Explicit and implicit regularization in supervised learning

Can Yang, MATH

October 20, 2022

Overview

Decision Tree

Adaboost

Gradient Boosting Machine

Historical remarks

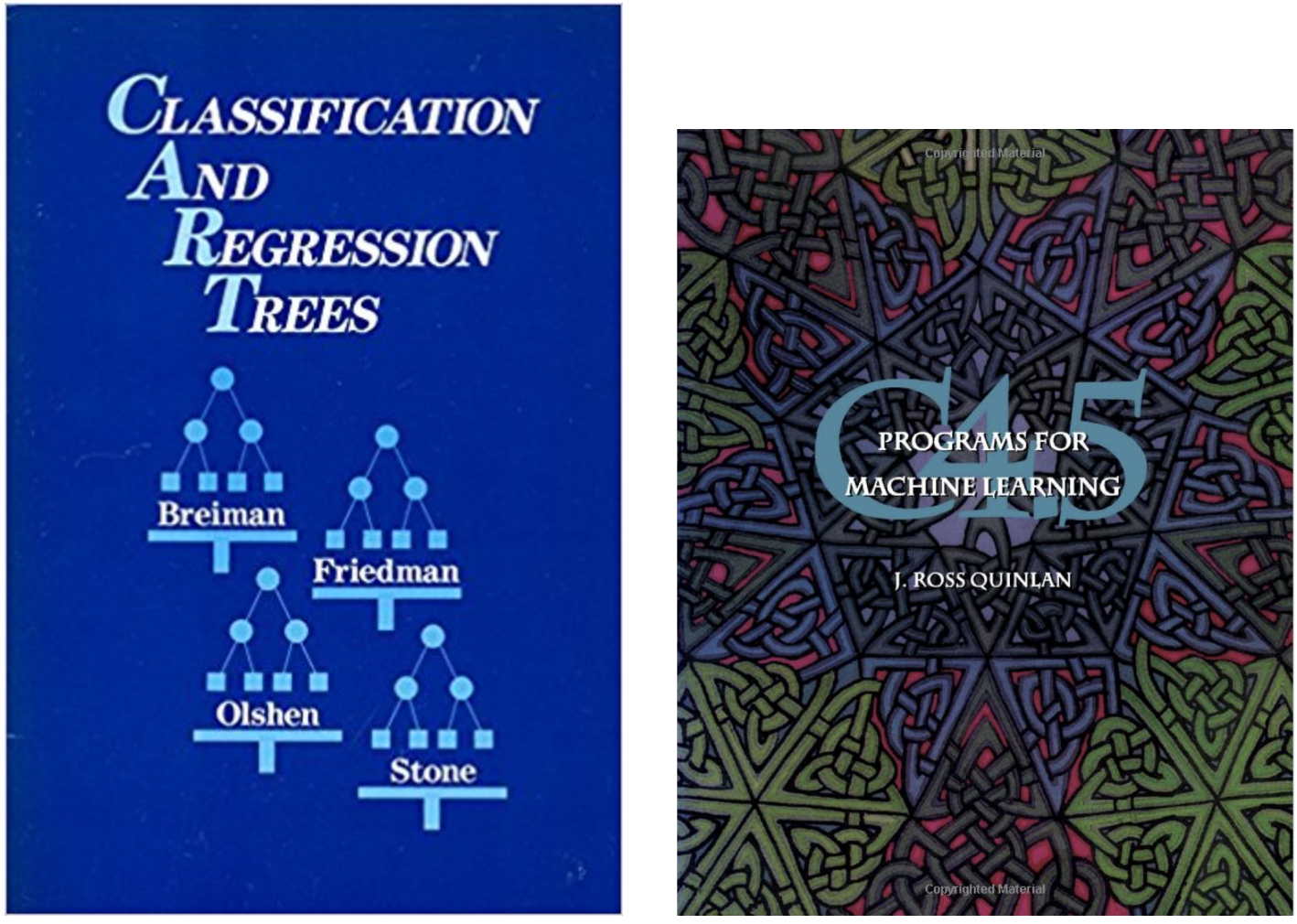


Figure: CART and C4.5

PREFACE

The tree methodology discussed in this book is a child of the computer age. Unlike many other statistical procedures which were moved from pencil and paper to calculators and then to computers, this use of trees was unthinkable before computers.

Binary trees give an interesting and often illuminating way of looking at data in classification or regression problems. They should not be used to the exclusion of other methods. We do not claim that they are always better. They do add a flexible nonparametric tool to the data analyst's arsenal.

Both practical and theoretical sides have been developed in our study of tree methods. The book reflects these two sides. The first eight chapters are largely expository and cover the use of trees as a data analysis method. These were written by Leo Breiman with the exception of Chapter 6 by Richard Olshen. Jerome Friedman developed the software and ran the examples.

Chapters 9 through 12 place trees in a more mathematical context and prove some of their fundamental properties. The first three of these chapters were written by Charles Stone and the last was jointly written by Stone and Olshen.

Trees, as well as many other powerful data analytic tools (factor analysis, nonmetric scaling, and so forth) were originated

by social scientists motivated by the need to cope with actual problems and data. Use of trees in regression dates back to the AID (Automatic Interaction Detection) program developed at the Institute for Social Research, University of Michigan, by Morgan and Sonquist in the early 1960s. The ancestor classification program is THAID, developed at the institute in the early 1970s by Morgan and Messenger. The research and developments described in this book are aimed at strengthening and extending these original methods.

Our work on trees began in 1973 when Breiman and Friedman, independently of each other, "reinvented the wheel" and began to use tree methods in classification. Later, they joined forces and were joined in turn by Stone, who contributed significantly to the methodological development. Olshen was an early user of tree methods in medical applications and contributed to their theoretical development.

Our blossoming fascination with trees and the number of ideas passing back and forth and being incorporated by Friedman into CART (Classification and Regression Trees) soon gave birth to the idea of a book on the subject. In 1980 conception occurred. While the pregnancy has been rather prolonged, we hope that the baby appears acceptably healthy to the members of our statistical community.

Figure: Historical story about CART

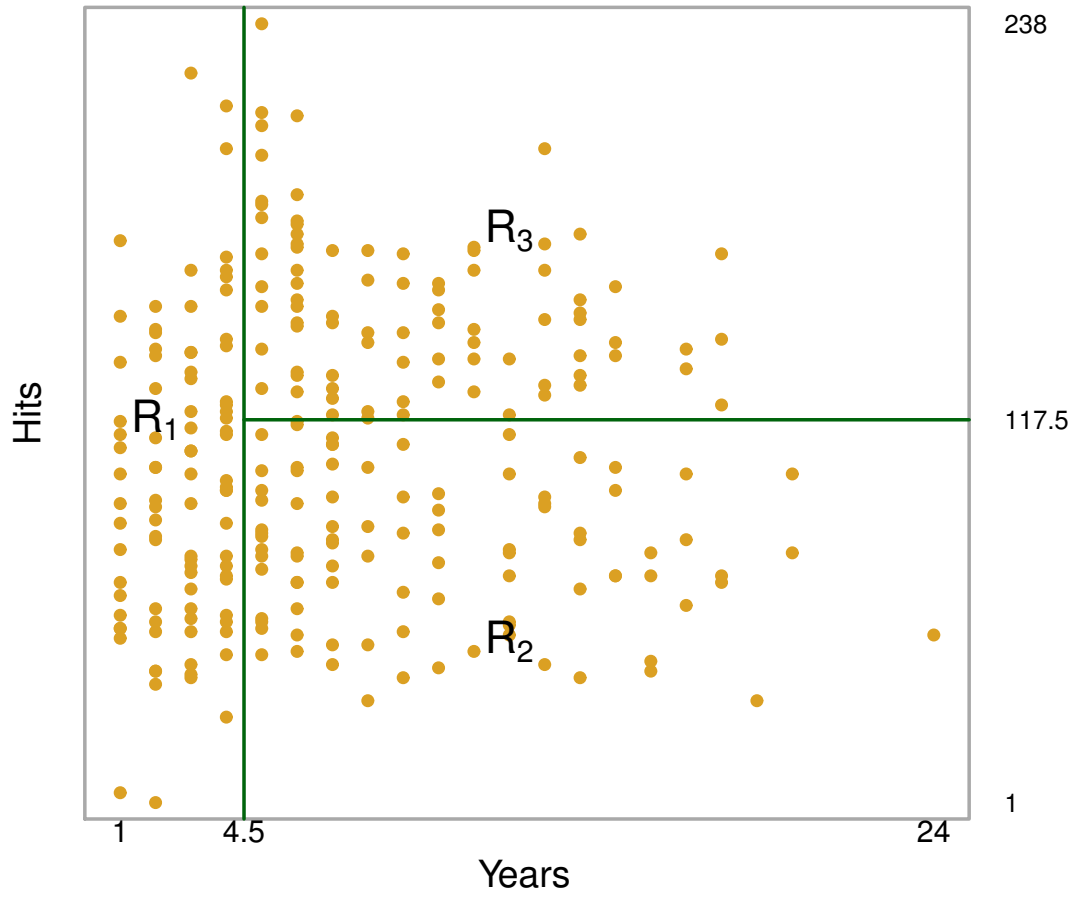
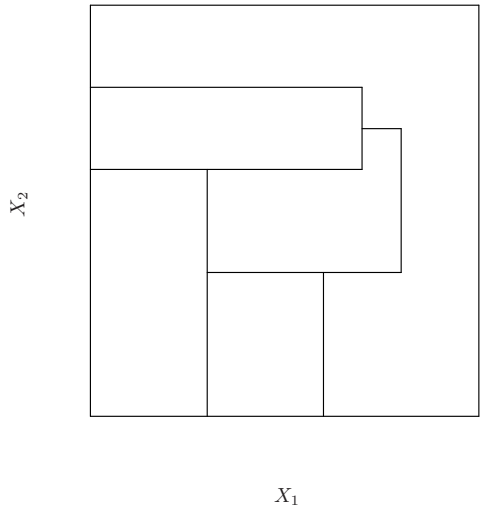
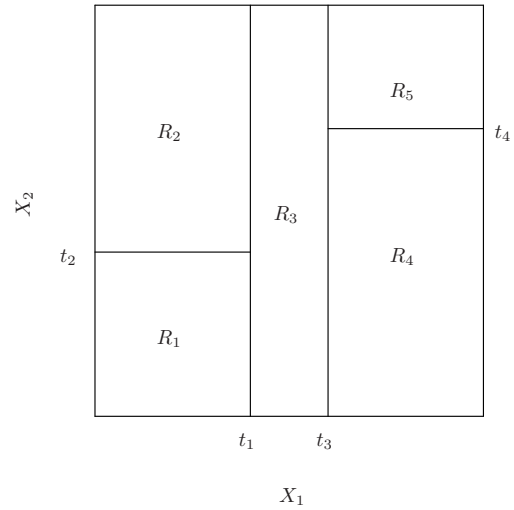
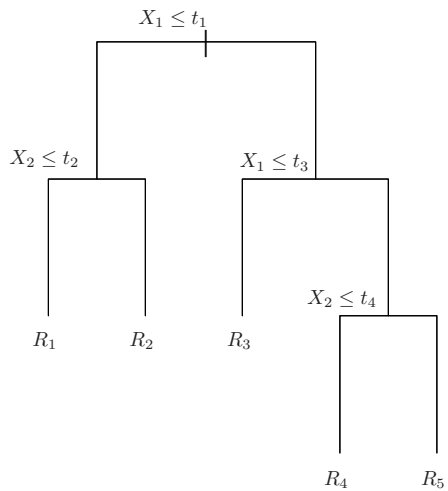


Figure: The three-region partition of the Hitters data set from the regression tree. Figure Source: ISL.


