Boosting: more than an ensemble method for prediction

Peter Bühlmann

ETH Zürich

1. Historically: Boosting is about multiple predictions

Data: $(X_1, Y_1), \ldots, (X_n, Y_n)$ (i.i.d. or stationary),
predictor variables $X_i \in \mathbb{R}^p$
response variables $Y_i \in \mathbb{R}$ or $Y_i \in \{0, 1, \ldots, J-1\}$

Aim: estimation of function $f(\cdot) : \mathbb{R}^p \to \mathbb{R}$, e.g.
$f(x) = \mathbf{E}[Y|X = x]$ or $f(x) = \mathbb{P}[Y = 1|X = x]$ with $Y \in \{0, 1\}$
or distribution of survival time Y given X depends on some function $f(X)$ only

"historical" view (for classification):

Boosting is a multiple predictions (estimation) & combination method

Base procedure:

$$\text{data} \xrightarrow{\text{algorithm A}} \hat{\theta}(\cdot) \text{ (a function estimate)}$$

e.g.: simple linear regression, tree, MARS, “classical” smoothing, neural nets, ...

Generating multiple predictions:

$$\text{weighted data 1} \xrightarrow{\text{algorithm A}} \hat{\theta}_1(\cdot)$$

$$\text{weighted data 2} \xrightarrow{\text{algorithm A}} \hat{\theta}_2(\cdot)$$

$$\ldots \qquad \ldots$$

$$\text{weighted data M} \xrightarrow{\text{algorithm A}} \hat{\theta}_M(\cdot)$$

Aggregation: $\hat{f}_A(\cdot) = \sum_{m=1}^{M} a_m \hat{\theta}_m(\cdot)$

data weights? averaging weights a_m?

classification of 2 lymph nodal status in breast cancer using gene expressions from microarray data:

$n = 33$, $p = 7129$ (for CART: gene-preselection, reducing to $p = 50$)

method	test set error	gain over CART
CART	22.5%	–
LogitBoost with trees	16.3%	28%
LogitBoost with bagged trees	12.2%	46%

this kind of boosting: mainly prediction, not much interpretation

2. Boosting algorithms

AdaBoost proposed for classification by Freund & Schapire (1996)

data weights (rough original idea): large weights to previously heavily misclassified instances (sequential algorithm)

averaging weights a_m: large if in-sample performance in mth round was good

Why should this be good?

Why should this be good?

some common answers 5 years ago ...

because

- it works so well for prediction (which is quite true)
- it concentrates on the “hard cases” (so what?)
- AdaBoost almost never overfits the data no matter how many iterations it is run (not true)

A better explanation

Breiman (1998/99): AdaBoost is functional gradient descent (FGD) procedure

aim: find $f^*(\cdot) = \text{argmin}_{f(\cdot)} \mathbf{E}[\rho(Y, f(X))]$

e.g. for $\rho(y, f) = |y - f|^2 \rightsquigarrow f^*(x) = \mathbf{E}[Y|X = x]$

FGD solution: consider empirical risk $n^{-1} \sum_{i=1}^{n} \rho(Y_i, f(X_i))$ and

do iterative steepest descent in function space

2.1. Generic FGD algorithm

Step 1. $\hat{f}_0 \equiv 0$; set $m = 0$.

Step 2. Increase m by 1. Compute negative gradient $-\frac{\partial}{\partial f}\rho(Y, f)$
and evaluate at $f = \hat{f}_{m-1}(X_i) = U_i \ (i = 1, \ldots, n)$

Step 3. Fit negative gradient vector $U_1, \ldots, U_n$ by base procedure

$$(X_i, U_i)_{i=1}^n \xrightarrow{\text{algorithm A}} \quad \hat{\theta}_m(\cdot)$$

e.g. $\hat{\theta}_m$ fitted by (weighted) least squares
i.e. $\hat{\theta}_m(\cdot)$ is an approximation of the negative gradient vector

Step 4. Up-date $\hat{f}_m = \hat{f}_{m-1}(\cdot) + \nu s_m \cdot \hat{\theta}_m(\cdot)$
$s_m = \text{argmin}_s n^{-1} \sum_{i=1}^n \rho(Y_i, \hat{f}_{m-1}(X_i) + s \cdot \hat{\theta}_m(X_i))$ and $0 < \nu \leq 1$
i.e. proceed along an estimate of the negative gradient vector

Step 5. Iterate Steps 2-4 until $m = m_{stop}$ for some stopping iteration m_{stop}

Why “functional gradient”?

Alternative formulation in function space:

empirical risk functional: $C(f) = n^{-1} \sum_{i=1}^{n} \rho(Y_i, f(X_i))$

inner product: $\langle f, g \rangle = n^{-1} \sum_{i=1}^{n} f(X_i) g(X_i)$

negative Gateaux derivative:

$$-dC(f)(x) = \frac{\partial}{\partial \alpha} C(f + \alpha 1_x)|_{\alpha=0}, \rightsquigarrow -dC(\hat{f}_{m-1})(X_i) = U_i$$

if $U_1, \ldots, U_n$ are fitted by least squares:

equivalent to maximize $\langle -dC(f_m), \theta \rangle$ w.r.t. $\theta(\cdot)$ (if$\|\theta\| = 1$)

(over all possible $\theta(\cdot)$'s from the base procedure)

i.e: $\hat{\theta}_m(\cdot)$ is the best approximation (most parallel)

to the negative gradient $-dC(f_m)$

By definition: FGD yields additive combination of base procedure fits

$$\nu \sum_{m=1}^{m_{stop}} s_m \hat{\theta}_m(\cdot)$$

Breiman (1998):

FGD with $\rho(y, f) = \exp((2y-1) \cdot f)$ for binary classification yields the

AdaBoost algorithm

(great result!)

Remark: FGD can not be represented as some explicit estimation function(al):

$$\hat{f}_m(\cdot) \neq \text{argmin}_{f \in \mathcal{F}} n^{-1} \sum_{i=1}^{n} \rho(Y_i, f(X_i)) \quad \text{for some function class } \mathcal{F}$$

$\rightsquigarrow$ FGD is mathematically more difficult to analyze but generically applicable (as an algorithm!) in very complex models

2.2. L_2Boosting

(see also Friedman, 2001)

loss function $\rho(y, f) = |y - f|^2$

population minimizer: $f^*(x) = \mathbf{E}[Y|X = x]$

FGD with base procedure $\hat{\theta}(\cdot)$: repeated fitting of residuals

$$m = 1: \ (X_i, Y_i)_{i=1}^n \rightsquigarrow \hat{\theta}_1(\cdot),\ \hat{f}_1 = \nu\hat{\theta}_1 \quad \rightsquigarrow \text{ resid. } U_i = Y_i - \hat{f}_1(X_i)$$

$$m = 2: \ (X_i, U_i)_{i=1}^n \rightsquigarrow \hat{\theta}_2(\cdot),\ \hat{f}_2 = \hat{f}_1 + \nu\hat{\theta}_2 \quad \rightsquigarrow \text{ resid. } U_i = Y_i - \hat{f}_2(X_i)$$

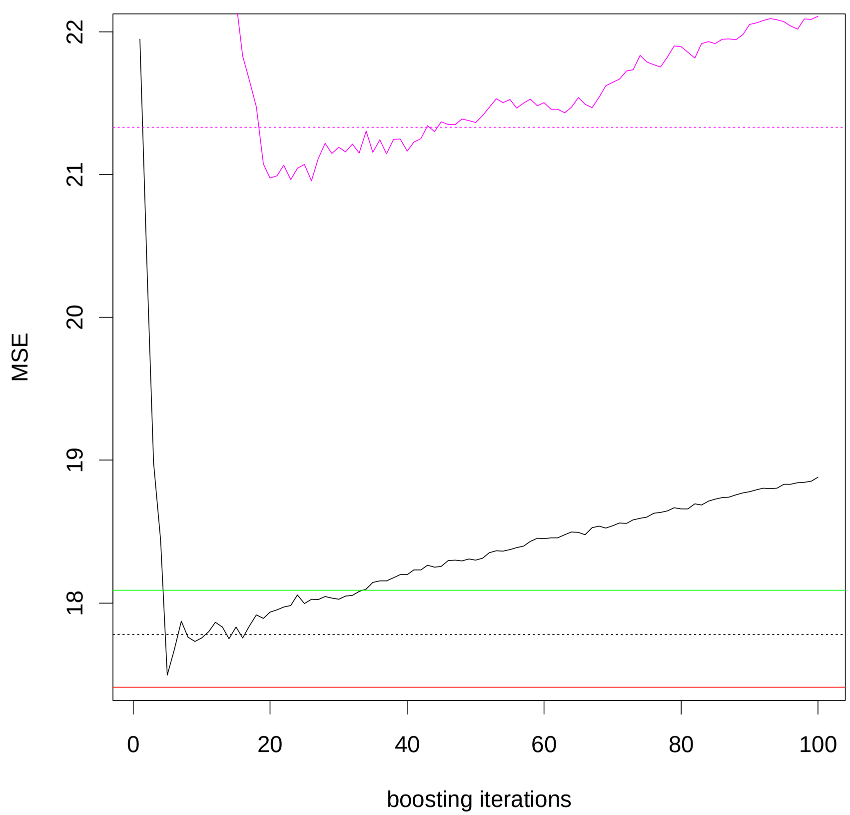
$$\ldots \qquad \ldots$$

$\hat{f}_{m_{stop}}(\cdot) = \nu \sum_{m=1}^{m_{stop}} \hat{\theta}_m(\cdot)$ (stagewise greedy fitting of residuals)

