Boosting and Twin Boosting for High-Dimensional Data

Peter Bühlmann

Seminar für Statistik, ETH Zürich

High-dimensional data

$(X_1, Y_1), \ldots, (X_n, Y_n)$ 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,...

High-dimensional data

$(X_1, Y_1), \ldots, (X_n, Y_n)$ 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,...

Examples from molecular biology

Microarray data

- predictor variables:
  expressions of $p \approx 5'000 - 20'000$ genes
- response variable: e.g. cancer sub-type or survival time
- sample size is $n \approx 10 - 200$

Splice site detection in DNA sequences

$p = 16'384, \ n \approx 11'000$

Examples from molecular biology

Microarray data

- predictor variables:
  expressions of $p \approx 5'000 - 20'000$ genes
- response variable: e.g. cancer sub-type or survival time
- sample size is $n \approx 10 - 200$

Splice site detection in DNA sequences

$p = 16'384, \; n \approx 11'000$

Ensemble methods

also called multiple predictions, aggregation methods, ...

seem to be mainly useful for high-dimensional data
and for large data-sets with p and n large

“philosophy”:

a combination of estimates is better
than a single individual prediction

“clear” in the Bayesian paradigm:
e.g. Bayesian model averaging

Ensemble methods

also called multiple predictions, aggregation methods, ...

seem to be mainly useful for high-dimensional data
and for large data-sets with p and n large

“philosophy”:

a combination of estimates is better
than a single individual prediction

“clear” in the Bayesian paradigm:
e.g. Bayesian model averaging

General description

Base procedure or weak learner:

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

e.g.: simple linear regression, tree, MARS, "classical" smoothing, ...

generating an ensemble of estimates (or 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 (or "voting"): $\hat{f}_A(\cdot) = \sum_{m=1}^{M} a_m \hat{\theta}_m(\cdot)$

data weights ? averaging weights a_m ?

two main settings:

- independently generated ensemble:
  Example: Bagging, where
  • reweighted data is generated by bootstrapping
  • aggregation is the mean of the estimates
- coordinated generation of ensemble:
  sequential: ensemble m depends on previous ensembles
  direct dependence typically only on ensemble $m-1$
  Example: Boosting

Bagging is essentially a variance reduction method
e.g. use large trees and reduce their variance via bagging

Boosting is “basis expansion” and a bias reduction method
e.g. small trees whose combination yields richer function class

Random Forests (Alit & Geman, 1996; Breiman, 2001) is an
indep. generated ensemble with “randomized tree learner”

two main settings:

- independently generated ensemble:
  Example: Bagging, where
  - reweighted data is generated by bootstrapping
  - aggregation is the mean of the estimates
- coordinated generation of ensemble:
  sequential: ensemble m depends on previous ensembles
  direct dependence typically only on ensemble $m - 1$
  Example: Boosting

Bagging is essentially a variance reduction method
e.g. use large trees and reduce their variance via bagging

Boosting is “basis expansion” and a bias reduction method
e.g. small trees whose combination yields richer function class

Random Forests (Alit & Geman, 1996; Breiman, 2001) is an
indep. generated ensemble with “randomized tree learner”

two main settings:

- independently generated ensemble:
  Example: Bagging, where
  - reweighted data is generated by bootstrapping
  - aggregation is the mean of the estimates
- coordinated generation of ensemble:
  sequential: ensemble m depends on previous ensembles
  direct dependence typically only on ensemble $m - 1$
  Example: Boosting

Bagging is essentially a variance reduction method
e.g. use large trees and reduce their variance via bagging

Boosting is “basis expansion” and a bias reduction method
e.g. small trees whose combination yields richer function class

Random Forests (Alit & Geman, 1996; Breiman, 2001) is an indep. generated ensemble with “randomized tree learner”

mathematically: definition(s) not precise...
because you could use/think of the ensemble method again as the base procedure

but there are many impressive empirical results

with a version of LogitBoost (an ensemble method)

Roman Lutz
Statistics, ETH Zurich

winner of the WCCI 2006 prediction/classification challenge

$\rightsquigarrow$ 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)

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 $\hat{p}_{opt} \approx 50$)

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

are we happy with that?