X_2
X_1


R_1
R_2
R_3
R_4
R_5
t_1
t_2
t_3
t_4
X_2
X_1


$X_1 \leq t_1$
$X_2 \leq t_2$
$X_1 \leq t_3$
$X_2 \leq t_4$
R_1
R_2
R_3
R_4
R_5

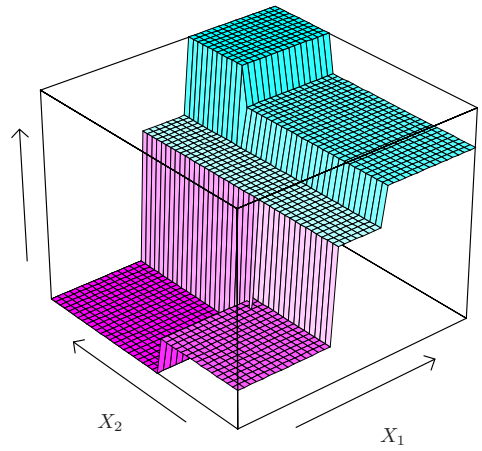

X_2
X_1

Figure 3. Top Left: A partition of two-dimensional feature space that could not result from recursive binary splitting. Top Right: The output of recursive binary splitting on a two-dimensional example. Bottom Left: A tree corresponding to the partition in the top right panel. Bottom Right: A perspective plot of the prediction surface corresponding to that tree (Figure 9.2 from ESL).

Adaboost

Machine Learning: Proceedings of the Thirteenth International Conference, 1996.

Experiments with a New Boosting Algorithm

Yoav Freund **Robert E. Schapire**

AT&T Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974-0636

{yoav, schapire}@research.att.com

Figure: Ref: Experiments with a New Boosting Algorithm. ICML 1996. `https://cseweb.ucsd.edu/~yfreund/papers/boostingexperiments.pdf`.

Adaboost ($y_i \in \{-1, 1\}$)

- 1. Initialize the data weights $\{w_i\}$ by setting $w_i^{(1)} = 1/n$ for $i = 1, \ldots, n$.
- 2. For $m = 1, \ldots, M$:
  - (a) Fit a classifier $f_m(\mathbf{x})$ to the training data by minimizing the weighted error function

$$J_m = \sum_{i=1}^{n} w_i^{(m)} \mathbb{I}(f_m(\mathbf{x}_i) \neq y_i), \tag{1}$$

    where $\mathbb{I}(f_m(\mathbf{x}_i) \neq y_i)$ is the indicator function and equals 1 when $f_m(\mathbf{x}_i) \neq y_i$ and 0 otherwise.
  - (b) Evaluate the quantities

$$\epsilon_m = \frac{\sum_{i=1}^{n} w_i^{(m)} \mathbb{I}(f_m(\mathbf{x}_i) \neq y_i)}{\sum_{i=1}^{n} w_i^{(m)}}, \alpha_m = \log\left\{\frac{1 - \epsilon_m}{\epsilon_m}\right\} \tag{2}$$

  - (c) Update the data weights

$$w_i^{(m+1)} = w_i^{(m)} \exp\{\alpha_m \mathbb{I}(f_m(\mathbf{x}_i) \neq y_i)\} \tag{3}$$

- 3. Make prediction by

$$F_M(\mathbf{x}) = \text{sign}\left(\sum_{m=1}^{M} \alpha_m f_m(\mathbf{x})\right). \tag{4}$$

Illustration of Adaboost

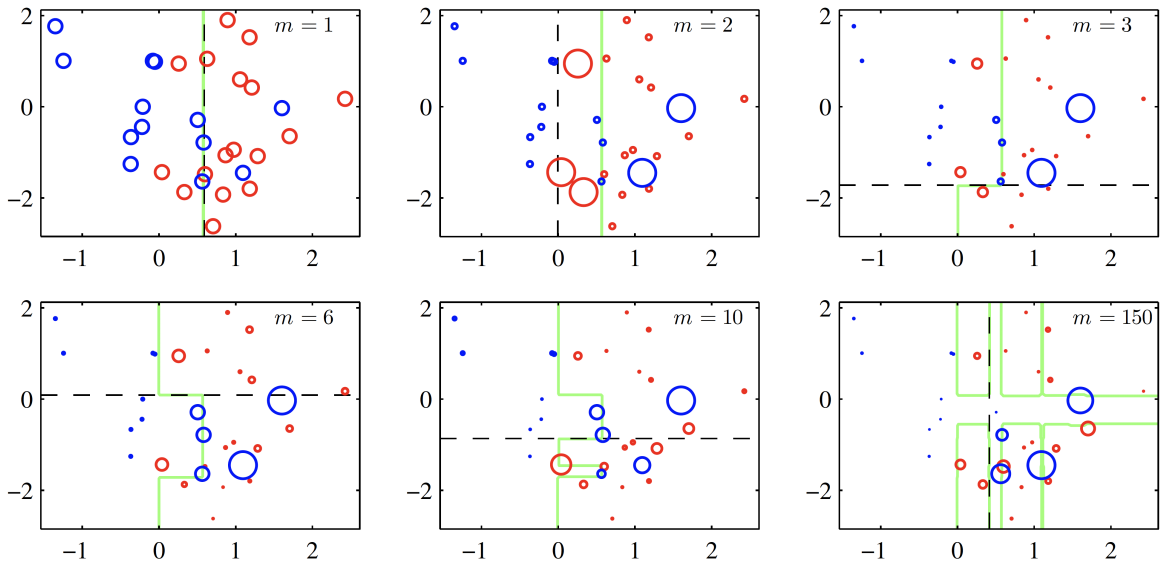


Figure: Each figure shows the number m of base learners trained so far, along with the decision boundary of the most recent base learner (dashed black line) and the combined decision boundary of the ensemble (solid green line). Each data point is depicted by a circle whose radius indicates the weight assigned to that data point when training the most recently added base learner. Thus, for instance, we see that points that are misclassified by the $m = 1$ base learner are given greater weight when training the $m = 2$ base learner. Bishop (2006).

Performance of Adaboost

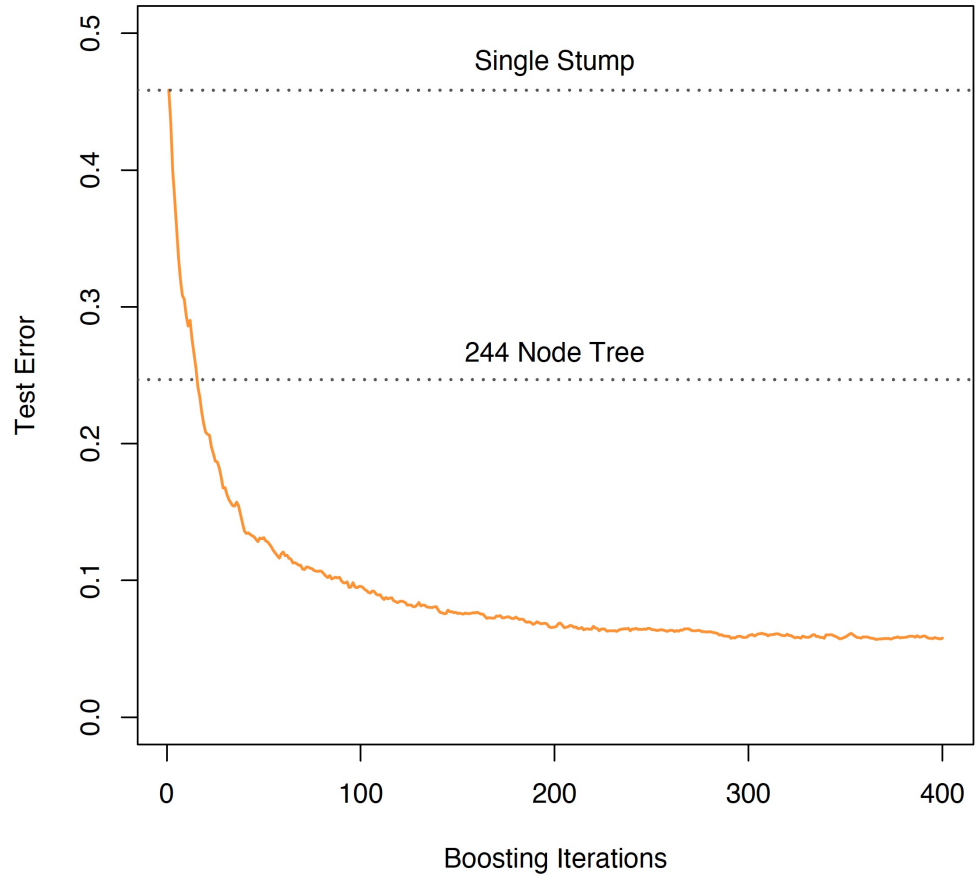


Figure: Test error rate for boosting with stumps, as a function of the number of iterations. Also shown are the test error rate for a single stump, and a 244-node classification tree. **Adaboost seems to be resistant to overfitting**. Figure Source: Figure 10.2 ELS.

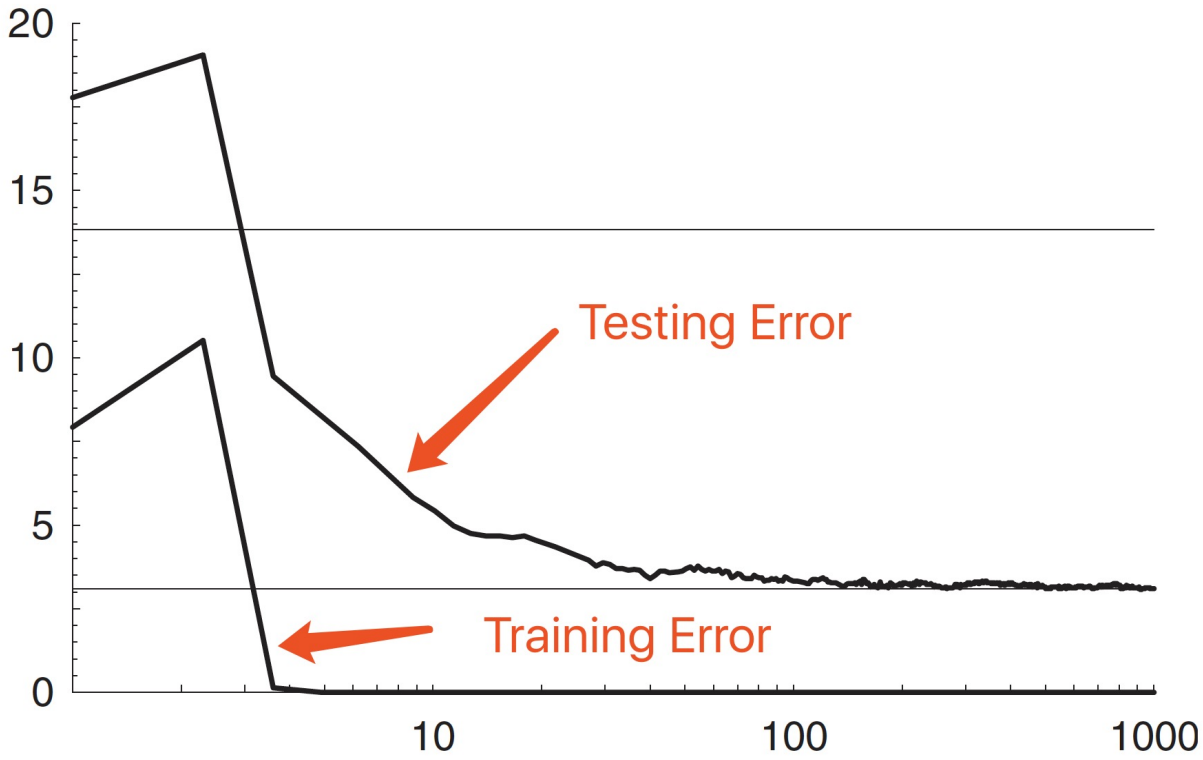


Figure: The training and test percent error rates obtained using boosting on an OCR dataset with C4.5 as the base learner. The top and bottom curves are test and training error, respectively. The top horizontal line shows the test error rate using just C4.5. The bottom line shows the final test error rate of AdaBoost after 1000 rounds. Ref: Peter Bartlett, Yoav Freund, Wee Sun Lee, and Robert E. Schapire "Boosting the margin: a new explanation for the effectiveness of voting methods". `https://projecteuclid.org/euclid.aos/1024691352`.

The "Best off-the shelf classifier in the world"

Much has been written about the success of AdaBoost in producing accurate classifiers. Many authors have explored the use of a tree-based classifier for $f_m(x)$ and have demonstrated that it consistently produces significantly lower error rates than a single decision tree. In fact, Breiman (1996) (referring to a NIPS workshop) called AdaBoost with trees the "best off-the-shelf classifier in the world" [see also Breiman (1998b)]. Interestingly, in many examples the test error seems to consistently decrease and then level off as more classifiers are added, rather than ultimately increase. For some reason, it seems that AdaBoost is resistant to overfitting.

