Iterated Regularization for High-Dimensional Data: from Boosting to Twin Boosting

Peter Bühlmann

Seminar für Statistik, ETH Zürich

The starting points

Regarding iterated regularization

1 tuning parameter (as in e.g. Lasso, Ridge, etc.) may not be sufficient to regularize in 1000-dimensional space

2-3 tuning parameters may be (much) better

Regarding Boosting: with a version of LogitBoost

Roman Lutz

Statistics, ETH Zurich

winner of the prediction/classification challenge

World Congress of Computational Intelligence 2006

$\rightsquigarrow$ Boosting is not an "out-dated" method

competitors were: weighted LS-SVM (S. Cawley)

Bayesian Neural Networks (R. Neal)

Random Forests (C. Dahinden)

SVM/Gaussian process classifier (W. Chu)

High-dimensional data setting

$(X_1, Y_1), \ldots, (X_n, Y_n)$ i.i.d. or stationary

X_i p-dimensional predictor variable

Y_i univariate response variable, e.g. $Y_i \in \mathbb{R}$ or $Y_i \in \{0, 1\}$

high-dimensional: $p \gg n$

areas of application: astronomy, biology, imaging, marketing research, text classification,...

sometimes n is large as well (and $p \approx n$ or $p \gg n$)
$\rightsquigarrow$ computational challenges

High-dimensional data setting

$(X_1, Y_1), \ldots, (X_n, Y_n)$ i.i.d. or stationary

X_i p-dimensional predictor variable

Y_i univariate response variable, e.g. $Y_i \in \mathbb{R}$ or $Y_i \in \{0, 1\}$

high-dimensional: $p \gg n$

areas of application: astronomy, biology, imaging, marketing research, text classification,...

sometimes n is large as well (and $p \approx n$ or $p \gg n$)
$\rightsquigarrow$ computational challenges

Examples from molecular biology

- Microarray data
  $p \approx 5'000 - 20'000$, $n \approx 10 - 200$
- Motif finding with motif regression
  $p \approx 4'000 - 10'000$, $n \approx 4'000 - 30'000$
- Dynamic (w.r.t. time) motif regression
  $p \approx 4'000 - 10'000$ and $n \approx 40'000 - 300'000$

Boosting

AdaBoost (Freund & Schapire, 1996): ensemble method

Breiman (1998) has demystified boosting as a
functional gradient descent method

Aim: find $f^*(\cdot) = \text{argmin}_{f(\cdot)} \mathbb{E}[\rho(Y, f(X))]$
e.g. for $\rho(y, f) = |y - f|^2 \rightsquigarrow f^*(x) = \mathbb{E}[Y|X = x]$

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

with the use of a base procedure (weak learner)

$$(X_1, U_1), \ldots, (X_n, U_n) \longrightarrow \hat{\theta}(\cdot) \approx \mathbb{E}[U|X = \cdot]$$

e.g. regression tree, componentwise smoothing spline, etc ...

Functional gradient descent: the concept

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

rough idea:

1. Initialize $\hat{f}_0(\cdot)$; then, for $m = 1, 2, \ldots m_{stop}$:
2. Calculate negative gradient (negative Gateaux derivative):

$$-dC(f)(x) = \frac{\partial}{\partial \alpha} - C(f + \alpha 1_x)|_{\alpha=0}$$

Approximate

$$-dC(\hat{f}_{m-1})(\cdot) \text{ by base procedure fit } \hat{\theta}_m(\cdot)$$

3. Up-date

$$\hat{f}_m(\cdot) = \hat{f}_{m-1}(\cdot) + \underbrace{\nu}_{\text{step-length}} \cdot \hat{\theta}_m(\cdot)$$

Computational implementation: Generic FGD algorithm

Step 1. $\hat{f}_0 \equiv 0$ (or $\equiv \overline{Y}$); 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 proced.

$$(X_i, U_i)_{i=1}^n \xrightarrow{\text{base proced.}} \hat{\theta}_m(\cdot)$$

i.e. $\hat{\theta}_m(\cdot)$ is an approximation of the negative gradient vector

Step 4. Up-date $\hat{f}_m(\cdot) = \hat{f}_{m-1}(\cdot) + \nu \cdot \hat{\theta}_m(\cdot)$
$(0 < \nu \leq 1$ step-length)
i.e: proceed along an estimate of the negative gradient vector

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

ν small will be good, e.g. $\nu = 0.1$

L_2Boosting (Friedman, 2001; PB & Yu, 2003)

loss function $\rho(y, f) = |y - f|^2$
population minimizer: $f^*(x) = \mathbb{E}[Y|X = x]$
FGD with base procedure $\hat{\theta}(\cdot)$: repeated fitting of residuals

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

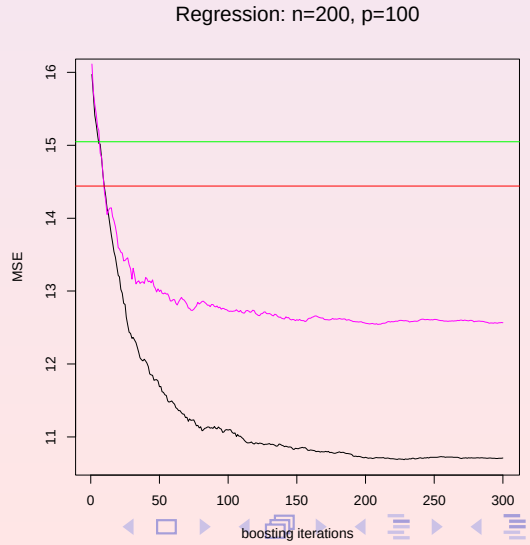
$$\hat{f}_{m_{stop}}(\cdot) = \nu \sum_{m=1}^{m_{stop}} \hat{\theta}_m(\cdot) \quad \text{(greedy fitting of residuals)}$$

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

any gain over classical methods?

