking (Wolpert 1992) and bumping (Tibshirani and Knight 1995) attempted simply to decrease prediction bias. Researchers found that in all these cases a viable solution involved fitting several models and merging the predictions that each model produced.

Boosting is one of the latest prediction methods that proponents believed, at least initially, was the latest addition to the class of model mixing procedures. Boosting, rapidly made popular in the machine learning community, is a topic slowly making its way into mainstream statistics research. Computational learning theorists desired a method for transforming a collection of weak classifiers into one strong classifier (Schapire 1990, Freund 1995). This work culminated in the work by Freund and Schapire (1997) who introduced the *AdaBoost* algorithm, now commonly referred to as the *Discrete AdaBoost* algorithm. They discovered an algorithm that sequentially fits "weak" classifiers to different weightings of the observations in a dataset. Those observations that the previous classifier poorly predicts receive greater weight on the next iteration. The final *AdaBoost* classifier is a weighted average of all the weak classifiers. In a wide variety of classification problems, their weighting scheme and final classifier merge have proven to be an effective method for reducing bias and variance, and improving misclassification rates (Bauer and Kohavi 1999, Dietterich 1998). Empirical evidence has shown that the base classifier can be fairly simplistic (shallow classification trees) and yet, when boosted, can capture complex decision boundaries (Breiman 1998).

From the statistician's point of view, the breakthrough in boosting occured in Friedman, Hastie, and Tibshirani (2000). Although still focused on classification problems, they showed that Freund and Schapire's *AdaBoost* algorithm is an optimization method for finding a classifier that minimizes a particular exponential loss function. Other authors have also made the connection between boosting and optimization (Breiman 1999, Mason et al. 2000). Friedman et al. proceeded to show that this exponential loss function used in *AdaBoost* is closely related to the Bernoulli likelihood and developed a new boosting algorithm that finds a classifier to directly maximize a Bernoulli likelihood. The important finding of this work is that it links the rather obscure work of the computational learning theorist to a likelihood method of standard statistical practice.

In this article I will trace the development of boosting methodology from their computational learning theory origin to the latest perception as functional optimization algorithms. Although boosting algorithms on the surface appear complex, closer inspection will show that they have much in common with how a statistician might fit familiar linear models. This similarity will lead directly into their application to

*Appeared in 1999 in *Computing Science and Statistics 31*, 172–181.

Initialize the weights of each of the N observations to $w_i^{(1)} = \frac{1}{N}$. For t in $1, \ldots, T$ iterate the following steps.

1. Using the weights, estimate $P(Y = 1|\mathbf{x})$.
2. Set $h_t(\mathbf{x}_i) = \log \frac{\hat{P}(Y=1|\mathbf{x})}{\hat{P}(Y=-1|\mathbf{x})}$.
3. Update the observation weights as $w_i^{(t+1)} = w_i^{(t)} \exp(-\alpha_t y_i h_t(\mathbf{x_i}))$.
4. Normalize w_i^{t+1} so that they sum to 1.

The final boosted model suggested by Freund and Schapire that combines the models of each iteration is

$$\mathrm{H}(\mathbf{x}_i) = \mathrm{sign}\left(\sum_{t=1}^{T} h_t(\mathbf{x_i})\right)$$

where α_t is a tuning parameter that we will set to 1.

Figure 1: The Real AdaBoost algorithm

exponential family and proportional hazards regression models, all of which can admit a boosted form. We also can exploit the modular structure of boosting algorithms to ensure that our model has the most desirable statistical properties, robustness, variance reduction, interpretability, and scalability to massive datasets. Lastly, I will address the question as to whether boosting algorithms belong in the class of model mixing and where I believe the future of prediction models lies.

2 Boosting for classification

Even before it was published, Freund and Schapire's *Discrete AdaBoost* was generating great interest. It seemed that the performance of just about any classifier could achieve improvements in both bias and variance. Furthermore, the procedure seemed immune to overfitting. One would simply fit the classifier, $h_1(\mathbf{x})$, to the dataset, upweight the observations it misclassified, and fit another classifier, $h_2(\mathbf{x})$, using the new observation weights. The algorithm iterates several times piling more weight on the difficult to classify observations. In the end all the classifiers participate in a weighted vote for the final classification rule. They developed their particular weight and merge scheme to iteratively reduce an upper bound on the training misclassification error.

Some early research has investigated boosting specific types of classifiers. Classification trees seem to dominate most boosting research because of their flexibility and performance in high-dimensions. Furthermore, boosting tends to improve classification trees greatly since they are often unstable, slight variations in the dataset drastically changing their predictions. Leo Breiman said that AdaBoost with trees is the best "off-the-shelf" classifier (Breiman 1998). The boosted naïve Bayes classifier has also seen some success. Charles Elkan's application of boosted naïve Bayes won first place out of 45 entries in the data mining competition KDD'97 (Elkan 1997). However, while the usual non-boosted naïve Bayes approach leads to elegant and effective explanations (see, for example, Madigan, Mosurski, and Almond 1996 and Becker, Kohavi, and Sommerfield 1997), the *AdaBoost*-ed version destroys this feature. Ridgeway, Madigan, Richardson, and O'Kane (1998) showed that the boosted naïve Bayes classifier forms decision boundaries that are approximately linear in the predictors. This allows easy interpretation of the model's reasoning process.