Discrete AdaBoost [Freund and Schapire (1996b)]

1. Start with weights $w_i = 1/N, i = 1, \ldots, N$.
2. Repeat for $m = 1, 2, \ldots, M$:
   (a) Fit the classifier $f_m(x) \in \{-1, 1\}$ using weights w_i on the training data.
   (b) Compute $\text{err}_m = E_w[1_{(y \neq f_m(x))}]$, $c_m = \log((1 - \text{err}_m)/\text{err}_m)$.
   (c) Set $w_i \leftarrow w_i \exp[c_m 1_{(y_i \neq f_m(x_i))}]$, $i = 1, 2, \ldots, N$, and renormalize so that $\sum_i w_i = 1$.
3. Output the classifier $\text{sign}[\sum_{m=1}^{M} c_m f_m(x)]$.

Figure: Ref: Ann. Statist. Volume 28, Number 2 (2000), 337-407. https://projecteuclid.org/euclid.aos/1016218223.

Additive Logistic Regression: a Statistical View of Boosting

JEROME FRIEDMAN * †
TREVOR HASTIE * ‡
ROBERT TIBSHIRANI * ‡

August 20, 1998
Revision April 24, 1999
Second Revision November 30, 1999

Abstract

Boosting (Schapire 1990, Freund & Schapire 1997) is one of the most important recent developments in classification methodology. Boosting works by sequentially applying a classification algorithm to reweighted versions of the training data, and then taking a weighted majority vote of the sequence of classifiers thus produced. For many classification algorithms, this simple strategy results in dramatic improvements in performance. We show that this seemingly mysterious phenomenon can be understood in terms of well known statistical principles, namely additive modeling and maximum likelihood. For the two-class problem, boosting can be viewed as an approximation to additive modeling on the logistic scale using maximum Bernoulli likelihood as a criterion. We develop more direct approximations and show that they exhibit nearly identical results to boosting. Direct multi-class generalizations based on multinomial likelihood are derived that exhibit performance comparable to other recently proposed multi-class generalizations of boosting in most situations, and far superior in some. We suggest a minor modification to boosting that can reduce computation, often by factors of 10 to 50. Finally, we apply these insights to produce an alternative formulation of boosting decision trees. This approach, based on best-first truncated tree induction, often leads to better performance, and can provide interpretable descriptions of the aggregate decision rule. It is also much faster computationally, making it more suitable to large scale data mining applications.

Figure: Ref: Ann. Statist. Volume 28, Number 2 (2000), 337-407. https://projecteuclid.org/euclid.aos/1016218223.

Statistical view of Adaboost

- Consider the exponential error function

$$E = \sum_{i=1}^{n} \exp\{-y_i F_M(\mathbf{x}_i)\}, \tag{5}$$

where $F_M(\mathbf{x}) = \frac{1}{2} \sum_{m=1}^{M} \alpha_m f_m(\mathbf{x})$ and $y_i \in \{-1, 1\}$.

- Our goal is to minimize E w.r.t. both α_m and $f_m(\mathbf{x})$.
- Instead of doing a global error function miminization, we shall suppose that the base classifiers $f_1(\mathbf{x}), \ldots, f_{m-1}(\mathbf{x})$ are fixed, as are their coefficients $\alpha_1, \ldots, \alpha_{m-1}$, and so we are minimizing only w.r.t. α_m and $f_m(\mathbf{x})$.

$$\begin{aligned} E &= \sum_{i=1}^{n} \exp\left\{-y_i F_{m-1}(\mathbf{x}_i) - \frac{1}{2} y_i \alpha_m f_m(\mathbf{x}_i)\right\} \\ &= \sum_{i=1}^{n} w_i^{(m)} \exp\{-\frac{1}{2} y_i \alpha_m f_m(\mathbf{x}_i)\} \end{aligned} \tag{6}$$

where $w_i^{(m)} = \exp\{-y_i F_{m-1}(\mathbf{x}_i)\}$ can be viewed as constants.

- Denote $\mathcal{T}_m$ and $\mathcal{M}_m$ as the sets of correctly classifed data points by $f_m(\mathbf{x})$ and the misclassified points, respectively.
- We have

$$
\begin{aligned}
E &= e^{-\alpha_m/2} \sum_{i \in \mathcal{T}_m} w_i^{(m)} + e^{\alpha_m/2} \sum_{i \in \mathcal{M}_m} w_i^{(m)} \\
&= (e^{\alpha_m/2} - e^{-\alpha_m/2}) \sum_{i=1}^{n} w_i^{(m)} \mathcal{I}(f_m(\mathbf{x}_i) \neq y_i) + e^{-\alpha_m/2} \sum_{i=1}^{n} w_i^{(m)}.
\end{aligned} \tag{7}
$$

- When we minimize this w.r.t. $f_m(\mathbf{x})$, the second term is constant, and so the is equivalent to minimizing (1) because the overall multiplicative factor in front of the summation does not affect the location of the minimum.

- Minimizing w.r.t. α_m

$$\frac{\partial E}{\partial \alpha_m} = (\frac{e^{\alpha_m/2}}{2} + \frac{e^{-\alpha_m/2}}{2}) \sum_{i=1}^{n} w_i^{(m)} \mathcal{I}(f_m(\mathbf{x}_i) \neq y_i) - \frac{e^{-\alpha_m/2}}{2} \sum_{i=1}^{n} w_i^{(m)} = 0$$

- Solving the above equation gives

$$\begin{aligned} \alpha_m &= \log \left(\frac{\sum_{i=1}^{n} w_i^{(m)}}{\sum_{i=1}^{n} w_i^{(m)} \mathcal{I}(f_m(\mathbf{x}_i) \neq y_i)} - 1 \right), \\ &= \log \frac{1 - \epsilon_m}{\epsilon_m}, \end{aligned}$$

where $\epsilon_m = \frac{\sum_{i=1}^{n} w_i^{(m)} \mathcal{I}(f_m(\mathbf{x}_i) \neq y_i)}{\sum_{i=1}^{n} w_i^{(m)}}$.

- Hence, we obtain (2).

- From (6), we see that, having found α_m and $f_m(\mathbf{x})$, the weights on the data points are updated using

$$w_i^{(m+1)} = w_i^{(m)} \exp\left\{-\frac{1}{2} y_i \alpha_m f_m(\mathbf{x}_i)\right\}.$$

- Making use of the fact that

$$y_i f_m(\mathbf{x}_i) = 1 - 2\mathbb{I}(f_m(\mathbf{x}_i) \neq y_i),$$

we see that the weight $w_i^{(m)}$ are updated at the next iteration using

$$w_i^{(m+1)} = w_i^{(m)} \exp(-\alpha_m/2) \exp\{\alpha_m \mathbb{I}(f_m(\mathbf{x}_i) \neq y_i)\}.$$

- Because the term $\exp(-\alpha_m/2)$ is independent of n, we see that it weights all data points by the same factor and so can be discarded. Thus we obtain (3).
- Finally, once all the base classifiers are trained, new data points are classified by evaluating the **sign** of the combined function. Because the factor of $1/2$ does not affect the sign, this gives (4).

Why exponiental loss function?

- We have shown that the Adaboost algorithm minimizes the exponential loss function (5). How to justify the minimization of the exponential loss function?
- Consider the population version of the exponential loss function,

$$\mathbb{E}_{\mathbf{x},y}[\exp\{-yF(\mathbf{x})\}] = \sum_y \int \exp\{-yF(\mathbf{x})\}p(y|\mathbf{x})p(\mathbf{x})d\mathbf{x}$$

- Taking the functional derivative w.r.t. $F(\mathbf{x})$, we get

$$\frac{\delta}{\delta F(\mathbf{x})}\mathbb{E}_{\mathbf{x},y}[\exp\{-yF(\mathbf{x})\}] = -\sum_y y\exp\{-yF(\mathbf{x})\}p(y|\mathbf{x})p(\mathbf{x})$$
$$= \{\exp\{F(\mathbf{x})\}p(y=-1|\mathbf{x}) - \exp\{-F(\mathbf{x})\}p(y=1|\mathbf{x})\}p(\mathbf{x}).$$

- Setting this equal to zero and rearranging, we obtain

$$F(\mathbf{x}) = \frac{1}{2}\log\left\{\frac{p(y=1|\mathbf{x})}{p(y=-1|\mathbf{x})}\right\},$$

justifying the use of the sign function to arrive the final classification decision.

Greedy Function Approximation: A Gradient Boosting Machine

Jerome H. Friedman*
IMS 1999 Reitz Lecture

February 24,1999 (modified March 15, 2000, April 19, 2001)

Abstract

Function estimation/approximation is viewed from the perspective of numerical optimization in function space, rather than parameter space. A connection is made between stagewise additive expansions and steepest–descent minimization. A general gradient–descent "boosting" paradigm is developed for additive expansions based on any fitting criterion. Specific algorithms are presented for least–squares, least–absolute–deviation, and Huber–M loss functions for regression, and multi–class logistic likelihood for classification. Special enhancements are derived for the particular case where the individual additive components are regression trees, and tools for interpreting such "TreeBoost" models are presented. Gradient boosting of regression trees produces competitive, highly robust, interpretable procedures for both regression and classification, especially appropriate for mining less than clean data. Connections between this approach and the boosting methods of Freund and Shapire 1996, and Friedman, Hastie, and Tibshirani 2000 are discussed.

Figure: Ann. Statist. Volume 29, Number 5 (2001), 1189-1232.
https://projecteuclid.org/euclid.aos/1013203451.

Loss functions for classification

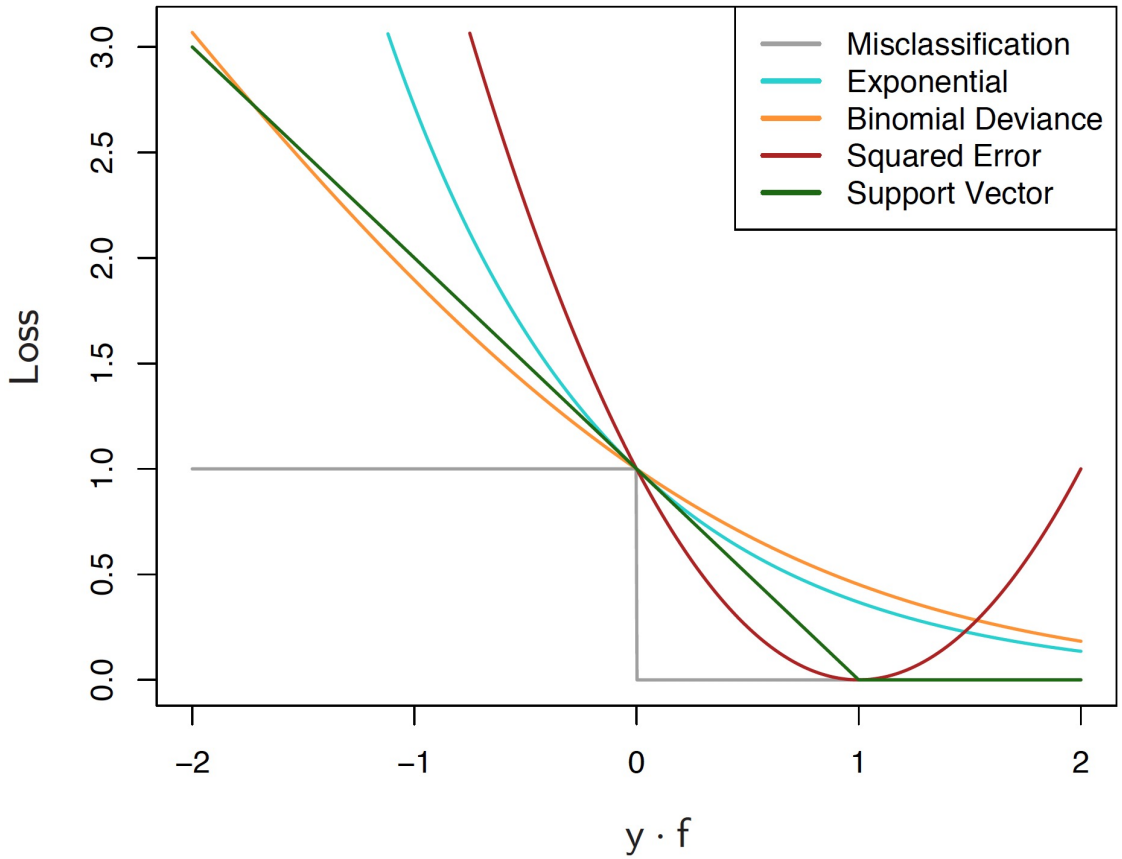


Figure: Loss functions for two-class classification. The response is $y = \pm 1$; the prediction is f, with class prediction $sign(f)$. The losses are misclassification: $I(sign(f) \neq y)$; exponential: $exp(-yf)$; binomial deviance: $log(1 + exp(-2yf))$; squared error: $(y - f)^2$; and support vector: $(1 - yf)_+$. Note that yf **is called margin for classification**.