- not at all if n/p is reasonable

- substantial gain if n/p is small



Componentwise linear least squares base procedure

linear ordinary least squares against the one predictor variable which reduces RSS most

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

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

2nd round: selected predictor variable $X^{(\hat{S}_2)}$ (e.g.$= X^{(21)}$)
corresponding $\hat{\beta}_{\hat{S}_2} \rightsquigarrow$ fitted function $\hat{f}_2(x)$

etc.

L_2Boosting: $\hat{f}_m(x) = \hat{f}_{m-1}(x) + \nu \cdot \hat{\theta}(x)$
$\rightsquigarrow$ linear model fit, including variable selection

i.e. a structured model fit

Componentwise linear least squares base procedure

linear ordinary least squares against the one predictor variable which reduces RSS most

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

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

2nd round: selected predictor variable $X^{(\hat{S}_2)}$ (e.g.$= X^{(21)}$)
corresponding $\hat{\beta}_{\hat{S}_2} \rightsquigarrow$ fitted function $\hat{f}_2(x)$

etc.

L_2Boosting: $\hat{f}_m(x) = \hat{f}_{m-1}(x) + \nu \cdot \hat{\theta}(x)$
$\rightsquigarrow$ linear model fit, including variable selection

i.e. a 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 Oxford

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 Oxford

Binary lymph node classification using gene expressions:
a high noise problem
$n = 49$ samples, $p = 7129$ gene expressions

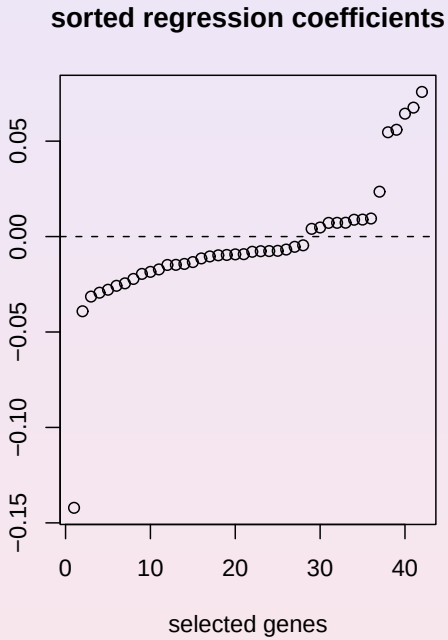
cross-validated misclassification error (2/3 training; 1/3 test)

Lasso	L_2Boosting	FPLR	Pelora	1-NN	DLDA	SVM
21.1%	17.7%	35.25%	27.8%	43.25%	36.12%	36.88%

multivariate gene selection

best 200 genes (Wilcoxon test)
no additional gene selection

42 (out of 7129) selected genes ($n = 49$) and gene importance



identifiability problem: strong correlations among some genes

$\rightsquigarrow$ consider groups of highly correlated genes
biological categories (e.g. GO ontology), ...

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

Zhao and Yu (2005): for general design matrices,
when adding some backward steps
the solutions from Lasso and modified Boosting "coincide"

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

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

Zhao and Yu (2005): for general design matrices,
when adding some backward steps
the solutions from Lasso and modified Boosting "coincide"

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

Consistency for high dimensions: an analysis of an algorithm

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

Theorem (PB, 2006)

L_2Boosting with comp. linear LS is consistent (with 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 $m = m_n$:

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

"no" assumptions about the predictor variables/design matrix

(similar result for Lasso: Greenshtein & Ritov (2004))

Consistency for high dimensions: an analysis of an algorithm

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

Theorem (PB, 2006)

L_2Boosting with comp. linear LS is consistent (with 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 $m = m_n$:

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

"no" assumptions about the predictor variables/design matrix

(similar result for Lasso: Greenshtein & Ritov (2004))

population version: $\langle f, g \rangle = \int f(x) g(x) dP(x)$

$g_j(x) = x^{(j)}, \quad \|g_j\| = 1$

$R_0 f = f$

$R_m f = R_{m-1} f - \langle R_{m-1} f, g_{\hat{S}_m} \rangle g_{\hat{S}_m}, \quad m = 1, 2, \ldots$

$\hat{S}_m = \text{argmax}_j |\langle R_{m-1} f, g_j \rangle|$

$$\begin{aligned} \|R_m f\|^2 &= \|f - \sum_{k=1}^{m} \gamma_k g_{\hat{S}_k}\|^2 \\ &= \|R_{m-1} f\|^2 - |\langle R_{m-1} f, g_{\hat{S}_m} \rangle|^2 \quad \searrow \end{aligned}$$

- $\searrow 0$?
- uniform bound ?