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 $\hat{p}_{opt} \approx 50$)

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

are we happy with that?

What about fitted models?
Interpretation of estimated parameters?

$\rightsquigarrow$ prediction, variable selection, variable importance

- Linear models
- Generalized linear models
- Additive models
- Interaction models

Boosting algorithms

AdaBoost proposed for classification:
Freund & Schapire (1996)

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

averaging weights a_m:
large if in-sample performance in *m*th round was good

Why should this be good?

Boosting algorithms

AdaBoost proposed for classification:

Freund & Schapire (1996)

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

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)

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)

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)

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

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 proced.

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

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

Step 4. Up-date $\hat{f}_m = \hat{f}_{m-1}(\cdot) + \nu \cdot \hat{\theta}_m(\cdot)$ $(0 < \nu \leq 1$ 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. $\nu = 0.1$

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:

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

as previously defined!

By definition: FGD yields additive combination of base procedure fits $\nu \sum_{m=1}^{m_{stop}} \hat{\theta}_m(\cdot)$

Breiman (1998):
FGD with $\rho(y, f) = \exp((2y - 1) \cdot f)$ for binary classification
yields the AdaBoost algorithm
(great result!)

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

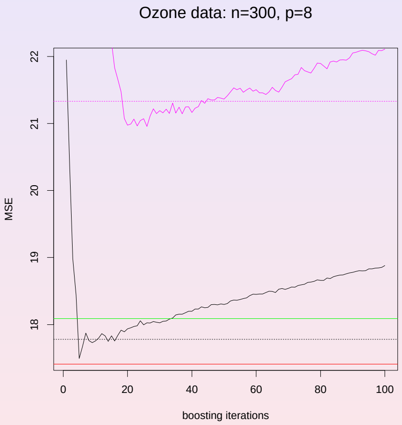
$$\begin{array}{lllll} m = 1: & (X_i, Y_i)_{i=1}^n & \rightsquigarrow \hat{\theta}_1(\cdot),\ \hat{f}_1 = \hat{\theta}_1 & \rightsquigarrow & \text{resid. } U_i = Y_i - \hat{f}_1(X_i) \\ m = 2: & (X_i, U_i)_{i=1}^n & \rightsquigarrow \hat{\theta}_2(\cdot),\ \hat{f}_2 = \hat{f}_1 + \hat{\theta}_2 & \rightsquigarrow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

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

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

any gain over classical methods? (for additive modeling)

$n = 300, \; p = 8$



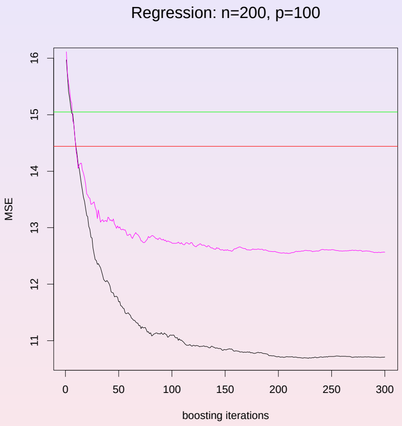
- magenta: L_2Boosting with stumps (horiz. line = cross-validated stopping)
- black: L_2Boosting with componentwise smoothing spline (horiz. line = cross-validated stopping) i.e: smoothing spline fitting against the selected predictor which reduces RSS most
- green: MARS restricted to additive modeling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

L_2Boosting with stumps or comp. smooth. spl: additive model,

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

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 backfitting and

  fwd. var. selection

similar for classification

very often:

Boosting performs comparatively well in high-dimensions

(there is a lot of empirical evidence for this!)