Tukey (1977): twicing for $m_{stop} = 2$ and $\nu = 1$

Any gain over classical methods? (for additive modeling)

Ozone data: n=300, p=8



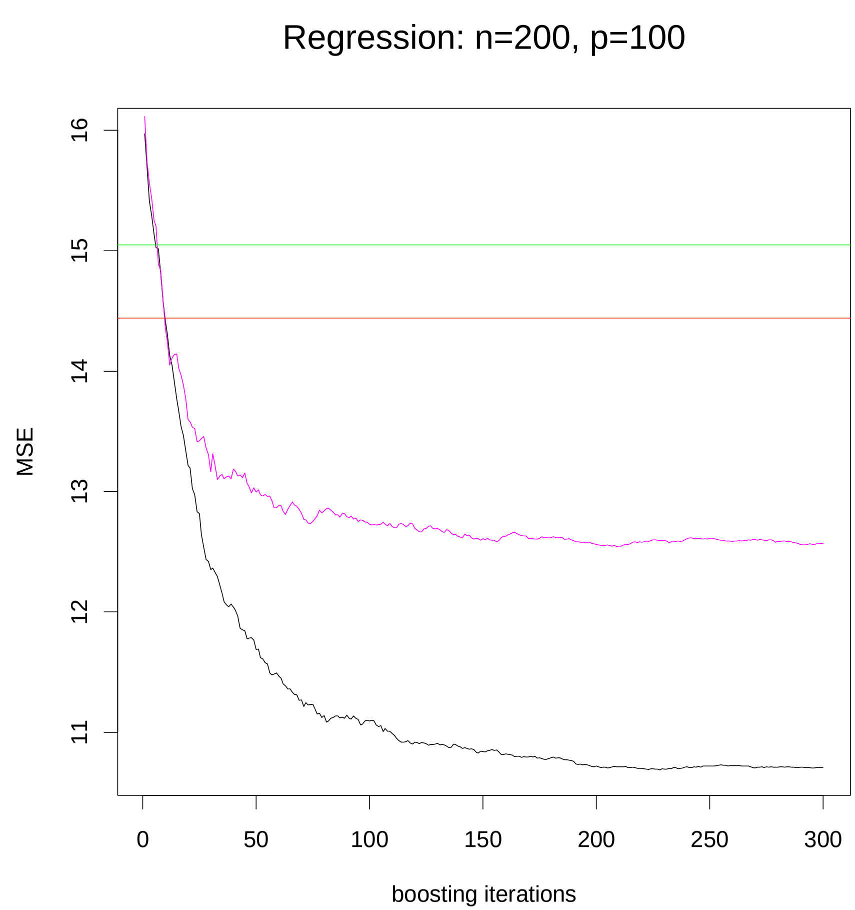
$n = 300,\ p = 8$

- magenta: L_2Boosting with stumps
  (horiz. line = cross-validated stopping)
- black: L_2Boosting with componentwise smoothing spline
  (horiz. line = cross-validated stopping)
  i.e: smoothing spline fi tting against the selected predictor which reduces RSS most
- green: MARS restricted to additive modeling
- red: additive model using backfi tting

L_2Boosting with stumps or comp. smoothing splines also yields additive model:

$$\sum_{m=0}^{m_s top} \hat{\theta}_m(x^{(\hat{\mathcal{S}}_m)}) = \hat{g}_1(x^{(1)}) + \ldots + \hat{g}_p(x^{(p)})$$

Simulated data: non-additive regression function, $n = 200, p = 100$



- magenta: L_2Boosting with stumps
- black: L_2Boosting with componentwise
- green: MARS restricted to additive modeling
- red: additive model using backfi tting and fwd. var. selection

similar for classifi cation

3. Structured models and choosing the base procedure

have just seen the

Componentwise smoothing spline base procedure

smoothes the reponse against the one predictor variable which reduces RSS most

we keep the degrees of freedom fixed for all candidate predictors, e.g. d.f. = 2.5

$\leadsto$ L_2Boosting yields an additive model fit, including variable selection

Componentwise linear least squares

simple linear OLS against the one predictor variable which reduces RSS most

$$\hat{\theta}(x) = \hat{\beta}_{\hat{\mathcal{S}}} x^{(\hat{\mathcal{S}})}, \ \hat{\beta}_j = \sum_{i=1}^{n} Y_i X_i^{(j)} / \sum_{i=1}^{n} (X_i^{(j)})^2, \ \hat{\mathcal{S}} = \arg\min_j \sum_{i=1}^{n} (Y_i - \hat{\beta}_j X_i^{(j)})^2$$

first round of estimation: selected predictor variable $X^{(\hat{\mathcal{S}}_1)}$ (e.g. $= X^{(3)}$)
corresponding $\hat{\beta}_{\hat{\mathcal{S}}_1} \rightsquigarrow$ fitted function $\hat{f}_1(x)$

second round of estimation: selected predictor variable $X^{(\hat{\mathcal{S}}_2)}$ (e.g.$= X^{(21)}$)
corresponding $\hat{\beta}_{\hat{\mathcal{S}}_2} \rightsquigarrow$ fitted function $\hat{f}_2(x)$

etc.

L_2Boosting: $\hat{f}_m(x) = \hat{f}_{m-1}(x) + \nu \cdot \hat{\theta}(x)$

$\rightsquigarrow$ L_2Boosting yields linear model fit, including variable selection,

i.e. structured model fit

for $\nu = 1$, this is known as

Matching Pursuit (Mallat and Zhang, 1993)

Weak greedy algorithm (deVore & Temlyakov, 1997)

a version of Boosting (Schapire, 1992; Freund & Schapire, 1996)

Gauss-Southwell algorithm

C.F. Gauss in 1803

“Princeps Mathematicorum”

R.V. Southwell in 1933

Professor in engineering, Oxford

binary lymph node classification in breast cancer using gene expressions:

a high noise problem

$n = 49$ samples, $p = 7129$ gene expressions

	L_2Boosting	FPLR	Pelora	1-NN	DLDA	SVM
CV-misclassif.err.	17.7%	35.25%	27.8%	43.25%	36.12%	36.88%

multivariate gene selection (left three methods) — best 200 genes from Wilcox. (right three methods)