Temlyakov (2000):
$\leq \|\beta\|_1 m^{-1/6}$

population version: $\langle f, g \rangle = \int f(x) g(x) dP(x)$

$g_j(x) = x^{(j)}, \quad \|g_j\| = 1$

$R_0 f = f$

$R_m f = R_{m-1} f - \langle R_{m-1} f, g_{\hat{S}_m} \rangle g_{\hat{S}_m}, \; m = 1, 2, \ldots$

$\hat{S}_m = \text{argmax}_j |\langle R_{m-1} f, g_j \rangle|$

$$\|R_m f\|^2 = \|f - \sum_{k=1}^{m} \gamma_k g_{\hat{S}_k}\|^2$$

$$= \|R_{m-1} f\|^2 - |\langle R_{m-1} f, g_{\hat{S}_m} \rangle|^2 \searrow$$

- $\searrow 0$?
- uniform bound ?

Temlyakov (2000):
$\leq \|\beta\|_1 m^{-1/6}$

population version: $\langle f, g \rangle = \int f(x) g(x) dP(x)$

$g_j(x) = x^{(j)}, \ \ \|g_j\| = 1$

$R_0 f = f$

$R_m f = R_{m-1} f - \langle R_{m-1} f, g_{\hat{S}_m} \rangle g_{\hat{S}_m}, \ m = 1, 2, \ldots$

$\hat{S}_m = \text{argmax}_j |\langle R_{m-1} f, g_j \rangle|$

$$\begin{aligned} \|R_m f\|^2 &= \|f - \sum_{k=1}^{m} \gamma_k g_{\hat{S}_k}\|^2 \\ &= \|R_{m-1} f\|^2 - |\langle R_{m-1} f, g_{\hat{S}_m} \rangle|^2 \ \searrow \end{aligned}$$

- $\searrow 0$?
- uniform bound ?

Temlyakov (2000):
$\leq \|\beta\|_1 m^{-1/6}$

population version: $\langle f, g \rangle = \int f(x) g(x) dP(x)$

$g_j(x) = x^{(j)}, \quad \|g_j\| = 1$

$R_0 f = f$

$R_m f = R_{m-1} f - \langle R_{m-1} f, g_{\hat{S}_m} \rangle g_{\hat{S}_m}, \; m = 1, 2, \ldots$

$\hat{S}_m = \text{argmax}_j |\langle R_{m-1} f, g_j \rangle|$

$$\begin{aligned} \|R_m f\|^2 &= \|f - \sum_{k=1}^{m} \gamma_k g_{\hat{S}_k}\|^2 \\ &= \|R_{m-1} f\|^2 - |\langle R_{m-1} f, g_{\hat{S}_m} \rangle|^2 \quad \searrow \end{aligned}$$

- $\searrow 0$?
- uniform bound ?

Temlyakov (2000):
$\leq \|\beta\|_1 m^{-1/6}$

population version: $\langle f, g \rangle = \int f(x) g(x) dP(x)$

$g_j(x) = x^{(j)}, \quad \|g_j\| = 1$

$R_0 f = f$

$R_m f = R_{m-1} f - \langle R_{m-1} f, g_{\hat{S}_m} \rangle g_{\hat{S}_m}, \; m = 1, 2, \ldots$

$\hat{S}_m = \text{argmax}_j |\langle R_{m-1} f, g_j \rangle|$

$$\begin{aligned} \|R_m f\|^2 &= \|f - \sum_{k=1}^{m} \gamma_k g_{\hat{S}_k}\|^2 \\ &= \|R_{m-1} f\|^2 - |\langle R_{m-1} f, g_{\hat{S}_m} \rangle|^2 \quad \searrow \end{aligned}$$

- $\searrow 0$?
- uniform bound ?

Temlyakov (2000):
$\leq \|\beta\|_1 m^{-1/6}$

population version: $\langle f, g \rangle = \int f(x) g(x) dP(x)$

$g_j(x) = x^{(j)}, \quad \|g_j\| = 1$

$R_0 f = f$

$R_m f = R_{m-1} f - \langle R_{m-1} f, g_{\hat{S}_m} \rangle g_{\hat{S}_m}, \ m = 1, 2, \ldots$

$\hat{S}_m = \text{argmax}_j |\langle R_{m-1} f, g_j \rangle|$

$$\begin{aligned} \|R_m f\|^2 &= \|f - \sum_{k=1}^{m} \gamma_k g_{\hat{S}_k}\|^2 \\ &= \|R_{m-1} f\|^2 - |\langle R_{m-1} f, g_{\hat{S}_m} \rangle|^2 \ \searrow \end{aligned}$$

- $\searrow 0$?
- uniform bound ?

Temlyakov (2000):
$\leq \|\beta\|_1 m^{-1/6}$

population version: $\langle f, g \rangle = \int f(x) g(x) dP(x)$

$g_j(x) = x^{(j)}, \quad \|g_j\| = 1$

$R_0 f = f$

$R_m f = R_{m-1} f - \langle R_{m-1} f, g_{\hat{S}_m} \rangle g_{\hat{S}_m}, \ m = 1, 2, \ldots$

$\hat{S}_m = \text{argmax}_j |\langle R_{m-1} f, g_j \rangle|$

$$\begin{aligned} \|R_m f\|^2 &= \|f - \sum_{k=1}^{m} \gamma_k g_{\hat{S}_k}\|^2 \\ &= \|R_{m-1} f\|^2 - |\langle R_{m-1} f, g_{\hat{S}_m} \rangle|^2 \ \searrow \end{aligned}$$

- $\searrow 0$?
- uniform bound ?