Besides empirical exploration into the performance of these classifiers, statisticians and computational learning theorists began the search for the explanation. Freund and Schapire (1997) showed that each *AdaBoost* iteration reduces the training error and produced some generalization error bounds based on VC dimension. However, the performance in practice often outpaced this bound.

More recently attention has shifted to a refinement of the original *AdaBoost* algorithm. The *Real AdaBoost* algorithm (Schapire and Singer 1998) produces "confidence rated predictions". Suppose we wish to predict a binary outcome $Y \in \{-1, 1\}$ with a classifier $F(\mathbf{x}) \in (-\infty, \infty)$ that puts out a positive number if it favors $Y = +1$ and a negative number if it favors $Y = -1$. The magnitude of $F(\mathbf{x})$ determines the confidence in the prediction. This is similar to the log-odds interpretation. In detail the *Real AdaBoost* algorithm proceeds as shown in figure 1.

In a rather obscure technical report, Breiman (1999) first noticed that *AdaBoost* was an optimization algorithm. This connection with optimization was clarified and expanded upon in Friedman, Hastie, and Tibshirani (2000). Their argument is as follows. As before, let $Y \in \{-1, 1\}$ and our classifier, $F(\mathbf{x})$, will produce a real-valued output. Let the loss of a particular classifier be

$$J(F) = \mathrm{E}\left[e^{-yF(\mathbf{x})}\middle|\mathbf{x}\right]. \tag{1}$$

The loss function in (1) is small when the signs of y and $F(\mathbf{x})$ agree, or in other words, when the classification is correct. One can also show that (1) is an upper bound, though not a very tight one, on the misclassification rate (Schapire and Singer 1998). Given a current classifier one might wish to improve upon it by adding a new term, $f(\mathbf{x})$, so that $J(F + f) < J(F)$. Minimizing $J(F + f)$ with respect to f yields

that

$$f(\mathbf{x}) = \frac{1}{2}\log\frac{P_w(Y=+1|\mathbf{x})}{P_w(Y=-1|\mathbf{x})}$$

where $P_w(\cdot)$ is a weighted probability estimate with weights $w = e^{-yF(\mathbf{x})}$. Friedman et al. show that repeatedly updating as $\hat{F}(\mathbf{x}) \leftarrow \hat{F}(\mathbf{x}) + f(\mathbf{x})$ is equivalent to the *Real AdaBoost* algorithm.

To the statistician, minimizing the exponential bound (1) seems unnatural since we more often work with maximizing likelihoods. Friedman et al. (2000) noted that the minimizer of (1) is the same as the maximizer of the expected Bernoulli log-likelihood

$$J(F) = \mathrm{E}\ell(F) = \mathrm{E}\left[y^*F(\mathbf{x}) - \log\left(1+e^{F(\mathbf{x})}\right)|\mathbf{x}\right] \qquad (2)$$

where $y^* = \frac{1}{2}(1+y) \in \{0,1\}$. Furthermore, the exponential criterion and the Bernoulli log-likelihood are equivalent to a second order Taylor expansion around $F = 0$. At this point Friedman et al. take the clear step to produce a boosting algorithm that directly maximizes the Bernoulli log-likelihood (2) rather than the exponential misclassification bound (1). Similar to their *Real AdaBoost* derivation, given a current classifier their *LogitBoost* algorithm adds new components to further increase the Bernoulli log-likelihood using Newton steps.

We can select one of many optimization methods to find an f that increases $J(F+f)$, the Bernoulli log-likelihood shown in (2). Friedman et al. chose to derive a Newton algorithm. Given a current estimate for $F(\mathbf{x})$ the Newton update for $\hat{F}(\mathbf{x})$ is

$$\hat{F}(\mathbf{x}) \leftarrow \hat{F}(\mathbf{x}) - \frac{\frac{\partial}{\partial f}J(\hat{F}+f)|_{f=0}}{\frac{\partial^2}{\partial f^2}J(\hat{F}+f)|_{f=0}} \qquad (3)$$

$$= \hat{F}(\mathbf{x}) + \mathrm{E}_w\left[\frac{y^*-p(\mathbf{x})}{p(\mathbf{x})(1-p(\mathbf{x}))}|\mathbf{x}\right] \qquad (4)$$

where $w = p(\mathbf{x})(1-p(\mathbf{x}))$ and $p(\mathbf{x}) = \dfrac{1}{1+e^{-\hat{F}(\mathbf{x})}}$ (5)

Newton optimization, therefore, tells us to repeatedly substitute weighted expectations to improve $F(\mathbf{x})$. Note that the weighting in (5) implies that the largest weights pile onto the difficult to classify observations, those near $p(\mathbf{x}) = \frac{1}{2}$. This is at least similar in spirit to the *AdaBoost* algorithm. Since we do not know the value of the expectations in (4) we can approximate them by choosing one of our favorite regression methods. Substituting a linear model in (4) for the weighted expectation reduces to the iteratively reweighted least squares algorithm for fitting linear logistic regression models. Friedman et al. use CART (Breiman et al. 1984) in their experimental results and show substantial improvement in accuracy over a non-boosted CART classifier. Table 1 shows the effect of boosting on the misclassification rate using five UCI classification datasets (Merz and Murphy 1998).