L_2Boosting selected 42 out of $p = 7129$ genes

for this data-set: not good prediction, with any of the methods

but L_2Boosting may be a reasonable(?) multivariate gene selection method

Pairwise smoothing splines

smoothes response against the pair of predictor variables which reduces RSS most

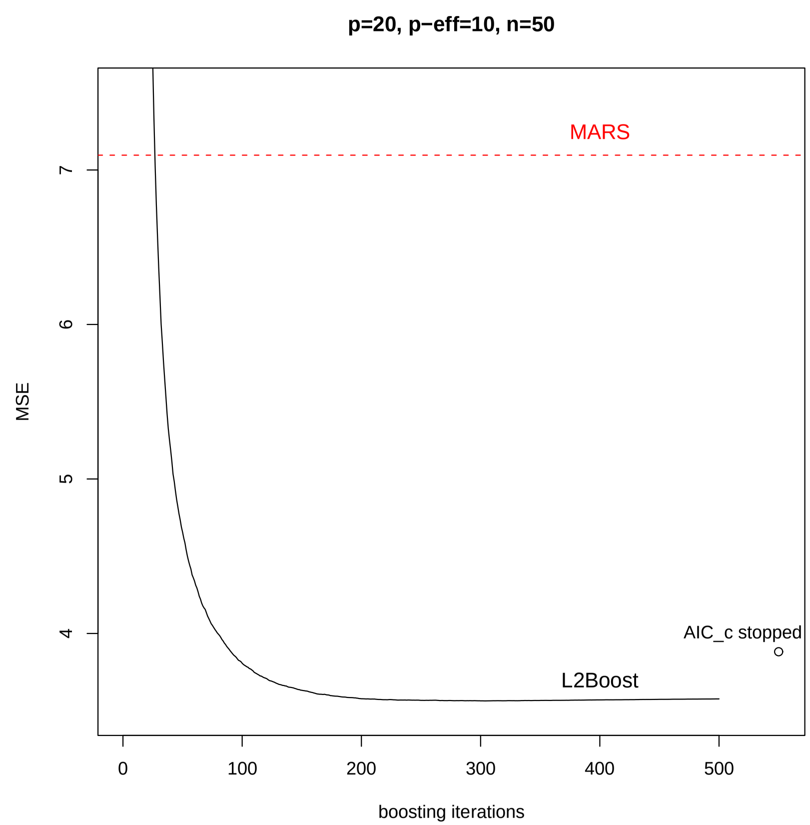
we keep the degrees of freedom fixed for all candidate pairs, e.g. d.f. = 2.5

$\rightsquigarrow L_2$Boosting yields a nonparametric interaction model, including variable selection

Example: degree 2 nonparametric interaction modelling

Friedman #1 model:

$$Y = 10\sin(\pi X_1 X_2) + 20(X_3 - 0.5)^2 + 10X_4 + 5X_5 + \mathcal{N}(0,1),\ X = (X_1, \ldots, X_{20}) \sim \text{Unif.}([0,1]^{20})$$



L_2Boosting with pairwise splines

sample size $n = 50$

$p = 20$, effective $p_{eff} = 5$

Regression trees

stumps (2 terminal nodes): L_2Boosting fits an additive model

trees with d terminal nodes: L_2Boosting fits an interaction model of degree $d - 2$

The low variance high bias “principle”

once we have decided about some structural properties

choose base procedure with low variance but potentially large estimation bias
bias can be reduced by further boosting iterations (which will increase variance)

example: low degrees of freedom in componentwise smoothing splines for additive modeling

a justification will be given later

4. More on L_2Boosting

L_2Boosting for linear models

use componentwise linear least squares base procedure

L_2Boosting converges to a least squares solution as boosting iterations $m \to \infty$
(the unique LS solution if design has full rank $p \leq n$)

when stopping early:

- it does variable selection
- coefficient estimates are typically shrunken version of LS

$\rightsquigarrow$ "similar to" the Lasso

Connections to Lasso (for linear models):

Efron, Hastie, Johnstone, Tibshirani (2004): for special design matrices,

iterations of L_2Boosting with "infinitesimally" small ν
yield all Lasso solutions when varying λ

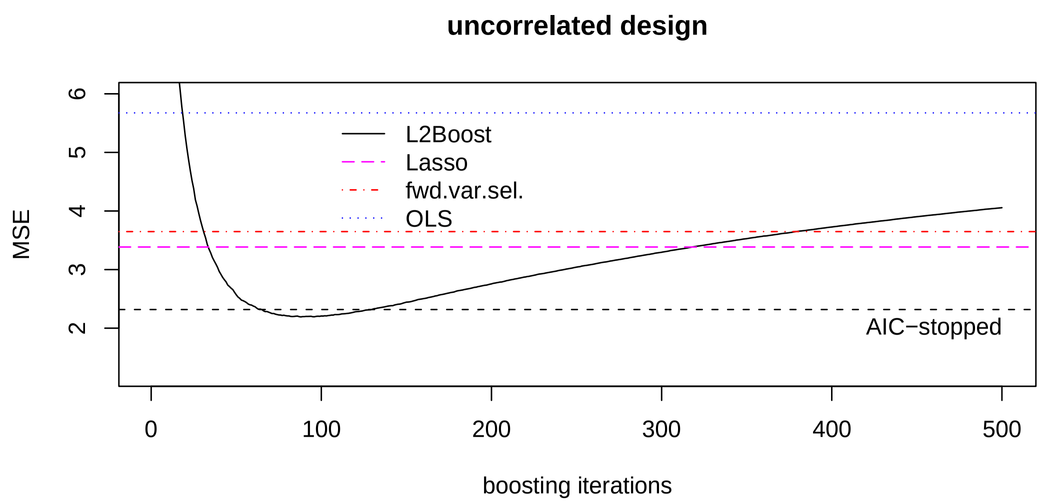
$\rightsquigarrow$ computationally interesting to produce all Lasso solutions in one sweep of boosting

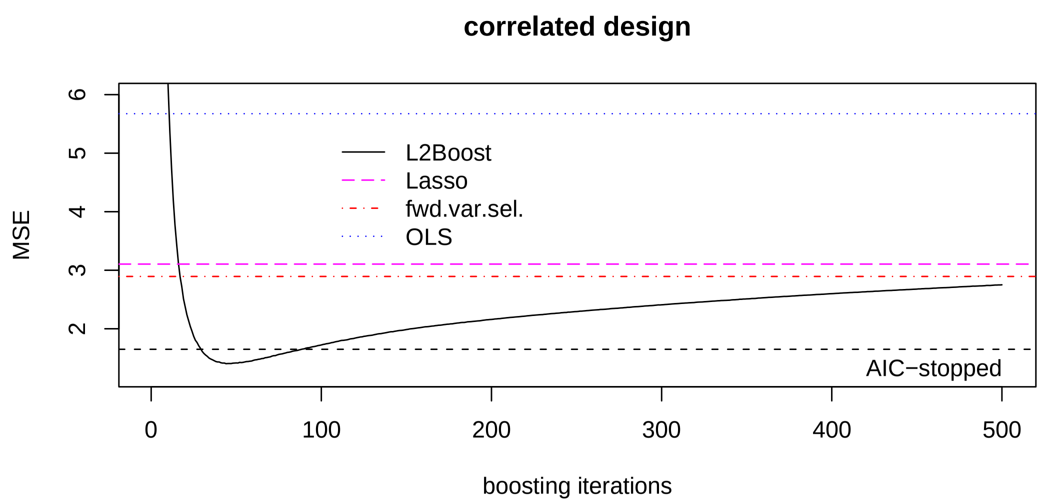
Least Angle Regression LARS (Efron et al., 2004) is computationally even more clever and efficient than L_2Boosting

Zhao and Yu (2005): in "general", when adding some backward step the solutions from Lasso and Boosting "coincide"

greedy (plus backward steps) and convex optimization are surprisingly similar

$$p = 10, p_{eff} = 3, n = 20$$





binary lymph node classification using gene expressions

$n = 49$ samples, $p = 7129$ gene expressions

	L_2Boosting	FPLR	Pelora	1-NN	DLDA	SVM
CV-misclassif.err.	17.7%	35.25%	27.8%	43.25%	36.12%	36.88%
	Lasso					
CV-misclassif.err.	21.2%					

multivariate gene selection — best 200 genes from Wilcox.

L_2Boosting selected 42 out of $p = 7129$ genes

Lasso selected 15 genes

how well can we do?

statistically consistent for very high-dimensional, sparse linear models

$$Y_i = \beta_0 + \sum_{j=1}^{p} \beta_j X_i^{(j)} + \varepsilon_i \; (i = 1, \ldots, n), \;\; p \gg n$$

Theorem (PB, 2004)

L_2Boosting with comp. linear LS is consistent (for suitable number of boosting iterations) if:

- $p_n = O(\exp(Cn^{1-\xi}))$ $(0 < \xi < 1)$ (high-dimensional)
  essentially exponentially many variables relative to n
- $\sup_n \sum_{j=1}^{p_n} |\beta_{j,n}| < \infty$ ℓ^1-sparseness of true function

i.e. for suitable, slowly growing $m = m_n$:

$$\mathbf{E}_X |\hat{f}_{m_n,n}(X) - f_n(X)|^2 = o_P(1) \; (n \to \infty)$$

"no" assumptions about the predictor variables/design matrix analogous results also for

- multivariate regression
- vector autoregressive time series

(Lutz & PB, 2005)

