Boosting methods: why they can be useful for high-dimensional data

Peter Bühlmann

ETH Zürich

1. Historically, it's about multiple predictions

Data: $(X_1, Y_1), \ldots, (X_n, Y_n)$ (i.i.d. realizations),

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 $\theta(\cdot) : \mathbb{R}^p \to \mathbb{R}$, e.g.

$\theta(x) = \mathbf{E}[Y|X = x]$ or $\theta(x) = \mathbb{P}[Y = 1|X = x]$ with $Y \in \{0, 1\}$

Approach: Boosting which can be viewed (at first sight!) as a

multiple predictions (estimates) & combination method

Base procedure:

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

e.g.: regression-/classification tree, MARS, “classical” smoothing, neural nets, ...

Generating multiple predictions:

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

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

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

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

data weights? averaging weights a_m?

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

$n = 33$, $p = 7129$ but reduced to $10 \leq p \leq 50$

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

2. Boosting

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?

(actually: other weighting schemes are equally good or better...)

Breiman (1999): AdaBoost is functional gradient descent (FGD) procedure

Aim: find $f_0(\cdot) = \text{argmin}_{f(\cdot)} \mathbf{E}[\rho(Y, f(X))]$

e.g. for $\rho(y, f) = |y - f|^2 \rightsquigarrow f_0(x) = \mathbf{E}[Y|X = x]$

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

- compute $\underbrace{\text{negative gradient vector } (U_1, \ldots, U_n)}_{\text{“pseudo-response”}}$
- fit negative gradient vector by base procedure

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

- Up-date $f_{new} = f_{old}(\cdot) + \underbrace{w}_{\text{line search}} \hat{\theta}(\cdot)$

$\rightsquigarrow \sum_m w_m \hat{\theta}_m(\cdot)$: additive combination of fits from base procedure

(AdaBoost is FGD with $\rho(y, f) = \exp((2y - 1)f),\ y \in \{0, 1\}$)

2.1 L_2Boosting

(see also Friedman, 2001)

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

population minimizer: $f_0(x) = \mathbf{E}[Y|X = x]$

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

$$
\begin{aligned}
&m = 1: \ (X_i, Y_i)_{i=1}^n \rightsquigarrow \hat{\theta}_1(\cdot),\ f_1 = \hat{\theta}_1 && \rightsquigarrow \text{resid. } U_i = Y_i - f_1(X_i)\\
&m = 2: \ (X_i, U_i)_{i=1}^n \rightsquigarrow \hat{\theta}_2(\cdot),\ f_2 = f_1 + \hat{\theta}_2 && \rightsquigarrow \text{resid. } U_i = Y_i - f_2(X_i)\\
&\qquad\qquad\qquad \ldots && \qquad\qquad \ldots
\end{aligned}
$$

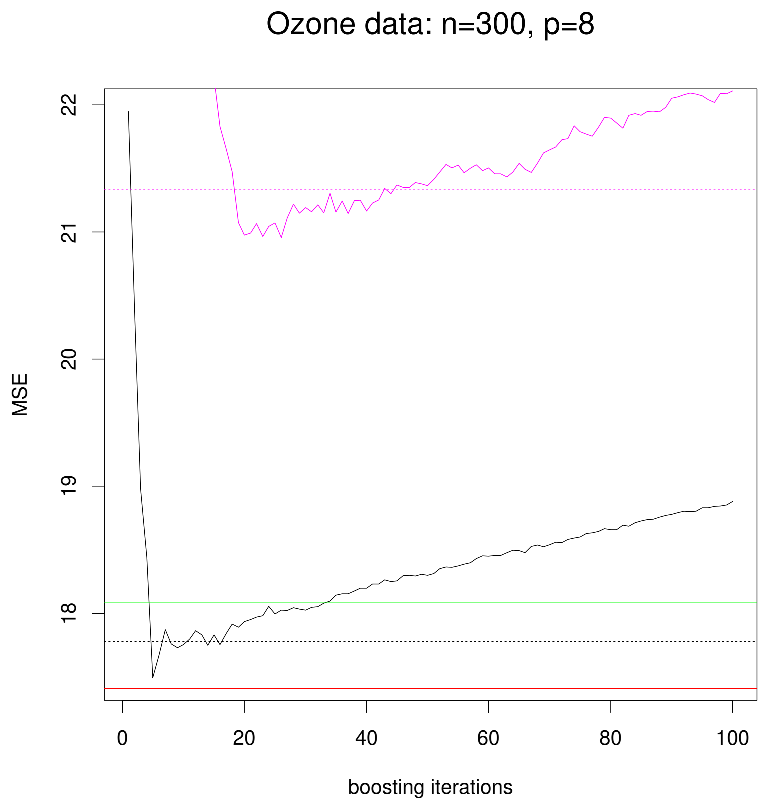
$f_M(\cdot) = \sum_{m=1}^M \hat{\theta}_m(\cdot)$ (stagewise greedy fitting of residuals)