	CART	AdaBoost	LogitBoost
Breast	4.5%	4.0%	2.9%
Ion	7.6%	6.8%	7.1%
Glass	40.0%	25.7%	26.6%
Sonar	59.6%	20.2%	20.2%
Waveform	36.4%	19.5%	20.6%

Table 1: Misclassification rates of boosted trees - from Friedman, et al (1998)

Computational learning theorists are more eager to claim boosting's association with the theory of margins and support vector machines than with optimization and likelihoods (Schapire 1999a, Schapire et al. 1998). However, more of these authors are beginning to discuss the relationship to statistics (Schapire 1999b, Mason et al. 2000). But once boosting and likelihood methods become related, the door opens to many other statistical models. And so the findings of Friedman, Hastie, and Tibshirani (2000) lead us directly into methodology for generating boosting algorithms for a large class of other likelihood based models.

3 Boosting for regression

At the conclusion of their paper, Freund and Schapire (1997) outline their ideas for applying the *AdaBoost* algorithm to regression problems. However, Drucker (1997) first proposed and tested practical methods for boosting regression. His *ad hoc* method followed the same spirit of the *AdaBoost* algorithm by repeatedly performing weighted tree regression followed by upweighting the poorly predicted observations and downweighting the well predicted ones. Compared with fitting just one tree, his method seems to be competitive. Breiman (1999) suggests how one might deal with boosting regression problems with his *arc-gv* methodology. He promised an investigation in the near future but as of yet has not produced such a study. His more recent work on adaptive bagging (Breiman 2001), which is quite similar to boosting ideas, appears to be a very promising direction showing improvements in terms of bias and variance. Ridgeway, Madigan, and Richardson (1999) proposed mapping the regression problem into a classification problem, applying their boosted naïve Bayes classifier, and transforming the resulting classifier back as a regressor. They showed that this method fits an additive model but also estimates a transformation of the response variable. Their empirical results show that it performs similarly to generalized additive models (Hastie and Tibshirani 1990) and outperforms CART in four simulated examples but slightly worse on one of the two real datasets investigated.

Of these works related to boosting for regression prob-

Initialize $\hat{F}(\mathbf{x}) = \arg\min_\rho \sum_{i=1}^N \Psi(y_i, \rho)$.
For t in $1, \cdots, T$ do

1. Compute the negative gradient as the working response

$$z_i = -\frac{\partial}{\partial F(\mathbf{x}_i)} \Psi(y_i, F(\mathbf{x}_i)) \Big|_{F(\mathbf{x}_i)=\hat{F}(\mathbf{x}_i)} \tag{6}$$

2. Fit a regression model, $f(\mathbf{x})$, predicting z_i from the covariates $\mathbf{x}_i$.

3. Choose a gradient descent step size as

$$\rho = \arg\min_\rho \sum_{i=1}^N \Psi(y_i, \hat{F}(\mathbf{x}_i) + \rho f(\mathbf{x}_i)) \tag{7}$$

4. Update the estimate of $F(\mathbf{x})$ as

$$\hat{F}(\mathbf{x}) \leftarrow \hat{F}(\mathbf{x}) + \rho f(\mathbf{x})$$

Figure 2: Friedman's Gradient Boost algorithm

lems only Breiman (1999) alludes to involving optimization of a regression loss function as part of the boosting algorithm. Friedman (2001) and the companion paper Friedman (1999) extended the work of Friedman, Hastie, and Tibshirani (2000) and created the ground work for a new generation of boosting algorithms. Using the connection between boosting and optimization made explicit in Friedman et al. (2000), this new work proposes the Gradient Boosting Machine.

In any function estimation problem we wish to find a regression function, $\hat{F}(\mathbf{x})$, that minimizes the expectation of some loss function, $\Psi(y, F)$, as shown in (8).

$$\begin{aligned} \hat{F}(\mathbf{x}) &= \arg\min_{F(\mathbf{x})} \mathrm{E}_{y,\mathbf{x}} \Psi(y, F(\mathbf{x})) \\ &= \arg\min_{F(\mathbf{x})} \mathrm{E}_{\mathbf{x}} \left[\mathrm{E}_{y|\mathbf{x}} \Psi(y, F(\mathbf{x})) \Big| \mathbf{x} \right] \end{aligned} \tag{8}$$

We will focus on finding estimates of $F(\mathbf{x})$ such that

$$\hat{F}(\mathbf{x}) = \arg\min_{F(\mathbf{x})} \mathrm{E}_{y|\mathbf{x}} \left[\Psi(y, F(\mathbf{x})) | \mathbf{x} \right]. \tag{9}$$

Parametric regression models assume that $F(\mathbf{x})$ consists of a finite number of parameters, β, and estimates them by selecting those values that minimize a loss function (i.e. squared error loss) over a training sample of N observations on $(y, \mathbf{x})$ pairs as in (10).

$$\hat{\beta} = \arg\min_\beta \sum_{i=1}^N \Psi(y_i, F(\mathbf{x}_i; \beta)) \tag{10}$$

When we wish to estimate $F(\mathbf{x})$ non-parametrically the task becomes more difficult. Again we can proceed similarly to (3) and modify our current estimate of $F(\mathbf{x})$ by adding a new function $f(\mathbf{x})$ in a greedy fashion. Letting $F_i = F(\mathbf{x}_i)$, we see that we want to decrease the N dimensional function

$$J(\mathbf{F}) = \sum_{i=1}^N \Psi(y_i, F(\mathbf{x}_i)) = \sum_{i=1}^N \Psi(y_i, F_i). \tag{11}$$