4.1. Degrees of freedom for boosting

(PB, 2004)

the only tuning parameter: number of boosting iterations

could use cross-validation $\rightsquigarrow$ works reasonably well

alternatively: use AIC, BIC or gMDL as model selection criteria which involve

degrees of freedom of boosting

hat-matrix of comp.wise linear LS base procedure:

$\mathcal{H}^{(j)} : (Y_1, \ldots, Y_n) \mapsto (\hat{Y}_1, \ldots, \hat{Y}_n)$ when using the jth predictor variable only:

$$\mathcal{H}^{(j)} = \mathbf{X}^{(j)} (\mathbf{X}^{(j)})^T / \|\mathbf{X}^{(j)}\|^2$$

L_2Boosting hat-matrix:

$$\begin{aligned} \mathcal{B}_m &= \mathcal{B}_{m-1} + \nu \cdot \mathcal{H}^{(\hat{\mathcal{S}}_m)} (I - \mathcal{B}_{m-1}) \\ &= I - (I - \nu \cdot \underbrace{\mathcal{H}^{(\hat{\mathcal{S}}_m)}}_{\text{selected in } m\text{th iter.}}) (I - \nu \cdot \mathcal{H}^{(\hat{\mathcal{S}}_{m-1})}) \cdots (I - \nu \cdot \mathcal{H}^{(\hat{\mathcal{S}}_1)}) \end{aligned}$$

degrees of freedom of boosting in iteration m:

$$d.f.(\mathcal{B}_m) = \text{trace}(\mathcal{B}_m)$$

$d.f.$ ignores the selection effect, i.e. "slightly" too small

("negligible" since we can allow for $o(\exp(n))$ candidate basis functions)

$d.f.$ is very different from the number of variables in the model

example: 3 (or more) correlated variables, $\nu = 1$

sequence of selected variables: 3,2,1,3,2,1 $\rightsquigarrow d.f.(\mathcal{B}_6) = 1.79 < 3$

sequence of selected variables: 1,2,3,2,3,1 $\rightsquigarrow d.f.(\mathcal{B}_6) = 1.54 < 3$

Stopping the boosting iterations

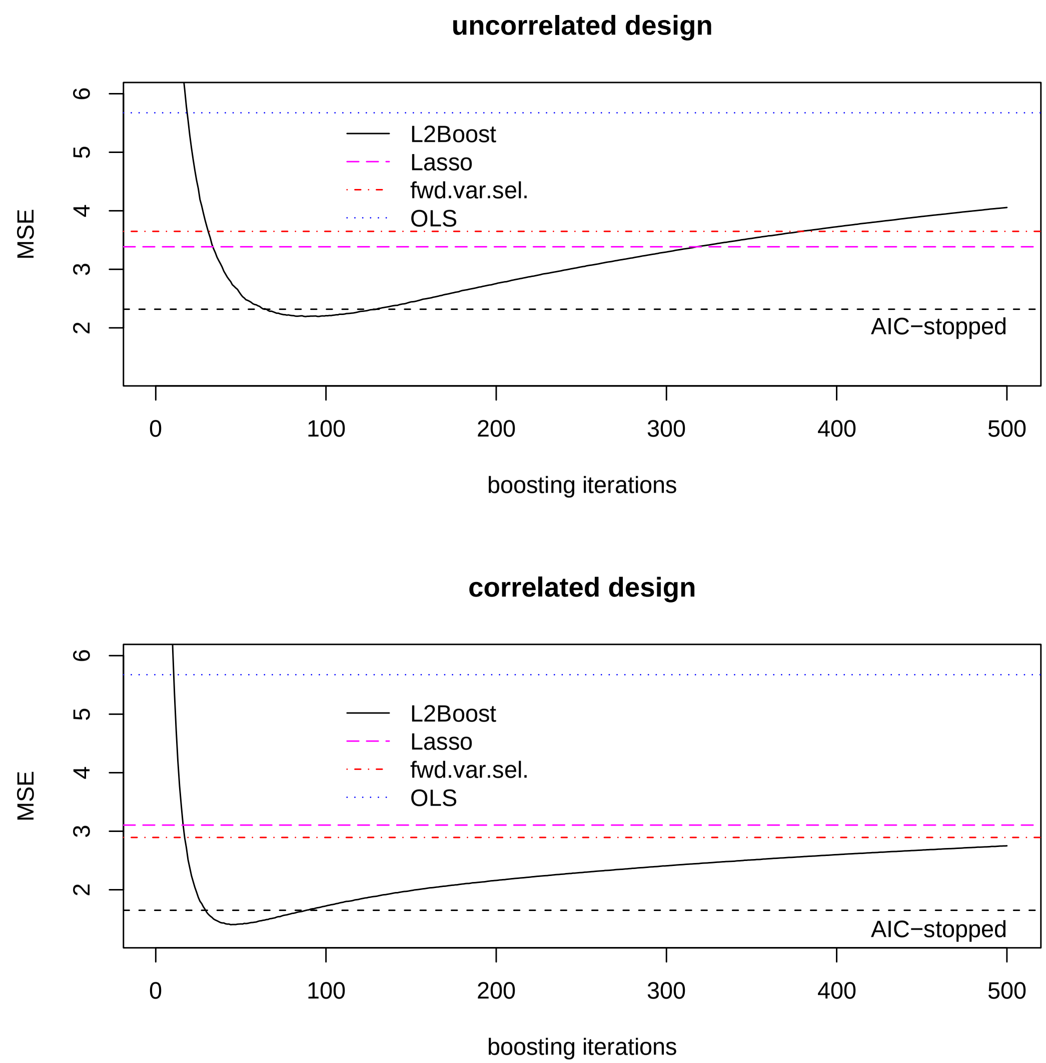
we often use the corrected AIC$_c$ criterion:

$$\text{AIC}_c(\mathcal{B}_m) = \log(RSS_m/n) + \frac{1 + \text{trace}(\mathcal{B}_m)/n}{1 - (\text{trace}(\mathcal{B}_m) + 2)/n}$$

estimate stopping iteration by

$$\hat{m}_{stop} = \text{argmin}_m \, \text{AIC}_c(\mathcal{B}_m)$$

$p = 10, p_{eff} = 3, n = 20$

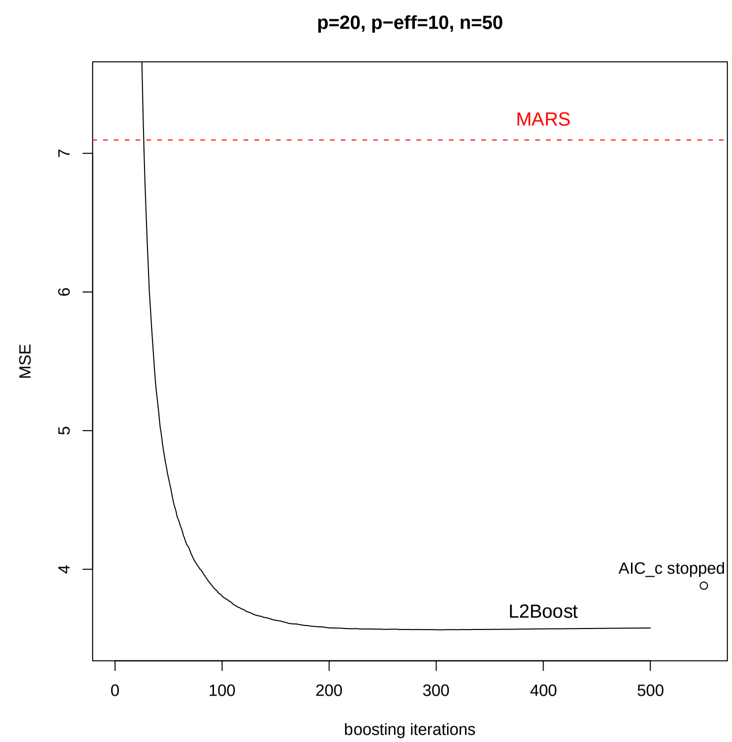


Analogously for nonparametric base procedures

hat-matrix $\mathcal{H}^{(\mathcal{S})}$ with a selected subset $\mathcal{S}$ of predictor variables

$$\mathcal{B}_m = I - (I - \nu \cdot \mathcal{H}^{(\hat{\mathcal{S}}_m)})(I - \nu \cdot \mathcal{H}^{(\hat{\mathcal{S}}_{m-1})}) \cdots (I - \nu \cdot \mathcal{H}^{(\hat{\mathcal{S}}_1)})$$

e.g. L_2Boosting with pairwise splines for nonparametric interaction modeling



More on degrees of freedom

example: L_2Boosting with componentwise smoothing splines for additive modeling

boosting hat-matrix $\mathcal{B}_m$:

since $\hat{f}(X_i) = \sum_{j=1}^{p} \hat{f}_j(X_i) \rightsquigarrow$ decompose

$$\mathcal{B}_m = \sum_{j=1}^{p} \underbrace{\mathcal{A}_m^{(j)}}_{\text{hat-matrix for } \hat{f}_j(\cdot)}$$