Temlyakov (2000):
$\leq \|\beta\|_1 m^{-1/6}$

linear model $\rightsquigarrow$ “prototype”-result

the methodology (and some of theory) is much more general

Other loss functions for boosting: beyond regression

for binary classification with $Y \in \{0, 1\}$:
$\rho(y, f) = \log_2(1 + \exp(-(2 \cdot y - 1)f))$
negative binomial log-likelihood
population minimizer: $f^*(x) = \frac{1}{2} \log(\frac{p(x)}{1-p(x)})$
$\rightsquigarrow$ can estimate probabilities $p(\cdot)$ from estimate $\hat{f}(\cdot)$

this is LogitBoost (Friedman, Hastie and Tibshirani, 2000)

for count data with $Y \in \{0, 1, 2, \ldots\}$:
$\rho(y, f) = \exp(f) - yf$
negative Poisson log-likelihood
population minimizer: $f^*(x) = \log(\mathbb{E}[Y|X = x])$

for survival data with $Y \in \mathbb{R}^+$:
$\rho(y, f)$ from Cox's partial likelihood

etc...

Other loss functions for boosting: beyond regression

for binary classification with $Y \in \{0, 1\}$:
$\rho(y, f) = \log_2(1 + \exp(-(2 \cdot y - 1)f))$
negative binomial log-likelihood
population minimizer: $f^*(x) = \frac{1}{2} \log(\frac{p(x)}{1-p(x)})$
$\rightsquigarrow$ can estimate probabilities $p(\cdot)$ from estimate $\hat{f}(\cdot)$

this is LogitBoost (Friedman, Hastie and Tibshirani, 2000)

for count data with $Y \in \{0, 1, 2, \ldots\}$:
$\rho(y, f) = \exp(f) - yf$
negative Poisson log-likelihood
population minimizer: $f^*(x) = \log(\mathbb{E}[Y|X = x])$

for survival data with $Y \in \mathbb{R}^+$:
$\rho(y, f)$ from Cox's partial likelihood

etc...

Other loss functions for boosting: beyond regression

for binary classification with $Y \in \{0, 1\}$:
$\rho(y, f) = \log_2(1 + \exp(-(2 \cdot y - 1)f))$
negative binomial log-likelihood
population minimizer: $f^*(x) = \frac{1}{2} \log(\frac{p(x)}{1-p(x)})$
$\rightsquigarrow$ can estimate probabilities $p(\cdot)$ from estimate $\hat{f}(\cdot)$

this is LogitBoost (Friedman, Hastie and Tibshirani, 2000)

for count data with $Y \in \{0, 1, 2, \ldots\}$:
$\rho(y, f) = \exp(f) - yf$
negative Poisson log-likelihood
population minimizer: $f^*(x) = \log(\mathbb{E}[Y|X = x])$

for survival data with $Y \in \mathbb{R}^+$:
$\rho(y, f)$ from Cox's partial likelihood

etc...

Computation

computation for general loss functions involves a
trivial extension only!
very different for LARS-type path-following algorithms for Lasso

instead of residuals in L_2Boosting

$$U_i = Y_i - \hat{f}_{m-1}(X_i),\ i = 1, \ldots, n$$

we use "generalized residuals"

$$U_i = -\frac{\partial}{\partial f}\rho(Y, f)|_{f=\hat{f}_{m-1}(X_i)},\ i = 1, \ldots, n$$

since there is (usually) a closed form, simple expression of the partial derivative
$\rightsquigarrow$ same computational cost as for L_2Boosting

The mboost package in R (Hothorn & PB, 2006)

for various boosting algorithms and corresponding model fitting

- easy to use and coherent implementation for
  - regression
  - classification
  - Poisson regression
  - survival analysis with Cox's partial likelihood
  - your own loss function
- allows for various base procedures
  - componentwise linear least squares
  - componentwise smoothing splines
  - trees
- computationally very fast for high-dimensional generalized linear models

CPU time

Binary lymph node classification example: $p = 7129$, $n = 49$
with L_2Boosting or BinomialBoosting (LogitBoost)
for large range of solutions

it's less than a second!

0.906 seconds using `mboost` in `R` (Hothorn & PB, 2006)

in comparison:
for linear models, computing Lasso solutions for all λ's

2.603 seconds using `lars` in `R` (with `use.gram=F`)

CPU time

Binary lymph node classification example: $p = 7129$, $n = 49$
with L_2Boosting or BinomialBoosting (LogitBoost)
for large range of solutions

it's less than a second!

0.906 seconds using `mboost` in `R` (Hothorn & PB, 2006)

in comparison:
for linear models, computing Lasso solutions for all λ's

2.603 seconds using `lars` in `R` (with `use.gram=F`)

CPU time

Binary lymph node classification example: $p = 7129$, $n = 49$

with L_2Boosting or BinomialBoosting (LogitBoost)

for large range of solutions

it's less than a second!