The negative gradient of $J(\mathbf{F})$ indicates the direction of the locally greatest decrease in $J(\mathbf{F})$. Gradient descent would then have us modify $\mathbf{F}$ as

$$\hat{\mathbf{F}} \leftarrow \hat{\mathbf{F}} - \rho \nabla J(\mathbf{F}) \tag{12}$$

where ρ is the size of the step along the direction of greatest descent. Clearly, this step alone is far from our desired goal. First, it only fits F at values of $\mathbf{x}$ for which we have observations. Second, it does not take into account that observations with similar $\mathbf{x}$ are likely to have similar values of $F(\mathbf{x})$. Both these problems would have disastrous effects on generalization error. However, Friedman suggests selecting a class of functions that use the covariate information to approximate the gradient. This line of reasoning produces his Gradient Boosting algorithm shown in figure 2. At each iteration the algorithm determines the direction, the gradient, in which it needs to improve the fit to the data and selects a particular model from the allowable class of functions that is most in agreement with the direction.

Figure 3 demonstrates the geometry of the gradient boosting machine. In this case the true regression function, $F(\mathbf{x})$, is the sinusoidal curve and our current estimate is the constant line at zero, $\hat{F}(\mathbf{x}) = 0$. Assuming that we had 100 observations from some likelihood based model, we can compute the gradient of the log-likelihood at each of those 100 observations. The vertical lines indicate the gradient direction that each of these points would like to move in order to further increase the associated log-likelihood. Clearly there is randomness, but on average the gradient indicates moves closer to the true $F(\mathbf{x})$. Incorporating covariate information we can approximate the gradient by a one split CART model shown in figure 3 as the step function. The CART model seems to capture the general trend of the gradient and makes a move from $\hat{F}(\mathbf{x}) = 0$ closer to the true $F(\mathbf{x})$. This step function would be the gradient step direction in the function space.

In the case of squared-error loss, $\Psi(y_i, F(\mathbf{x}_i)) = \sum_{i=1}^N (y_i - F(\mathbf{x}_i))^2$, this algorithm corresponds exactly to residual fitting. Friedman also uses this algorithm to develop

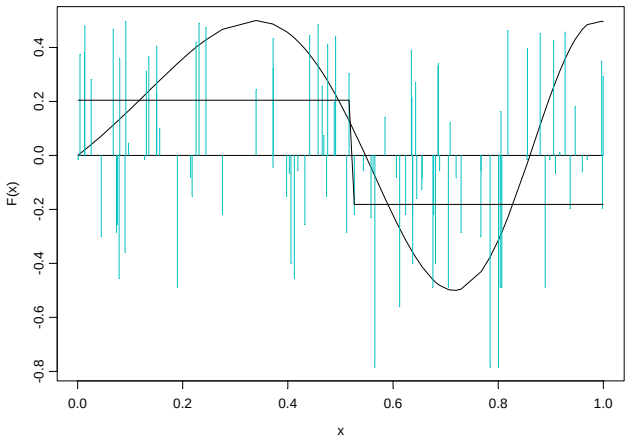

Figure 3: Geometry of the Gradient Boosting Machine

new boosting algorithms for robust regression with least absolute deviation and Huber loss functions (Huber 1964). In the following section I will extend ideas from the *LogitBoost* algorithm and the gradient boosting machine to apply more directly to likelihood based models.

4 Boosting exponential family and survival regression

Now having the machinery to extend boosting to likelihood based models, we can investigate the problem of boosting exponential family and proportional hazards regression models. In this section I will show that we can easily generalize the *LogitBoost* algorithm of Friedman, Hastie, and Tibshirani (2000) to create boosting algorithms for the exponential family regression models. These new boosting algorithms are based on a variant of the Newton-Raphson optimizer known as Fisher scoring (Fisher 1935). Similarly we can also fit these models by selecting likelihood based loss functions for use in Friedman's gradient boosting machine. Lastly, I will present results from a boosted Cox model for proportional hazards regression. Using the gradient framework I will show that we can create boosting algorithms that can utilize existing software for fitting the model's linear counterpart.

4.1 Boosting exponential family regression

Exponential family regression models are most commonly used in their linear form as the generalized linear model. They encompass a large class of methods common to statistical practice including Gaussian, logistic, and log-linear regression models. McCullagh and Nelder (1989) set up the framework for the generalized linear model as follows.

1. *Distributional assumption*: Y has a distribution in the exponential family of the form

$$f(y|\theta,\phi) = \exp\left(\frac{y\theta - b(\theta)}{a(\phi)} - c(y,\phi)\right) \tag{13}$$

   where $b(\theta)$ is the cumulant generating function so that $b'(\theta) = \mathrm{E}(Y) = \mu$.