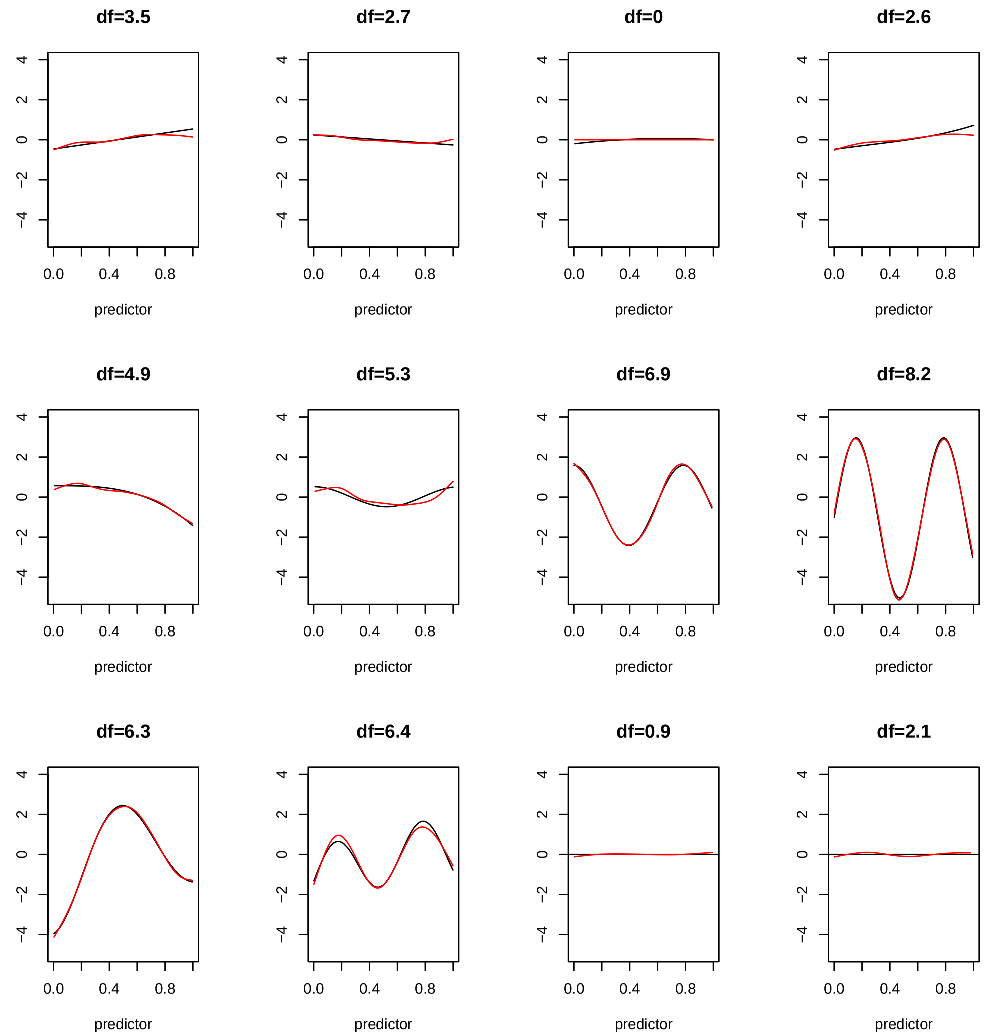
easy to compute recursively:

$$\mathcal{A}_m^{(j)} = \mathcal{A}_{m-1}^{(j)} + \delta_{j,\hat{\mathcal{S}}_m} \nu \cdot \mathcal{H}^{(\hat{\mathcal{S}}_m)}(I - \mathcal{B}_{m-1})$$

thus

$$\underbrace{d.f.}_{\text{trace}(\mathcal{B}_m)} = \sum_{j=1}^{p} \underbrace{d.f.^{(j)}}_{\text{trace}(\mathcal{A}_m^{(j)})}$$

$Y = \sum_{j=1}^{10} g_j(X^{(j)}) + \varepsilon,\ X \sim \text{Unif}[0,1]^{100};\ n = 200,\ p = 100,\ p_{eff} = 10$



L_2Boosting does a "very reasonable" assignment of degrees of freedom a very interesting way to search and estimate in high dimensions!

with classical methods (backfitting) for large p:

“infeasible” to do variable selection and variable amount of d.f.

L_2Boosting runs with one (!) tuning parameter for standard errors in additive modelling

$$s.e.(\hat{f}_j(X_i)) = \sqrt{\sigma_\varepsilon^2 (\underbrace{\mathcal{A}_m^{(j)}}_{\text{hat matrix for } j\text{th comp.}} (\mathcal{A}_m^{(j)})^T)_{ii}}$$

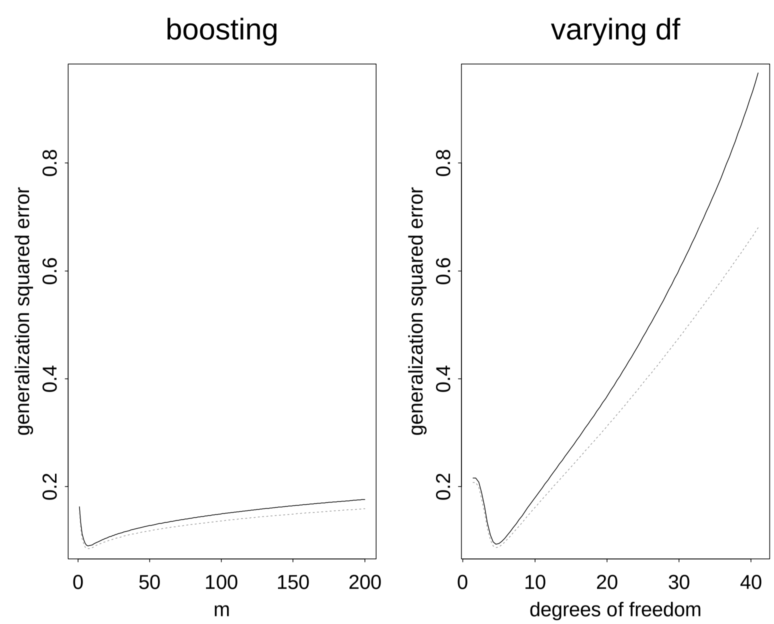
in our experience: seems quite OK

maybe slightly too small becuase we ignore the selection effect

for comparing $\underbrace{\text{“nested”}}_{\text{before variable selection}}$ models: use AIC, BIC, gMDL, etc.

4.2. The MSE curve and asymptotic optimality

toy example: L_2Boosting with smoothing spline for $p = 1$-dimensional predictor



sub-linear increase of MSE in Boosting

L_2Boosting quite resistant against overfitting; "easy to tune" consider (any) base procedure as operator:

$$\mathcal{H} : Y = (Y_1, \ldots, Y_n)' \overset{\text{base procedure}}{\longmapsto} \hat{Y} = (\hat{Y}_1, \ldots, \hat{Y}_n)'$$

L_2Boosting operator in iteration m:

$$\mathcal{B}_m = I - (I - \mathcal{H})^m$$

if $\mathcal{H}$ is strictly shrinking, i.e. $\|I - \mathcal{S}\| < 1$

$\leadsto$ L_2Boosting converges to identity I (fully saturated model)

$\leadsto$ need for early stopping

in case where $\mathcal{H}$ is a smoothing spline:

L_2Boosting does shrinkage in the same eigenspace as the smoothing spline $\mathcal{H}$

eigenvalues of smoothing spline:

$$\lambda_1 = \lambda_2 = 1,\ \ 0 < \lambda_i < 1\,(i = 3, \ldots, n)$$

eigenvalues of L_2Boosting:

$$ev_1 = 1, ev_2 = 1,\ \ 0 < ev_i = 1 - (1 - \lambda_i)^m\ (i = 3, \ldots, n)$$

change these eigenvalues (spectrum) by varying the iteration number m

$\rightsquigarrow$ tuning via m leads to sublinear increase of MSE w.r.t. m

Theorem (PB & Yu, 2003)