Tukey (1977): twicing for $M = 2$

Base procedure for $\hat{\theta}_m(\cdot)$ typically with low variance but high bias (“weak”)

Any gain over classical methods?

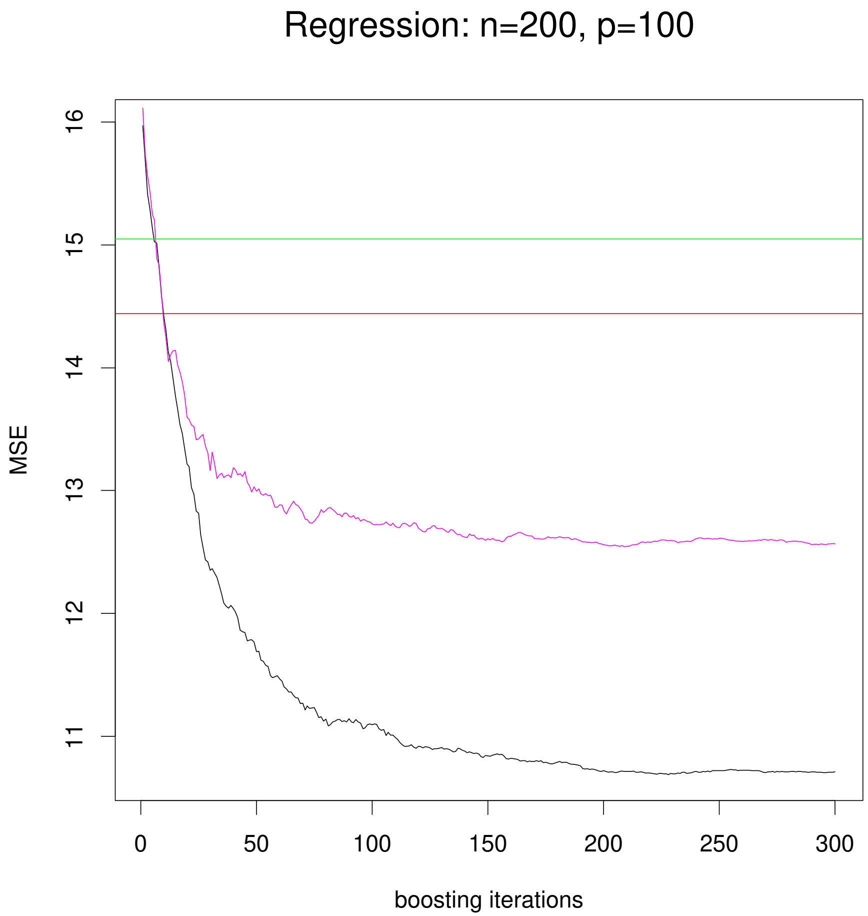
Ozone data: $n = 300$, $p = 8$



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

$$\sum_{m=0}^{M} \hat{\theta}_m(x_{\hat{c}(m)}) = \hat{g}_1(x_1) + \ldots + \hat{g}_p(x_p)$$