2. *Model for the mean*: The conditional mean of the response is the modeled functional.

$$\mu_i = \mathrm{E}(Y|\mathbf{x}_i)$$

3. *Systematic component*: The covariates of the regression problem, $x_1, x_2, \cdots, x_d$, form a linear predictor

$$\eta_i = \sum_{j=1}^{d} \beta_j x_{ij} \tag{14}$$

4. *A link function*: The link function is a monotone, differentiable function relating the linear predictor to the response. When g is such that $\theta = \eta$, it is known as the canonical link function.

$$g(\mu_i) = \eta_i$$

Boosting the exponential family regression models relaxes the linear assumption in (14), only assuming that

$$g(\mu_i) = F(x_{i1}, x_{i2}, \ldots, x_{id}). \tag{15}$$

The boosting algorithm searches in a greedy fashion for a function $\hat{F}$ that maximizes the log-likelihood under the assumed model.

$$\mathrm{E}_{Y|\mathbf{x}}\ell(\theta,\phi|y,\mathbf{x}) = \mathrm{E}_{Y|\mathbf{x}}\left[\frac{y\theta - b(\theta)}{a(\phi)}\Big|\mathbf{x}\right]. \tag{16}$$

Generalized additive models (Hastie and Tibshirani 1990) are similar except that they assume that $g(\mu_i)$ is an additive function of the predictor variables. Although boosting allows for more general expansions of $g(\mu_i)$ than GAM, in practice it may be reasonable to restrict F to be additive with low order interactions with an interpretable link function such as the log-odds for the logistic model.

To derive a boosting algorithm for the general case of exponential family models let $b(\theta)$ be a specific cumulant generating function and $g(\mu)$ be a specific link function relating the regressor, $F(\mathbf{x})$, to the conditional mean, μ. In the usual

Initialize $F(\mathbf{x}) = g(\bar{y})$ for all values of $\mathbf{x}$.
For t in $1, \cdots, T$ do

1. Compute the working response

   $$z_i = (y_i - \mu_i) g'(\mu_i).$$

   where $\mu_i = g^{-1}(F(\mathbf{x}_i))$.

2. Fit a regression model, $f(\mathbf{x})$, predicting z_i using $\mathbf{x}_i$ with weights

   $$w_i = \frac{1}{g'(\mu_i)^2 V(\mu_i)}$$

3. Update the boosted regressor as

   $$\hat{F}(\mathbf{x}) \leftarrow \hat{F}(\mathbf{x}) + f(\mathbf{x})$$

Return the final boosted regressor $\hat{F}(\mathbf{x})$.

Figure 4: Boosting by Fisher scoring

GLM notation $b''(\theta) = a(\phi)V(\mu)$ where $V(\mu)$ is known as the variance function. The derivation proceeds similarly to that for the *LogitBoost* algorithm, the details of which are available in Ridgeway (1999). Figure 4 shows a boosting algorithm for the exponential family regression models using Fisher scoring to optimize (16).

We will initialize the estimated regression function to be a constant across all $\mathbf{x}$. We can start with the constant function that maximizes (16) and therefore satisfies the equation $b'(\theta) = \bar{y}$. Since $\mu = b'(\theta)$ and $g(\mu) = F(\mathbf{x})$ then we should initialize $\hat{F}(\mathbf{x}) = g(\bar{y})$ for all values of $\mathbf{x}$.

When we consider boosting the Bernoulli likelihood with a canonical link function we have $g(\mu) = \log \frac{\mu}{1-\mu}$ and $V(\mu) = \mu(1-\mu)$. In this special case the algorithm in figure 4 reduces to the *LogitBoost* algorithm. The results in table 1 show that this algorithm is effective in the Bernoulli case. Friedman (2001) and Friedman (1999) reported extensive experimentation on regression models with Gaussian and slash (heavy tailed) error distributions. He concluded that boosting outperformed MARS (Friedman 1991) in most of the settings that he considered. Ridgeway (1999) considered Poisson regression on a simulated dataset with a non-linear intensity and showed that the boosted Poisson model consistently outperformed GAM.

Initialize $\hat{F}(\mathbf{x}) = 0$. For t in $1, \cdots, T$ do

1. Compute the working response.

   $$z_i = \delta_i - \sum_{j=1}^{N} \delta_j I(t_i \geq t_j) \frac{e^{\hat{F}_i}}{\sum_{k=1}^{N} I(t_k \geq t_j) e^{\hat{F}_k}} \quad (17)$$

2. Construct a regressor, $f(\mathbf{x})$, predicting z_i from $\mathbf{x}_i$.

3. Fit a linear proportional hazards model to the response (t, δ) with predictor $f(\mathbf{x}_i)$, offset $F(\mathbf{x}_i)$, and regression coefficient ρ.

4. Update the boosted estimate of $\hat{F}(\mathbf{x})$.

   $$\hat{F}(\mathbf{x}) = \hat{F}(\mathbf{x}) + \rho f(\mathbf{x}) \quad (18)$$

Figure 5: A boosting algorithm for Cox's proportional hazards regression model

4.2 Boosting proportional hazards regression

Statistical methods for survival analysis model data in which the response variable is a lifetime or failure time (Kalbfleisch and Prentice 1980). Although these methods are mostly thought of in the context of survival times of patients in a medical experiment, they may also apply in other situations such as the time to failure of a mechanical component or the time until criminal recidivism. For these problems we simply wish to estimate the distribution of failure time given covariates. Censoring, the condition of an observation that is yet to fail at the end of the experiment, complicates the estimation process. For each subject in the dataset we observe t_i, the last inspection time of the subject, and δ_i, which takes the value 1 if the subject failed at time t_i and 0 if it was censored.