0.906 seconds using `mboost` in `R` (Hothorn & PB, 2006)

in comparison:

for linear models, computing Lasso solutions for all λ's

2.603 seconds using `lars` in `R` (with `use.gram=F`)

because it is so fast

$\rightsquigarrow$ local modeling over different inhomogeneous but related sub-populations ("borrowing strength from neighborhood")

$$\beta(t) \approx \beta(\text{neighb.}(t))$$

Motif detection using DNA sequence and gene expression data

(Meier, Liu, Liu & PB; work in progress)

$p = 4'312$, $n = 79'974$ from 18 sub-populations

Can we easily improve?

maybe (?) not that much with respect to prediction but often substantially with respect to variable/feature selection

approach for variable selection with Boosting:
variables which have been selected by the base procedure in the process of boosting

e.g. in a (generalized) linear model fit:
variables with corresponding regression coefficient $\neq 0$

$\rightsquigarrow$ no significance testing involved

Can we easily improve?

maybe (?) not that much with respect to prediction but often substantially with respect to variable/feature selection

approach for variable selection with Boosting:
variables which have been selected by the base procedure in the process of boosting

e.g. in a (generalized) linear model fit:
variables with corresponding regression coefficient $\neq 0$

$\rightsquigarrow$ no significance testing involved

Can we easily improve?

maybe (?) not that much with respect to prediction but often substantially with respect to variable/feature selection

approach for variable selection with Boosting:
variables which have been selected by the base procedure in the process of boosting

e.g. in a (generalized) linear model fit:
variables with corresponding regression coefficient $\neq 0$

$\rightsquigarrow$ no significance testing involved

Building on the analogy with the Lasso

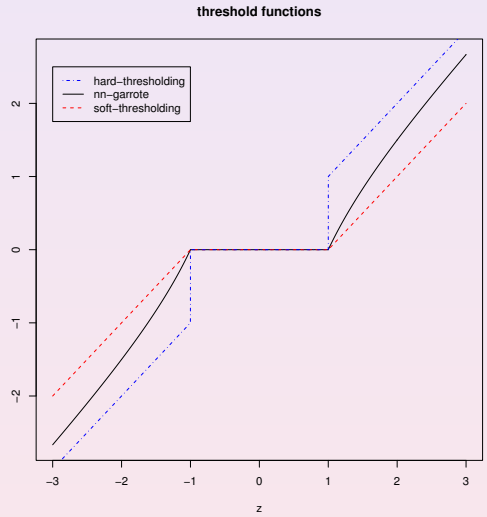
Meinshausen & PB (2006):
for linear models

- Lasso is consistent for variable selection, even for $p \gg n$, if the design matrix is not "too correlated"
  $P[\text{selected model} = \text{ true model}] \underbrace{\to 0}_{\text{quickly}} \quad (n \to \infty)$
- if the design is "too correlated"
  $\rightsquigarrow$ Lasso is inconsistent for variable selection

see also Zou (2006), Zhao and Yu (2006)

The “reason”

too much bias (or shrinkage), even for large values



Bias in soft-thresholding
is disturbing (at least sometimes)

better:
Nonnegative Garrote (Breiman, 1995)
and similar proposals

Twin Boosting (PB, 2006): Boosting-type answer addressing the problem of bias

is different from Sparse Boosting (PB & Yu, 2006)
but much more general and computationally much faster

Rough idea of Twin Boosting

- first round of boosting as usual: first twin
- second round of boosting which is forced to resemble the first round: second twin
- final estimate from the second round

Rough idea of Twin Boosting

- first round of boosting as usual: first twin
- second round of boosting which is forced to resemble the first round: second twin
- final estimate from the second round

Rough idea of Twin Boosting

- first round of boosting as usual: first twin
- second round of boosting which is forced to resemble the first round: second twin
- final estimate from the second round

Twin L_2Boosting for linear models

recap: L_2Boosting with componentwise linear least squares

chooses variable j which reduces RSS most

$$\Leftrightarrow \quad |n^{-1}\sum_{i=1}^{n} U_i X_i^{(j)}| = |\widehat{\mathrm{Cor}}(U, X^{(j)})| \text{ maximal w.r.t. } j$$

if predictor variables are standardized

first round: boosting estimate $\hat{\beta}_{init}$ from L_2Boosting

second round: as L_2Boosting but selecting variable using

$$|n^{-1}\sum_{i=1}^{n} U_i X_i^{(j)}| \cdot |\hat{\beta}_{init,j}| \text{ maximal w.r.t. } j$$

final estimate from second round, "pulled toward" initial estim.

$\rightsquigarrow$ very easy and computationally efficient modification

Twin Boosting = iterated regularization

(we first tune round 1;
and fixing the tuning from round 1, we then tune round 2)

PB (2006):
for orthogonal linear models, as step-size factor $\nu \to 0$

Twin L_2Boosting and Adaptive Lasso coincide
(with β_{init} = estimate from first round of boosting)

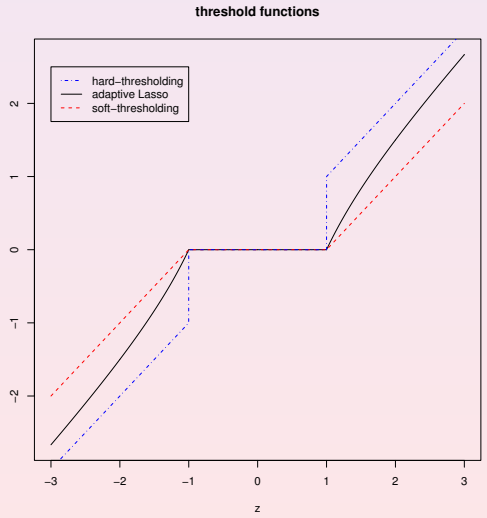
and Twin Boosting extends to very general settings

Adaptive Lasso (Zou, 2006)

$$\hat{\beta} = \text{argmin}_\beta \sum_{i=1}^{n} (Y_i - (X\beta)_i)^2 + \lambda \sum_{j=1}^{p} \frac{|\beta_j|}{\underbrace{|\beta_{init,j}|}_{\text{e.g. OLS if } p < n}}$$