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



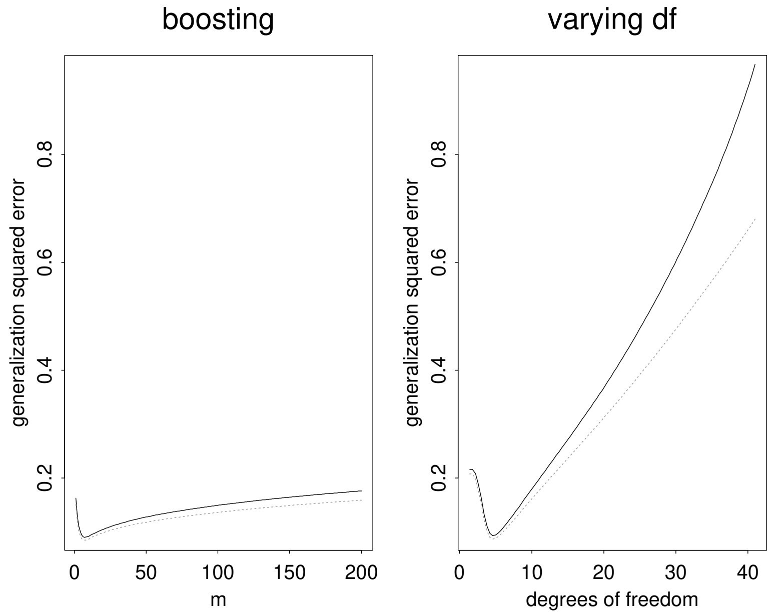
oracle MSE = 4

Empirically, very similar for classification

2.1. Some theory

1. Asymptotic optimality in 1-dim. curve estimation (PB & Yu, 2001)

consider L_2Boost with smoothing spline



sub-linear increase of MSE in Boosting

L_2Boost quite resistant against overfitting; "easy to tune"

Consider (any) base learner as operator:

$$\mathcal{S} : Y = (Y_1, \ldots, Y_n)' \overset{\text{algorithm A}}{\mapsto} \hat{Y} = (\hat{Y}_1, \ldots, \hat{Y}_n)'$$

L_2Boost operator in iteration m:

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

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

$\rightsquigarrow$ L_2Boost converges to identity I (fully saturated model)

in case of $\mathcal{S}$ a smoothing spline:

L_2Boost does shrinkage in the same eigenspace as the smoothing spline $\mathcal{S}$

eigenvalues of L_2Boost: $0 < 1 - (1 - \underbrace{\lambda}_{\text{eigenv. of } \mathcal{S}})^m \leq 1$

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

Results:

L_2Boost 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

2. Consistent function approximation in high dimension (PB & Yu, in prog.)

Example: L_2Boost with componentwise linear least squares regression

- it does variable selection

  since some of the predictors may never be chosen

  (for stopped number of boosting iterations)

- it does variable amount of degrees of freedom for every selected predictor x_j^*