Fitting survival models by Cox's partial likelihood (Cox 1972, Cox 1975) is the most widely used method for linear proportional hazards regression. Since boosting fits non-linear regression models we can search for $F(\mathbf{x})$ to maximize Cox's log-partial likelihood.

$$\log PL(F|t, \delta, \mathbf{x}) = \sum_{i=1}^{N} \delta_i \left[F(\mathbf{x}_i) - \log \left(\sum_{j=1}^{N} I(t_j \geq t_i) e^{F(\mathbf{x}_j)} \right) \right] \quad (19)$$

The log-partial likelihood takes into account only the order of the failure and censoring times. Since we have set (19) to be

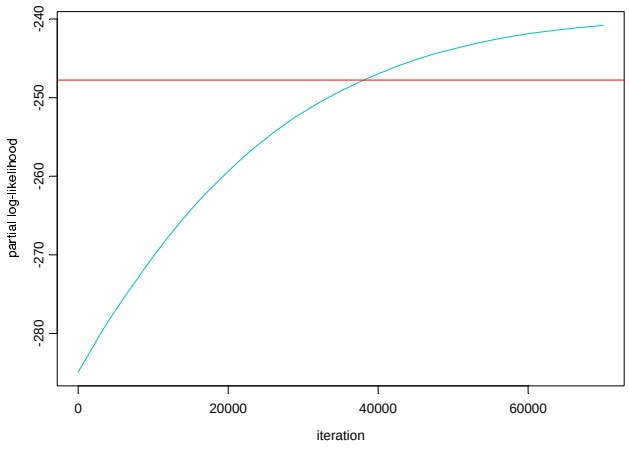

Figure 6: Partial log-likelihood for PBC data. The curve indicates the partial log-likelihood for test set data as the number of boosting iterations increases. The horizontal line indicates the partial likelihood from the linear Cox model.

the term that we want maximized, we can adapt Friedman's gradient boost algorithm to use this as a loss function. In figure 2 we simply need to set $\Psi(y, F) = -\log PL(F|t, \delta, \mathbf{x})$, the negative partial likelihood, and follow the algorithm's remaining steps.

Figure 5 shows an algorithm for fitting a boosted Cox model using gradient boost. The first step is to compute the negative gradient as shown in (6). The result is (17). Conveniently in our case, computing ρ by equation (7) is equivalent to finding the maximum likelihood estimator for a linear Cox model where $f(\mathbf{x}_i)$ is a linear predictor with regression coefficient ρ and $F(\mathbf{x}_i)$ is an offset term. We can take advantage of existing software for fitting linear Cox models for this computation. Terms for which we wish to fit as linear terms may also enter the model at this stage.

For a real data set example to examine the performance of boosting Cox's proportional hazards model, I turn to a clinical trial for testing the drug DPCA for the treatment of primary biliary cirrhosis of the liver (PBC). This dataset has been the subject of several modern data analyses (Fleming and Harrington 1991).

I tested this method by comparing the out-of-sample predictive performance of the linear Cox model to the Cox model boosted by gradient ascent. To judge the out-of-sample predictive performance of the two models, I fit each model on half of the observations, reserving the rest for a test set. I used CART with only one split as the base regressor so that the estimate only captured the main additive effects. Figure 6 shows the value of the log-likelihood on the test cases as the algorithm proceeds. Particularly we see that after many iterations the boosted estimate surpasses the linear Cox model. Subsequently the boosted version continues to improve at a diminishing rate of return. Even though the competition between a linear and non-linear procedure is not entirely fair, this experiment does show that the data carry enough information to support non-linear inference and that boosting can extract that information.

4.3 Improvements on boosting

I have shown so far that boosting is a valuable regression tool. The algorithms' modular structures expose themselves to further improvements. All of the algorithms consist of an update stage of the form

$$\hat{F}(\mathbf{x}) \leftarrow \hat{F}(\mathbf{x}) + \mathrm{E}_w\left[z(y, \hat{F}(\mathbf{x}))|\mathbf{x}\right]. \tag{20}$$

We have already seen that different models lead to different functional forms for $z(y, \hat{F}(\mathbf{x}))$ within this framework. We can also estimate the conditional expectation by the regression method of our choice, planes, smoothers, and trees for example. These are not the limits of the possible variations.

Controlling the learning rate: Earlier work believed that boosting performed especially well in terms of generalization error because it achieved "slow learning". Breiman (1999) theorizes that the success of boosting algorithms depends on its ability to "skate around the inner rim [of the training error surface] instead of slogging to the bottom". If slow learning is an important ingredient than we can introduce a learning rate parameter $\lambda \in (0, 1]$ to control the rate. In the update step of any boosting algorithm we can use λ to dampen the proposed move.

$$\hat{F}(\mathbf{x}) \leftarrow \hat{F}(\mathbf{x}) + \lambda\mathrm{E}_w\left[z(y, \hat{F}(\mathbf{x}))|\mathbf{x}\right]. \tag{21}$$

This modification has been related to shrinkage (Friedman 2001) and to smoothness (Ridgeway 1999).

Variance reduction: Breiman (1996) introduced bagging (*b*ootstrap *agg*regating) as a variance reduction method for unstable regression and classification models. With weak theoretical arguments and substantial empirical ones he shows that bootstrapping simulates an endless stream of new data adequately enough so that averaging reduces prediction variance. Friedman (1999), inspired by Breiman (2001), proposed the stochastic gradient boosting algorithm that simply samples uniformly without replacement from the dataset before estimating the expectation in (20). He found that this additional step greatly improved performance.