L_2Boosting with smoothing splines having any fixed deg. of freedom ("low variance")

- when stopping iterations suitably, it achieves asymptotically the optimal minimax MSE rate (over Sobolev space)
- it adapts to unknown greater smoothness of underlying function (adaptation to optimal MSE rate)

  e.g. L_2Boost with cubic smoothing splines automatically achieves faster rate than $O(n^{-4/5})$ if underlying function is smooth

Summary about (L_2-)Boosting

- need for early stopping
  "obvious" but has been still debated in 2000
- choose the base procedure to obtain the qualitative model fit of your own "choice"
  having decided on structure: use low variance and high estimation bias "principle"
- reasonable degrees of freedom and hat-matrices can be easily derived
  for L_2Boosting with base proc. involving $\underbrace{\text{linear fitting after selection of variables}}_{\text{non-linear boosting algo.}}$

all this applies also to boosting with other loss functions

5. Boosting for binary classification

binary lymph node classification using gene expressions: data
$(X_i, Y_i),\ X_i \in \mathbb{R}^{7129},\ Y_i \in \{-1, 1\}$

Various loss functions

$\rho(y,f) = \log_2(1 + \exp(-yf))$: negative binomial log-likelihood
$f^*(x) = \log(\frac{p(x)}{1-p(x)})$

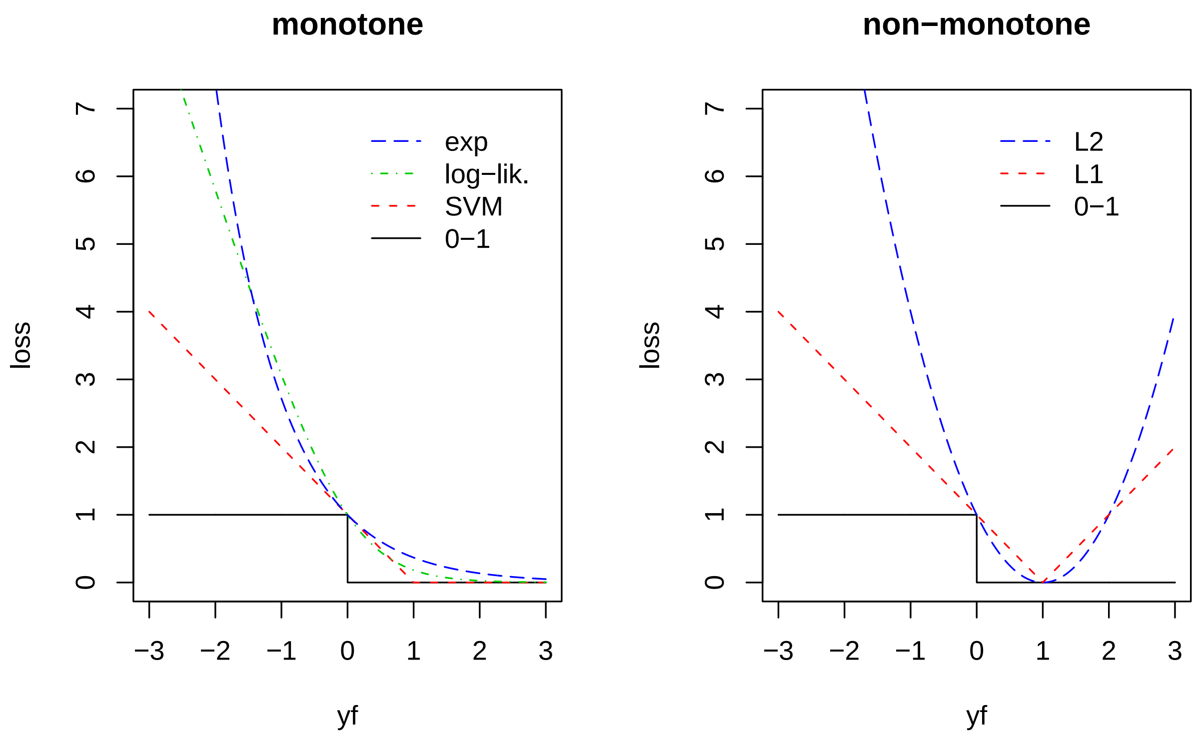
$\rho(y,f) = |y - f|^2 = 1 - 2yf + (yf)^2$: squared error
$f^*(x) = \mathbf{E}[Y|X = x] = 2p(x) - 1$

$\rho(y,f) = \exp(-yf)$: exponential loss in AdaBoost
$f^*(x) = \frac{1}{2}\log(\frac{p(x)}{1-p(x)})$

$\rho(y,f) = \mathbb{I}_{[yf<0]}$: misclassification loss
$f^*(x) = \mathbb{I}_{[p(x)\geq 1/2]}$

all these loss functions: $\rho(y,f) = \rho(yf)$:

function of the margin value yf



minimization of the non-convex misclassification loss: computationally infeasible

other loss functions: convex surrogate loss functions, dominating misclass. error

Buja, Stuetzle and Shen (2005): all these surrogate loss functions are “proper”

almost no difference from asymptotic point of view

my favourite: log-likelihood

- monotone
- approximately linear for large negative values yf

5.1. LogitBoost

(Friedman, Hastie & Tibshirani, 2000)

algorithm: FGD with negative log-likelihood and Hessian instaed of line-search

$\leadsto$ iterative weighted LS fitting: in iteration m,

$$n^{-1} \sum_{i=1}^{n} \underbrace{w_i}_{\hat{p}_{m-1}(X_i)(1-\hat{p}_{m-1}(X_i))} \left(\frac{Y_i - \hat{p}_{m-1}(X_i)}{\hat{p}_{m-1}(X_i)(1 - \hat{p}_{m-1}(X_i))} - \theta(X_i)\right)^2$$

since $f^*(x) = \log(\frac{p(x)}{1-p(x)}$ $\leadsto$ $\hat{f}_m(\cdot)$ is an estimate of the log-odds ratio

examples:

- componentwise weighted linear LS: $\leadsto$ logistic linear model fit
- weighted componentwise smoothing splines: $\leadsto$ logistic additive model fit
- weighted stumps: $\leadsto$ logistic additive model fit

works quite nicely for high-dimensional logistic linear or additive or low-order interaction models

6. Boosting in survival analysis

acute myeloid leukemia (AML) study from Bullinger et al., 2004:

survival times of $n = 116$ patient; 68 died during the study period

$p = 155$ predictors: 8 clinical variables, 147 gene expression levels

full data:

survival time $T_i \in \mathbb{R}^+$, predictor $X_i \in \mathbb{R}^p$ $\rightsquigarrow$ we use here $Y_i = \log(T_i)$

full data loss function: $\rho(y, f) = (y - f)^2$

observed data:

$O_i = (\tilde{Y}_i, X_i, \Delta_i),\ \tilde{Y}_i = \log(\tilde{T}_i),\ \tilde{T}_i = \min(T_i, C_i)$

censoring indicator $\Delta_i = \mathbb{1}_{[T_i \le C_i]}$

assume: censoring time C_i conditionally independent of T_i given X_i

$\rightsquigarrow$ coarsening at random assumption holds inverse probability censoring weights and observed data loss:

define observed data loss

$$\rho_{obs}(o, f) = (\tilde{y} - f)^2 \Delta \cdot \underbrace{\frac{1}{G(\tilde{t}|x)}}_{\text{inverse probability: } G(c|x) = \mathbb{P}[C > c | X = x]}$$

then (van der Laan & Robins, 2003):