Loss functions for regression

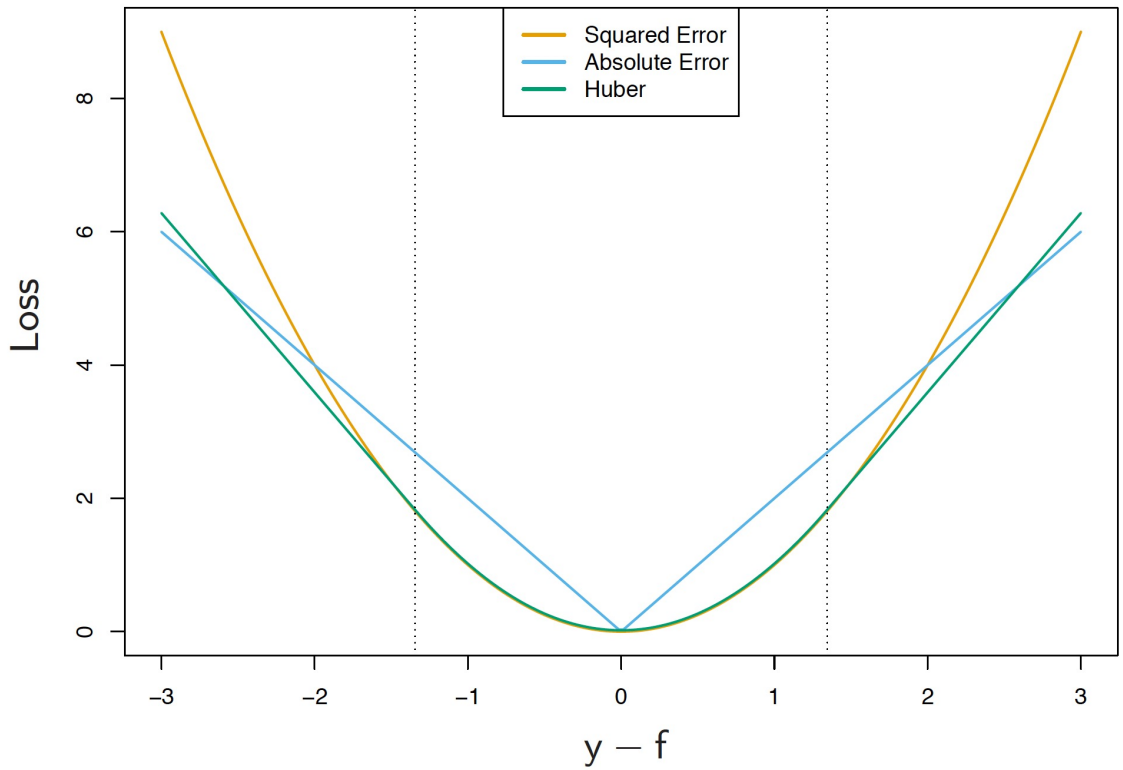


Figure: A comparison of three loss functions for regression, plotted as a function of the margin $y - f$. The Huber loss function combines the good properties of squared-error loss near zero and absolute error loss when $|y - f|$ is large. ELS Figures 10.4 and 10.6.

Forward Stagewise Additive Modeling

Consider a function with basis function expansions

$$f(x) = \sum_{m=1}^{M} \beta_m b(x; \gamma_m),$$

where $\beta_m, m = 1, \ldots, M$ are the coefficients, and $b(x; \gamma) \in \mathbb{R}$ are ususally simple functions of the multivariate arguement x, characterized by a set of parameters γ.

- Initialize $f_0(x) = 0$.
- For $m = 1$ to M
  (a) Compute

$$(\beta_m, \gamma_m) = \arg\min_{\beta,\gamma} \sum_{i=1}^{N} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)).$$

  (b) Set $f_m(x) = f_{m-1}(x) + \beta_m b(x; \gamma_m)$.

Forward Stagewise Additive Modeling: Squared-error loss

- Consider squared-error loss

$$L(y, f(x)) = (y - f(x))^2,$$

- We have

$$\begin{aligned} L(y_i, f_{m-1}(x_i) + \beta b(x_i; \gamma)) &= (y_i - f_{m-1}(x_i) - \beta b(x_i; \gamma))^2 \\ &= (r_{im} - \beta b(x_i; \gamma))^2, \end{aligned}$$

where $r_{im} = y_i - f_{m-1}(x_i)$ is simply the residual of the current model on the ith observation.

Friedman's Gradient Boosting algorithm

Initialize $\hat{F}_0(\mathbf{x})$ to be a constant, $\hat{F}_0(\mathbf{x}) = \arg\min_\rho \sum_{i=1}^n L(y_i, \rho)$.
For m in $1, \ldots, M$ do

1. Compute the negative gradient as the working response

$$r_i = -\frac{\partial}{\partial F_{m-1}(\mathbf{x}_i)} L(y_i, F_{m-1}(\mathbf{x}_i)) \Big|_{F_{m-1}(\mathbf{x}_i)=\hat{F}_{m-1}(\mathbf{x}_i)} \tag{8}$$

2. Fit a regression model, $g(\mathbf{x})$, predicting r_i from the covariates $\mathbf{x}_i$.
3. Choose a gradient descent step size as

$$\rho = \arg\min_\rho \sum_{i=1}^n L(y_i, \hat{F}_{m-1}(\mathbf{x}_i) + \rho g(\mathbf{x}_i)) \tag{9}$$

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

$$\hat{F}_m(\mathbf{x}) = \hat{F}_{m-1}(\mathbf{x}) + \rho g(\mathbf{x}) \tag{10}$$

Gradient Boosting Trees for Regression

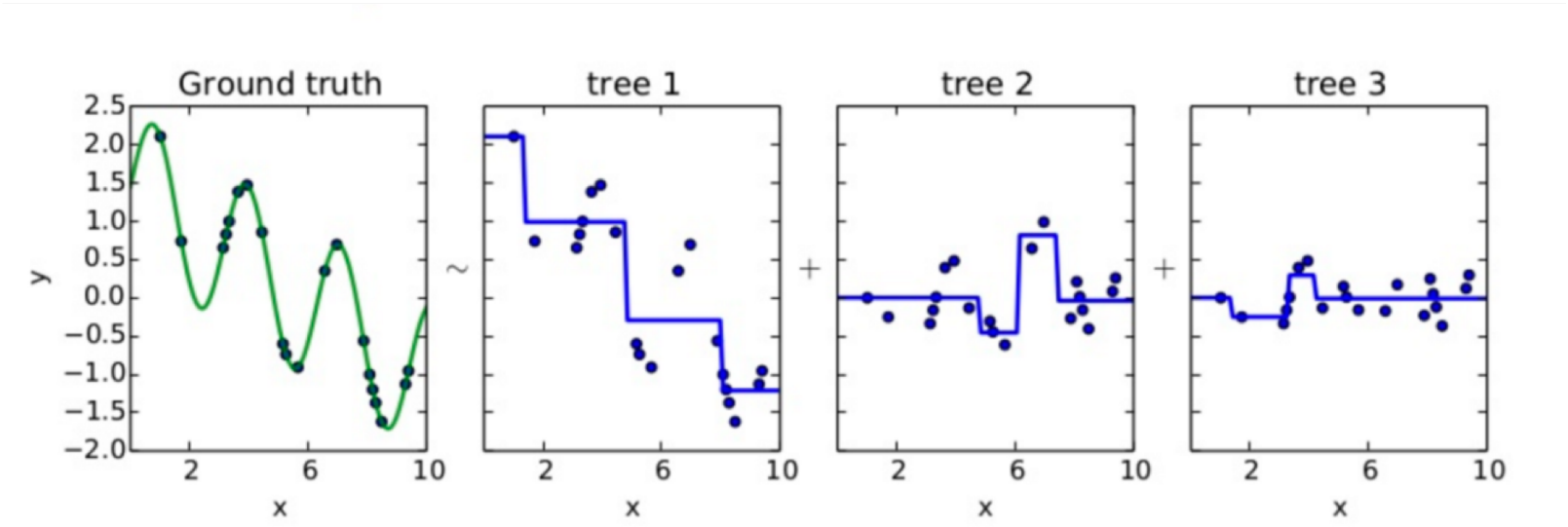


Figure: Source https://www.quora.com/How-would-you-explain-gradient-boosting-machine-learning-technique-in-no-more-than-300-words-to-non-science-major-college-students

Stochastic Gradient Boosting Tree

Initialize $\hat{F}_0(\mathbf{x})$ to be a constant, $\hat{F}_0(\mathbf{x}) = \arg\min_\rho \sum_{i=1}^n L(y_i, \rho)$
For m in $1, \ldots, M$ do

1. Compute the negative gradient as the working response

$$r_i = -\frac{\partial}{\partial F_{m-1}(\mathbf{x}_i)} L(y_i, F_{m-1}(\mathbf{x}_i)) \Big|_{F_{m-1}(\mathbf{x}_i)=\hat{F}_{m-1}(\mathbf{x}_i)} \tag{11}$$

2. Randomly select $\text{prop} \times n$ cases from the dataset, e.g., $\text{prop} = 0.5$.
3. Fit a regression tree with K terminal nodes, $g(\mathbf{x}) = \mathbb{E}(r|\mathbf{x})$. This tree is fitted using only those randomly selected observations
4. Compute the optimal terminal node predictions, $\rho_1, \ldots, \rho_K$, as

$$\rho_k = \arg\min_\rho \sum_{\mathbf{x}_i \in S_k} L(y_i, \hat{F}_{m-1}(\mathbf{x}_i) + \rho) \tag{12}$$

   where S_k is the set of $\mathbf{x}$s that define terminal node k. Again this step uses only the randomly selected observations.

5. Update $\hat{F}_m(\mathbf{x})$ as

$$\hat{F}_m(\mathbf{x}) = \hat{F}_{m-1}(\mathbf{x}) + \lambda \rho_{k(\mathbf{x})} \tag{13}$$

   where $k(\mathbf{x})$ indicates the index of the terminal node into which an observation with features $\mathbf{x}$ would fall.

An example with more details

- Consider the deviance of the logistic regression model:

$$-\frac{2}{n}\sum_{i=1}^{n}[y_i f(\mathbf{x}_i) - \log(1+\exp(f(\mathbf{x}_i)))]$$

  where $y_i \in \{0,1\}$.
- Initial value $f_0 = \log \frac{\sum_i y_i}{\sum_i (1-y_i)}$.
- The negative gradient $-g_i = y_i - p_i$, where $p_i = \frac{1}{1+\exp(-f(\mathbf{x}_i))}$.
- At m-th iteration, grow a tree to approximate the current negative gradient $-g_{i,m}$, with $f(x_i)$ evaluated at $\hat{F}_{m-1}(\mathbf{x}_i)$.

$$\Theta_m = \arg \min_{\Theta; i \in \text{minibatch}} (-g_{i,m} - T(x_i;\Theta))^2,$$

  where Θ_m includes K partitioned regions and fitted constants, $\{S_k, \rho_k\}_{k=1,\ldots,K}$.
- Adjust the fitted constants $\rho_k = \frac{\sum_{i \in S_k}(y_i - p_i)}{\sum_{i \in S_k} p_i(1-p_i)}$.