Scalability to massive datasets: Subsampling for variance reduction has a beneficial side-effect. On each iteration the regression procedure need consider only a small fraction of

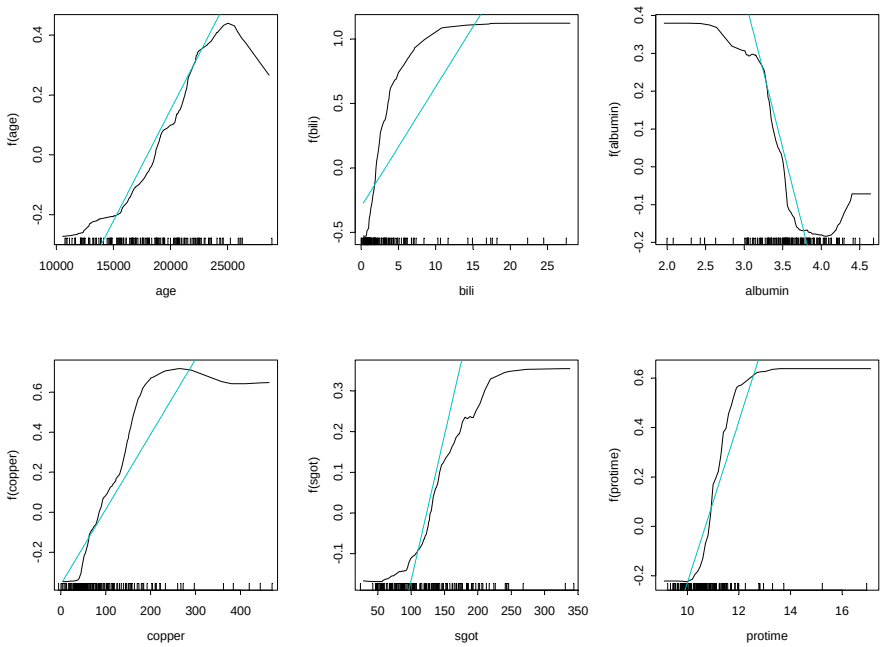

Figure 7: Boosted estimates of the main effects of the PBC data. The curve is the boosted estimate of the function and the line is the estimate from the linear model

the dataset. Great performance is still obtainable subsampling 20% in several situations. When the boosting algorithm involves observation weights, trimming the observations with small weights can further reduce the size of the dataset inspected each time.

Robust regression: Robust regression procedures are less sensitive to outliers. Friedman (2001) develops an entire boosting algorithm for least absolute deviation and M-regression. In (20) we can substitute a robust expectation that would keep the entire process of boosting estimation from being attracted to outlying observations.

Interpretation: In his work on MARS, Friedman (1991) noted that certain function approximation methods are decomposable in terms of a "functional ANOVA decomposition" as

$$F(\mathbf{x}) = \sum_j f_j(x_j) + \sum_{jk} f_{jk}(x_j, x_k) + \cdots. \qquad (22)$$

Friedman et al. (2000) note that this applies directly to boosting trees. One split decision trees depend on only one variable and fall into the first term of (22). Trees with splits on two different predictors fall into the second term of (22) and so on. By restricting the depth of the trees produced on each boosting iteration we can control the order of approximation. When the approximation is restricted to a first order we can also produce plots of x_j versus $f_j(x_j)$ to demonstrate how changes in x_j might affect changes in the response variable.

Figure 7 shows the boosted estimates along with those based on the linear model for the PBC dataset. Every variable shows evidence of either a threshold effect, a saturation effect, or both. Simple linear models clearly can never capture threshold and saturation effects. Although in the region where most of the data points are concentrated, for most of the variables the underlying regressor is nearly linear. For this reason the linear model performs reasonably well. However, for a patient with a more extreme value for any of the six variables the boosted model is far superior. Even though most applications of survival models are not concerned with survival prediction but rather estimation of effects, when effects depart substantially from linearity, as in the PBC dataset, accurately understanding the predictors as well as survival prediction depends on a non-linear estimation procedure like boosting.

For the PBC results in figures 6 and 7 I set a small learning rate $\lambda = 0.00001$, a subsampling fraction of 10%, and fit one split CART models at each of 70,000 iterations.

5 Discussion

Researchers have proposed techniques like bagging (Breiman 1996), Bayesian model averaging (Madigan et al. 1996, Hoeting et al. 1999), stacking (Wolpert 1992), and bumping (Tibshirani and Knight 1995) showing that their methods solved problems associated with dependence on one selected model. At its first introduction computational learning theorists believed that boosting also fit in this class since it built several models on different weightings of the dataset and merged them together. When boosting simple base classifiers, predictive performance on many of the classic benchmark datasets increased. Researchers believed these results implied that boosting was now the best of this class of algorithms.

Research, including this work, is now showing that boosting represents a new class of learning algorithms that Friedman correctly named "gradient machines". As all the extensions described here show, boosting iteratively fits some form of the residual, regions of the sample space where the model's predictions are missing the target data. This being the case the original class of model mixing procedures are not in competition with boosting but rather can coexist inside or outside a boosting algorithm. That is, one could average across boosted estimates or perform bagging and bumping within a boosting iteration. The future of regression may hold new hybrid methods that are robust, have low bias and variance, and adequately account for model uncertainty.

Acknowledgments

The author is grateful to David Madigan, Thomas Richardson, and Werner Stuetzle for many discussions on boosting and for helping in the search for new ways to apply boosting methods. A grant from the National Science Foundation supported this work (DMS 9704573).

References

Bates, J. and C. Granger (1969). The combination of forecasts. *Operations Research Aquarterly 20*, 451–468.