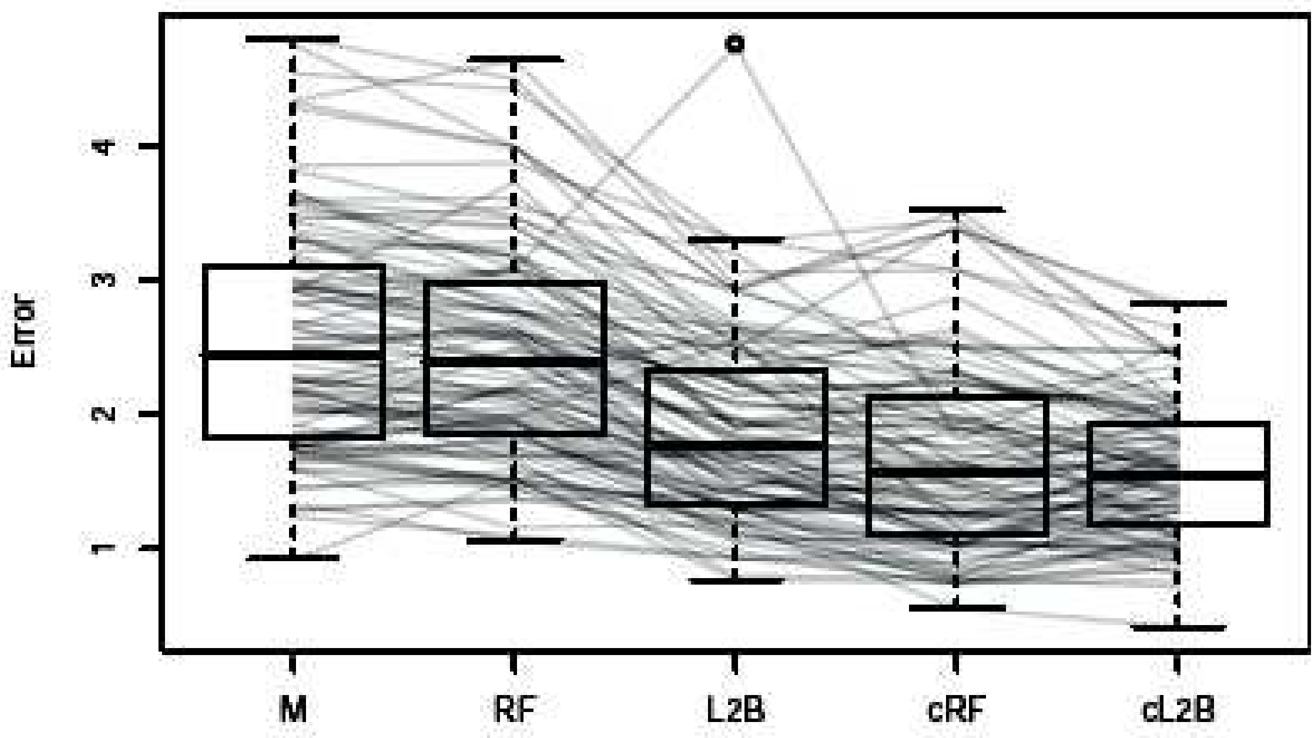
$$\mathbf{E}_{Y,X}[(Y - f(X))^2] = \mathbf{E}_O[\rho_{obs}(O, f)]$$

strategy: estimate $G(\cdot|x)$ e.g. by Kaplan-Meier and do boosting on weighted squared error loss:

$$\sum_{i=1}^{n} \underbrace{\Delta_i \frac{1}{\hat{G}(\tilde{T}_i|X_i)}}_{\text{weight } w_i} (\underbrace{\tilde{Y}_i}_{\log(\min(C_i, T_i))} - f(X_i))^2$$

we did componentwise weighted linear least squares

$\rightsquigarrow$ linear fit of the regression function $f(\cdot)$



M: location model; RF: random forest for survival data; L2B: L_2Boosting;

cRF: RF with 8 clinincal variables only; cL2B: L2B with 8 clinical variables only not possible to do the Henderson et al. (2001) loss:

$$\rho(T, f) = 1 - \mathbb{1}_{[T/2 \leq f \leq 2T]} \Leftrightarrow \rho(y, f) = \mathbb{1}_{[|y-f| > \log(2)]}$$

which is non-convex...!

in many real applications:

main interest is finding the relevant variables

(and prediction is of “minor” importance)

- tumor classification based on gene expression: which genes are important?
- Bullinger et al. survival study: which genes and variables are important?
- riboflavin concentration (vitamin B2) produced by Bacillus subtilis
  which genes are important? (in collaboration with DSM)

7. Variable selection and additional sparsity

is boosting a good variable selection method?

The analogy with the Lasso for linear models

consider again linear model (or highly overcomplete dictionary)

$$Y = f(X) + \varepsilon, \quad f(x) = \sum_{j=1}^{p} \beta_j x^{(j)}, \quad p \gg n$$

Lasso or ℓ^1-penalized regression (Tibshirani, 1996):

$$\hat{\beta}_{Lasso} = \text{argmin}_{\beta} n^{-1} \sum_{i=1}^{n} (Y_i - \sum_{j=1}^{p} \beta_j X_i^{(j)})^2 + \underbrace{\lambda}_{\geq 0;\ \text{penalty par.}} \sum_{j=1}^{p} |\beta_j|$$

Lasso:

- does variable selection: some (many) $\hat{\beta}_j$'s exactly equal to 0
- does shrinkage
- involves a convex optimization only
  (instead of exhaustively checking 2^p sub-models)

Some theory for high dimensions

Theorem (Meinshausen & PB, 2004)

For $\lambda_n \sim C n^{-1/2+\delta/2}$,

$$\mathbb{P}[\text{estimated sub-model}(\lambda_n) \; = \; \text{true model}] = 1 - O(\exp(-Cn^{\delta})) \quad (n \to \infty)$$

$$(0 < \delta < 1)$$

if

- Gaussian data
- $p = p_n = O(n^r)$ for any $r > 0$ (high-dimensional)
- number of effective variables $p_{eff} = O(n^k)$ $(0 < k < 1)$ (sparseness)
- plus some other technical conditions

justification for relaxation with a computationally simple convex problem!

Choice of λ

Theorem doesn't say much about choosing λ...

first (not so good) idea: choose λ to optimize prediction

e.g. via some cross-validation scheme

but: for prediction oracle solution

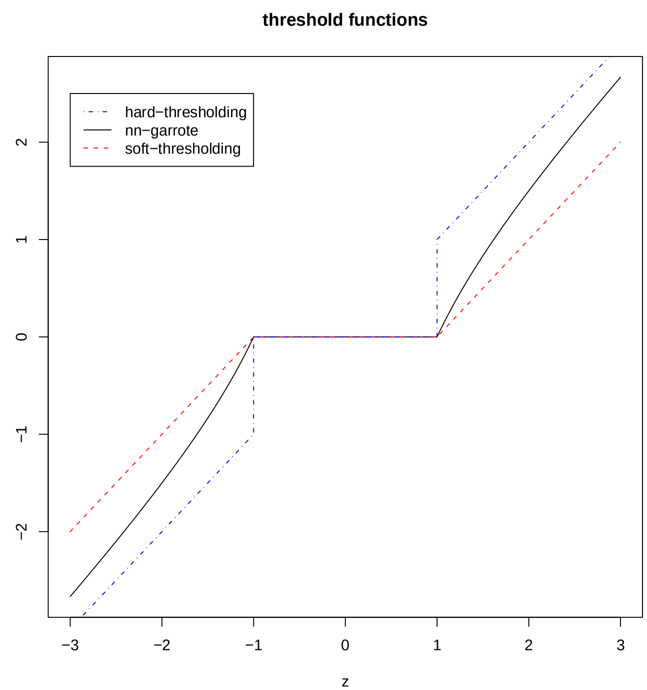
$$\lambda^* = \arg\min_{\lambda} \mathbf{E}[(Y - \sum_{j=1}^{p} \hat{\beta}_j(\lambda) X^{(j)})^2]$$

$\mathbb{P}[\text{estimated sub-model}(\lambda^*) = \text{true model}] \to 0 \ (p_n \to \infty, n \to \infty)$

asymptotically: the prediction optimal graph is too large

(Meinshausen & PB, 2004; related example by Meng et al., 2004) reason: need large λ for variable selection $\rightsquigarrow$ strong bias/strong shrinkage

for orthogonal design: strong bias in soft-thresholding



Better:

- SCAD (Fan and Li, 2001)
- Nonnegative Garrote (Breiman, 1995)
- Bridge estimation (Frank and Friedman, 1993)

they all work for general $\mathbf{X}$

for non-orthogonal $\mathbf{X}$:

- non-convex optimization for SCAD or Bridge estimation
- NN-Garrote only for $p \leq n$

The good message

Lasso produces a set of sub-models

$$M_1 \subset \ldots \subset \ldots \quad \underbrace{M_{pred-opt}}_{\text{optimal for prediction with Lasso}} \quad \subset \ldots \subset M_N$$

with $N = O(\min(n,p))$

and M_{true} is with probability $1 - O(\exp(-Cn^{\delta}))$ among these models

but $M_{true} \neq M_{pred-opt}$

Solutions using this "good message":

- relaxed Lasso (Meinshausen, 2005)

  a second round of Lasso on selected sub-models

  but surprisingly: computationally no need to do a second round of Lasso fitting
- BIC-scoring for selected submodels (?)