Details for adjusting the fitted constant

- Consider the optimization problem at a given terminal node

$$\min_{\rho} L(\rho) = \min_{\rho} -\frac{2}{n} \sum_{i \in S_k} [y_i(\hat{F} + \rho) - \log(1 + \exp(\hat{F} + \rho))].$$

- The gradient $g = \frac{\partial L}{\partial \rho} = -\frac{2}{n} \sum_i (y_i - p_i)$, where $p_i = 1/(1 + \exp(-(\hat{F} + \rho))$.
- Hessian $H = \frac{\partial L^2}{\partial \rho^2} = -\frac{2}{n} \sum_i p_i(1 - p_i)$.
- Initialize $\rho = 0$, here one-step Newton update applies

$$\rho = 0 - H^{-1} g = \frac{\sum_{i \in S_k} (y_i - p_i)}{\sum_{i \in S_k} p_i(1 - p_i)}.$$

Stochastic Gradient Boosting

Jerome H. Friedman*

March 26, 1999

Abstract

Gradient boosting constructs additive regression models by sequentially fitting a simple parameterized function (base learner) to current "pseudo"–residuals by least–squares at each iteration. The pseudo–residuals are the gradient of the loss functional being minimized, with respect to the model values at each training data point, evaluated at the current step. It is shown that both the approximation accuracy and execution speed of gradient boosting can be substantially improved by incorporating randomization into the procedure. Specifically, at each iteration a subsample of the training data is drawn at random (without replacement) from the full training data set. This randomly selected subsample is then used in place of the full sample to fit the base learner and compute the model update for the current iteration. This randomized approach also increases robustness against overcapacity of the base learner.

Figure: Computational Statistics & Data Analysis Volume 38, Issue 4, 28 February 2002, Pages 367-378

Stochastic Gradient Boosting

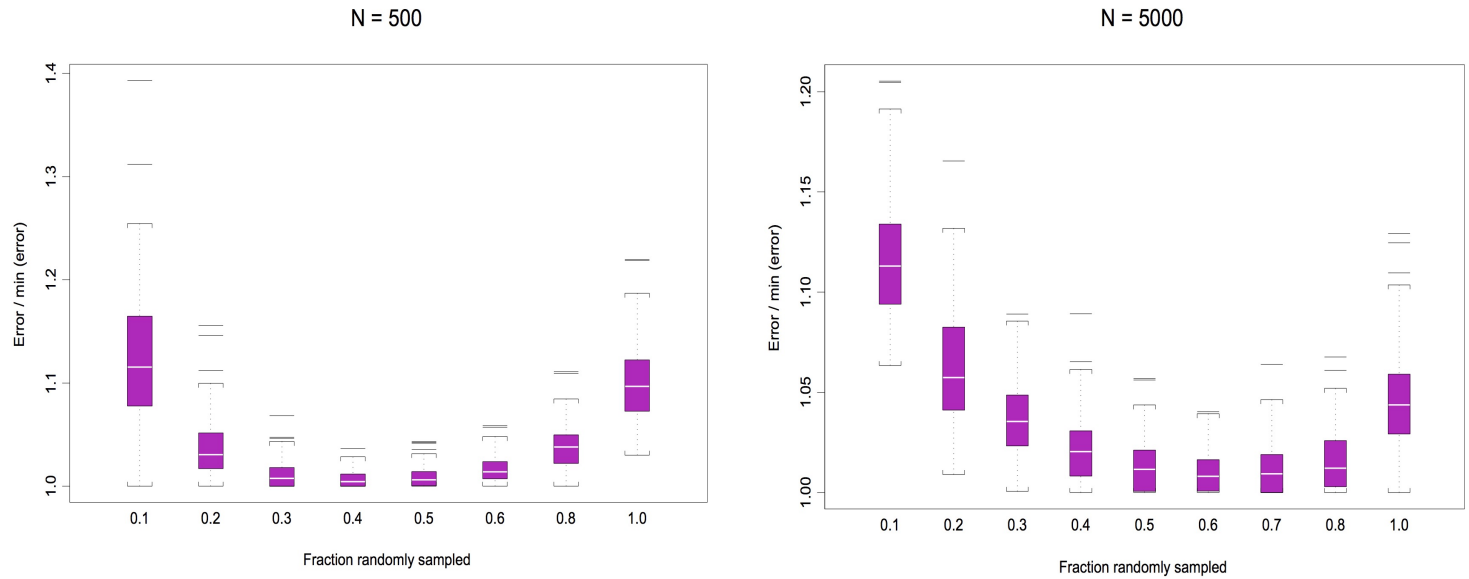


Figure: Distributions of absolute approximation error, relative to the best, for training on different fractions of randomly selected observations from the full training sample at each iteration, for small ($N = 500$) and moderate ($N = 5,000$) sized data sets and Gaussian errors.

Shrinkage and Overfitting

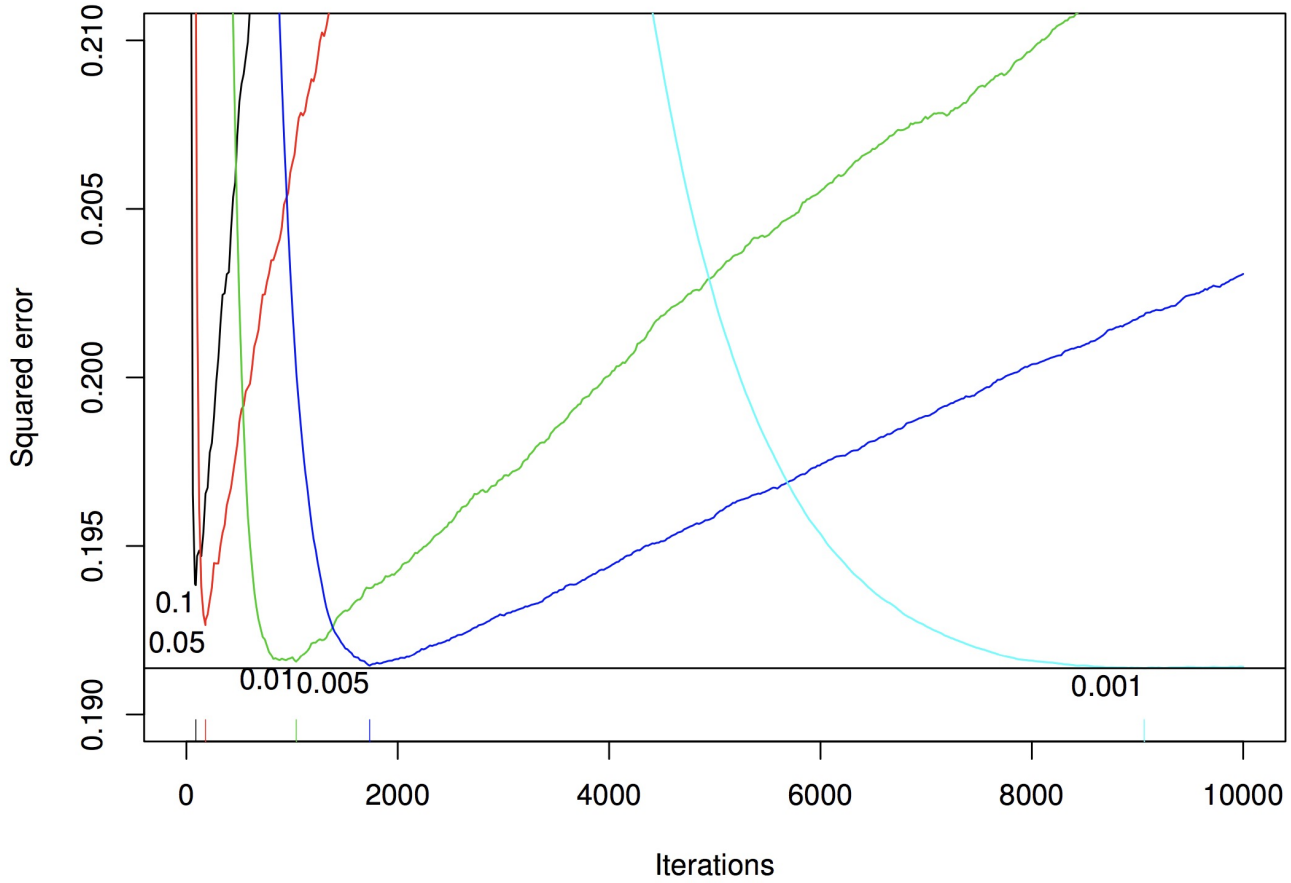


Figure: Out-of-sample predictive performance by number of iterations and shrinkage. Smaller values of the shrinkage parameter offer improved predictive performance, but with decreasing marginal improvement. Figure Source: vignettes of 'gbm'.

R package: gbm

```
library(gbm)
# Simulate data
set.seed(101) # for reproducibility
N <- 1000
X1 <- runif(N)
X2 <- 2 * runif(N)
X3 <- ordered(sample(letters[1:4], N, replace = TRUE), levels = letters[4:1])
X4 <- factor(sample(letters[1:6], N, replace = TRUE))
X5 <- factor(sample(letters[1:3], N, replace = TRUE))
X6 <- 3 * runif(N)
mu <- c(-1, 0, 1, 2)[as.numeric(X3)]
SNR <- 10 # signal-to-noise ratio
Y <- X1 ^ 1.5 + 2 * (X2 ^ 0.5) + mu
sigma <- sqrt(var(Y) / SNR)
Y <- Y + rnorm(N, 0, sigma)
X1[sample(1:N, size = 500)] <- NA # introduce some missing values
X4[sample(1:N, size = 300)] <- NA # introduce some missing values
data <- data.frame(Y, X1, X2, X3, X4, X5, X6)
```

R package: gbm

```
# Fit a GBM
set.seed(102) # for reproducibility
gbm1 <- gbm(Y ~ ., data = data, var.monotone = c(0, 0, 0, 0, 0, 0),
            distribution = "gaussian", n.trees = 100, shrinkage = 0.1,
            interaction.depth = 3, bag.fraction = 0.5, train.fraction = 0.5,
            n.minobsinnode = 10, cv.folds = 5, keep.data = TRUE,
            verbose = FALSE, n.cores = 1)
# Check performance using the out-of-bag (OOB) error; the OOB error typically
# underestimates the optimal number of iterations
best.iter <- gbm.perf(gbm1, method = "OOB")
print(best.iter)
# Check performance using the 50% heldout test set
best.iter <- gbm.perf(gbm1, method = "test")
print(best.iter)
# Check performance using 5-fold cross-validation
best.iter <- gbm.perf(gbm1, method = "cv")
print(best.iter)
# Plot relative influence of each variable
par(mfrow = c(1, 2))
summary(gbm1, n.trees = 1) # using first tree
summary(gbm1, n.trees = best.iter) # using estimated best number of trees
```

Summary: Statistical view of boosting methods

Boosting methods have three important properties that contribute to their success:

- they fit an additive model in a flexible set of basis functions.
- they use a suitable loss function for the fitting process.
- they regularize by forward stagewise fitting; with shrinkage this mimics an L_1(Lasso) penalty on the weights; Without shriankge this approximates an L_0 penalty.

How is boosting connected with L_1?
Why is Boosting often resistant to overfitting?

The Lasso and Boosting

Start with $\mathbf{y} = \mathbf{y} - mean(\mathbf{y})$ and assume $\mathbf{x}_j$ standardized.

The Lasso

$$\hat{\beta}(t) = \arg\min_{\beta} \frac{1}{2}\|\mathbf{y} - \mathbf{x}\beta\|^2, \text{subject to } \|\beta\|_1 \leq t.$$

Boosting (Stagewise fitting) for linear regression

- Start with $\beta^{(0)} = 0$.
- Repeat, for $k = 1, 2, 3 \ldots$
  - Compute negative gradient $\mathbf{g} = \mathbf{x}^T(\mathbf{y} - \mathbf{x}\beta^{(k-1)})$
  - Find j such that
  $$j^* = \arg\max_j |g_j|.$$
  - Only update the j^*-th coefficient with a small amount, say, ϵ:
  $$\beta_{j^*}^{(k)} = \beta_{j^*}^{(k-1)} + \epsilon \cdot \text{Sign}(g_{j^*}).$$

Overfitting

REJOINDER

JEROME FRIEDMAN, TREVOR HASTIE AND ROBERT TIBSHIRANI

Stanford University

We thank the discussants for their generous and thoughtful comments, and the editors for arranging this discussion. Since the paper and discussions are long, we can only reply to a few of the many important points raised by the discussants.