  depending when and how many times we picked that x_j^*

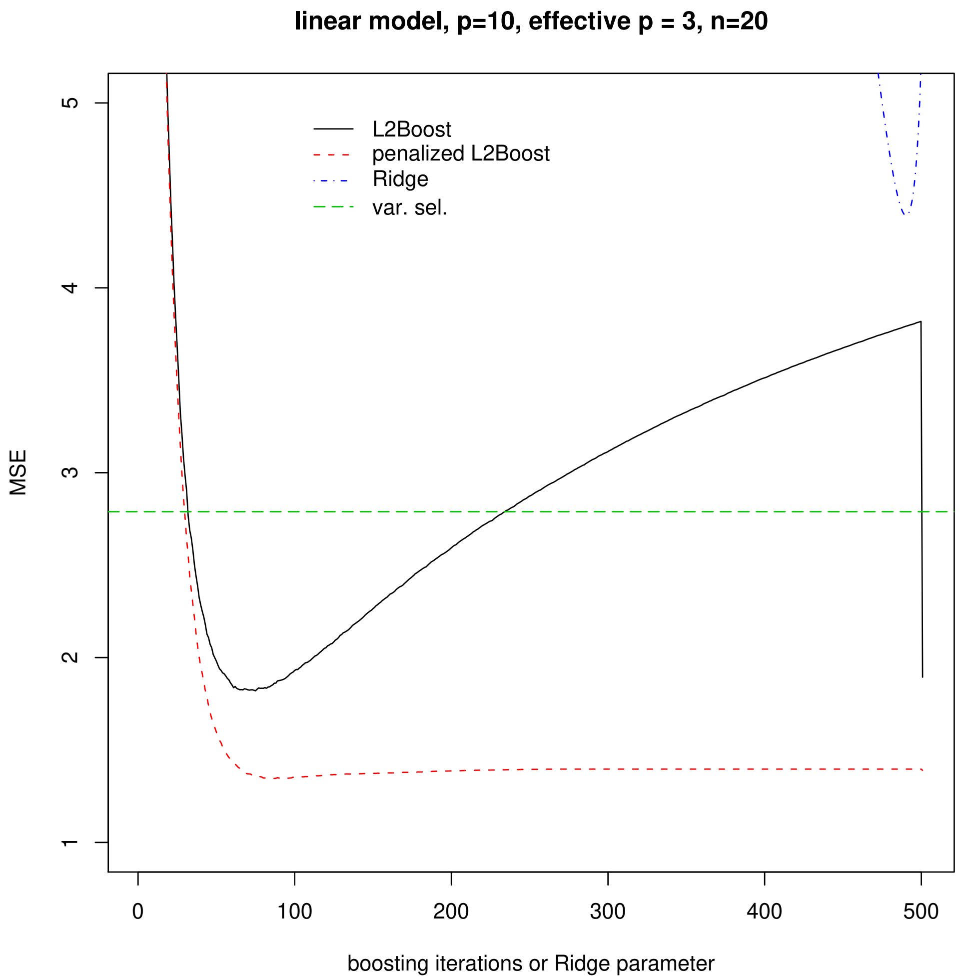
Such L_2Boost is consistent for p_n-dimensional linear regression with

$$p_n = O(C\exp(n^1)) \text{ for all } C > 0 \ \ (\text{very fast increase!}),$$

and true regression function "semi-sparse":

$$f(x) = \sum_{j=1}^{p_n} \beta_{j,n} x_j, \ \ \sup_n \sum_{j=1}^{p_n} |\beta_{j,n}| < \infty.$$

and L_2Boost "fundamentally different" from variable selection



2.2. L_2Boost in action with many predictors

Model: $p = 100$

additive regression function $f(\cdot)$ with effective p = 10

Sample size: n = 200

Method: L_2Boost with componentwise smoothing splines (picking variable producing lowest RSS) having df = 2.5 (or df = 5)

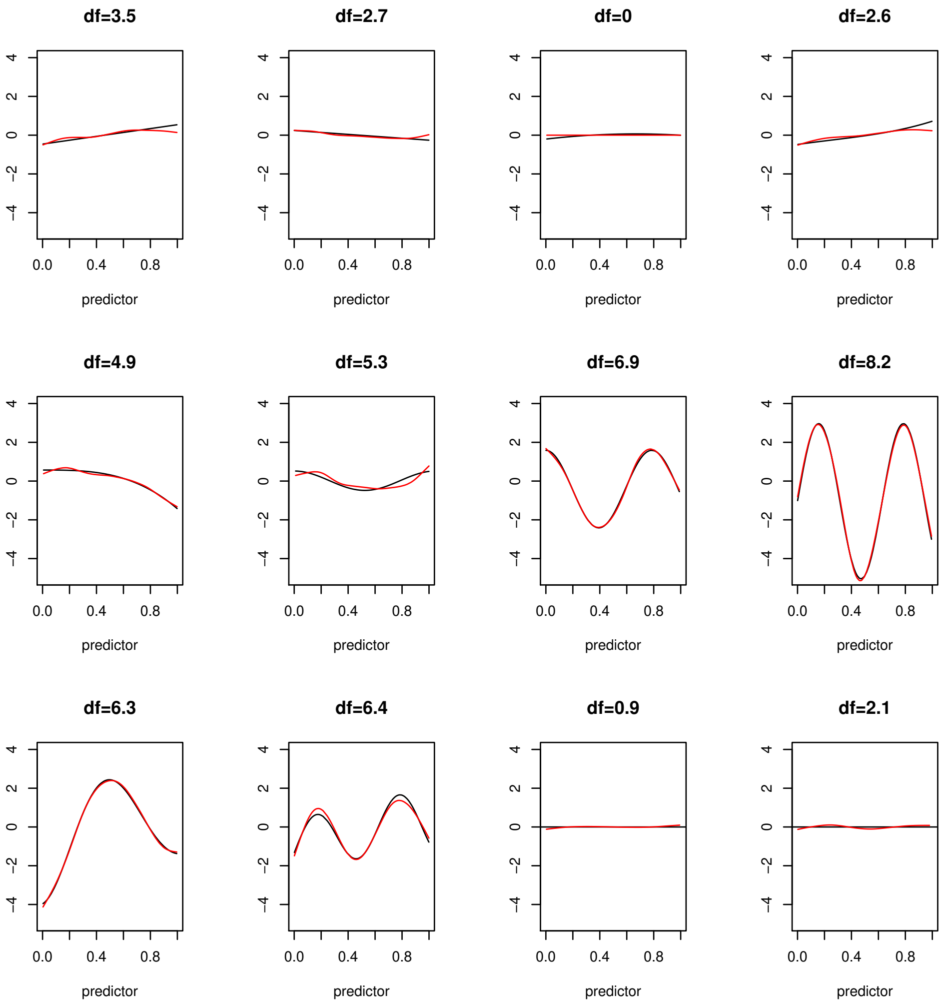
$\rightsquigarrow$ additive model fitting