Structured models and choosing the base procedure

have just seen: with the
componentwise smoothing spline base procedure

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

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

i.e. structured model fit

Structured models and choosing the base procedure

have just seen: with the
componentwise smoothing spline base procedure

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

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

i.e. structured model fit

Componentwise linear least squares base procedure

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

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

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

2nd round: 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: $\hat{f}_m(x) = \hat{f}_{m-1}(x) + \nu \cdot \hat{\theta}(x)$
$\rightsquigarrow$ linear model fit, including variable selection

i.e. structured model fit

Componentwise linear least squares base procedure

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

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

first round: selected predictor variable $X^{(\hat{\mathcal{S}}_1)}$ (e.g. $= X^{(3)}$)
corresponding $\hat{\beta}_{\hat{\mathcal{S}}_1} \rightsquigarrow$ fitted function $\hat{f}_1(x)$
2nd round: 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: $\hat{f}_m(x) = \hat{f}_{m-1}(x) + \nu \cdot \hat{\theta}(x)$
$\rightsquigarrow$ 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 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

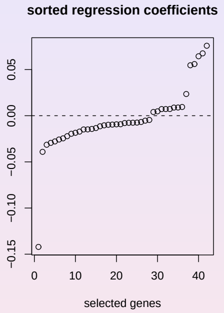
cross-validated misclassification error (2/3 training; 1/3 test)

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 ($n = 49$)



identifiability problem: strong correlations among some genes

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

Pairwise smoothing spline base procedure
smoothes response against the pair of predictor variables which reduces RSS most
⇝ L_2Boosting yields a nonparametric interaction model, including variable selection

i.e. structured model fit

Pairwise smoothing spline base procedure
smoothes response against the pair of predictor variables which reduces RSS most

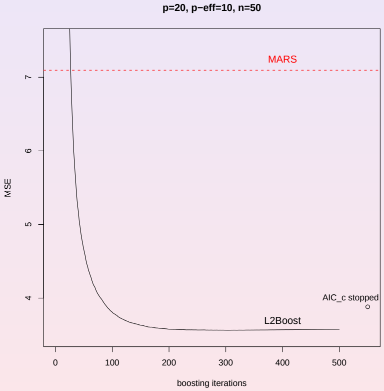
$\rightsquigarrow$ L_2Boosting yields a nonparametric interaction model, including variable selection

i.e. structured model fit

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 $p_{eff} = 5$

both methods have the same (high) degree of interpretation

Regression tree base procedure

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

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

i.e. "fairly" structured model fit

note: 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

Regression tree base procedure

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

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

i.e. “fairly” structured model fit

note: 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

Regression tree base procedure

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

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

i.e. “fairly” structured model fit

note: 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

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

there is a supporting
asymptotic minimax theory for this principle (PB & Yu, 2003)

asymptotically as $n \to \infty$,
L_2Boosting with smoothing spline is optimal (achieves minimax rate) for 1-dimensional curve estimation ($x \in \mathbb{R}^1$)

if degrees of freedom (number of equivalent parameters) in the smoothing spline are fixed (i.e. "low")

and number of boosting iterations asymptotically increases

More on 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

$$\hat{\beta}(\lambda) = \text{argmin}_{\beta}(n^{-1}\|Y - X\beta\|^2 + \underbrace{\lambda}_{\geq 0} \underbrace{\|\beta\|_1}_{\sum_{j=1}^{p}|\beta_j|})$$

More on 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

$$\hat{\beta}(\lambda) = \text{argmin}_{\beta}(n^{-1}\|Y - X\beta\|^2 + \underbrace{\lambda}_{\geq 0} \underbrace{\|\beta\|_1}_{\sum_{j=1}^{p}|\beta_j|})$$

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 clever as well

Zhao and Yu (2005): in "general",
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

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

Least Angle Regression LARS (Efron et al., 2004) is computationally clever as well

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

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

The theoretical limit for dimensionality

linear model

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

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

The theoretical limit for dimensionality

linear model

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

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

analogous results also for
- multivariate regression
- vector autoregressive time series

(Lutz & PB, 2006)

Other loss functions for boosting: beyond regression

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

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

etc...

Requirement: $\rho(y, f)$ is differentiable with respect to f almost everywhere
for example: L_1-loss $\rho(y, f) = |y - f|$ is OK

Other loss functions for boosting: beyond regression

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

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

etc...

Requirement: $\rho(y, f)$ is differentiable with respect to f almost everywhere
for example: L_1-loss $\rho(y, f) = |y - f|$ is OK

Other loss functions for boosting: beyond regression

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

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

etc...