1. Overfitting. All of the discussions raise the issue of overfitting in the context of boosting. Breiman, and to a lesser extent, Freund and Schapire suggest that our explanation is inadequate since it suggests that enough boosting steps can lead to overfitting. Breiman cites empirical evidence suggesting that "AdaBoost almost never overfits the data no matter how many iterations it is run." It would be a nice world if that were the case, but unfortunately it isn't quite. Besides the example in our paper, Ridgeway provides a simple example that exhibits dramatic overfitting in Figure 2 of his discussion. Quoting Ratsch, Onoda and Muller (2000), "Recent studies with highly noisy patterns [Quinlan (1996), Grove and Schuurmans (1998), Ratsch (1998)] showed that it is clearly a myth that Boosting methods do not overfit." Still, it does appear that boosting is more resistant to overfitting that one might expect based on general modeling experience. Buja suggests that the suboptimal greedy "stagewise" nature of the fitting procedure may provide some resistance. This could be the case, and merits further investigation. However, we agree with Bühlmann and Yu that it is largely the nature of zero–one loss that provides general resistance to overfitting as judged in classification.

Overfitting

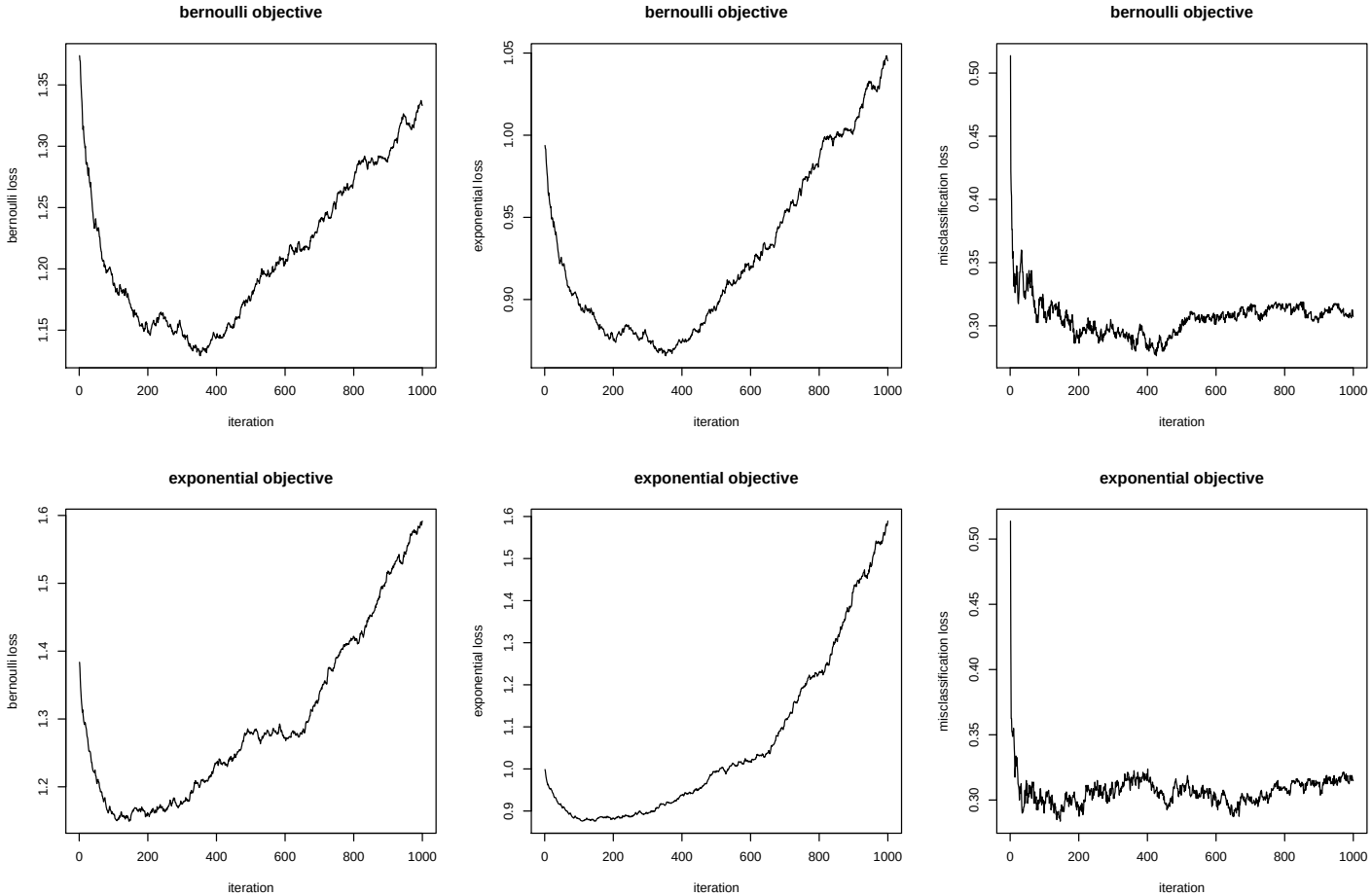


Figure: Upper: Trained with Bernoulli loss function (i.e., logistic loss) and evaluated with Bernoulli loss, Exp loss and zero-one loss on testing data. Lower: Trained with Exp loss and evaluated on Bernoulli loss, Exp loss and zero-one loss.

Rethinking about overfitting for classification problems

- Let $p(x) = p(y = 1|x)$ and $\hat{p}(x) = \hat{p}(y = 1|x)$ be the true probability and estimated probablity, respectively.
- Suppose we have testing data $x_1^*, \ldots, x_n^*$. Squared loss

$$\frac{1}{n}\sum_{i=1}^{n}[p(x_i^*) - \hat{p}(x_i^*)]^2$$

- The averaged negative log-likelihood of the Bernoulli model is

$$-\frac{1}{n}\sum_{i=1}^{n}[y_i \log(\hat{p}(x_i^*)) + (1 - y_i)\log(1 - \hat{p}(x_i^*))]$$

- Expected negative log likelihood of the Bernoulli model (population version, also known as log loss)

$$-\frac{1}{n}\sum_{i=1}^{n}[p(x_i^*) \log(\hat{p}(x_i^*)) + (1 - p(x_i^*))\log(1 - \hat{p}(x_i^*))].$$

Rethinking about overfitting for classification problems

- Exponential loss ($y_i \in \{0, 1\}$) is given as

$$L = \frac{1}{n}\sum_i \exp(-(2y_i - 1)F(x_i))$$
$$= \frac{1}{n}\sum_i [y_i \exp(-F(x_i)) + (1 - y_i)\exp(F(x_i))].$$

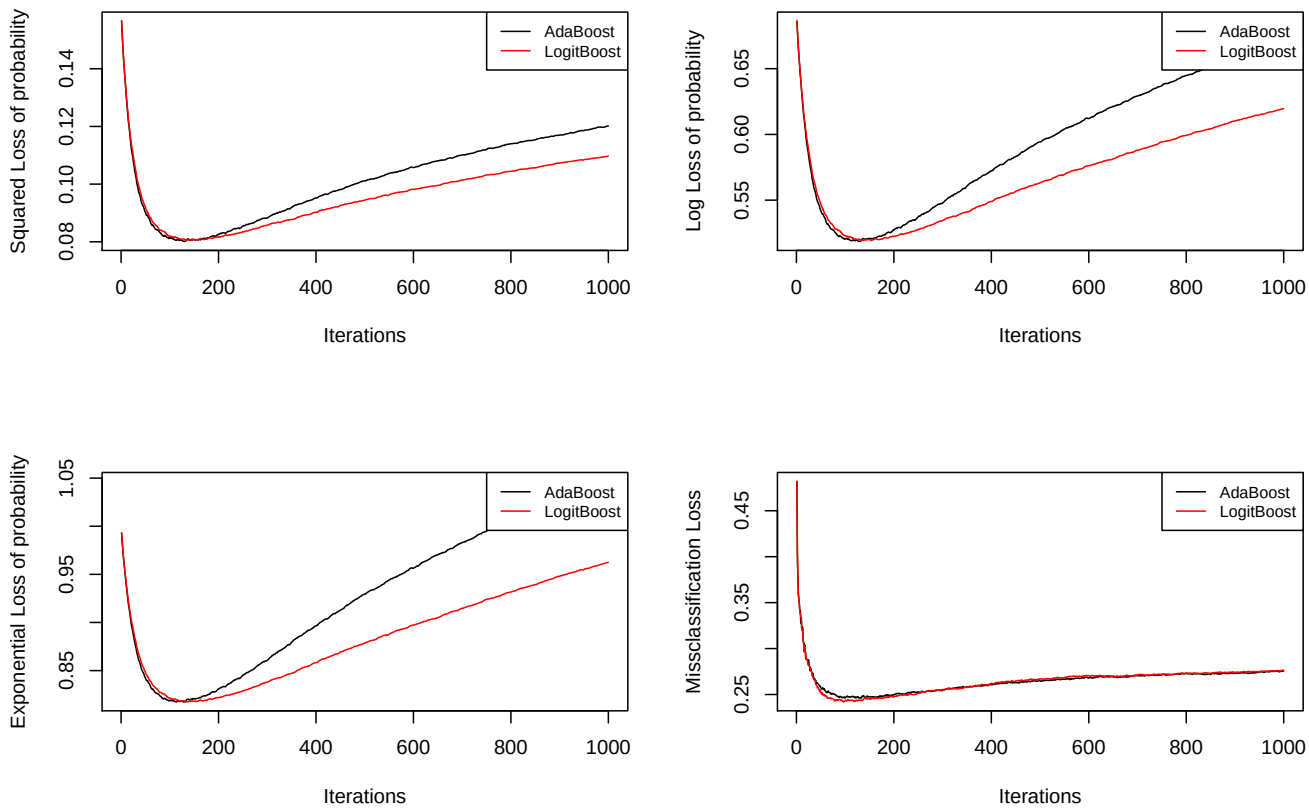
- Recall that $F(x_i) = \frac{1}{2}\log\frac{p(y_i=1|x_i)}{p(y_i=0|x_i)}$.
- Let $\hat{p}(x_i^*) = \hat{p}(y_i^* = 1|x_i^*)$. We have $\exp(\hat{F}(x_i^*)) = \sqrt{\frac{\hat{p}(x_i^*)}{1-\hat{p}(x_i^*)}}$ and the expected loss function becomes

$$\mathbb{E}[L] = \mathbb{E}\left[\frac{1}{n}\sum_i \left(y_i\sqrt{\frac{1-\hat{p}(x_i^*)}{\hat{p}(x_i^*)}} + (1-y_i)\sqrt{\frac{\hat{p}(x_i^*)}{1-\hat{p}(x_i^*)}}\right)\right]$$
$$= \frac{1}{n}\sum_i \left[p(x_i^*)\sqrt{\frac{1-\hat{p}(x_i^*)}{\hat{p}(x_i^*)}} + (1-p(x_i^*))\sqrt{\frac{\hat{p}(x_i^*)}{1-\hat{p}(x_i^*)}}\right].$$

- It turns out that calibrating the conditional class probability is much more challenge than estimating its median.

An example

Consider $p(y_i = 1|x_i) = q + (1 - 2q)\mathbb{I}\left[\left(\sum_{j=1}^{J} x_{ij}\right) > J/2\right]$, where x_i is i.i.d. from $[0, 1]^d$. The class label is only related to the first J variables. We set $q = 0.1$, $J = 5$, $d = 20$ and $n = 200$, and used stumps as base learners.