Bauer, E. and R. Kohavi (1999). An empirical comparison of voting classification algorithms: bagging, boosting, and variants. *Machine Learning 36*(1/2), 105–139.

Becker, B., R. Kohavi, and D. Sommerfield (1997). Visualizing the simple Bayesian classifier. In *KDD 1997 Workshop on Issues in the Integration of Data Mining and Data Visualization*.

Breiman, L. (1996). Bagging predictors. *Machine Learning 26*, 123–140.

Breiman, L. (1998). Arcing classifiers. *The Annals of Statistics 26*(3), 801–849.

Breiman, L. (1999, October). Prediction games and arcing algorithms. *Neural computation 11*(7), 1493–1517.

Breiman, L. (2001). Using iterated bagging to debias regressions. *Machine Learning 45*(3), 261–277.

Breiman, L., J. H. Friedman, R. A. Olshen, and C. J. Stone (1984). *Classification and Regression Trees*. Wadsworth.

Cox, D. (1972). Regression models and life tables. *Journal of the Royal Statistical Society (Series B) 34*, 187–203.

Cox, D. (1975). Partial likelihood. *Biometrika 62*, 269–276.

Dietterich, T. G. (1998). An experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. Technical report, Department of Computer Science, Oregon State University.

Drucker, H. (1997). Improving regressors using boosting techniques. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pp. 107–115.

Elkan, C. (1997, September). Boosting and naïve Bayes learning. Technical Report CS97-557, University of California, San Diego.

Fisher, R. (1935). The case of zero survivors (appendix to C.I. Bliss 1935). *Annals of Applied Biology 22*, 164–165.

Fleming, T. and D. Harrington (1991). *Counting processes and survival analysis*. New York: Wiley.

Freund, Y. (1995). Boosting a weak learning algorithm by majority. *Information and Computation 121*(2), 256–285.

Freund, Y. and R. Schapire (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences 55*(1), 119–139.

Friedman, J. (1991). Multivariate adaptive regression splines (with discussion). *Annals of Statistics 19*(1), 1–82.

Friedman, J. (1999, March). Stochastic gradient boosting. Technical report, Stanford University, Statistics Department.

Friedman, J. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics 29*(4).

Friedman, J., T. Hastie, and R. Tibshirani (2000). Additive logistic regression: a statistical view of boosting (with discussion). *Annals of Statistics 28*(2), 337–374.

Hastie, T. and R. Tibshirani (1990). *Generalized Additive Models*. Chapman and Hall.

Hoeting, J., D. Madigan, A. Raftery, and C. Volinsky (1999). Bayesian model averaging: A practical tutorial. *Statistical Science*.

Huber, P. (1964). Robust estimation of a location parameter. *Annals of Mathematical Statistics 35*, 73–101.

Kalbfleisch, J. D. and R. L. Prentice (1980). *The Statistical Analysis of Failure Time Data*. Wiley.

Madigan, D., K. Mosurski, and R. Almond (1996). Explanation in belief networks. *Journal of Computational and Graphical Statistics 6*, 160–181.

Madigan, D., A. E. Raftery, C. Volinsky, and J. Hoeting (1996). Bayesian model averaging. In *AAAI Workshop on Integrating Multiple Learned Models*, pp. 77–83. AAAI Press.

Mason, L., J. Baxter, P. Bartlett, and M. Frean (2000). Boosting algorithms as gradient descent. In S. Solla, T. Leen, and K. Müller (Eds.), *Advances in Neural Information Processing Systems*, Volume 12. MIT Press.

McCullagh, P. and J. Nelder (1989). *Generalized Linear Models* (second ed.). Chapman and Hall.

Merz, C. and P. Murphy (1998). UCI repository of machine learning databases. http://www.ics.uci.edu/ mlearn/MLRepository.html. Irvine, CA: University of California, Department of Information and Computer Science.

Ridgeway, G. (1999). *Generalization of Boosting Algorithms and Applications of Bayesian Inference for Massive Datasets*. Ph. D. thesis, University of Washington.

Ridgeway, G., D. Madigan, and T. Richardson (1999). Boosting methodology for regression problems. In D. Heckerman and J. Whittaker (Eds.), *Proceedings of Artificial Intelligence and Statistics '99*, pp. 152–161.

Ridgeway, G., D. Madigan, T. Richardson, and J. O'Kane (1998). Interpretable boosted naïve Bayes classification. In R. Agrawal, P. Stolorz, and G. Piatetsky-Shapiro (Eds.), *Proceedings, Fourth International Conference on Knowledge Discovery and Data Mining*, pp. 101–104.

Schapire, R. E. (1990). The strength of weak learnability. *Machine Learning 5*(2), 197–227.

Schapire, R. E. (1999a). A brief introduction to boosting. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*. AAAI Press.

Schapire, R. E. (1999b). Theoretical views of boosting. In *Computational Learning Theory: Fourth European Conference, EuroCOLT'99*, pp. 1–10.

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

Schapire, R. E. and Y. Singer (1998). Improved boosting algorithms using confidence-rated predictions. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*.

Tibshirani, R. and K. Knight (1995, November). Model search and inference by bootstrap "bumping". Technical report, Stanford University. To appear in the Journal of Computational and Graphical Statistics.

Wolpert, D. (1992). Stacked generalization. *Neural Networks 5*, 241–259.