Requirement: $\rho(y, f)$ is differentiable with respect to f almost everywhere
for example: L_1-loss $\rho(y, f) = |y - f|$ is OK

Computation

computation for general loss functions involves a
trivial extension only!

instead of residuals in L_2Boosting

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

we use “generalized residuals”

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

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 weak learners
  - componentwise 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 (Binomial log-likelihood)
for large range of solutions

it's less than a second!

0.906 seconds   using `mboost` in `R` (PB & Hothorn, 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 (Binomial log-likelihood)
for large range of solutions

it's less than a second!

0.906 seconds using `mboost` in R (PB & Hothorn, 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 (Binomial log-likelihood)
for large range of solutions

it's less than a second!

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

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

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

DNA splice site detection: a classification problem

DNA sequence

$$\dots ACGGC \dots \quad \underbrace{NNN \quad \underbrace{GC}_{\text{potential donor site}} \quad NNNN}_{\text{3 positions exon } GC \text{ 4 positions intron}} \quad \dots AAC \dots$$

response $Y \in \{0, 1\}$: splice or non-splice site
predictor variables: 7 factors each having 4 levels
(full dimension: $4^7 = 16'384$)
data: $p = 16'384$, $n = 11'220$

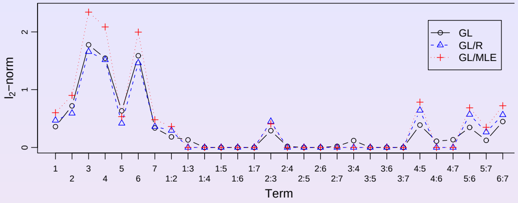
logistic regression:

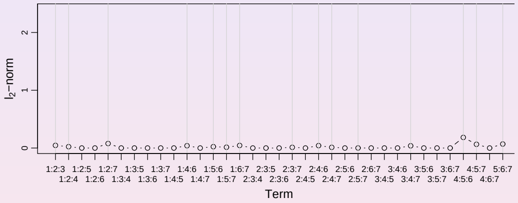
$$\log\left(\frac{p(x)}{1-p(x)}\right) = \beta_0 + \text{ main effects} + \text{first order interactions} + \ldots$$

with sum-to-zero constraints

use groupwise linear least squares base procedure
and binomial likelihood loss

“groupwise”:
e.g. the whole interaction term between factor 2 and 5 (which is encoded with 9 free parameters/dummy indicators) is fitted at a time





- mainly neighboring DNA positions show interactions (has been “known” and “debated”)
- no interaction among exon and intron positions
- no second-order interactions

predictive power:
competitive with “state to the art” maximum entropy modeling from Yeo & Burge (2004)

test set correlation between true and predicted class

Boosting	0.6593
max. entropy (Yeo & Burge)	0.6589

- our model is simple and has clear interpretation
- it is as good or better than many of the complicated non-Markovian stochastic process models (e.g. Zhao, Huang and Speed (2004))

we have other examples on

- survival analysis
- high-order contingency tables
- etc...

Can we easily improve?

(probably) 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 correspodning regression coefficient $\neq 0$

$\rightsquigarrow$ no significance testing involved

Can we easily improve?

(probably) 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 correspodning regression coefficient $\neq 0$

$\rightsquigarrow$ no significance testing involved

Can we easily improve?

(probably) 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 correspodning 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"
  $P[\text{selected model} = \text{true model}] \underbrace{\rightarrow 0}_{\text{quickly}} \quad (n \rightarrow \infty)$
- prediction optimal solution selects too many variables
- if the design is "too correlated"
  $\rightsquigarrow$ Lasso is inconsistent for variable selection

Adaptive Lasso (Zou, 2006) resolves the inconsistency problem for variable selection:

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

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

Twin Boosting (PB, 2006)

encompassing the adaptive Lasso for special cases

rough idea: two runs of boosting

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

Twin Boosting (PB, 2006)

encompassing the adaptive Lasso for special cases

rough idea: two runs of boosting

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

Twin Boosting (PB, 2006)

encompassing the adaptive Lasso for special cases

rough idea: two runs of boosting

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

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

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

"pulling it towards" the initial estimate
$\rightsquigarrow$ very easy and computationally efficient modification