Such L_2Boosting does

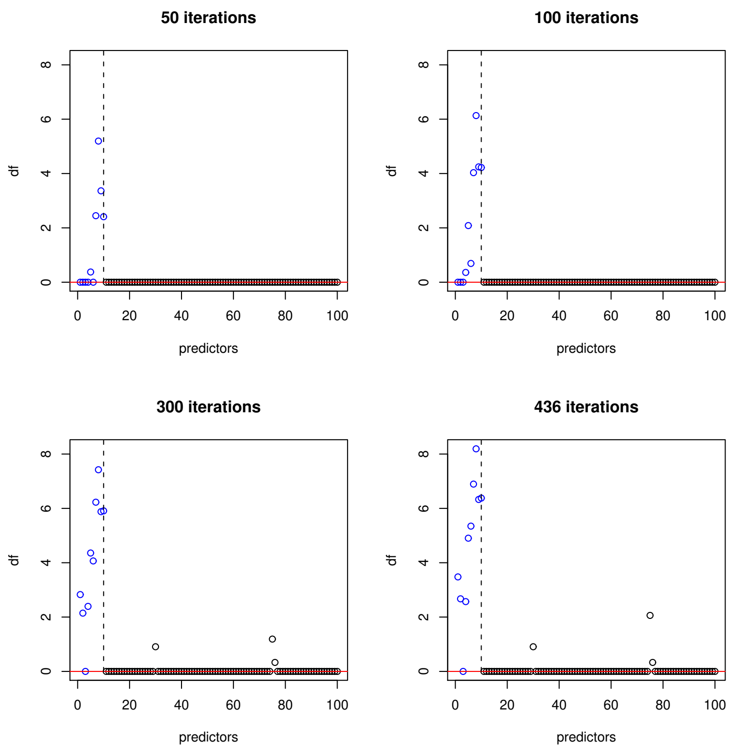
variable selection and variable amount of df for every additive function

12 true and estimated functions:

additive regression function, p-effective = 10, different curve complexities



A very interesting way to search and estimate in high dimensions!



with classical methods for large p:

“unfeasible” to do variable selection and variable amount of df

L_2Boost runs with one (!) tuning parameter and is resistant against overfitting

Behind this was a better version of L_2Boost with some penalization

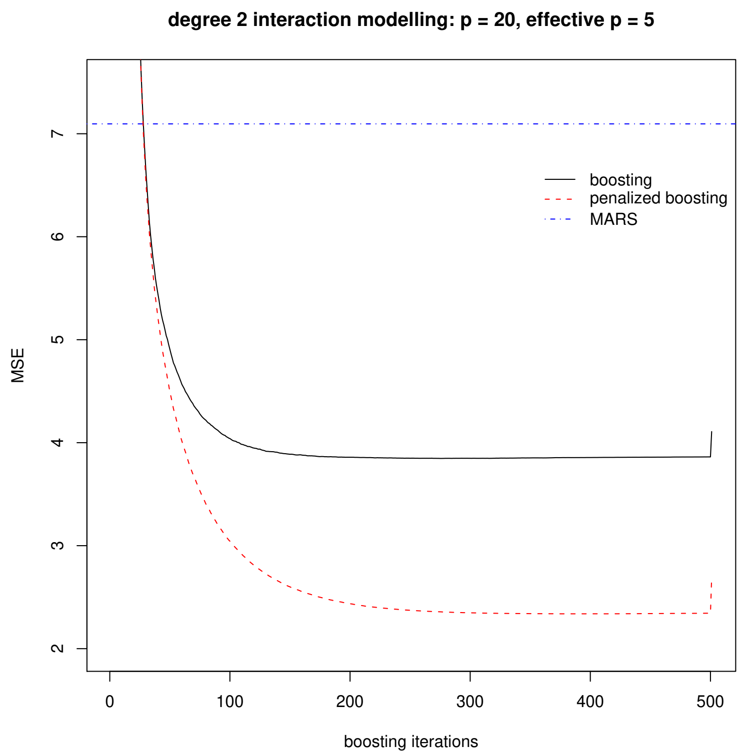
(nothing complicated... and motivated from Hilbert space theory)

Straightforward to extend to modelling with second-order interactions

Friedman #1 model:

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

Sample size $n = 50$



L_2Boost yields a second-order interaction model allowing for interpretation

3. An application to high multivariate GARCH modelling

(Audrino & PB, 2002)

daily asset prices $P_{t,i},\ t = 1, \ldots, n; i = 1, \ldots d$ and their returns

$R_{t,i} = (P_{t,i} - P_{t-1,i})/P_{t-1,i}$ (assumed to be stationary within suitable window)

Notation: $\mathbf{R}_t = (R_{t,1}, \ldots, R_{t,d})'$

aims:

- distribution of $R_{t,i}$ given $\mathcal{F}_{t-1} = \sigma\{\mathbf{R}_s,\ s = t-1, t-2, \ldots\}$
- $\mathrm{Cov}_{d \times d}(\mathbf{R_t} | \mathcal{F}_{\mathbf{t-1}})$

for d of the order 100 – 1'000

basic model: "generalized" CCC-GARCH model

$$\mathbf{R}_t = \Sigma_t \mathbf{Z}_t$$

- (innovations) $\{\mathbf{Z}_t\}_{t\in\mathbb{Z}}$ i.i.d. $\sim \mathcal{N}_d(0, I)$
- (CCC construction) conditional covariance $V_t = \mathrm{Cov}(\mathbf{R}_t|\mathcal{F}_{t-1}) = \Sigma_t\Sigma_t^T$:
  typical element: $v_{t,ij} = \rho_{ij}(v_{t,ii}v_{t,jj})^{1/2}$, where
  $\rho_{ij} =$ Corr$(R_{t,i}, R_{t,j}|\mathcal{F}_{t-1})$ is the constant conditional correlation
- (functional form) conditional variances of the form

$$v_{t,ii} = \sigma_{t,i}^2 = \mathrm{Var}(R_{t,i}|\mathcal{F}_{t-1}) = \underbrace{F_i(\{R_{t-k,j};\ k = 1, 2, \ldots; j = 1, \ldots, d\})}_{\text{nonlinear, potentially depending on all cross-series}}$$

Implementation of FGD with log-likelihood loss function ρ (w.r.t. basic model)

$$-\log f(\underbrace{\mathbf{R}_t}_{\text{multivariate response}} \mid \underbrace{\mathbf{R}_{t-1}, \ldots, \mathbf{R}_{t-p}}_{p \cdot d \text{ predictor variables}})$$

$\rightsquigarrow$ use component-wise steepest functional descent

base learner: regression trees (they do some variable selection)