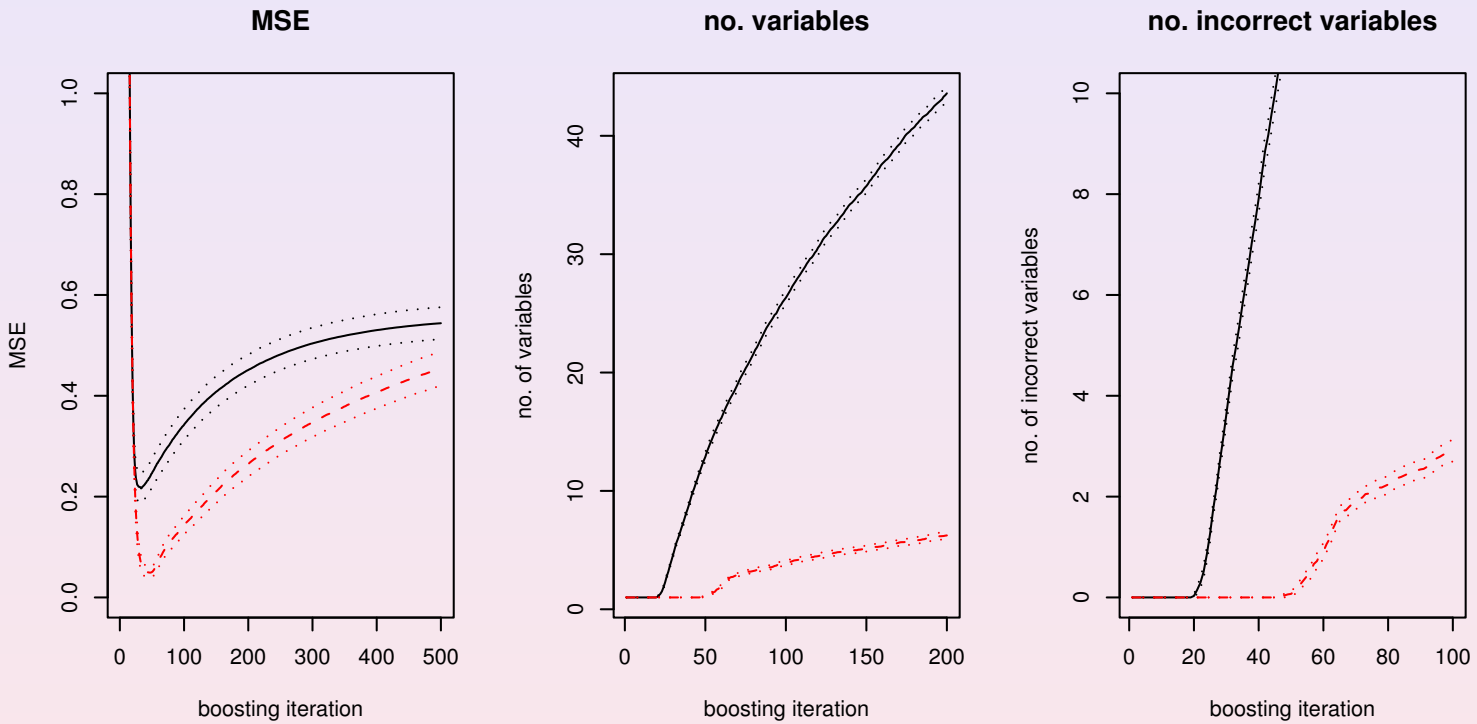
nice result (Zou, 2006):
adaptive Lasso is consistent for variable selection "in general"
(proof for low-dimensional problems only)