8. SparseL_2Boosting

(PB and Yu, 2005)

instead of minimizing RSS in every iteration,

minimize a final prediction error (FPE) criterion: we propose gMDL,

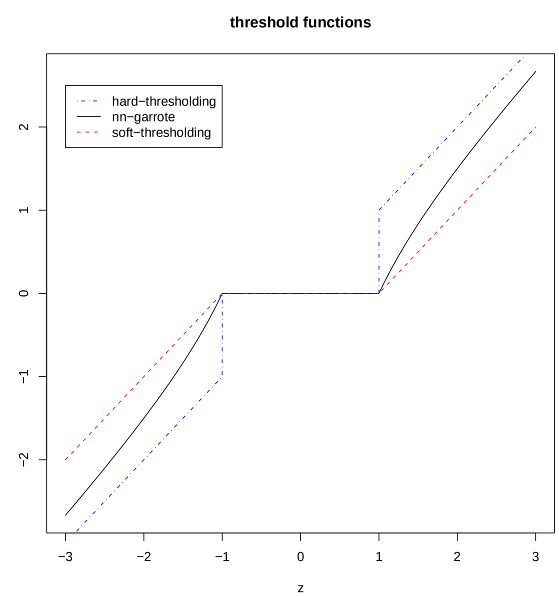
$$\hat{\theta}_m = \arg\min_{\theta(\cdot)} \sum_{i=1}^{n} (Y_i - \hat{f}_{m-1}(X_i) - \theta(X_i))^2 + \underbrace{\text{gMDL-penalty}}_{\text{or AIC, BIC,...}}$$

another use of degrees of freedom

Theorem (PB & Yu, 2005)

for orthonormal linear model:

SparseL_2Boosting with componentwise linear least squares yields Breiman's nonnegative garrote estimator



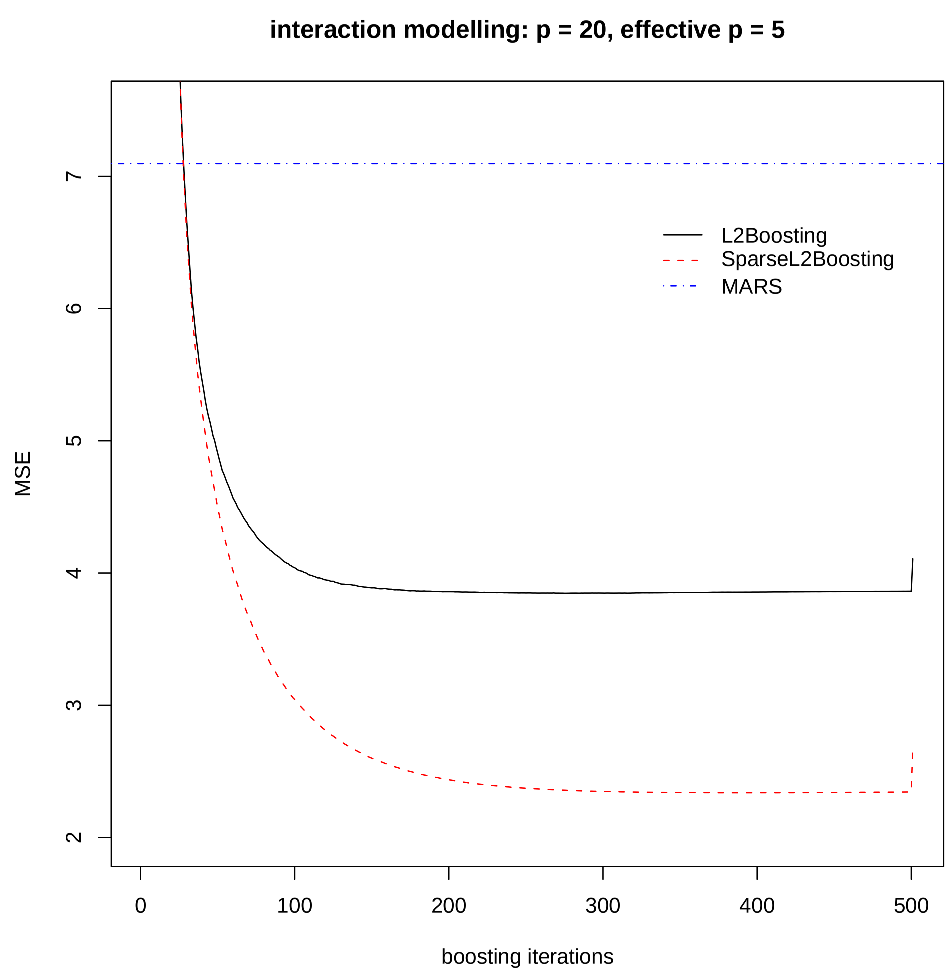
- SparseL_2Boosting yields sparser solutions than L_2Boosting
- SparseL_2Boosting still very generic (although less generic than L_2Boosting)
  e.g. nonparametric problems, non-quadratic loss functions

Linear modeling: L_2Boosting with componentwise linear LS

sample size $n = 50$, dimension $p = 50$

model	SparseL_2Boosting	L_2Boosting
$Y = 1 + 5X^{(1)} + 2X^{(2)} + X^{(3)} + \mathcal{N}(0,1)$		
$X = (X^{(1)}, \ldots, X^{(49)}) \sim \mathcal{N}_{49}(0, I)$		
MSE	0.16 (0.0018)	0.46 (0.0041)
$\mathbf{E}$[no. of seleccted variables]	5	13.68
$Y = \sum_{j=1}^{50} \beta_j X^{(j)} + \mathcal{N}(0,1)$		
$\beta_1, \ldots, \beta_{50} \sim$ Double-Exponential; X as above		
MSE	3.64 (0.188)	2.19 (0.083)

Nonparametric first-order interaction modeling



Friedman #1 model:

$Y = 10\sin(\pi X_1 X_2) + 20(X_3 - 0.5)^2 +$

$10X_4 + 5X_5 + \mathcal{N}(0,1)$

$X = (X_1, \ldots, X_{20}) \sim$ Unif.$([0,1]^{20})$

Sample size $n = 50$

Dimension $p = 20, p_{eff} = 5$

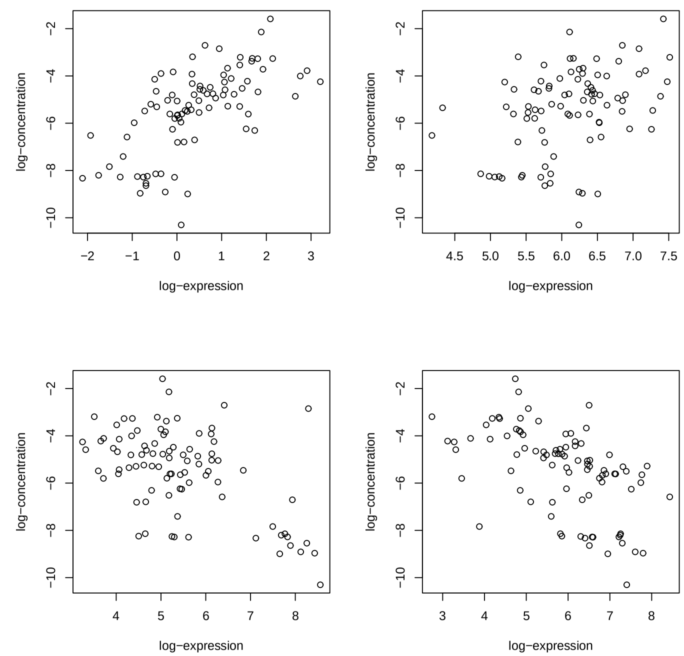
Riboflavin concentration in bacillus subtilis

$Y_i \in \mathbb{R}$: log-concentration of ribolflavin

$X_i \in \mathbb{R}^{6839}$:

$p = 6939$ gene expressions

sample size $n = 89$



L_2Boosting with componentwise linear least squares: selected 41 genes

SparseL_2Boosting with comonentwise linear least squares: selected 21 genes

15 genes are in common

note the identifiability problem due to high correlations among genes!

quite a few other measurements are available for this dataset...

9. Conclusions

statistical view of boosting:

a regularization method for estimation and variable selection

mainly useful for high-dimensional data problems

- boosting is very generic
- boosting is computationally attractive: complexity $O(p)$ for $p \gg n$
- simple statistical inference is possible, but more needs to be done