Journal of Machine Learning Research 9 (2008) 131-156 Submitted 10/05; Revised 7/07; Published 2/08

Evidence Contrary to the Statistical View of Boosting

David Mease MEASE_D@COB.SJSU.EDU
Department of Marketing and Decision Sciences
College of Business, San Jose State University
San Jose, CA 95192-0069, USA

Abraham Wyner AJW@WHARTON.UPENN.EDU
Department of Statistics
Wharton School, University of Pennsylvania
Philadelphia, PA, 19104-6340, USA

Editor: Yoav Freund

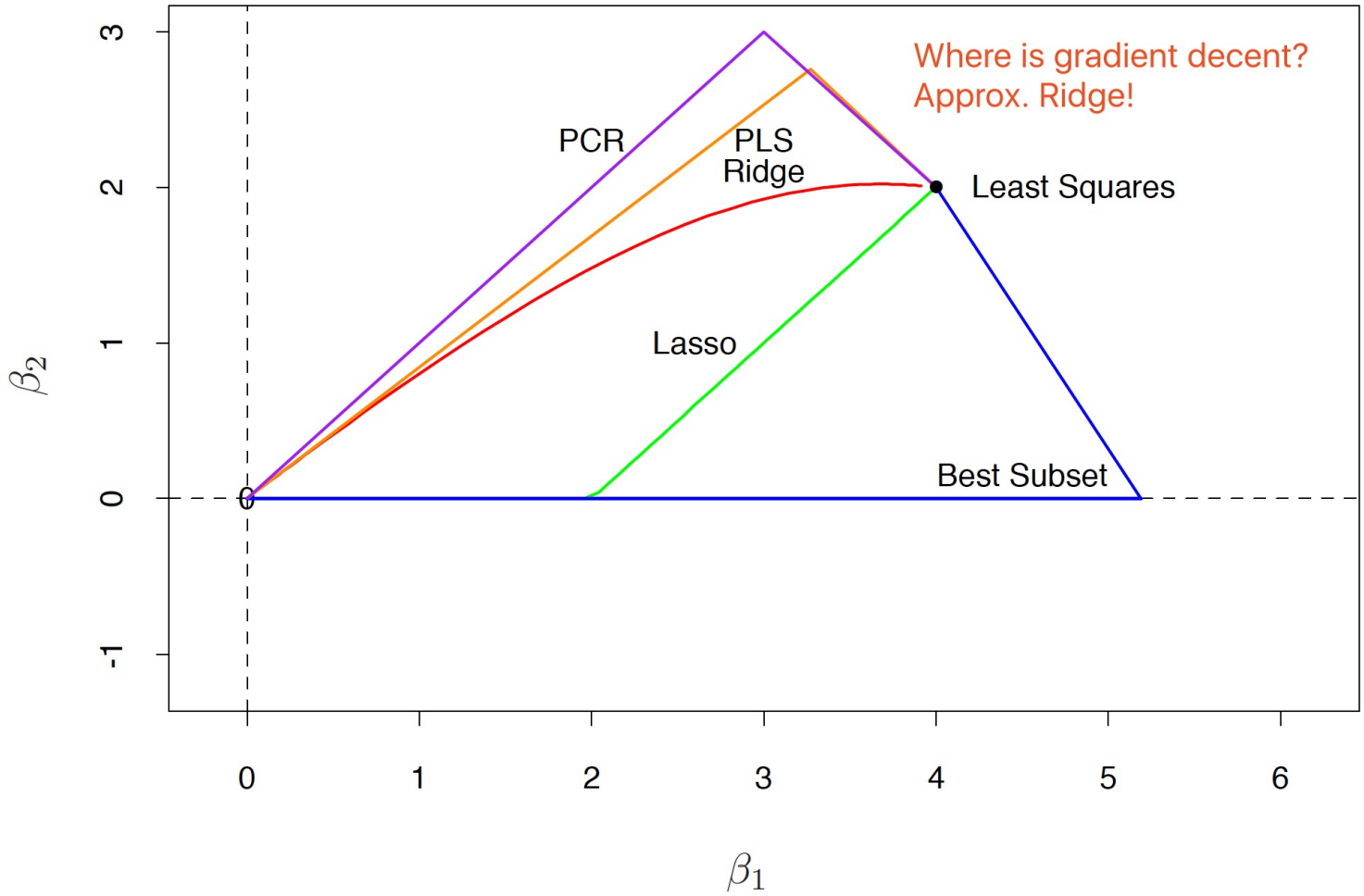
Abstract

The statistical perspective on boosting algorithms focuses on optimization, drawing parallels with maximum likelihood estimation for logistic regression. In this paper we present empirical evidence that raises questions about this view. Although the statistical perspective provides a theoretical framework within which it is possible to derive theorems and create new algorithms in general contexts, we show that there remain many unanswered important questions. Furthermore, we provide examples that reveal crucial flaws in the many practical suggestions and new methods that are derived from the statistical view. We perform carefully designed experiments using simple simulation models to illustrate some of these flaws and their practical consequences.

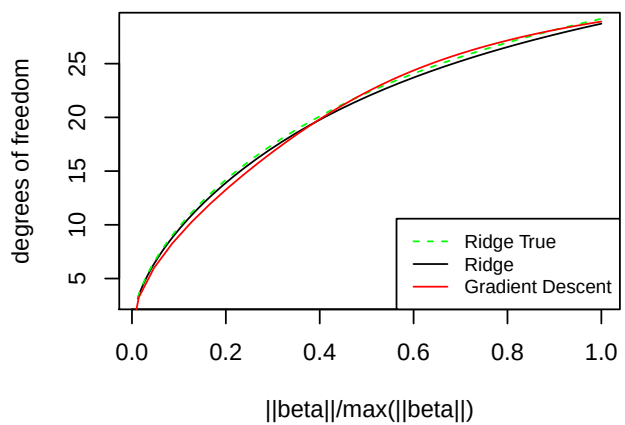
Keywords: boosting algorithms, LogitBoost, AdaBoost

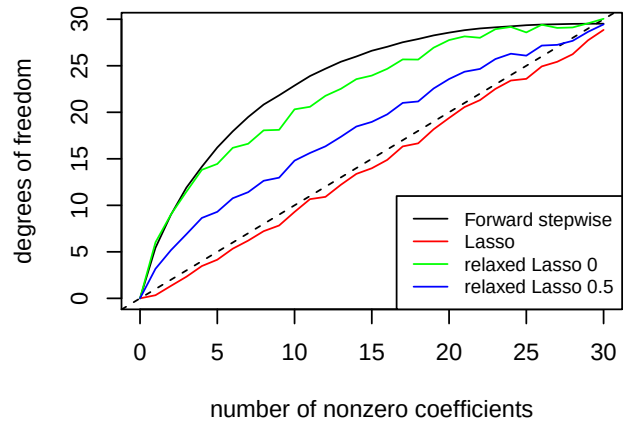
Figure: Ref link: `http://www.jmlr.org/papers/v9/mease08a.html`

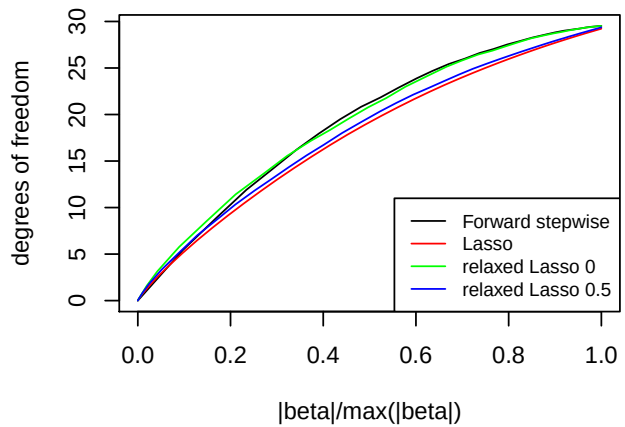
Discussion: Explicit and inexplicit regularization



Discussion: Degrees of freedom $df = \frac{1}{\sigma^2} \sum_{i=1}^{n} \mathrm{Cov}(y_i, \hat{y}_i)$







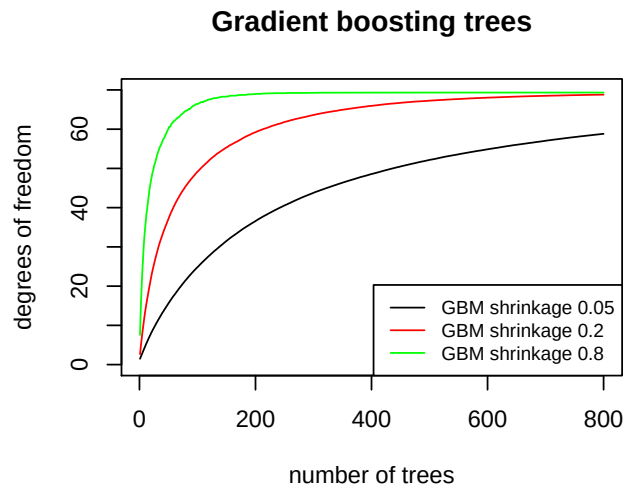


Figure: $n = 70$, $p = 30$, $y = X\beta + \epsilon$, $x_{ij} \sim \mathcal{N}(0, 1)$, SNR = 3:7. See my source code "df_demo.html" on Canvas

Discussion: Bias-Variance Tradeoff for Gradient Boosting

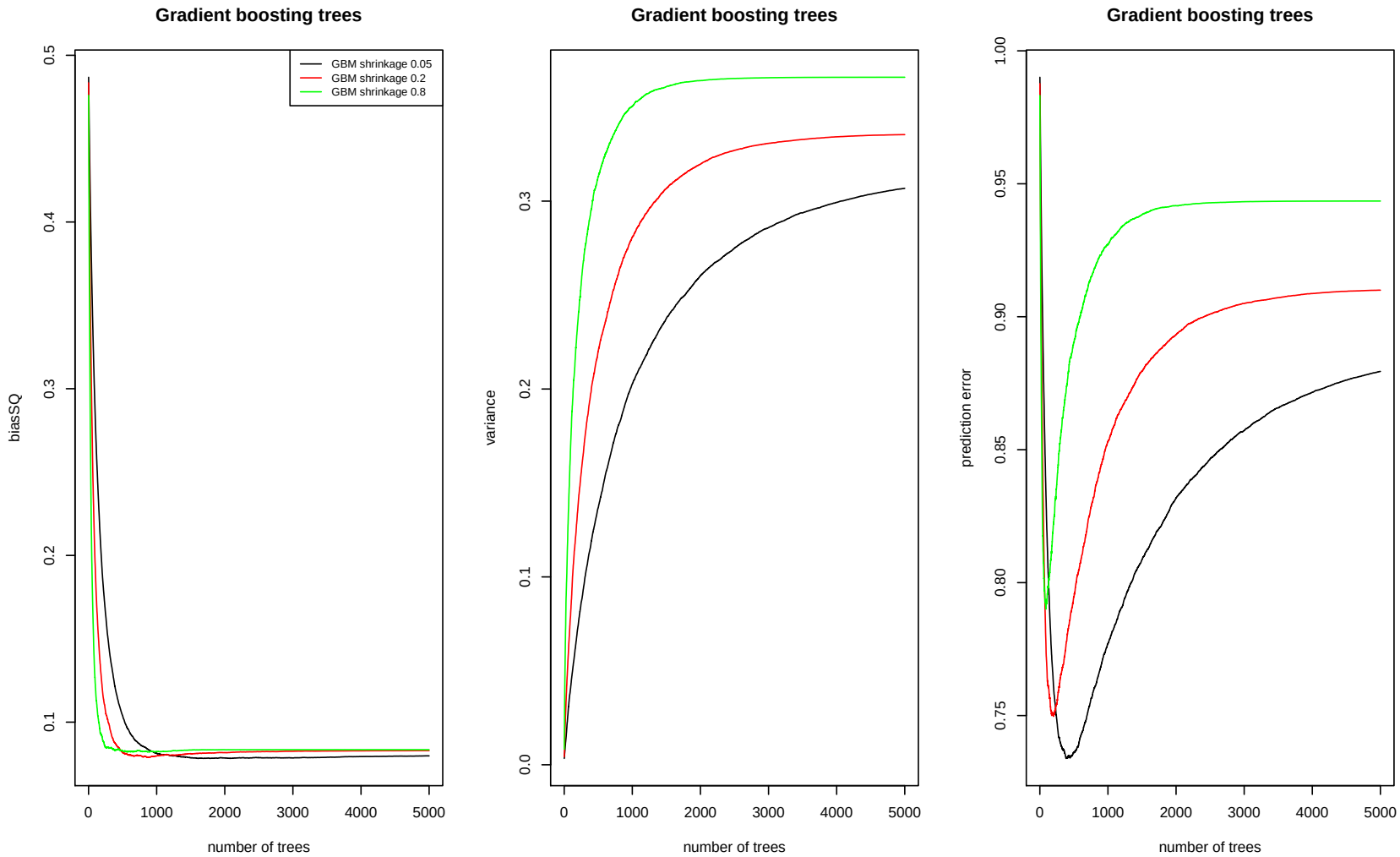


Figure: $n = 100$, $p = 50$, $y = X\beta + \epsilon$, $x_{ij} \sim \mathcal{N}(0, 1)$, SNR = 1:1. See my source code "Bias_Variance_Boosting.html" on Canvas.