for orthogonal design
if β_{init} = OLS,
Adaptive Lasso = NN-garrote



Simulated example: n = 50, p=500

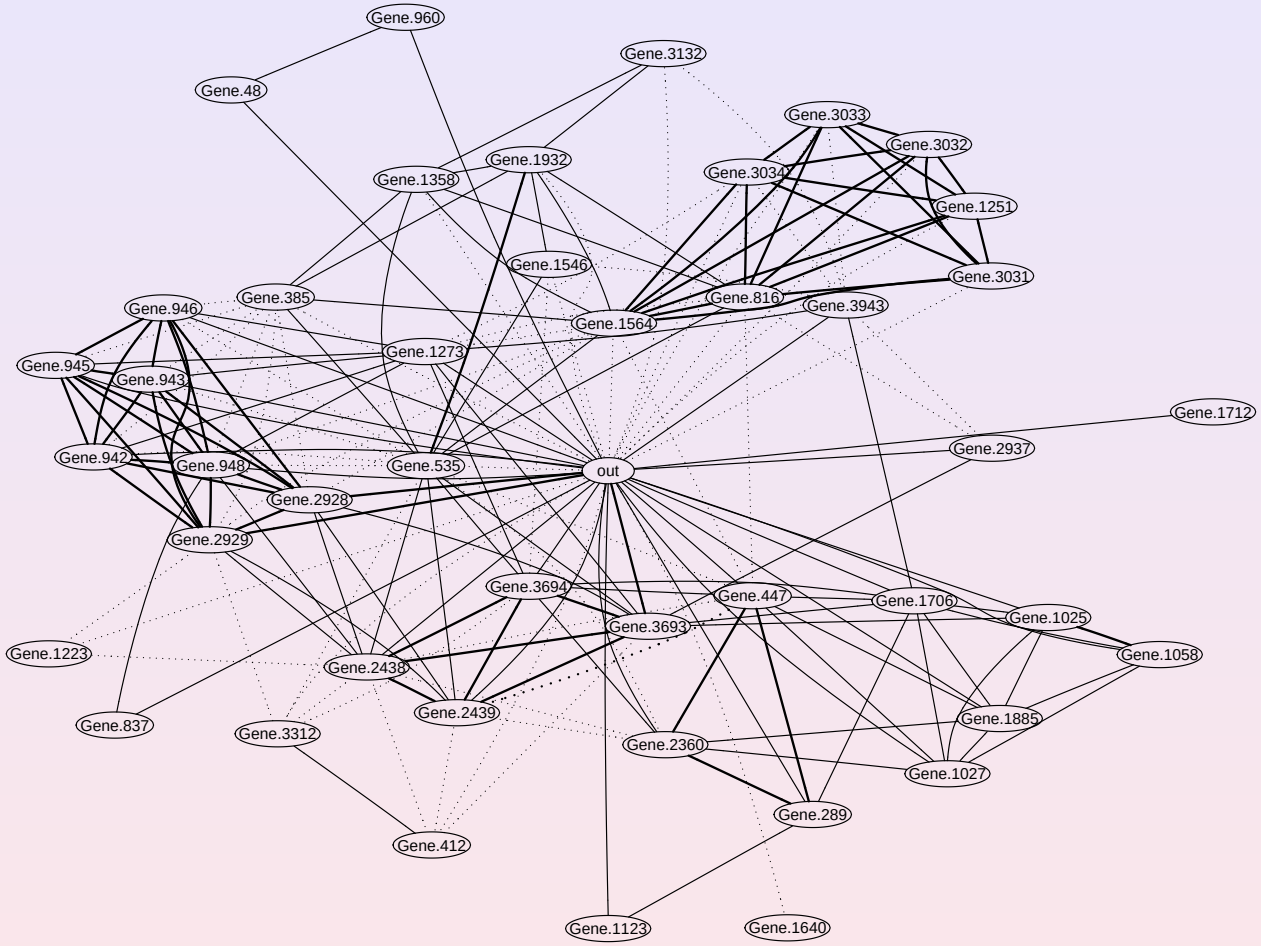
$$Y = \sum_{j=1}^{500} \beta_j X^{(j)} + \varepsilon, \ \beta_1 = 5, \beta_j = 0 \ (j = 2, \ldots, 500)$$



black: L_2Boosting red: Twin L_2Boosting

Twin L_2Boosting: more sparse and better variable selection than boosting

Response Y versus $p = 4088$ gene expressions in Bacillus Subtilis



neighbours of Y (selected genes) and their conditional dependencies n=115, p=4088

Boosting with regression tree base procedure
very popular in machine learning

trees can be very useful because:

- they can easily handle missing data
- they can easily deal with mixed categorical, ordinal, continuous data
- they are invariant under monotone covariate-transformations

Twin L_2Boosting for trees

first round: boosting estimate $\hat{f}_{init}$

second round: as boosting but select in each iteration the best tree $\hat{g}(\cdot)$ which reduces RSS and "resembles" $\hat{f}_{init}$

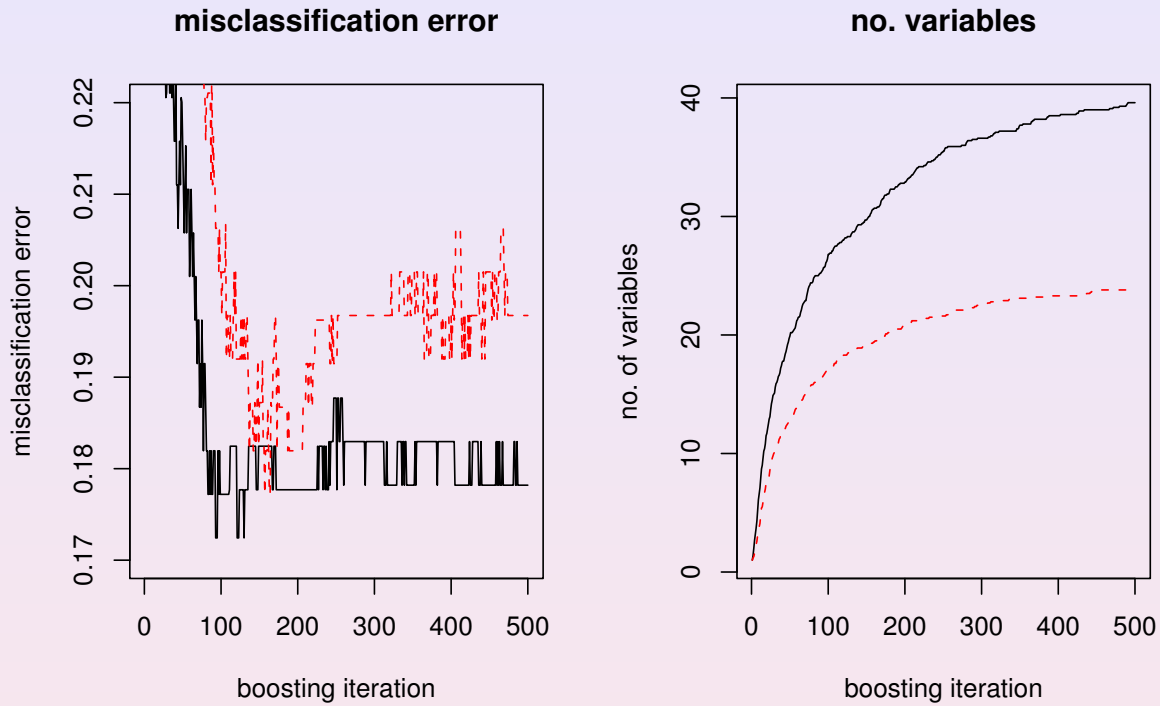
$$\underbrace{C^2(\hat{g})}_{\widehat{Cor}^2(\hat{g},\hat{f}_{init})} \cdot \underbrace{\left(2\sum_{i=1}^{n} U_i \hat{g}(X_i) - \sum_{i=1}^{n} \hat{g}(X_i)^2\right)}_{\text{"penalized correlation"}} \text{ is maximized w.r.t. } \hat{g}$$