PB (2006):
for orthogonal linear models,

Twin L_2Boosting and adaptive Lasso coincide
(with β_{init} = estimate from first round of boosting)
as step-size factor $\nu \to 0$

Twin Boosting (as well as adaptive Lasso) involve
2 tuning parameters
$\leadsto$ more powerful for regularization in high-dimensional spaces

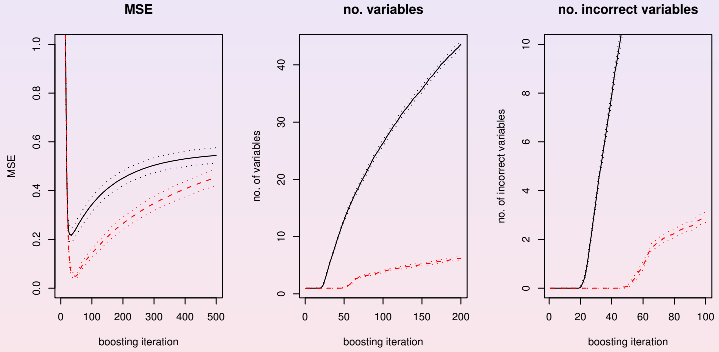
PB (2006):
for orthogonal linear models,

Twin L_2Boosting and adaptive Lasso coincide
(with β_{init} = estimate from first round of boosting)
as step-size factor $\nu \to 0$

Twin Boosting (as well as adaptive Lasso) involve
2 tuning parameters
$\rightsquigarrow$ more powerful for regularization in high-dimensional spaces

Simulated example: n = 50, p=500

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



black: L_2Boosting red: Twin L_2Boosting

easily extends to logistic linear and generalized linear models

Twin Boosting for trees

recap: trees can be excellent base procedures because

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

first round: boosting estimate f_{init}

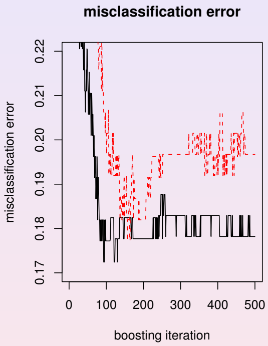
second round: as boosting but select in each iteration the best tree $\hat{g}(\cdot)$ which reduces RSS and "resembles" f_{init}

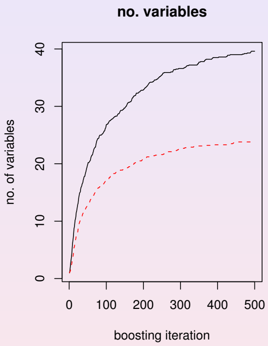
$$\underbrace{C_{\hat{g}}}_{Cor(\hat{g}, f_{init})} \cdot \underbrace{\left(2\sum_{i=1}^{n} U_i \hat{g}(X_i) - \sum_{i=1}^{n} \hat{g}(X_i)^2\right)}_{\text{"penalized correlation"}} \text{ is maximized w.r.t. } \hat{g}$$

(there is some mathematical justification for this)

$\rightsquigarrow$ concept and formulae easily extend to classification, etc...

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

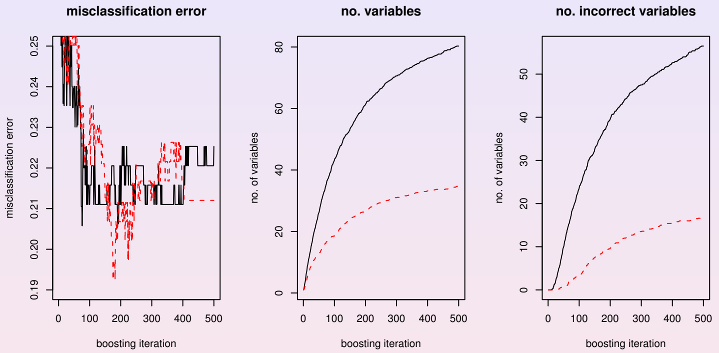




black: L_2Boosting red: Twin L_2Boosting

with synthetically enlarged predictor space

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



black: L_2Boosting red: Twin L_2Boosting

$\rightsquigarrow$ 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

- and Twin Boosting improves upon
  - variable selection
  - assigning variable importance in structured models (linear, additive, interaction)