Discussion: Bias-Variance Tradeoff for Gradient Boosting

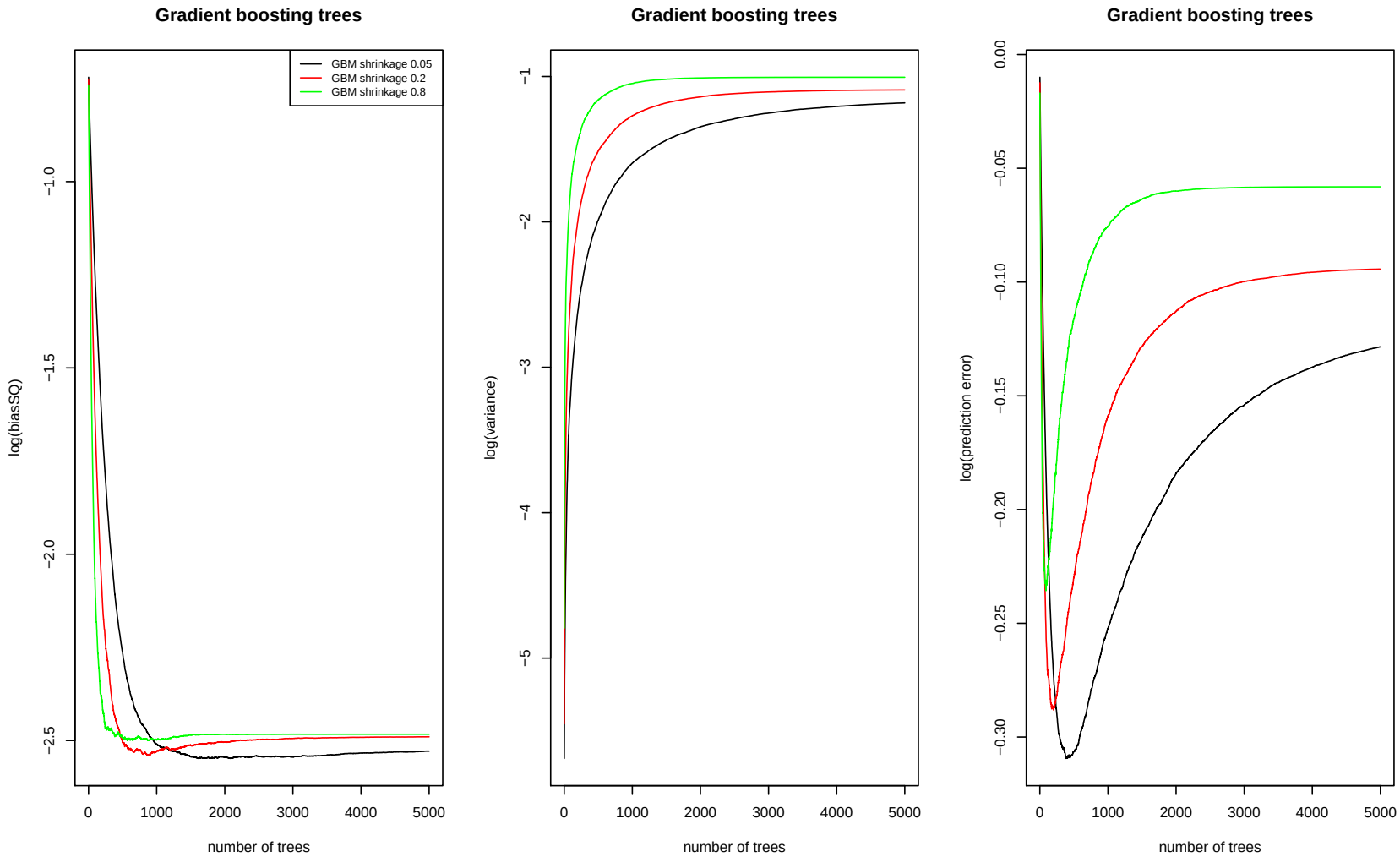


Figure: Log scale. $n = 100$, $p = 50$, $y = X\beta + \epsilon$, $x_{ij} \sim \mathcal{N}(0, 1)$, SNR = 1:1. See my source code "Bias_Variance_Boosting.html" on Canvas.

Discussion: Bias-Variance Tradeoff for Gradient Boosting

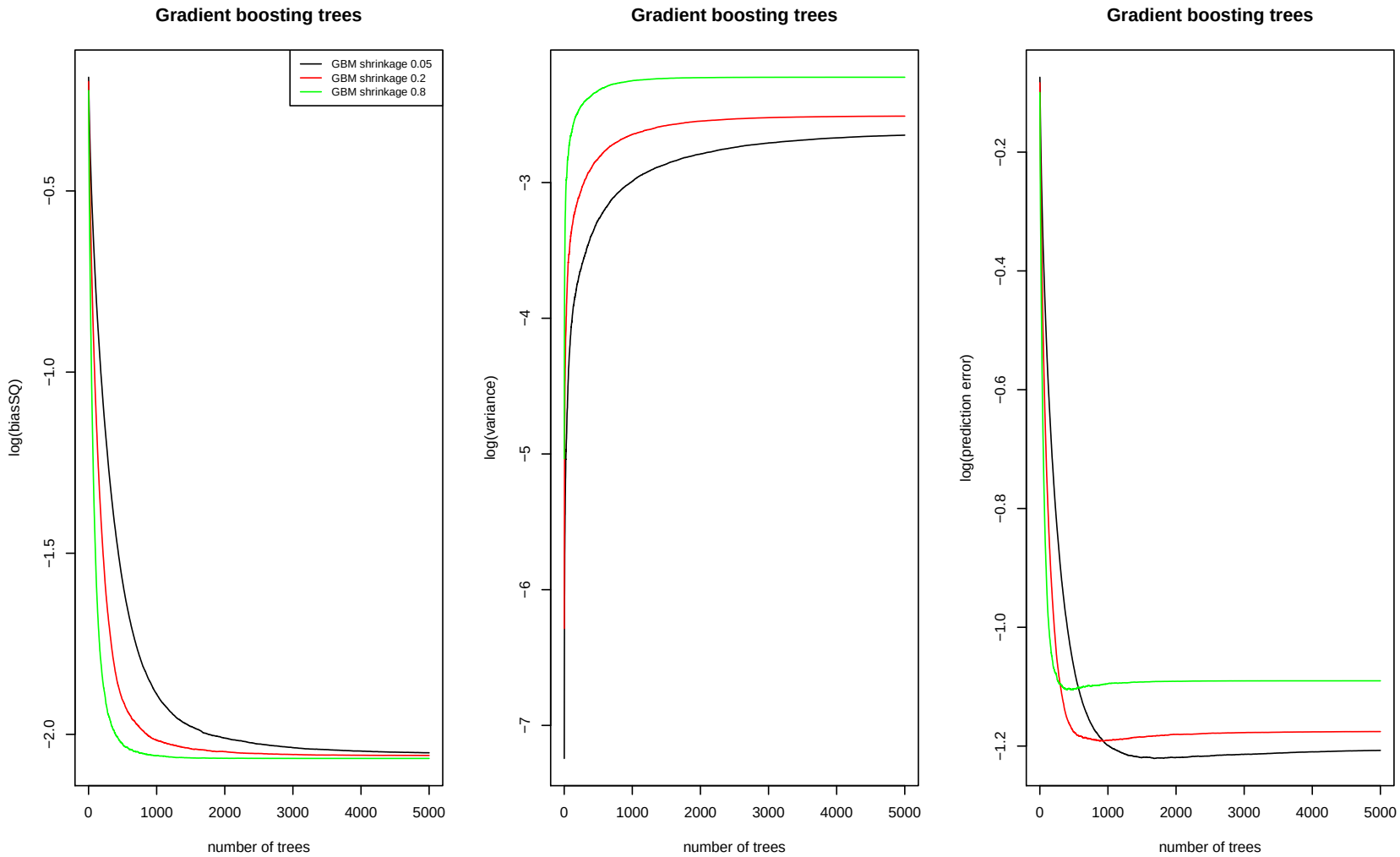


Figure: Log scale. $n = 100$, $p = 50$, $y = X\beta + \epsilon$, $x_{ij} \sim \mathcal{N}(0, 1)$, SNR = 9:1. See my source code "Bias_Variance_Boosting.html" on Canvas.

Stochastic approximation

A STOCHASTIC APPROXIMATION METHOD[1]

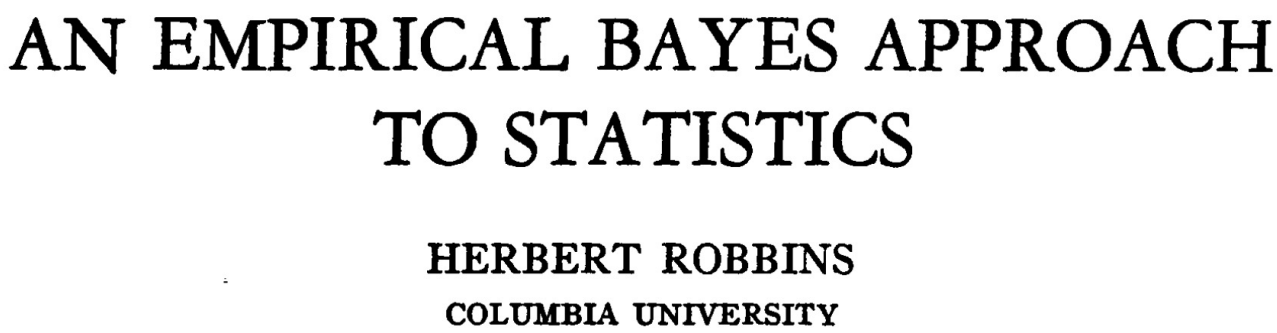
By Herbert Robbins and Sutton Monro

University of North Carolina

1. Summary. Let $M(x)$ denote the expected value at level x of the response to a certain experiment. $M(x)$ is assumed to be a monotone function of x but is unknown to the experimenter, and it is desired to find the solution $x = \theta$ of the equation $M(x) = \alpha$, where α is a given constant. We give a method for making successive experiments at levels $x_1, x_2, \cdots$ in such a way that x_n will tend to θ in probability.

Figure: Ann. Math. Statist. Volume 22, Number 3 (1951), 400-407. https://projecteuclid.org/euclid.aoms/1177729586.

Empirical Bayes



AN EMPIRICAL BAYES APPROACH
TO STATISTICS

HERBERT ROBBINS
COLUMBIA UNIVERSITY

Figure: Proc. Third Berkeley Symp. on Math. Statist. and Prob., Vol. 1 (Univ. of Calif. Press, 1956), 157-163
https://projecteuclid.org/euclid.bsmsp/1200501653.

Leo Breiman
Statistics Department, University of California, Berkeley, CA 94305;
e-mail: leo@stat.berkeley.edu

Reflections After Refereeing Papers for NIPS

Our fields would be better off with far fewer theorems, less emphasis on faddish stuff, and much more scientific inquiry and engineering. But the latter requires real thinking.

For instance, there are many important questions regarding neural networks which are largely unanswered. There seem to be conflicting stories regarding the following issues:

- Why don't heavily parameterized neural networks overfit the data?
- What is the effective number of parameters?
- Why doesn't backpropagation head for a poor local minima?
- When should one stop the backpropagation and use the current parameters?

It makes research more interesting to know that there is no one universally best method. What is best is data dependent. Sometimes "least glamorous" methods such as nearest neighbor are best. We need to learn more about what works best where. But emphasis on theory often distracts us from doing good engineering and living with the data.

The Mathematics of Generalization, Ed. David Wolpert, SFI Studies in the Sciences of Complexity, Proc. Vol XX, Addison-Wesley, 1995

The Annals of Statistics
1998, Vol. 26, No. 3, 801–849

ARCING CLASSIFIERS[1]

BY LEO BREIMAN

University of California, Berkeley

Recent work has shown that combining multiple versions of unstable classifiers such as trees or neural nets results in reduced test set error. One of the more effective is bagging. Here, modified training sets are formed by resampling from the original training set, classifiers constructed using these training sets and then combined by voting. Freund and Schapire propose an algorithm the basis of which is to adaptively resample and combine (hence the acronym "arcing") so that the weights in the resampling are increased for those cases most often misclassified and the combining is done by weighted voting. Arcing is more successful than bagging in test set error reduction. We explore two arcing algorithms, compare them to each other and to bagging, and try to understand how arcing works. We introduce the definitions of bias and variance for a classifier as components of the test set error. Unstable classifiers can have low bias on a large range of data sets. Their problem is high variance. Combining multiple versions either through bagging or arcing reduces variance significantly.

Figure: Ref: `https://projecteuclid.org/euclid.aos/1024691079`.