Results

7-dimensional real daily data ($n = 1000$): returns from US DJIA, French CAC40, German DAX, Italian BCI, Dutch CBS, British FTAS and Japanese NIKKEI index

out-sample negative log-likelihood loss:

FGD with trees: 5547.5 (1.19% gain)

linear CCC-GARCH(1,1): 5614.1

differences among volatility estimates masked by additional common noise term

but: FGD with trees is significantly better than linear CCC-GARCH(1,1)

p-value = 0.003

simulated 100-dimensional time series ($n = 1000$): true model is CCC-type

- 1/4 of the volatility functions are linear GARCH(1,1)-type
- 1/2 of the volatility functions are not depending on a cross-series
- 1/2 of the volatility functions depend only on one other cross-series

$\rightsquigarrow$ simulated model "not too far" from linear CCC-GARCH(1,1)

out-sample negative log-likelihood loss:

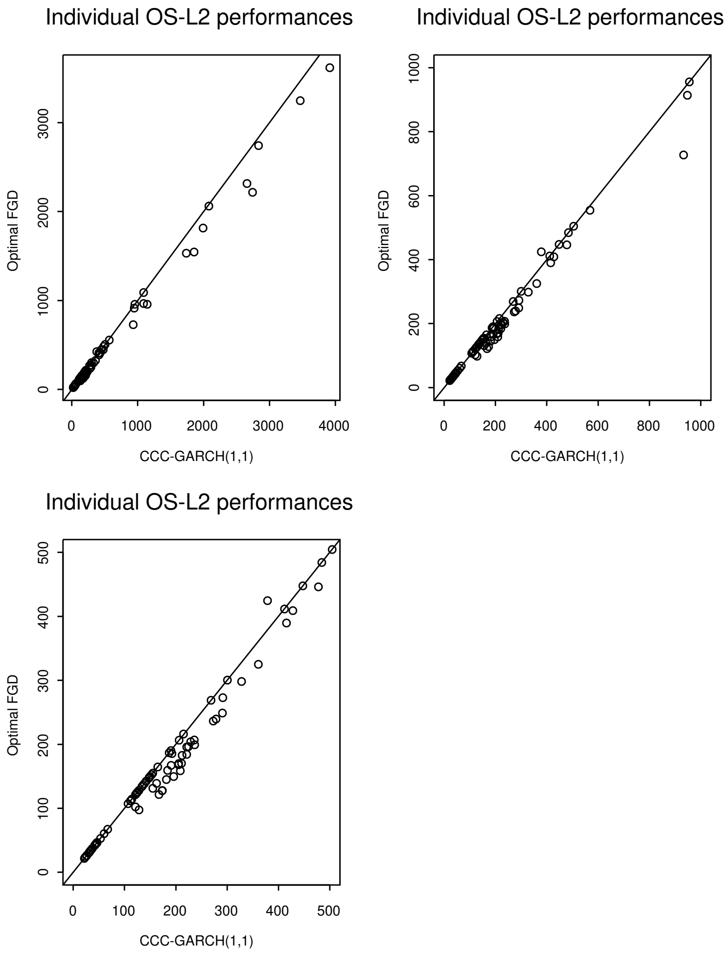
1.54% gain with FGD over linear CCC-GARCH(1,1) (again masked by noise)

individual out-sample L_2 loss $\sum_t |\hat{\sigma}^2_{t,i} - \sigma^2_{t,i}|^2$ for true squared volatility:

on average: 8.4% gain with FGD

maximal: 35% gain with FGD

minimal: in 1 out of 100 series, –10% loss with FGD



most gains for series where linear CCC-GARCH is bad

more gains expected when true model is further away from linear CCC-GARCH

3. Conclusions

Boosting "demystified" as Functional Gradient Descent Method (FGD) (Breiman, 1999)

- FGD can be used within a structured problem
- FGD has "interesting" asymptotic properties:

  consistency, adaptiveness, high-dimensional settings
- FGD is computationally attractive for variable (basis) selection

  and variable degrees of freedom to different variables (basis functions)

moreover: for linear models,

FGD approx. yields all Lasso solutions in one sweep! (Efron et al., 2002)

take home:

FGD ("Boosting") great potential for problems with high-dimensional predictors X