in case of componentwise least squares and uncorrelated design

$$\propto |\hat{\beta}_{init,j}|^2 \cdot |\widehat{Cor}(U, X^{(j)})|^2$$

$\rightsquigarrow$ concept easily extends to other loss functions (e.g. classification)

Sonar data: binary classification with n=208, p = 60

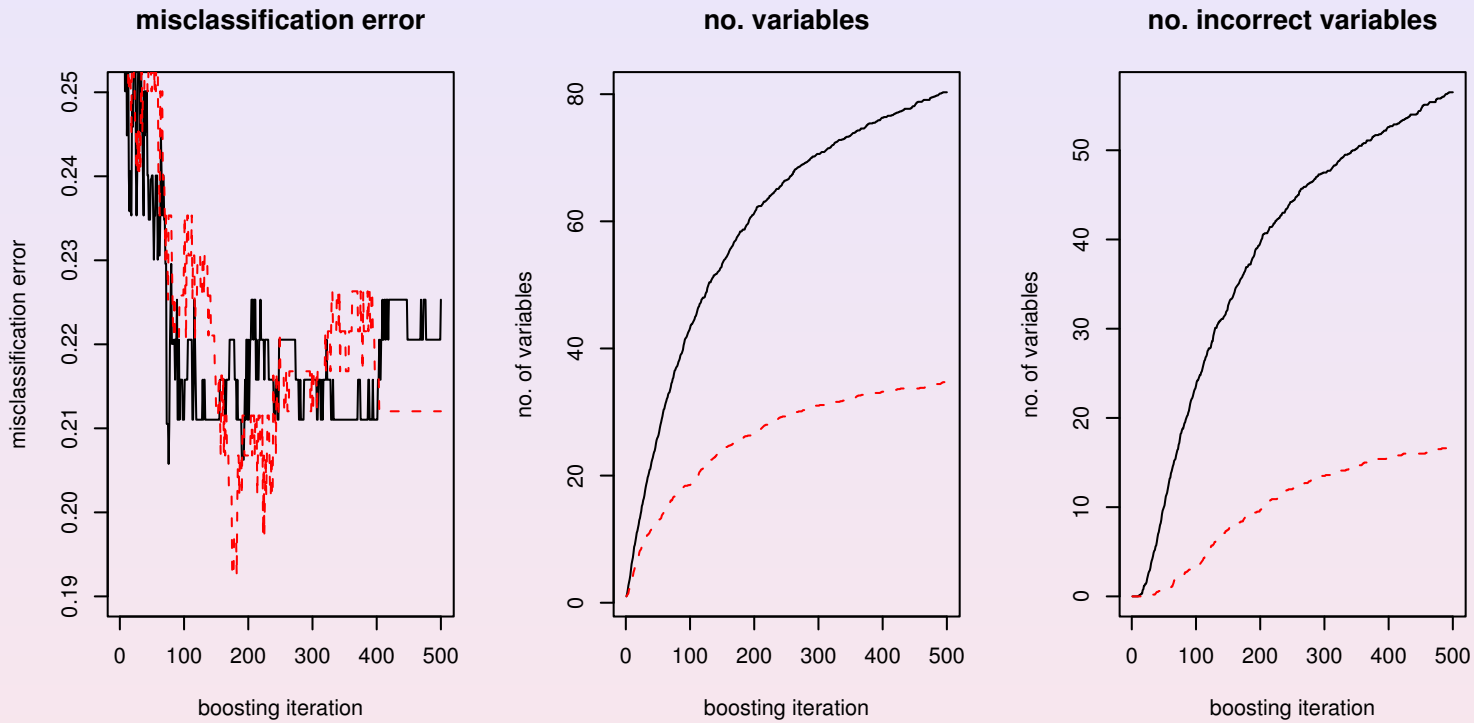


black: BinomialBoosting red: Twin BinomialBoosting

Twin Boosting: more sparse than boosting

with synthetically enlarged predictor space

adding 500 $\mathcal{N}(0,1)$-distributed ineffective predictor variables



black: BinomialBoosting red: Twin BinomialBoosting

$\rightsquigarrow$ improved variable selection with Twin Boosting

Conclusions

Boosting

- is mainly useful for high-dimensional and/or large datasets
- is computationally very efficient
- is very competitive for prediction

- Twin Boosting (e.g. iterated regularization) improves upon
  - variable selection
  - assigning variable importance in structured models (linear, additive, interaction)